DRUBIS: A Distributed Face-Identification Experimentation Framework-

Design, Implementation and Performance Issues

A thesis submitted in fulfilment of the requirements for the degree of

MASTER OF SCIENCE

RHODES UNIVERSITY

South Africa

By

MBONELI NDLANGISA

November 2003

Abstract:

We report on the design, implementation and performance issues of the DRUBIS (Distributed Rhodes University Biometric Identification System) experimentation framework. The Principal Component Analysis (PCA) face-recognition approach is used as a case study. DRUBIS is a flexible experimentation framework, distributed over a number of modules that are easily pluggable and swappable, allowing for the easy construction of prototype systems. Web services are the logical means of distributing DRUBIS components and a number of prototype applications have been implemented from this framework. Different popular PCA face-recognition related experiments were used to evaluate our experimentation framework. We extract recognition performance measures from these experiments. In particular, we use the framework for a more in-depth study of the suitability of the DFFS (Difference From Face Space) metric as a means for image classification in the area of race and gender determination.

Acknowledgements:

This project was made possible by the support from a number of people: my family, Rhodes University Computer Science Department staff, friends and many more.

- I would like to express my sincere gratitude to my supervisor, Prof. E.P. Wentworth for the privilege of working under his supervision. I am deeply indebted to him for his constant support and guidance throughout this research, his critical comments and suggestions and his ever-willing assistance during the process of producing this thesis. Thank you for your unfailing enthusiasm, encouragement and genuine interest in my academic work. You did not only guide me through this project but you also equipped me with life-long research skills.
- Thanks to Madeleine Wright for lending me her exceptional proof reading skills. Your time and effort in improving my grammar has greatly improved the standard of English in this thesis.
- Thanks to the Computer Science 3 class of 2002 for allowing me to use their images in this research.
- Thanks to Matthew Turk and Alex Pentland for developing the MIT Principal Component Analysis (PCA) face-recognition system and making it freely available.

- Thanks to Mike Krueger for developing the SharpZip library and making it freely available.

- I am grateful to the support that the Computer Science support staff has given to me, specifically the following: Jody for technical assistance, Cheryl and Tina for administrative help.
- Thanks to Telkom (SA) for the financial support over the years and for allowing me to undertake this project. Without your support I was not going to reach this far. Thank you!

Lastly thanks to all of those that directly or indirectly contributed to the success of this project. Your inputs are gratefully acknowledged.

----Mboneli Ndlangisa

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

At the high-security end of the spectrum, biometric recognisers, particularly face recognisers, still have serious deficiencies. However, in spite of disappointing "high-end" results, we are of the opinion that there are a number of low-risk situations where current, imperfect biometric recognisers would be useful. Some scenarios in which we could tolerate sloppy matching include: controlling the use of a photocopier machine by staff in a small department; verifying class attendances; and perhaps helping the reception desk in a busy academic department to identify students.

We aim to use PCA- (Principal Component Analysis) based face recognition for a relaxed accuracy problem. We define a relaxed accuracy environment as an application domain where a mismatch will not have the detrimental effects it might have in high security environments. Hence, our study differs from the rest in the sense that we are willing to accept "sloppy" matches. We define a sloppy match as one in which the false acceptance rate or the false rejection rate, or the rate at which the system is unable to make a determination is higher than might usually be acceptable. Using a PCA-based system in this way steps outside the usual domain of face-recognition applications; it imposes new characteristics for the target application area, (e.g. a household or an office with few known individuals) and places convenience before accuracy.

Our first aim in this project is to build up a longer-term experimental infrastructure, and incrementally grow our expertise in the field. We implement a framework for experimentation that allows pluggable modules, and the easy construction of prototype systems.

Overall this project is aimed at studying existing face-recognition technologies so that we might better understand the state of the art; developing an experimental framework; investigating the implementation trade-offs involved in developing a simple PCA-driven face-recognition system; creating a face database, and assessing the performance of the system against our face database. We implement our system on a Microsoft .NET architecture, using C# as an implementation programming language. Microsoft .NET XML Web Services are used to distribute and link different system components.

1.2 Scope

This project does not aim to reproduce work that is already covered in face-recognition literature. Our study focuses on PCA-driven face-recognition algorithms. PCA-based systems were chosen purely on the grounds of their success and wide deployment, and the wide acceptance of PCA as a baseline technique against which more advanced systems are often benchmarked. We propose a simple XML web service-driven distributed framework that exposes the low-level building blocks of a face-recognition system. Our system, the Distributed Rhodes University Biometric Identification System (DRUBIS), is aimed at an academic audience i.e. the research community, and does not claim to be a real-world application.

Our goal is not to produce a super face-recognition system, but to understand how current systems operate and to gain some insight into the difficulties they raise and into the reasons for so many reported failures in face-identification systems [Dearne, (2003)]. We use our own face database to compare results with those obtained by Turk and Pentland (1991a) at MIT. This helps us to understand the effects of factors such as image quality and preprocessing on the overall face-recognition process.

In summary, this project implements a face-recognition framework by logically distributing a PCA algorithm as a number of independent, flexible and easily swappable components. An open-source face-recognition algorithm is used as a basis for our work [Turk and Pentland (1991a)]. Factors influencing face-recognition performance are exposed and studied in depth. The possible extension of the role of face recognisers to image classifiers is also examined.

1.3 Conventions and terminology used

1.3.1 Training, gallery and probe sets

In a face-recognition process there are usually three sets of images involved: a training set, a gallery set and a probe set.

- A *training set* of images includes those images used for training the system, i.e. setting up or refining some framework by which the images will be represented. Representing a specific face in the framework generates a *signature* which may be used both for enrolment and for recognition Training takes place prior to any signature generation.
- A *gallery set* is a set of known images that have been enrolled with the system, i.e. their signatures have been computed and stored. The gallery set often overlaps with the training set. If an image which was part of the original training set is enrolled, we expect it to have a more accurate representation than what we might get if an image which is not part of the original training set is enrolled. (We will cover this in more detail in due course. In our system the gallery and the training set are usually the same set of images.)
- A *probe set* is a set of images that is used for querying against the enrolled gallery set.

1.3.2 Image identification and classification

We use these two terms for different types of recognition.

Image identification means recognising a probe image in the gallery-set images, i.e. given a probe face and a set of enrolled faces in a database; find the N enrolled faces closest to the probe. A face-recognition system usually assists a human to determine the identity of the probe (test) face by computing closeness metrics between the probe and each gallery face stored in the system database, and by ranking them [Kotropoulus et al, (1999)].

Image classification means classifying an image according to a group without considering its identity. A good example is deciding if a given image belongs to a male or female. The success of recognition is measured by the system's ability to pick a correct image cluster.

1.3.3 FERET

The Face Recognition Technology (FERET) program is a US initiative sponsored by the Department of Defence Counterdrug Technology Development Program through the Defence Advanced Research Projects Agency (DARPA), with the U.S. Army Research Laboratory (ARL) serving as technical agent.

According to Phillips et al (1996), the primary mission of FERET is to develop automatic face-recognition capabilities that can be employed to assist security, intelligence, and law-enforcement personnel in the performance of their duties. In order to achieve its objectives, the FERET program conducted multiple tasks over several years from September 1993.

To date the program has focused on three major tasks:

- The first major FERET task was the development of the technology base required for a face-recognition system. This was achieved by sponsoring research groups to develop face-recognition algorithms.

- The second major task was the collection of a large database of facial images (the FERET database). The FERET database remains a vital part of the overall FERET program, because it provides a standard database for algorithm development, testing, and evaluation. The database is divided into two parts: the development portion, which is given to researchers, and the sequestered portion, which is used to test algorithms. Images in the FERET database were taken under only semi-controlled conditions. This is in contrast to many of the algorithms in the literature, where results are reported for small databases collected under highly controlled conditions.

- The third major ongoing task is the government-monitored testing and evaluation of face-recognition algorithms using standardised tests and test procedures. The purpose of the tests is to measure overall progress in face recognition, determine the maturity of face-recognition algorithms, and have an independent means of comparing algorithms. The tests measure the ability of the algorithms to handle large databases, changes in people's appearance over time, variations in illumination, scale, and pose, and changes in the background. The algorithms tested are fully automatic, and the images presented to the algorithm are not normalised. If an algorithm requires that a face be in a particular position, then the algorithm must locate the face in the image and transform the face into the required position.

In our work we do not use the FERET database, but some face-recognition results reported in the literature do use the FERET database. We use the FERET results for reference in our experiments.

We also use the FERET evaluation protocol as a guideline in our face-recognition experiments, in view of its widely accepted methodology in the face-recognition research community.

1.4 Summary of the chapters

Chapter 2 introduces face recognition and lays a foundation for the rest of the thesis. It covers the background information on face-recognition algorithms, the steps involved in their operation, and different approaches to the design of face-recognition algorithms. One very popular approach to face recognition, PCA, is covered in depth, and some hybrid systems that use PCA are also discussed.

Chapter 3 looks at our implementation environment. We see long-term flexibility and advantages in structuring our experimental framework as a collection of loosely-coupled components. Web services are introduced as the platform for this decoupling and distribution. The role that web services can play in the face-recognition framework forms the theme of this chapter. The trade-offs of using an XML web service for face recognition are discussed.

Chapter 4 gives a detailed analysis of the design and implementation of DRUBIS. Different components implemented as part of this project are detailed in this chapter. These include simple web service-driven Windows applications, real-time face recognisers, our evaluation toolbox and our system for third-party support. The chapter starts by looking at framework components necessary to support biometric systems in general, and later focuses more specifically on the framework components needed to support PCA face recognition. XML web services are used to distribute our system. DRUBIS is designed to support guest users' applications with the aim of encouraging research in the field. With this framework established, we are in a position now to turn our attention to biometric identification.

Chapter 5 moves into "proof of concept": can our framework provide support for experiments that will deepen our understanding of the key factors and allow us to experiment with different aspects of the problem? The chapter looks at one of the most important factors in face-recognition algorithms – face- space quality. This chapter explores the use of visual eigenface (the entities in terms of which a signature is computed) information to start to understand what constitutes a good or a poor face space. The relationship between a poor face space and the recognition scores is examined using scaled and unscaled Rhodes University face images and MIT face images. Visual eigenface information can help explain poor performance of a face-recognition system or can serve as a simple and cheap guideline to improve recognition scores.

Chapter 6 uses the framework to reproduce a set of experiments using well-published techniques from the literature to evaluate the performance of our face-recognition system [Phillips et al (2000)]. This chapter differs from work by other researchers in that it is aimed at unearthing the primary considerations that a developer must be aware of when implementing PCA recognisers. It statistically examines the quality of training-set images (i.e. face-space images), the gallery and probe-set images, the optimum face-space size, and eigenface-subset selection techniques. In short, this chapter brings together and organises different aspects of face-space recognition from various published sources.

Chapter 7 again uses the framework for a set of experiments in which we examine the use of DFFS- (Difference From Face Space) values from a PCA algorithm in classifying images. The literature suggests that a face-recognition problem can be simplified by first grouping images according to race, gender or age. This chapter explores this concept. While not all groupings are effective, in our experiments we go further than previous work to illustrate that PCA algorithms can be effectively used to classify images according to their object class i.e. in differentiating faces from non-faces.

Chapter 8 is a conclusion chapter. It summarises the whole thesis, highlighting our own contributions and limitations and suggesting future research directions.

Chapter 2

Overview of Face-Recognition and Related Technologies

2.1 Introduction

Face recognition has been one of the active research fields in computer vision in the past two decades [Zhao et al (2000), Gross et al (2001), Turk (2001), Kotropoulus et al (1999)]. In spite of all the effort put into finding face-recognition algorithms capable of handling real-world identification problems, face recognition is still far from being a solved problem. This project contributes to this pool of knowledge.

The project does not aim to produce or contribute towards the production of a new face-recognition algorithm, but rather it utilizes the existing algorithms, PCA-based algorithms to be specific, for solving a relaxed-accuracy-type problem. In short, the design of a new face-recognition algorithm is outside the scope of this project.

This chapter covers broader aspects of face-recognition technologies and attempts to highlight the significance of our study in the field of biometrics.

2.2 Biometric recognition

There are many variations of biometric systems, but they all follow the same framework [Zhang (2000), Hong et al (1999)] that we have chosen to use which consists of two main phases. First there is an *enrolment* phase: the biometric aspect (a fingerprint, retina, face, voiceprint, etc) is presented to the biometric system. The system captures the biometric information necessary for later recognition, encodes this information in some way and stores the registered feature or template. Thereafter there is a *recognition* phase: the same procedure is followed of presenting the biometric aspect to the system and generating the biometric signature. This time the goal is to match it with enrolled signatures with the aim of establishing its identity. In this project we specifically look at face recognition.

Face recognition is a useful domain to explore because, although simple technologies are used, there are few quality off-the-shelf systems available. Standard video-capture hardware (a webcam in our case) is used to capture facial images and a face-image signature is generated. The exposure of low-level building blocks in a face-recognition system makes it a best contender for our study. The flexibility offered by this low-level approach is not possible with some biometric technologies that use dedicated hardware.

The distributed nature of our system (using web services) makes it different from most of the previous work in biometrics. Our distributed framework uses face recognition as a specific proof-of-concept for our biometric framework. But the framework supporting our face-recognition system is applicable to a larger set of biometric recognisers. Our structure and organisation can also be compared to and contrasted with off-the-shelf biometric software. A modern systems-development environment is characterised by different platforms working towards the same goal. Biometric systems development should also benefit from this organisation. A solution using Microsoft .NET XML web services is explored in this project.

2.3 Face-signature encoding schemes

A number of face-signature encoding schemes (also known as face-feature or feature-signature schemes) have been produced. These schemes are all aimed at efficiently and effectively representing unique features of face images. These schemes facilitate the recognition problem later on. Pandya and Szabo (1999), and Campos et al (2000) classify these encoding schemes into two broad classes, each best suitable for a certain family of object-recognition problems. These classes are:

- Syntactic or structural (based on face geometry or face patterns),
- Statistical (information-theory based i.e. they seek to encode information that differentiates one image from another).

In this project we look at only statistical encoding schemes using single-view-based images. Face-recognition algorithms are very sensitive to view changes and they differ in how they handle changing views.

2.3.1 Syntactic or structural?

Earlier approaches in face recognition relied on detecting individual facial features such as the nose, the eyes or the mouth, and defining a face model by the position, size and relationship among these features [Turk and Pentland (1991a), Zhao et al (2000), Howell (1999)]. These systems are also called geometric systems. Lam and Yan (1998) refer to geometric-based systems as analytic systems since they consider geometric patterns of the face and use the face's fiducial features as an aid in face identification.

In some structural recognition algorithms, the features are not important in themselves but the critical information regarding pattern class, or pattern attributes, is contained in the structural relationship among the features [Pandya and Szabo (1999)]. This is sometimes referred to in the literature as a template-based approach, and we adopt the term with this meaning in this work. Applications involving the recognition of pictorial patterns that are distinguished by their shapes, such as faces and characters (e.g. 'a', 'b', 'c', and handwriting), fall into this category. Structural relationships are very important in classifying face images from non-face images. Elastic graph matching [Wiskott et al (1999)] is one effective technique that matches according to structure**.** In the syntactic approach, the patterns are represented in a hierarchical fashion, and are viewed as being composed of sub-patterns. A set of rules governing the syntax is called a grammar and these methods are capable of handling structural information that is lacking in statistical approaches [Pandya and Szabo (1999)].

It has proved hard to extend the face-geometry based systems to include multiple views and the systems themselves have often been quite fragile. Turk and Pentland (1991b) allege that research into human strategies for face recognition has shown that individual geometric features such as the mouth, the eyes and the nose, and the distance between them, is insufficient to account for how well humans identify faces. Haddadnia et al (2002) also share this observation; they argue that it has been shown that the structure-based approaches, using explicit modelling of facial geometric-features, have been troubled by the unpredictability of facial appearance under changing environmental conditions e.g. changing light conditions. Another disadvantage of picture-plane measurements (geometric templates) is that it is not obvious which features and configurable information will categorise a face efficiently and accurately [Howell (1999)]. Despite the speculations of Turk and Pentland (1991b) about the limited use of this approach, Yoshino et al (2001) have implemented a computer-assisted facial-image identification system using face templates. The elastic graph-matching scheme from the University of Southern California (USC) competed in the FERET '96 evaluations.

2.3.2 Statistical

The most general encoding scheme used to formulate solutions to face-recognition problems is a statistical approach [Pandya and Szabo (1999), Turk and Pentland (1991a)]. This is sometimes referred to in the literature as a "statistical feature-based" approach. There are a number of statistical signature-representation schemes in this category and PCA is one of the more popular ones [Howell (1999), Zhao and Chellapa (2002), Turk and Pentland (1991a), Campos et al (2000)]. The statistical approaches do not attempt to handle pattern structures and their relationships. Statistical approaches are referred to as holistic since their matching is based on global properties of face patterns [Lam and Yan (1998), Fang et al (2002a)]. They are considered inferior to the syntactic approaches for handling pattern structures and relationships [Pandya and Szabo (1999)].

Turk and Pentland (1991a) define the statistical-based category as an information-theory-based approach, which seeks to encode only the most relevant information from a group of faces in order better to distinguish them from one another. The advantage of this approach lies in the way it makes it possible for new faces to be recognised automatically. Some successful, commercial face-recognition applications use this approach for face-signature encoding, such as the Viisage face-recognition technologies [Viisage (2002)].

