NTNU
Innovation and Creativity

A General Face Recognition System

Achim Sanjay Manikarnika

Master of Science in Physics and Mathematics
Submission date: July 2006
Supervisor: John Sølve Tyssedal, MATH
Co-supervisor: Kjetil Bø, IDI

Norwegian University of Science and Technology
Department of Mathematical Sciences

Problem Description

This project looks into some of the difficulties about face recognition in a general setting and gives an overview over existing methods. A complete algorithm was suggested and in the end implemented in a C program.

Assignment given: 16. February 2006
Supervisor: John Sølve Tyssedal, MATH

Abstract

The recent years of study has brought up a lot of methods for face recognition. A problem with the majority of these methods is that they are either time consuming, or that they require detailed pre-knowledge about the images they are analyzing. This project suggests an algorithm which can detect faces in a general setting and which can operate in real time. The algorithm use a color segmentation in order to locate suitable regions which are then merged together to give a collection of candidate regions. The candidate regions are then analysed using a wavelet strategy. For fast processing the Haar wavelet was used. Features are extracted from the wavelet coefficients by using a set of simple statistical measurements. Candidate regions are accepted or rejected as a face by calculating the Bhattacharya between the feature vectors. For face recognition, the simple wavelet decomposition using the Haar base was also used.

Preface

This project has been written as a fulfillment of the requirement to the study program at the Department of Mathematical Sciences of the Norwegian University of Science and Technology (NTNU) during spring 2006.

I would like to thank my supervisors Professor Ketil Bø and Professor John Sølve Tyssedal for their guidance and feedback during the process.

Trondheim, July 2006

Sanjay Manikarnika

Contents

1. Introduction

Face detection and face recognition from still images and video streams has become an active research area in the last decade and are useful in numerous applications. For example a recognition system can be used to allow access to ATM machines, computers, controlling entries into restricted areas and help searching in face databases. The main area of research has been recognition systems where the individual appear in a systematic and controlled way.

The goal of this project is to develop a system which can be used for face detection and face recognition in a more general setting. The recognition system should be able to operate in real-time and must be able to handle images under different conditions. Images or video sequences have varying background, illumination and faces of different scales. In addition images or video sequences might have more than one face in the scene. Many of the current techniques require a good normalization to give desired results. The lack of a priori knowledge about existence and location of human faces in the processed images makes normalization hard and means that detecting the face is equally important as recognizing it.

Section 2 introduces some terms and notation used throughout this project for describing methods for the face detection and recognition. Some distance measurements are also defined to describe the similarities between faces. Section 3 gives a short overview over techniques used for detecting faces. For example the use of color to find skin like areas can narrow down the search for finding faces. To do this, a color model must be chosen. As a next step in a detection system, color quantization can be used to get areas with more homogenous color and reducing noise in the image. Some of the face detection methods mentioned in section 3 are closely related to face recognition methods which are described more fully in section 4. While section 4 is meant as an overview of current face recognition methods, section 5 gives a strategy for building a complete recognition system. A wavelet implementation was chosen over the more popular methods such as the principal component analysis (PCA) or linear discriminant analysis (LDA). These methods give good results if the images have similar conditions, but are not very robust when it comes to variations in scale, illumination and rotation. While the hidden Markov chains (HMM) and neural networks have shown good results under more general conditions, they have a time consuming training phase and can be hard to customize. Aiming for fast speed and a flexible system the wavelet strategy is a good choice,

and combined with a skin detection algorithm the runtime can be reduced even more. The final recognition system from section 5, has a number of parameters such as distance thresholds which can be altered to change the performance. In section 6 the parameters are tested with different values to try finding the optimal detection and recognition rates. Section 7 summarizes this project, while pointers for further work are given in section 8.

2. Terms and notation

2.1 Image representation

A two dimensional digital image can be represented as function *I(x, y)* where *x* and *y* are discrete coordinate quantities. For simple notational clarity and convenience, integer values are used for these discrete coordinates. The origin (0,0) is located on the top left of an image. The coordinate convention used to represent a digital image in this project is shown in figure 2.1.

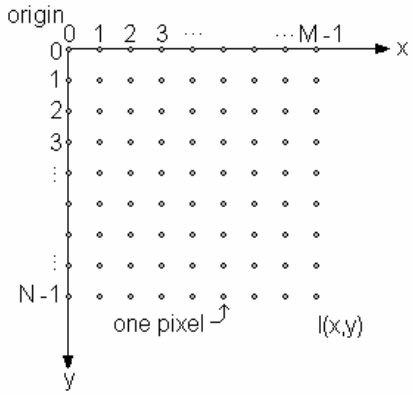

Figure 2.1: Standard coordinates used to represent digital images.

The representation of images can be written as a *N* x *M* matrix on the form

$$I(x,y)=\begin{bmatrix} I(0,0) & I(0,1) & \cdots & I(0,M-1) \\ I(1,0) & I(1,1) & \cdots & I(1,M-1) \\ \vdots & \vdots & & \vdots \\ I(N-1,0) & I(N-1,1) & \cdots & I(N-1,M-1) \end{bmatrix}=\begin{bmatrix} I_{0,0} & I_{0,1} & \cdots & I_{0,M-1} \\ I_{1,0} & I_{1,1} & & I_{1,M-1} \\ \vdots & & & \vdots \\ I_{N-1,0} & I_{N-1,1} & \cdots & I_{N-1,M-1} \end{bmatrix}.$$

Each element of the matrix is called an *image element, picture element, pixel* or *pel*. The terms *image* and *pixel* will be used throughout the rest of this project to denote a digital image and its elements. The pixels are registered by a digital sensor and may encode color or intensity. In this project we use color images, but they can be converted to a greyscale intensity image by taking the average of the red, green and blue color components.

Sometimes it can be more convenient to represent the image as vector or array instead of a matrix. An image can be represented as a *face vector* with p = N*M elements. The vector is constructed by adding each row of *I(x,y)*,

$$c = \left[I_{0,0}, I_{0,1}, .., I_{1,m} \mid I_{1,0}, I_{1,2}, .., I_{1,m} \mid \cdots \mid I_{n,0}, I_{n,1}, .., I_{n,m} \right]^T .$$

Each pixel of the image then corresponds to a coordinate in a *p*-dimensional space often referred to as the *image space*.

A *p* x *n* matrix can then represent a collection of *n* images

$$C = \left[c_1 \quad c_2 \quad \cdots \quad c_n \right],$$

where each column represents an image.

2.2 Similarity matching

Each face recognition system generates at the last step a vector of numerical values called *feature vectors.* The representation and size of these numerical vectors depends on the algorithm, but they are used to describe a face. The recognition is then based on the similarity of feature vectors. If two feature vectors are almost the same they are very likely to be of the same face. Mathematically the closeness of two vectors can be calculated by a distance measurement. Some common distance measurements used by face recognition systems follows.

2.2.1 Euclidean distance

$$\mathrm{d}(\mathrm{x}, \mathrm{y}) = \sqrt{(\mathrm{x} - \mathrm{y})^{\mathrm{T}} (\mathrm{x} - \mathrm{y})},$$

where *x* and *y* are *n*-dimensional vectors.

2.2.2 Bhattacharyya distance

$$d(x,y) = \sum_{i=1}^{n} \frac{2x_i y_i}{x_i + y_i},$$

where x_i and y_i are components of *n*-dimensional vectors. The Bhattacharyya is popular when it comes to pattern recognition because of some if its properties [18], [19] and will be used to calculate distances between feature vectors in this project.

3. Face detection

Face detection is the process of identifying faces in a scene and is usually the first step in a process of recognizing faces. For example if there is no a-priori knowledge about background, position, orientation or illumination the task of extracting regions of interest in the image is nontrivial. The process of detecting faces is closely related to the process of recognizing faces which is presented in section 4. Existing methods can be divided into three broad categories: local facial features detection, template matching and image invariants. Local facial feature detection tries to find key points in the face such as eyes, mouth, noise and chin. Once a possible key point is found it is discarded or accepted by different statistical models of human faces. Image invariant schemes assume that there are certain spatial image relationships common and possibly unique for all faces. Detection is based on learning the underlying rules of a given collection of samples. Neural networks and hidden Markov chains are examples (see section 4) of such methods. The drawback of image invariant methods is that the computation time is usually complex and it can be challenging to find a description of a “common face” which is used to represent the underlying rules.

3.1 Color based face detection

Color allows for fast processing and is also robust against different facial expressions. Skin color has a characteristic look which is easily recognized by humans. Even though color alone cannot be used to determinate if a image region is a face it can narrow down the search in the image. This is crucial in real time applications which depend on the algorithm speed. Color is usually defined as a three dimensional point in a color space. Different color spaces exist and one of them must be chosen before proceeding with the actual skin detection.

3.1.1 The RGB color space

The RGB colorspace originated from various display applications where it was convenient to describe color as a combination of three colored rays (red, green and blue). Today it is widely used when processing digital images. The model is based on Cartesian coordinate system. The

three axes give the intensity of the red, green and blue color and are combined together to give us the perception of color. The RGB colorspace has a high correlation between the channels and is not very intuitive to use. For example similar colors which vary in illumination will change all three components in the RGB colorspace. Converted to grayscale the diagonal from the origin to the far top corner of the box will give the grayscale intensities.

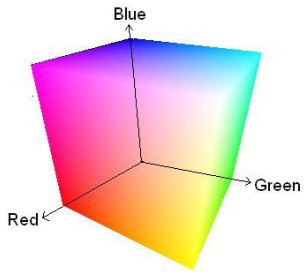

Figure 3.1: An illustration of the RGB colorspace.

3.1.2 The hue-saturation based color models

Hue-saturation based colorspaces are more intuitive and are widely used in applications where artists have to mix and create colors. The component values are based on artist's idea of tint, saturation and tone. Hue defines the dominant color area and saturation describes colorfulness in an area in proportion to its brightness. The last value (intensity, lightness, value) measures the color luminance. Studies in the last decades [1] indicate that the hue component is highly invariant to highlights when using a white light source. Also orientation with matte surfaces had little influence on the hue component with ambient light sources. This makes the colorspace more suited as variations in illumination have little effect on the hue component.

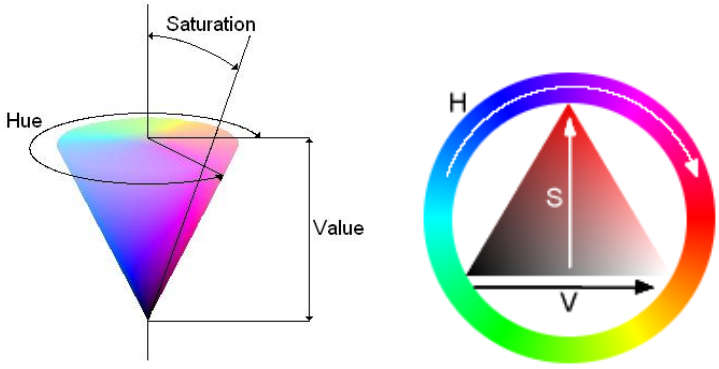

Figure 3.2: An illustration of the HSV colorspace.

