Automated Face Detection and Recognition for a Login System

Lloyd A. B. Louw

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering at the University of Stellenbosch.

Promoter: Prof. B. M. Herbst

Co–Promoter: Dr. N. Muller

March 2007

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously, in its entirety or in part, submitted it at any university for a degree.

Signature: ________________________ Date: ________________________

Abstract

The face is one of the most characteristic parts of the human body and has been used by people for personal identification for centuries. In this thesis an automatic process for frontal face recognition from 2–dimensional images is presented based on principal component analysis. The goal is to use these concepts in eventual face–recognizing login software.

The first step is detecting faces in images that are allowed a certain degree of clutter. This is achieved by skin colour detection in the HSV colourspace. This process indicates the area of the image most likely corresponding to the face. Extracting the face is achieved by morphological processing of this area of the image. The face is then normalized by a transformation that uses the eye coordinates as input. Automatic eye detection is implemented based on colour analysis of the facial images and a 91.1% success rate is achieved.

Recognition of the normalized faces is achieved using eigenfaces. To calculate these, a large enough database of facial images is needed. The xm2vts database is used in this thesis as the images have very constant lighting conditions throughout – an important factor affecting the accuracy of the recognition stage. Distinction is also made between identification and verification of faces. For identification, up to 80.1% accuracy is achieved, while for verification, the equal error rate is approximately 3.5%.

Opsomming

Die gesig is een van die mees karakteristieke dele van die menslike liggaam en word al vir eeue deur mense gebruik vir persoonlike identifikasie. In hierdie tesis word 'n outomatiese proses, gebaseer op PCA, voorgestel vir die herkenning van vooraangesigte in 2–dimensionele beelde. Die uiteindelike doel is om dié konsepte te gebruik in gesigsherkennende aanteken sagteware.

Die eerste stap is om gesigte op te spoor in beelde wat matige besige agtergronde mag hê. Dit word gedoen deur velkleur opsporing in die HSV kleurruimte. Dié proses gee 'n aanduiding van die area in die beeld wat mees waarskynlik die gesig bevat. Gesigsekstraksie word gedoen deur morfologiese verwerking op hierdie dele van die beeld. Die gesig word dan genormaliseer deur 'n transformasie wat die koordinate van die oë as toevoer gebruik. Outomatiese opsporing van oë is geïmplimenteer, gebaseer op kleur analise van die gesigsbeelde, en 'n 91.1% sukseskoers is behaal.

Herkenning van die genormaliseerde gesigte word gedoen deur gebruik te maak van eiegesigte. Om dié te bereken benodig ons 'n groot genoeg databasis van gesigte. Die xm2vts databasis word in dié tesis gebruik aangesien die beelde konstante beligting het – 'n belangrike faktor wat die akkuraatheid van die herkenningsproses affekteer. Verder word daar onderskeid gemaak tussen identifikasie en verifikasie van gesigte. Vir identifikasie word tot 80.1% akkuraatheid verkry, terwyl die gelykbreekpunt vir vals–aanvaardings– en vals–verwerpingsfoute ongeveer 3.5% is.

Acknowledgements

I would like to extend my sincere thanks to the following:

- My parents for their continued encouragement and support towards both my academic and non–academic life.
- My promoter, Ben Herbst, for his friendly support and stress–free approach to my work.
- My co–promoter, Niel Muller, for his aid and guidance during the early stages of my thesis.
- My friends for the socialities and balance they provided during the seven years of my tertiary studies.
- The National Research Foundation for their financial assistance.

Contents

List of Figures

Chapter 1

Introduction

Face recognition is ubiquitous for personal identification. It is how human beings recognize each other in everyday circumstances where the other senses play little to no role. Passports, identification documents, drivers licences and many other official type documents make use of photos to confirm the identity of the owners. In many cases, the photo must be manually compared to the owner in order for a third party to verify the match. This is usually not a problem if the volume of data is small. In some cases, however, it may be necessary to identify a particular person between hundreds of possibilities (or more) given only an image of that particular person's face. Over the past few decades, advances in science and computer technology have made it computationally feasible to define and carry out face recognition mathematically, automate the process, and thus process large volumes of facial images in relatively short periods of time.

1.1 Background

The term *biometric* refers to any measurement or characteristic of the human form that is unique and that differs from person to person. Examples include the face, fingerprints, irises, retinas, handwritten signatures, voice, etcetera. All these can be, and are, used in practice in an attempt to uniquely identify individuals. An overview of the current state of biometrics can be found in [44].

Biometric identification has its roots in Paris, France, with a clerk at the Paris Prefecture of Police named Alphonse Bertillon. Described as somewhat of a problem child, it was he who first proposed in 1879 that criminal records be taken in the form of measurements instead of descriptions. His ideas were originally met with much resistance but after he had solved a murder, he was offered additional clerks to help him with his new filing system. Over several months they collected 11 measurements from 1800 individuals. On February 20, 1883, he identified his first repeat offender, a man guilty of stealing milk bottles, despite the false identity claim that he had been given. This accomplishment was very positively publicized and marks the birth of biometric identification methods. See [3] for a full account of these events and how they lead to the eventual discovery and use of fingerprints.

Automatic face recognition is a much newer, passive, non–intrusive biometric method of personal identification dating back to the 1960s. For decades people have tried to understand which features help us to recognize individuals and how they indicate age, gender and even beauty. Several scientific methods have been developed in an attempt to answer these questions in two–dimensional "flat" images. One of the first methods investigated was that of feature based recognition where features such as the distance between the eyes are measured. While this approach is relatively insensitive to factors such as lighting conditions, they are sensitive to others like facial expression and aging. It is also not clear which features are the most representative of the human face. The first documented work on computerized face recognition by pioneers Woody Bledsoe, Helen Chan Wolf and Charles Bisson in the 1960s followed this approach [7]. Little publicity of their work was allowed, however, as funding was provided by an unnamed intelligence agency.

1.1.1 Existing Face Recognition Methods

Since the 1960s many other algorithms have been developed. One of the most popular is Principal Component Analysis (PCA). In fact, many other methods use PCA as a basis and extend on the idea. A description of the different methodologies is mentioned below. Also given are many publications pertaining to them that are freely available at the face recognition website [66]:

- Principal Component Analysis (PCA) [41][42][43][34]
- Independent Component Analysis (ICA) [2][28]
- Linear Discriminant Analysis (LDA) [4][13][30][32][63]
- Elastic Bunch Graph Matching (EBGM) [55][56]
- Kernel Methods [1][29][48][59][60][64][65]
- Active Appearance Models (AAM) [10][11]
- 3–D Face Recognition [8][9][19]
- Support Vector Machines (SVM) [18][22][26]
- Hidden Markov Models (HMM) [37][38][39]

One method currently showing much promise is three–dimensional face recognition. These methods first acquire a three dimensional representation of the face, usually by using more than one camera or projecting a known light pattern onto the face and using simple triangulation techniques. This 3–D surface geometry can then be used in the recognition process [17]. This approach is not studied further here as it is a very new field of research and still has many unaddressed problems. This thesis is focused on the PCA (or eigenface) approach for the recognition of entire faces in ordinary 2–D images.

1.2 The FERET and FRVT Standards

As face recognition systems have been implemented in practice over the past several years, standards had to be developed according to which they could be tested and compared. In 1993 the Department of Defence (DoD) Counterdrug Technology Development Program Office sponsored the FacE REcognition Technology (FERET) program [46] with the goal of developing face recognition algorithms. Twenty–four proposals were initially received and five development contracts were awarded to the selected winners:

- Massachusetts Institute of Technology (MIT), Alex Pentland
- Rutgers University, Joseph Wilder
- The Analytic Science Company (TASC), Gale Gordon
- University of Illinois at Chicago (UIC) and University of Illinois at Urbana–Champagne, Lewis Sandler and Thomas Huang
- University of Southern California (USC), Christoph von der Malsburg

The FERET program also set out to create a standardized database of facial images. The FERET database was collected between 1993 and 1996 and contains a total of 14,126 images of 1199 individuals. Images presently represent varieties in factors such as compression, lighting conditions, head pose, facial expressions, and images taken more than a year apart, among others.

The creation of this database allows for informed comparisons between face recognition algorithms as they can be tested on the same data. Only algorithms developed under FERET–funding are allowed to participate in the FERET evaluations with the exception of a few that were invited. The purpose of the evaluations is to measure the performance of algorithms that can automatically detect, normalize and identify faces from a database.

The successor to FERET is the Face Recognition Vendor Test (FRVT) [5] that now provides independent government evaluations of commercially available face recognition technologies. The FRVT also helps identify future research directions for the face recognition community. By 2000 several face recognition systems were available commercially that were evaluated by the FRVT which, in 2002, organized the Face Recognition Grand Challenge (FRGC) to develop new face recognition technologies. The FRVT 2006 outcome will determine whether the FRGC goals are met. A list of participants can be found on the website [67].

The FRVT evaluation of a single algorithm consist of several tests. Briefly stated, the primary focus of the experiments is to gather statistical data on the performance of the algorithm when variations are introduced to the following factors:

- **Compression:** Specifically lossy image compression such as JPEG.
- **Distance:** Varying the distance of the subject from the camera.
- **Expression:** Using images of the same subject with different facial expressions.
- **Illumination:** Varying the lighting between certain images.
- **Media:** Using images acquired with digital, film and video cameras.
- **Pose:** Varying the head pose (rotation and tilt, both in and out–of–plane) of the subject.
- **Resolution:** Varying the resolution of the images.
- **Temporal:** Waiting extended periods of time between acquisition of images of the same subject (typically years).

Some of these factors can be complementary to one another. For example distance and resolution are separate variations while the important factor is the number of pixels constituting the face area. Under the distance experiments, in order to mimic security camera footage, the FRVT also carries out recognition tests from short video sequences of a subject walking towards a camera. The results of these experiments for the different algorithms tested in 2000 can be found in [5].

As we are considering face recognition for a login system, we will only consider specific problems. It is important that a face be detected in an image and normalized quickly, thereby avoiding inconvenient waiting periods while logging in. Even though face recognition using PCA is well documented, it is also necessary to streamline this stage and save as much preprocessed data as possible for the same reason.

1.3 The xm2vts Database

We make extensive use throughout this thesis of the xm2vts (Extended Multi Modal Verification for Teleservices and Security applications) database [33] which is a sufficiently large collection of biometric information. The primary intention of the database is for research and development of biometric identification technologies and contains high resolution still images of several head poses, video sequences, and recorded speech of predefined sentences. The database was collected in four sessions over a period of five months to allow for changes in appearance, mood and hair styles. Completed, the xm2vts database contains this information of 295 adult individuals (predominantly caucasian) of both genders, well distributed over age. The wearing of eyeglasses was also allowed and left as a choice to the person being recorded.

In this thesis we only use the still frontal images of individuals. These are sufficient for research and testing of a system where the users are expected to cooperate with the acquisition process. The images are provided in full colour at a resolution of 720×576 pixels and were captured using a Sony VX1000E digital cam–corder. The images were all taken under controlled conditions with constant lighting and a uniform blue background. More on the xm2vts database can be found on the website [68].

1.4 A Login System

In this thesis we are interested in a particular application of face recognition: a login system that can allow users access to a workstation and/or network. Typically, each participating login point needs to be fitted with a webcam or similar image acquisition device. To login, a user must face the device directly and fill the image frame with his/her face at which point a snapshot is taken. This image is then presented to the software that will attempt to locate the face, normalize it, and identify the user. It is assumed that lighting conditions are well controlled at the time of acquisition, and if not, that external colour normalization software is present. At this point we can make simplified assumptions about the images specific to this application:

- The images contain a frontal view of an entire face that adequately fills the image frame.
- The lighting is constant and uniform.
- The images are of adequate quality (resolution, focus, etcetera).

The problem with identifying a face in an image is in fact threefold and can be summarized as follows:

- **Face Detection:** Finding the area in the image that contains the face, i.e. finding the face itself.
- **Face Normalization:** Normalizing the face with respect to position scale and rotation.

- **Face Recognition:** Recognizing the face and matching it to that of an individual in a database of known users.

These problems are independent, generally requiring fundamentally different methods for solution which are the central focus of this thesis.

1.4.1 Face Detection and Normalization

The first problem we are faced with is that of detecting a face within an image. This is a challenging problem in image processing, especially if the photographic conditions are inconsistent. Still, one can find extensive research in literature on this subject and face detection can be accomplished by a variety of methods [24]. One method commonly used is that of skin colour detection where areas of colour close enough to the colour of skin are considered. Often used in close conjunction with this method is feature detection to detect areas unique to a human face, such as eyes, nose, mouth, etcetera. This can be used to determine whether the area of the image indicated by the skin detection algorithm is a face or not. This is the methodology followed in [24]. To build a robust face detection system in practice it is usually necessary to combine several such methods.

A hybrid system is developed in this thesis. Skin colour detection is implemented by providing the computer with a training set of known skin pixels. Once the skin area is detected, morphological operators are used to extract the face from the image. Morphological face extraction is a relatively difficult task and, with the extraction method proposed in this thesis, accuracy of the cutout shapes is sometimes poor and there is much variety between them on a pixel level. This has a significant effect on recognition using principle component analysis. Overall, the face detection method proposed in this thesis performs well, both in the xm2vts database and in real–life cluttered images.

The second stage, after extraction, entails feature detection to detect the eye position within the extracted face. Their coordinates are used to normalize the face as accurately as possible with respect to position, scale and (in–plane) rotation in an attempt to minimize the negative effect of variations in the recognition stage. These stages of pre–processing on the image are highly image processing orientated and are covered in chapters 2 through 5.

1.4.2 Face Recognition

Recognition of the faces in this thesis is accomplished using principal component analysis, that is based on singular value decomposition and eigenfaces. This method has been studied extensively and is well docunented. An extension implemented in the development of our system is that of using images in full colour for recognition. The principle is identical to that of grey–scale recognition as is commonly found in literature, only it is repeated for each of the independent colour channels. This is implemented for completeness and is not likely to improve final recognition statistics.

Face recognition as developed in this thesis is a tool that can be operated in two different modes. Chapter 6 makes an important distinction between identification and verification of faces, both of which use the same recognition method. Experiments are further carried out to determine the effects of omitting eigenfaces as well as the use of different thresholds during the verification stage.

Part I

Face Detection and Normalization

Chapter 2

Spatial Filtering and Morphological Image Processing

The term morphology is commonly used in biology to describe the shape of organisms. Here we use it in a mathematical sense to describe shapes of objects and attributes we wish to extract from an image. In binary black and white images we can describe an entire image by defining a set, A, of all white pixels. Each element in A is a 2–D coordinate $(x, y) \in \mathbb{Z}^2$. Gray–scale images would then be described by $(x, y, i) \in \mathbb{Z}^3$ with i describing the intensity at position (x, y). Colour images (with three colour components, for example) would be in $\mathbb{Z}^5$. It is possible to extend many ideas introduced in this chapter to n–dimensional Euclidian space, $\mathbb{E}^n$, although the focus of this chapter is on binary and gray–scale images. Many morphological operations are written in set notation. See appendix A for basic set theory and definitions important in this chapter.

2.1 Spatial Filtering

Spatial filtering can be seen as the application of functions acting on n–D arrays in a way that the output not only depends on the value at the coordinate where they are evaluated, but on the values of a neighbourhood of points. It is convenient to conceptualize a *filter* or *mask*, $S(i, j)$, of size $m \times n$ in which the elements can be seen as weights. If these weights are binary, i.e. either 0 or 1, then $S \in \mathbb{Z}^2$. This filter is incrementally moved over the entire set A (an image) and at every point the *response* is calculated by a predefined relationship between the filter and the underlying points in A. An example of a 3×3 filter as well as a 3×3 subset (a sub–image) of A is shown in figure 2.1. Note that S need not necessarily be square. In general the response, $g(x, y)$, of the filtering of a function $f(x, y)$ with a filter S, is given by

$$g(x, y) = \sum_{p=-a}^{a} \sum_{q=-b}^{b} S(p, q) f(x - p, y - q) \tag{2.1}$$

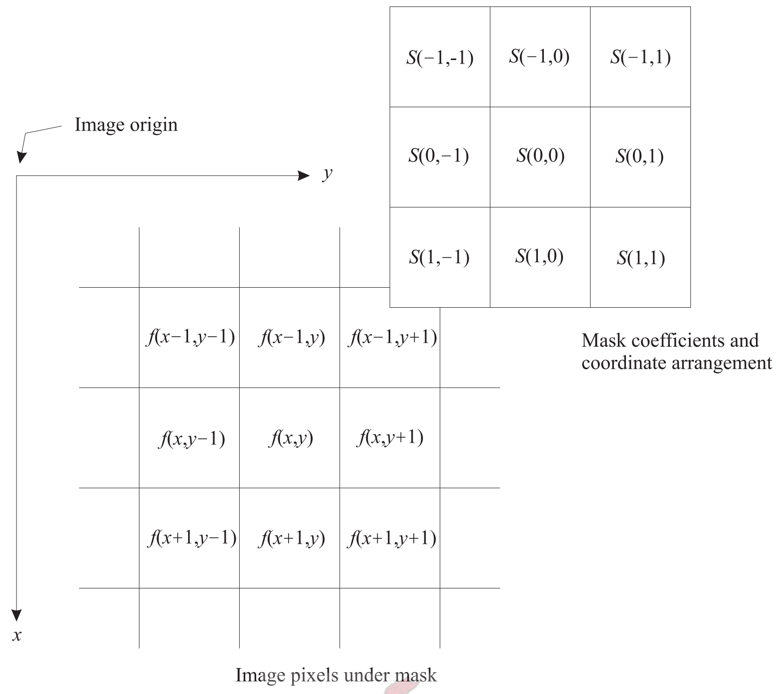


Figure 2.1: The process of spatial filtering.

where $[-a, a]$ and $[-b, b]$ span the length and breadth of S. This notation is often simplified to

$$g = \sum_{i,j=1}^{m,n} S_{ij} z_{ij} \tag{2.2}$$

where the S's are the filter coefficients, the z's the values in the image set A, and m and n the number of rows and columns in S and the underlying sub–image.

The concepts of spatial filtering come from the Fourier transform and the frequency domain of normal 2–D sets. In fact, many filters can be further analyzed by finding their frequency domain equivalent by means of 2–D Fourier transforms [16]. Filtering is also known as a convolution and holds close ties to correlation. Filtering has many practical applications such as smoothing, sharpening, noise removal, gap filling and edge detection.

Morphological operations that we consider from a filtering point of view in this chapter, are edge detection, dilation and erosion [16]. We also examine how dilation and erosion lead to morphological opening and closing. A table of more morphological operations in set notation can be found in [16, p. 548-549].

2.1.1 Edge Detection

To detect edges in an image, we first assume we are considering a gray–scale image, that is $A \in \mathbb{Z}^3$. Our goal is to enhance the image in such a way that the edges are emphasized. An obvious way to do this is to calculate the first derivative (gradient) of the image. More specifically, we need the magnitude of the gradient. For a continuous 2–D function $f(x, y)$, the gradient at a point (x, y) is defined as the vector

$$\nabla f = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}, \tag{2.3}$$

while magnitude of ∇f is given by

$$\begin{aligned} |\nabla f| &= \left(f_x^2 + f_y^2\right)^{\frac{1}{2}} \\ &= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{\frac{1}{2}}. \end{aligned} \tag{2.4}$$

We need to approximate (2.4) so that it can be applied to a discrete image. When we consider a 3×3 subimage

$$\begin{array}{|c|c|c|} \hline z_1 & z_2 & z_3 \\ \hline z_4 & z_5 & z_6 \\ \hline z_7 & z_8 & z_9 \\ \hline \end{array},$$

$|f_x|$ and $|f_y|$ at z_5 can be approximated by

$$|f_x| = |z_4 - z_6| \tag{2.5}$$

and

$$|f_y| = |z_2 - z_8|. \tag{2.6}$$

This method, however, does not take the pixels z_1, z_3, z_7 and z_9 into consideration. We can modify the above approximations by letting

$$|f_x| = |(z_1 - z_3) + (z_4 - z_6) + (z_7 - z_9)| \tag{2.7}$$

and

$$|f_y| = |(z_1 - z_7) + (z_2 - z_8) + (z_3 - z_9)| \tag{2.8}$$

and therefore use the filters

$$\frac{1}{3}\begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \qquad \text{and} \qquad \frac{1}{3}\begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

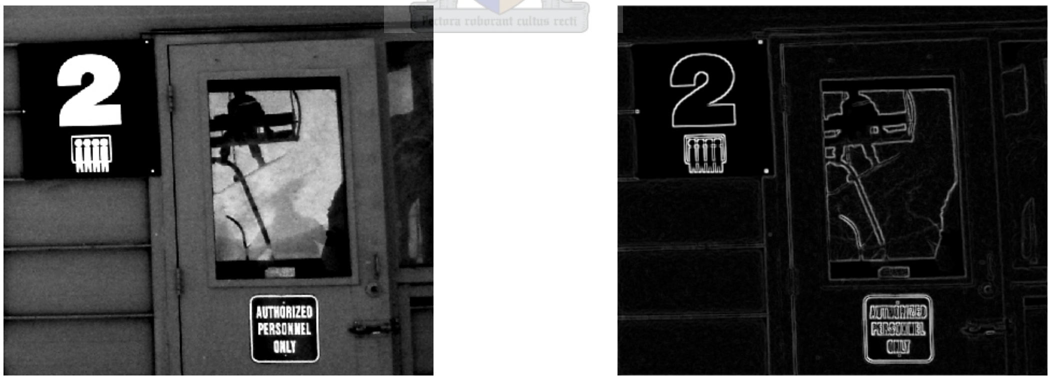
to approximate the vertical and horizontal components of the gradient respectively.

The above filters are known as Prewitt operators. A problem with these filters is that they are very sensitive to noise in an image. An idea would be to first smooth the image with a lowpass filter and then apply the filters for edge detection. These two steps can in fact be performed at once using similar gradient filters known as Sobel operators. These are given by

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array} \qquad \text{and} \qquad \frac{1}{4}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$$

for the estimation of the vertical and horizontal gradient components respectively. This variation uses a weight of two in the center, giving more importance to the center point. By doing so they achieve slightly superior noise suppression when approximating the gradient of a noisy image. Note that the sum of the coefficients of both the Prewitt and Sobel filters sum to zero indicating a zero response in areas of constant gray–level.

Using Sobel operators to approximate (2.4), we extract the edges from an image by filtering the image with the given masks and sum the absolute value of the responses. In this way we obtain an image of the same size as the original[1] which represents the gradient of the original image at each point. Edge detection using Sobel operators is demonstrated in figure 2.2. Note that the steeper the gradient is in the original image, the more pronounced it will be in the edge image.



(a) Original gray–scale image $A \in \mathbb{Z}^3$.

(b) Edges detected after filtering with Sobel operators.

Figure 2.2: Edge detection in a gray–scale image.

[1]Provided we make provision for the elements at the edge of the image set.

2.1.2 Dilation

Suppose we have two sets, A and $S \in \mathbb{Z}^2$, then the binary dilation of A by S, denoted by $A \oplus S$, is given by

$$\begin{aligned} A \oplus S &= \left\{ z | A \cap (\hat{S})_z \neq \emptyset \right\} \\ &\equiv \left\{ z | \left(A \cap (\hat{S})_z \right) \subseteq A \right\}. \end{aligned} \tag{2.9}$$

S is commonly known as a *structuring element* and is, in most cases, a much smaller set than A. Suppose we have selected a suitable S and a relative origin (say, in the middle of S), then the graphical interpretation of $A \oplus S$ is all coordinates z so that A and the translation of $\hat{S}$ by z overlap by at least one element. This is illustrated in figure 2.3.

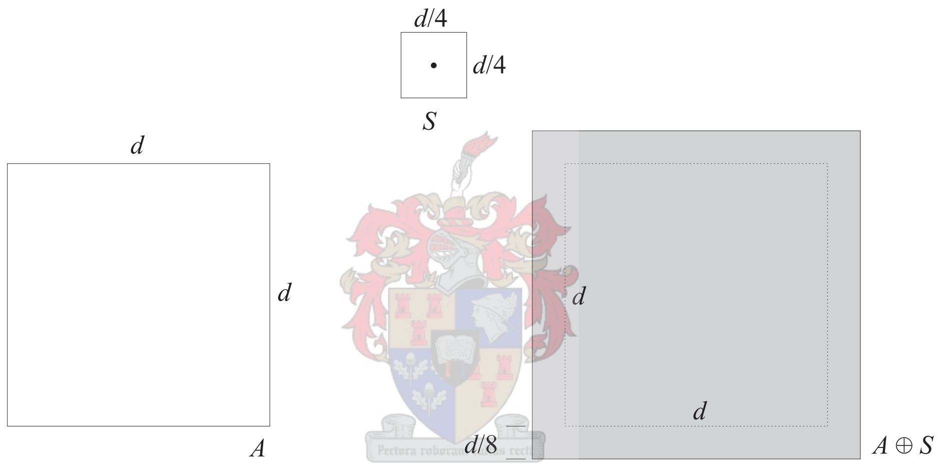


Figure 2.3: Dilation of an image set A by a structuring element S.

Extending dilation to $\mathbb{Z}^3$, in this case gray–scale images, is quite simple, though it is more convenient to view A and S as functions instead of sets. To obtain the gray–scale dilation of A by S, we also filter A with S. The dilation of a pixel is then the maximum of the selected neighbourhood plus the reflection of the structuring element. Gray–scale dilation is given by

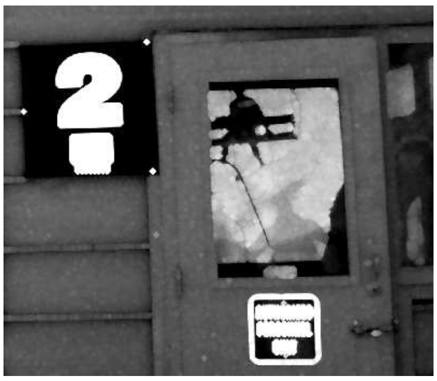
$$\begin{aligned} (A \oplus S)(s,t) = \max_{x,y} \{ & A(s-x, t-y) + S(x,y) \\ & |(s-x, t-y) \in D_A \text{ and } (x,y) \in D_S \} \end{aligned} \tag{2.10}$$

where D_A and D_S are the domains of A and S respectively. In other words, $(s-x, t-y)$ is a coordinate in A, (x,y) is a coordinate in S, $A(s-x,t-y)$ are the function values and $S(x,y)$ are the filter coefficients. As written, $A \oplus S$ can exceed the maximum allowed value in the image. This is solved by scaling the resulting dilation back to the original range, say $[0, 255]$ as is typically used.

The dilation of figure 2.2(a) using the given structuring element is given in figure 2.4(b).

0	0	1	0	0
0	1	1	1	0
1	1	1	1	1
0	1	1	1	0
0	0	1	0	0

(a) Structuring element S.



(b) Dilated image.

Figure 2.4: Dilation of a gray–scale image.

2.1.3 Erosion

Erosion is in principle very similar to dilation. If A and S are sets in $\mathbb{Z}^2$, erosion is defined as

$$A \ominus S = \{z | (B)_z \subseteq A\}. \tag{2.11}$$

The binary erosion of A by S is thus all points z such that the entire set S, translated by z, is completely contained in A. Erosion effectively cuts away a shape's outer edge as seen in figure 2.5. Erosion is useful for eliminating irrelevant detail in binary images such as free standing pixels and very small clusters. Any feature in the original image smaller than the size of the structuring element will be eliminated after filtering, while larger features will be reduced in size.

Gray–scale erosion is predictably similar to gray–scale dilation and is given by

$$\begin{aligned}(A \ominus S)(s,t) = \min_{x,y}\{&A(s+x, t+y) - S(x,y) \\ &|(s+x, t+y) \in D_A \text{ and } (x,y) \in D_S\}\end{aligned} \tag{2.12}$$

where the variables are defined as in the case of gray–scale dilation. The effect of this is the growing of dark regions and elimination of small, light clusters of pixels, much like in the binary case. The erosion of figure 2.2(a) with structuring element of figure 2.4(a), is given in figure 2.6.