The statistical approaches capture the unique discriminating statistical *features* of an object, but cannot describe the structural *relationship* between these features and the object in question (a face in this instance). [Note: a statistical feature refers to a statistical-signature representation of the face as opposed to a geometric representation of facial features such as the nose and eyes.] The failure to describe the relationships implies that a face-recognition system has no inherent understanding of the objects it recognises. It could just as well be trained with modes of transport, or types of fruit, and asked to recognise these.

The interesting aspect of these algorithms is their ability to be extended to handle three-dimensional data and multiple-view images. Phillips et al (2003), in the official report of the 2002 Face Recognition Vendor Test results, indicate an improvement in handling non-frontal images after extending 2-D images to 3-D through morphing (transforming one image to another). The issue of matching faces with the differences that spring from positional variation is being actively researched and is critical for surveillance and tracking systems.

In summary we have 3 approaches:

1) *Geometric/structural* refers to part or geometric-feature identification.
2) *Template matching* involves the relationships between the geometric-features.
3) The *statistical* approach is based on information theory.

In practice, hybrid approaches are possible. For example, in the next section we will discuss a hybrid approach that decomposes a face structurally into eyes, nose and mouth, and then uses a statistical PCA approach in an attempt to improve matching.

2.4 What to encode to a signature? – global vs. local information

Signature encoding schemes for face recognition have been very actively researched. Their popularity has been fuelled by a need to detect and measure salient facial points. For a face-recognition system to perform effectively, it has to isolate and extract the salient statistical features from its input data in order to represent the face efficiently [King and Xu (1997), Howell (1999)]. King and Xu (1997) refer to the orthodox approach developed by Turk and Pentland (1991a), using global image information. This approach includes image background information as well as facial details not critical for face recognition. The alternative to the orthodox approach is the PCA methodology that uses localised face information. These are the two significant approaches to signature encoding: the information can be either global to the whole image (including its background), or to the overall face; or it can be decomposed structurally to local parts of the face such as the eyes, nose, mouth.

Fang et al (2002a) and Zhao et al (2000) maintain that both global (overall face with no background) and local (parts of the face) face information is critical for identification. When meeting different individuals, people unconsciously select and combine the salient features for purposes of identification. Different subjects may be seen to have different salient features, and the adoption of a common-feature combination approach for all subjects may be at risk of losing useful information carried by salient individual features. Fang et al (2002a) and Fang et al (2002b) advocate a personalised feature combination – i.e. a separate mix of local and structural data per individual - for face recognition.

2.4.1 Global-encoded signatures

One of the weaknesses of the PCA algorithm developed by Turk and Pentland (1991a) is its computation of face-image signatures from global face-image information. A global-face-image information-based signature is a signature computed from a face image that includes background information. This is not an effective mechanism for extracting face signatures since a lot of image data that is irrelevant to the facial portion, such as hair, shoulders and background, may contribute to the creation of erroneous feature vectors (image signatures) that can have a deleterious effect on recognition results [Haddadnia et al (2002)]. If the face forms only a minimal portion of the image, it will be effectively outweighed by irrelevant data, which will cause the results to be based on an erroneous signature. This is evidenced by the following image examples:

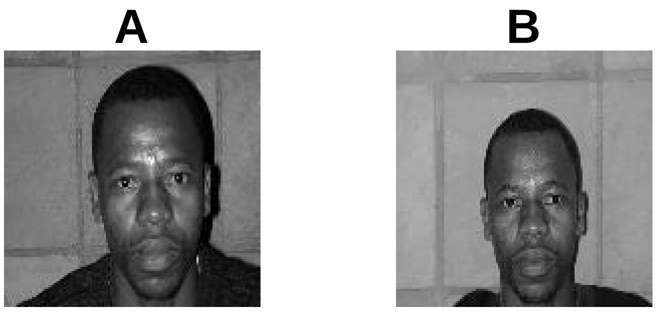


Figure 2a: The influence of global image information in feature extraction

The above diagram looks at two images of the same individual. Image A is in our probe set and image B is in our gallery set. There is a huge difference between the two images in the proportion of the area occupied by the face. Turk and Pentland (1991b) highlight the effect of face size in relation to the overall image as one of the major factors affecting recognition performance. It is almost certain that an algorithm without knowledge of structural face information will struggle to match the two images.

A number of different approaches have been suggested to remedy this problem: these include cropping a face part from an image or generating signatures for different face parts, such as the eyes only, or the eyes with the nose and mouth. These implementations are discussed in the following sections.

2.4.2 Local-encoded signatures

Local face-image-based feature extractors attempt to address the weaknesses of global face-image-based systems. The original PCA approach, in which the whole face is considered, serves as the baseline against which improvements can be measured. Local face-image information-encoded signatures are computed from the face *excluding* irrelevant image information. The central idea is to eliminate image information that is irrelevant to recognition.

Phillips et al (2000) describe face-recognition algorithms as being made up of two stages: face detection and normalisation, and face verification. This description emphasises the importance of eliminating noise from irrelevant face-image data such as background, hair or shoulders. It also stresses the need for image preprocessing to capture salient face features. In our implementation of DRUBIS we designed a face uprighter module that first crops the face from an image, and then scales it and rotates it if necessary.

2.4.2.1 The eigenfeature approach: using a single face-image signature

A variety of solutions have been proposed and some of these have been proved to be more effective than others [Campos et al (2000), Haddadnia et al (2002), Pentland et al (1994), King and Xu (1997)]. In their search for an efficient, effective and computer-inexpensive (in the use of resources) statistical feature-extraction mechanism for face-image signatures, the findings of Campos et al (2000) highlight a huge improvement over the global-feature-based systems discussed above.

In the vanilla PCA algorithm, which is based on the whole image as opposed to concentrating only on salient features, the term *eigenface* is used to refer to the principal components – the entities in terms of which a signature is computed [Turk and Pentland (1991a)]. The term *eigenfeature* has been used to refer to the application of PCA in restricted areas of the image in order to obtain the main components or feature points of the face, such as the mouth (*eigenmouth*), the nose (*eigennose*), and the eyes (*eigeneyes*). This approach is based on a modular breakdown of the face into smaller features and computing smaller signatures based on these features i.e. *eigeneyes*, *eigennose* and *eigenmouth*. Pentland et al (1994) have generalised this approach to include view-based (frontal, side, etc.) and modular (*eigenfeature*) *eigenspaces* for detection and recognition.

Campos et al (2000) define *efficiency* as the extent to which only the relevant face-image data is encoded in the face signature. They improved on global PCA efficiency by cropping only the portion of the face around the eyes and using this for signature extraction. They use the term *eigeneyes* rather than *eigenfaces* to denote the principal components of their cropped images.

They also define *effectiveness* as the extent to which the matching has improved, and confirm that the eigeneyes approach is more effective than global PCA. Pentland et al (1994) support this finding.

The cropping has another useful side-effect in that it is computationally less demanding. In the study performed by Campos et al (2000) a 512 by 342 pixel full-face image was reduced to a 64 by 64 pixel image of the eyes. This is a huge improvement in terms of the pixel processing demanded by the PCA method. King and Xu (1997) use salient features (eyes, nose, etc) of size 32 by 32 which is one quarter of the cropped image sizes produced by Campos et al (2000). Overall, this approach also proves to be inexpensive in terms of processing time.

Haddadnia et al (2002) improve on the findings of Campos et al (2000) by creating a sub-image that neutralises the irrelevant information from the face-input image. This solution addresses only the first two issues highlighted by Campos et al (2000): efficiency and effectiveness. The sub-image creation process involves face-portion localisation in a face image and the cropping of the whole face image, which is then inserted onto a blank image (a white background). The idea here is that irrelevant information such as the white background will be there but, because it will be constant, will have less effect on recognition results. The performance of this approach is hard to assess independently, since Haddadnia et al (2002) used it together with other face-extraction systems to have an N-feature neural-network human-face recognition. Their experimental findings do not allow one to independently assess either the contributions of their preprocessing algorithms, or the contribution of their neural network recognition system.

2.4.2.2 Local face-information encoding: an example

A face processor from the MIT vision and modelling (VISMOD) group applies the same masking technique [VISMOD (2002)] as that described above. The ultimate goal is effectively to detect the face and then exclude irrelevant information when coding a face signature. The process below resembles that of Haddadnia et al (2002). The following images show the underlying steps followed in this process:

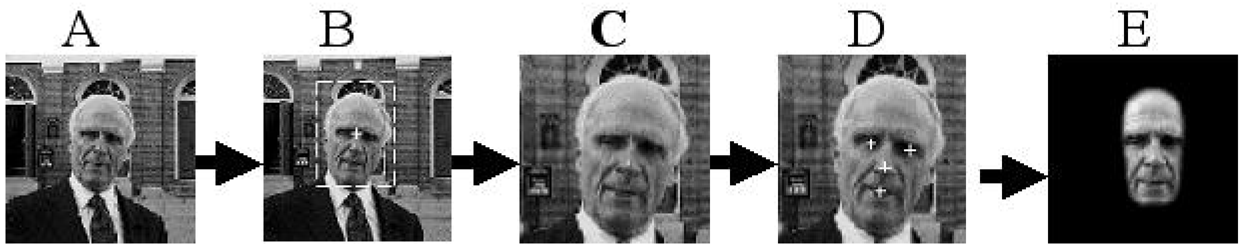


Figure 2b: The process followed in finding local face-image features

In figure 2b above, A is the original input image presented to the system; the multi-scale head-search algorithm locates the estimated head as shown in B; a head-cantered image is produced at C; face-feature estimations are computed as portrayed in D, where eyes nose and mouth are marked; and the last step is to mask and straighten the head. The resulting image is shown in E above. This final image is normalised by having some of the background removed and is then fed to the PCA to generate a corresponding feature signature.

2.4.2.3 The eigenfeature approach with different signatures for different face features

This approach is based on computing multiple signatures for the same image, using different views of the image. View-based approaches address the problem that PCA is very sensitive to where the camera is relative to the face, e.g. a full-frontal view, a view from the left or a view from the right. The view-based approach advocates different face spaces for different face views. The training images are therefore grouped into left, right, frontal, etc. views before the face spaces for each group of images are constructed. The view-based approach improves the confidence intervals of the system when deciding on a match.

Before the face match for a particular image can be decided, a DFFS (Difference From Face Space) metric is computed for the image in each of the view-based face spaces. Only one of the available view-spaces – the one that returned the smallest residual error (according to the DFFS metric) – is then used to attempt the match. The chosen face space best encodes the salient features of the probe image [Pentland et al (1994)].

The decision on whether the best face space (i.e. the one with the smallest DFFS) is useful at all is based on a defined threshold. This threshold might be tuned to decide if the given image is a face image or not. A huge DFFS is a good indication of a non-face image. More on DFFS is covered in chapter 7 of this work.

Pentland et al (1994) take this analogy further by using the DFFS computed from face-image features to decide if a given eigenfeature is an eigeneye or eigennose or none of the salient features. This is achieved by computing small, geometric, feature-based face spaces. Thus the DFFS mechanism is used to identify the kind of eigenfeatures in a face image. A feature-distance map is created by computing a DFFS at each pixel.

Interestingly the following pattern was observed from the recognition results; the use of global face image features alone performed the worst, followed by the eigenfeatures approach using local face features and the best performance was achieved when using combined local and global face-image information [Pentland et al (1994)].

2.5 Feature-extraction algorithms

This section looks at different feature-extraction (signature-extraction to be specific) algorithms that are generally used in face recognition. There are a significant number of face feature-extraction algorithms. An in-depth study of some of these algorithms can be found in Zhao et al (2000), *Face Recognition: Feature vs templates* [Brunelli and Poggio (1993)], *Face Detection* [Frischholz (2003)], and Wiskott's *Face Bibliography* [Wiskott (2003)] web sites. Colorado State University hosts a face-recognition algorithms evaluation web page with downloadable code for individual use, and this is a very useful resource for those experimenting with different face-processing algorithms [Beveridge and Draper (2003)].

Feature-extraction algorithms can be broadly classified into three categories: statistical, syntactic or graph-based, and neural-network-based algorithms [Lin (2000), Zhao et al (2000), Pandya and Szabo (1999)]. Each category is made up of a number of algorithms and there are sometimes subcategories as in neural networks.

2.5.1 Structural-based representation

Structural-based algorithms generate a face signature as a graph uniquely representing a face. The graph is made up of nodes and edges. Nodes are usually the salient face points like the eyes, the corners of the mouth, the tip of the nose, etc and the edges are generated from computing the distances between the nodes. A face is first represented as a grid with N nodes and E edges, where N is the number of nodes in a grid [Wiskott et al (1997), Leung et al (1995)]. The following diagram from Wiskott et al (1997) highlights the salient points (nodes) and the relation between them (edges).

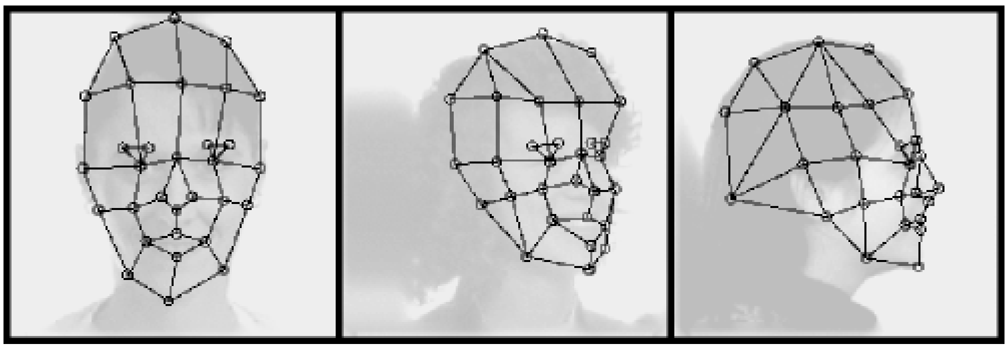


Figure 2c: Nodes and edges chosen for graphic representation of a face

2.5.1.1 Elastic-bunch graph technique

Wiskott et al (1999) define this technique as the basic process for the comparison of graphs with images and for the generation of new graphs. In its simplest version, a single labelled graph is matched onto an image. The labelled graph has a set of nodes arranged in a particular spatial order. Each node is called a jet, which is a set of values describing a small patch of grey values in an image around a given pixel. It is based on a wavelet transform (in this case a transform called the Gabor-wavelet transform) of the image (with patches described). The set of convolution coefficients for kernels of different orientations and frequencies at one image pixel comprise the jet.

2.5.1.2 Face Bunch Graphs

A *Face Bunch Graph* (FBG) is a general representation of faces used to extract image graphs automatically. An FBG covers a wide range of variations in face appearances, such as differently shaped eyes, mouths, variations due to race, sex etc. It would be too expensive to cover each feature combination with a separate graph. Instead, a representative set of M individual model graphs $\mathbf{G}^{Bm}(m = 1,...M)$, in which B represents a set of jets corresponding to one fiducial point called a bunch (e.g. an eye bunch), are combined into a stack-like structure, called a face bunch graph (FBG). The following diagram can best represent a face bunch graph:

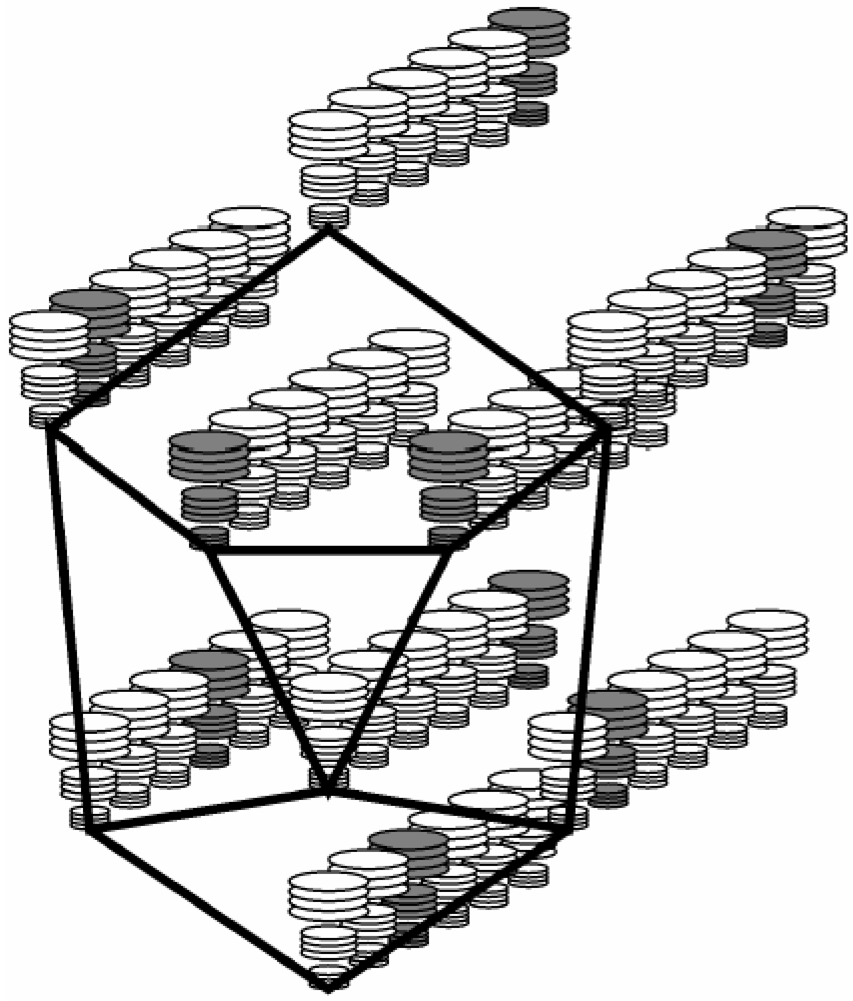


Figure 2d: A sample of a face bunch graph with jets at salient face points

In this diagram, each salient point covers a patch (a group of pixels), and has 6 stacks that represent possible choices of variations. It is helpful to think of this as a face overlaid with a graph, with the points of the central triangle representing the patches around the eyes and mouth. The shaded stack attached to each salient point (i.e. eyes, mouth, chin, etc) denotes the best-fitting stack for a particular image. Each stack is the set of coefficients for one jet; the differently sized disks denote the different resolution coefficients that are being described by the wavelet transform.