Convertion from RGB to HSV coordinates can be done as follows. Given an image in RGB color format, the H component of each RGB pixel is obtained by using the equation

$$H = \begin{cases} \theta & if\ B \le G \\ 360 - \theta & if\ B > G \end{cases}$$

where

$$\theta = \arccos\left\{ \frac{1/2(R-G)+(R-B)}{\left[(R-G)^2 + (R-B)(G-B)\right]^{1/2}} \right\} \quad (1)$$

The saturation component is given by

$$S = 1 - \frac{3}{(R+G+B)} \min(R,G,B) \quad (2)$$

Finally the last component is given by

$$V = \frac{1}{3}(R+G+B) \quad (3)$$

It is assumed that the RGB values in (1-3) have been normalized to the range [0,1]. The output of hue is given in degrees while saturation and value range between [0,1]. More information about conversion from RGB to HSV colorspace can be found in [9].

3.1.3 Color quantization

To enchant the skin detection the original image can be color quantized. Color quantization is a form of vector quantization and will try to bring forth the dominant colors in the image and homogenizing regions of similar color. The basic principle is to map *n* vectors $X = x_1, x_2, ..., x_n$ of dimension R^3 into *k* vectors $Y = y_1, y_2, ..., y_k$ of dimension R^3 where $k < n$. Quantization can also be used in image compression since it reduces the number of colors in a image. It is a lossy transform which means the original image cannot be reconstructed after the quantization is complete. The set of vectors *Y* is called a *code book* and y_i a *code word*. Ideally the code words are the dominant colors of the original image. In color quantization a code word is usual the mean or median color of some color cluster in the original image. Even though there exists a lot of different methods, most of them can be categorized into three phases. The first phase is a sampling phase from the original image which usually consists of constructing color histograms. The second phase called code book or palette design and determine how to choose

the representative colors in *Y*. The third phase consists of mapping each color of the original image to the colors in the code book. The resulting image from the quantization will have less color and should try to minimize the error between the color values from the original image and the quantized image. More about vector quantizer design can be found in [21] and [27].

The key element of a skin color detection algorithm is to decide if a given pixel is from a human skin or not. This decision rule is usually done by measuring a distance of the given pixel to skin tone. Explicitly defined skin regions can be done by defining a number of constraints the color of the pixel has to fulfill. Such predefined classifiers work very fast, but good constraints are crucial to get the wanted results. These are in most cases found empirically.

3.2 Detection concepts

Most of the face recognition methods can be used to detect faces. This is done by creating a feature vector which is supposed to represent faces in general. Then the recognition algorithm is used to scan areas of the image and calculate the distance between the image areas and the face feature vector, see figure 3.3. The face feature vector is typically the average of a collection of images where the face regions have been extracted manually. Section 4 describes in more detail the different face recognition algorithms and how their feature extraction is preformed.

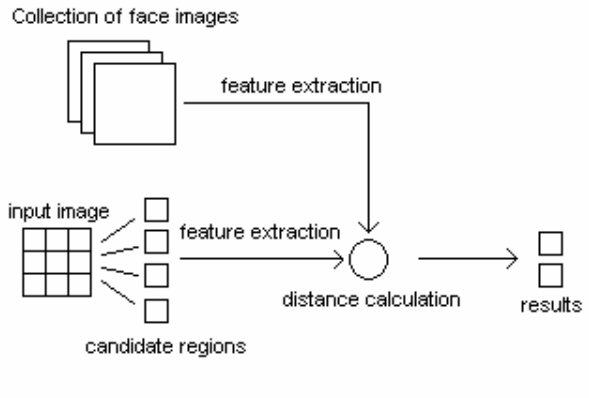

Figure 3.3: A general detection scheme.

4. Face Recognition

This section describes current face recognition algorithm and is meant to give an overview over existing methods. Only section 4.4 about wavelets will be used later in this project.

4.1 Feature based recognition

Geometric feature based matching use basic geometrical features for recognition of a face. This are typically features like the height and the width of eyes, eyebrows, nose, mouth, ears and the face outline. The measuring techniques can differ in complexity, but they all try to reduce the influence of light conditions and changes in facial expression. These methods will have to find the regions in the image containing the features at interest. Integral projections are useful techniques for this purpose and are defined in the following way.
Let $I(x, y)$ be the whole image. The vertical integral projection for an image region bounded by the $[x_1, x_2] \times [y_1, y_2]$ rectangle is

$$V(x) = \sum_{y=y_1}^{y_2} I(x, y), \qquad x_1 < x < x_2 .$$

Similarly the horizontal image projection will be

$$H(y) = \sum_{x=x_1}^{x_2} I(x, y), \qquad y_1 < y < y_2 .$$

Combined with edge detection algorithms they can be effective in determining the positing of features. However they require the rectangle in which they operate to be suitably located. They are therefore mainly used after a normalization process. Information about the geometrical features can be stored as numerical values in a feature vector.

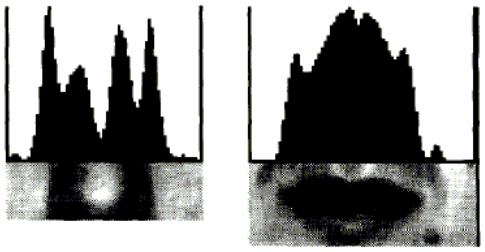

Figure 4.1: A illustration of a vertical integral projection.
Original image is from [6].

The recognition process can then be done by calculating the difference between two vectors representing the faces using some kind of metric. The face with the smallest distance will be the recognized face.

A Bayesian approach was done by [31]. Each feature vector is normally distributed with mean representing the given feature and the covariance matrix Σ. The covariance was assumed to be common for each face so an estimate for the matrix is the average for all the available images in the training set $\Sigma = \frac{1}{N}\sum_{i=1}^{N}\Sigma_i$ where Σ_i is the estimated covariance matrix for face *i*.

Using the metric

$$d(x, y) = (x - y)^T \Sigma^{-1} (x - y).$$

If a feature vector *x* is to be matched with the database we will have to minimize the distance between *x* and all the feature vectors y_i in the database. This is often referred to as finding the nearest neighbour to *x* and will maximize the probability of y_i being the observed face.

4.2 Template matching

A template matching strategy uses whole facial regions or pixel areas for comparison with the stored images of known individual in the database. The earliest techniques used the grey-scale intensity values for the recognition process directly, but more sophisticated methods have arisen in the past decades. These are methods which involve pre-processing and transformation of the extracted grey-level intensities. The principal component analysis or eigenfaces approach is an example of this and will be discussed in Section 4.3. Recognition is

based upon calculating scores for each region by comparing with all the images in the database. The unknown face will be the face with the highest cumulative score.

Let *C* be the set of all the images we want to recognise $C = \{ c_i : i = 0,..., n \}$. While some methods use the pixel data directly, others have some transformation done to reduce the dimensionality. Let X be the set of mapped images onto the chosen coordinates $X = \{ x_i = f(c_i) : i = 0,..., n \}$ where $f(\cdot)$ is a mapping function. For example when using the raw data directly $f(\cdot)$ is the identity function. In the eigenfaces method discussed in Section 4.3 $f(\cdot)$ maps each image c_i onto the corresponding eigenfaces. The process of applying $f(\cdot)$ to the set C is often referred to as training the database. This process can often be time consuming and can be a problem when *n* becomes large. The recognition process can be formulated as follows. Let y be an input image. The image y can either be of an individual in the set C or it is an unknown individual. Ideally we should find the right image c_k if $y \in C$ or reject y if $y \notin C$. We start by transforming the input image by applying the mapping function $y' = f(y)$. We can now compute a measure of distance between y' and all the images in the set *X* with some metric $d(x, y)$. Let x^* be the closest match to y' i.e.

$$d(x^*, y') \le d(x_i, y') \qquad \text{for } i = 0,1,\ldots,N.$$

In this classical approach we choose the nearest neighbour of the input image *y*. A problem can arise when $y \notin C$. The obvious solution is to introduce a threshold value *T* to decide if *y* should be rejected or identified as image x^*

$$result \longleftarrow \begin{cases} x^* & if \quad d(x^*, y') \le T \\ otherwise, & unknown \end{cases}$$

More complex methods can be used, by for example using individual threshold values for each image c_i i = 0,1,…, N.

This was done by [23] where they operated with two threshold values. A lower threshold L_i and upper threshold U_i was used. The lower threshold is used to measure a sufficient match for y' representing x^*. The upper threshold is used when there is a sufficient mismatch for y' representing x^*. In the cases where the values of $d(x^*, y)$ lies between the lower and the upper

threshold a heuristic method was used. In [23] they used databases with multiple images of the same individual taken in a laboratory setting to minimize any pre-processing. The heuristic method will accept *x** if the second best match is another image of the same person. If not *x** will be rejected. The multiple set of images for each person was also used when creating the threshold values.

$$result \longleftarrow \begin{cases} x^* \quad if \quad d(x^*, y') \le L_i \\ unknown \quad if \quad d(x^*, y') > U_i \\ use \quad heuristic \quad if \quad L_i < d(x^*, y') \le U_i \end{cases}$$

Using the Euclidian metric a correct classification rate of about 99% was achieved in a database with 600 individuals. However each individual had 8 training images which were taken in a very systematic manner and all with a frontal view.

4.3 Linear subspace methods

Often when using high-detailed data, like high resolution images or large image databases leads to problems concerning high dimensionality. This is typically a problem with template matching algorithms. Linear subspace methods try to approximate face vectors into vectors with lower dimension. Then recognition of the faces can be performed in this reduced space. These approaches use a training phase where the face database and projection matrix is created. During the training, the mean face of the image database is also calculated and subtracted from each face vector. The projection matrix projects a high dimensional face vector into a feature vector with lower dimension.

$$s = W^T x \tag{4}$$

where x is a *n*-dimensional is a face vector, *W* is a *n* x *m* projection matrix and *y* is a *m* -dimensional feature vector. Dimension is reduced by having $m < n$. The feature vectors in R^m are then compared by some similarity measure.

4.3.1 Principal component analysis (PCA)

The principal component analysis (PCA) also known as Karhunen-Loeve transformation or eigenfaces projection is a widely known technique for addressing high dimensionality problems. The method reduces the dimensionality of the problem by estimating a linear orthogonal subspace that spans the samples. A new image will be mapped onto this orthogonal subspace and then compared with the image sample.

The orthogonal subspace used to map the images consists of the eigenvectors of the covariance matrix of the image data. The eigenvectors can be thought of as a set of features which together characterize the variation between face images. Each face image in the training set can be represented exactly in terms of a linear combination of the eigenfaces. The faces will be approximated using only the "best" eigenfaces—those that have the largest eigenvalues, and which therefore account for the most variance within the set of face images.