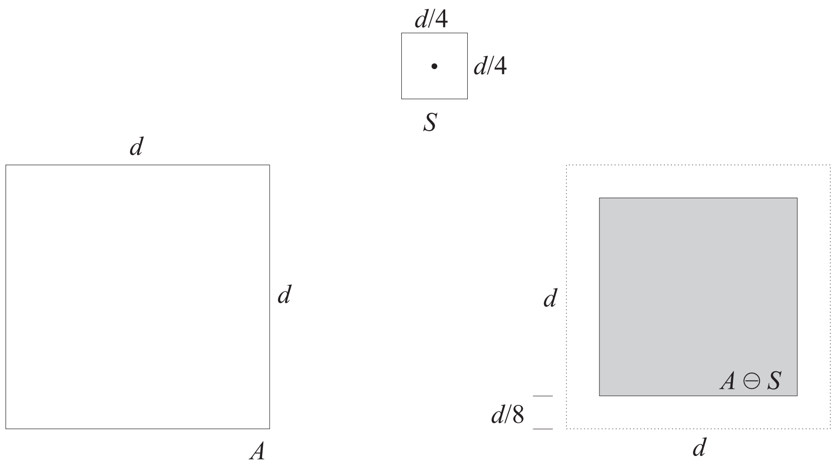


Figure 2.5: Erosion of an image set A by a structuring element S.

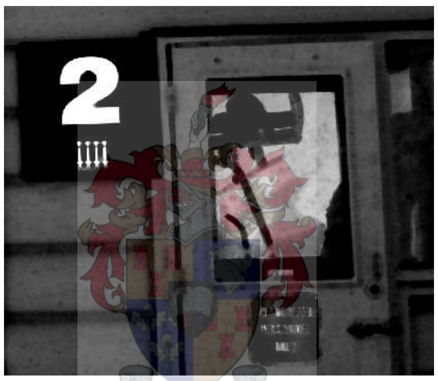


Figure 2.6: Erosion of a gray–scale image.

2.1.4 Opening and Closing

Opening and closing are morphological operations derived from dilation and erosion. We have seen that dilation expands an image, widening narrow bridges and protrusions. Erosion, on the other hand, shrinks the region and will eliminate these features. It is by a combination of these operations that we achieve opening and closing.

Opening of a set A with a structuring element S is denoted by $A \circ S$ and is given by

$$A \circ S = (A \ominus S) \oplus S. \tag{2.13}$$

In words the opening of A by S is the erosion of A by S, followed by the dilation of the result by S. The geometric interpretation of opening is the union of all $(S)_z$ where S, translated by z, is entirely contained in A. In set notation, opening is given by

$$A \circ S = \cup\{(S)_z | (S)_z \subseteq A\} \tag{2.14}$$

and is equivalent to (2.13).

The closing of A with S, $A \bullet S$, is only the reverse order of opening, given by

$$A \bullet S = (A \oplus S) \ominus S, \tag{2.15}$$

or equivalently

$$A \bullet S = \{\cup\{(S)_z | (S)_z \cap A = \emptyset\}\}^c . \tag{2.16}$$

Open and closing thus have much the same effect on sets as erosion and dilation (respectively) with respect to their effects on detailed features such as protrusions or gaps. The important difference is that they only effect these features and do not enlarge or shrink the overall set. There are certain mathematical relations between opening and closing [16] as there are between erosion and dilation, however they are theoretical in nature and not of importance to this discussion.

2.2 A Pseudo–Convex Hull Algorithm

A set A is said to be convex if the straight line L connecting any two points in A lies entirely in A. The *convex hull* of a set A is the smallest convex set H that entirely contains A. The difference, $H - A$, is called the *convex deficiency* of A. To make a set in $\mathbb{Z}^2$ (a binary image) convex, we can use the process of filtering and the structuring elements

$$\begin{array}{|c|c|c|}\hline 1&1&1\\\hline 0&0&0\\\hline 0&0&0\\\hline\end{array}, \quad \begin{array}{|c|c|c|}\hline 1&0&0\\\hline 1&0&0\\\hline 1&0&0\\\hline\end{array}, \quad \begin{array}{|c|c|c|}\hline 0&0&0\\\hline 0&0&0\\\hline 1&1&1\\\hline\end{array} \text{ and } \begin{array}{|c|c|c|}\hline 0&0&1\\\hline 0&0&1\\\hline 0&0&1\\\hline\end{array}$$

in the process described in [16, p. 539-541]. This process removes any concavity in the image and results in an octagonal convex shape (of arbitrary side lengths) containing the original shape A entirely. The difference $H - A$ can thus be very large.

Suppose now we want to approximate A by removing only some of it's concavity or, more specifically, only small concave regions. Our goal is to smoothen the edges of the set A in such a way that its general form is maintained. Let L be a line connecting points $\mathbf{p}$ and $\mathbf{q}$ as in figure 2.7. We want to construct a pseudo–convex hull H_p so that L is completely contained in H_p if the distance between $\mathbf{p}$ and $\mathbf{q}$ is small enough. That is, we want to eliminate the small concavity crossed by L but retain the concavity crossed by L'. Note that if A has a continuous boundary, this idea leads to $H_p = H$. The reason for this is that

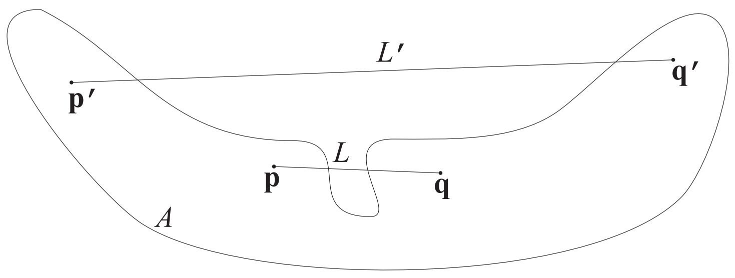


Figure 2.7: Lines connecting points within a set A.

in the case of a *continuous* and *concave* set, it is always possible to find a straight line, of any length, that is not entirely contained in A. However, in practice A is discrete and we are not faced with this problem.

One method of creating such a pseudo–convex hull would be by approximating the edge of A with a polygon using the Douglas–Peuker algorithm or its modification as suggested in [58]. The method, however, requires A to be defined as a polygon. As we will be manipulating discrete binary images, converting the edge of A to a polygon would create unnecessary processing overhead. An alternative that would seem attractive at first is to perform a closing (or some combination of opening and closing) on the set A. However, experimental implementation yielded unsatisfactory results as it can generally be shown that $(A \bullet S) \bullet S = A \bullet S$. Many of the concave regions are not sufficiently filled especially if the image is large and the structuring elements are small.

We instead follow an approach based on a slight modification of opening and closing. Instead of closing the image by performing one step of dilation followed by one step of erosion, we dilate the image n times, then erode it n times. We can define this as

$$\begin{aligned}(A \bullet S)^n &= (A \oplus^n S) \ominus^n S \\ &= [\ldots(([\ldots((A \oplus S) \oplus S)\ldots \oplus S] \ominus S) \ominus S)\ldots \ominus S]. \end{aligned} \tag{2.17}$$

The effect of applying 2.17 is illustrated in figure 2.8 and 2.9 with $n = 3$ and the structuring element S as shown. Note that different structuring elements would yield slightly different results. The shaded squares are the original elements of A while the hatched squares represent the elements added by each dilation as indicated.

Following the three steps of dilation are three steps of erosion as seen in figure 2.9. Note that none of the original elements are eliminated by the erosion process. We can also see from the result that the largest concavity (6 pixels wide) is filled by the dilations but not removed by the erosion. If the radius of the structuring element is r (in this case $r = 1$), then it is true in general that concavities of width $\leq 2nr$ will be filled by (2.17).

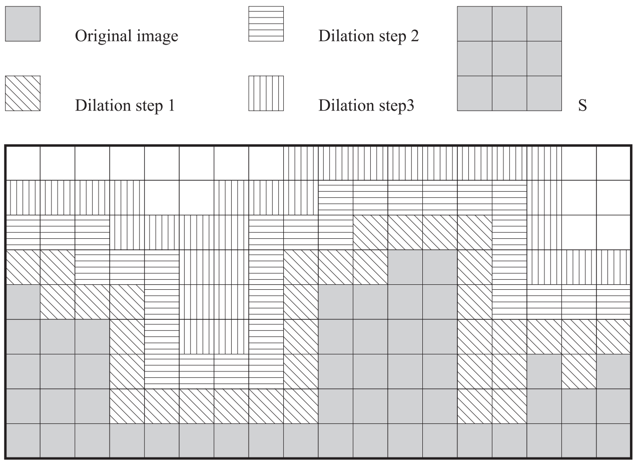


Figure 2.8: Three steps of dilation.

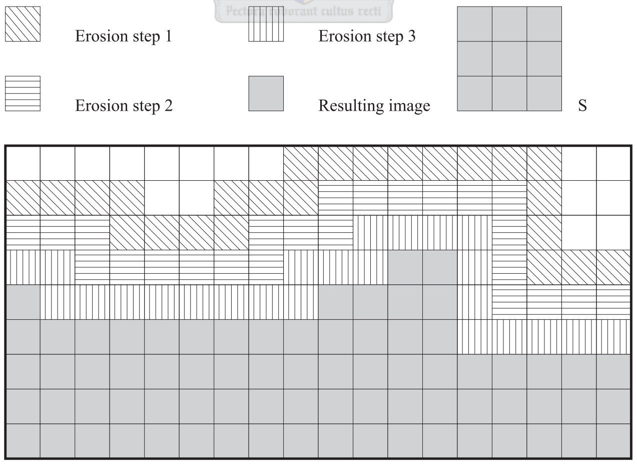


Figure 2.9: Three steps of erosion.

Chapter 3

Colour Spaces

When working with colour images, we need a way to represent them. That is we need a way to describe the colour information of each individual pixel. Colour, as the human eye sees it, is often considered three–dimensional in the sense that it is usually represented by three components that, combined, result in the final colour. Several representations of these three components are possible [54], including RGB, YUV, HSV and TSL to name but a few. Each has its own advantages and reasons for being used in practice. In this chapter we take a brief look at some of these representations as well as how to convert between them. We focus primarily on the RGB and HSV colourspaces, RGB because it is very commonly used in hardware and software, and HSV because it is a more intuitive separation of colour and is useful for face detection. We also briefly consider the normalized RGB and YC_rC_b colourspaces.

3.1 The RGB Colourspace

Colour images (in the visible light spectrum) are made up of three primary components. For an object *emitting* light, these components are red, green and blue (RGB). Any other colour can be created by mixing these three primary colours in different ratios. The RGB colourspace works on this principle and is extensively used in digital image processing as it is easy to implement in computer hardware.

If we scale each component between 0 and 1 (1 being full intensity), then table 3.1 shows the representations of different colours as an (r, g, b) triple where r, g end b are the red, green and blue components respectively. In effect we have a three–dimensional cube with red green and blue on the axes. This representation is shown in figure 3.1 and it can be seen that the coordinates and colours correspond to table 3.1.

In tandem with the RGB colourspace comes the cyan, magenta and yellow (CMY) colourspace. The CMY colourspace is obtained by shifting the origin to the white corner and flipping the axes of so that they lie between white, cyan, magenta and yellow. From this description

(r,g,b)	Colour
$(0,0,0)$	Black
$(1,1,1)$	White
$(1,0,0)$	Red
$(0,1,0)$	Green
$(0,0,1)$	Blue
$(1,1,0)$	Yellow
$(1,0,1)$	Magenta
$(0,1,1)$	Cyan
$(0.5,0.5,0.5)$	Gray
$(0,0,0.3)$	Dark blue
$(0.49,1,0.83)$	Aquamarine

Table 3.1: Colour representations in RGB.

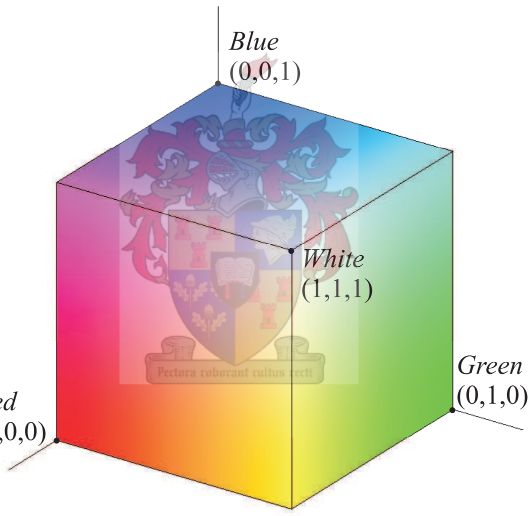


Figure 3.1: The RGB cube.

we can see that the relation between the RGB and CMY colourspaces is

$$(c,m,y) = (1-r, 1-g, 1-b). \tag{3.1}$$

CMY is often referred to as CMYK where K is "key" and refers to a separate black component. The CMY and CMYK representations are often used in close conjunction with RGB in a *light absorbing* environment such as in printers and photocopiers where ink is the medium of display. Black is considered as a separate component because it is cheaper than colour ink and because mixed colours result in poorer colour quality than pure black ink.

3.2 The Normalized RGB Colourspace

The normalized RGB colourspace [54] is easily obtained from the RGB values. Suppose the RGB triplet is $(r, g, b) \in ([0,1],[0,1],[0,1])$, then the normalized RGB values are given by

$$R = \frac{r}{r+g+b}, \qquad G = \frac{g}{r+g+b}, \qquad B = \frac{b}{r+g+b}. \tag{3.2}$$

We notice that $R + G + B = 1$ so that any one of the three components is redundant and can be omitted, reducing the space's dimensionality.

The remaining two components are called "chrominance" components and represent pure colour. A very useful property of this colourspace is that, under certain assumptions and in constant lighting, normalized RGB for matte surfaces is invariant to surface orientation relative to a light source. In other words, a dark colour will have the same normalized RGB representation as a light colour if the *colour* is the same. Normalized RGB thus separates the chrominance components from brightness or *luminance*.

Another colourspace achieving very similar separation of luminance and chrominance is the YC_rC_b representation. It is calculated by

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.3}$$

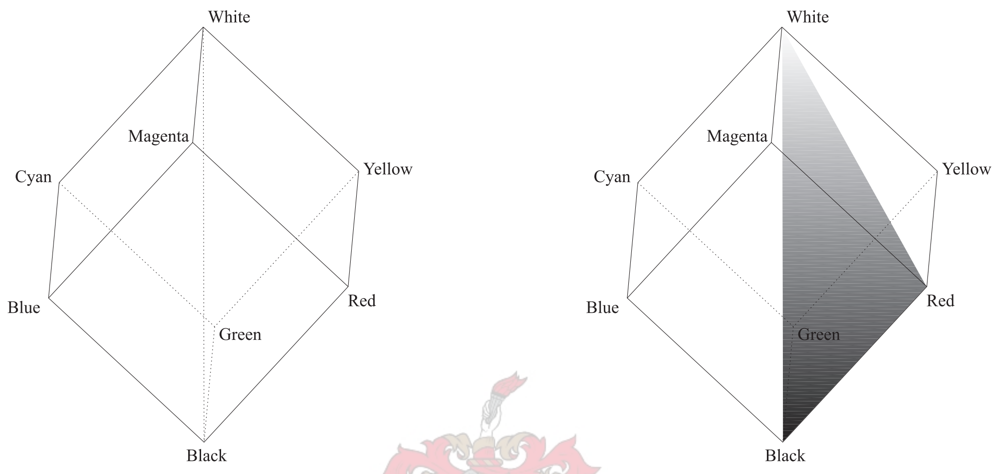
and is often used to achieve image compression by lowering the fidelity of the C_r and C_b components. In this representation, Y is the luminance component and C_r and C_b are the red and blue chrominance components respectively. Note that when $\begin{bmatrix} R & G & B \end{bmatrix}^T = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$ (white) then $\begin{bmatrix} Y & C_r & C_b \end{bmatrix}^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ (maximum luminance, no colour). At this combination of values, luminance is at a maximum and the two remaining colour components are meaningless.

3.3 The HSV Colourspace

The RGB colourspace is useful in practical computing as it is easy to output images in this format to monitors or printers. A major drawback is that it is less intuitive when it comes to mixed colours. People usually prefer to think in terms of the actual colour, its brightness and its purity and not as a fractional combination of primary colours. An alternative to the RGB representation is the hue, saturation and value (HSV, also known as hue, saturation, intensity – HSI) representation that has this intuitive advantage.

3.3.1 Constructing the HSV Colourspace

Let us consider the RGB cube of figure 3.1 and place it so that the diagonal intensity axis (between black and white) is vertical as in figure 3.2(a). Let us define the distance



(a) Rotated RGB cube and intensity axis between black and white.

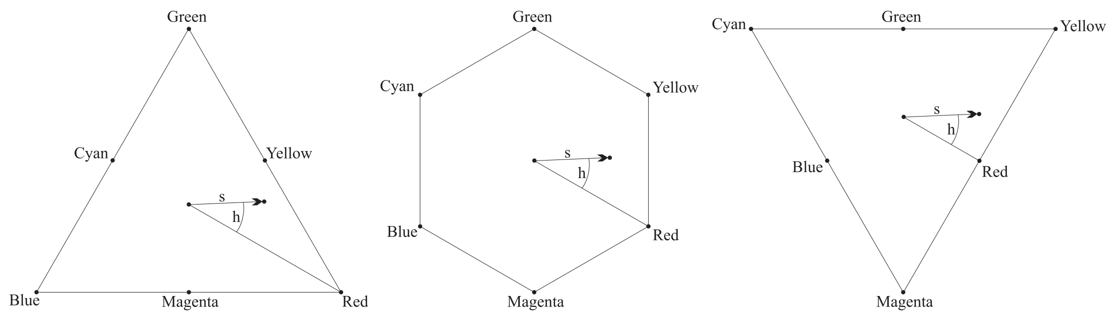
(b) The plane of all colours with red hue.

Figure 3.2: Rotated RGB cube depicting the intensity axis and a hue plane.

between the black and white corners as one. To determine the intensity (value) of a colour somewhere within this cube, we imagine a horizontal plane (perpendicular to the intensity axis) going though this colour point. The point on the intensity axis at which this plane intersects it would give us a point with value in the range $[0, 1]$ and is a measure of the brightness of the colour. The saturation is the perpendicular distance of our chosen colour point from the intensity axis and is a measure of the purity of the colour. Zero saturation is on the intensity axis while maximum saturation, 1, is at *any* point on the outer edge of the cube. The closer the colour point is to the intensity axis, the more it has been "diluted" by white (or shades of gray, depending on the intensity value). Colours further from this axis have a stronger, purer colour.

To determine the hue, we consider a plane containing the intensity axis as in figure 3.2(b). As this plane contains the three points, black, white, and in this case red, all colours on this plane will have the *same* hue. This is because the intensity (shades of gray) does not influence the actual colour but only lightens or darkens it. The same is true for the saturation – the closer we are in the plane to the axis, the more diluted the colour will be, though it will still be the same colour. Different hues are obtained by rotating this plane about the intensity axis and so is measured as an *angle* from a chosen zero axis, conventionally red.

Consider again a plane perpendicularly intersecting the intensity axis. As this plane moves from black to white, the cross–section of the cube will first be a triangle (figure 3.3(a)), then a hexagon (figure 3.3(b)) and then again a triangle (figure 3.3(c)). The dot is an



(a) Lower triangle. (b) Mid area hexagon. (c) Upper Triangle.

Figure 3.3: Cross–sections of the rotated RGB cube at different intensity values.

arbitrary colour with the hue and saturation as shown. The intensity is the distance of the plane from black. Note that the saturation value, $s \in [0, 1]$, is a fractional value of the distance of the colour point from the intensity axis. $s = 0$ is always on the axis while $s = 1$ is always on the edge irrespective of the intensity value and therefore the current cross–section. This allows us to replace these cross–sections with a circle of radius one as shown in figure 3.4. This is done without loss of generality or meaning of the values.

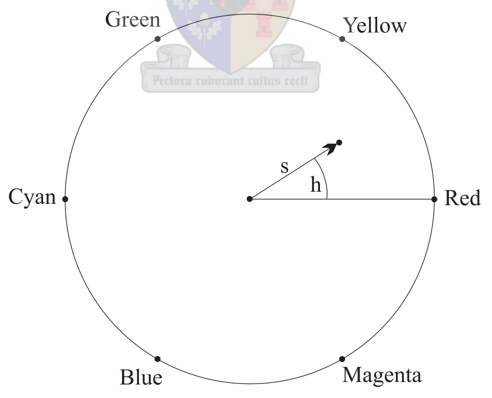


Figure 3.4: The hue – saturation circle.

A modification often made to the HSV space for simplification purposes is to *define* the intensity value to be the maximum of the r, g and b values. This means that any (r, g, b) colour where *any* of the components have maximum intensity, will be of maximum intensity. The HSV colourspace is therefore often represented as a cylinder or a cone as in figure 3.5.

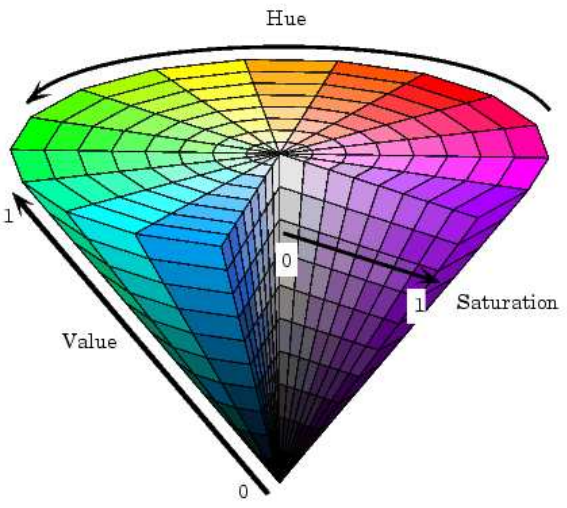


Figure 3.5: The HSV cone.

3.3.2 Converting Between RGB and HSV

Given a colour defined by (r, g, b) where $r, g, b \in [0, 1]$, we can calculate a *cylindrical* HSV triplet, (h, s, v), by the series of formulaes that follow. Their derivation is of tedious geometrical nature and can be found as supplementary material to [16]. Let $\text{MAX} = \max(r, g, b)$ and $\text{MIN} = \min(r, g, b)$, then from the geometry of figures 3.2 and 3.3,

$$h = \begin{cases} \left(0 + \frac{g-b}{\text{MAX}-\text{MIN}}\right) \times 60, & \text{if MAX} = r \\ \left(2 + \frac{b-r}{\text{MAX}-\text{MIN}}\right) \times 60, & \text{if MAX} = g \\ \left(4 + \frac{r-g}{\text{MAX}-\text{MIN}}\right) \times 60, & \text{if MAX} = b \end{cases} \tag{3.4}$$

$$s = \frac{\text{MAX} - \text{MIN}}{\text{MAX}} \tag{3.5}$$

$$v = \text{MAX} \tag{3.6}$$

where the output h is in degrees and s and $v \in [0, 1]$. For those who favour the conical model as in figure 3.5, s is given by $s = \text{MAX} - \text{MIN}$. At this point, some combinations of values deserve consideration:

- If MAX = MIN (i.e. $s = 0$) then h is undefined. In this case $s = 0$ and thus the colour is on the line of grays and has no hue, making angular coordinates meaningless.
- If MAX = 0 (i.e. $v = 0$) then the colour is pure black and therefore has no hue, nor

saturation. This is well illustrated in the conical model where the colour collapses to a single point where angle and radius are meaningless.

The inverse process, given an HSV triple (h, s, v), to calculate the corresponding (r, g, b) triple follows. First we let

$$\begin{aligned} h_i &= \left\lfloor \frac{h}{60} \right\rfloor \\ f &= \frac{h}{60} - h_i \\ p &= v(1-s) \\ q &= v(1-fs) \\ t &= v(1-(1-f)s), \end{aligned}$$

then

$$(r, g, b) = \begin{cases} (v, t, p) & \text{if } h_i = 0 \\ (q, v, p) & \text{if } h_i = 1 \\ (p, v, t) & \text{if } h_i = 2 \\ (p, q, v) & \text{if } h_i = 3 \\ (t, p, v) & \text{if } h_i = 4 \\ (v, p, q) & \text{if } h_i = 5 \end{cases} . \tag{3.7}$$

Chapter 4

Face Detection

Face detection has become an important part in many applications that involve some sort of human computer interaction such as surveillance, face recognition and facial image databases. Several methods have been developed for face detection based on neural networks, machine learning, template matching or Hough transforms, statistical wavelet analysis and skin colour detection. See [24] for a more detailed list. Most of these methods detect frontal facial images although the learning based as well as a modification of the feature based approach may be extended to detecting faces in profile. As we are aiming for eventual face recognition, we assume the facial images to be frontal and quite prominent in the image. This simplifies the problem to extracting what will probably be the most prominent object in the image.

To extract a face from a given image we need to somehow separate it from other objects and the background. We use skin colour detection in order to segment skin parts (such as the face) from the rest of the image. In order to do this we must first decide on an appropriate colour representation and then determine the actual colour range of skin in this representation.

4.1 Skin Colour Detection Methods

To detect skin colour we try to isolate a cluster of values in a colourspace within which skin colours occur. To do this we need to construct a three dimensional histogram in a chosen colourspace using only colours known to be that of skin. We thus need a training set of known skin colour pixels that have been obtained manually from digital photos. As lighting affects the skin tones in an image, those we use for the training set must have uniform and untinted lighting. An example of a cut–out containing 32,632 known skin pixels is shown in figure 4.1. The training set of images used is predominantly of caucasian individuals although several images of darker races are also used. Using a set of approximately 60 such cut–out images, we obtain 4,396,064 individual skin pixels to make up the training set. Their (r, g, b) values are then used to construct the histogram in RGB space given in

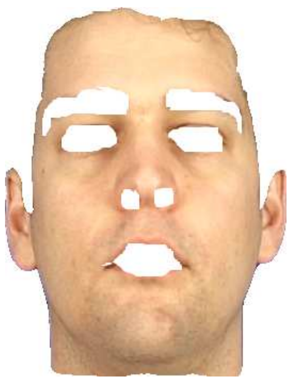


Figure 4.1: A cut–out containing 32,632 known skin pixels.

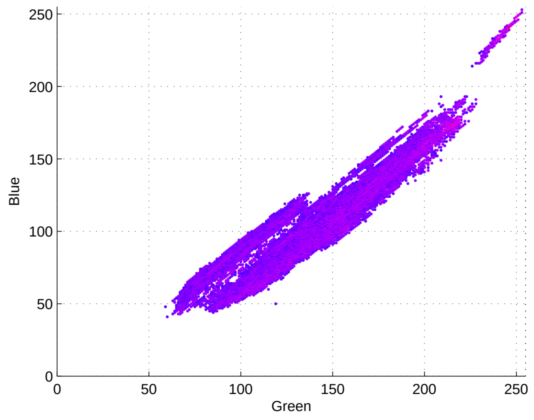
figure 4.2. The more hits an (r, g, b) coordinate in the histogram has received, the more likely it is that this colour is part of skin when it is encountered in a test image. Note that in computing, it is common standard that r, g and $b \in [0, 255]$ and not $\in [0, 1]$ as discussed in chapter 3.

As can be seen, the distribution in RGB–space is very widespread along the R, G and B dimensions. A methodology proposed and implemented in [24] is to enclose this volume in a three dimensional polygon. Alternatively, the histogram could be used directly in our eventual goal of determining skin colour probability, although this has its drawbacks. Even though methods exist to approximate this distribution to an arbitrary degree of accuracy, they may prove overly complicated for the final purpose of skin colour detection. The RGB colourspace is a bad choice as the histogram is not sufficiently clustered.