Each model graph has the same grid structure and the nodes refer to identical salient points. A bunch is a set of jets referring to one salient point [Wiskott et al (1997)]. A mouth bunch might include jets representing, for example, *open*, *male*, *closed*, *smiling*, to cover some possible mouth variations. For each face image a graph with salient points represented as best-fitting jets is produced. This graph then becomes a signature for the specific image [Wiskott et al (1995)].

2.5.1.3 Elastic bunch graph training

The first step is to train the FBG. This is done by manually locating salient points on the face and then defining a relation between different nodes and poses: different graphs represent different poses. Once the system has an FBG, graphs for new images can be generated automatically. We refer to these new graphs as image-graph signatures to avoid confusion with the graphs used for FBGs, since they uniquely represent an image or they are used to match an image. Wiskott et al (1997) warn that, when the FBG contains only a few faces, it is necessary to review and correct the resulting graph signatures but, once the FBG is rich enough (with approximately 70 graphs), one can rely on the graph signature generation and can generate large galleries of model graphs automatically.

Thus the probe and gallery image signatures are generated automatically when the FBG is rich enough. This technique has the same training-first approach as the PCA systems.

2.5.1.4 Elastic bunch graph recognition

The extracted graph signatures from the probe and gallery-set images are compared against each other and the similarity values are computed. A similarity measure is computed from each probe image graph against all the gallery image graphs. The gallery image graph with the highest similarity value is taken as a match.

When a bunch graph is used for matching, the procedure gets only a little bit more complicated. Besides selecting different salient point locations in the face, the graph similarity is also maximised by the selection of the best-fitting jet in each bunch, independently of the other bunches. Without this independent matching, the computational effort would be combinatorially more expensive [Wiskott et al (1997)].

Wiskott et al (1999) report very good results with this algorithm on frontal views (a 98% recognition rate). The weakness of this algorithm in handling rotation is, however, still indicated by very poor results in pose-varying images (i.e. when a given pose is not represented as an FBG), where a recognition rate as low as 9% is reported in some experiments. The overall success of this approach is highlighted by the development at the University of Southern California (USC) of an elastic graph-matching commercial system called ***eyematic interfaces***[1] . The USC's elastic graph-matching algorithm also competed in the FERET evaluations.

In the FBG approach each matching signature component clearly represents a specific special patch around a salient point. So it is easy to identify a particular area of good or poor matching, e.g. the nose matches well but the eyes do not. By contrast, PCA-signature components encode global information over the whole image, making it impossible to know which particular parts of the face give a good or poor performance.

2.5.2 Statistical-based face representation

Statistical approaches are characterised by the mathematical encoding of face-image features as vectors. These vectors are constructed from pixel intensity values [Turk and Pentland (1991a), Pentland et al (1994), Lin (2000), Zhao et al (2000), Pandya and Szabo (2000)]. There are a number of techniques that fall into this category. A more recent study comparing different statistical-based approaches can be found in Zhao et al (2000), and Navarrete and Ruiz-del-Solar (2002). PCA seems to be the most popular of them all [Zhao et al (2000), Pentland et al (1994), Turk and Pentland (1991a), Campos et al (2000)].

The popularity of the PCA algorithm can be attributed partly to its simplicity and accuracy in constrained environments and perhaps also to its reasonable speed [Turk and Pentland (1991b)]. The good performance of different implementations of PCA systems in FERET evaluations has also contributed to its popularity amongst the statistical approaches [Phillips et al (2000)]. In our system we also use this algorithm.

[1]Available online: http://www.eyematic.com

2.5.2.1 Face representation through principal component analysis

This technique is based on generating a face space and then projecting each face image onto the face space to arrive at a unique face signature for the image. In most instances there has to be preprocessing prior to these two stages. In this section we do not cover the preprocessing.

2.5.2.2 Face-space generation

PCA-based algorithms use a training set of images to establish the best base vectors of a vector space in which the faces will be represented as points. The points will have some property of being maximally distinguishable from one another, so that the images can be best differentiated [Turk and Pentland (1991a)]. The algorithm is based on computing the covariance amongst the face-image class of objects. It should be noted that the PCA algorithm is applicable not only to identifying faces: it can be taught to identify any distinguishable objects within a generic class. A sample best representing the faces to be recognised is chosen for creating the face space. Face-space generation can be thought off as sensitising a PCA algorithm to differentiate between face objects.

In training, the face space (which includes a mean image) is computed. The mean is computed by adding all the images (images of the same size, 128 by 128 pixels in our system) in the training set and then finding an average image. This is a simple matrix-addition problem, given the fact that each images can be thought off as a two-dimensional matrix of pixel intensity values [Haddadnia et al (2002), Pentland et al (1994)].

First we prepare an initial set of face images [M1, M2, ..., Mn] where Mn is the nth image. The mean of the whole face distribution is **M = (M1 + M2 + ... + Mn) /n.** The mean is used for normalising images in the face-enrolment and recognition phases of the algorithm. Each training set image, **Mi',** is normalised by subtracting the mean, **Mi' = Mi - M, i = 1, 2, ... , n.**

The eigenvectors are calculated from the new image set [M1', M2', ... Mn'] as [Y1, Y2, ..., Yn]. These eigenvectors do not correspond directly to any face features such as eyes, nose and mouth [Turk and Pentland (1991b)]. Instead they contain abstract facial information and are referred to as eigenfaces. They are a set of important features that describe the variation in the face-image set. The following diagram (Figure 2e) depicts a face space with a mean image and three eigenfaces.

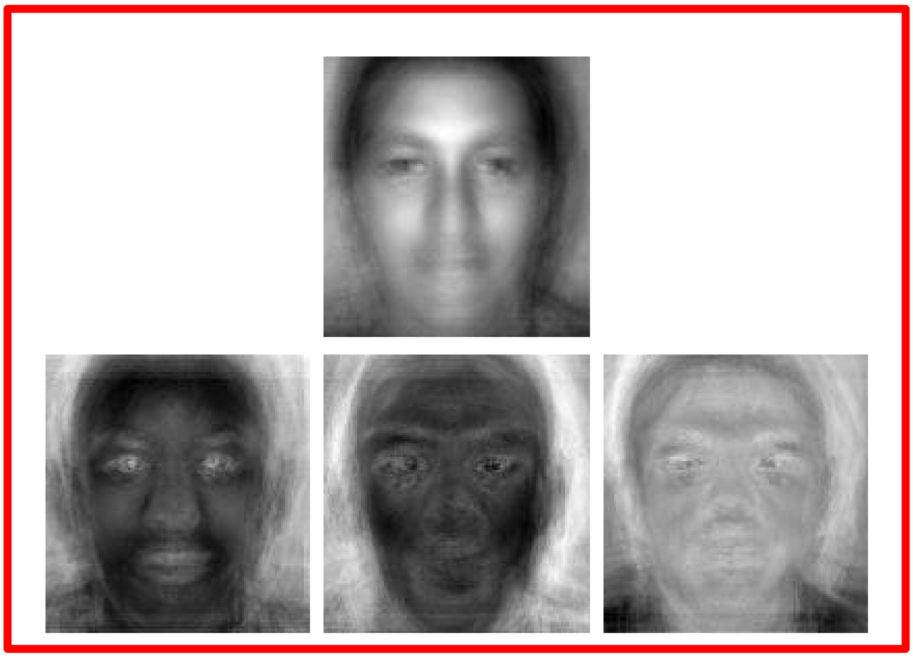


Figure 2e: A face space with a mean image (above) and three eigenfaces (below)

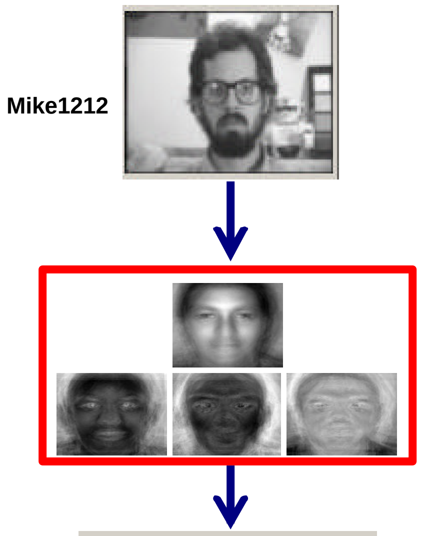
The central idea is that each of the original training faces can now be represented as a linear combination of the average face, plus some multiples of the extracted eigenfaces. For example, if the faces above were labelled M (mean), M1, M2, M3, a particular face in the training set (F) would be exactly representable as F = M + aM1 + bM2 + cM3, for some coefficients (a, b, c). The particular values of (a, b, c) are called the *signature* of F in this face space.

Each eigenvector (for an eigenface) has a scalar eigenvalue associated with it. Eigenvectors with bigger eigenvalues account for more of the variation in the training set than those with smaller eigenvalues [Turk and Pentland (1991a), Zhao et al (2000)]. We always order the eigenfaces in descending order of their eigenvalues. Thus any eigenface is considered a more significant contributor than those that come after it. The sequence of partial sums (M, M+aM1, M+aM1+bM2 …) provides an increasingly accurate approximation to the original training face. The second eigenface characterises the points that are different from the first eigenface, the third characterises the points that are different from the second, and this notion follows up to the last eigenvector [Turk and Pentland (1991a), Pentland et al (1994), Campos et al (2000)]. Training with N training images gives rise to N-1 eigenfaces, i.e. the eigenvalue of the remaining eigenface is 0 and it characterises no points that are different from the rest of the eigenfaces [Jiang (1996)]. It is common to retain only the most significant coefficients as an acceptable approximation.

2.5.2.3 Feature extraction or signature generation

The face-image signatures are computed relative to a specific face space. A preprocessed (normalised) image (128 by 128 pixels in our case) is converted to a 16384-element vector, **M**. A transpose of **M** ($\mathbf{M}^{\mathbf{T}}$) is then computed. $\mathbf{M}^{\mathbf{T}}$ elements are multiplied by each eigenface element in the face space and the products are added to form one value for each eigenface. These numbers together form a face signature within the face space.

An approximation to the original image can be reconstructed from the signature elements by multiplying each eigenface in the face space by a signature component and adding these together, so reversing the process described above [Turk and Pentland (1991b), Campos et al (2000)]. The following diagram illustrates the face-image signature generation process:



```
mike1212.png
  -40    99    14
```

[2]Figure 2f: A feature/signature extraction process

Above, figure 2f shows a projection of Mike’s face image into a face space. The input to the face space is Mike’s face image. The output is Mike’s signature containing 3 entries. As mentioned already, the number of eigenfaces in a face space determines the size of the signature, here 3 eigenfaces. In feature extraction, a face space takes an image as input and generates a corresponding feature signature.

[2] Images used are from MIT VISMOD data base

Taking Mike’s signature above and multiplying it with the eigenfaces, then adding the mean, will produce a very close approximation of Mike’s original image.

2.6 Signature size

Another factor that is crucial in statistical-based face-signature encoding schemes is the quantity of information that must be encoded in a signature, i.e. the number of elements in the signature. This factor seems to be overlooked in the literature, and some publications report recognition results without mentioning the number of elements in the face signature (e.g. the number of eigenfaces in a PCA result). The other interesting factor is the relation between the signature size and the gallery set size: is there a good rule of thumb, such as that the size of a signature must be 60% of the number of images in the gallery?

In PCA, each eigenface accounts for some percentage of the variance in the training set. The number of these principle components, or eigenfaces, is always less than or equal to N-1, where N is the number of images in the gallery set, since N-1 base components can account for all the variances in a training set of size N, without any residual error. The objective, however, is to use only a small number of the most important components: sufficient to account for "enough" variance, but few enough to be computationally efficient. Once the eigenfaces have been determined, they are ordered by significance. Each eigenface or principal component has an associated eigenvalue that effectively represents the proportion of variance it accounts for in the analysis. In classical statistical approaches, one normally sets a threshold, say of 90%, and selects a sufficient number of eigenfaces so that the proportion of variance that is accounted for exceeds the threshold.

Campos et al (2000) confirm the effect of the *number* of eigenfaces used in face-space generation in their study where a variant number of eigenvectors were used in assessing performance of the PCA–based algorithm. The recognition rate improved as more eigenfaces were used. A recognition rate of 25.00% was achieved when using three eigenvectors and this improved to 50.00% when using five eigenvectors; a 56.25% recognition rate was achieved with ten eigenvectors. Interestingly the recognition rates converged when using 13-48 eigenvectors at 62.50% [Campos et al (2000)]. These findings are interesting but they are not a statistical trend since the research carried out by Campos et al (2000) was not primarily aimed at investigating the effect of signature size.

Haddadnia et al (2002) use 30 eigenvectors for the PCA component of their N-feature neural-network classifier, which further highlights the significance of using larger numbers of eigenvectors. Pentland's photo book uses 40 eigenvectors and claims a 95% recognition rate on frontal face images [Lin (2000)]. Turk and Pentland (1991a) used 7 eigenvectors from 16 images from the training set and achieved 96% correctness in classification averaged over lighting variation, 85% correctness averaged over orientation variation, and 64% correctness averaged over size variation. The above recognition rates are all comparable but significantly they are all based on diverse signature and population sizes. Campos et al (2000) emphasise that PCA-generated matching results can be effectively compared only if the signature and gallery sizes are known. We report on our observations on this in later chapters.

The quantity of information necessary to effectively discriminate between faces is an issue not only in statistical systems. Yoshino et al (2001) in their template-based system use 16 anthropometrical points (i.e. points selected on the basis of human anatomy) in their study. They report good face-recognition results i.e. a low rate of 4.2% false matches. Their system is different in that its aim is to identify images of suspected criminals, and it is therefore intended to be used with carefully posed images (mugshots) of possible matches. Its matching rates are therefore not directly comparable with the systems discussed in the above section. It differs further in that the recognition takes place in the mind of the user who, from the list returned by the system, must decide which image best matches the probe.

2.7 Intra-face vs. inter-face differences

Intra-face difference is the difference computed between different images of the same individual. In contrast, inter-face difference is the difference computed between images of different individuals which should logically be greater than any intra-face difference.

An ideal system should minimise intra-face difference and maximise inter-face difference. A question not yet solved is the effect of signature size on these differences. The distinction between the two can act as a good guideline in helping to decide on an ideal signature size.

2.8 Face-recognition reasoning

Face-recognition reasoning is a method of comparison between probe face signatures and gallery face signatures. This might involve a comparison of two or more vectors in an examination of statistical-based algorithms; or a comparison of different face-image graphs when considering structural face-recognition systems. As an example of the first, we might ask how close the vector [-40, 99, 14] is to another signature [-35, 90, 18].

Standard distance metrics such as City-Block (L1 norm), Euclidean (L2 norm), the Cosine Angle difference between two vectors, or some weighted distance measure such as Mahalanobis[3] (for eigenvector spaces), can be used for matching signature vectors.

Different variants of neural networks have also been used for face reasoning. The ability of neural networks to learn from their past experience and their tolerance of noisy and incomplete data make them the best contender in this area [Pandya and Szabo (1999), Howell (1999)]. Kohonen associative networks, Multi-Layer Perceptrons and Associative Networks, and Hierarchical Neural Networks are all different variants of neural networks used for face reasoning [Howell (1999)].

2.9 Hybrid PCA-powered face-recognition systems

A number of PCA-based face-recognition systems have been implemented. Most of these systems use a PCA algorithm to represent face-images in a low-dimensional face space. PCA face-representation technique has been integrated into other statistical and neural-network-based techniques in face recognition. These hybrid systems capitalise on the success of the PCA approach in reducing face-space dimensions.

[3] The Mahalanobis matching metric uses eigenvalues to attach greater importance to differences in the more significant components.

2.9.1 Principal Component Analysis and the Linear Discriminant Analysis function

A new set of techniques needs to be applied in situations where we are able to enrol more than one image for each individual so that each individual is represented by a group of images or signatures.

Discriminant function analysis is used to determine which variables discriminate between two or more naturally occurring groups. The aim of the Linear Discriminant Analysis (LDA) technique in face recognition is to minimise variability among face signatures for the same person and maximise variability among face signatures for different people. This technique groups together images of the same person and distances these images from the rest of the training images. When a test image is projected onto the new LDA face space (sometimes called "Fisher space", from Fisher one of the pioneers in LDA), it is classified as belonging to one of the face image clusters.