Given a set of image samples $C = \{ c_i : i = 1..n \}$ where each image c_i has dimension p, the solution is obtained by solving the eigensystem of the covariance matrix in the original image space

$$\Sigma = E\{(c - \bar{c})(c - \bar{c})^T\},$$

where *c* is a *p* x *n* matrix containing the normalized image vectors, and $\bar{c}$ is the mean face image. The axes of the subspace are given by the eigenvectors while the corresponding eigenvalue magnitude tells us the variance along the axis. Choosing only the axes with the largest variance will reduce dimensionality. This is done by sorting the eigenvalues and then pick the *m* largest. However for implementation, the covariance matrix Σ will in most cases become too large to work with. Instead we use the eigenvalues of the *n* x *n* covariance matrix

$$\hat{\Sigma} = X^T X$$

where $X = [(c_1 - \bar{c}) \quad (c_1 - \bar{c}) \quad \cdots \quad (c_1 - \bar{c})]$. This way the number of eigenfaces must be less or equal to the number of images in the training set. The original Σ has rank less or equal to *n*

depending on the linear dependence of the vectors in the training set. In addition it can be shown that the eigensystem of $\hat{\Sigma}$ has the same non-zero eigenvalues of Σ. Using the singular decomposition $X = USV^T$ where U and V are orthogonal matrices gives

$$\begin{aligned} X^T X &= VSU^T \cdot USV^T = VS^2V^T \\ X^T XV &= VS^2 \end{aligned},$$

and

$$\begin{aligned} XX^T &= USV^T \cdot VSU^T = US^2U^T \\ XX^T V &= US^2 \end{aligned}.$$

This shows that both systems have the same eigenvalues, which comes from the diagonal elements of the S matrix.

The optimal W is then constructed by letting the columns in W be the eigenvectors of $\hat{\Sigma}$. Let $\hat{\Sigma}$ have the eigenvalue-eigenvector pairs $(\lambda_1, e_1), (\lambda_2, e_2), ..., (\lambda_n, e_n)$ where $\lambda_1 \geq \lambda_2 \geq \lambda_n \geq 0$, then

$$W_{opt} = [e_1 \mid e_2 \mid ... \mid e_n] \tag{5}$$

Now every image c_i can be projected to the eigenspace space by

$$s_i = W^T (c_i - \bar{c}) \tag{6}$$

and reconstructed by

$$c_i = Ws_i + \bar{c}\ .$$

Further reduction in dimensionality is achieved by discarding the least significant eigenvectors in W. These are the eigenvectors paired with the smallest eigenvalues. After the eigenvectors with the least significant variance information is removed the projection matrix is of dimension *p* x *k*,

$$W_{opt} = [e_1 \mid e_2 \mid ... \mid e_k],$$

and projects a data point c_i onto a linear orthogonal subspace in such a way that the reconstruction loss is minimized. The mean squared error associated with projecting and then reconstructing an image can be shown to follow the expression

$$\sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{k} \lambda_i = \sum_{i=k+1}^{n} \lambda_i .$$

This is useful for giving an indication on how many eigenvalue-eigenvector pairs to discard when reducing the dimensionality.

The face recognition process is done by projecting the unknown image *y* and all the sample images c_i, $i = 1,2,\ldots,n$ into the eigenspace by using (eq 6). A distance measure is then calculated between the *y* and the rest of the trained images s_i. See Section 2.2 and 4.2 for more details about the distance measure and how to decide if the image is one of the trained or an unknown image.

4.3.2 Linear Discriminant Analysis (LDA)

Whereas the Principal Component Analysis (PCA) tries to find a subspace whose basis vectors correspond to the maximum variance direction in the original image space, the Linear Discriminant Analysis (LDA) tries to find the basis vectors that best discriminate among classes. Suppose that we have a set $M = \{m_1, m_2, ..., m_n\}$ of *n* classes and for each class we know the images x_i and their identity. This class information can be used to reveal the structure of the data. The projection matrix *W* is set in such way that the ratio between between-class scatter and the within-class scatter is maximized. Defining the between-class scatter matrix as

$$\Sigma_B = (\mu_i - \overline{\mu})(\mu_i - \overline{\mu})^T \text{, where } \overline{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mu_i$$

and $\mu_j = \sum_{i \in m_j} x_i$ the mean image for the images in class *j*. The within-class scatter matrix describes the variation between the images in one class. The within-class scatter matrix is

$$\Sigma_W = \sum_{j=1}^{n} \sum_{i \in m_j} (x_i - \overline{\mu}_j)(x_i - \overline{\mu}_j)^T$$

and describes the variation between the classes. A ratio between two matrices is maximized by maximizing the ratio of their determinants.

$$W_{opt} = \arg\max_W \left(\frac{\det \left| W^T \Sigma_B W \right|}{\det \left| W^T \Sigma_W W \right|} \right)$$

See [4] or [17] for more detailed theory on this. This ratio is maximized when W_{opt} is the eigenvectors of $\Sigma_W^{-1} \Sigma_B$. If there is only one image per class case, the within-class matrix Σ_W will be the identity matrix and W_{opt} becomes the same expression as (5) in the PCA. From the general eigenvalue problem, $\Sigma_B W = \Sigma_W W \lambda$ only *m-l* or less eigenvalues can be nonzero depending on how many of the columns in Σ_W are linearly independent. To prevent a non-singular within scatter matrix the dimensionality can be reduced by first applying the PCA. The reduction should be so that all the columns are linearly independent so it ensures the existence of Σ_W^{-1}.

4.3.3 Independent Component Analysis (ICA)

The goal of ICA is to find a linear transformation to express a set of random variables as linear combinations of statistically independent source variables. The search makes use of higher order statistics and thus provides a more powerful data representation than PCA. The PCA can only separate pair wise linear dependencies between pixels so higher order dependencies will not be properly separated.

ICA of a random vector searches for a linear transformation which minimizes the statistical dependencies between each component in the vector. Let x be a random vector representing

an image. The ICA of x factorises the covariance matrix $\sum = E\{(x - E(x))(x - E(x))^T\}$ into the form

$$\sum = W \Delta W^T,$$

where Δ is a real valued positive diagonal matrix and *W* transforms x into s from (eq 4) in such a way that independency of the components of x are maximized. Unlike the PCA and LDA there is no closed expression for the projection matrix *W*. Instead there exist iterative algorithms that are based upon different search criteria. The ICA is closely related to the *blind source separation problem*: decomposition of the input signal (image) *x* into a linear combination of independent source signals. For more information about ICA algorithms like the InfoMax and FastICA please refer to [7] and [16].

4.4 Wavelets

Wavelets have arisen as the most recent solution to overcome the shortcomings of the Fourier transform in certain applications. One of the weaknesses of Fourier theory is that it has only frequency resolution and no time or space resolution. This means that if a signal is transformed it can no longer be known when or where the frequencies in a signal are although we know all the frequencies present.

The wavelet is based upon the idea to cut the signal into several parts and analyze the parts separately. This gives more information about the when and where of different frequency components, but a fundamental problem is how to cut. However it turns out from Heisenberg's uncertainty principle that a signal cannot be represented as a point in the time-frequency space. This shows that it is very important how one cuts the signal. Wavelets use a fully scalable window for cutting a signal. The window is shifted along the image and for every position the spectrum is calculated. The process is repeated many times with slightly modified window (shorter or longer) for each repetition. We now have a collection of time-frequency representations of the image, all with different resolutions. The wavelets are functions defined over a finite interval and having an average zero value. The basic idea is to represent any arbitrary function as a superposition of a set of basis functions called baby wavelets which are

all obtained from a single base called the mother wavelet. The baby wavelets are created by dilations or scaling and translations of the mother wavelet.

4.4.1 Fast Wavelet transform

The implementation of the *fast wavelet transform* (FTW) is a computationally efficient method that resembles a two band sub band coding scheme. (See Digital Image Processing). Evaluating convolutions at nonnegative, even indices is equivalent to applying a filter and then downsampling by 2. Downsampling is the process of reducing the sampling rate of a signal and will then give us the signal of different scales. Figure 4.2 shows the structure for computing wavelet coefficients at two or more successive scales.

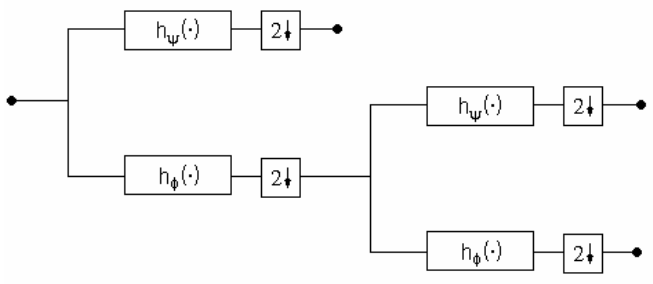

Figure 4.2: A two-scale FWT analysis bank structure. The input is a 1 dimensional signal which is split into a low pass component and high pass component.

The first filter bank splits the original signal function into a lowpass, approximation component given by the h_ψ coefficients and a highpass, detailed components given by h_ϕ. The two-stage filter bank shown in figure 4.6.1 can be extended to any number of scales.

For the use of wavelets in image processing it has to handle two dimensions. The extension to the two-dimensional case needs a two dimensional scaling function $h_\phi(x,y)$ and three two dimensional wavelets $h_\psi^H(x,y)$, $h_\psi^V(x,y)$ and $h_\psi^D(x,y)$. The functions are separable functions and are the results after applying the scaling function in one dimension (along the columns) and then applying a one dimensional wavelet in along the other dimension (along the rows).

$$h_\phi(x,y) = h_\phi(x)h_\phi(y)$$

where the functions

$$h_{\psi}^{H}(x,y) = h_{\psi}(x)h_{\phi}(y)$$

$$h_{\psi}^{V}(x,y) = h_{\phi}(x)h_{\psi}(y)$$

$$h_{\psi}^{D}(x,y) = h_{\psi}(x)h_{\psi}(y)$$

are directional in the sense of measuring functional variations-intensity or gray-level for images along different directions. The h_{ψ}^{H} measure variations along columns, h_{ψ}^{V} measure variations along rows and h_{ψ}^{D} corresponds to variations along diagonals. The digital filters and downsamplers can be applied just as in the one dimensional case by first taking the one dimensional FWT on the rows followed by the FWT on the resulting columns. The two-dimensional fast wavelet transform filter bank structure is shown in figure 4.3.

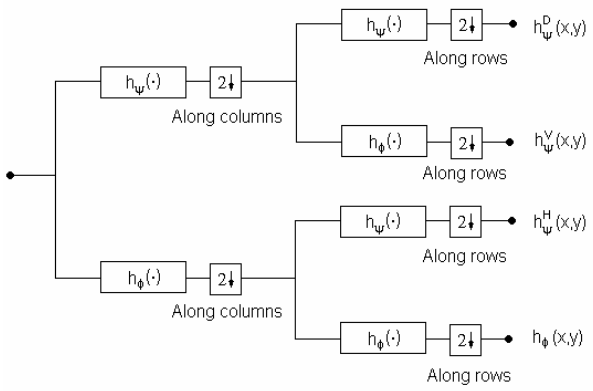

Figure 4.3: A one-scale FWT analysis bank structure. The input is a 2 dimensional image which is split into four sub-images. To increase the number of scales, the filters are applied iterative to the output $h_{\phi}(x,y)$.

The filter can be extended to any number of scales by applying the filterbanks iterative to the output of the approximation coefficients $h_{\phi}(x,y)$. To get a richer analysis one can perform the filterbanks on each of the outputs $h_{\psi}^{D}(x,y)$, $h_{\psi}^{H}(x,y)$ and $h_{\psi}^{V}(x,y)$. This will result in a tree shown in the right of figure 4.4, while the regular decomposition is in the left of figure 4.4.

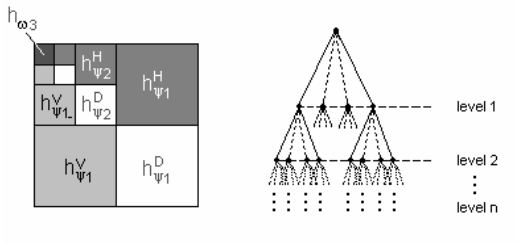

Figure 4.4: The structure of the wavelet decomposition and the wavelet package tree. The left figure shows a 3 scale regular decomposition. The right figure shows the tree structure obtained after *n* levels with the wavelet package analysis.