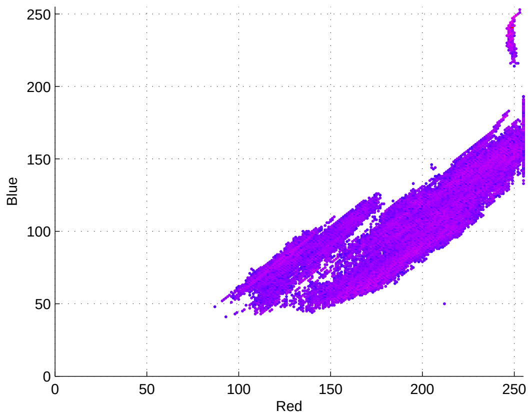
A better option would be to transform the colours to the YC_rC_b colourspace using (3.3). The use of this colourspace for skin colour detection is very popular, see [24], [25], [6] and [50], as it also reduces the dimensionality of the problem. It is also fast and therefore a good choice in applications that involve real–time face/skin tracking. Very similar to the YC_rC_b representation is the normalized RGB representation using (3.2). This colourspace also reduces to two dimensions and yields a more compact skin colour cluster as shown in figure 4.3. See [14] for more on using the normalized RGB colourspace for skin colour detection.

An even better colourspace to use for skin colour segmentation, however, is the HSV colourspace [47][62]. Using the same training set of skin colour pixels mentioned earlier, we calculate their corresponding $(h, s, v) \in ([0, 359], [0, 255], [0, 255])$ representations[1] using (3.4)–(3.6) and construct the HSV histogram shown in figure 4.4. We now wish to use this

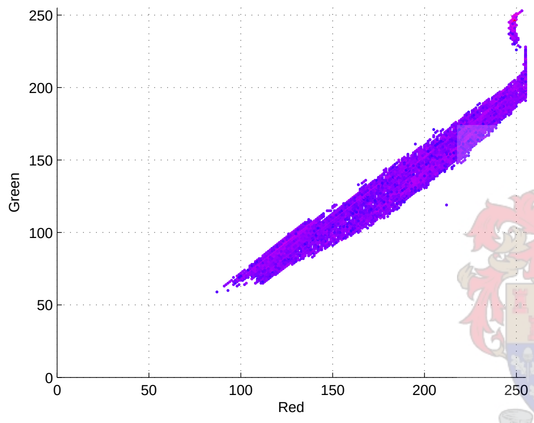
[1]The h value is an angle and represented here in degrees.



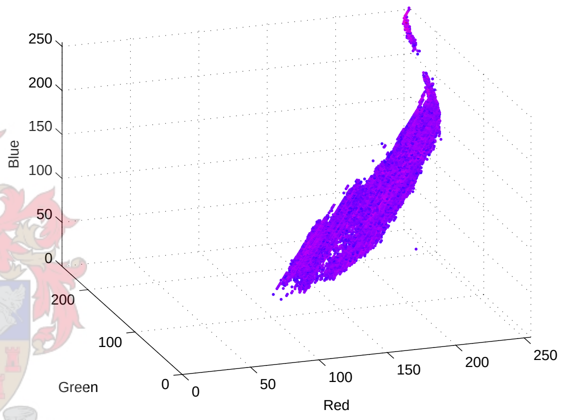
(a) View of the G–B plane.



(b) View of the R–B plane.



(c) View of the R–G plane.



(d) Skin colours in the RGB volume.

Figure 4.2: Distribution of skin colours in the RGB colourspace.

data obtained from the training set to determine whether a pixel from a test image is of skin colour or not. Methods to do this can generally be categorized into parametric and non–parametric methods as described below.

4.1.1 Non–Parametric Methods

Non–parametric methods typically use histograms directly to achieve skin segmentation. They have two major advantages. They are fast in training and usage, and they are theoretically independent of the shape of the distribution. That is they can be used in any colourspace. The disadvantage is that of storage space. The full RGB histogram, stored in a $256 \times 256 \times 256$ matrix can easily reach 64Mb – a hefty amount to be loaded from disk into memory every time we perform skin detection. It is possible to use coarser matrices though further analysis would need to be conducted to determine the effects of this [54].

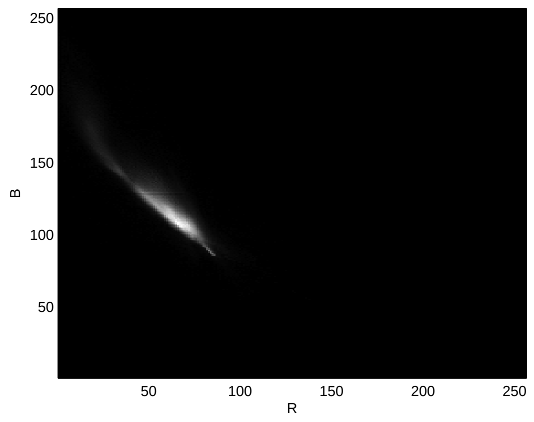


Figure 4.3: Distribution of skin colours in the normalized RGB colourspace.

We can also explicitly define a skin colour region (in HSV space for example) by inspecting figure 4.4 and manually setting upper and lower bounds for the (h, s, v) values. In this case we can build the rectangular bounding box

$$\begin{array}{rcccl} 0 & \leq & h & \leq & 40 \\ 60 & \leq & s & \leq & 160 \\ 90 & \leq & v & \leq & 255 \end{array} \tag{4.1}$$

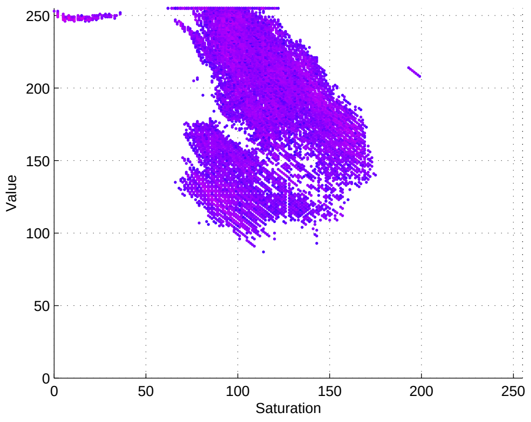
and consider all pixels with (h, s, v) values inside to be skin colour. This method is simple and fast but is a very coarse approximation and is predictably inaccurate at times. Strictly speaking, this is parametric approach although we will not consider it further due to its overly simplistic and inaccurate nature.

4.1.2 Parametric Methods – Gaussian Distributions

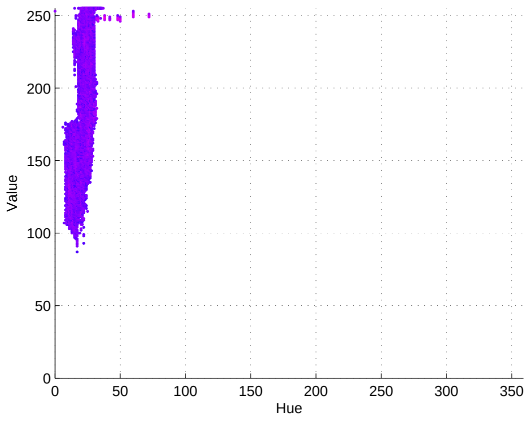
Several different approaches exist to approximate the distribution in figure 4.4 parametrically and thus save storage space at the cost of a few extra calculations. The most common include the single Gaussian distribution model, mixture of Gaussians, Gaussian clusters and the elliptical boundary model [54]. Considering the shape of the distribution of skin colour in HSV space, we will focus on the single Gaussian approximation.

Consider a colour pixel $\mathbf{c} = \left[\begin{array}{ccc} h & s & v \end{array}\right]^T$. The task becomes to calculate the probability that it is indeed skin. We thus want to construct a probability distribution function (pdf), more specifically an ellipsoidal Gaussian, that best fits the given histogram. The probability that $\mathbf{c}$ is of skin colour is then given by

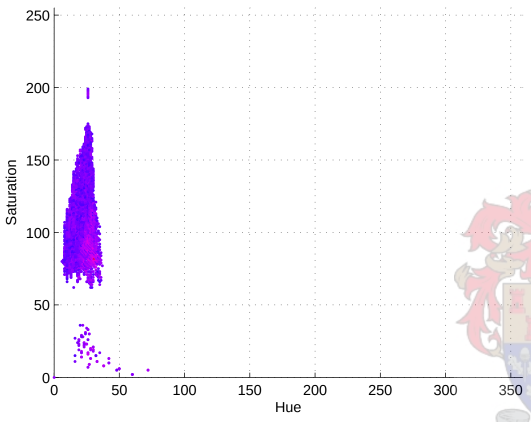
$$p(\mathbf{c}|\text{skin}) = \frac{1}{(2\pi)^{\frac{3}{2}} |\Omega_s|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\mathbf{c}-\mu_s)^T \Omega_s^{-1} (\mathbf{c}-\mu_s)} \tag{4.2}$$



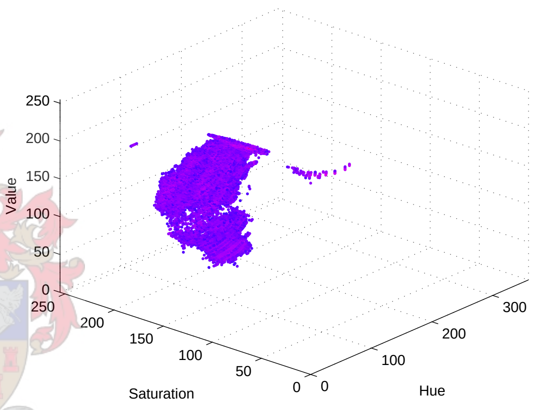
(a) View of the S–V plane.



(b) View of the H–V plane.



(c) View of the H–S plane.



(d) Skin colours in the HSV volume.

Figure 4.4: Distribution of skin colours in the HSV colourspace.

where μ_s and Ω_s are the mean vector and covariance matrix of the distribution respectively. Assuming we have a training set of skin pixels $\mathbf{c}_j,\ j = 1, \ldots, n$ then

$$\mu_s = \frac{1}{n}\sum_{j=1}^{n}\mathbf{c}_j \tag{4.3}$$

and

$$\Omega_s = \frac{1}{n-1}\sum_{j=1}^{n}(\mathbf{c}_j - \mu_s)(\mathbf{c}_j - \mu_s)^T. \tag{4.4}$$

To extend this idea and create a pdf consisting of multiple Gaussians, we simply construct it from a linear combination of k single Gaussians so that

$$p(\mathbf{c}|\text{skin}) = \sum_{i=1}^{k} \pi_i p_i(\mathbf{c}|\text{skin}) \tag{4.5}$$

where π_i are mixing parameters satisfying $\sum_{i=1}^{k} \pi_i = 1$ and each $p_i(\mathbf{c}|\text{skin})$ is a single Gaussian with distinct mean and covariance.

This alternative approach requires the histogram to be split into k regions with a Gaussian pdf constructed for each. This idea was implemented but lead to poorer results for skin detection than when using the single Gaussian approximation. The reason for this is that the distribution of skin colours in the HSV space is already very compact and the individual Gaussians become too localized, resulting in much false negative skin clasification. The general failure of the Gaussian mixture approach is counter–intuitive and can be attributed to the fact that clusters were only detected/defined manually in the HSV space. This inadvertently led to suboptimal Gaussians used in the mixture model. Alternatively, more sophisticated clustering methods, such as the K–means clustering algorithm [51], should be used on the original histogram to determine the primary clusters. This approach promises a possible increase in accuracy and should be further investigated in future work.

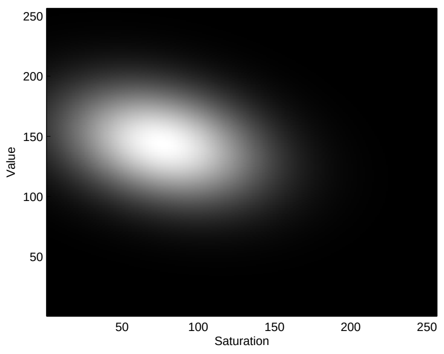
For our purposes, the single Gaussian approximation is sufficiently accurate. All the results in sections that follow are thus based on this approximation (4.2). See [14] for more on Gaussian approximations of skin colour distributions.

To further justify our choice for using (4.2) we compare the pdf generated by (4.2) to the actual (h, s, v) distribution obtained from the training set. If the distribution is indeed near–Gaussian, we should have a relatively good approximation. First we calculate μ_s and Ω_s from (4.3) and (4.4). These values are given as

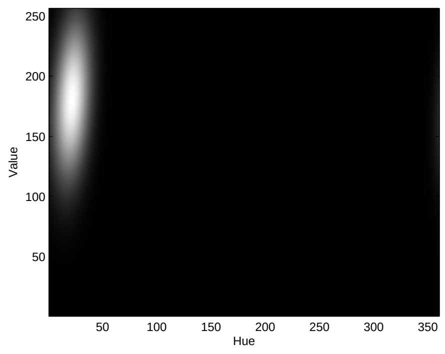
$$\mu_s = \begin{bmatrix} 21.0688 \\ 112.4239 \\ 179.5906 \end{bmatrix} \qquad \text{and} \qquad \Omega_s = \begin{bmatrix} 135.2 & 35.4 & 104.1 \\ 35.4 & 1096.0 & -355.3 \\ 104.1 & -355.3 & 2064.7 \end{bmatrix}.$$

By considering all combinations of (h, s, v) triples, we see that (4.2) has a maximum value at $(h, s, v) = (26, 101, 177)$. We now evaluate the pdf (4.2) on the three orthogonal planes intersecting this point. The pdf is shown in figure 4.5 and should be compared to the distribution histogram in figure 4.4.

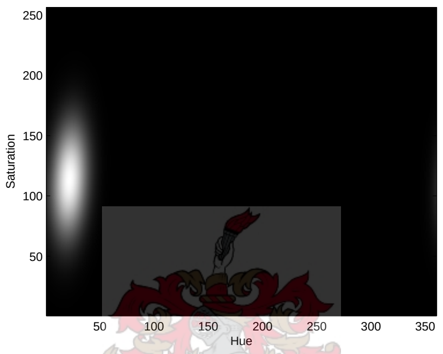
At this point there is a subtlety about calculating the mean and covariance worth noting. As hue is an angle, and thus measured in degrees, $h = 0°$ is adjacent to $h = 359°$. This fact is easily overlooked when considering a training set of (h, s, v) triplets and means that a color pixel with $h = 350°$ is equivalent to one with $h = -10°$. Even though these two representations are equivalent, they will not yield the same mean and covariance. As



(a) The S–V plane.



(b) The H–V plane.



(c) The H–S plane.

Figure 4.5: The Gaussian pdf (4.2) in the different planes, through the maximum.

the histogram is in fact much more widespread than depicted in figure 4.5 which has been subjected to a threshold, it is in fact continued off $h = 0°$ and onto $h = 359°$. These colours with very large h values are in fact very close to the primary cluster but would not convey this information to the mean and covariance. It is therefore for purposes of better approximation that we change all (h, s, v) triplets in the training set with $h > 330$ (empirically determined) to $(h - 360, s, v)$.

In the sections that follow, we will first determine a *skin probability map*, $P((h, s, v)|\text{skin})$, using (4.2) on the (h, s, v) representation of the original image (figure 4.6(a)). P is then converted to a binary mask using a suitable threshold value (figure 4.7(a)). Note that this method is equivalent to replacing the bounding box given in (4.1) with an ellipsoid with precalculated major and minor axes lengths and direction.

To reduce the noisy nature of this binary mask we use morphological operations. After this, the largest connected area of the mask is considered to be the face area around which we construct a pseudo–convex hull. Only the pixels in the original image corresponding to selected pixels in this hull are extracted and the resulting image, F, is assumed to be the face which is then cropped (figure 4.8).

4.2 Face Area Segmentation

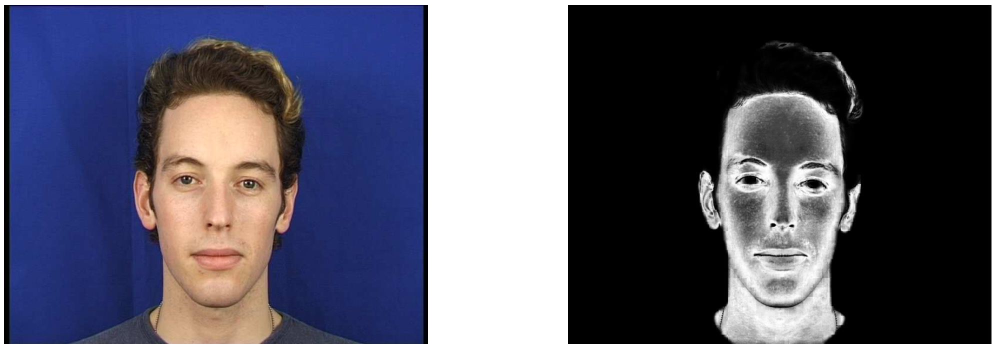
With (4.2) we now have a method to determine probabilistically whether a pixel forms part of skin in an image or not. We wish to use this to extract the face from an image that we will eventually align and use in the recognition process. At this stage we remind the reader of the following important assumptions pertaining to the original image:

- There is only one face in the image.
- The face is the largest connected region that is of skin colour.
- Lighting is sufficient, relatively uniform and not tinted (i.e. the light is white).

If adverse lighting conditions are present, the image should be manually adjusted, either in specialized software or by a user specified skin colour selected from the image. See appendix B for a brief discussion on manual colour normalization that can be applied to images if needed.

4.2.1 Obtaining an Initial Binary Skin–Map

Let us consider an image from the xm2vts database in figure 4.6(a). The first step, is to convert every pixel in the image to its HSV representation using (3.4) – (3.6). Once this is done, we apply (4.2) to every pixel to obtain a probability map $P((h, s, v)|\text{skin})$. The



(a) Original photo containing a face. (b) Skin probability map $P((h, s, v)|\text{skin})$.

Figure 4.6: Original photo and its corresponding skin probability map.

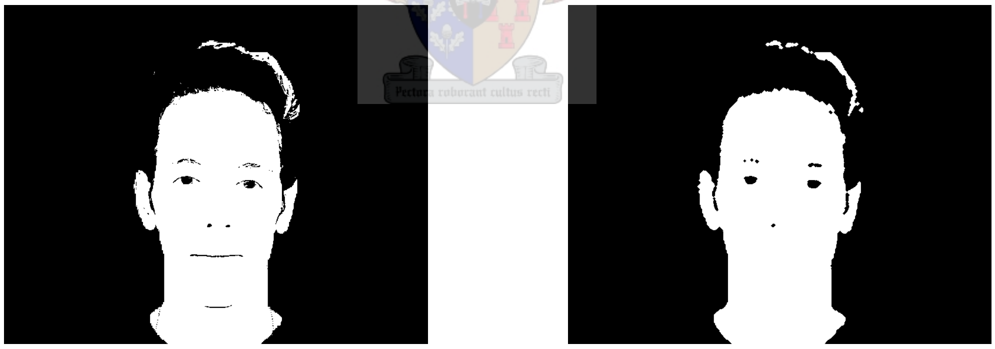
resulting skin map is given in figure 4.6(b) where high values (light areas) indicate high skin probability. We see that this estimation is quite accurate, however some dark patches are in this case evident in the center of the forehead and cheeks. The reason for this is these pixels have a very low saturation and high intensity values. Refereing to the HSV histogram in figure 4.4, these colours correspond to the small cluster of values near the $(h, s, v) = (0, 0, 255)$ corner. We will see shortly that this is not a serious problem, even in extreme cases, as we will always have low responses around the eyes and usually the mouth, which will all have to be remedied.

The next step is to obtain a binary skin map, $B_1(h, s, v)$, from $P((h, s, v)|\text{skin})$. Considering the relatively good accuracy of (4.2), we find that normalizing $P((h, s, v)|\text{skin})$ to the range $[0, 1]$ and applying a threshold of $\frac{1}{5}$ is very suitable. In other words, any value in the normalized skin map having a value greater than $\frac{1}{5}$, we consider to be a skin pixel. So we let

$$B_1(x, y) = \begin{cases} 1 & \text{if } \frac{\text{P((h,s,v)|skin)}}{\text{max(P((h,s,v)|skin))}} \geq \frac{1}{5} \\ 0 & \text{otherwise} \end{cases} . \tag{4.6}$$

Note that (4.6) suffers from the fact that it can be foiled by outlier pixels. In practice, to avoid this, we would set the top 5% (or so) of the values in the skin map to 0 or to the average of the others, thus removing potential outliers.

The result of applying (4.6) to the original skin map can be seen in figure 4.7(a).



(a) Result of threshold application to figure 4.6(b).

(b) Result of opening then closing (a).

Figure 4.7: Processing of figure 4.6(b) to obtain a binary skin map.

4.2.2 Refining the Binary Skin–map

At this point we have a binary skin map that might still be quite noisy. That is we might have many small unconnected clusters or free standing skin pixels that we would like to eliminate. Another problem with this initial binary map is that there may be disjoint clusters that actually form part of the same skin area. To overcome these two issues, we first perform morphological opening using (2.13) to eliminate small clusters and pixels. To fuse disjoint but very near clusters together, we perform a closing using (2.15). The result, $(B_1 \circ S) \bullet S$, can be seen in figure 4.7(b).

We now calculate the sizes of the different connected clusters. This can be done with morphological operations [16] or the process outlined in [20, p. 28-48] using run–length encoding and setting up equivalency tables. From our assumption that the face is the largest connected skin area in the image, we can discard all other regions to be left with one, large connected area, assumed to be the face. In the case of figure 4.7(b), this means we would discard the smaller connected regions corresponding to the hair in the original image.

A last remaining nuisance are the holes in the face area typically found at the eyes, mouth and so forth. This is also very common if the person in the image is wearing eyeglasses. These areas are not of skin colour but still form part of the face and we would therefore like to include them. There are several ways to fill these holes. We could use a combination of morphological extraction of connected components and region filling [16]. However, these sort of algorithms usually need a defined starting point inside the region to be filled. A methodology to overcome this is to fill all pixels between minimum and maximum rows and columns and use z–based erosion [6] to re–obtain the original outline.

An additional complication to these holes is that, in rare cases, they may be connected to the outside non–skin area by thin segments of non–skin pixels. In this case, any region filling would inadvertently refill the entire background. The solution lies in a modification of (2.17). The modification calculates the pseudo–convex hull and fills the holes in the image after the n dilation steps but before n erosion steps and is similar to the process used in [50]. Let us assume that we possess a function f that fills all holes that are entirely contained inside skin areas. Then we modify (2.17) to

$$(A \bullet S)^n(f) = f(A \oplus^n S) \ominus^n S \tag{4.7}$$

where A is in this case our set of skin pixels and S is an arbitrary structuring element. In reality, f can be implemented as follows. We first determine the entire region connected to the origin $(0, 0)$ of the image (the background). The face area, with holes included, is then simply the binary NOT of the background.

In words, we view (4.7) as a function that dilates A n times, eliminating thin segments of non–skin pixels, then fills the holes that still remain, and then performs n erosions. This is then our final skin map B_2 given by

$$\begin{aligned} B_2 &= ([(B_1 \circ S) \bullet S] \bullet S)^n(f) \\ &= f([(B_1 \circ S) \bullet S] \oplus^n S) \ominus^n S \end{aligned} \tag{4.8}$$

and shown in figure 4.8(a). B_2 is now used to extract and crop the face from the original image as in figure 4.8(b).

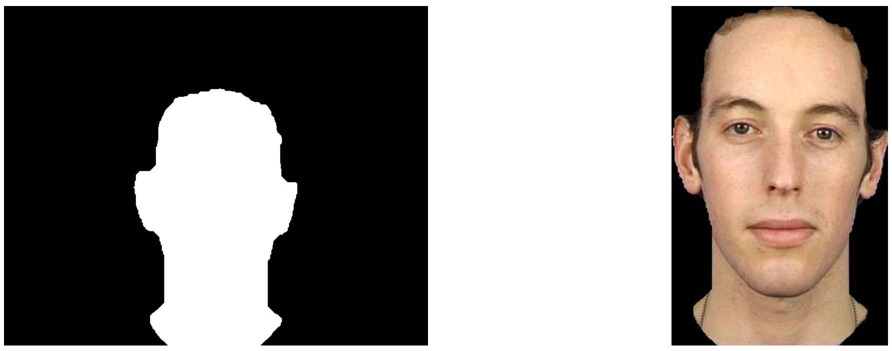

(a) Final skin mask B_2 obtained using (4.8). (b) Face extracted and cropped using B_2.

Figure 4.8: Application of (4.7) and region filling to extract the face region.

Figure 4.9 shows 182 faces randomly selected from the xm2vts database that have been extracted using the method proposed in this chapter.

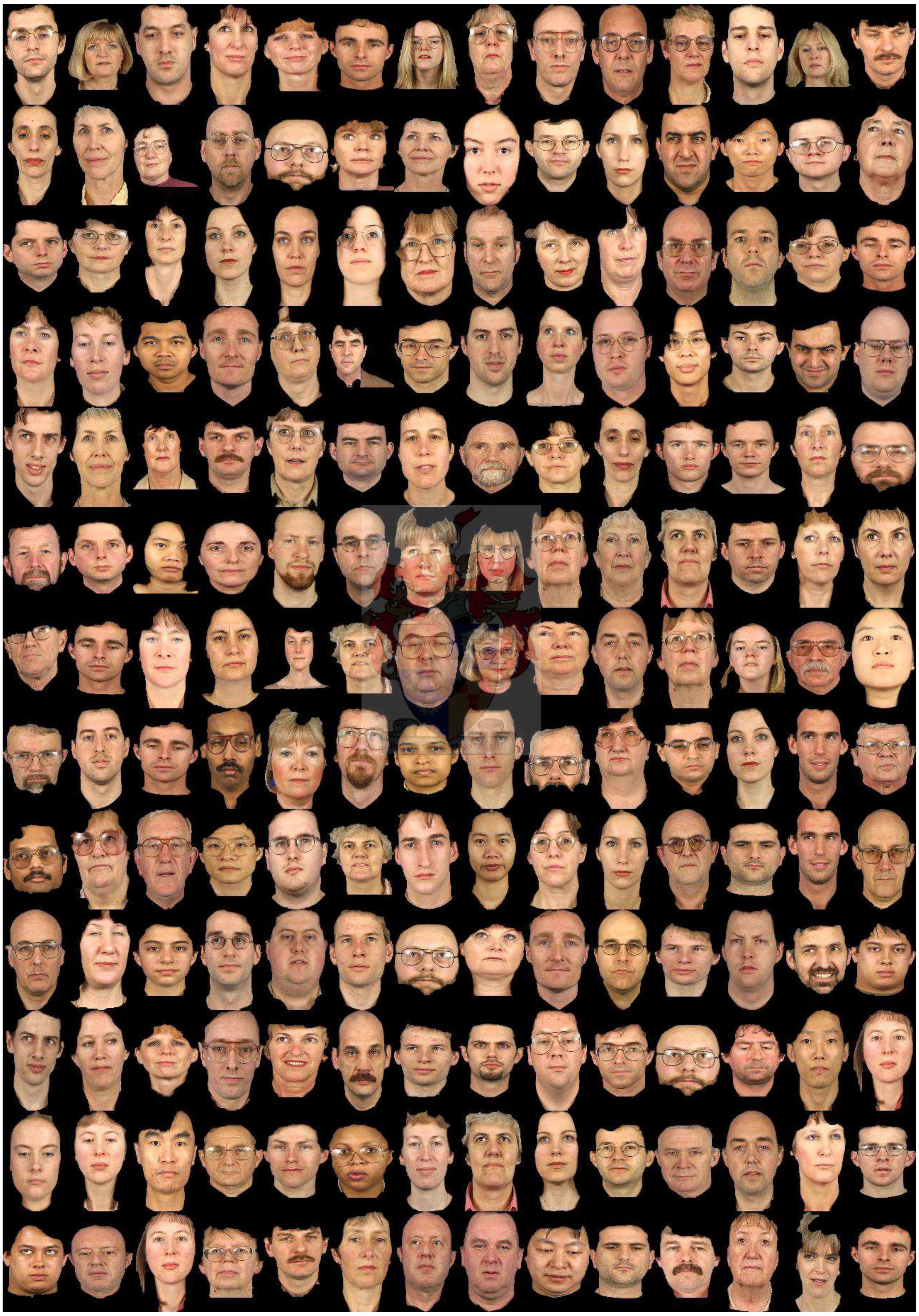
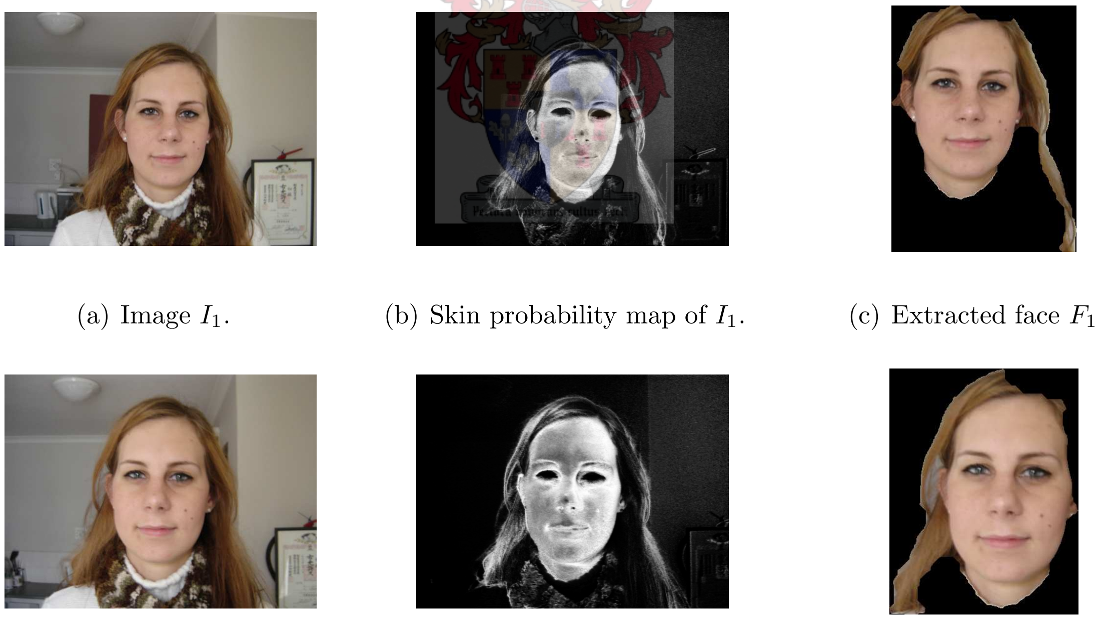


Figure 4.9: A collage of 182 extracted faces.