LDA can be mapped to the concept of "*a priori* classification probabilities". An additional factor needs to be considered when classifying images based on clustering (i.e. having multiple images of the same individual in the training set): sometimes, it is known in advance that there are more images (of the same individual) in one group than in any other, which leads to the greater a *priori* probability that an image will belong to that group [Zhao et al (1998)]. This is the case with face recognition after LDA processing; chances are higher that the image belongs to its cluster than to any of the other clusters. The Bayesian technique of modelling face similarity utilises this concept [Moghaddam et al (1999)]. The algorithm developed by Moghaddam et al (1999) was a best performer in the 1996 FERET tests.

2.9.1.1 Applying the LDA function to PCA projections

Applying the Linear Discriminant Analysis (LDA) function to principal components before computing matching scores is one of the common techniques used. Zhao et al (1998) project face images onto a face space to get PCA coefficients. Then they feed the resulting PCA coefficients into an LDA algorithm to get a linear classifier. The object in combining PCA and LDA is to improve the generalisation capability of LDA when only a few samples per class are available. Zhao et al (1998) observed that LDA on it own performs poorly at handling both images that are not part of the training set and images with varying backgrounds. Improved recognition rates of up to 97% are reported against the Yale and AT&T face databases. The problem with the use of the LDA algorithm is that you need image clusters (i.e. more than one image per person) for training, making this technique not suitable for some applications where only a few images per individual are available.

2.9.1.2 Fused Principal Component Analysis and Linear Discriminant Analysis projections

LDA can be performed directly on an image but computing it on low-dimensional PCA signatures saves computing costs since, instead of having a 128 by 128 pixel-image, LDA can now be computed on a 10-co-efficient PCA-generated signature. The problem with the latter approach is that some information critical for discrimination might be lost when computing PCA projections. Marcialis and Roli (2002) improve on the above (section 2.9.1.1) approach from Zhao et al (1998) by performing both LDA and PCA on the same set of images and then combining the signatures for use at the recognition phase. A special algorithm fuses the two signatures. K-nearest-neighbour techniques (e.g. Euclidean, City-Block) are then used to compute matching scores. This approach attempts to retain benefits from both PCA and LDA techniques. With this approach, performance scores of between 95.90% and 97.25% have been reported, using different classifiers on an AT&T faces database. These recognition scores are comparable with those of Zhao et al (1998) discussed above. The problem with fusing PCA and LDA signatures is that there are more computing costs involved.

2.9.1.3 Unified LDA/PCA algorithm for face recognition

Yang et al (2000) suggest a unified LDA/PCA algorithm. This algorithm avoids having first to compute PCA coefficients and then perform LDA on them to maximise discrimination. Its goal is to produce an algorithm that maximises the LDA criterion directly, without a separate PCA step, and at the same time eliminates the possibility of losing discriminative information in the absence of a PCA step. Yang et al (2000) report 95% recognition rates against the Olivetti Research Lab (ORL) database using this technique. Its main advantage is computational efficiency and it also eliminates the possibility of losing discriminative information in the absence of a separate PCA step. Generally speaking, the recognition results are comparable with those of Zhao et al (1998) and with Marcialis and Roli's (2002) technique, even though separate face databases were used.

2.9.1.4 PCA and neural networks

The PCA feature-extraction algorithm has been used with different variants of neural networks for face processing [Jiang (1996), Zhang (2000), Er et al (2002), Haddadnia et al (2002), Sun et al (2002)]. Radial Basis Function (RBF) neural networks have recently attracted intensive interest in image-recognition research. According to Er et al (2002) and Haddadnia et al (2002), interest has been drawn by the following strengths of the RBF neural networks;

- They are universal approximators.
- They posses the best approximation property.
- Their learning speed is fast because of locally tuned neurons.
- They have more compact topology than other neural networks.

For more on RBF neural networks see Er et al (2002) and Haddadnia et al (2002) and for recent reviews of other variants of neural networks in face see Zhao et al (2000), Egmont-Petersen et al (2002).

A comparable and sometimes better face-recognition performance of RBF neural networks using PCA for feature extraction is reported in the literature. Abdi et al (1995), using a simple perceptron and an RBF network, report a best performance of 90% when PCA feature extraction is used prior to RBF classification. A more recent study by Er et al (2002), in which PCA followed by LDA is used for feature extraction, reports up to 99% best classifications against the Olivetti Research Laboratory (ORL) face database. Haddadnia et al (2002) using their RBF-powered Hybrid N-Feature Neural Network (HNFNN) report 99.7% recognition scores against the ORL faces database.

2.9.2 Combining face identification techniques

One might wonder whether the face images misclassified by one method are also misclassified by the other methods and if combining the methods could improve the matching performance scores. One of the problems in combining face-identification techniques is the independence of both the methods and the data used. A good illustration of this is the work of Yambor et al (2000), where combined measures such as Euclidean, City-block Mahalanobis and cosine angle were used for face recognition. None of the different combinations managed to give a significant improvement over what the original methods had achieved. The problem with such results is that all the scores were performed on PCA-generated signatures and, as a result, the techniques were not statistically independent but positively correlated. Face-recognition techniques (known as *experts*) must be statistically independent for positive results [Czyz et al (2002)].

Tang et al (2003) and Czyz et al (2002) implement statistically independent techniques in an attempt to improve face-recognition scores. Tang et al (2003) in their “Face Recognition Committee Machine” employ five well-known techniques, namely PCA (eigenface), LDA (Fisherface), EGM (Elastic Graph Matching), SVM (Support Vector Machines) and NN (Neural Networks). In their system DFRCM (Dynamic Face-Recognition Committee Machine) they achieved significant improvements of between 97% and 81% against six popular face databases (FERET, ORL, Yale, etc.). In this approach each expert’s results are weighed using a special gating NN network for optimum performance.

Czyz et al (2002) report improved results when combining different LDA variants with different probabilistic matching algorithms based on PCA. Improved matching scores with combined experts are also reported by Achermann and Bunke (1996). Their study also reports on the effects of combining outputs from different techniques using approaches such as voting, ranking and scoring. These are discussed in depth in Achermann and Bunke (1996).

Czyz et al (2002) assert that there are a few factors with a part to play in the potential improvement of the performance of expert combinations. Here preprocessing techniques rate above matching schemes. The reasons behind their assertion explain why the combined expert scheme developed by Yambor et al (2000) yielded such poor results: their matching schemes were all based on signatures extracted by an eigenface technique.

2.10 Conclusion

This chapter looked at the basic building blocks of face-recognition algorithms. Factors that are crucial in deciding on a face-recognition system were discussed. These comprised: the information in the signature, i.e. global versus local; the type, i.e. template-based or feature-based; and, lastly, the effect of the quantity of information encoded in the signature, i.e. the size or number of elements in the signature.

In addition to examining the basic techniques for comparing a single probe signature to a single gallery signature, we also reviewed some work that uses Linear Discriminant Analysis to cluster groups of gallery signatures (each group represents one individual) and considered the combined experts methods. We have not pursued these approaches in the rest of this work.

The above factors serve as a guideline towards the production of better face-recognition systems.

Chapter 3

XML Web Services as a Glue for Distributed Face-Recognition Framework Modules

3.1 Introduction

The previous chapter introduced the face-recognition process as a modular task made up of a number of components. Now we turn to the distribution of our face-recognition experimentation framework. The reason for distribution is to enforce decoupling; this emphasises the distributed nature of the face-recognition process, makes it easy for others to use our work remotely and separates the design aspect into well modularised components. The application is distributed over a number of computers (the physical distribution) as independent modules (the logical distribution) which work together towards facilitating our experiments.

This chapter gives an overview of web services and the role they can play in distributed face-recognition systems. Microsoft .NET was chosen as the implementation environment. The goal was to assess the suitability of the Microsoft .NET framework in developing a scientific application. This chapter gives a high-level overview of distributed technologies and web services and does not aim to discuss these in depth. The technologies are discussed here to draw attention to our choice of implementation platform and its relevancy to the face-recognition process. Detailed information on web services can be found at the Web Services[4] home page, which is a community portal for web services.

3.2 Distributed systems technologies

A number of solutions have been developed to facilitate the deployment of distributed systems. These solutions are usually grouped together under the topic of component-based programming. A

[4] Available online at: http://www.webservices.org

few of these are: the Microsoft Distributed Component Model (**DCOM**), Common Object Request Broker Architecture (**CORBA)** and Java Remote Method Invocation (**Java/RMI**).

3.2.1 CORBA

CORBA relies on a protocol called the Internet Inter-ORB Protocol (IIOP) for locating remote objects. Everything in the CORBA architecture depends on an Object Request Broker (ORB)**.** The ORB acts as a central object bus over which each CORBA object interacts transparently with other CORBA objects located either locally or remotely. Each CORBA server object has an interface and exposes a set of methods. To request a service, a CORBA client acquires an object reference to a CORBA server object. The client can now make method calls on the object reference as if the CORBA server object resided in the client's address space. The ORB is responsible for finding a CORBA object's implementation, preparing it to receive requests, communicating requests to it and carrying the reply back to the client. A CORBA object interacts with the ORB either through the ORB interface or through an Object Adapter **-** either a Basic Object Adapter (BOA) or a Portable Object Adapter (POA)**.**

Since CORBA is just a specification, it can be used on operating system platforms as diverse as mainframes, UNIX boxes, Windows machines and handheld devices, as long as there is an ORB implementation for that platform [Raj (1998)]. The limitation is that there must be an ORB implementation for the platform and the ORB implementations that currently exist are proprietary. This is one of the challenges that led to a poor acceptance of CORBA.

3.2.2 Java/RMI

Java/RMI relies on a protocol called the Java Remote Method Protocol (JRMP)**.** Java relies heavily on Java Object Serialisation, which allows objects to be marshalled (or transmitted) as a stream. Since Java Object Serialisation is specific to Java, both the Java/RMI server object and the client object have to be written in Java. Each Java/RMI Server object defines an interface which can be used to access the server object outside of the current Java Virtual Machine (JVM) and on another machine's JVM. The interface exposes a set of methods which indicate the services offered by the server object. For a client to locate a server object for the first time, RMI depends on a naming mechanism called an RMIRegistry that runs on the Server machine and holds information about available Server Objects. A Java/RMI client acquires an object reference to a Java/RMI server object by doing a lookup for a Server Object reference and invokes methods on the Server Object as if the Java/RMI server object resided in the client's address space. Java/RMI server objects are named using URIs and, for a client to acquire a server object reference, it must specify the URI of the server object, very much as it would the URI for an HTML page. Since Java/RMI relies on Java, it can be used on diverse operating system platforms, again from mainframes to UNIX boxes to Windows machines to handheld devices, as long as there is a Java Virtual Machine (JVM) implementation for that platform [Raj (1998)].

3.2.3 DCOM

DCOM supports remote objects by running over a protocol called the Object Remote Procedure Call (ORPC) [Raj (1998)]. The ORPC interacts with COM's run-time services. A DCOM server is a body of code that is capable of serving up objects of a particular type at runtime. Each DCOM server object can support multiple interfaces, each of these representing a different behaviour of the object. A DCOM client makes a call to the exposed methods of a DCOM server by acquiring a pointer to one of the server object's interfaces. The client object then starts calling the server object's exposed methods through the acquired interface pointer as if the server object resided in the client's address space. As specified by COM, a server object's memory layout conforms to the C++ vtable (the *vtable is a structure containing code memory addresses*) layout. Since the COM specification is at the binary level, DCOM server components may be written in diverse programming languages such as C++, Java, Object Pascal (Delphi), Visual Basic and even COBOL. As long as a platform supports COM services, DCOM can be used on that platform. DCOM is heavily used on the Windows platform, although Microsoft now appears to be pushing .NET as a replacement for DCOM in many situations.

3.3 Open protocols as an alternative

There is a growing move to replace current tightly-coupled distribution models with a more flexible architecture, yielding systems that are more amenable to change. The web services architecture is an open architecture that is acquiescent to changing business-applications needs.

3.3.1 Internet-level distributed systems

Short (2002) and Curbera et al (2001) maintain that technologies like CORBA, Java/RMI and DCOM are best suited for the homogenous Intranet level. The primary reason for this is that these technologies by default use proprietary protocols and these protocols are inherently connection-oriented, making it hard for them to be of practical global Internet use. Curbera et al (2001) maintain that most of the existing protocols expose artefacts that limit their ability to interoperate with alternate environments, and thus are unsuitable for use in a truly heterogeneous system. A connection-oriented (and network platform dependent) protocol is usually not a best solution over the global Internet, due to system administrators' security concerns and firewalls, the inability of such systems to handle network disruptions and the management problems of complex system running over different networks.

3.3.2 Stateless request/response remote service calls

The central common features in DCOM, RMI and CORBA are that they provide ways to:

- locate remote systems that can offer some kind of service, or create objects of a certain type,
- instantiate objects on the remote system,
- somehow marshal calls and data backwards and forwards to permit the client to use the remote objects that have been created,
- terminate or finalise the remote objects.

Each technology resolves each of these issues in its own proprietary way: for example, the DCOM marshalling rules create binary data packets that are DCOM-specific. All of these technologies are inherently connection-based with "stateful" lifetimes and the problems that go with maintaining long-term connections.

For distributed web services to be useful, they must provide similar functionality. A critical difference is their use of stateless request/response services, which has its roots in HTTP, and has become the de-facto paradigm for communication with busy web sites. The World Wide Web (WWW) has evolved using primarily stateless systems and protocols and, when the exception arises and some form of state or session is required, it is provided by secondary artefacts like cookies. This has proven to be a workable and much more highly scalable paradigm than “a connected, session-based, I-am-responsible-for-your-object-lifetime” approach to distributed computing. So web services are now using the World Wide Web for software interaction, in contrast to our previous use of the World Wide Web as primarily for human/browser interaction.

3.4 Web Services

3.4.1 Definition

The Web Services Architecture Working Group [Haas and Brown (2002)] define a web service as a software system designed to support interoperable machine-to-machine interaction over a network. A web service has an interface described in a machine-processable format (Web Service Definition Language (WSDL)). Other systems interact with the web service in a manner prescribed by the WSDL description, using SOAP-messages, typically conveyed using HTTP with XML serialisation in conjunction with other web-related standards. From this definition it is becomes clear that web services have been designed with the goal of supporting application interoperability over the network using standard protocols.

Web services build on open standards. XML is used for all data transport: loosely, it replaces the role of HTML in the “human-operated” web. XML is self-describing, more regular than HTML and simple to process automatically. SOAP (formerly an acronym for the simple object access protocol) is used for a one-shot message-response sequence. It encourages connectionless designs with very loosely coupled components interacting with each other over the web. SOAP request/response cycles replace the HTTP request/response cycle, and permit a far richer (and extensible) set of requests and (logical) port connections. HTTP has only a small number of request-type options, e.g. GET, POST, HEAD, and HTTP is generally is associated with port 80. Although SOAP requests can be physically transported in a number of ways (e.g. email, ftp), the currently most popular method is to use the HTTP protocol to carry the SOAP requests. Because most firewalls cannot afford to block HTTP, piggy-backing SOAP onto HTTP solves many of the difficulties that proprietary transport technologies run into. (Tunnelling SOAP through the firewall like this is not something that is particularly popular with system administrators!)

3.4.2 Web services: the process

Flurry (2001) maintains that the fundamental principle underlying web services is the service-oriented architecture (SOA). A SOA focuses on how service components are described and organised to support dynamic, automated discovery and use. The following essential activities need to happen in any service-oriented environment:

- A web service needs to be created, and its interfaces and invocation methods must be defined. This is achieved through the Web Service Definition Language (WSDL). WSDL is an XML extension, and a WSDL document usually describes a web service. Among other things, it specifies the location of the service and the operations (or methods) the service exposes.
- A web service needs to be published to one or more Intranet or Internet repositories for potential users to locate. Web services have a mechanism to advertise themselves and their locations for discovery by other web services. The Universal Description, Discovery and Integration (UDDI) standard is a mechanism whereby web services may be published in a repository together with the locations where they may be found.
- A web service needs to be located in order to be invoked by potential users. This is achieved via HTTP after the service has been advertised by UDDI. The service is then located on the Internet through its host.
- A web service needs to be invoked (remotely) to be of any benefit. Invoking a service can be the result of a simple click by a user to check stock markets or make purchases online, right up to more complex application-to-application invocations to process business-to-business data.
- A web service needs to be revoked when it is no longer available or needed. This is also taken care of by UDDI. If the service is no longer available it is deregistered through UDDI and now will no longer be advertised.