4.4.2 Haar wavelet

The Haar wavelets are the simplest wavelets given by filters

$$f = \frac{1}{\sqrt{2}}[1 \quad 1]$$

$$g = \frac{1}{\sqrt{2}}[-1 \quad 1].$$

The Haar wavelet decomposes the signal by replacing adjacent pair of gray level or intensities in discrete intervals with average and difference of the pairs. Here detailed coefficients are differences of two pairs while average of two pairs gives the approximation coefficients. In the above equations $\sqrt{2}$ is used instead of 2 to preserve Euclidean distance. Applying the Haar wavelet on a two dimensional image, each adjacent pair of data in a discrete interval is replaced with its average and difference. Figure 4.6.3 shows how the detailed values h_ψ^H, h_ψ^V and h_ψ^D correspond to the summation of rows, summation of columns and summation of diagonals respectively. The nodes A, B, C and D correspond to pixels and hold the average or intensity value of the RGB or HSV color.

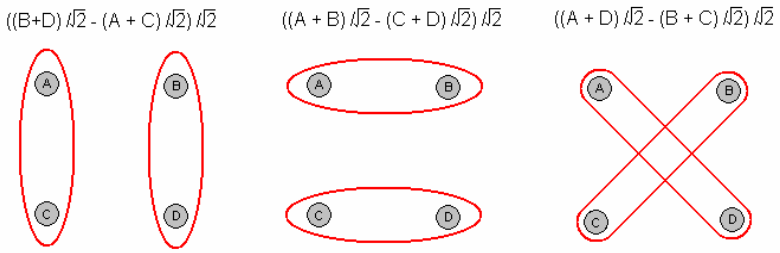

Figure 4.5: Shows how to compute the detailed coefficients h^V, h^D and h^H.

4.4.3 Gabor wavelet

Gabor wavelets are often used for image analysis because of their computational properties. They are optimally localized in the space and frequency domains and have therefore an advantage over the Fourier transform which is only localized in frequency.
The Gabor wavelet kernel is defined as:

$$\psi_{\mu,\nu}(z) = \frac{{k_{\mu,\nu}}^2}{\sigma^2}\exp\left\{-\frac{{k_{\mu,\nu}}^2 z^2}{2\sigma^2}\right\}\left[\exp\left(ik_{\mu,\nu}z\right)-\exp\left(-\frac{\sigma^2}{2}\right)\right]$$

The *k* determines the oscillation frequency and the direction of the wavelet, while σ controls how fast the wavelet collapses to zero as one move away from its center. For image applications the Gabor wavelet is used by creating non overlapping regions. These can be of any shape, but normally a rectangular grid is preferred due to simplicity. At the junction of the regions called grid points the image is decomposed into a set of wavelets with different direction and frequency. Typically *k* is limited to a few values at each grid point while σ is held constant. The set of wavelets at one grid point is often referred to a as a feature vector or a *jet* and the set of jets are used to describe an image. Depending on the number of wavelets per jet and the number of grid points used the method reduces the dimensionality and is an effective way of characterizing images. More about Gabor wavelets can be found in [20] and [33].

Recognition is done by comparing the jets of the unknown image *y* with the jets of the known images in the gallery. It is known that the Gabor wavelets have a problem describing edges in images. The real and imaginary parts of $J_{\mu,\nu}$ oscillate instead of providing a smooth peak. To overcome this problem the absolute value $\left|J_{\mu,\nu}\right|$ is used the components in the jets. Two jets can then be compared with some measure of distance $d(J_i, J_j)$. The normed dot product

$$d_{nd}(J_i, J_k) = \frac{J_i \cdot J_k}{\|J_i\| \, \|J_k\|} \tag{7}$$

is a common function for measuring similarity because it is robust with respect to varying illumination and it is also invariant to the jet length

4.4.4 Elastic grid matching

Elastic grid matching is a technique to improve the positions of the grid points where the jests are calculated. Moving the jets positions can improve the systems ability to handle different facial expressions and also improve recognition where the face orientation has changed. Having a fixed grid will only be effective if the images are similar configured. This requires a very careful preprocessing and might be difficult in many applications. Lades and Malsburg were the first to use an elastic grid where the points could be distorted within some constraint to make a best match between images. The construction of the grid is controlled by a cost function which favors similarity of jets attached to corresponding grid points and penalizes grid deformation. Say we want to compare an unknown image $I(y)$ against the sample images $I(x_i)$, *i* = 0,1,2,… *n*. The *n* images have been stored with a fixed grid and jets calculated at each grid point. For each image $I(x_i)$, i = 0,1,…,n the grid points in *I(y)* will deform so that they minimize a cost function. Consider the edges between grid point of the *i*'th image x_i^k and x_i^p using the distance measure

$$\Delta x_{kp}^i = x_i^k - x_i^p, \qquad (k, p) \in E,$$

where *E* is the set of edges in the image. The distance between the grid points in image x_i will be compared with the grid points in image *y* by the function

$$d_2(\Delta x_{kp}^i, \Delta y_{kp}) = (\Delta x_{kp}^i - \Delta y_{kp})^2 \tag{8}$$

Now that we have a measure of the similarity of two jets (eq 7) and a distance measure between grid points of two images (eq 8) we can define the cost function between *I(y)* and image $I(x_i)$.

$$C^i_{total} = \lambda C^i_{grid} + C^i_{jets}$$

$$C^i_{total} = \lambda \sum_{k,p \in E} d_2(\Delta x^i_{kp}, \Delta y_{kp}) - \sum_{k \in V} d_{nd}(J_k(x_i), J_k(y)) \quad (9)$$

The grid points for the stored images *I(x)* are fixed while the grid points of the unknown image *I(y)* can be moved and will then also change the jets of since they depend on the positions of the grid points. The normed dot product will be close to one if the jets compared are paralell and close to zero if the jets are perpendicular to each other. From (eq 9) we see that two similar jets will decrease the cost function whiles the opposite is true for two different jets.

The goal is to find the positions of the grid nodes that minimize the total cost. This is a hard optimization problem and one has to apply some heuristic approach to solve the problem. This must be done for every image in the gallery. Some method must then be used to tell if the unknown image *y* is sufficient close to the image with the smallest cost. The method is computational complex and limits the use of this technique on regular computers.

4.5 Neural networks

Neural networks aroused when researchers were trying to understand the brains way of processing information. A neural network may perform various tasks such as classification, estimation, simulation and prediction of some underlying process generating some observed data Y. The neural network consists of nodes (names like neurons, processing elements or units are also commonly used) connected together to form a network of nodes. A neural network needs to be adapted by some algorithm to produce the desired results. These algorithms are designed to alter the strength (weights) of the connections between the nodes such that the final output is the desired signal.

These models can be thought of defining a function $f : X \to Y$ where the function $f(\cdot)$ is a composition of other function $g_i(x)$, which again are compositions of other functions. The networks are represented as a graph showing each node and their directed connections. The input at each node is often processed by the nonlinear weighted sum, where the output is given by $f(x) = K(\sum w_i g_i(x))$ where $K(\cdot)$ is some predefined function. Figure 4.6 shows a dependency graph of a neural network.

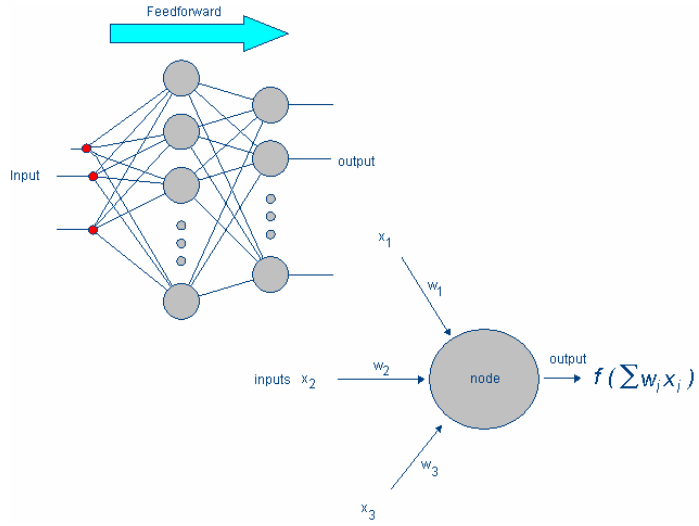

Figure 4.6: The directed graph of a feed forward neural network.

The neural network shown at figure 4.6 is called a feed forward network because their graph is a directed acyclic graph, i.e. the signals can only flow one way in the network. More complex neural networks may have cycles and are then called recurrent.

Learning or training the neural network is a task of tuning the weights w_i so that the function $f(\cdot)$ who gives the output is optimal. This is done by defining a cost function *C*. Learning algorithms search through the solution space in order to find the function $f'(\cdot)$ who has the smallest possible cost

$$C(f') \leq C(f) \qquad \text{for all } f \text{ in the solution space.}$$

The cost function must be a function of the observed data *Y*.

A cost function can be the expected error between the squared observed data and the squared output generated by the neural network $C = E\left\{ f(X)^2 - Y^2 \right\}$.

For the use of classification the neural networks usually falls under the learning class called *supervised learning*. Here we have pairs of samples $(x, y)\ \ x \in X, y \in Y$ of inputs and we know the wanted output. The goal of the training or learning algorithm will be to find the weights that matche together the samples (x, y). The cost function depends on the mismatch between the given output *f(x)* and the correct observation *y* taken over all available pairs. *Multi-Layed Perceptrons* is a class of neural networks which uses supervised learning. The cost function is the mean-squared error, and to find the minimum one uses the steepest descend or gradient descend method. Basically this is a line search method where the search direction is the negative gradient $-\nabla C$. With iterations a new position can be found by $x_{k+1} = x_k - \alpha \nabla C(x_k)$ where α is some step length so that $x_1 \geq x_2 \geq x_k \geq x_{k+1}$. More on this subject can be found in [15]. For neural networks this training algorithm with the specific cost function is called the *backpropagation algorithm*. Unfortunately a signal has to travel through the network in order to calculate $-\nabla C$ which can be time consuming. Then the correction of the weights starts at the end of the network travelling backwards minimizing the cost function at each node. The process is then repeated until the changes in the error *C* is sufficient small.

4.6 Hidden Markov Models (HMM)

Hidden Markov Models have been used in numerous areas for recognition. They have been successfully implemented for speech recognition, lip reading or video indexing. They work well when the data is one dimensional varying over time. The recent years they have also been used on two dimensional problems like character recognition, word spotting and face recognition. The strength of HMM is that they are robust against invariance in scale and they make use of the basic structure of a face.