Chapter 5

Face Normalization

Once we have extracted a facial area from an image we need to normalize it in a way that it's shape and alignment coincides with other extracted faces. This normalization is important for the recognition process, as we want to compare like features. The example given in §4.2 is indeed already quite well aligned but other factors come into play in many real–world cases. For example clothes, hair and parts of the background may be falsely identified as skin colour and calculated to be part of the face. Figure 5.1 shows an example



(a) Image I_1.

(b) Skin probability map of I_1.

(c) Extracted face F_1.

(d) Image I_2.

(e) Skin probability map of I_2.

(f) Extracted face F_2.

Figure 5.1: Poor facial alignment due to false positive skin detection.

of two very similar photos taken of the same individual under the same conditions. However due to variations in the images and the resulting skin probability map, hair has been falsely identified as part of the face resulting in the misalignment of the extracted faces. More severe cases can easily be seen in figure 4.9. The alignment achieved by simply cropping the face from the original image is obviously unacceptable. In the sections that follow we develop a system to remedy this problem. We do however make the assumption that the unaligned face is already near–vertical irrespective of its actual position. The reason for this will become evident in §5.3.2.

5.1 The Human Face

The human face has been studied for centuries by scientists and artists alike. Faces abound with interesting mathematical relationships. The Golden Ratio or Divine Proportion is one example that often occurs in the human face and head. The head, height versus width, forms a golden rectangle with the eyes in the center. The mouth and nose are each to be found at the golden sections of the distance between the eyes and the bottom of the chin. The ear, viewed from the side, forms a Fibonacci spiral, to name but a few examples.

At this point we need to pin down some basic face properties useful to us for alignment. Features such as the eyes, mouth and nose are always found at the same relative positions in the face – the nose is above the mouth and the eyes are above the nose. The most common facial features used in practice are the eyes, nose and mouth. The eyes and nose, for example, form a triangle with *very* similar proportions on all normal human faces. This leads us to the notion that if we can find these features in an image of a face, we can determine and correct its position and orientation as necessary.

Consider figure 5.2 representing the general positioning of features on the human face. Ideally the angles $N\widehat{E_1}E_2$ and $N\widehat{E_2}E_1$ are equal and $\overline{MN}$ is vertical. From a series of 56

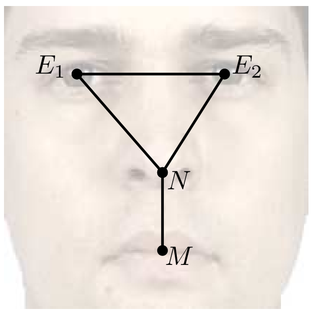


Figure 5.2: Facial feature alignment.

faces (using manual feature detection) we empirically find some average measurements as given in table 5.1. The last entry in the table especially could give an indication of the heads vertical tilt. In practice however, the nose is a much better feature to use as it is rigid and, unlike the mouth, does not change form with facial expression. It would be reasonable to assume the average values in the first three entries of table 5.1 to be 80, 50

Property	Average value
$E_1\widehat{N}E_2$	79.6°
$N\widehat{E_1}E_2$	49.8°
$N\widehat{E_2}E_1$	50.6°
$\frac{\lVert\overline{MN}\rVert}{\lVert\overline{E_1E_2}\rVert}$	0.57
$\frac{\lVert\overline{MN}\rVert}{\lVert\overline{E_1E_2}\rVert} \cdot E_1\widehat{N}E_2$	0.808

Table 5.1: Some average human face measurements.

and 50 degrees. Assuming that the image size is $m \times n$ and that the face is cropped, we empirically find that the eyes are, on average, found at $0.3n$ from the sides and and $0.35m$ from the top. This ideal template can be used in alignment in the manner discussed in §5.2.

5.2 Normalization Methods

To Normalize faces we need to transform the images so that corresponding features line up as accurately as possible between images. A common approach found in the literature is to consider the edges of the face, typically found with Sobel operators described in §2.1.1, and then find the ellipse that best fits this outline. The human face is known to have specific proportions and so the ratio of the ellipse's major and minor axes is chosen as the golden ratio. Factors such as scale and rotation are then considered to find the ellipse that best fits the face in the image.

A problem with the ellipse method is to find a suitable criteria for a "good fit". Assuming we have already extracted the face from the image and calculated the edge, we could count the number of ellipse pixels that correspond to edge pixels. Alternatively we could fit an ellipse where the number of face pixels outside the ellipse equals the number of non–face pixels inside. Whatever criteria we choose, the problem remains that the internal features of the face may not be optimally transformed and could result in sub–optimal recognition results. For this reason we rather focus on detecting and aligning features *within* extracted faces, specifically the eyes, instead of aligning the faces as a whole.

If we were to automatically detect features in an extracted face giving us four independent parameters (for example, the two eyes), we could line them up to predetermined ideal positions. Seeing as we would then have two point correspondences, it would be a matter of calculating an isometric transformation, involving only scale, translation and rotation (see [21]), and then projecting the face so that selected feature positions are idealized. For purposes pertaining to face recognition, we make the assumption that the facial images are frontal. Therefore, even though nose and mouth detection are possible, we only consider eye detection for face alignment and assume that the other features automatically line up accurately enough that their detection is not necessary.

5.3 Eye Detection

When looking at an image, a person can spot the eyes in an image of a face with no effort, but for an untrained computer this task is much more complicated. Two existing methodologies for eye detection are template matching [27], and use of the actual colour information in the image [24] to construct so–called eyemaps. Template matching is the more intuitive of the two and amounts to 2–D spatial filtering with an eye–shaped structuring element, called a template. A high response is naturally expected around the eye areas. However, this method suffers from the obvious drawback that the eye templates must be correctly scaled and orientated. The scale factor can be partially remedied by scaling the extracted face down to a fixed, predefined size but the rotation would still be a problem.

The eyemap alternative uses colour information directly from the image without using any templates. An eyemap is an image of the same size as the original that tells us where the eyes are most probably located. To directly locate the eyes using eyemaps, we construct and combine four separate eyemaps based on different properties of the eye area in images. More specifically we use luminance and chrominance information from the YC_rC_b colourspace, as well as the information of the original RGB colourspace and it's gradient as discussed in 2.1.1.

5.3.1 Constructing the Eyemap

Consider the extracted face in figure 5.3. Figures 5.4 and 5.5 give the separated components for the RGB and YC_rC_b colourspaces respectively. In these figures we can see the R, G, B, Y, C_r and C_b components individually and use them to determine consistent eye area characteristics that can be used to construct the eyemaps.

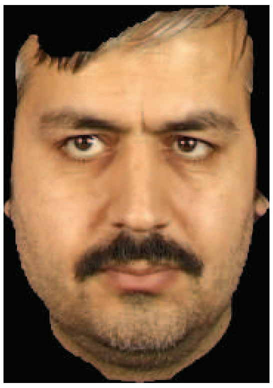
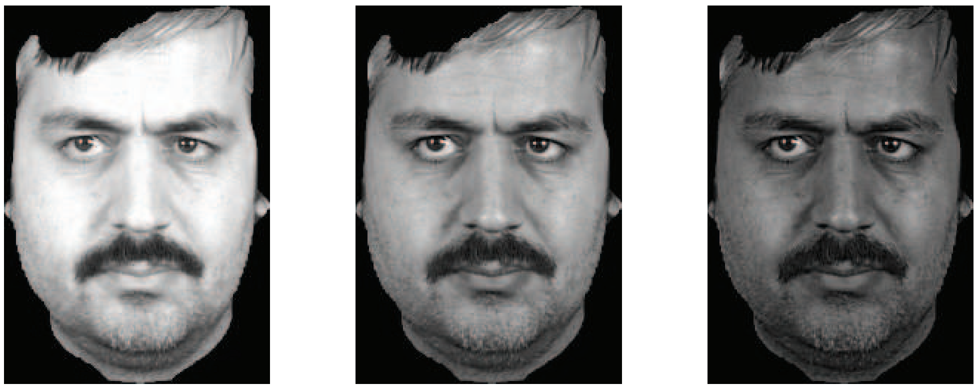
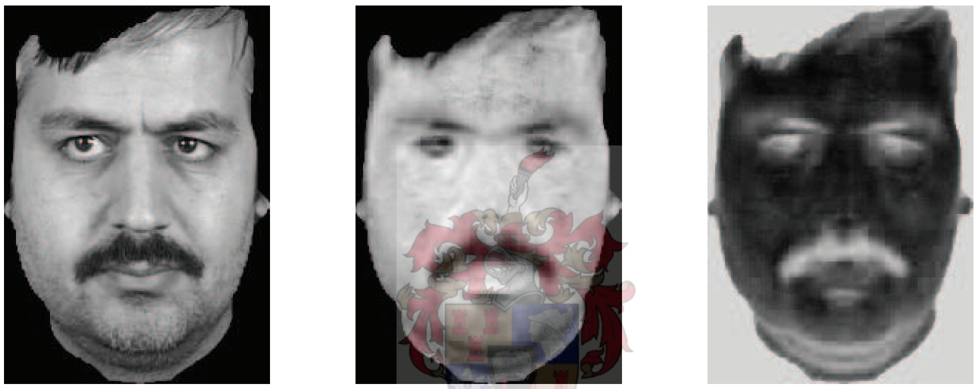


Figure 5.3: Original face in full RGB colour.



(a) Red (b) Green (c) Blue

Figure 5.4: Separation of red, green and blue components.



(a) Y (luminance) (b) C_r (red chrominance) (c) C_b (blue chrominance)

Figure 5.5: Separation of luminance and chrominance components.

Chrominance Eyemap

Referring to figure 5.5 we observe that the eyes have a relatively high response in the C_b colour subspace while their response in the C_r subspace is generally lower. This was found to be true in general by testing several facial images and the average face discussed in §6.1. We calculate the chrominance eyemap by

$$E_{chrom} = C_b^2 + \overline{C}_r^2 + \frac{C_b}{C_r} \tag{5.1}$$

where $\overline{C}_r$ is the inverse of C_r (i.e. $255 - C_r$). Pixels where $C_r = 0$ are not calculated and set to the maximum of the remaining output. The resulting eyemap, E_{chrom}, is then normalized to fall inside the range $[0, 255]$.

Luminance Eyemap

We note that in the luminance component of figure 5.5, the eye areas have both dark an light pixels. We can thus design morphological operations to accentuate these areas. Referring to §2.1.2 and §2.1.3, we can use erosion to emphasize dark areas in the image, and dilation to emphasize light areas. We calculate the luminance eyemap by

$$E_{lum} = \frac{Y \oplus A}{Y \ominus A + 255} \tag{5.2}$$

where A is a circular structuring element of diameter roughly an eighteenth of image width. E_{lum} is again normalized to $[0, 255]$.

Edge Eyemap

Because eye areas contain light and dark areas and are generally much smaller than the entire image, there are sharp transitions between light and dark pixels in these areas. This can be seen in all three RGB components. We can therefore incorporate edge detection from §2.1.1 to aid us in eye detection. First we calculate the edges in the entire image by filtering the average of the R, G and B components with Sobel operators. The obtained edge image is then smoothed by filtering it with an averaging filter[1]. This can be written as

$$E_{edge} = P\left(\left|P\left(\frac{(R+G+B)}{3}, S_h\right)\right| + \left|P\left(\frac{(R+G+B)}{3}, S_v\right)\right|, A\right) \tag{5.3}$$

where $P(X, A)$ denotes spatial filtering of X by A, S_h and S_v are the Sobel operators for horizontal and vertical edge detection respectively, and A is the averaging filter.

RGB Eyemap

A fourth eyemap is constructed by noting that the eyes always have dark areas in all three the R, G and B components. The RGB eyemap is calculated by multiplying the inverses of the separated R, G and B components and filtering the result with the same averaging filter as before. We write this as

$$E_{RGB} = P\left(\left(\overline{R} \times \overline{G} \times \overline{B}\right), A\right). \tag{5.4}$$

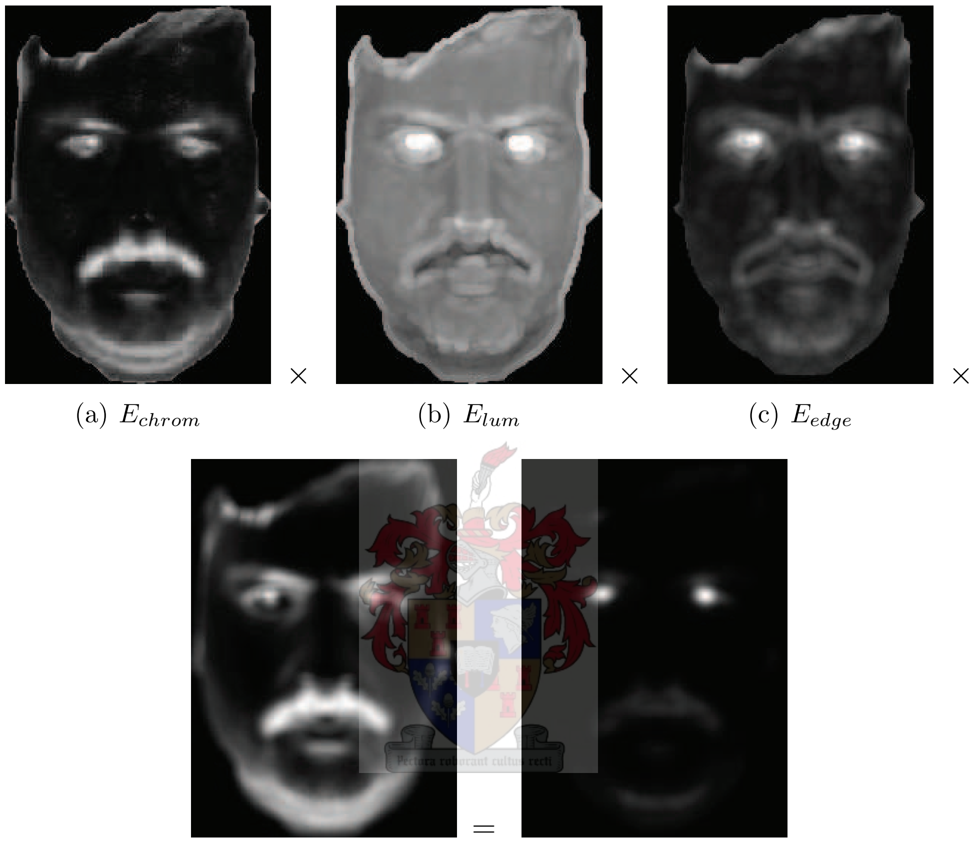
Final Eyemap

A single eyemap cannot reliably predict the position of the eyes. The more eyemaps are constructed, the more reliable the estimate will be and it is for this reason that four are constructed. These four eyemaps are combined by multiplication to yield the final eyemap,

$$E_{map} = E_{chrom} \times E_{lum} \times E_{edge} \times E_{RGB}. \tag{5.5}$$

[1]A filter in which the filter coefficients sum to one.

Multiplication is used rather than addition as it is more effective at suppressing false positives from an individual eyemap. However, multiplication also accentuates false negatives but these are assumed from empirical tests to be much fewer than false positives. The separate eyemaps and their product can be seen in figure 5.6. More eyemaps can easily be constructed from other colour properties such as those described in [23].



(a) E_{chrom} (b) E_{lum} (c) E_{edge}

(d) E_{RGB} (e) E_{map}

Figure 5.6: Combination of separate eyemaps.

5.3.2 Determining Eye Positions

At this point we assume that the extracted face from which the eyemap in figure 5.6(e) is derived is vertical to about 30°. Using the eyemap to further refine the faces' alignment now involves an initial rotation determined by the Radon transform [16] of the eyemap and determining the strongest vertical axis of symmetry.

The Radon transform is essentially the projection of an image intensity along a radial line oriented at specific angles. Typically these angels span from 0° to 180°. Radon transforms are the two dimensional equivalent of the more general Hough transforms which are the underlying mathematics of CAT–scans in medical science [40]. This discussion will not consider the mathematics behind Radon transforms, a detailed discussion can be found in [16].

The initial rotation aided by the Radon transform is followed by a calculation of a measure of reflective symmetry of the eyemap. Several methods exist to determining the strongest axis of reflective symmetry, such as the reflective symmetry transform described in [45]. It must be noted that these methods are often complicated or lacking in efficient running time with that of the reflective symmetry transform being of order $O(n^3)$ with n the number of pixels in the image.

Most symmetry detection algorithms are focused on determining whether a shape is symmetrical under rotation or reflection. If not, a *distance* from perfect symmetry is calculated indicating how symmetrical the shape is [61]. The method used in [45] calculates a measure of symmetry along all possible lines through a 2–D image (or planes in N–D volumes), and not only for those through the center of mass of the shape, as is usually the norm. When searching for symmetry, if we have somehow determined the orientation of the symmetry axis, we can drastically reduce this running time by only searching for symmetry in one dimension. Henceforth we will refer to *reflective symmetry* simply as *symmetry*.

The study of symmetries and their detection can be quite involved. For our purpose, we would need to find a method to detect symmetries that is computationally inexpensive but still useful in the context of eye detection. We can assume that the initial rotation of the face (and eyemap) determined by aid of the Radon transform will have rotated the face to a near–vertical position. As a result the eyes will have very similar vertical positions in the image which significantly simplifies symmetry detection in the eyemap. The method proposed below, although relatively cheap computationally, appears to work well if this assumption is true.

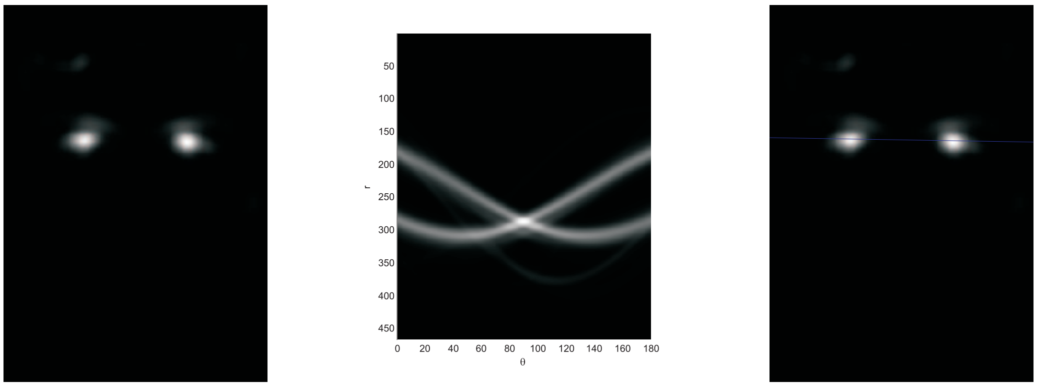
An overview of the procedure for finding the eyes in the eyemap is as follows:

1. Apply an initial rotation of the face and eyemap by aid of the Radon transform.
2. Find an initial estimate for the vertical position of the eyes.
3. Find an initial estimate for the horizontal position of the eyes based on the symmetry of the eyemap.
4. Refine the initial estimate with a gradient step algorithm.

Step 1: Initial rotation of the face

To rotate the face to an initial near–vertical position, the first step is to eliminate the areas of the eyemap where we will never expect to find eyes. Consider the eyemap E_{map}

from figure 5.6(e) as an $m \times n$ matrix. From empirical tests it can confidently be assumed safe to entirely eliminate the bottom $5/9^{\text{th}}$ as well as the top $1/10^{\text{th}}$ of the eyemap as an initial reduction shown in figure 5.7(a). Following this, we calculate the Radon transform of the result as shown in figure 5.7(b). Due to the nature of the radon transform, the (r, θ) coordinate with the largest response is likely to correspond to the line that goes through the center of both eyes in the eyemap as shown in figure 5.7(c). The θ coordinate of the



(a) Initially reduced eyemap.

(b) Radon transform of the eyemap.

(c) Line corresponding to the maximum response in the Radon transform.

Figure 5.7: Determining an initial rotation for the extracted face using the Radon transform.

maximum value in the Radon transform is obtained and used to rotate the face and eyemap to an estimated vertical position. In the example in figure 5.7, an initial rotation of only $1°$ is necessary.

Step 2: Estimating vertical position

Let us consider the eyemap E_{map} as a $m \times n$ matrix. After step 1 we assume the face is close to vertically aligned and the idea now is to examine the sum of the rows of E_{map} to give us a clue as to the vertical position of the eyes. Let M_i be the sum of row i of the eyemap, $M_i = \sum_{j=1}^{n} E_{map}(i, j)$. The position of the maximum value in M is then

$$m^* = \max_i M_i. \tag{5.6}$$

We now further reduce the area of the eyemap under consideration by only considering rows $m^* - \frac{m}{10} \leq r \leq m^* + \frac{m}{10}$. We are left with a small vertical interval in which we assume the eyes are located, however m^* is still our preliminary estimate. Figure 5.8 depicts this process graphically.

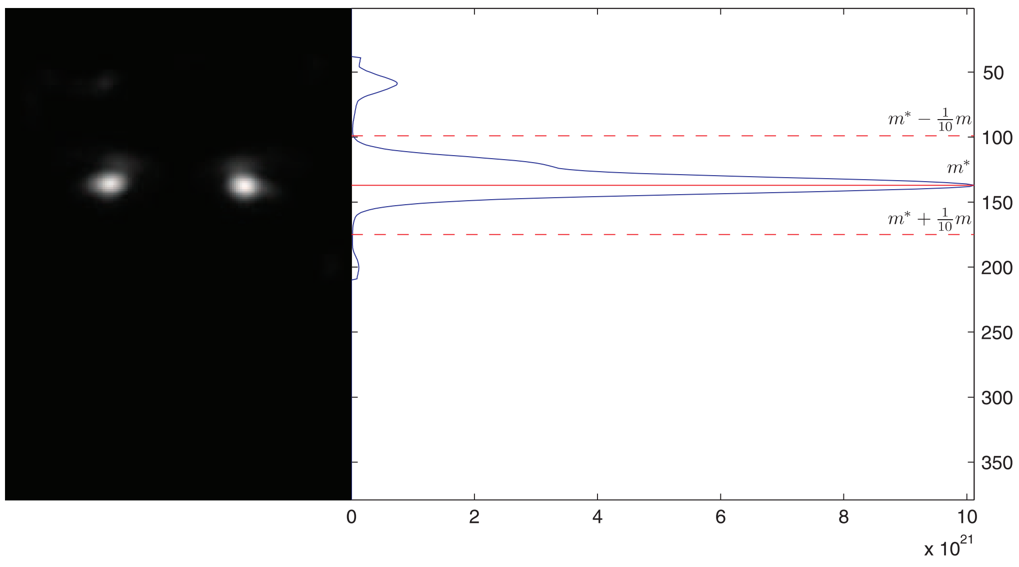


Figure 5.8: Finding a preliminary vertical interval for eye location.

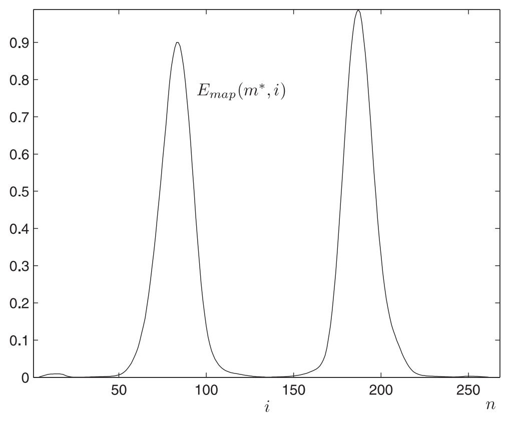
Step 3: Estimating horizontal position

As noted, symmetry detection is greatly simplified once we know the orientation of the axis of symmetry, at this point assumed to be vertical. So far, our best guess for the vertical position of the eyes is m^*. A plot of $E_{map}(m^*, j)$, $j = 1 \ldots n$ is given in figure 5.9(a) with values normalized to [0,1]. One can clearly see the peeks corresponding to the two eyes. Our horizontal estimates will be at the cusps of these peaks.

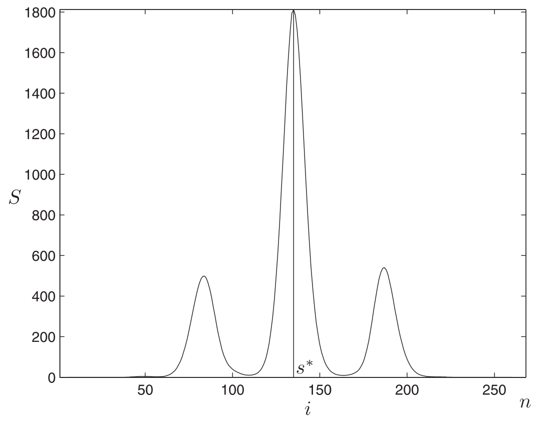
The example in figure 5.6 has a particularly well behaved eyemap in the sense that the peaks with the maximum values indeed correspond to the eyes. In many situations this is not the case, especially for individuals wearing eyeglasses. Automatic eyeglass removal in images is possible [57] but is not discussed or implemented here. An example of a more tricky eyemap is given in figure 5.10. Refereing to this eyemap, it is easy to see that we want to find the two inner peaks and not those corresponding the the frame of the glasses. To do this we search horizontally outwards from the axis of symmetry, on the row m^*.

Reminding the reader that the eyemap is an $m \times n$ matrix, we now obtain a measure of symmetry of the row m^* of the eyemap. We first let $\alpha_i = \min(i, n - i)$, $\forall\, i \in [1, n]$, be a weight giving more importance to axes close to the center of the image. We calculate the symmetry function $S(i)$ by

$$S(i) = \alpha_i \sum_{k=1}^{\alpha_i} \left(E_{map}(m^*, i - k) \cdot E_{map}(m^*, i + k)\right), \; i \neq 1, n. \tag{5.7}$$



(a) Row m^* of E_{map}.



(b) Symmetry function $S(i)$.

Figure 5.9: The row $E_{map}(m^*, j)$ from the eyemap in figure 5.8 and its measure of symmetry.