The following diagram illustrates some of the above steps:

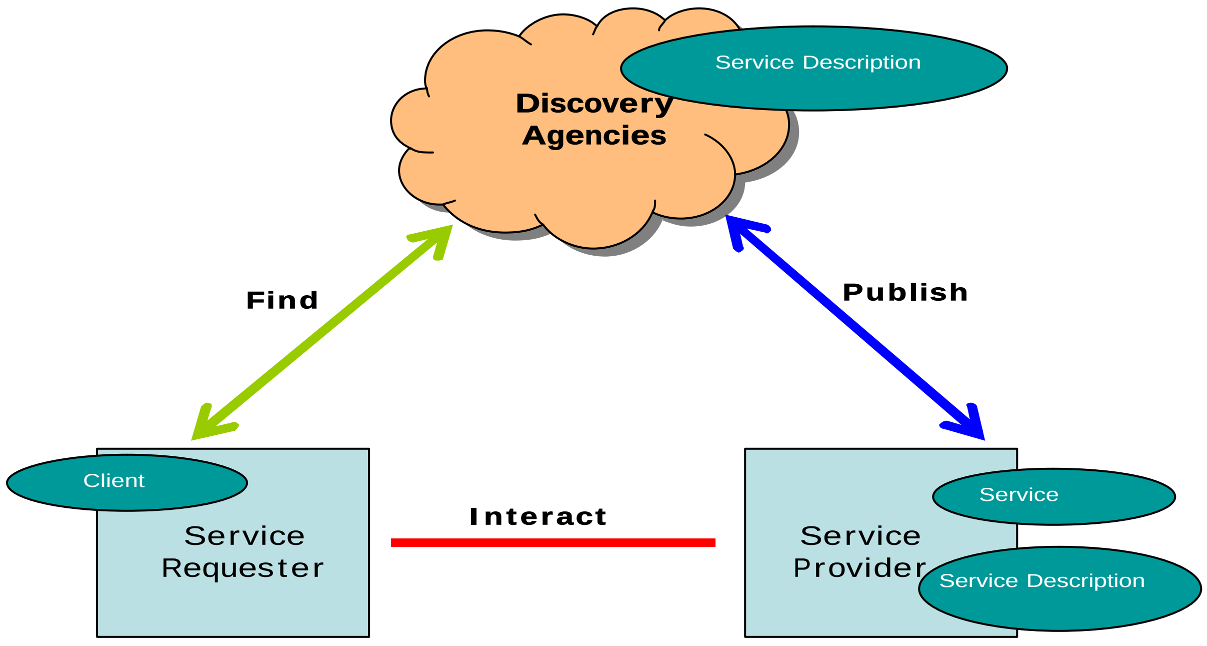


Figure 3a: An illustration of a Service Oriented Architecture (SOA)

The figure above illustrates the relationships between requesters, providers, services, descriptions, and discovery services in a case where agents take on both requester and provider roles. For example, XML messages compliant with the SOAP specification are exchanged between the requester and provider; the provider publishes a WSDL file that contains a description of the message and endpoint information to allow the requester to generate the SOAP message and send it to the correct destination.

3.4.3 Microsoft .NET XML web services

Short (2002) asserts that Microsoft's .NET XML web services were built in response to the following underlying requirements:

- **Interoperability** – the remote service must be able to be consumed by clients on other platforms, meaning that a web service must be platform independent.

- **Internet friendliness** – the solution should work support clients that access the remote service from the Internet. The goal here is to build solution-based standards or protocols that have already a proven success in a multi-platform environment.

- **Strongly typed interfaces –** there must be no ambiguity about the type of data exchanged between the web service and the client application.

- **Ability to leverage existing Internet standards –** a solution must not try to reinvent the solutions to problems that have already been solved. This approach promotes interoperability and avoids the introduction of proprietary protocols.

- **Support for any language –** the solution should not be tightly coupled to a particular programming language. Java RMI, for example, is tightly coupled to the Java programming language. A client should be able to implement a new web service or use an existing web service regardless of the programming language in which the client is written.

- **Support for any distributed component infrastructure –** again the solution must be compatible with existing distributed component-based systems. It should not be a burden to integrate a web service-based solution with an existing solution, and to expose the result as part of a new web service.

The above characterisation from Short (2002) pinpoints the goals shared by stakeholders involved in the web services initiative: interoperability, and the use of well-established Internet protocols. Microsoft claims this to be a revolution in applications development and integration, a software-convergence where the target application environment will no longer be a key issue in systems design.

3.5 Benefits offered by web services in the field of face recognition

The above sections have looked broadly at web services as a platform capable of easily handling distributed applications, particularly those that can be designed to be stateless. Web services are easy and convenient to use, build on existing internet technologies, and allow interoperability amongst heterogeneous applications. A face-recognition process can capitalise on a number of these aspects.

3.5.1 Easy handling of distributed modules

As discussed in the previous chapter, the face-recognition process is made up of a number of components or modules that can be easily decoupled to achieve our goal of a flexible framework for face-recognition experimentation. It must be easily pluggable, have swappable modules and allow for the easy construction of prototype systems. The modules can be distributed over a number of machines as web services. The web services platform adds value in handling communication between the different modules. In short, our goal of a framework that facilitates easy experimentation is easily achieved with the web services architecture.

3.5.2 Stateless connections

One of the problems with other distributed technologies is that they tend to be stateful and connection oriented – i.e. remote objects are created and state information is critical for effective performance. A consequence of this is that, when a remote host machine fails for any reason, remote objects lose their current state. When the host machine comes up again, the remote methods will no longer be active and the local system has to be restarted to reconfigure state information. Web services on the other hand do not maintain state information. (If persistency is needed, it can be achieved via mechanisms such as cookies, but it is not the preferred or inherent mode of operation for a web service.) Web services encourage one to think in terms of stateless transactions and loose coupling between modules. This is particularly to our advantage in creating an experimental framework with exchangeable components. Loose and stateless coupling keeps the interfaces simpler, and makes it easier to swap new modules in and out. This is highly appropriate for a framework intending to handle the distributed face-recognition problem.

3.5.3 Enhanced systems integration

Popular face-recognition systems run on UNIX (or Linux) boxes, and a web services approach has the potential to bridge the communication barriers between UNIX and Microsoft Windows-based applications. Because the web services architecture is based on standard, platform independent, Internet protocols such as SOAP, our system, running on Microsoft Windows, will be able to interact with other face-recognition systems running as web services anywhere else. This platform independence minimises time spent on porting code. All the difficulty we had when porting the

Principal Component Analysis (PCA) face-recognition code from MIT Media Labs (more on this code on chapter 4) could have been avoided if it had originally been exposed as a few web services.

3.5.4 Extension of legacy applications

Face-recognition research spans almost two decades. Integrating applications (or algorithms) written mostly in C has been a huge problem because it has usually involved rewriting the entire code in the new development environment and then adding new enhancements. Such an approach is time consuming and sometimes the porting-process results compromise the original code efficiency through programming language differences.

Web services promise to offer a new paradigm where porting is no longer necessary. Chaudhary et al (2002) claim that it is relatively straightforward to take a legacy application, generate a SOAP wrapper and cast the application as a web service. This claim still remains to be tested. (In our case we chose to port the C code. This gave us a better understanding of the low-level operations of the algorithm, as opposed to generating a SOAP wrapper with poor understanding of the algorithm). Allowing legacy face-recognition algorithms to be exposed as web services permits a new interoperability between these applications. This in turn can enhance research into face-recognition algorithms since the barrier to using legacy applications can be eliminated.

3.5.5 Support for new standards in biometrics

The need for a simple biometric framework where any biometric method can be plugged in with minimum effort is highlighted by the development of Common Biometric Exchange File Format (CBEFF[5]) and XML Common Biometric File Format (XBFF[6]) standards. Web services have a potential role that they can play in such initiatives. They can act as an interface to different biometric systems.

[5] http://www.ibia.org/CBEFFtechnicalinfo.htm
[6] http://www.oasis-open.org/committees/xcbf/#documents

3.5.5.1 CBEFF – the Common Biometric Exchange File Format

CBEFF describes the set of data elements necessary to support biometric technologies in a common way. CBEFF facilitates biometric data interchange between different system components or between systems, promotes interoperability of biometric-based application programs and systems, provides forward compatibility for technology improvements, and simplifies the software and hardware integration process. Web services can add value to CBEFF since interoperability is also one of the primary goals in CBEFF and web services run on standard protocols such as HTTP, SOAP and XML.

3.5.5.2 XBFF-XML Common Biometric File Format

XBFF is the XML implementation of the concept developed in the CBEFF standard. The XML implementation capitalises on benefits, such as standardised data presentation, offered by XML. Web services can also facilitate the integration and interaction of different biometric applications implemented with XBFF. Until these two standards have been finalised there is no obvious way of achieving a common language for biometric systems.

3.5.6 Just-In-Time integration

Dynamic service discovery, invocation (publish, find and bind) and message-oriented collaboration yield applications with looser coupling, enabling just-in-time integration of new applications and services. This capability might be very useful in future if we extended our system to be self-configuring; at present we do not exploit this capability. Glass (2000) asserts that just-in-time integration can yield systems that are self-configuring, adaptive and robust with fewer single points of failures. This claim is still to be tested in web services.

3.5.7 Ubiquity

This is another field where web services can add value. Web-services-friendly hand-held devices are already on the market and it is no longer far-fetched to embed face recognisers into these. Face-recognition systems can now be easily adapted for hand-held computer devices like PDA’s using web services. This is possible since web services communicate using HTTP and XML and any device that supports these technologies may be both host and access web services as a client. The embedding of face-recognition technology in smaller devices implies the use of thin clients for front-end recognition and powerful back-end servers for the face-recognition process. Thin clients could then be embedded in phones, cars, etc. Web services still need to be tested on this application area.

3.6 Conclusion

This chapter looked at the current state of distributed systems technologies. Web services architecture is discussed as an alternative to RMI, DCOM and CORBA. The value of web services as an open distributed architecture is highlighted, together with software convergence. Lastly the role of web services in a distributed face-recognition system is discussed together with its possible benefits. In this project we put some of the above benefits from web services to a practical test. Microsoft .NET XML web services running on Visual Studio .NET are used to integrate the building blocks of our face-recognition system. The face-recognition system is both physically and logically distributed over more than one machine. C# is used as an underlying programming language.

Chapter 4

DRUBIS: Distributed Framework for Biometric Identification Systems – *Face-Recognition Case Study*

4.1 Introduction

There are a number of possible benefits to be gained from using a distributed biometric recognition model. These include: flexible and decoupled system modules; code re-use; system scalability; interoperability; centralised maintenance of infrastructure and software; perhaps distributed departmental-level maintenance of enrolments in a university setting. Nevertheless, little attention has been paid to the investigation of such a model. One of the reasons for this is that, until the core recognition problem has been adequately solved, little attention can be paid to issues of implementation, deployment and distribution. More attention is being given to developing new and better biometric recognition algorithms that can best handle real-world automatic biometric identification challenges. Only minimal effort has gone into formalising the biometric recognition experimentation framework. The author knows only of the NIST HumanID Evaluation Framework of Micheals et al (2003). In our case, the most compelling reason for distributing the framework is that a distributed system makes it easier to swap modules in or out, and is therefore easier to experiment with. In this chapter, a distributed framework for face recognition is presented, and applications developed to test this framework are discussed.

A distributed face-recognition framework for large-scale systems is tackled by Rzeszutek (2002) as a solution to the computer-intensive nature of the required image processing. His study is informed by an emphasis on having a number of machines (a pool) form one component of the face-recognition task, and it is also aimed at surveillance applications. The offline storage of captured video frames for later analysis or playback demands a huge budget that is beyond the scope of our study. Instead, DRUBIS is a flexible experimentation framework (laboratory-like system and sidestep large-scale environment issues), distributed over a number of modules that are easily pluggable and swappable, allowing for the easy construction of prototype systems. Web services are the logical means of distributing DRUBIS components and a number of prototype applications have been implemented from this framework.

The approach of Rzeszutek (2002) to a distributed framework is shared by Yang et al (2002) in their web-based face-recognition model using mobile agent technology. They advocate using mobile agents to reduce the processing time and the latency (i.e. the time of the client/server round trip request/response pairs) and to increase flexibility (i.e. with platform independence).

The two systems discussed above share some characteristics with our system: their distributed nature, flexibility and their modular design, where system components can be replaced and compared with ease. Apart from these, our system is unique, has different system requirements and covers a small-scale environment with limited scope and resources. This chapter covers the goals of our system, our design, implementation and some applications that we implemented as part of this project.

4.2 System requirements for Distributed Rhodes University Biometric Identification System (DRUBIS)

DRUBIS has been designed with a number of broad requirements aimed at better understanding the PCA face-recognition methods, the role that can be played by Microsoft .NET XML web services in the face-recognition process, rendering support to third party applications, and introducing a generic web-based platform for PCA-driven face recognition. The system requirements are:

- A better understanding of the low-level operation of the PCA face-recognition process.
- The development of a toolbox that encourages PCA-based face-recognition.
- The investigation of the use of Microsoft .NET XML web services outside of its primary commercial target domain, and an assessment of its usefulness in addressing a much more scientific problem such as image processing.
- The implementation of a centralised face-recognition service that can easily be plugged into other possible tasks. The distribution via a common protocol like HTTP enables this.
- The enablement of access to DRUBIS by third-party applications.
- An investigation of the critical factors in implementing PCA-based proprietary face-recognition applications.

As mentioned in earlier chapters, the MIT media laboratory code of Turk and Pentland [1991a] did not modularise the face-recognition process in a way that might be used in a flexible educational laboratory framework. Our first step was to port the existing code to C# on a Microsoft .NET platform. We opted for rewriting the code rather than using C or C++ on .NET framework, again as a learning process, and to make it more accessible to peers in our group. In the new system, a more open and explicitly modular approach was adopted.

The need for a centralised, flexible face-recognition system played a major role in the structuring of our framework, as did the requirement that it be also extensible so that, as a system, it could be used in a number of different contexts and for different purposes. These purposes include: a face-recognition demonstration as part of a showcase facility for raising Science, Education and Technology (SET) awareness among the general public; a general research platform that allows experimentation with new modules; and prototype recognition systems e.g. to help the personnel in a reception area to identify students, or to help a university department control and monitor the use of resources such as a photocopier.

A distributed architecture based on Microsoft .NET XML web services has been adopted. This allows the exposure of most of the face-recognition processes as small web functions. These functions are accessed via the Internet through SOAP over HTTP. The data is exchanged as XML data. The idea is that, by structuring the system as a collection of loosely coupled components, with standardised XML data interchange and web-based URL connections, we will have created a flexible plug-and-play environment. Researchers working on some sub-problem such as locating and uprighting a face, or removing background noise, can simply "rewire" the framework to use their own components rather than the default. The redirection of the URLs for the new services can be done on the fly, without rebuilding or disruption to other components or services.

4.3 A generic face-recognition framework

A face-recognition process can be broken down into a number of sub processes. There is no standard number of sub processes; descriptions in the literature focus rather on system requirements and resources as in Rzeszutek (2002) and Yang et al (2002). The issue of a generic framework is partially covered by Lin (2000) and Howell (1999).

Lin (2000) suggests a broad framework that covers only two key components of the face-recognition process, namely:

- a face-image detector, to extract the face(s) from a current image,
- a pattern recogniser to match the extracted face(s) against stored images.

According to Lin (2000), face recognition can be thought of as a two-step process, resolving into face *detection* and face *recognition*. A face-image detector locates the faces within a picture, against simple or complex backgrounds; and a face recogniser determines who the person is, if the image has been stored in the system's database of faces. Face detection and face recognition both use a *feature extractor* and a *pattern recogniser*. The feature extractor transforms pixels of the facial image into a useful vector representation called a *feature signature* or, more concisely, a *signature,* for storage and comparisons. The pattern recogniser (or matcher) searches the database to find the best match to the incoming signature [Lin (2000)]. This conceptual framework is valid but broad. It includes under its feature extractor the more practical image preprocessing stages that are necessary for a better image recognition.

Howell (1999) covers a more detailed generic face-recognition framework. Here the recognition process is broken down into three broad categories, each dealing with a critical component of the face-recognition process. This approach seems to be less tightly coupled to the underlying algorithm used for face recognition and may be applied to generic object recognition.

The three categories are:

- *Acquisition* – what are the important factors in the way face information is collected?
- *Representation* – what is it in a face that can allow a face-recognition system to remember it when it sees it again and distinguish from others?
- *Reasoning* – how can a face-recognition system compare faces most effectively?

Face acquisition looks at how the original data is acquired before the issue of representation is raised. The size and number of variations in face images is decided at this stage. The complete process of normalisation takes place here. Lin (2000) left out this stage in his framework. Intensive computation is involved in the face acquisition stage, and sometimes the practical outcome of image recognition depends on the success of the normalisation process carried out at this stage. This view is also shared by Phillips et al (2000) who assert that all face-recognition algorithms known to them consist of face detection, and normalisation before face identification.

Face representation is a process of isolating and extracting the salient features from the input face image to represent the face in the most efficient way. Lin (2000) refers to this as the feature extraction part of the face-recognition process. Feature extraction can be categorised into two main classes: geometric feature based, and template based. The template-based approaches, such as PCA seem to be more commonly used in face image representation [Howell (1999)]. In this project, only the feature-based representation is considered – PCA to be specific. Turk and Pentland (1991a) refer to feature-based representation as an information-theory-based system.

Face reasoning is about deciding on matching results, on what approach to follow in order to decide if face image A best matches face image B. This is a pattern-recognition problem and a number of techniques exist in the literature for solving this problem. They range from nearest-neighbour classifiers such as Euclidian distance (L2 norm), City-Block (L1 norm), Mahalanobis distance measure to Neural Networks and probability models such as the Bayesian model [Howell (1999), Moghaddam et al (1999)].

After a study of the proposed generic frameworks for face recognition, the following categories were identified. These, together with the system requirements, formed the basis of our distributed face-recognition model.

In our design we identified the following core broad categories:

- Image acquisition and presentation to the system,
- Feature extraction and representation,
- Pattern recognition,
- the presentation of results **(**recognition reasoning in Howell's (1999) terms*)*.