A hidden Markov model consists of a Markov chain with finite states. After the initialization the states can not be observed directly. However each state generates observations which are

drawn from some probability distribution. The HMM requires a state transition probability matrix and an initial state probability distribution. The states are represented in the set $S = \{s_1, s_2, ..., s_N\}$ where the state at time *t* is denoted by $q_t \in S$. The initial probability distribution $\prod = \{\pi_1, \pi_2, ..., \pi_n\}$ tells us how the chain starts $\pi_i = P(q_0 = s_i)$ for i = 1,2,…,n. The state transition matrix gives information on how the state can go from one given state to another.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{n1} & & & a_{nn} \end{bmatrix},$$

where $a_{ij} = P(q_t = s_i \mid q_{t-1} = s_j)$ is the probability of transition between state s_i and s_j. At each time step the state has to be defined (even if it remains the same) so we have the constraint $\sum_{j=1}^{n} a_{ij} = 1$ for i = 1,2,…,n. This makes sure chain is somewhere at a given time. If the states could be observed directly this would be a Markov chain, but in the hidden Markov chain we observe from a probability function defined at each state $B = \{b_1(o_t), b_2(o_t), ..., b_n(o_t)\}$. The function $b_j(o_t) = P(o_t \mid q_t = s_j)$ gives the probability of observation o_t at time *t* given the state $q_t = s_i$. The probability function can be continuous or discrete. In the discrete case $b_j(o_t) = P(o_t = c_k \mid q_t = s_j)$ where $C = \{c_1, c_2, ..., c_m\}$ is the set of all possible observations, often referred to as the codebook of the model. A HMM is defined by the parameters A, B and $\prod$.

For face recognition the HMM makes use of significant facial regions like hair, forehead, eyes and mouth. Since these regions appear in a natural order from top to bottom in a face, each region can be assigned to a state in a 1d HMM as illustrated in figure 4.7

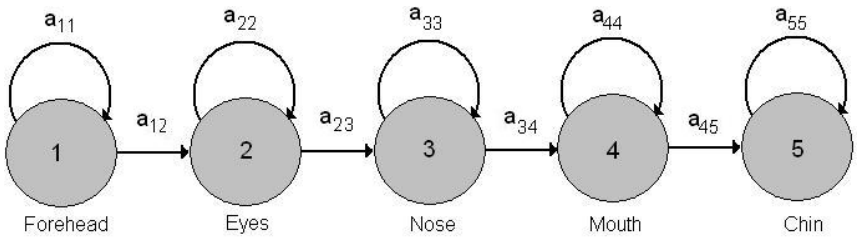

Figure 4.7: A five state one dimensional HMM which can be used for face recognition.

The observations o_t are usually the pixel values in a small rectangular window stretching all across the face image, but with a limited height. The height should be specified in order to try capturing features as good as possible. For the next observation o_{t+1} the rectangular window moves vertically down some distance but it should overlap with some amount *p* with the last observation o_t. This prevents a disjoint partitioning of the image which could result in truncation of features occurring at the rectangle boundaries. See figure 4.8.

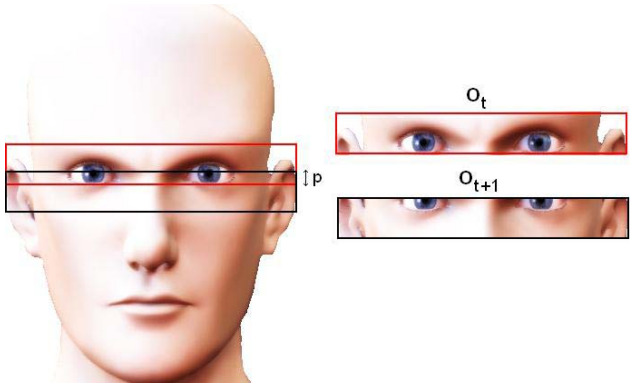

Figure 4.8: Two observation vectors for the HMM. Each step moves the rectangle vertically down overlapping the previous rectangle with some amount *p*.

Instead of using the pixel values inside the rectangle one can use different transformation instead like the PCA (see section 4.3). The disadvantage of using the pixel values directly is that they are sensitive to noise, rotations, shifts and changes in illumination. Also the dimensionality of the observation vector is generally larger when using all the pixels which may lead to more complex calculations.

The training process consists of representing each face in the database by a HMM model. Preferably multiple images with different expression and illumination for each person is used to

train each HMM. After extracting the observation vectors (Figure 4.8) the input face is split up into parts representing the states of the HMM. The observation vectors associated with each state is then used to obtain an initial estimate of the observation probability B. The initial values for the transition matrix A should be values that are consistent with the states defined. If using the example in figure 4.7 we got $a_{ij} = 0$ for $i > j$ and $i + 1 < j$ which only allows transition forward or to the same state. Since the process should start in state 1 the initial probability distribution should be $\pi_1 = 1$, $\pi_i = 0$ for *i* > 1. After the initialization the parameters are updated using the Baum-Welch algorithm [3] which maximizes the probability of observing the training images with the given models.

Recognition is done by adapting a HMM for the input image. Then the model with its parameters is checked up against the models acquired during the image training. Some distance measure is then used to pick the image from the gallery whose HMM model is most similar to the HMM of the input image.

More advanced models can be created by using 2-dimensional HMM. However the complexity increase makes it inefficient in practical applications (see Figure 4.9).

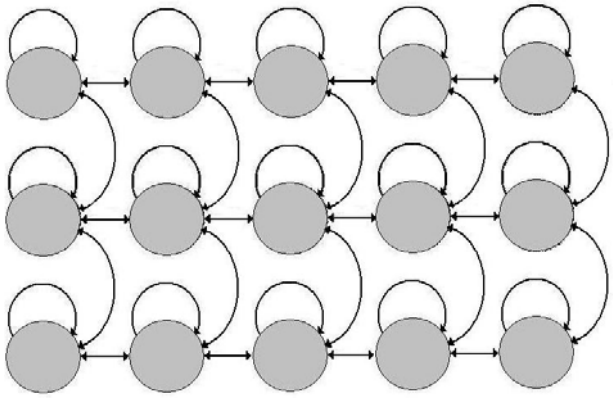

Figure 4.9: A general two dimensional HMM.

Because of this the less complex Embedded Hidden Markov Chain is often used. The embedded HMM is basically a one dimensional HMM where each state (*super states*) is in it self a one dimensional HMM (*embedded states*). The super states is usually the regions from figure 4.7 defining the direction vertically on the image, while the embedded states describe the horizontal direction (see Figure 4.10)

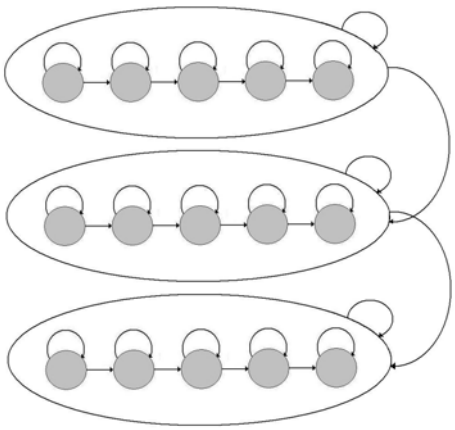

Figure 4.10: A embedded HMM. Transition between embedded states in different super states is not allowed.

In the embedded structure we need to keep track of the super states and the embedded states which lead to more parameters. The number of embedded states in the k'th super state is given by $S^k = \{s_1^k, s_2^k, ..., s_N^k\}$. Similarly the initial state probability distribution of the k^{th} embedded states is given by $\Pi^k = \{\pi_1^k, \pi_2^k, ..., \pi_n^k\}$. Each embedded state also requires a state transition matrix $A = \{ a_{ij}^k \}$ where a_{ij}^k gives the probability of making a transition from i to j in the k^{th} embedded state. Additional theory and implementation details about HMM can be found in [24].

5. Algorithm strategy

In this section the implementation of a face recognition system is described. The system is thought to be used through a camera feed watching over unknown scenery. Because of the little a priori knowledge the system aimed at being as general as possible. However some conditions are assumed. The area of survey should have decent lightning in order to capture faces. Also the camera should be aimed along a path such that persons being subjected to the camera will be in approximate frontal view. The recognition system is aimed at being a real-time system so speed is crucial and will be in focus when choosing methods for implementation.

Most of the algorithm in this section is based upon the work of Garcia, Zikos and Tziratas [12] and [13], but with modification in order to increase the overall speed. The consequence can be a less accurate detection which will be investigated in section 6.

5.1 Face detection

The first step of recognizing a face will be to determine if there is a face in the scene. Size and position is unknown and will have to be found before proceeding with the actual recognition process. For this purpose a skin detection algorithm is chosen for implementation as a first step. This will give locations and areas of possible face regions. For implementation the HSV color space is used for the reason that skin color regions are more clustered than in the RGB space and it more intuitive to use. In [12] 950 skin color samples were used as training data to construct 6 bounding planes in the HSV space. These planes will be used to determinate if a given color is a skin color.

$$
\begin{aligned}
& S \geq 10 \\
& V \geq 40 \\
& S \leq -H - 0.1\ V + 110 \\
& H \leq -0.4\ V + 75 \\
& \textit{if} \quad H \geq 0 \qquad S \leq 0.08(100 - V)H + 0..5\ V \\
& \quad \textit{else} \qquad S \leq 0.5H + 35
\end{aligned}
$$

The planes enclose a region in the HSV colorspace which allows for larger scatter along the saturation and value axis than along the hue axis. This is because lightning variance in the skin color is more invariant in the hue component than in the saturation and value component. The bounding planes are plotted in figure 5.1.

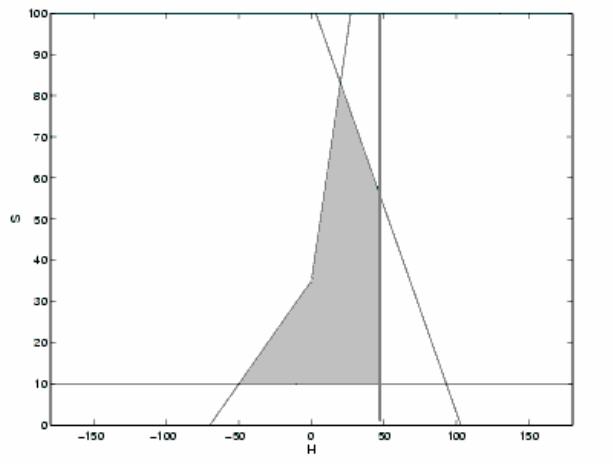

Figure 5.1: Shows the bounded region of skin colors in the Hue-Sat plane for Val=70.

The shaded area denotes color values which are labeled as skin color. If a color value is not in the region bounded by the plane it is discarded and will not be included in further searches. In figure 6.1.1-3 discarded pixels are shown as black. Overall the segmentation results shows a good detection on the skin color present, but also some unwanted noise like bright hair color and background with “skin like” appearance. This indicates that the segmentation alone can have trouble in pinpointing a face in the scene.

In [12] a color quantization into 16 different colors was done. This was done to create areas with homogeneous color. However as described in section 3, to get a good color quantization it becomes a hard optimizations problem. In this approach quantization will not be done, instead regions are created by expanding regions. The creation of a region starts at a non-black pixel in the segmented image. From there the color distance between neighboring pixels are calculated by the Euclidian metric. If the distance is sufficient small the neighbor pixel is included in the region. The neighbor pixels are connected as a 4-connected path (see figure 5.1.2). The starting color value will be passed from one pixel to the next. This means that a region containing an area gradually changing from one color to another will be split into several regions. If the original color value was not to be passed on, we would only get one big region

which is an unwanted property. Too small regions will be disregarded at once in order to prevent the numbers of regions growing large if there is a lot of noise present in the image. Also experiments show that small regions are often contained in bigger region if they appear inside a face.