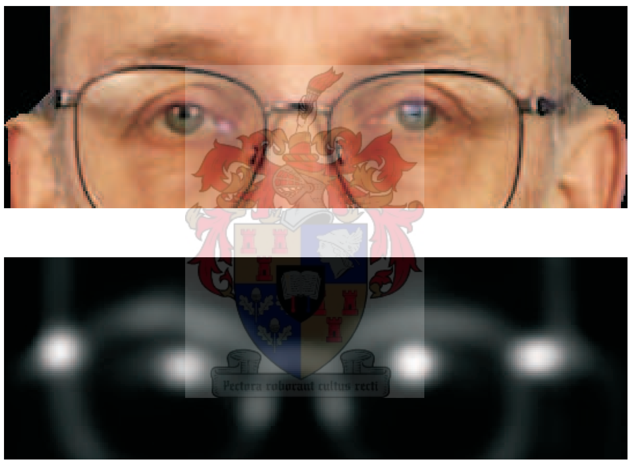


Figure 5.10: An example of a potentially problematic eyemap.

A plot of $S(i)$ is given in figure 5.9(b) where the position of the maximum value is clearly the axis of symmetry of the eyemap. Let this position be s^*.

Let $M = \max(E_{map}(m^*, j))$, $j \in [1, n]$ then starting at s^* we move left (or right) remembering the position, n^*, of the maximum value encountered thus far. We assume that we are moving within a peak if the value at the current position is greater than $0.35 \times M$. If we continue moving and the current value drops by more than $3/5^{\text{th}}$'s of the maximum encountered value, we assume to have passed the peak and consider n^* as our horizontal estimate. This procedure is carried out twice, once to the left and once to the right of the axis of symmetry. We now have preliminary estimates for the eye positions, given by $\mathbf{e}_1 = [m^*,\ n_1^*]$ and $\mathbf{e}_2 = [m^*,\ n_2^*]$.

Step 4: Refining the initial estimate

At this point we have a vertical and horizontal estimates for both eyes. To refine the estimate we can now apply a gradient search algorithm to find a local maximum in the eyemap using our initial estimate as starting points. That is we change the estimate, stepping in the direction of the maximum gradient until the values of all neighbouring points are smaller or equal to the value at the current position. Considering this stepping method, it is often useful to smooth the final eyemap, E_{map}, with a smoothing filter one last time to eliminate small irregularities and spikes near the local maximum. More on such optimization methods for continuous as well a discrete functions can be found in [36, p. 338-344].

Once we have detected the eye positions, it is a simple matter to rotate, scale and translate the image using a similarity transformation [21] so that the two eyes are moved to fixed positions as determined in §5.1. Nose and/or mouth detection could be implemented in a similar fashion as eye detection and incorporated in alignment to extend this to a projective transformation as mentioned earlier. However, consider the features triangle discussed in §5.1. We have assumed that the face in the image is facing directly towards the camera. The variations then in the triangle proportions due to (i) inaccurate feature detection, (ii) slight head pose differences and (iii) natural face differences, are all of the same order. Unless extremely accurate eye, mouth and nose detection is possible, calculating the triangle gives us little to no information useful in face alignment. A complete eye detection test was run on the xm2vts database of facial images, the complete results of which are given in appendix C. Using the method described in §5.3, 91.3% of eyes were detected correctly. The failures are due to eyemaps with eye responses that are simply too weak. This in turn can be due to either extreme head pose (tilt, closed eyes, etc.) or the presence of other factors in the image affecting the construction of the individual eyemaps (with eyeglasses and facial hair being the most common).

Figure 5.11 shows 182 extracted faces normalized by the method described in this chapter. This should be compared to figure 4.9. After the faces have been normalized, a much better internal feature correspondence is clearly visible between all the images. The entire xm2vts database was processed and the faces extracted and normlized in the manner described. This edited database is electronically provided to the University of Stellenbosch in tandem with this thesis. Take note that in some extreme cases the faces of certain users were incorrectly or poorly extracted. However, the successful cases greatly outnumber these failures and provide a large enough database of normalized faces for future research.

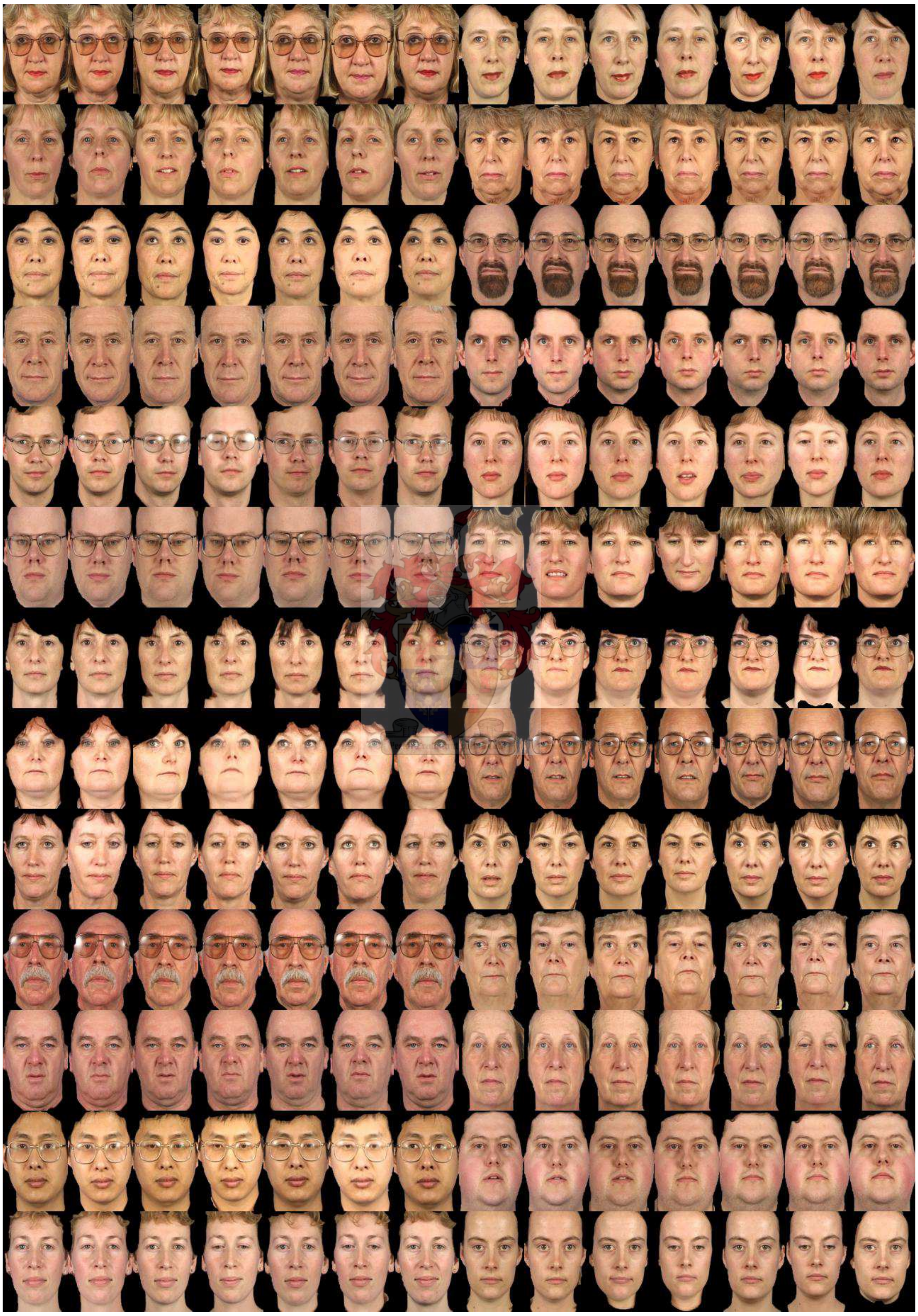


Figure 5.11: A collage of 182 extracted and aligned faces.

Part II

Face Recognition

Chapter 6

Face Recognition Using SVD

For actual recognition of faces we shift our focus away from the image processing of the previous chapters to linear algebra. We specifically consider singular value decomposition (SVD) as it provides us with a way of comparing the positions of faces in a so–called *face space*. A more detailed discussion on the SVD and its properties is given in appendix D.

The process of face recognition using SVD is relatively simple. We consider the images we work with as vectors in a very high dimensional image space and make the assumption that facial images span a lower dimensional subspace of this image space. Our first goal is to find an orthonormal basis for this subspace. The goal of face recognition can then be achieved by comparing different faces's projections onto this basis.

6.1 Eigenfaces and Face Recognition

Using SVD as our primary tool, we have what is needed to begin construction of a working face recognition system. Let us assume we have a set of k facial images. These images are known as the *face training set* and should be a good representation, containing enough variation, of all faces. We proceed to process these images by extracting and normalizing the faces as discussed in chapters 4 and 5. At this point the images may still be of different sizes so we need to resize them to a constant size, say $p \times q$, whilst making sure that the eye alignment property is not lost. These images are then rearranged to $pq \times 1$ vectors $\mathbf{x}_j$. Each face is now considered as a vector in a pq dimensional face space. Their mean is calculated as

$$\mathbf{a} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{x}_j \tag{6.1}$$

and is known as the *average face*. The average face of 265 facial images, all from different individuals in the xm2vts database, can be seen in figure 6.1. Note the detail achieved due to the alignment method using eye detection as discussed in §5.3.

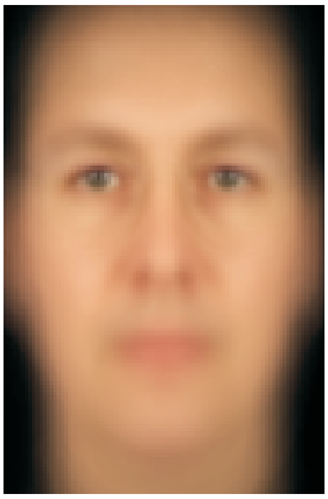


Figure 6.1: The average face.

We do not work with the faces $\mathbf{x}_j$ directly but rather with their deviation from the mean. That is we use the average face to produce the vectors

$$\mathbf{f}_j = \mathbf{x}_j - \mathbf{a} \tag{6.2}$$

which have zero mean and are assembled into the columns of a matrix F so that

$$F = \left[\begin{array}{ccc} \mathbf{f}_1 & \cdots & \mathbf{f}_k \end{array}\right]. \tag{6.3}$$

These vectors can be likened to the data points in figure D.3 only in much higher dimension. Note that even for images of size 100×100, which is considered small by today's standards, every $\mathbf{f}_j$ is a vector with 10000 entries. As the matrix F is of size $pq \times k$, it can, and indeed must, be assumed that $k \ll pq$.

6.1.1 Eigenfaces

Once we have constructed the matrix F, we proceed to calculate its (reduced) SVD such that

$$F = U\Sigma V^T. \tag{6.4}$$

The columns of U are known as the *eigenfaces* and can be interpreted in the way discussed in appendix D.3, i.e. as directions of maximum deviations. The singular values on the main diagonal of Σ indicate the magnitudes of these deviations.

As the facial images are in 3–channel colour, it would be possible to reshape the images to $3pq \times 1$ vectors that would make up the matrix F (see §7.2.4 for a conclusion on this approach). Here we note that the number of calculations required to compute the SVD of a matrix grows cubically as the size of the matrix increases. From a computational point of view it would therefore be cheaper to calculate the SVDs of three $pq \times k$ matrices than one $3pq \times k$ matrix. It is for this reason that the SVD is calculated for each of the R, G and B colour channels, i.e.

$$\begin{aligned} F_R &= U_R \Sigma_R V_R^T \\ F_G &= U_G \Sigma_G V_G^T \\ F_B &= U_B \Sigma_B V_B^T \end{aligned}$$

although we shall only refer to the variables as in (6.4) unless we require information from specific colour channels.

Figure 6.2 shows some of the different eigenfaces. Note that these images are actually

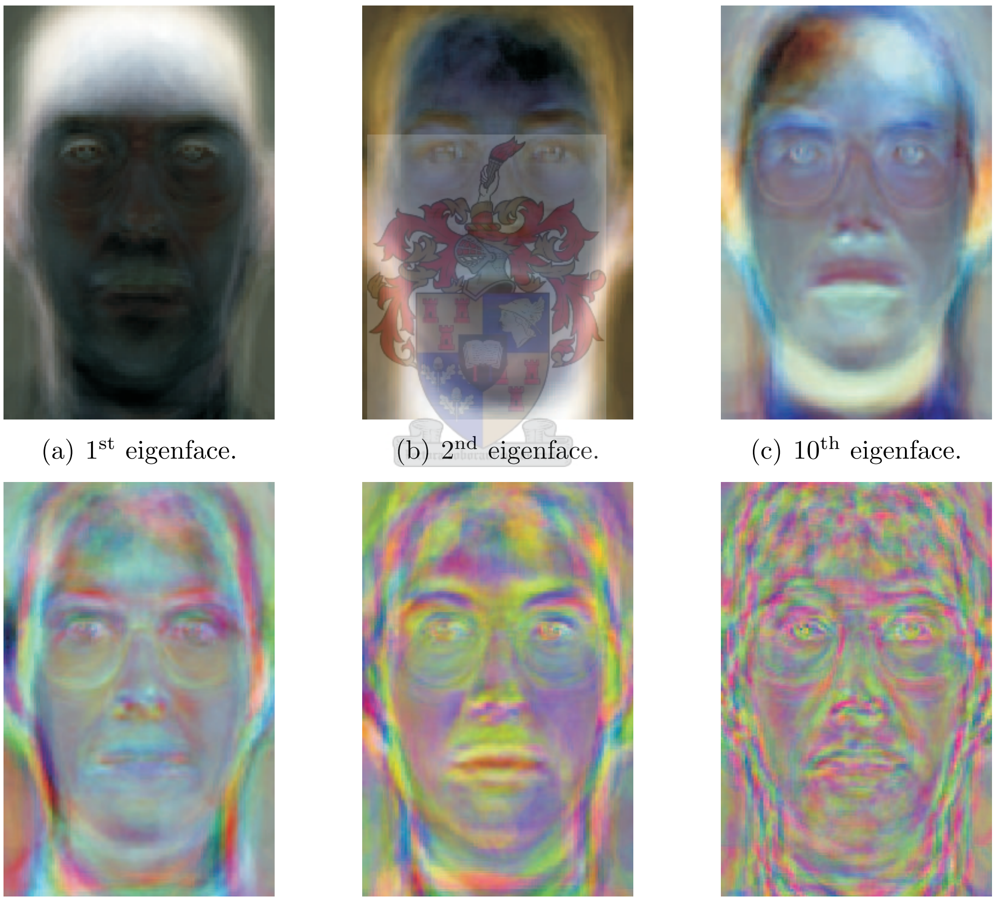

(a) 1st eigenface. (b) 2nd eigenface. (c) 10th eigenface.
(d) 30th eigenface. (e) 50th eigenface. (f) 150th eigenface.

Figure 6.2: Some of the eigenfaces.

the red, green and blue channels's eigenfaces combined. The eigenfaces associated with the larger singular values contain only very general information of the total variation of the faces, while those associated with smaller singular values contain more detail. Recall that the eigenfaces are perpendicular to one another and therefore linearly independent. As shown in appendix D.1, any of the faces in F can be written as a linear combination of these eigenfaces. Furthermore, as we have calculated the reduced SVD, U is of size $pq \times k$, the same size as F. Thus

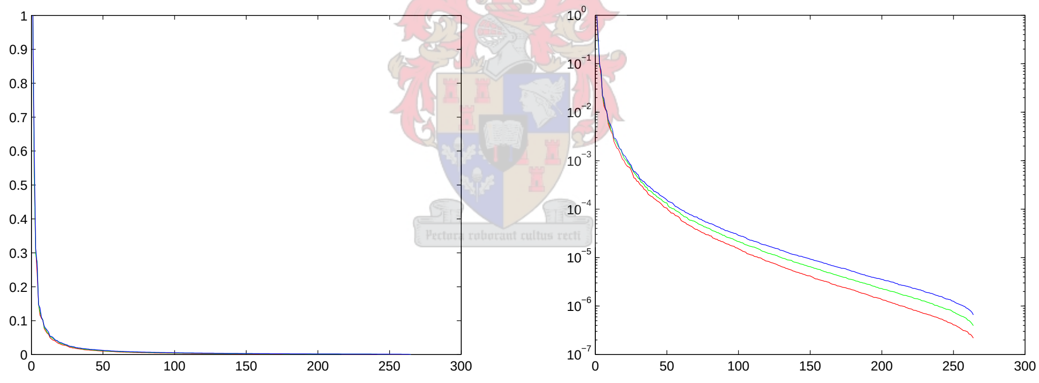
$$U = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_k \end{bmatrix} \tag{6.5}$$

and

$$\Sigma = \operatorname{diag}(\sigma_1 \ldots \sigma_k). \tag{6.6}$$

6.1.2 Omission of Eigenfaces

As we saw, the U referred to in §6.1.1 is in fact the reduced U with k columns. We know that any vector (face) in F can be reconstructed by a linear combination of the vectors (eigenfaces) in U. It is however possible to use even fewer eigenfaces for this reconstruction and may even become necessary if the training set of faces becomes large, implying a very large U matrix. To delve into this matter we consider the singular values of F, i.e. the values on the main diagonal of Σ. The normalized singular values of F are given in figure 6.3. The important thing to note is how fast the singular values decrease. As the



(a) Singular values of F. (b) Singular values of F on a log scale.

Figure 6.3: Singular values of the training set F.

singular values correspond to eigenfaces, it stands to reason that we can omit some of the eigenfaces associated with the smallest singular values and still be left with a relatively accurate reconstruction of a face in F. In other words we can use U_ω where

$$U_\omega = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_\omega \end{bmatrix},\ \omega \leq k. \tag{6.7}$$

Using fewer eigenfaces to reconstruct a face in F would thus lead to an imperfect reconstruction (approximation) of that particular face. Let us consider as an example the reconstruction of a face shown in figure 6.4(a) that is in F. We know that if we reconstruct such a face by a linear conbination of *all* the eigenfaces, the reconstruction is perfect. Using fewer eigenfaces, i.e. using U_ω with $\omega < k$, we obtain the results as illustrated in figure 6.4. It can be shown that the error between the original face $\mathbf{x}$ and the reconstruction $\mathbf{r}_\omega$

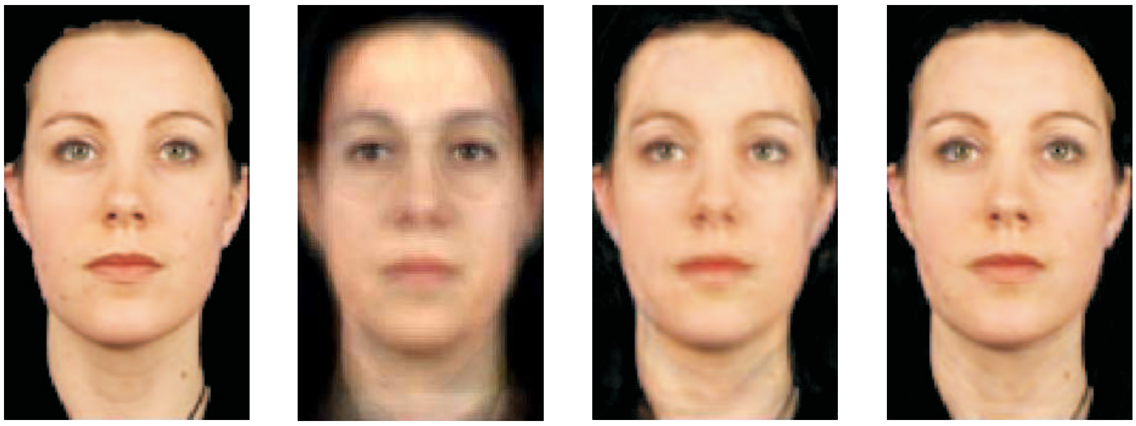

(a) Original face in training set. (b) Reconstruction using 10 eigenfaces, (U_{10}). (c) Reconstruction using 50 eigenfaces, (U_{50}). (d) Reconstruction using 120 eigenfaces, (U_{120}).

Figure 6.4: Reconstructions of a face in the training set using 10, 50 and 120 eigenfaces.

using ω eigienfaces is such that

$$\delta_\omega = \|\mathbf{x} - \mathbf{r}_\omega\|_2 < \sigma_{\omega+1}. \tag{6.8}$$

This inequality would not be valid if $\mathbf{x}$ was not in F.

We see that it is thus not necessary to retain all k eigenfaces after the computation of the SVD. If we are reconstructing a face not in F, the error also decreases as more eigenfaces are used, but will not reach zero. Studying this error to see when it is of the same order of magnitude as the error obtained when omitting eigenfaces and reconstructing a face indeed in F, can be further investigated. This information can be used to tell us how many eigenfaces we can omit without loss of accuracy in the system. Another influential factor on how many eigenfaces to retain is the purpose of the recognition system and the need for accuracy. The topic of omitting eigenfaces is discussed in some more detail in §6.3.2.

6.1.3 Projection and Reconstruction Using Eigenfaces

Suppose that we have calculated a set of eigenfaces and have omitted those associated with the smallest singular values. This gives us the matrix U_ω. Now we are given an extracted test face T (of size $p \times q$) to be compared to other known faces. We first rearrange this image to a $pq \times 1$ vector and subtract the average face to obtain $\mathbf{t}$, i.e.

$$\mathbf{t} = \mathbf{x}_{\text{test}} - \mathbf{a}. \tag{6.9}$$

This is done in order to get T in the same format as the $\mathbf{f}_j$'s earlier. The new vector $\mathbf{t}$ is also a face[1] but does not necessarily lie in the column space of U_ω. We thus orthogonally project it onto this space and obtain a coefficient vector

$$\mathbf{c}_\omega = U_\omega^T \mathbf{t}. \tag{6.10}$$

From here we can calculate a reconstruction of the face using only the eigenfaces in U_ω. This best possible reconstruction is given by

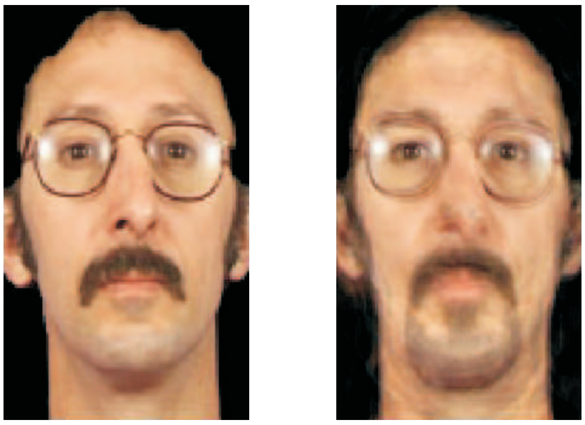
$$\mathbf{r}_\omega = U_\omega \mathbf{c}_\omega + \mathbf{a} \tag{6.11}$$

which can now be rearranged to a $p \times q$ image. For purposes of face recognition it is not necessary to calculate the reconstruction although we will shortly discuss it here as it provides further insight.

Suppose the test face $\mathbf{t}$ we consider is in F. Then it is in the column space of U (but not necessarily in that of U_ω) and can thus be perfectly reconstructed by a linear combination of U's columns (where the entries of $\mathbf{c}$ in (6.10) are the coefficients). If $\mathbf{t}$ is not in F, but there are several faces (vectors) in it that are very similar to $\mathbf{t}$ (i.e. other images of that particular individual), then the reconstruction is exceptionally accurate as the information of the original face that is lost in the projection is very little. This is shown in figure 6.5. If there are no similar faces in F, the reconstruction is slightly poorer although still resembles a face due to the eigenfaces and is more that adequate for recognition from a mathematical point of view. This case is depicted in figure 6.6. If $\mathbf{t}$ is not a face to begin with, the reconstruction is completely inaccurate and hardly recognizable as seen in figure 6.7. As the original image is an example of a non–face object in the image space under consideration, it is not possible to accurately reconstruct it using eigenfaces. The reconstruction still vaguely resembles a face and has recognizable facial features such as eyes, nose and mouth. The reason for this is that it is forced to take this general form as it is a linear combination of eigenfaces, each of which generally contain these features as seen in figure 6.2.

In practice, the individual in the test image will not have been used in the training set to calculate the eigenfaces. The normal case would therefore be that of figure 6.6. This particular reconstruction is poor compared to reconstructions found in literature using fewer eigenfaces. This can primarily be attributed to the fact that the variance in the shape of the cutouts has resulted in sub–optimal eigenfaces. A method to discount face areas further away from the center may well solve this problem.

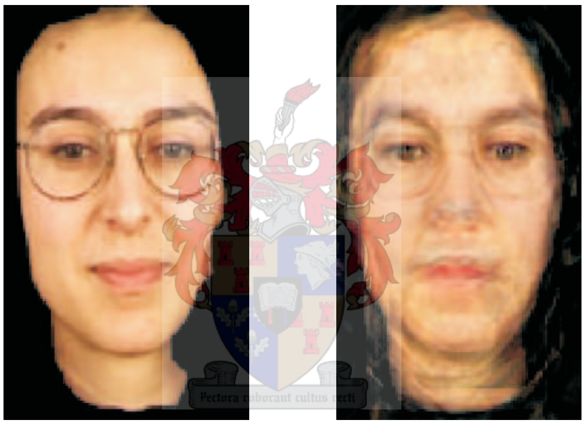
[1]Technically, $\mathbf{t}$ is a deviation from the average face like every $\mathbf{f}_j$.



(a) Test face. Similar faces in training set.

(b) Reconstructed image.

Figure 6.5: Reconstruction of a test face with very similar faces in the training set.



(a) Test face. No similar faces in training set.

(b) Reconstructed image.

Figure 6.6: Reconstruction of a test face with no similar faces in the training set.

6.1.4 Recognition

Let us again suppose we are using ω eigenfaces for recognition. The actual comparison of the test face with other faces is done by comparing their projection coefficients $\mathbf{c}_\omega$. Similar faces' projections on the column space of U_ω should yield similar $\mathbf{c}_\omega$. Let

$$\mathbf{c}_\mathbf{t} = U_\omega^T \mathbf{t} \tag{6.12}$$

be the coefficients of the projection of the test face on the column space of U_ω. Similarly we let

$$\mathbf{c}_{\mathbf{f}_j} = U_\omega^T \mathbf{f}_j,\ j = 1 \dots k \tag{6.13}$$

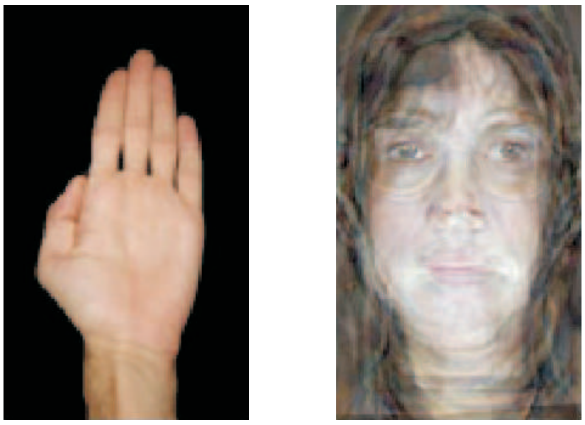

(a) Not a face. Nothing similar in training set.
(b) Reconstructed image.

Figure 6.7: Reconstruction of a non–face object using eigenfaces.

be the projection coefficients of every face $\mathbf{f}_j$ already entered into the system and therefore subject to comparison with $\mathbf{t}$. There are several ways we can now proceed to calculate the difference between $\mathbf{c_t}$ and $\mathbf{c}_{\mathbf{f}_j}$. We use the L_2–norm as in (D.6). That is we calculate

$$\epsilon_j = \|\mathbf{c_t} - \mathbf{c}_{\mathbf{f}_j}\|_2 \tag{6.14}$$

for every face $\mathbf{f}_j$. After we have calculated $\epsilon_j,\ j = 1 \ldots k$ the best match to the test face is the database face yielding the smallest ϵ_j.