4.4 DRUBIS – component design

DRUBIS has been designed iteratively to meet different goals. At each level the design was implemented and some simple applications were developed. In this section we present our design and the simple applications (with minimal components) that were implemented at each level, which we have called versions.

4.4.1 DRUBIS version 1 – a simple system using web services

The first system that we implemented ran as a single web service and exposed a number of other functionalities that were not transparent from the MIT code. This service used directory structures to keep and present face images to the system. The following diagram can best present this service.

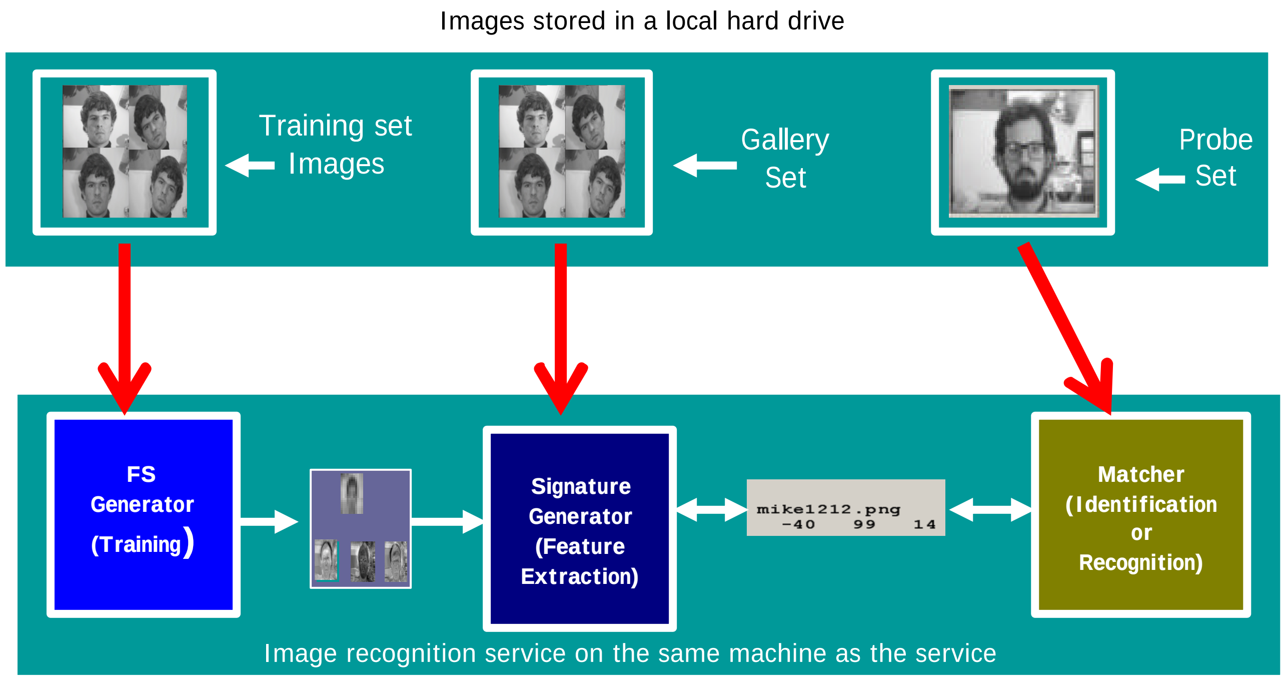
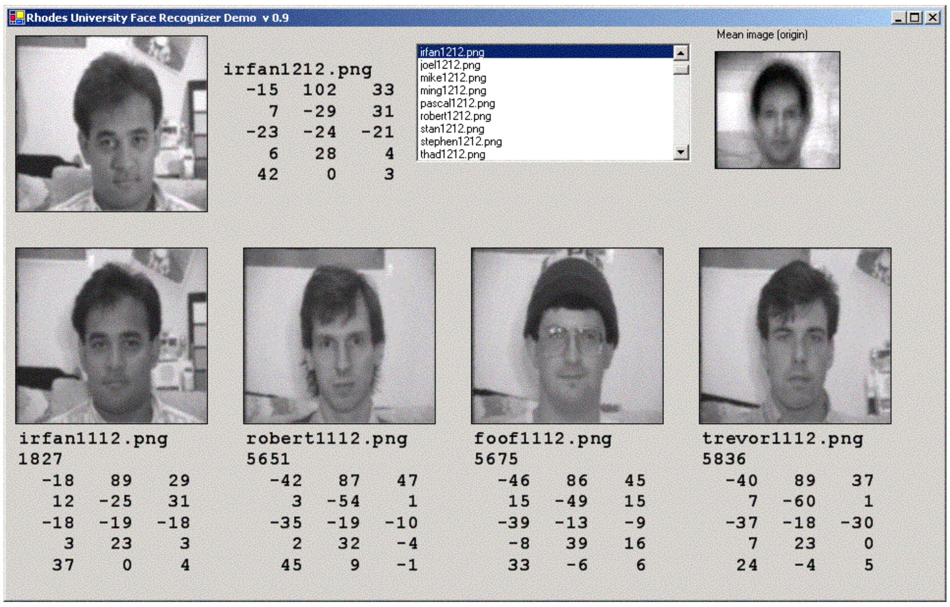

Images stored in a local hard drive
Training set Images
Gallery Set
Probe Set
FS Generator (Training)
Signature Generator (Feature Extraction)
mike1212.png -40 99 14
Matcher (Identification or Recognition)
Image recognition service on the same machine as the service

Figure 4a: Version 1: Web service driven face recognition back-end service

Here the service exposed the three core components. The first is the face-space (FS) generator, which does the training part. Then the second component, the signature generator takes an image and computes a corresponding signature. These signatures are computed on the fly and they are not kept by the system. Finally the third component, the matcher, takes a probe image, calls the signature generator to compute a signature for that image and also to compute a match from the gallery set image signatures.

4.4.1.1 Applications developed from this implementation

This design was the first step in porting the MIT code to a web service environment. The next step was to expose the outputs from various stages of the face-recognition process. We achieved this by implementing a Windows client application. We define client applications as applications that send processing requests to the web services. In this case it can be either a Windows form-based application sending a request to a web service or a web service sending requests to another web service. The presentation component at that stage looked like the following:



```
irfan1212.png
 -15  102   33
   7  -29   31
 -23  -24  -21
   6   28    4
  42    0    3
```

```
irfan1112.png     robert1112.png    foof1112.png      trevor1112.png
1827              5651              5675              5836
 -18   89   29     -42   87   47     -46   86   45     -40   89   37
  12  -25   31       3  -54    1      15  -49   15       7  -60    1
 -18  -19  -18     -35  -19  -10     -39  -13   -9     -37  -18  -30
   3   23    3       2   32   -4      -8   39   16       7   23    0
  37    0    4      45    9   -1      33   -6    6      24   -4    5
```

Figure 4b: Client application using images stored on local disk

The first client application developed and visually exposed the outputs of various stages of a face-recognition framework. This application was both physically and logically distributed over two machines (*Facerec* and *Netserv*). This client application "demo" served as a small-scale prototype in our overall design of the framework. The prototype was modified, scaled and logically spread to a number of components.

The above client application snapshot shows the face-space representation of images (i.e. images reconstructed from face space), the matching scores (i.e. zero where the recognised image is found to match a stored image) and lastly the scaled image signatures. The first row shows a probe image with the image name and signature (scaled) and a mean image computed from the face-image space. The second row shows the matching top 4 images with their names, matching scores and corresponding signatures.

The above demo uses only still images for its operation and its primary purpose is to visually expose the nuts and bolts of face-recognition algorithms in a way that may be easily understood by beginners. The images are stored on the server and passed between client and server (through a web service) as XML data.

4.4.1.2 Limitations on the design of DRUBIS version 1

The design improved our understanding of the PCA-based face-recognition algorithm. It gave us some hands-on experience with the algorithm, from porting it to C# up to extending it to expose different functionalities and outputs. It also allowed us to use the MIT data to validate that our distributed C# implementation was giving identical results to the original MIT code. On the other hand the algorithm is not efficient. The need to re-compute image signatures each and every time a new probe is projected to the gallery set is unnecessary. This problem can be easily solved by pre-computing signatures and storing them in a database. We subsequently extended our design to cater for this.

Another extension arose from the requirements of a design goal: to construct a system that may easily be plugged into a number of simple applications. This requirement necessitated a central database which can be easily accessed by different applications. Some applications demand real-time interaction with the server and we upgraded our framework to cater for both cases.

4.4.2 DRUBIS version 2 – a database-driven web service

The concept of separating data from the processes that process data is not new in computer science. The second step in meeting our predefined requirements is to separate the image information from the web service. This ensures easy and central access to the images by different applications.

At this stage we have our own Rhodes University image sets. The images are grouped into a number of categories such as gender, ethnicity, and tutorial group. Image management demands a more structured and formal approach. The following design was adopted:

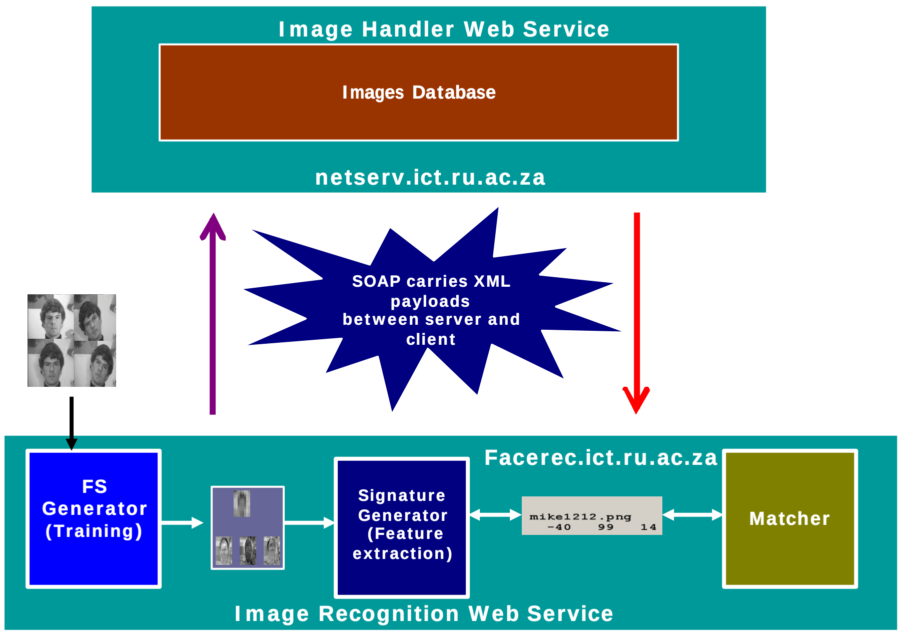


Image Handler Web Service
Images Database
netserv.ict.ru.ac.za
SOAP carries XML payloads between server and client
FS Generator (Training)
Signature Generator (Feature extraction)
Facerec.ict.ru.ac.za
mike1212.png -40 99 14
Matcher
Image Recognition Web Service

Figure 4c: Logical overview of a database-driven DRUBIS version 2 architecture

Above, there are two web services, Image Handler *and* Image Recognition web services distributed over the Internet. These together make the fundamental platform for our face-recognition toolbox where different face-recognition performance characteristics can be monitored. Our real-time face-recognition demonstration application is also plugged into this back-end system. The next section discusses each web service in detail.

4.4.2.1 The Image Handler web service

The *ImageHandler* service is principally concerned with storing and retrieving image information from an SQL database. Our aim is to separate the recognition service logically from the image management service. The benefits of such a design are obvious: a more modular system that is easy to debug; code that is easy to understand; modularity promoting code reuse, etc. The Image Handler service also allows us to keep full-colour original images in the database alongside the corresponding monochrome, preprocessed and cropped versions that are used in the recogniser. For the presentation layer, such a design allows us to present the user-friendly version of the enrolled image, rather than the recogniser-friendly variant.

A number of image attributes are provided by the image service. Characteristics of the image might include: its size, its owner, its photographic type (e.g. whether colour or greyscale), the image signatures that belong to it (all images have a number of different signatures generated for the recognition tests), and the groups to which the image belongs. *ImageHandler* also provides an interface to the underlying database by handling the queries directed at and the insertions made into the image database.

The *ImageHandler* web service is broken down into a string of small web functions, each dealing with a specific functionality of the image management component of the overall system. These subroutines are grouped into:

- Image information management: this comprises subroutines dealing with retrieving and inserting images into the database e.g. `ReturnallProbeImgNames("week4")`.
- Image signature information management: this deals with computing, storing and retrieving signature information (such as face spaces, signature size), e.g. `GetGallerySig (spaceName,ImageName)`, `GetProbeSig(spaceName,ImageName)`.
- Image owner information management: these subroutines manage the image owner information such as the grouping to which the individual belongs, his or her gender or race. `ReturnImagesByCateg(gender,race,Imgtype)`.

4.4.2.2 The Image Recognition web service

This web service implements the algorithm used for the last two stages of our generic distributed framework for face recognition, feature extraction and matching. The service is a core engine for signature generation and recognition. This is the only service that will be modified if a different matching or feature-extraction algorithm is required.

This service can be further broken down as in the *Nemesis* component of the Rzeszutek (2002) distributed face-recognition system. The difference is that *Nemesis* is made up of multiple high-performance machines, all sharing the computational load of the face-image-recognition component of the process. In this environment, image frames are backed up on more than one machine: on *Demeter* the images are stored as video and on *Nemesis* as images for the actual recognition process. With the small-scale nature of our system, such a use of resources cannot be justified and is also unnecessary, because our system can be easily extended to distribute more of its functionality across the Internet.

The three-dimensional system operational model developed by Yang et al (2002) closely resembles our image-recognition web service. The processing scheme is distributed over three levels, with intelligent detection on a single PC, the feature extraction scheme in the Intranet and a recognition scheme (graph-matching in this case) distributed via the Internet. In our case, the feature extraction and matching are both handled at the distributed Internet level. A single PC is used to input the data and receive the recognition results and its position in our framework resembles that of client applications.

As shown in the work of Yang et al (2002) and Rzeszutek (2002), the framework is tightly coupled to the underlying algorithm being hosted, following the principle of the algorithm dictating the design. Our framework hosts the PCA algorithm from the MIT media Lab [Turk and Pentland (1991a)]. The web service structure corresponds to the PCA algorithm's critical stages: face-space generation maps to training the algorithm; enrolling the face image relates to the computation of its signature and its storage in a database; lastly, computing the matching results corresponds to the generation of an image signature for the probe image (the image to recognised) and computing matching scores against registered signatures from a database to get a best approximate match.

The key web functions from this web service, grouped by categories, are:

- <u>Face-space generation</u>: two web functions perform this - `createImageSpace(spacename,imageNames)`, `createImageSpacewithNeigF (imageNames, spacename, numEigenfaces)`.
  What distinguishes these two functions is that in one you specify the size of the signature (the number of eigenfaces in your space) while in the other you generate a signature of size n, where n is the number of images used in face-space generation.

- <u>Signature generation</u>: `getFeature(spaceName,imageName)` is a web function exposed for signature generation. The image feature (here its signature) is generated against a given space.

- <u>Matching</u>: `findMatches(spaceName, imageName, howMany)` finds an ordered list of the best 'howMany' matching results for a given image name. This web function uses a pre-computed face-image signature from the SQL face database via the Image Handler web service.

This composite web service is the engine for the system. But web services can in turn make use of other web services which implement sub-functionality. In our design of providing for third-party software integration of DRUBIS, we further decomposed this service into three subcomponents, each of them a web service in its own right. The next section discusses client applications using this implementation of DRUBIS.

4.4.2.3 Applications developed from this implementation

At this stage our face-recognition system is made up of two back-end servers (*ImageHandler and ImageRecServ*) and a number of client applications may be plugged onto these servers for different face-recognition tasks. At present, the system has two operational modes: batch processing and a real-time mode.

4.4.2.4 The batch-processing mode of DRUBIS version 2

This mode serves as a PCA face-recognition toolbox available for experiments and analysis of the PCA algorithm performance. In this mode, the probe images are pre-registered with the system. The goal is usually to extract hit rates using a specific face space or after some preprocessing on the gallery, training and probe-set images. In this mode only the two back-end servers running the web services and an SQL data base server are used.