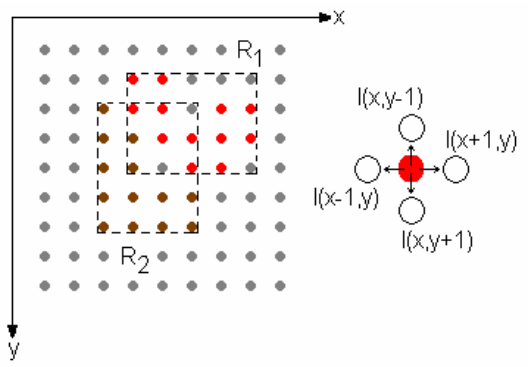

Figure 5.2: Shows the generation of regions during first phase. Pixels of similar color will be combined as a 4-connected region. The final rectangle contains the whole region. Here 2 regions R_1 and R_2 are generated from red-like and brow-like pixel colors.

Basically a region expands while satisfying the condition

$$d\left(I(x_{start}, y_{start}), I(x_{start \pm i}, y_{start \pm j})\right) < T_{color}.$$

This ensures that the region will keep expanding as long as the distance between the start color and the border colors is sufficient small.

After finding the base regions containing homogenous color areas, new regions are created by merging one or more regions together. The merging is similar to the proposed scheme in [9]. One difference though is that we allow the merging of two regions placed above each other in the vertical direction easier than in the horizontal direction. This is done because there in many cases will be two regions covering a face if a person is wearing glasses. The two regions are often separated by the change in color coming from the glasses.

Let $C_0 = \{R_1, R_2, ..., R_n\}$ be the set of base regions created after searching the areas with similar color. New potential regions are built iteratively by merging adjacent regions. Adjacent regions

are defined to be regions which overlap or which are sufficient close to each other in the vertical direction.

A new region C_{k+1} is created by merging adjacent regions in C_k with regions in C_0. The final set containing all the possible face regions is $C_{final} = \bigcup_{k=0}^{K} C_k$. The number of iterations *K* should not be too large as it will increase combinations which are time consuming. Also the final regions should have their aspect ratio checked, and discard the ones with too extreme values. Too small regions can also be discarded.

The final step should be a texture detection algorithm in order to find the face among the candidate regions in C_{final}. The method must be able to cope with different facial expressions, and small changes in illumination, scale and orientation. A gross but fast method is preferred as the number of candidate regions can be large. The result is to classify a region as a face or a non-face and aims at removing false alarms raised by objects with color similar to skin color and with similar aspect ratios, such as other exposed body or parts of the background. Basically the method is the last step of the face detection algorithm and will have to decide wherever a region in C_{final} is a face or not. This part of the algorithm will also require the most time as an image can have hundreds of regions to investigate. From the work of Garcia, Zikos and Tziratas a complete wavelet package tree was done and the variance was extracted from each of the coefficients. They used the Daubechies wavelet [8] with eight points in its filters, while in this proposed algorithm, the Haar filters are used as a wavelet basis. Convolution between the Haar filter and the image regions work very fast because the Haar filter only has two points. Also instead of using the wavelet package tree decomposition, the standard transform is preformed. The simplicity allows for a very fast processing.

From the methods listed in section 4 only the wavelets provide a multiscale analysis. It also decomposes the image into spatial and frequency components at the same time, and is robust under varying illumination. The decomposition can use the average of the RGB values (grayscale) of each pixel in the candidate region. The levels of a sub-band filtering and the coefficients are show below in figure 5.3. Each box contains information of a specific scale and orientation (see section 4.6).

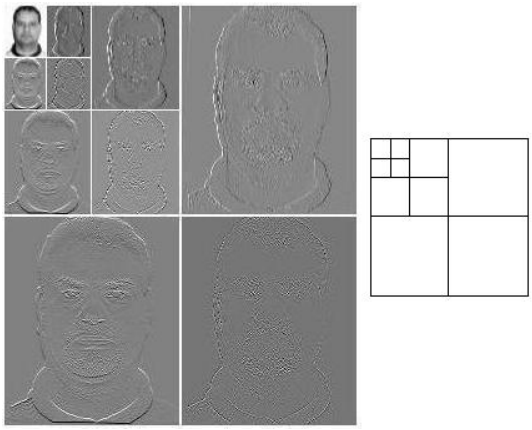

Figure 5.3: A discrete wavelet transform using Haar basis functions.

Discriminatory information is extracted from the coefficient images by calculating statistical moments like variance. The feature vector of an image region can use different parts of the coefficient images to get detailed information about the variance. In [12] and [13] the variance was extracted from the whole wavelet coefficient. Here we will try tree different methods for the extraction of variance. For example the coefficient images can be split horizontally in three equal regions representing forehead, eyes, mouth and chin. Figure 5.4 shows how the variance can be calculated from wavelet coefficients in different ways.

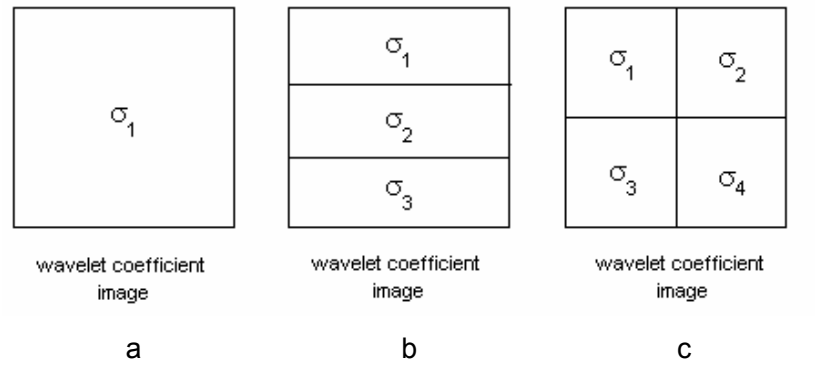

Figure 5.4: Extraction of variance from the wavelet coefficients

A feature vector can then be created by gathering the calculated variances.

Each region will then be described by a feature vector v_{region}. To determine if a region represents a face the distance between the feature vector and a vector representing an average face is calculated

$$d(v_{region}, v_{averageface}) \leq T_{face}$$

If the distance is sufficiently small this region will ideally be of a face. Finally one region of overlapping regions will have to be chosen. Regions with the lowest distance can be picked. They can be normalized by size so that larger areas are picked out more easily. For example if two regions or more accepted regions are overlapping, or one region is embedded in others, the one region with the lowest distance $d(v_{region}, v_{averageface})$ is chosen. Normalization by size can by done by accepting the lowest distance

$$d_{final} = \frac{d(v_{region}, v_{averageface})}{r_w r_h},$$

where r_w and r_h are the regions width and height. In Section 6 face detection with different parameters settings are tested.

5.2 Face recognition

The face recognition method is applied to one or more regions from the face detection results where each region *i* satisfy the constraint $d(v_i, v_{averageface}) \leq T_{face}$. Similar to the face detection the face recognition also has to cope with different facial expressions, illumination and variations in scale and orientation. A multiresolution approach like the wavelet done in the face detection is believed to be best suited for these situations, as they are very robust against variation in scale, small rotations and illuminations. However a more detailed analysis should be applied than was done for the face detection since it is to distinguish between faces of different persons. The number of regions is now reduced dramatically as each region should represent a face in the image. Here we will use the wavelet tree decomposition like in [13], which is a more detailed way to analyze the regions, (see section 4.7). Figure 5.5 shows the coefficients from a two scale decomposition.

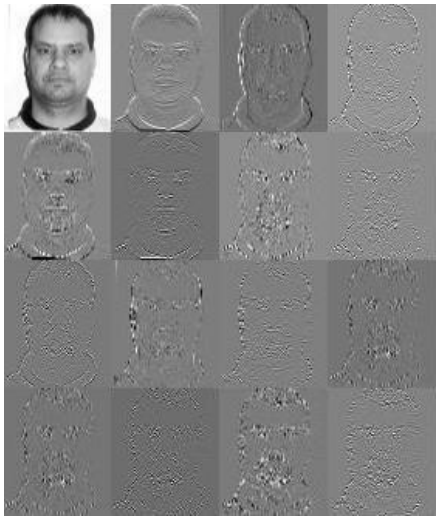

Figure 5.5: The 16 coefficients obtained from the wavelet tree decomposition.

Again discriminatory can be extracted by calculation the mean and the variance of the wavelet coefficients. The values are stored in a feature vector describing the regions and its content. Just as in the face detection case, the feature vector size depends on the number of scales used by the wavelet decomposition and how the calculations of the statistical moments are done. In section 6 the face recognition is tested with different feature vectors and different number of scales. The goal is to find settings which give the highest recognition results while keeping the false recognition as low as possible.

5.3 Implementation

To test the suggested algorithm a complete C program was written. For loading and saving images the IJG's JPEG-library was used. Also some code samples written by Fredrik Orderud were of great help during the implementation as they give a good example of how to use the library. The rest of the algorithms and methods discussed in section 5 were implemented from scratch and written in the emacs editor. Table A.1 in the Appendix shows an overview over some of the implemented methods and comments on what they do. The whole source code can be found on the CD attached with this project.

6. Results

In this section the algorithms described in Section 5 is tested with different parameters settings. The goal is to find settings which give high detection and recognition rates. Results are made by a program made in C using the gcc compiler in a UNIX operating system. The source code can be found in the Appendix. The face detection algorithms are tested on a collection of images picked from online newspapers and can also be found on the CD following this project. Some of the images contain multiple faces while others contain none. The images were picked in order to ensure variations in size, illumination and background. For the face recognition a standard face database was converted to JPEG and used. The goal is to find good settings at which the algorithms have high detection and recognition rates.

6.1 Test data

The Images found in the test data for the face detection are collected from different online newspapers. They are all color images and in the JPEG format. The resolution varies from images as small as 100 x 100 pixels to larger images with 1024 x 1024 pixels. The images were picked to ensure that faces with different size, location, different facial expressions and illumination are present. Variation in skin color is also an important issue, so the test data contains faces with darker skin color as well. Two images in the test data contain no faces in order to check for false detection results.

For the face recognition the dataset AT&T “The Database of Faces” [2] is used. The database consists of ten different images of each of 40 distinct subjects. For some subjects the images were taken at different times, varying the lightning, facial expressions and facial details like glasses. All images are taken against a dark homogenous background with the subjects in an upright frontal position. See [2] for online web address where one can also download the library.

6.2 Skin segmentation

The skin segmentation is the first step in the process of face detection and recognition. It is done to narrow down the search area in an image. If a color area is regarded as a non-skin color it will be disregarded from any further searches. This means that it is better to include non-face areas than to rule out face areas. The non-face areas which are included will ideally be ruled out by the wavelet analysis which is preformed after the segmentation. The segmentation was applied to the test data. Figure 6.1 show some results from the segmentation. Table 6.1 lists the number of successful, partly successful and failed face segmentation results. A result is said to be successful when the whole face is included regardless if any background included. Partly successful is defined to be the results where only parts of the face is included as skin color regardless if any background is included. Lastly failed segmentation results will be images where a face is ruled because the algorithm failed to identify the skin color.

successful:	58	89.2 %
partly successful:	6	9.2 %
failed:	1	1.6 %

Table 6.1: Results after segmentation of images containing 65 faces.