6.2 Identification and Verification

Identification and verification are two terms often interchanged or taken to be synonymous. They are in fact fundamentally different and independent from the underlying recognition methods used, being in this case, eigenfaces. To shortly distinguish between the two cases, an *identification* system entails finding the best match to a test face among known faces, whereas a *verification* system must decide whether a face belongs to a particular user given an identity claim. In a system where no identity claim is given it may not be enough to only identify the best match to the test face. This match could indeed be used as an identity claim (only if two or more images of that user are enrolled in the system, as explained in §6.3.3) that a verifier could use to output a confidence value, thereby determining whether the test face is a good enough match or not. Verifiers have more information to work with can therefore become quite complex. We will not focus on the technical details of the workings of the different verifiers, but rather on their place and purpose in a face recognition system.

6.2.1 Verifiers

The purpose of a verifier is to output a single boolean “accept” or “reject” answer by subjecting a calculated confidence value to a threshold. Verifiers can fail in the following two ways:

- **False Acceptance (FA):** Mistaking an imposter for a genuine user.
- **False Rejection (FR):** Mistaking a genuine user for an imposter.

The term “imposter” refers to all faces except those belonging to the claimed identity. Imposters are usually faces of users also enrolled in the system and are used to determine the score distribution of imposters.

The simplest of verifiers is known as the *ranking verifier* and also has the most intuitive operation. Consider the ϵ_j’s of (6.14) as scores of *all* the faces enroled in the system, imposters or otherwise. The score of the claimed identity is then compared to that of all imposter faces and the confidence value is returned as the number of imposters with scores worse than that the claim, divided by the total number of imposters. The confidence, therefore, can never reach 1 but approaches it as the number of imposters increase.

The ranking verifier is known to produce slightly poorer results than its more sophisticated counterparts. An improvement on the ranking verifier, known as the continuous ranking verifier, is to interpolate the confidence value from the imposter scores ranked just above and just below the score of the claim. Other more complicated verifiers are possible, including the T-Norm and and C-Norm verifiers. These two variations both use the imposter scores to approximate a Gaussian PDF which in turn is used to calculate the confidence value using this Gaussians cumulative distribution function. More on these verifiers can be found in [49].

6.3 Experimental Results

In the following sections we briefly discuss the experimental results pertaining to identification and verification. Different methodologies are followed for the experiments as they seem fit and the ranking verifier is used throughout. Generally the statistics for recognition are poor while those for verification are good, indicating the importance for the latter.

6.3.1 Example 1: Single User Identification

The xm2vts database contains 295 individuals with 8 images of each. For this example, the training set of faces used to calculate the eigenfaces is taken as the first image of the first 265 users and all the eigenfaces are retained. A dummy login system is then constructed with 18 different individuals (mostly from the xm2vts database) enrolled as users of this system. Seven images of each of these users are supplied to the system as training images (thus a total of 126 training images), while the remaining one is used as a test image. Given one of the test images, we calculate (6.14) for each of the 126 faces enrolled in the system and then plot the ϵ_j's as shown in figure 6.8. The minimum ϵ_j is clearly indicated at $j = 105$ corresponding to the interval of user 016 which is therefore the best match to the given test face.

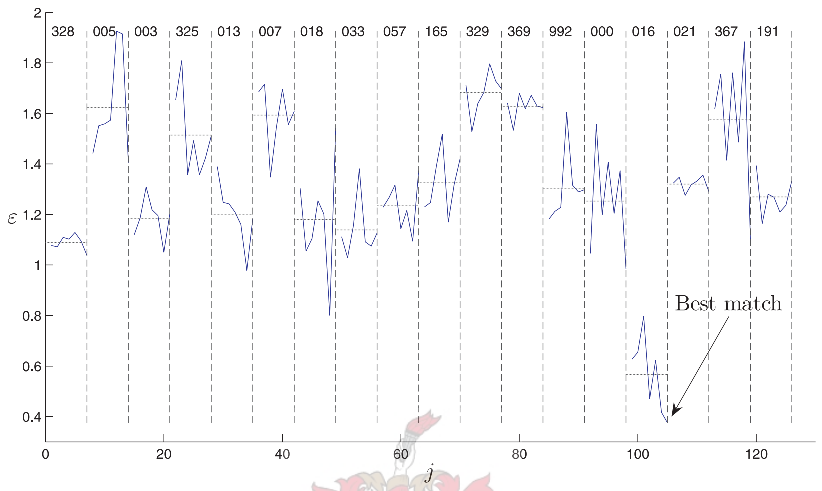


Figure 6.8: A plot of $\epsilon_j = \|\mathbf{c_t} - \mathbf{c_{f_j}}\|_2$ for a test face $\mathbf{t}$ and every face $\mathbf{f}_j$ in the database.

The ϵ_j's should logically be small for all j belonging to the same individual. We see this is indeed the case and the average ϵ has also been plotted for each user. This example clearly shows how having several (different) images of each individual in the database is useful as these ϵ values would typically be used in the verification stage. As an intuitive example, if the average ϵ for a user is not small enough, it could be an indication that the current choice of j is a coincidental outlier. Alternatively, instead of considering the single smallest ϵ_j, we could consider the individual in the database with the smallest *average* ϵ, or a combination of such predicates to determine an identity claim.

6.3.2 Example 2: Identification Using U_ω, $\omega < k$

As a second example we use the same set of eigenfaces as before. This time we enroll into the system the 29 individuals *not* used in the calculation of the eigenfaces. Only two images per user are supplied to the system while the remaining six are used as test images. To Analyze how omission of eigenfaces affects recognition, we use, for projections given by (6.10), U_ω with $\omega = 1 \ldots k$, where $k = 265$ in this case. Figure 6.9 shows, as a percentage, the number of the 174 (6×29) faces that were correctly identified using the minimum ϵ_j predicate.

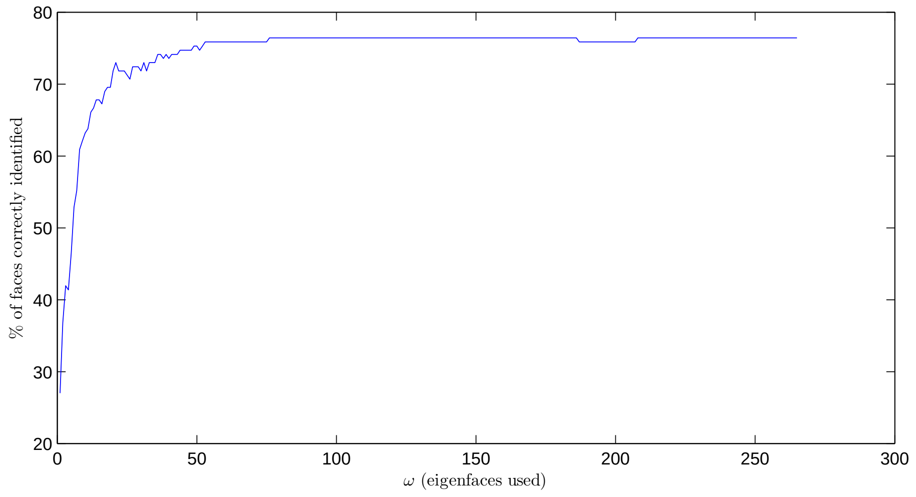


Figure 6.9: Recognition statistics for users not used in the training set.

As a slight modification to the above example, we have also enrolled two faces of *every* user in the xm2vts database into the system. It is therefore unavoidable that similar images of some of these users were also used in the training set to calculate the eigenfaces. Figure 6.10 Shows the same statistics as figure 6.9 only encompassing *all* the users of the xm2vts database.

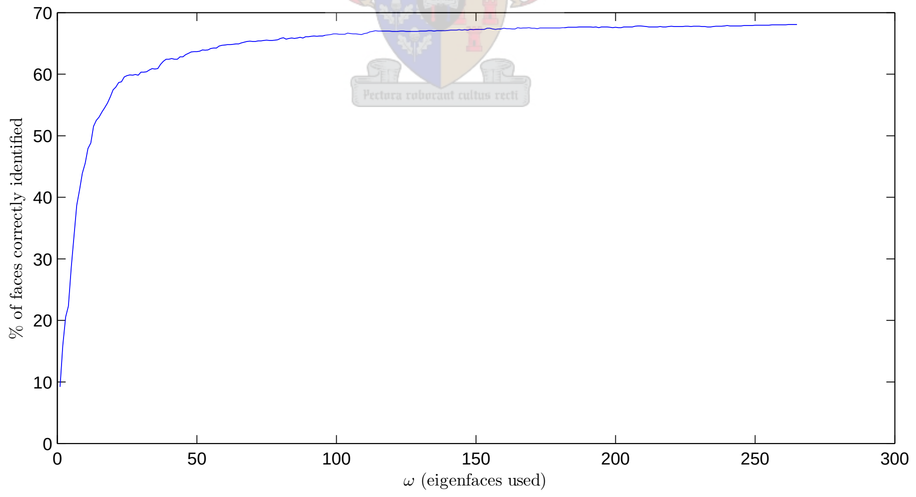


Figure 6.10: Recognition statistics for all users in the xm2vts database.

These results clearly show the dependence of the accuracy of recognition on the number of eigenfaces used. It further gives an indication of the number of redundant eigenfaces. From the figures it is clear that the eigenfaces associated with the smaller singular values can be omitted as they do very little to increase the accuracy of the system. Using only the first 100 eigenfaces would therefore be sufficient in these examples. It must be noted that the database constructed in the first case (with only 29 users) is too small to draw any conclusive results while the second case (with all the users included) includes users also used in the training set used to calculate the eigenfaces. Ideally we would want to use two large enough and disjoint sets of faces, one for training and the other for recognition. These statistics do however give a good indication of the behavior of such a system.

A further important factor influencing the final accuracy of the system is the number of faces supplied to the system per enrolled user. If the above experiment is repeated supplying the system with, instead of two, four faces of every user while retaining the other four as test images, the overall accuracy of the system increases to 80.1%.

Given the the above statistics, results for the recognition process are admittedly poor. The reason is that there is simply too much variation in the processed images resulting from out of plane face rotations, poor colour normalization, and variations in the shapes of the extracted faces of a single individual (as can be seen in some cases in figure 5.11). It is very possible that the latter factor could carry the bulk of the blame though the others are certainly influential. It must also be noted that in the calculation of the above statistics, some faces were included that would not be used in practice. Faces with severe out of plane rotations, extremely poor image quality an so forth, obviously influenced the statistics negatively. A quick manual count of such images, though, shows that they are in the vast minority – less that 2%.

The poor results of identification alone also clearly show the need for the verification stage. Verification could certainly improve these statistics by means of a threshold as a "reject" answer would be generated by the verifier in many of the cases of incorrect identification. However there is a logical point to ponder when it comes to letting the verifier use the best match (the identifier's output) as the identity claim. Since the best match is the enrolled face with the best score in the identification stage, it would logically also produce the highest confidence value during verification. In a system where only one face is supplied per enrolled user, this method of obtaining an identity claim for the verifier would be pointless, rendering the verifier useless. In this case the identity claim must be provided to the system from the outside, typically by the user. In the next example we will see that having more than one face per user is very beneficial as it solves this problem.

6.3.3 Example 3: Verification

As noted, there are two possibilities for obtaining an identity claim: using the best match from the identifier or letting it be user input. If the first option is used, the verifier will, as noted, output the highest possible confidence. At this point, having more than one face of each user enrolled as in figure 6.8, serves us very well in overcoming this problem. In this case the identity claim is still obtained from the best match but now the other faces belonging to that user can be used to generate different confidence values (lower than that of the claim), that combined, can yield a more accurate final confidence value. Figure 6.11 shows the results using this methodology where the confidence has been subjected to a

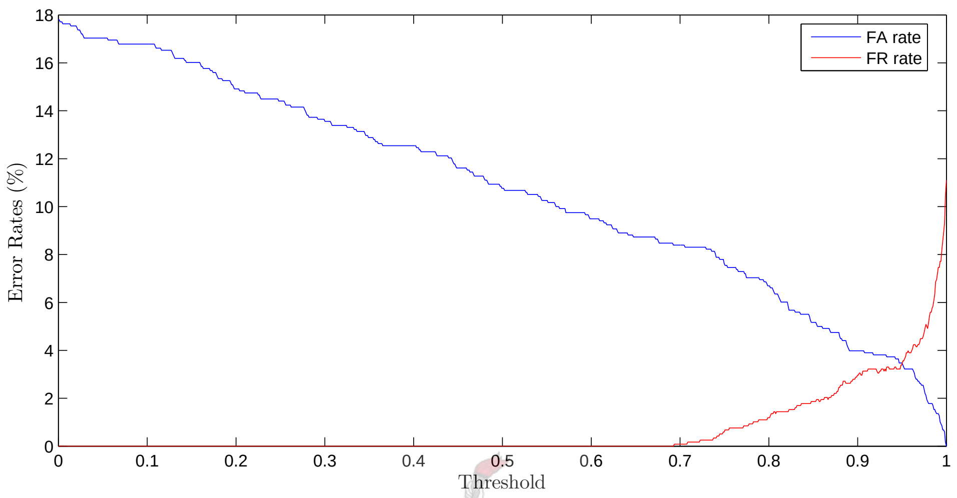


Figure 6.11: False acceptance and false rejection rates of a simple ranking verifier.

threshold. For this example four faces of each user were enrolled and four used as test faces. The optimal threshold value is often considered to be at the crossing point of the FA and FR curves, known as the equal error (EE) threshold, although it may be chosen otherwise depending on the practical implementation of the system.

If the system is set up in such a way that the user provides the identity claim (assumed honest and correct) then the FA case is not applicable. Figure 6.12 shows the FR results for the same system as before, only given the correct identity claim every time.

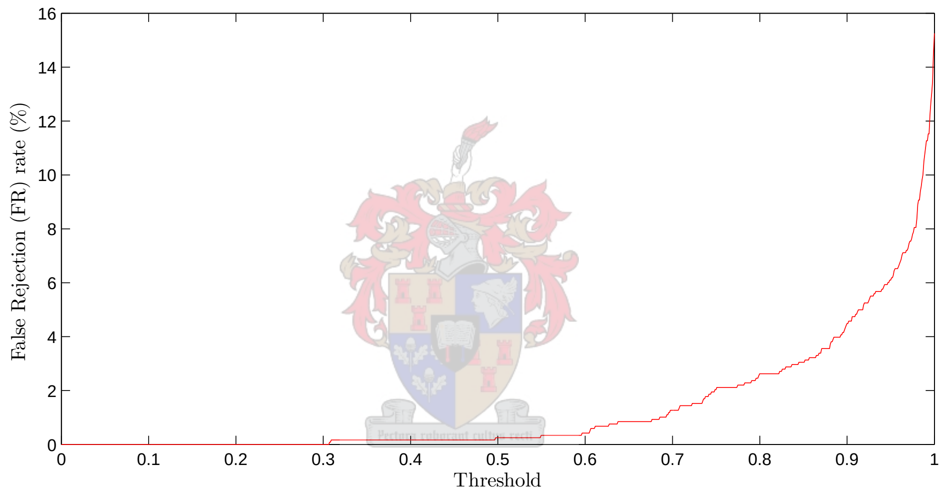


Figure 6.12: False rejection rates given correct identity claims.

Chapter 7

Conclusion

7.1 Summary and Achieved Objectives

The purpose of this thesis was to consider in detail the aspects necessary to implement a face recognition login system in software. Here we briefly summarize the assumptions made as well as the original objectives and how they were achieved.

• **Face detection and extraction:** The first input to face recognition software is an image. This image is assumed to contain a face and must be in full RGB colour. The face is assumed to be prominent, filling the image frame adequately. The background is permitted any degree of clutter as long as it does not contain large connected areas that are of near skin colour. Face detection is then achieved by skin colour detection in the HSV representation of the image. Training the computer to the most likely colour of skin in this colourspace involves approximating the skin colour distribution histogram with a Gaussian function and then using it to classify individual pixels. Face extraction is achieved by morphological image processing, specifically using 2–D filtering in an attempt to smooth the edges of the shape detected as the face. An issue that persists is the integration of the skin colour distribution histograms for individuals with light and those with dark skin tones. The latter is less accurately approximated by a Gaussian curve and may require a different approach.

• **Face normalization:** Faces presented to the software are allowed moderate in–plane rotations, generally not more than 40° from vertical. Out of plane rotations are treated more strictly with the tolerance being no more than a few degrees. We also assume that the image is of adequate quality (size, focus, etcetera), that the faces are frontal and that the eyes are open and clearly visible. Methods developed in chapter 5 based on a combination of colour analysis and morphological techniques, then detect the eyes and normalize the extracted face such that the eyes line up to idealized positions. The biggest issue with this methodology is the occasional failure of eye detection due to, among other factors, eyeglasses, closed eyes and low quality images. This inhibits fully automatic face normalization.

• **Face recognition, identification and verification:** Face recognition is achieved by principal component analysis. Here pixel values directly influence the mathematical process and is the reason why the faces are further assumed normalized with respect to colour (which can be achieved with controlled lighting or post processing of the image). Identification is achieved by considering the lowest score in the score vector given by (6.14). At this point, enrolling several faces of each enrolled user improves recognition statistics although no formal statistics were gathered on the subject. Face verification, given an identity claim, is achieved using a ranking verifier, resulting in a confidence value and an EE rate of approximately 3.5%. Other verifiers were not tested.

7.2 Practical Implementation

An experimental face recognition system was implemented in MATLAB®. Here we shortly discuss some of the practical aspects and shortcomings encountered. The facial images used in the implementation are mostly those from the xm2vts database although several images with more cluttered backgrounds were also considered and produced good results in general.

7.2.1 Face Detection

Using the face extraction method described in chapter 4, over 95% of faces (from the xm2vts database or otherwise) were isolated and extracted correctly. Cases where shoulders or hair of individuals were included in the detected face area due to incorrect skin pixel classification is, to an extent, successfully remedied by image cropping after face normalization. Still, incorrect skin pixel classification was found to be the primary cause for inaccurate face detection. In some cases, eyeglasses with thick frames caused problems as the frames tend to segment the face into two separate areas of skin colour. Figure 7.1 shows some examples where face extraction failed for various reasons. As mentioned earlier, the problems encountered when processing images of individuals with dark skin tones is a problem. It is in fact a serious one as it results in impractical general discrimination between potential users. This issue must certainly be addressed in future work.

Some of these problems, on the other hand, can be remedied in practice by, for example, requiring individuals to remove eyeglasses or any other headgear before the photo is taken, as well as making sure that no hair is obscuring the face. However, the problem of severely miss-shaped cutouts is in some specific cases not as severe as originally percieved. The idea is to produce faces that are mathematically consistent and give good separability from other images even if they appear miss-shaped to humans. Consider for example figure 7.1(c), where the individual has a full beard. In most such cases, the observed cutout problem will be reproduced in most, if not all, of the individual's images. This leads to

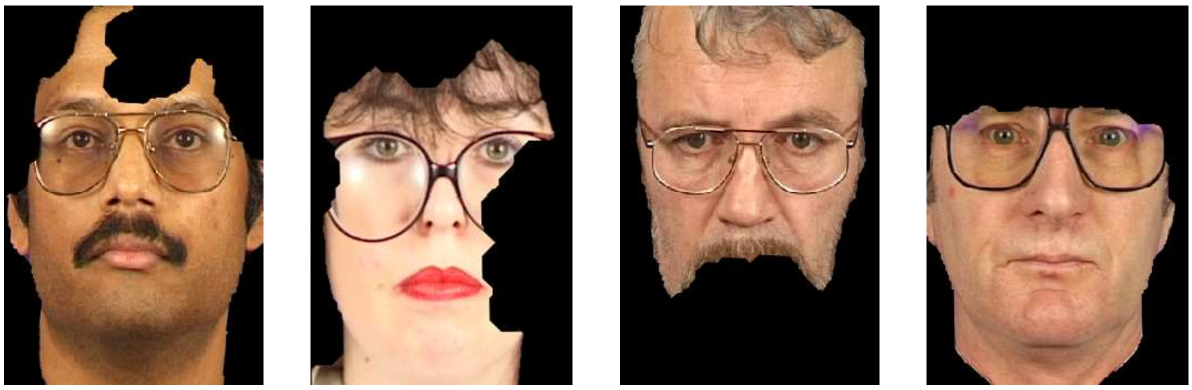

(a) Individual with dark skin. (b) Image with bad lighting. (c) Individual with a full beard. (d) Eyeglasses with thick frames.

Figure 7.1: Failures in correct face extraction due to various reasons.

the fact that, if the error is consistent, all the extracted faces of that individual differ from extracted faces of other individuals. This is in fact beneficial for face recognition as these miss–extracted faces' projections will differ more from those of the other well extracted faces, leading to higher confidence values in the verification stage. This argument is intuitive but no testing was done on the matter.

A serious problem not adequately solved in this thesis is that of colour adjustment to compensate for adverse lighting conditions. The images in the xm2vts database have constant lighting so that no adjustment is necessary. When taking our own photos, especially when not using a diffused photographic flash, the lighting is often found to be unacceptable. In some cases, this can be remedied by manual colour adjustment as described in appendix B but still fails in others, especially if the light is directional. The biggest problem with implementing *automatic* colour adjustment is to automatically find a reference colour according to which the rest of the image can be adjusted. This problem is only overcome in the proposed system by the assumption of pre–normalized colour in the images.

As far as histogram approximation is concerned, the single Gaussian approximation is experimentally found to work the best. However, cases of images of individuals with extreme skin colour (due to race, poor image quality or incorrect lighting) or facial hair, did poorly in the face detection stage. A possible improvement might be to construct a second skin colour histogram using skin pixel samples only from individuals with particularly dark skin tones. The approximation of this histogram will need to be studied further as it is experimentally found to be more widespread than that obtained from caucasian samples alone. This could be used in conjunction with further shape detection and normalization algorithms, for example, forcing all cutouts to be elliptical. It must be stated that the normalized RGB and YC_rC_b colourspaces perform well for skin colour detection across population groups and that the HSV colourspace indeed has a similar advantage – almost all the variation between these groups occur in the saturation and value components.

A major conclusion of face detection is thus that detection based solely on colour information is not enough if PCA is used for recognition. To achieve accuracy appropriate for commercial systems, shape information will need to be incorporated into the detection and extraction process in such a way that it is also normalized.

7.2.2 Face Normalization

The correct normalization of faces is crucial as accurate recognition would be impossible otherwise. Two methods for achieving face alignment were discussed, namely ellipse fitting and eye detection. The latter is the better option in the case of facial images as it aligns internal features and not only the general shape of the cutout. Furthermore, the ellipse method would fail in many cases where non–face areas such as clothes or hair were extracted due to incorrect skin pixel classification.

The eye detection method proposed for face alignment performs very well experimentally as the results in appendix C clearly show. It must be pointed out that these results can only be expected if the face is properly extracted and if the earlier assumptions about the faces hold, namely that they are frontal, fill the image, and are in full RGB colour. These are reasonable assumptions to make considering that the final product is aimed at a login system and that the conditions under which the photos are taken can be well controlled by the users and administrators of the system.

As the facial images considered are two dimensional representations of three dimensional objects, some information of the face is lost when the photo is taken. The normalization method proposed does not solve the problem of out–of–plane rotations of the face and is the primary reason for the requirement that the images be frontal. The xm2vts database contains several examples of images with such rotations and should in practice be considered unacceptable as either training or test faces.

The eye detection method proposed appears to work well and achieves a 91.3% overall success rate with a 78.8% rate for individuals with eyeglasses and 98.3% for those without. Eyeglasses are therefore a major reason for the failure of correct eye detection although other factors also play a role.

7.2.3 ASM and AAM for Face Detection and Normalization

Active shape models (ASM) and active appearance models (AAM) are two popular methods recently proposed for face detection and normalization. Both inherently make use of the faces' shapes and may well provide a solution to the earlier problem of inaccurate cutout as they could be used to normalize their shapes. ASM uses local models to search for a face shape under the constrains of a global model. AAM uses PCA based subspace analysis to model shape and texture variation and the correlations between them. Linear relationships are assumed between appearance and texture variations and between texture and position variations.

Typically, several faces constituting a training set are represented by a sequence of 2–D coordinates corresponding to predefined landmarks of the face. The mean shape is calculated and subsequent face shapes' distances from this mean is calculated using a PCA approach and orthogonal modes of variation. At this point, in light of AAM, the texture of the face is wrapped to the shape model by appropriate interpolation. A more detailed discussion of these methods can be found in [31].

7.2.4 Face Recognition

Face recognition using the SVD is much better defined mathematically than face detection and normalization as implemented in this thesis. The methods developed in this thesis for detection and normalization are to a great extent ad hoc and depend greatly on the assumptions made about the images, where recognition is mathematically more elegant and comparatively simple. The most costly computation is calculating the SVD of the set of training faces. This is an unavoidable step and requires all the training faces to be loaded into memory at once. This however only needs to be done once after which only the eigenfaces are saved. These would originally number as many as the number of training faces used but, as discussed in §6.1.2, it is possible to omit many of them without noticeably affecting recognition statistics.

When enrolling new users, the projections given by (6.10) are all that is needed for recognition. These can be calculated once, stored on disk and then loaded only when they are needed. This is also a great space saver as it is not mathematically necessary to retain the images of a given user but only their projection coefficients.

Even though the lighting conditions in the xm2vts database are consistent, experiments show that the slight variations still affect the recognition process and that PCA is sensitive to the colour and shape of the face cutout. Quoting Moses, Adini and Ullman [35]: "The variations between the images of the same face due to illumination and viewing direction are almost always larger than the image variation due to change in face identity". An example can be seen in figure 5.11 (leftmost seven columns, second row from the bottom) where the difference in lightness is likely due to camera aperture settings. As small as this error may seem, it is enough to sometimes foil the recognition process, emphasizing the importance of colour correction and normalization of the images. It is interesting to note that variations in illumination are generally contained in a linear subspace of their own and can also be modeled with eigenfaces [19].

It must be noted that out of plane rotations of faces can be handled by the PCA method if training images of such cases are provided in the original training set. However this would increase the size of the training set considerably, as these images will have to be representative of possible faces *and* possible rotations – a much larger permutation. Another conclusion of the eiginfaces is that, in retrospect, it may indeed be useful to recognition not to calculate the eigenfaces for the R, G and B channels separately but rather to place all the information in one matrix (see 6.1). This would include the potentially significant correlation between colours which is otherwise ignored.

A final observation is that the shape of the cutouts affected the eigenfaces (quite possibly negatively). Instead of using the face in its cutout shape to calculate the eigenfaces, we could use the original face area from the RGB image (similarly aligned and cropped) and assign Gaussian weights to pixels based on their distance from the center of the face (which can easily be approximated once we have determined the eye positions). This approach would taper the images gradually avoiding the sudden transitions between face and black areas as observed in cutout images.

7.2.5 Face Training Sets

Consideration and choice of the faces in the training set used to calculate the eigenfaces is important and there are different ways to go about this during practical implementation. The most common methodology is to choose a training set of faces that is a good representation of faces in general and calculate the eigenfaces from this set. This principle is commonly followed in systems where users number in the hundreds or thousands (or more) and recalculation of the eigenfaces is not an option.

For very small face database/recognition systems (where enrolled users would typically number less than fifty), we could consider including the faces of each enrolled user in the training set used to calculate the eigenfaces. This would require the recalculation of the eigenfaces every time a new user is enrolled but is feasible if the database is expected to remain small and enrollment of new users is expected to be infrequent. This methodology has the advantage that test faces of currently enrolled users will be more accurately reconstructible, resulting in better identification and verification statistics. This option could be further studied to determine the increase in accuracy versus loss of efficiency.

7.2.6 Testing

Statistics were gathered for the eye detection, face identification and face verification using the entire xm2vts database. Face extraction was also extensively tested although statistics are not presented as the criteria for successful extraction are more subjective. The system was not tested on any other databases due to time and access constraints.

Eye detection in extracted faces yielded a 91.3% overall success rate. This can be improved to over 97% if the requirement is imposed that all individuals remove eyeglasses at the time of photography, or if the eyeglasses are automatically removed by software [57].

Face recognition using two training and six test images of each individual resulted in a 77% correct recognition rate. If four images are used for training and the other four for testing, the correct recognition rate increases to 80.1%.

Using a ranking verifier and varying the acceptance threshold, an EE rate of approximately 3.5% is achieved at a threshold of 0.95.