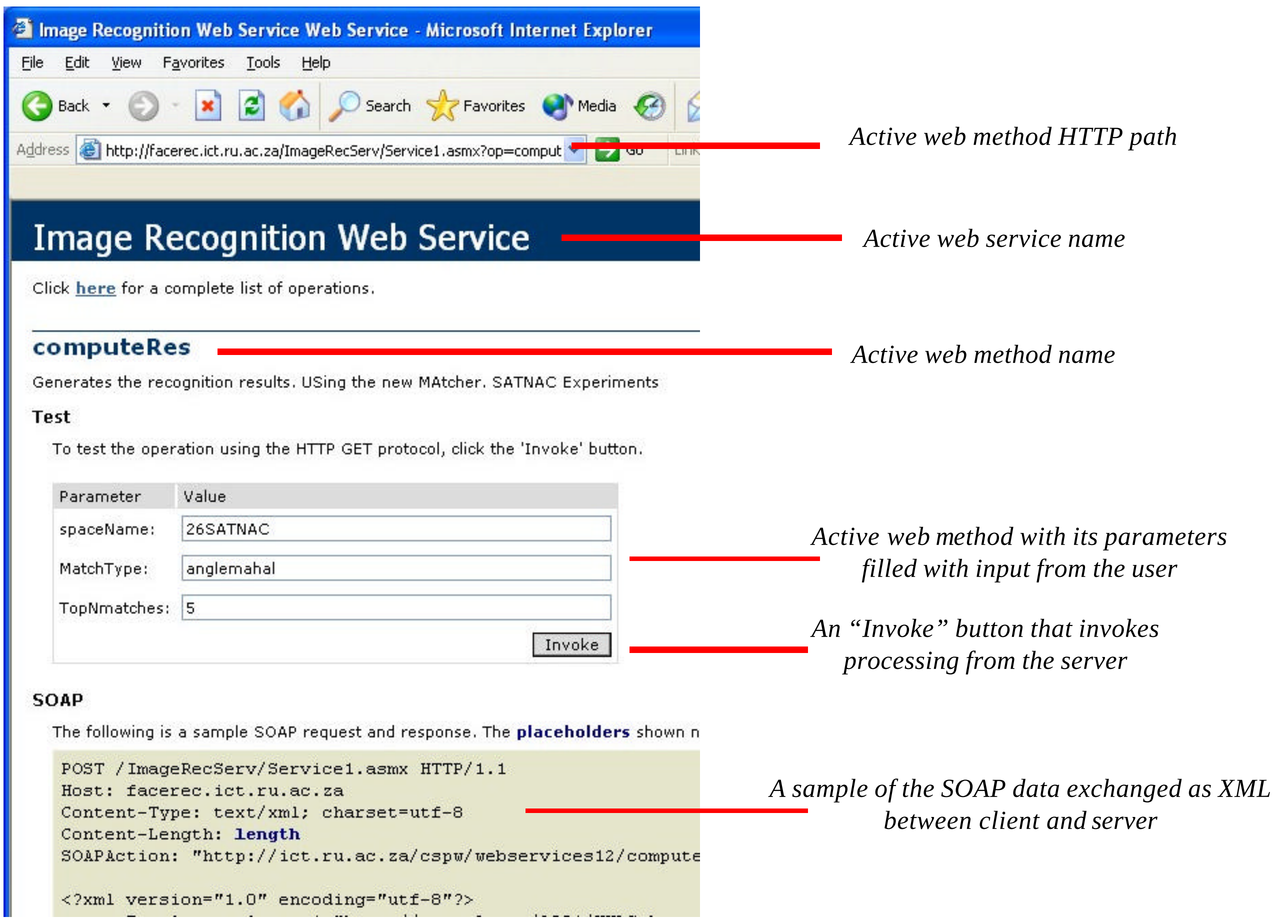


Image Recognition Web Service Web Service - Microsoft Internet Explorer

http://facerec.ict.ru.ac.za/ImageRecServ/Service1.asmx?op=comput

Image Recognition Web Service

Click here for a complete list of operations.

computeRes

Generates the recognition results. USing the new MAtcher. SATNAC Experiments

Test

To test the operation using the HTTP GET protocol, click the 'Invoke' button.

Parameter	Value
spaceName:	26SATNAC
MatchType:	anglemahal
TopNmatches:	5

SOAP

The following is a sample SOAP request and response. The **placeholders** shown n

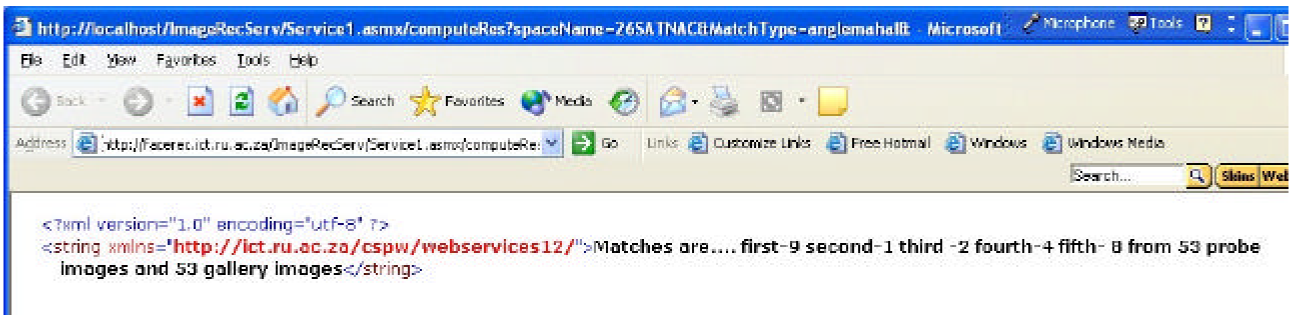
```
POST /ImageRecServ/Service1.asmx HTTP/1.1
Host: facerec.ict.ru.ac.za
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://ict.ru.ac.za/cspw/webservices12/compute

<?xml version="1.0" encoding="utf-8"?>
```

Active web method HTTP path
Active web service name
Active web method name
Active web method with its parameters filled with input from the user
An "Invoke" button that invokes processing from the server
A sample of the SOAP data exchanged as XML between client and server
Invoke

Figure 4d: A snapshot of an active web service method[7]

We computed our performance scores using the above interface, with no Windows client application. Pressing the "Invoke" button above activates the processing. An image handler web service supplies images as required by this service. The output from the `computeRes` method shown above looks like the following:



http://localhost/ImageRecServ/Service1.asmx/computeRes?spaceName=26SATNAC&MatchType=anglemahal&t - Microsoft

http://facerec.ict.ru.ac.za/ImageRecServ/Service1.asmx/computeRe:

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://ict.ru.ac.za/cspw/webservices12/">Matches are.... first-9 second-1 third -2 fourth-4 fifth- 8 from 53 probe
  images and 53 gallery images</string>
```

Figure 4e: Output from the `computeRes` web method.

The output above is shown as an XML string. The matches from first to fifth position are listed. The size of the gallery and probe sets is also listed. The above output can be directed to another

[7] Since the time of first writing, the newer release of web services software has disabled remote use of some of the testing functionality shown here, because of security concerns.

application if, for example, the service is consumed by another application. The next section looks at the client applications that work at a real-time mode.

An interesting consequence of the design and use of a web service via a browser on a remote machine is that we have tended to catalog and isolate each of the experiments; they are mostly stored for future repeat experiments. For example, in Figure 4D we have used a face space called "26SATNAC". The source images from this face space are stored in our databases, and the set of images used for gallery and probe sets are also preserved. This makes it relatively easy to perform comparative experiments if we introduce a new cropping or image normalisation technique.

4.4.2.5 Real-time processing mode of DRUBIS version 2

In this operating mode, we implement a larger scale real-life demonstration application. Two additional client Windows applications are added to the image recognition and image handler web services. One Windows client application is dedicated to capturing and enrolling face images and the other is responsible for capturing images to be recognised and then presenting results. The client applications run on different machines: altogether this demonstration operates on four physical PCs and was implemented like this for the 2003 SASOL National Science Festival.

4.5 Science Festival face-recognition demonstration application

This section outlines the design, implementation and performance of our 2003 SASOL Science Festival[8] face-recognition demonstration application. The demonstration serves as a low-scale proof of our concept of web-service-driven, real-time biometric identification system. The "demo" is small, but large enough to demonstrate the capabilities of the technologies involved and the feasibility of our framework.

The purpose of the "demo" was to assess realistically the accuracy and usefulness of our proposed framework in the design and implementation of distributed biometric systems. The demonstration application specifically fulfils the following main aims:

- it serves as a proof-of-concept for live rendering of our distributed face-recognition web-service-driven environment;

[8] SASOL Science Festival is an annual schools science festival aimed at promoting science amongst school kids.

- it brings to the public a greater awareness of face-recognition technologies and the capabilities of Microsoft .NET XML web services;
- the demo also gave us an opportunity to assess the accuracy of the code re-use feature (write once and use it many times from anywhere) claimed by web services evangelists;
- lastly, since 4 different PCs were involved in the process, we managed successfully to simulate on a small scale the possibilities of a larger biometric recognition system.

The rest of the following sections cover the design and the performance of our system over the Science Festival period. It should be mentioned that no statistical data was collected over the demonstration period. The reason for this was that in the past data collection had sometimes presented problems which we wished to avoid in a "live", running system. We needed to monitor the system's ability to perform without such problems as crashing due to any inability to handle live data or network congestion.

4.5.1 System physical design

This application is distributed over four machines:

- **Facerec** hosts the face-recognition web service, *ImageRecServ*;
- **Netserv** hosts the image information handler web service, *ImageHandler* and the SQL face database;
- **Pweb02** hosts the face-recognition client;
- **Pweb03** hosts another face-registration client.

There are two client applications each connected to a web camera for capturing face images. The back-end is made up of two servers **Facerec** and **Netserv**. The servers are not visible to the human user and communicate with clients only via the local Intranet. The interaction between different system parts is discussed in the next section.

At the Science Festival where we used this system, we were set up rather like an Expo fair. Passers-by were channelled past the registration client machine where they could enrol their faces. They then went further along the counter to the recognition client, where we attempted to match a new image of their face against their prior registration.

4.5.2 Logical design – face-registration client design

Prior to face recognition each face image must be first registered with the system. This is done through the face-registration client interface. The process of registering a face image involves the following:

4.5.2.1 Capturing a face image

This process involves capturing an image using a web camera and saving this image in a local temporary file. A copy of the image is scaled to 128 by 128 pixels and converted to greyscale. This copy is sent to the *ImageRecServ* web service running remotely on **Facerec**.

The original colour image is sent to the *ImageHandler* web service to be stored. Both a colour image and its corresponding signature (computed relative to the current face-space) are stored in different tables in the database in order to achieve the successful enrolment of a face image.

4.5.2.2 Generating the image signature

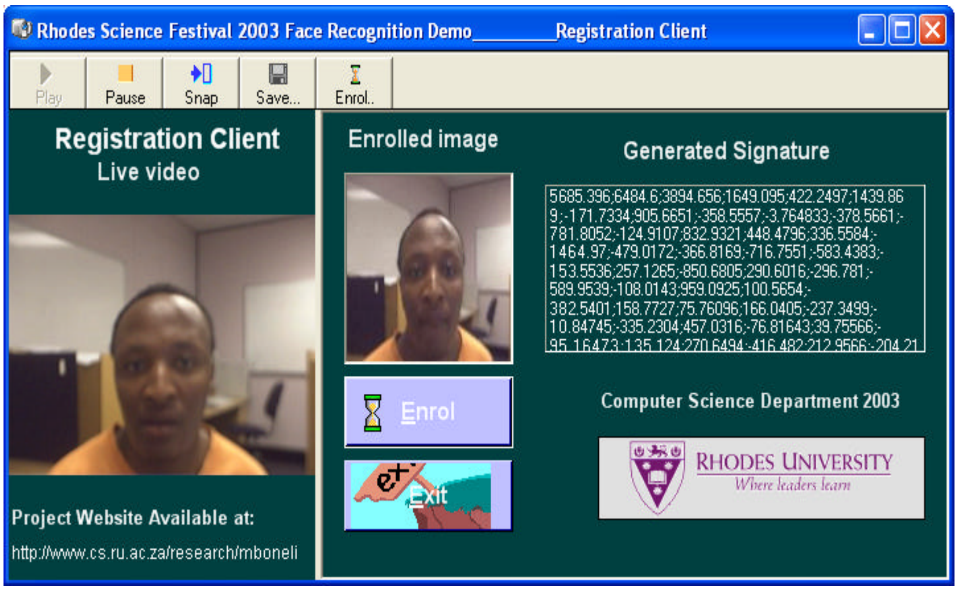
The *ImageRecServ* web service generates an image signature. It accepts a grayscale image from the client and the PCA algorithm is used for the extraction of the principal or most discriminating components of the face image. These components effectively represent a face-image signature. The signature is then sent back to the client, where it serves as a confirmation that the signature was effectively generated. Then the client displays it on the client application form and sends another copy to the *ImageHandler* web service.

4.5.2.3 Storing the image and its signature

ImageHandler handles the insertion of an image into a corresponding table in an SQL database. The process involves first the allocation of an appropriate name for the image. The image and the signature must share the name as a link since they are stored in separate tables in the database. The accuracy of this process is ensured by the client application. All the image registration is done on the client instead of some of it being delegated to the *ImageRecServ* web service; the signature is not sent back to the client but instead is sent straight to the *ImageHandler* web service in order to minimise network traffic.

4.5.2.4 Registering the client interface

The interface for the registration client is designed to be simple and very easy to use. Only one button is involved when an image is registered. The rest of the steps are hidden from the user. Feedback after a successful registration is given through the display onscreen of a copy of the registered image, together with its corresponding signature. The interface is shown below.


Rhodes Science Festival 2003 Face Recognition Demo________Registration Client
Play
Pause
Snap
Save...
Enrol..
Registration Client
Live video
Enrolled image
Generated Signature
Enrol
Exit
Computer Science Department 2003
Project Website Available at:
http://www.cs.ru.ac.za/research/mboneli
RHODES UNIVERSITY
Where leaders learn