Figure 6.1 shows some results from the successful skin segmentation while figure 6.2 shows results where parts of the face were discarded. Figure 6.3 shows the single image where the segmentation failed.

Figure 6.1.: Some successful results from the skin segmentation

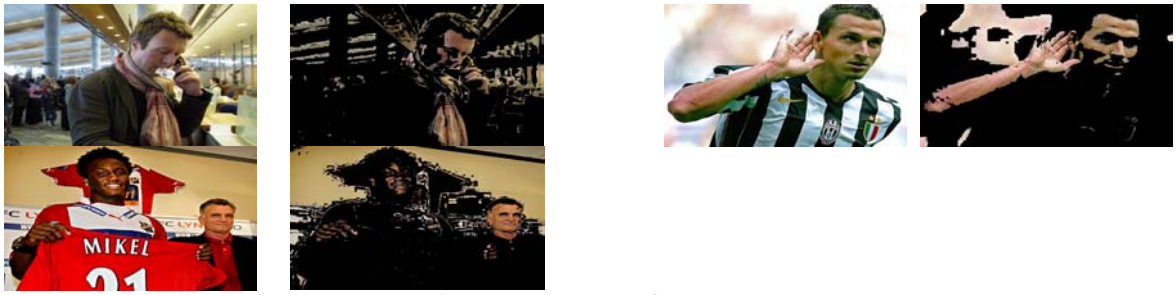

Figure 6.2.: Some partly successful segmentation results.

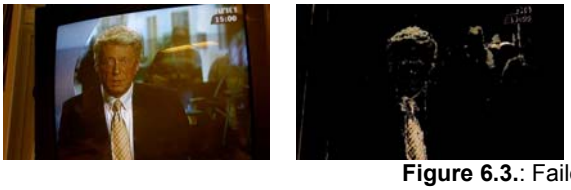

Figure 6.3.: Failed segmentation results

The results suggest that the segmentation has difficulties at detecting faces where illumination is poor or where the person has a very dark skin color. However if the area the partly successful images cover is large enough, the whole face can still be included as possible face region because of the region merging procedure. This will be investigated in the next section. The single image where the segmentation failed is from an image where a photo is taken from a television where the face appears. The capture of an image which is again sampled through a camera might give too poor color information for the segmentation to work.

6.3 Region merging

The algorithm as described in section 5 attempts to create possible regions in the image containing a face based on the segmentation results. The number of regions created is based K times adjacent regions are merged together and the Euclidean color distance d_{color} between adjacent pixels. Allowed pixel values have a color distance $d_{color} \leq T_{color}$. An analysis is said to be successful if one of the regions in C_{face} is a suitable region containing a face. In a failed analysis no region in C_{face} is of a face. This failure will result in no face being detected just as in the case of a failed segmentation and should be avoided.

Parameter setting:	*K = 3*	T_{color} = 15
successful:	21	32.3 %
failed:	44	67.7 %
average number of regions per image	130.12	

Parameter setting:	*K = 3*	T_{color} *= 35*
successful:	54	83.1 %
failed:	11	16.9 %
average number of regions per image	99.25	
Parameter setting:	*K = 3*	T_{color} *= 55*
successful:	45	69.23%
failed:	20	30.76 %
average number of regions per image	57.25	

Table 6.2: Results after the merging procedure with different T_{color}. values.

Parameter setting:	*K = 3*	T_{color} *= 30*
successful:	46	70.77 %
failed:	19	29.23 %
average number of regions per image	123.34	

Parameter setting:	*K = 3*	T_{color} *= 40*
successful:	47	72.30 %
failed:	18	27.69 %
average number of regions per image	95.43	

Parameter setting:	*K = 4*	T_{color} *= 35*
successful:	54	83.1 %
failed:	11	19.9 %
average number of regions per image	166.1	

Table 6.2: Results after the merging procedure with different T_{color}. values.

In general images get more, but smaller regions to investigate when T_{color} has a low value. The opposite is true when T_{color} grows large. From the results in table 6.2 we see that when $T_{color} = 15$ gives a low success rate. The reason is that the regions are too small to cover an entire face with only *K*=3 merging iterations. An increase in *K* will lead to a large number of new areas which is undesired. Instead the color threshold is increased in order to get larger areas. When $T_{color} = 55$ some of the regions are getting too large and cover more than a face. However the average number of regions in the images drop significantly which will increase the overall speed required to analyze an image. In the next sections the settings *T=35* with *K=3* are used because they give good performance while the average number of regions per image is low.

.

6.4 Detection results

The final stage in the face detection algorithm is the wavelet analysis which will be used to find extract regions with face like texture in the image. A feature vector representing a common face has to be built which can be done is several ways.

The average feature vector $v_{averageface}$ is constructed by taking the average of 15 feature vectors image samples where the face regions are extracted manually. The image samples used for the training are not the same as used in the test samples. A region is labeled as a face if $d(v_{region}, v_{averageface}) \leq T_{face}$ where *d* is some similarity function like the Euclidean distance or the Bhattachatyya distance, Table 6.3 shows the final detection results and the parameter settings which were used. Optimal detection results are regions which enclose a face and only a face, while suitable detection indicates regions which encloses most of the face or include the face and some additional features like hair or background. Suitable detections can still be useful for recognition especially if some additional normalization methods are applied. If a face was not found it means that no regions covered the face or a region covered it but it also included too much of the surroundings. Wrong regions are regions where $d(v_{region}, v_{averageface}) \leq T_{face}$, but the region contain no face.

Feature vector type: A

d = Bhattachatyya

	Scale = 2, T_{face}= 3.5	*Scale = 2, T_{face}= 0.5*	*Scale = 3, T_{face}= 3.5*	*Scale = 3, T_{face}= 0.5*
optimal detection:	47.5 %	43.9 %	39.30 %	52.38 %
suitable detection:	25 %	29.3 %	42.85 %	26.19 %
face not found:	27.5 %	26.8 %	17.85 %	21.42 %
wrongly accepted	63	33	56	14

Feature vector type: B

d = Bhattachatyya

	Scale = 2, T_{face}= 3.5	*Scale = 2, T_{face}= 1.5*	*Scale = 3, T_{face}= 3.5*	*Scale =3, T_{face}= 1.5*
optimal detection:	58.33 %	52.78 %	55.56 %	58.33 %
suitable detection:	19.44 %	19.44 %	13.89 %	16.67 %
face not found:	22.22 %	27.78 %	30.56 %	25 %
wrongly accepted	42	18	27	9

Feature vector type: C

d = Bhattachatyya

	Scale = 2, T_{face}= 3.5	*Scale = 2, T_{face}= 1.5*	*Scale = 3, T_{face}= 3.5*	*Scale = 3, T_{face}= 1.5*
optimal detection:	53.84 %	40 %	43.24 %	32.21 %
suitable detection:	16.92 %	23.08 %	18.92 %	9.76 %
face not found:	29.23 %	36.92 %	37.8 %	61.03 %
wrongly accepted	29	12	14	3

Table 6.3: Results after the merging procedure with different T_{color}. values.

The results listed in table 6.3 show that the best results are produced when the variance is calculated as show in figure 5.4 A and figure 5.4 B. A three scale decomposition gives in most of the cases better result, but only if the regions are not getting too small. For example when the variance of the wavelet coefficients are split into four like in figure 5.4 C, the areas will get too

small to give any valuable information in the third level of the decomposition. In general a higher threshold value T_{face} gives a higher percentage of optimal and suitable detected regions, but also increases the amount of wrongly accepted regions. While some faces are lost in the color segmentation it is still a vital stage to have a decent run speed. Detecting faces in images takes about 1-5 seconds on a standard workstation. Considering that the 30 test images were picked to ensure a large variation in facial expressions, illumination and scale, a detection rate of 70-75% is a good result and shows the wavelet approach is a viable one. Figure 6.4 shows some of the optimal detection results while figure 6.5 shows some of the suitable detection results. Lastly figure 6.6 shows some of the failed detection results.

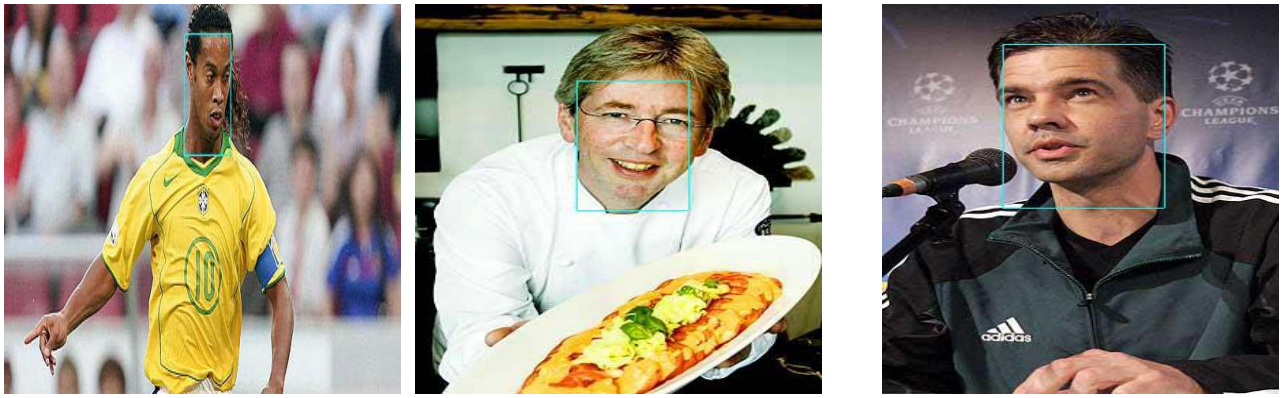

Figure 6.4.: Some optimal detection results from the face detection.

.

Figure 6.4.: Some optimal detection results from the face detection

Figure 6.5.: Suitable detection results.

.

Figure 6.6.: Failed results. Note that successful results are also present in images with multiple faces.

6.5 Recognition results

For testing the face recognition the AT&T “The Database of Faces” is used. It consists of 10x40 = 400 images in total, where 10 images are of the same individual and a total of 40 individuals exist. The images are stored in the PGM file format, but in order for the implemented program to read them, they had to be converted into the JPEG format. This was done manually with an image editor and because of that, the 400 images were reduced to 3x40 = 120 images, where 3 images are of the same individual and there are 40 individuals. Sample images from the image database is shown in figure 6.7.

Figure 6.7: Sample images from the AT&T face database.

The implemented C program starts by extracting the feature vector of each image and then forms the mean vector for each class. A given class *i* will have its average feature vector v^i_{mean}. Next we check that each image *k* is classified into the correct class, looking for the minimum distance $d(v_k, v_i^{mean})$. Table 6.4 shows the results form the face detection.

	Number of Misclassified	Recognition rate
Feature vector type A	10	75 %
Feature vector type B	5	87.5 %
Feature vector type C	3	92.5 %

Table 6.4.: Face recognition results using the wavelet package tree.

From the results, it can be seen that the rates vary from 92.5% to 75% depending on how the feature vector is built. The results are decent, but not satisfying considering how normalized the face database is. In [14] a similar approach was done, but with the Daubechies wavelet base instead of the Haar wavelet and with a feature vector build like type A (see figure 5.4 A). They achieved a recognition rate to 100% to 96% on the FERET database. This strongly indicates that the Haar wavelet is too simple to give good results compared to other more complex wavelets. Training the whole database with 120 images takes about 10 seconds which is very fast. However speed is not that important when training the image database as it was when doing the face detection. Training of the face database does only need to be done once until a new image is added to the database.