Appendix A

Basic Set Theory

We will generally follow the notation of [16]. Let A be a set in Z^2. Then if $a = (a_1, a_2)$ is and element of A, we write

$$a \in A. \tag{A.1}$$

Similarly, if a is not an element of A, then

$$a \notin A. \tag{A.2}$$

A set with no elements is known as the *null* or *empty set* and denoted by $\emptyset$. If we have a second set B, then A is a subset of B if every element in A is also in B and is written as

$$A \subseteq B. \tag{A.3}$$

The union of two sets are all elements belonging to *either A or B* and we write

$$A \cup B, \tag{A.4}$$

while the intersection consists of elements that are in *both A and B*, denoted by

$$A \cap B. \tag{A.5}$$

Sets with no elements in common are known as *disjoint* or *mutually exclusive* and have the property

$$A \cap B = \emptyset. \tag{A.6}$$

We will also denote sets in terms of the contents of braces $\{\bullet\}$. For example

$$\hat{B} = \{w | w = -b, \text{ for b} \in \text{B}\} \tag{A.7}$$

is known as the *reflection* of a set and is obtained by multiplying all the elements in set B with -1. The compliment of a set A is all the elements not in A,

$$A^c = \{w | w \notin A\}. \tag{A.8}$$

The difference between sets A and B is then

$$A - B = \{w|w \in A,\ w \notin B\} = A \cap B^c. \tag{A.9}$$

The translation of a set A to a point $z = (z_1, z_2)$, written as $(A)_z$, is given by

$$(A)_z = \{w|w = a + z,\ \text{for } a \in A\}. \tag{A.10}$$

We can now describe common binary operations in terms of set theory. Table A.1 gives some of the most common binary operations' representation in set notation.

Logical	Set Notation
AND	$A \cap B$
NOT	A^c
OR	$A \cup B$
XOR	$(A \cap B^c) \cup (A^c \cap B)$

Table A.1: Common logical operations written in set theory.

Appendix B

Manual Colour Adjustment

The facial images provided in the xm2vts database are taken under very controlled circumstances. One factor that is particularly constant is the uniformity of lighting on the faces. As the skin colour detection method proposed in chapter 4 is completely dependent on the colours in the image, this thesis would be incomplete without supplying, at least simply, a method for colour adjustment in facial images yielding poor skin detection.

Hsu, Abdel–Mottaleb and Jain [24] propose a method to automatically adjust the colour of an image according to the average value of the pixels in the top 5% of the luma (gamma–corrected luminance). This is useful in real–world images as lighting is rarely controlled. For our purpose, however, we assume that the lighting can be controlled to some extent when the faces are photographed. This allows us to leave colour adjustment as a process requiring human input as it will only be required in special cases.

As the HSV colourspace is ultimately used in this thesis for skin colour detection, we will shortly focus on a method for adjusting the HSV representation of selected facial images to improve skin colour detection and therefor face extraction. The method relies on a reference colour manually specified by the user to adjust the rest of the image. Consider an image as a set of (h, s, v) triples. First the user specifies a point on the image that is part of skin. To overcome the potential problem of noise we consider a small neighbourhood of this point and calculate the average HSV representation of these pixels given by

$$\overline{\mathbf{c}} = (\overline{h}, \overline{s}, \overline{v}). \tag{B.1}$$

We have form (4.3) calculated from a training set of skin colour pixels, the h, s and v means of skin colour given by μ_h, μ_s and μ_v respectively. The hue of the entire image can now be rotated so that the hue of the new image is given by

$$h_n = h - (\overline{h} - \mu_h) \pmod{360}. \tag{B.2}$$

Adjusting the saturation and value of the image can now be done either linearly or non–linearly by means of gamma–correction [16]. For the linear case we would simply calculate

$$\begin{aligned} s_n &= s - (\overline{s} - \mu_s) \text{ and} \qquad \text{(B.3)} \\ v_n &= v - (\overline{v} - \mu_v) \qquad \text{(B.4)} \end{aligned}$$

and set any of the new values that lie outside $[0, 255]$ to either 0 or 255. The gamma–correction alternative involves non–linearly transforming the saturation and value by

$$\begin{aligned} s_n &= s^{\alpha_s} \text{ and} \qquad \text{(B.5)} \\ v_n &= v^{\alpha_v} \qquad \text{(B.6)} \end{aligned}$$

where the exponents α_s and α_v can be calculated by

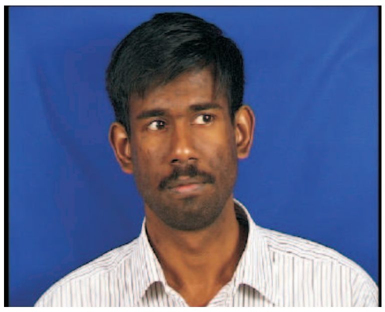
$$\begin{aligned} \alpha_s &= \log_{\overline{s}} \mu_s \text{ and} \qquad \text{(B.7)} \\ \alpha_v &= \log_{\overline{v}} \mu_v . \qquad \text{(B.8)} \end{aligned}$$

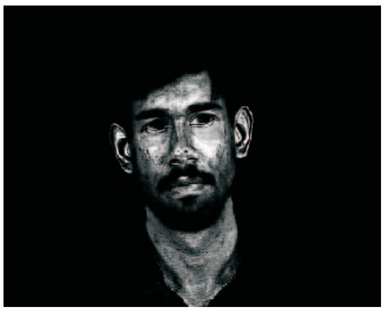
Note that gamma–correction using these equations requires us to first scale the saturation and value to $[0, 1]$ before applying (B.5) and (B.6), after which the values are rescaled to $[0, 255]$. This ensures that no values are outside the allowed range after applying gamma–correction.

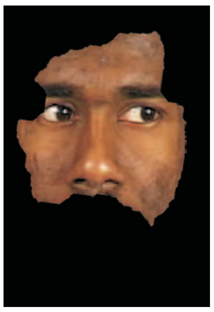
As an example, let us consider an image from the xm2vts database of an individual with a particularly dark skin colour as in figure B.1. The original image in figure B.1(a) is unchanged before we apply (4.2) to calculate the skin probability map given in figure B.1(b). Having done this, the face is extracted using the process described in §4.2, the result of which can be seen in figure B.1(c). As can clearly be seen, the process failed to extract the entire face due to poor skin pixel classification.

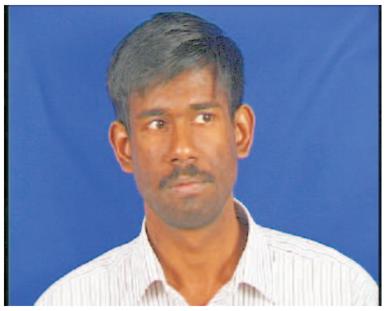
Figure B.1(d) shows the same image as in B.1(a), only with the hue, saturation and value components adjusted by (B.2), (B.5) and (B.6). This required a user specified neighbourhood on the original image that was chosen on the left cheek. After this colour adjustment, the skin tones are sufficiently close to the skin colour cluster shown in the histogram in figure 4.4 resulting in superior skin pixel classification. This in turn leads to a better extraction of the face from the original image as shown in figure B.1(f).

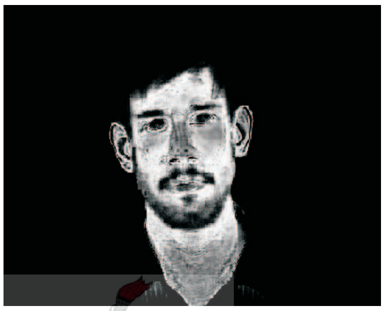
The drawbacks of this manual adjustment method are that there is no obvious way to automatically determine whether this adjustment is necessary, nor what the adjustment parameters should be. This method can therefor only be used in specific cases to compensate for specific image conditions. In contrast to this, the method proposed in [24] is fully


(a) Original image.


(b) Skin probability map of B.1(a).


(c) Extracted and normalized face.


(d) Image with corrected HSV components.


(e) Skin probability map of B.1(d).

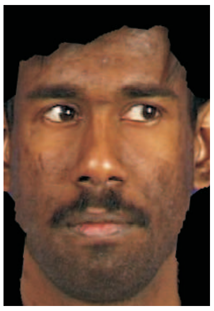

(f) Extracted and normalized face.

Figure B.1: Applying user aided gamma correction to an image in order to improve skin detection and face extraction.

automatic and could be used to process all the images *before* the set of skin training pixels is acquired. This would yield a better skin training set and, as test images would also be automatically adjusted, result in better automated skin colour detection.

Appendix C

Eye Detection Statistics

The following table (table C.1) contains statistics of the eye detection method proposed in §5.3 after extracting the face from the images in the manner described in §4.2. The method was tested on the xm2vts database which contains 8 photos of 295 individuals (2360 images, 4720 eyes). An important distinction to make between faces is between individuals with eyeglasses and those without. The eye detection rate for individuals with eyeglasses is predictably lower than for those without. For those without eyeglasses, a detection rate of 98.3% was achieved while for those with eyeglasses only 78.8% of eyes were detected. Factors that lead to miss–detection were closed eyes, extreme head pose, tinted glasses, prominent eyeglass frames and low quality images. Overall, a 91.3% eye detection rate is achieved.

xm2vts User Name	With Eyeglasses Detection	Without Eyeglasses Detection	User Eye Detection Result
000	0/0	16/16	100.0%
001	0/0	16/16	100.0%
002	10/16	0/0	62.5%
003	0/0	16/16	100.0%
004	0/0	16/16	100.0%
005	8/8	8/8	100.0%
006	0/0	16/16	100.0%
007	12/16	0/0	75.0%
008	0/0	16/16	100.0%
009	0/0	14/16	87.5%
010	0/0	16/16	100.0%
011	0/0	16/16	100.0%
012	0/0	16/16	100.0%
013	11/12	4/4	93.8%
016	16/16	0/0	100.0%
017	0/0	16/16	100.0%
018	0/0	16/16	100.0%
019	13/16	0/0	81.3%
020	0/0	16/16	100.0%
021	0/0	16/16	100.0%
022	11/16	0/0	68.8%
023	0/0	16/16	100.0%
024	0/0	16/16	100.0%
025	14/16	0/0	87.5%
026	16/16	0/0	100.0%
027	0/0	16/16	100.0%
028	16/16	0/0	100.0%
029	12/16	0/0	75.0%
030	0/0	16/16	100.0%
031	0/0	16/16	100.0%
032	0/0	16/16	100.0%
033	12/16	0/0	75.0%
034	12/16	0/0	75.0%
035	11/16	0/0	68.8%
036	0/0	16/16	100.0%
037	0/0	16/16	100.0%
038	0/0	16/16	100.0%

039	0/0	16/16	100.0%
040	0/0	16/16	100.0%
041	0/0	16/16	100.0%
042	11/16	0/0	68.8%
043	16/16	0/0	100.0%
044	0/0	16/16	100.0%
045	0/0	16/16	100.0%
046	0/0	16/16	100.0%
047	0/0	16/16	100.0%
048	11/16	0/0	68.8%
049	0/16	0/0	0.0%
050	0/0	12/16	75.0%
051	0/0	12/16	75.0%
052	0/0	16/16	100.0%
053	7/12	4/4	68.8%
054	0/0	16/16	100.0%
055	16/16	0/0	100.0%
056	4/4	12/12	100.0%
057	1/4	12/12	81.3%
058	16/16	0/0	100.0%
059	0/0	16/16	100.0%
060	1/16	0/0	6.3%
061	16/16	0/0	100.0%
062	0/0	16/16	100.0%
064	0/0	16/16	100.0%
065	6/16	0/0	37.5%
066	10/16	0/0	62.5%
067	0/0	16/16	100.0%
068	0/0	16/16	100.0%
069	0/0	16/16	100.0%
070	4/4	10/12	87.5%
071	0/0	16/16	100.0%
072	0/0	16/16	100.0%
073	0/0	16/16	100.0%
074	0/0	14/16	87.5%
075	0/0	16/16	100.0%
078	0/0	16/16	100.0%
079	0/0	16/16	100.0%
080	0/0	16/16	100.0%
081	16/16	0/0	100.0%
082	12/16	0/0	75.0%
083	0/0	16/16	100.0%
084	0/0	16/16	100.0%
085	0/0	16/16	100.0%
086	0/0	15/16	93.8%
087	8/12	4/4	75.0%
088	16/16	0/0	100.0%
089	14/16	0/0	87.5%
090	16/16	0/0	100.0%
091	0/0	16/16	100.0%
092	0/0	16/16	100.0%
093	14/16	0/0	87.5%
095	0/0	16/16	100.0%
096	0/0	16/16	100.0%
098	0/0	16/16	100.0%
099	10/12	4/4	87.5%
101	11/16	0/0	68.8%
102	0/0	16/16	100.0%
103	14/16	0/0	87.5%
104	4/4	12/12	100.0%
105	0/0	16/16	100.0%
107	0/0	16/16	100.0%
108	0/0	16/16	100.0%
109	0/0	16/16	100.0%
110	0/0	16/16	100.0%
111	0/0	16/16	100.0%
112	0/0	16/16	100.0%
113	1/16	0/0	6.3%
114	11/16	0/0	68.8%
115	16/16	0/0	100.0%
116	0/0	16/16	100.0%
119	12/12	4/4	100.0%
120	7/12	4/4	68.8%
121	16/16	0/0	100.0%
122	9/16	0/0	56.3%
123	0/0	16/16	100.0%
124	0/0	15/16	93.8%
125	0/0	16/16	100.0%
126	12/16	0/0	75.0%
127	0/0	16/16	100.0%
128	0/0	16/16	100.0%
129	0/0	16/16	100.0%

130	0/0	16/16	100.0%
131	0/0	16/16	100.0%
132	5/16	0/0	31.3%
133	14/16	0/0	87.5%
134	0/0	16/16	100.0%
135	16/16	0/0	100.0%
136	0/0	16/16	100.0%
137	0/0	16/16	100.0%
138	0/0	16/16	100.0%
140	0/0	16/16	100.0%
141	16/16	0/0	100.0%
142	0/0	16/16	100.0%
143	0/0	16/16	100.0%
145	12/16	0/0	75.0%
146	0/0	16/16	100.0%
147	2/16	0/0	12.5%
148	0/0	12/16	75.0%
149	0/0	16/16	100.0%
150	16/16	0/0	100.0%
152	0/0	16/16	100.0%
153	13/16	0/0	81.3%
154	0/0	14/16	87.5%
155	0/0	16/16	100.0%
157	12/16	0/0	75.0%
158	0/0	16/16	100.0%
159	0/0	16/16	100.0%
160	0/0	16/16	100.0%
161	16/16	0/0	100.0%
163	0/0	16/16	100.0%
164	0/0	16/16	100.0%
165	0/0	16/16	100.0%
166	0/0	14/16	87.5%
167	0/0	16/16	100.0%
168	0/0	16/16	100.0%
169	12/16	0/0	75.0%
170	0/0	16/16	100.0%
171	14/16	0/0	87.5%
172	0/0	14/16	87.5%
173	0/0	16/16	100.0%
174	13/16	0/0	81.3%
175	2/4	12/12	87.5%
176	0/0	16/16	100.0%
177	0/0	16/16	100.0%
178	0/0	16/16	100.0%
179	0/0	16/16	100.0%
180	16/16	0/0	100.0%
181	0/0	16/16	100.0%
182	16/16	0/0	100.0%
183	16/16	0/0	100.0%
185	16/16	0/0	100.0%
187	0/0	16/16	100.0%
188	0/0	16/16	100.0%
189	11/16	0/0	68.8%
190	10/16	0/0	62.5%
191	16/16	0/0	100.0%
193	0/0	16/16	100.0%
196	0/0	16/16	100.0%
197	14/16	0/0	87.5%
198	0/0	16/16	100.0%
199	12/16	0/0	75.0%
200	0/0	16/16	100.0%
201	16/16	0/0	100.0%
202	0/0	16/16	100.0%
203	0/0	16/16	100.0%
206	0/0	7/16	43.8%
207	10/12	4/4	87.5%
208	0/0	16/16	100.0%
209	0/0	16/16	100.0%
210	0/0	15/16	93.8%
211	0/0	16/16	100.0%
212	0/0	16/16	100.0%
213	3/16	0/0	18.8%
215	10/16	0/0	62.5%
216	5/16	0/0	31.3%
218	16/16	0/0	100.0%
219	0/0	16/16	100.0%
221	0/0	16/16	100.0%
222	0/0	16/16	100.0%
224	0/0	16/16	100.0%
225	0/0	16/16	100.0%
226	0/0	16/16	100.0%
227	13/16	0/0	81.3%

228	5/8	8/8	81.3%
229	15/16	0/0	93.8%
231	0/0	14/16	87.5%
232	11/12	4/4	93.8%
233	16/16	0/0	100.0%
234	0/0	16/16	100.0%
235	0/0	16/16	100.0%
236	16/16	0/0	100.0%
237	0/0	16/16	100.0%
240	0/0	16/16	100.0%
241	16/16	0/0	100.0%
242	16/16	0/0	100.0%
243	0/0	16/16	100.0%
244	0/0	16/16	100.0%
246	10/16	0/0	62.5%
248	0/0	15/16	93.8%
249	0/0	14/16	87.5%
250	16/16	0/0	100.0%
253	0/0	16/16	100.0%
255	4/4	12/12	100.0%
258	0/0	14/16	87.5%
259	0/0	16/16	100.0%
261	0/0	16/16	100.0%
263	0/0	16/16	100.0%
264	2/4	12/12	87.5%
266	0/0	16/16	100.0%
267	13/16	0/0	81.3%
269	4/4	12/12	100.0%
270	0/0	16/16	100.0%
271	0/0	16/16	100.0%
272	0/0	16/16	100.0%
274	12/16	0/0	75.0%
275	9/16	0/0	56.3%
276	8/16	0/0	50.0%
278	8/8	8/8	100.0%
279	0/0	16/16	100.0%
280	0/0	16/16	100.0%
281	0/0	16/16	100.0%
282	0/0	16/16	100.0%
283	15/16	0/0	93.8%
284	0/0	16/16	100.0%
285	0/0	16/16	100.0%
286	0/0	16/16	100.0%
287	0/0	16/16	100.0%
288	11/16	0/0	68.8%
289	0/0	16/16	100.0%
290	7/8	8/8	93.8%
292	12/12	4/4	100.0%
293	8/12	4/4	75.0%
295	0/0	16/16	100.0%
300	0/0	12/16	75.0%
301	0/0	12/16	75.0%
305	0/0	16/16	100.0%
310	15/16	0/0	93.8%
312	16/16	0/0	100.0%
313	0/0	16/16	100.0%
314	16/16	0/0	100.0%
315	0/0	16/16	100.0%
316	0/0	16/16	100.0%
317	8/16	0/0	50.0%
318	15/16	0/0	93.8%
319	0/0	16/16	100.0%
320	16/16	0/0	100.0%
321	0/0	16/16	100.0%
322	14/16	0/0	87.5%
323	6/16	0/0	37.5%
324	6/16	0/0	37.5%
325	0/0	16/16	100.0%
328	0/0	16/16	100.0%
329	0/0	16/16	100.0%
330	0/0	16/16	100.0%
331	0/0	16/16	100.0%
332	0/0	16/16	100.0%
333	0/0	14/16	87.5%
334	0/0	16/16	100.0%
335	0/0	16/16	100.0%
336	16/16	0/0	100.0%
337	12/12	4/4	100.0%
338	0/0	16/16	100.0%
339	0/0	16/16	100.0%
340	14/16	0/0	87.5%
341	15/16	0/0	93.8%

342	5/12	4/4	56.3%
357	14/16	0/0	87.5%
358	0/0	16/16	100.0%
359	0/0	16/16	100.0%
360	0/0	16/16	100.0%
362	4/4	12/12	100.0%
364	0/0	16/16	100.0%
365	0/0	16/16	100.0%
366	10/16	0/0	62.5%
367	0/0	16/16	100.0%
369	0/0	16/16	100.0%
370	14/16	0/0	87.5%
371	0/0	16/16	100.0%

	Overall % with Eyeglasses	Overall % without Eyeglasses	Overall Detection %
	78.8%	98.3%	91.3%

Table C.1: Eye detection statistics as applied to the xm2vts database.

Appendix D

Singular Value Decomposition (SVD)

Here we consider singular value decomposition in theory and its application to face recognition. We assume at this point that all images considered previously are matrices with real (more specifically natural and > 0) entries. We can therefore interchange the transpose A^T and complex conjugate A^* of a matrix A since for real matrices $A^* = A^T$.

To shorten the text we assume that the reader has a basic background knowledge of linear algebra and thus elementary terms are not explained. Literature on SVD or topics relevant to this discussion can be found in almost any book on elementary linear algebra, see [52].

D.1 Calculation and Interpretation of the SVD

The singular value decomposition (or SVD) is a tool for extracting a lot of useful information in a very compact form from an arbitrary matrix. It is particularly useful for finding an orthonormal basis for a set of vectors but has many other uses and contains much more information than other decomposition methods such as the QR factorization.

Definition of the SVD

The SVD of an arbitrary $m \times n$ matrix A is given by

$$A = U\Sigma V^T \tag{D.1}$$

where U is of size $m \times m$, V is of size $n \times n$ and are both unitary containing the left and right singular vectors of A (or alternatively, the eigenvectors of AA^T and A^TA). Σ is of size $m \times n$ and contains the singular values[1] of A (alternatively, the eigenvalues of AA^T or A^TA) on the main diagonal. The singular values are usually ordered decreasingly down the diagonal, i.e. $\sigma_j \leq \sigma_i$ if $j > i$ and there are always exactly $\min(m, n)$ singular values, some of which may be zero. Additionally, the number of non zero singular values is the rank, r, of the matrix.

[1]The singular values are the positive square roots of the eigenvalues. That is $\sigma_j = \sqrt{\lambda_j}$.

D.1.1 Calculating the SVD

Consider the definition of the SVD in (D.1). We note that if we calculate AA^T we get

$$\begin{aligned} AA^T &= (U\Sigma V^T)(U\Sigma V^T)^T \\ &= U\Sigma\Sigma^T U^T \\ &= U\Lambda U^T \end{aligned} \tag{D.2}$$

where the $\Lambda = \Sigma^2$ contains the eigenvalues on the main diagonal. In a similar fashion we obtain

$$A^T A = V\Lambda V^T. \tag{D.3}$$

Since U and V are unitary, $U^T = U^{-1}$ and $V^T = V^{-1}$ and we see that (D.2) and (D.3) are the diagonalizations [52] of AA^T and $A^T A$ respectively. In other words, the matrices U and V in (D.1) contain the eigenvectors of AA^T and $A^T A$ respectively. Since AA^T and $A^T A$ are symmetric, it can be proven that their eigenvalues are all real as well as positive and that U and V are real. Simplistically, the steps for calculating the SVD of a matrix A are as follows:

1. Calculate the eigenvalues of AA^T or $A^T A$. Then Σ is the $m \times n$ matrix with the positive square roots of these values in decreasing order on the main diagonal.
2. Calculate the eigenvectors of AA^T. They form the columns of U.
3. Calculate the eigenvectors of $A^T A$. They form the columns of V.

D.1.2 Reduced Form of the SVD

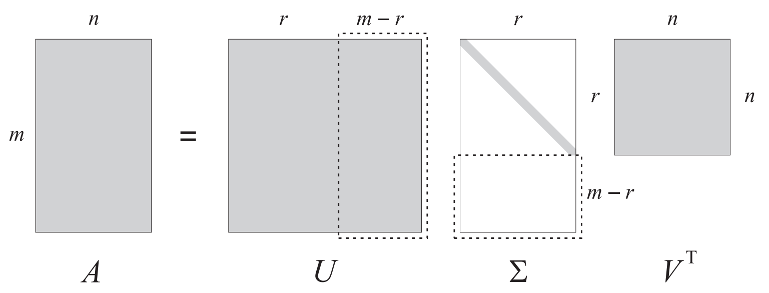
Let us consider an $m \times n$ matrix A with $m > n$. A graphic illustration of the SVD of A is given in figure D.2. A similar scenario develops when $m < n$ and is left to the reader. We know that Σ has many zero entries and, depending on the shape and rank of A, many elements in all three matrices U, Σ and V, may prove redundant. Even if A has full rank then we see that, owing to the zero rows in Σ, the last $m - r$ columns of U can be ignored without loss of information. If A has less than full rank, then parts of V can also be ignored. By omitting the information in the dashed boxes, we can write the reduced form of the SVD as

$$A = \widehat{U}\Sigma_+\widehat{V}^T \tag{D.4}$$

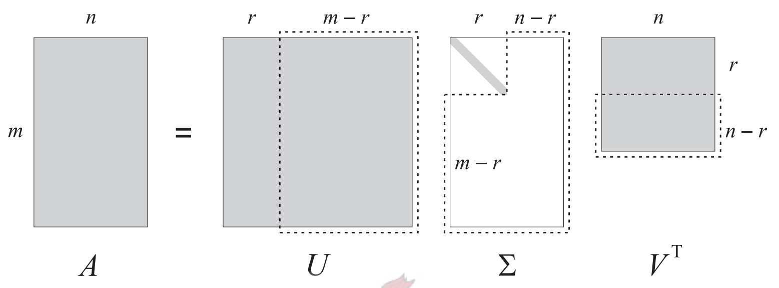
where $\widehat{U}$ is an $m \times r$ matrix consisting of the first r columns of U, $\widehat{V}$ is an $n \times r$ matrix consisting of the first r columns of V and Σ_+ is an $r \times r$ matrix with the r non–zero singular values on the main diagonal.

From the above discussion we note that we can write A as

$$A = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^T \tag{D.5}$$



(a) A has full rank. $n = r$.



(b) A has less than full rank. $n \neq r$.

Figure D.1: Graphic illustration of the SVD of a matrix A. For the reduced form, omit areas inside dashed boxes.

where $\mathbf{u}_j$ and $\mathbf{v}_j$ are the columns of U and V respectively. Furthermore we note that the singular values in most cases decrease very quickly indicating that we can ignore some of the smallest eigenvalues and still reobtain a relatively good reconstruction of A. This principle can be used in (lossy) data compression such as image compression and is discussed further in the context of face recognition and reconstruction in §6.1.2.

At this point we can state some implications of the SVD.

Theorem
If the SVD of an $m \times n$ matrix A with rank r is $A = U\Sigma V^T$ with $U \in \mathbb{R}^{m\times m}$, $\Sigma \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{n\times n}$, then:

1. *The first r columns of U form an orthonormal basis for the column space of A.*
2. *The first r columns of V form an orthonormal basis for the row space of A.*
3. *The last $m-r$ columns of U form an orthonormal basis for the left–null space of A.*
4. *The last $n-r$ columns of V form an orthonormal basis for the null space of A.*

Proof
From (D.4) and visually from figure D.2 we see that the columns of A can be written as a linear combination of the first r columns U, thus the columns of A span a subspace of the space spanned by the first r columns of U. Conversely we can write $\widehat{U} = A\widehat{V}\Sigma_+^{-1}$ from which we see that $\widehat{U}$ can be written as a linear combination of the columns of A and so the columns of U span a subspace of the space spanned by the columns of A. This proves 1.

Let us now consider (D.1) as $AV = U\Sigma$ and consider the structure of the resulting matrix. We see that $A\mathbf{v} = \mathbf{0}$ for the last $n-r$ columns of V. In other words, they all lie in the null–space of A. Conversely, suppose $A\mathbf{x} = \mathbf{0}$. Then

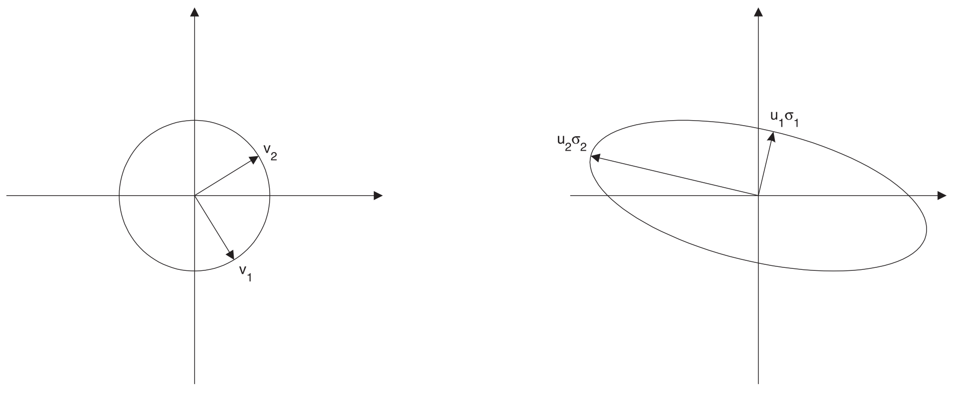
$$\begin{aligned} U\Sigma V^T\mathbf{x} &= \mathbf{0} \\ \Sigma V^T\mathbf{x} &= \mathbf{0} \end{aligned}$$

which implies that $\mathbf{x}$ is orthogonal to the first r columns of V. Therefore $\mathbf{x}$ must lie in the space spanned by the last $n-r$ columns of V which proves 4. The proofs for 2 and 3 are obtained in a similar fashion by considering A^T instead of A.

$\blacksquare$

D.1.3 Geometric Interpretation of the SVD

Let us consider the effect A has on a vector $\mathbf{x}$ upon multiplication. For simplicity we consider the 2–dimensional case and two unit vectors parallel to the primary axes, i.e. forming the 2×2 identity matrix I_2. Multiplying these vectors with A we write $AI_2 = U\Sigma V^T I_2$ where the process is split into three multiplications (by V^T, then by Σ and finally by



(a) Step 1: Multiplication with V^T.

(b) Steps 2 & 3: Multiplication with Σ and U.

Figure D.2: Geometric interpretation of the effect of a matrix on the unit vectors in terms of the SVD.

U). Multiplying I_2 with V^T is a transformation (invertible if A has full rank) depicted in figure D.2(a). The second step, multiplication by Σ, is a stretching of the resulting vectors, effectively transforming the unit circle into an ellipse. The third step is a second transformation achieved by multiplication with U. The final result is depicted in figure D.2(b). It is easy to generalize this principle to any higher dimension n where we consider unit hyper–spheres transformed to hyper–ellipses in n–dimensions.

As was just illustrated, the singular values provide us with useful information about the "stretching power" of a matrix A. This can also be put algebraically. Let us define the L_2–norm of a vector $\mathbf{x}$ as

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}, \tag{D.6}$$

i.e. the normal Euclidian length. Then we define the 2–norm of a matrix as

$$\|A\|_2 = \sup_{\|\mathbf{x}\|_2=1} \|A\mathbf{x}\|_2 \tag{D.7}$$

where $\mathbf{x}$ is any vector on the unit hyper–sphere, i.e. $\|\mathbf{x}\|_2 = 1$. From this principle (and figure D.2) it can be seen that

$$\|A\|_2 = \sigma_1, \tag{D.8}$$

that is the largest of the singular values of A. This can also be shown mathematically by noting that since A^TA is symmetrical, it can be diagonalized so that $SA^TAS^T = \Lambda$ where S is orthonormal and Λ contains only the eigenvalues of A^TA on the main diagonal. It can be proven that since A^TA is semi–positive definite, its eigenvalues are non–negative. Now we can write

$$\begin{aligned}
\|A\|_2 &= \sup_{\|\mathbf{x}\|_2=1} (\mathbf{x}^T A^T A\mathbf{x})^{\frac{1}{2}} \\
&= \sup_{\|\mathbf{x}\|_2=1} (\mathbf{x}^T S^T \Lambda S\mathbf{x})^{\frac{1}{2}} \\
&= \sup_{\|\mathbf{x}\|_2=1} ((S\mathbf{x})^T \Lambda (S\mathbf{x}))^{\frac{1}{2}} \\
&= \sup_{\|\mathbf{y}\|_2=1} (\mathbf{y}^T \Lambda \mathbf{y})^{\frac{1}{2}} \\
&= \sup_{\|\mathbf{y}\|_2=1} \sqrt{\sum \lambda_j y_j^2} \\
&\leq \sup_{\|\mathbf{y}\|_2=1} \sqrt{\lambda_{\max} \sum y_j^2} \\
&= \sqrt{\lambda_{\max}} = \sigma_1
\end{aligned}$$

if we choose $\mathbf{y}$ correctly.

D.2 Calculating the SVD Practically

Consider the basic steps for calculating the SVD in §D.1.1. From a theoretical standpoint it is perfectly correct and relatively simple. From a practical standpoint, however, in the light of computer matrix calculations, we encounter problems caused by rounding and truncation in the calculation of AA^T and A^TA. This causes the method to be generally unstable. It must be noted that entire books have been devoted to the calculation of the SVD and the problems realating to it [12][53][15]. These problems are varied and make up broad field that is presently very active in linear algebra. In this discussion we cover only the basic principles and will not go into any further detail of them.

The problem with the three steps proposed in §D.1.1 to calculate the SVD lies in the calculation of A^TA and its eigenvalues. It can be shown by perturbation theory [12][53][15] that the difference between the analytic and computed singular values (denoted by $\tilde{\sigma}_k$ and σ_k respectively) is given by

$$|\tilde{\sigma}_k - \sigma_k| = O(\epsilon_{\text{machine}} \|A\|^2/\sigma_k) \tag{D.9}$$

where $\epsilon_{\text{machine}}$ is treated as a constant and refers to the smallest number representable on a given computer. $O(\bullet)$ refers to the order of magnitude of a number. It can be seen from (D.9) that the largest of the singular values remain, for all practical purposes, unaffected by this error. It is the small singular values with $\sigma_k \ll \|A\|$ that can be seriously affected and require the use of different algorithms in practice.

There are several alternative methods for calculating the SVD of a general matrix A. All except Jacobi's method follow the same general principle:

1. Reduce A to bidiagonal[2] form B with orthogonal matrices U_1 and V_1 so that $A = U_1 B V_1^T$. This can be done in steps using Golub–Kahan bidiagonalization.

2. Find the SVD of B: $B = U_2 \Sigma V_2^T$. This can be done stably due to the reduced complexity of B.

3. Combine these decompositions as $A = (U_1U_2)\Sigma(V_1V_2)^T$. Now Σ contains the singular of A and the columns of $U = U_1U_2$ and $V = V_1V_2$ are the left and right singular vectors of A respectively.

Methods with this general form have been the standard from the 1960's. In recent years divide–and–conquer algorithms have started to gain ground due to their efficiency and may well become standard in the future.

D.3 Covariances and the SVD

Suppose we are given a set of n m–dimensional vectors $\mathbf{f}_j$, $j = 1, \ldots, n$. Let us suppose further that their mean is $\mathbf{0}$, i.e. $\frac{1}{n}\sum_{j=1}^{n} \mathbf{f}_j = \mathbf{0}$. Suppose now we want to find the direction and magnitude of maximum deviation of these vectors. We want to find the direction $\mathbf{u}$ that maximizes μ where

$$\mu = \max_{\|\mathbf{u}\|_2=1} \frac{1}{n}\sum_{j=1}^{n}(\mathbf{u}^T\mathbf{f}_j)^2. \tag{D.10}$$

We define the covariance matrix of the vectors as

$$C = \frac{1}{n}\sum_{j=1}^{n}\mathbf{f}_j\mathbf{f}_j^T \tag{D.11}$$

then we can rewrite (D.10) as

$$\mu = \max_{\|\mathbf{u}\|_2=1}(\mathbf{u}^T C\mathbf{u}). \tag{D.12}$$

Since C is symmetric, it can be diagonalized as $C = U\Lambda U^T$ where Λ is again a diagonal matrix containing the eigenvalues of C. Now we have that

$$\begin{aligned}\mu &= \max_{\|\mathbf{u}\|_2=1}(\mathbf{u}^T U\Lambda U^T\mathbf{u}) \\ &= \max_{\|\mathbf{y}\|_2=1}(\mathbf{y}^T\Lambda\mathbf{y})\end{aligned} \tag{D.13}$$

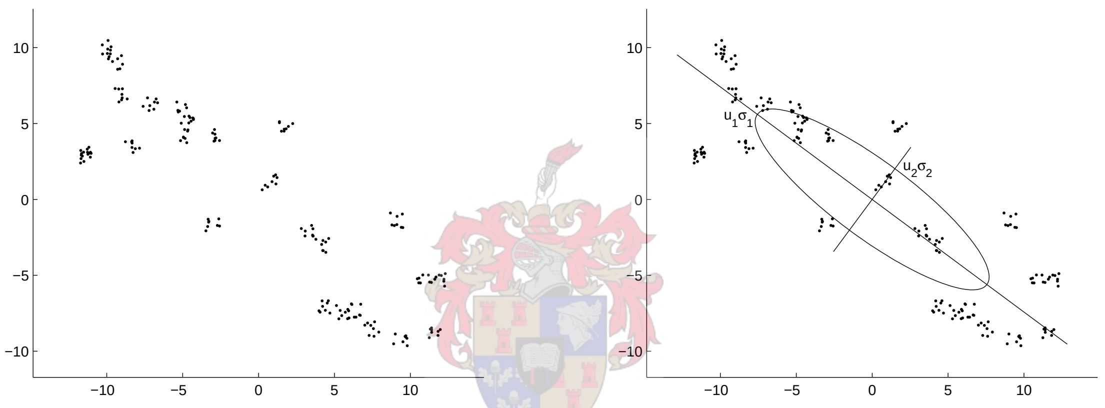
[2]A matrix is said to be bidiagonal if it only has non–zero elements on the main diagonal and the first superdiagonal.

where $\mathbf{y} = U^T\mathbf{u}$. We must now calculate

$$\mu = \max_{\|\mathbf{y}\|_2=1} \sum_{j=1}^{m} \lambda_j y_j^2. \tag{D.14}$$

Since U is unitary, we still have that $\|\mathbf{y}\|_2 = 1$ which implies that $\sum_{j=1}^{m} y_j^2 = 1$. We thus see that $\mu = \lambda_1$ and that the direction of maximum deviation, $\mathbf{u}$, is the first eigenvector of the covariance matrix C, i.e. $\mathbf{u}_1$. If we now ask for the direction of maximum deviation *orthogonal* to $\mathbf{u}_1$, we similarly find it to be the second eigenvector $\mathbf{u}_2$ and so forth. Furthermore, λ_j provides information of the deviation in direction $\mathbf{u}_j$.

A graphical interpretation of the above concept is given in figure D.3. In this example



(a) Data points in 2–D.

(b) Directions of maximum deviations and variance in these directions.

Figure D.3: The SVD used to determine direction of maximum deviation and variances in these directions for a set of 2–D vectors with zero mean.

we are given a set of 200 2–D vectors, the x and y values of which we put in a 2×200 matrix F. Their mean is made zero by subtracting the mean from each of the vectors. After calculating the SVD of F, we find that

$$\widehat{U} = \begin{bmatrix} -0.7844 & 0.6202 \\ 0.6202 & 0.7844 \end{bmatrix} \quad \text{and} \quad \Sigma_+ = \begin{bmatrix} 137.9555 & 0 \\ 0 & 35.5126 \end{bmatrix}.$$

By scaling the eigenvalues in Σ_+ by $\frac{1}{\sqrt{200}}$ we obtain the variance of the 200 vectors as 9.7549 and 2.5111 in the direction of the first and second eigenvectors respectively.

Now let us consider what would happen if we didn't subtract the mean from all the vectors. Suppose that before, the vectors had a mean that was very far from the origin. The first eigenvector would point in the direction of the mean and not in the direction of maximum deviation. The second eigenvector is restricted to be perpendicular to the first and so we loose descriptive information of the distribution of the data points. In higher dimensions, the the eigenvectors corresponding to smaller eigenvalues would not be as badly affected but we still loose information that would be provided by the first.

Appendix E

Implementation

Here we shortly discuss the basic modules necessary to implement a working face detection and recognition system in practice. We will generally follow the order of modules as they are described in chapters 2 through 6. It would be the responsibility of the programmer to piece them together into a working system. A module will be described in the following manner:

Module name: Brief explanation of the modules functionality.

Input: **Input 1:** Description of Input 1.

Input 2: Description of Input 2.

Input 3: Description of Input 3.

Output: **Output 1:** Description of Output 1.

Output 2: Description of Output 2.

Output 3: Description of Output 3.

Morphological Modules

Open{Close}(I, S)**:** Morphological opening{closing} of a binary or gray–scale image I using the structuring element S.

Input: I**:** An image (matrix) with elements 0 or 1 for binary or $\in [0, 255]$ for gray–scale.

S**:** A predefined structuring element, typically a small matrix.

Output: I_S The image I opened{closed} by the structuring element S.

Pseudo_Convex_Hull(I): Creates a pseudo–convex hull around the shape in the given binary image I.

Input: I: A binary image containing a single, 8-connected shape.

Output: H_c The pseudo–convex hull of I.

Colour Processing Modules

RGB_to_HSV(RGB): Converts a set of rgb triples to hsv triples using (3.4), (3.5) and (3.6).

Input: RGB: An array of rgb values with each of the entries $\in ([0, 255], [0, 255], [0, 255])$.

Output: HSV: An array of the same size as RGB but with elements $\in ([0, 359], [0, 255], [0, 255])$.

RGB_to_YC$_r$C$_b$(RGB): Converts a set of rgb triples to YC$_r$C$_b$ triples using (3.3).

Input: RGB: An array of rgb values with each of the entries $\in ([0, 255], [0, 255], [0, 255])$.

Output: **YC$_r$C$_b$:** An array of the same size as RGB but with entries converted to the YC$_r$C$_b$ colourspace.

Face Detection Modules

Get_Skin(I): Analyzes and separate a general image I into skin and non–skin areas.

Input: I: An image (photo) containing a both skin and non–skin areas.

Output: M: A binary mask of the same size as I indicating the largest connected area of skin color.

Get_Face(P): Analyzes a general image P and isolate the face within it.

Input: P**:** An image (photo) containing both skin and non–skin areas.

Output: F**:** The face in the image P that has been extracted and cropped. F is derived from masking P with Pseudo_Convex_Hull(Get_Skin(I)).

Face Normalization Modules

Get_Eyes(F): Detects the positions of the eyes in the facial image F.

Input: F**:** An extracted and cropped facial image from the output of **Get_Face(P)**.

Output: eye_1**:** x and y coordinates of the first eye.

eye_2**:** x and y coordinates of the second eye.

Align_Face(F, eye_1, eye_2): Rotates, scales and translates the face F to an idealized position according to the eye coordinates.

Input: F**:** An extracted and cropped facial image from the output of **Get_Face(P)**.

eye_1**:** x and y coordinates of the first eye.

eye_2**:** x and y coordinates of the second eye.

Output: F_n**:** The face F which has been normalized to a predefined position.

Face Recognition Modules

Project(F_n, U): Calculates the orthogonal projection of a face F_n on the space spanned by the eigenfaces in U.

Input: F_n**:** A face to be projected on the column space of U.

U**:** The eigenfaces spanning the space of known faces.

Output: **c:** A vector of coefficients needed to reconstruct, as accurately as possible, F_n using the eigenfaces in U.

Match($T, \mathbf{c}, C, U$): Finds the best match to the test face T (assumed to be normalized) by comparing its projection on U with that of all other known faces.

Input: T**:** A test face to be matched.

$\mathbf{c}$**:** The projection coefficient vector of T on the column space of U.

C**:** The projections of all other known faces on the column space of U.

U**:** The eigenfaces spanning the space of known faces.

Output: m**:** An index indicating the face in the set of known faces best matching T.

d**:** A scalar indicating the orthogonal distance of T from the column space of U.

Verify(T, n): Verifies the test face T against the claimed identity n.

Input: T**:** A test face to be verified.

n**:** The identity claim provided with the test face.

Output: v**:** The boolean verification result: “accept” or “reject”.

Bibliography

[1] F.R. Bach and M.I. Jordan, *Kernel Independent Component Analysis*, Journal of Machine Learning Research, Vol. 3, pp. 1–48, (2002)

[2] M.S. Bartlett, J.R. Movellan and T.J. Sejnowski, *Face Recognition by Independent Component Analysis*, IEEE Transactions on Neural Networks, Vol. 13, No. 6, pp. 1450–1464, (2002)

[3] C. Beavan, *Fingerprints: The Origins of Crime Detection and the Murder Case that Launched Forensic Science*, Hyperion, 77 W. 66th Street, New York, (2001)

[4] P.N. Belhumeur, J.P. Hespanha and D.J. Kriegman, *Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection*, Proc. of the 4th European Conference on Computer Vision, ECCV'96, Cambridge, UK, pp. 45–58, (1996)

[5] D.M. Blackburn, M. Bone, and P.J. Phillips, *Face Recognition Vendor Test 2000 Evaluation Report*, Sponsored by: DoD Counterdrug Technology Development Program Office, Defense Advanced Research Projects Agency, National Institute of Justice (2001)

[6] C. Boehnen and T. Russ, *A Fast Multi–Modal Approach to Facial Feature Detection*, WACV05, pp. 135–142, (2005)

[7] R.S. Boyer (Ed.), *Automated Reasoning: Essays in Honor of Woody Bledsoe*, Automated Reasoning Series, Kluwer Academic Publishers, (1991)

[8] A. Bronstein, M. Bronstein and R. Kimmel, *Expression–invariant 3D face recognition*, Proceedings Audio & Video–based Biometric Person Authentication (AVBPA), Lecture Notes in Computer Science 2688, Springer, pp. 62–69 (2003)

[9] A. Bronstein, M. Bronstein, R. Kimmel, and A. Spira, *3D face Recognition Without Facial Surface Reconstruction*, Proceedings of ECCV 2004, Prague, Czech Republic, (2004)

[10] T.F. Cootes and C.J. Taylor, *Statistical Models of Appearance for Computer Vision*, Technical Report, University of Manchester, 125 pages, (2000)

[11] T.F. Cootes, C.J. Taylor and K. Walker, *View–Based Active Appearance Models*, Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition, Grenoble, France, pp. 227–232, (2000)

[12] J.W. Demmel, *Applied Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, (1997)

[13] K. Etemad and R. Chellappa, *Discriminant Analysis for Recognition of Human Face Images*, Journal of the Optical Society of America, Vol. 14, No. 8, pp. 1724–1733, (1997)

[14] J. Fritsch, S. Lang, M. Kleinehangenbrock, G.A. Fink and G. Sagerer, *Improving Adaptive Skin Color Segmentation by Incorporating Results from Face Detection*, Proc. IEEE Int. Workshop on Robot and Human Interactive Communication, (2002)

[15] G.H. Golub and C.F. Van Loan, *Matrix Computations, Second Edition*, The Johns Hopkins University Press, 701 West 40th Street, Baltimore, Maryland 21211, (1989)

[16] R. Gonzalez and R. Woods, *Digital Image Processing, Second Edition*, Prentice Hall, Upper Saddle River, New Jersey 07458, (2002)

[17] G. Gordon, *Face Recognition Based on Depth and Curvature Features*, In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (Champaign, Illinois), pp.108–110, (1992)

[18] G. Guo, S.Z. Li and K. Chan, *Face Recognition by Support Vector Machines*, Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition, pp. 196–201, (2000)

[19] P.L. Hallinan, G. Gordon, A.L. Yuille, P. Giblin and D. Mumford, *Two– and Three–Dimensional Patterns of the Face*, A.K. Peters, Natick, Massachusetts, (1999)

[20] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision, Volume I*, Addison–Wesley, (1992)

[21] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision, Second Edition*, Cambridge University Press, (2003)

[22] B. Heisele, P. Ho and T. Poggio, *Face Recognition with Support Vector Machines: Global versus Component–based Approach*, Proc. of the Eighth IEEE International Conference on Computer Vision, ICCV, Vol. 2, Vancouver, Canada, pp. 688–694, (2001)

[23] T. Heseltine, N. Pears, J. Austin and Z. Chen, *Face Recognition: A Comparison of Appearance–Based Approaches*, Proc. VIIth Digital Image Computing: Techniques and Applications, (2003)

[24] R.L. Hsu, M. Abdel–Mottaleb, and A.K. Jain, *Face Detection in Color Images*, IEEE Transactions on Pattern Analysis and machine Intelligence, Vol. 24, No. 5, (2002)

[25] R.C.K. Hua, L.C. De Silva and P. Vadakkepat, *Detection and Tracking of Faces in Real–Time Environments*, Department of Electrical and Computer Engineering, National University of Singapore

[26] K. Jonsson, J. Matas, J. Kittler and Y.P. Li, *Learning Support Vectors for Face Verification and Recognition*, Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition, Grenoble, France, pp. 208–213, (2000)

[27] S. Kawato and N. Tetsutani, *Detection and Tracking of Eyes for Gaze–camera Control*, ATR Media Information Science Laboratories Proc. VI, (2002)

[28] C. Liu and H. Wechsler, *Comparative Assessment of Independent Component Analysis (ICA) for Face Recognition*, Proc. of the Second International Conference on Audio– and Video–based Biometric Person Authentication, AVBPA'99, Washington D.C., USA, pp. 211–216, (1999)

[29] J. Lu, K.N. Plataniotis and A.N. Venetsanopoulos, *Face Recognition Using Kernel Direct Discriminant Analysis Algorithms*, IEEE Trans. on Neural Networks, Vol. 14, No. 1, pp. 117–126, (2003)

[30] J. Lu, K.N. Plataniotis and A.N. Venetsanopoulos, *Face Recognition Using LDA–Based Algorithms*, IEEE Trans. on Neural Networks, Vol. 14, No. 1, pp. 195–200, (2003)

[31] J. Lu and S. Li, *Face Detection, Alignment, and Recognition*, Emerging Topics in Computer Vision, University of Southern California, pp. 385–455, (2004)

[32] A.M. Martinez and A.C. Kak, *PCA versus LDA*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 23, No. 2, pp. 228–233, (2001)

[33] K. Messer, J. Matas, J. Kittler, J. Luettin and G. Maitre, *XM2VTSDB: The Extended M2VTS Database*, Second International Conference on Audio and Video–Based Biometric Personal Authentication (AVBPA'99), (1999)

[34] H. Moon and P. Phillips, *Computational and Performance aspects of PCA–based Face Recognition Algorithms*, Perception, Vol. 30, pp. 303–321, (2001)

[35] Y. Moses, Y. Adini and S. Ullman, *Face Recognition: The Problem of Compensating for Changes in Illumination Direction*, Proceedings of the European Conference on Computer Vision, vol. A, pp. 286–296, (1994)

[36] S.G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw–Hill Series in Industrial Engineering and Management Science, (1996)

[37] A.V. Nefian, *Embedded Bayesian Networks for Face Recognition*, Proc. of the IEEE International Conference on Multimedia and Expo, Vol. 2, Lusanne, Switzerland, pp. 133–136, (2002)

[38] A.V. Nefian and M.H. Hayes III, *Hidden Markov Models for Face Recognition*, Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 2721–2724, (1998)

[39] A.V. Nefian and M.H. Hayes, *Maximum Likelihood Training of the Embedded HMM for Face Detection and Recognition*, Proc. of the IEEE International Conference on Image Processing, ICIP 2000, Vol. 1, Vancouver, BC, Canada, pp. 33–36, (2000)

[40] F. Noo, M. Defrise and H. Kudo, *General Reconstruction Theory for Multislice X–ray Computed Tomography With a Gantry Tilt*, IEEE Transactions On Medical Imaging, Vol. 23, No. 9, pp. 1109–1116, (2004)

[41] A. Pentland and M. Turk, *Eigenfaces for Recognition*, Journal of Cognitive Neurosicence, Vol. 3, No. 1, pp. 71–86, (1991)

[42] A. Pentland and M. Turk, *Face Recognition Using Eigenfaces*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Maui, Hawaii, USA, pp. 586–591, (1991)

[43] A. Pentland, B. Moghaddam and T. Starner, *View–Based and Modular Eigenspaces for Face Recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, Washington, USA, pp. 84–91, (1994)

[44] P.J. Phillips, A. Martin, C.L. Wilson and M. Przybocki, *An Introduction to Evaluating Biometric Systems*, National Institute of Standards and Technology, (2000)

[45] H. Pottmann and M. Desbrun (Editors), *A Reflective Symmetry Transform*, Eurographics Symposium on Geometry Processing, pp. 1-11, (2005)

[46] S.A. Rizvi, P.J. Phillips and H. Moon, *The FERET Verifcation Testing Protocol for Face Recognition Algorithms*, NISTIR 6281, National Insttute of Standards and Technology, (1998)

[47] K. Sandeep and A.N. Rajagopalan, *Human Face Detection in Cluttered Color Images Using Skin Color and Edge Information*, Department of Electrical Engineering, Indian Institure of Technology – Madras

[48] B. Scholkopf, A. Smola and K.R. Muller, *Nonlinear Component Analysis as a Kernel Eigenvalue Problem*, Technical Report No. 44, 18 pages, (1996)

[49] C. Sindale, *Handwritten Signature Verification Using Hidden Markov Models*, Department of Engineering, University of Stellenbosch, (2003)

[50] S. Singh, D.S. Chauhan, M. Vasta and R. Singh, *A Robust Skin Color Based Face Detection Algorithm*, Tamkang Journal of Science and Engineering, Vol. 6, No. 4, pp. 227–234, (2003)

[51] H. Späth, *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*, Ellis Horwood, (1980)

[52] G. Strang, *Introduction to Linear Algebra*, Wellesley–Cambridge Press, Wellesley, MA 02181, (1993)

[53] L.N. Trefethen and D. Bau, *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, (1997)

[54] V. Vezhnevets, V. Sazonov, and A. Andreeva, *A Survey on Pixel–Based Skin Color Detection Techniques*, Proc. Graphicon, (2003)

[55] L. Wiskott, J.M. Fellous, N. Krueuger and C. von der Malsburg, *Face Recognition by Elastic Bunch Graph Matching*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, pp. 776–779, (1997)

[56] L. Wiskott, J.M. Fellous, N. Krueuger and C. von der Malsburg, *Face Recognition by Elastic Bunch Graph Matching*, Chapter 11 in Intelligent Biometric Techniques in Fingerprint and Face Recognition, eds. L.C. Jain et al., CRC Press, pp. 355–396, (1999)

[57] C. Wu, C. Liu, H.Y. Shum, Y.Q. Xu and Z. Zhang, *Automatic Eyeglass Removal from Face Images*, The 5th Asian Conference on Computer Vision, (2002)

[58] S.T. Wu and M.R.G. Márquez, *A non–self–intersection Douglas–Peuker Algorithm*, In Sibgrapi, pp 60–66, (2003)

[59] M.H. Yang, *Face Recognition Using Kernel Methods, Advances in Neural Information Processing Systems*, T. Diederich, S. Becker and Z. Ghahramani, Eds., vol. 14, 8 pages, (2002)

[60] M.H. Yang, *Kernel Eigenfaces vs. Kernel Fisherfaces: Face Recognition Using Kernel Methods*, Procedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 215–220, (2002)

[61] H. Zabrodsky, S. Peleg and D. Avnir, *Symmetry as a Continuous Feature*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 12, pp. 1154–1166, (1995)

[62] B.D. Zarit, B.J. Super and F.K.H. Quek, *Comparison of Five Color Models in Skin Pixel Classification*, ICCV99 Intl Workshop on recognition, analysis and tracking of faces and gestures in Real–Time systems, pp. 58–63, (1999)

[63] W. Zhao, R. Chellappa and A. Krishnaswamy, *Discriminant Analysis of Principal Components for Face Recognition*, Proc. of the 3rd IEEE International Conference on Face and Gesture Recognition, FG'98, Nara, Japan, p. 336, (1998)

[64] S. Zhou and R. Chellappa, *Multiple–exemplar Discriminant Analysis for Face Recognition*, Proc. of the 17th International Conference on Pattern Recognition, ICPR'04, Cambridge, UK, pp. 191–194, (2004)

[65] S. Zhou, R. Chellappa and B. Moghaddam, *Intra–personal Kernel Space for Face Recognition*, Proc. of the 6th International Conference on Automatic Face and Gesture Recognition, FGR2004, Seoul, Korea, pp. 235–240, (2004)

[66] *http://www.face–rec.org*

[67] *http://www.frvt.org/*

[68] *http://www.ee.surrey.ac.uk/Research/VSSP/xm2vtsdb/.*