7. Summary

In this project a real-time face detection and recognition system has been discussed and implemented. The main focus has been on the detection process which is the first and most important step before starting with the actual recognition. Computably intensive can give good results [24] but at the cost of the execution speed. The implemented algorithm which was done is project is build upon the work of [12] and [13], but the algorithm accuracy is traded for faster speed. The program needs between 1-5 seconds on a standard workstation to analyze an image. On an image database with a lot of variety in the images, the system found 70-75% of the faces. About 5% of the faces were lost in the segmentation phase, while the region merging stage lost about another 10%. The last percents were lost by the texture analysis. However for face detection the Haar wavelets have been proven useful by working fast and still manage good results. In a more controlled environment even better results are possible. For example a camera can be aimed along a path where the background is easily segmented out and the faces appear in similar scales. Then a new average face vector can be calculated and unknown faces will be detected more easily since there will be no interference with background and the average face vector is attuned to the most likely scale and orientation of the appearing faces.

For the face recognition the standard wavelet seems to be too simple. Also the fast processing done for the face detection is not required for the face recognition since there is not multiple regions to analyze. In [14] a wavelet package decomposition aimed for face recognition was done with the Daubechies [8] wavelet. On the popular image database FERET, the results were as good as 95 - 100% which outlines the strength of a multi resolution approach such as the wavelet.

8. Further research

Even though this project has discussed both a detection and recognition system no testing was done on how they perform together at the same time. The lack of a test dataset for this kind of setup makes it hard. Extending the algorithm to handle video stream would make it easier to test the whole detection and recognition as a complete system. Most of the content in this project has been aiming for usage at still images, but many of the methods discussed can also be extended to video streams. Issues on how to handle frame selection should be looked into since it can take a few seconds to do a full analysis and a camera might transmit 20 frames per second. For a practical implementation a reference frame is good place to begin, using it to check for motion in the video. When motion is detected the algorithm can run more throughout analysis like the one described in section 5.

Another interesting aspect which could be looked into is to extend the average face vector used for the face detection. The average face vector used by texture analysis was constructed by using 15 images where the faces were extracted manually. All were appearing as in an approximate frontal view. During the last stage of the texture analysis the distance is calculated between the feature vector of and the given region and the average face vector. So in other words the detection will detect faces of frontal view easier since they will be closest to the average face vector. An interesting approach can be to have several trained average face vectors. One face vector could represent frontal views while the others are averages of faces either rotated to the left or the right. This could enable possibilities of detecting not only the faces, but also directions the faces are turned.

References

[1] Albiol, A., Torres, L., and Delp, E. J. “Optimum color spaces for skin detection.” *Proceedings of the International Conference on Image Processing,* vol. 1,122-123 (2001)

[2] AT&T Laboratories Cambridge, “The Database of Faces”
webpage: http://www.cl.cam.ac.uk/Research/DTG/attarchive/facedatabase.html (last online 05.07.2006)

[3] Baum, L. E., Peterie, T., Souled, G. and Wiess, N. “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains”, *Ann. Math. Statist.* vol 41. No. 1 (1970)

[4] Belhumeur, P. H., Hespanha, J. P. and Kriegmand, D. J. ”Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection.” *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 19 (1997)

[5] Bell, A. H. and Sejnowski, T. H. “The Independent Components of Natural Scenes are Edge Filters”, *Vision Research.* (1997)

[6] Brunelli, R. and Poggio, T. “Face Recognition: Features versus Templates”, *IEEE Trans. On pattern analysis and machine intelligence,* vol 15. No. 10 (1993)

[7] Cardoso, J. F. “High-order contrast for independent component analysis” *Neural Computation, 11(1):157* (1999)

[8] Daubechies, I. “The wavelet transform, time-frequency localization and signal analysis.” *IEEE Trans. Inform. Theory,* vol. 36 (1990)

[9] Dazibao, M. “the HSV colorspace - Tutorial”,
webpage: http://www.mandelbrot-dezibao.com/HSV/HSV.htm, (last online 22.06.2006)

[10] Feris, R. S., Gemmell, J. and Toyama, K. “Facial Feature Detection Using a Hierarchial Wavelet Face Database.” *University of Maryland* (2002)

[11] Fu, Y. “Handbook of Pattern Recognition and Image Processing”, *New York: Academic* (1986)

[12] Garcia, C. and Tziritas, T. “Face Detection Using Quantized Skin Color Regions Merging and Wavelet Packet Analysis.” *IEEE Transaction on Multimedia, v*ol 1, NO. 3 (1999)

[13] Garcia, C., Zikos, G. and Tziratas G. “Face Detection in Color Images using Wavelet Packet Analysis.” *Institute of Computer Science Foundation for Research and Technology-Hellas*

[14] Garcia, C., Zikos, G. and Tziratas G. “A Wavelet-based Framework for Face Recognition” *ICS – Foundation for Research and Technology – Hellas*

[15] Gundimada, S. and Asari, V. “Face Detection Technique Based on Rotation Invariant Wavelet Features”, *IEEE ITCC* (2004)

[16] Hyvârinen, A. and Oja, E. “Independent component analysis: algorithms and applications.” *Neural Networks, 13(4-5):411* (2000)

[17] Johnson, R. A and Wichern, D. W. “Applied Multivariate Statistical Analysis”, *Prentice Hall,* fifth edition (2002)

[18] Kazakos, D. “The Bhattacharyya distance and detection between Markov chains”, *IEEE Trans. Inform. Theory,* vol. IT-24 (1978)

[19] Kailath, T. “The divergence and Bhattacharyya distance measures in signal selection”, *IEEE Trans. Commun.* Vol. COM-15 (1967)

[20] Krueger, V. “Gabor Wavelet Network for Object Representation”, *Christian-Albrechts-Universitat, Kiel.*

[21] Linde, Y., Buzo A. and Gray R. M. “An Algorithm for Vector Quantizer Design.” *IEEE Tras. On Communications,* vol 28, NO 1 (1980)

[22] Moon, H. and Phillips, J. “Analysis of PCA-based face recognition algorithms.” *Empirical Evaluation Techniques in Computer Vision. IEE Computer Society Press,* (1998)

[23] Mu, X., Artiklar, M., Artiklar, M. and Hassoun, M. “Training Algorithms for Robust Face Recognition using a Templete-Maching Approach”, *National Science Foundation (NSF)* (1999)

[24] Nefian, A. “A Hidden Markov Model-Based Approach for Face Detection and Recognition.” Georgia Institue of Technology (1999)

[25] Nocedal, J. and Wrigth, S. J. “Numerical Optimization”, *Springer Series in Operations Research, Springer.* (1999)

[26] Shakhnarovich, G. and Moghaddam, B. “Face Recognition in Subspaces”, *Handbook of Face Recognition, Sprieger-Verlag.* (2004)

[27] Sirisathikul, Y., Auwatanamongkol, S. and Uyyanovara, B. “Color image quantization using distance between adjacent colors along the color axis with highest color variance.” *Pattern Recognition Letters 25,* (2004)

[28] Skarbek, W. and Koschan, A. “Colour image segmentation – a survey -.” *Tech. rep, Institute for Techincal Informatics, University of Berlin*. (1994)

[29] Vezhnevets V., Sazonov, V. and Andreeva, A. “A Survey on Pixel-Based Skin Color Detection Techniques.” *Moscow State University* (2003)

[30] Zhao, W., Chellappa, R., Rosenfeld, A. and Phillips, P. J. “Face recognition: A literature survey”, *ACM Computing Surveys (CSUR),* vol. 35, issue 4. (2003)

[31] Xue, Z., Li, S. Z. and Teoh, E K. “Bayesian Shape Model for Facial Feature Extraction and Recognition”, *Manuscript submitted to Pattern Recognition*. (2002)

[32] Yang, C. Y. and Lin, J. C. “Color Quantization by RWM-cut”, *IEEE* (1995)

[33] Wundrich, I. J., Malsburg, C. and Wûrtz, R. P. “Image Representation by Complex Cell Responses”, *Neural Computation* (2004)

Appendix

The table below list some of the main methods used in the implemented C program and some comments on their usage.

Converts rgb color to hsv
cHsv **hsv_from_rgb** (cPixel rgb)

Preforms a skin based treshold on input image
void **treshSkin** (imageRGB *img)

Preforms a color quantization on the image. Input: number of colors, and sample rate
void **quantColor** (imageRGB *img, int colorsize, int sample)

Calculate and display the integral projection for the input data along X axis
void **integralProjX**(cPixel *pixeldata, int width, int height)

Convert given rgb color data into grayscale
void **grayScale**(cPixel *pixeldata, int width, int height)

Calculate and return colordistance between two rgb pixels
int **rgb_dist**(cPixel col1, cPixel col2)

Draws region onto the input image, input: image, region and draw color
void **plot_box**(imageRGB *img, cBox *box, cPixel color)

Generate regions from similar color, input: source image, target image to draw on
cBox ***generateRegion**(imageRGB *img, imageRGB *drawon)

Merge adjacant regions in two region data vectors. Input: region data 1, region data2 region data 1 count, region data 2 count, new region count
cBox ***merge_regions**(cBox *reg1, cBox *reg2, int rc1, int rc2, int *counter)

Creates a imageRGB from a region, input: region to create image of, original image
imageRGB **image_from_region**(cBox region, imageRGB *original)

Haar decompose a image, input: image, type referrs to direction
trans_image **har_avg**(imageRGB *img,int type)

Wrties a transformed coefficient as a jpeg file
void **write_transformed_JPG**(char *file_out,int quality,trans_image *img)

Table A.1.: Overview over some of the main methods implemented in the C program.

calculate the bhattacharyya distance between a featurevector and the average face image
double **dist_bhatta**(double *featurevec)

calculate the euclidean between a featurevector and the average face image
double **dist_leastsq**(double *featurevec)

caclulate the variance from a coefficients in the specified region
double **var_in_region**(double *source_data,cBox data_region)

extract the feature from a image using the wavelet tree decomposition
void **feature_extract_tree**(imageRGB *img_reg, double *feature_vec, int scale)

extract the feature from a image using the regular decomposition and variance is extracted after type A
void **feature_extractA**(imageRGB *img_reg, double *feature_vec, int scale)

extract the feature from a image using the regular decomposition and variance is extracted after type B
void **feature_extractB**(imageRGB *img_reg, double *feature_vec, int scale)

extract the feature from a image using the regular decomposition and variance is extracted after type C
void **feature_extractC**(imageRGB *img_reg, double *feature_vec, int scale)

display a featurevector in the console-window
void **print_featurevec**(double *featurevec)

find the region with the minimun face_distance proprty, input: vector of regions, number of regions
cBox **min_face_dist**(cBox *reg,int n)

train feature vector, input: number of files in training folder, feature vector size, number of scales to use in decomposition
void **train_feature**(int file_number, int fsize,int scale)

Adds accepted regions into groups. Regions are grouped if they are overlapping
void **add_to_group**(cbox_group *group,cBox *baseregion) {

Table A.1.: Overview over some of the main methods implemented in the C program.