```
5685.396;6484.6;3894.656;1649.095;422.2497;1439.86
9;-171.7334;905.6651;-358.5557;-3.764833;-378.5661;-
781.8052;-124.9107;832.9321;448.4796;336.5584;-
1464.97;-479.0172;-366.8169;-716.7551;-583.4383;-
153.5536;257.1265;-850.6805;290.6016;-296.781;-
589.9539;-108.0143;959.0925;100.5654;-
382.5401;158.7727;75.76096;166.0405;-237.3499;-
10.84745;-335.2304;457.0316;-76.81643;39.75566;-
95.16473;135.124;270.6494;-416.482;212.9566;-204.21
```

Figure 4f: Registration Client Interface

Above, is a simple well-labelled client application interface with a registered image and its corresponding signature.

The next section looks at the client recognition system interactions.

4.5.2.5 Face-recognition client design

The recognition client captures an image to be recognised, sends this image to the *ImageRecServ* web service for recognition along with the top six matching images. The interactions amongst different system components can be diagrammatically represented as follows:

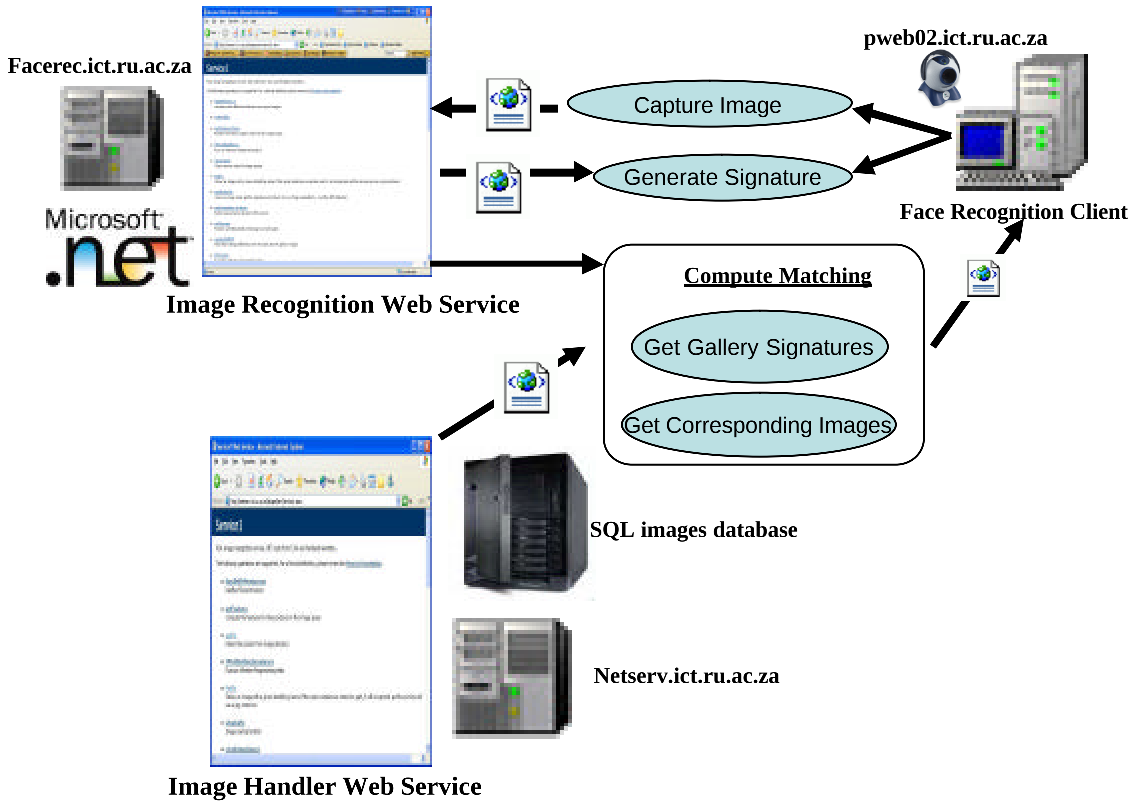


Figure 4g: Interaction between different system components of DRUBIS

In the diagram above, the information is exchanged as XML data between the face-recognition client and the two web services running on remote machines. There are three steps involved: capturing a probe image (using a webcam on **pweb02.ict.ru.ac.za**) and sending it to image-recognition web service, generating an image signature and computing matching results. The first two steps have already been discussed in the above section on the design of the registration client. In this section we cover only the computation of matching results.

4.5.2.6 Computing matching results

The process of computing a match involves projecting a probe signature onto a list of registered signatures and using an Euclidian distance measure to find the top six matching signatures. This whole process takes place within the **facerec.ict.ru.ac.za** machine running the *ImageRecServ* web service. After the images corresponding to the matching signatures have been fetched from the database via the *ImageHandler* web service, and passed back to the client application as XML data, the client application displays the images as matching results.

4.5.2.7 The recognition client interface

The interface was also designed to be simple and easy to use. The top six matches together with matching scores are given as feedback. The other distributed components of the system are hidden from the user. A single click on a "Recognise" button gives the matching results. The system interface is shown in the following image:

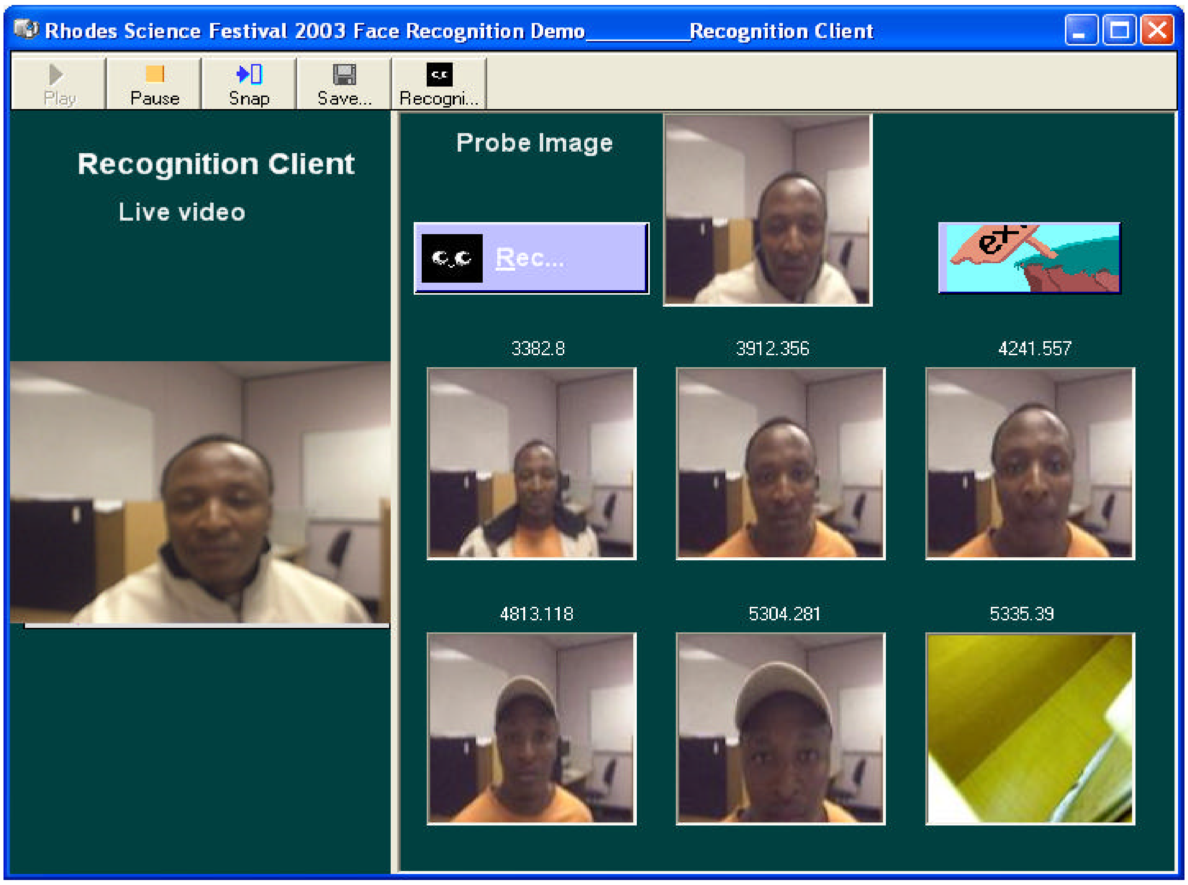


Figure 4h: Recognition Client Interface

Above, a probe image together with matching results is shown. Above each matching image is a matching score. The score becomes zero if the same image is found in the gallery set i.e. with no difference between them. The scores represent the closeness between the probe image signature and the corresponding gallery signatures. A small score value means that the probe image is similar to a gallery image while a big score means that the probe is very different from the corresponding gallery image.

4.5.2.8 The overall system operation

The images are registered with a specific face space. The face space training images must resemble the target environment: the face-space must be able to capture a reasonable discrimination amongst the registered images (for optimum results features such as background, web camera type, and illumination must be normalised).

Pressing a "Recognise" button on the face-recognition client application for the first time loads a face space together with images of individuals known and registered to the face space. The face space and known registered individuals (comprising computer-generated image names and image signatures) are cached to a static structure created to avoid recompiling a face space every time a new probe is to be recognised. This improves the average time taken by a system to recognise one image.

There is a problem though, in that there are two clients PCs involved and some individuals might be registered at a time when the face space has cached a stale copy of known individuals. The implication is that the newly registered individuals will not at that moment be in the face space. We therefore need to detect when our cache is stale and we achieve this by adding a lightweight check of the number of known individuals in the space, and then comparing this number with the total number of registered images in the database. If the cache is discovered to be stale, we incrementally freshen it by adding to it only those latest individuals from the database that have not yet been cached.

A probe image signature is then matched against the known individuals' signatures from the face space. The results are then presented to the client application. XML data is exchanged between different system components in all these interactions. An image signature can be another type of biometric signature and the framework will still be intact since the XML data which is exchanged is not hardwired to represent only face signatures.

The above scenario highlights the interactions involved between components when performing one simple operation to project a probe signature in a gallery set.

4.5.3 Observations from the Science Festival

The system achieved some of its objectives over the Science Festival period, namely: to create awareness of face recognition and of Microsoft .NET XML web services. Poor performance in recognizing face images was apparent (our matcher used only the Euclidian distance measures in this experiment). This was attributed to an uncontrolled real-life environment. Most face-recognition systems are evaluated in laboratory-like environments where all the noise is filtered out. The experiment made us acutely aware of the gap between laboratory and practical use, and highlighted a need for better noise-tolerant algorithms (or approaches) for face recognition. DRUBIS clearly demonstrated the role that can be played by web services in distributed biometric systems and highlighted a need for noise-tolerant image-recognition algorithms.

4.6 DRUBIS version 3– enhanced design with third-party software support

The implementations discussed above have been tightly coupled to the underlying file structures and SQL databases. These made it impossible to integrate the exposed web services with third-party software. Web services, however, are marketed as a technology that facilitates system integration. In this section we look at this in depth and attempt to find a solution to the problem raised by such tight-coupling to proprietary software.

We made a decision to refactor the image-recognition module into 3 separate web services. The web services are:

- Image Space Generator,
- Image Signature Generator,
- Matcher.

Each of these three web services runs independently, without knowledge of the other two. This design improves the face-recognition system by enforcing the division into separate phases of the separate applications. As mentioned earlier, such strong decoupling facilitates experimentation and rewiring of the modules [Ndlangisa and Wentworth (2003)]. The following diagram shows different clients interacting with DRUBIS over the Internet.

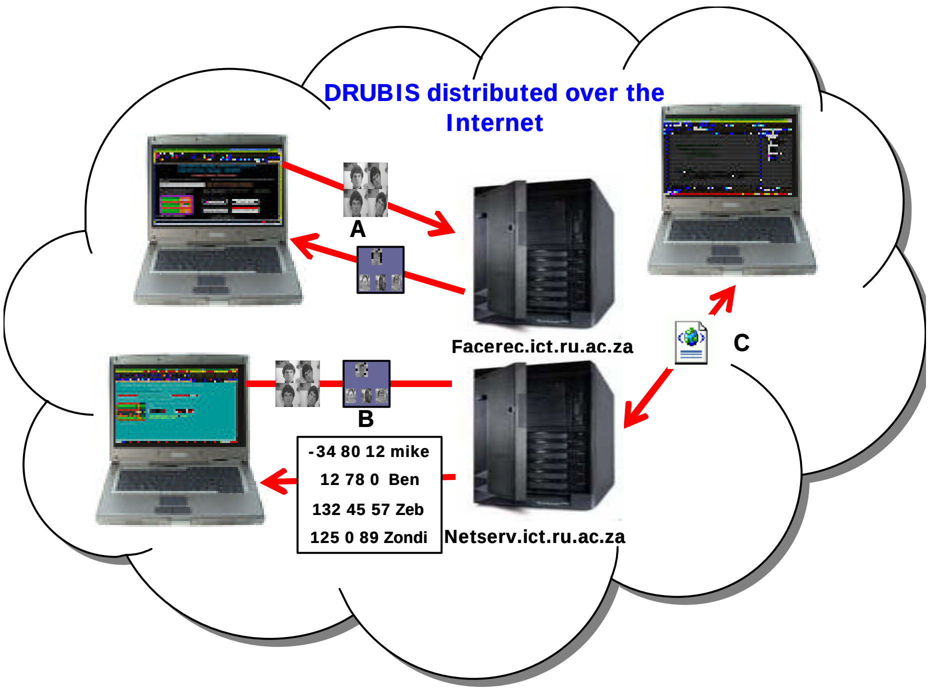


Figure 4i: Clients interacting with DRUBIS Version 3 over the Internet

In the diagram above, there are three remote client machines interacting with the DRUBIS web services hosted by **Netserv** and **Facerec** respectively. In **A,** a client passes a list of images to the image-space generator web service, via an ASP.NET web page. The web service returns an image space to the user who can then either use it within his or her applications, or use the DRUBIS signature-generator web service to compute image signatures. In **B,** a client passes an image space, together with images to be used in signature generation, and gets back signatures and corresponding image names. In **C**, a remote client integrates his or her applications with the DRUBIS matcher, and XML data is exchanged between DRUBIS and the guest users' software.

The image handler web service is not further discussed, since we do not plan to provide dedicated persistent image storage to guest users. The image-space generator generates an image space which the image-service generator needs in order to compute an image signature. The idea here is that the users possess their own image spaces and download them to their own machines for safekeeping and storage. If they want to use a specific image space with DRUBIS, they can upload it first. DRUBIS works either with image spaces generated from within, or from proprietary image spaces generated by user software (as long as the image space conforms to the required size of 128 by 128 pixels for its images).

4.6.1 Implementation

4.6.1.1 Image-space-generator web service

The image-space-generator web service accepts training-set images from the user and generates an image space based on user specifications. Note that this service can be used to generate any kind of classifier, whether for faces, cars, or even for fruits!

The user also has to specify the number of eigenfaces to be kept, i.e. the dimensions of the image space. The images are uploaded either as a single zipped file [Krueger (2001)], or individually.

ASP.NET is used to enhance the interface of this web service. The file upload is done via ASP.NET and some scripting in C# is used to keep track of the number of files uploaded and to set the file paths within the server hosting the web service.

When the image files are all uploaded, the user presses the “Generate Space” button that calls a corresponding web function in the web service. This function first checks to see if it is a zipped file that has been loaded. If it is a zip file, the function first unzips it and then uses all the JPEG images to generate an image space.

This service also generates a list of weightings derived from eigenvalues that encode variance amongst training images. These weights are used for more sophisticated vector-matching algorithms, e.g. those of the Mahalanobis family. This information may therefore need to be passed back to the user for subsequent presentation to the matcher.

The Image-Space-Generator ASP.NET Web Service has the following interface:

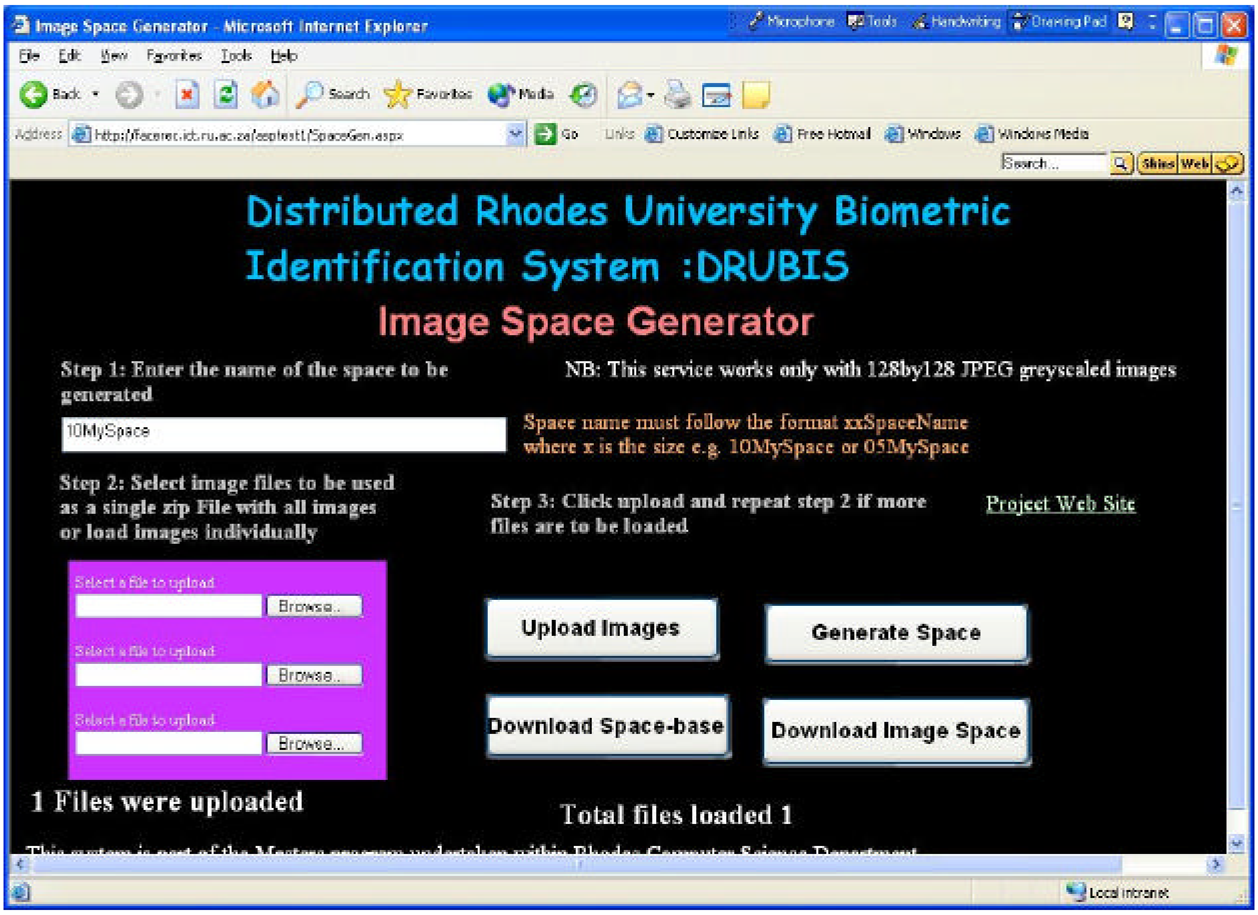

Distributed Rhodes University Biometric Identification System :DRUBIS

Image Space Generator

Step 1: Enter the name of the space to be generated

10MySpace

NB: This service works only with 128by128 JPEG greyscaled images

Space name must follow the format xxSpaceName where x is the size e.g. 10MySpace or 05MySpace

Step 2: Select image files to be used as a single zip File with all images or load images individually

Step 3: Click upload and repeat step 2 if more files are to be loaded

Project Web Site

1 Files were uploaded

Total files loaded 1

This system is part of the Masters program undertaken within Rhodes Computer Science Department

Image Space Generator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://facerec.ict.ru.ac.za/eeptest1/SpaceGen.aspx
Select a file to upload
Browse...
Upload Images
Generate Space
Download Space-base
Download Image Space
Local intranet

Figure 4j: Image Space generator ASP.NET interface.

The main objective of this service is to let the user create, manage and own his or her image space. The user can either download base images that make an image space or the actual image space (the *.dat files with pixels represented as numbers instead of colour) or both. Base images or eigenfaces are useful in determining the quality of an image space; we cover this in the next chapters. Exposing the face generator allows us to test easily; for example, we can look into the extent to which demographic groupings (Asian, African, male, female, and children) impact on recognition rates.

4.6.1.2 Image-signature-generator web service

The image-signature-generator web service takes a face space, and images to be projected to a face space, and produces image signatures. Image signatures can be returned to the user as a text file or passed directly as XML to the user's software integrated with our system or, if one file is loaded, an image signature is shown on the ASP web page.

ASP.NET is used for uploading files. This service allows remote users to upload their face spaces and image files and have the option of integrating web services with their applications. This should allow remote users to experiment with their own images. However, we are not intending to provide a permanent, persistent service to guest users

4.6.1.3 Image-signature matcher

The goal of an image-signature matcher web service is to take a signature and compute its closeness to another signature or signatures. The signatures are simple vectors generated from a signature-generator module. This service can be used for any other distance measure between two vectors, i.e. outside the scope of image identification.

There are a number of competing ways to measure the distance between two signatures, i.e. ways of finding nearest neighbours in a face space. They all have different merits and limitations. In this web service, the city block (L1 norm), squared Euclidean (L2 norm), Angle, and Mahalanobis distance measures are implemented, and we incorporated the Mahalanobis distance measure to the first 3 distance measures.

City-block (L1 norm) Distance: this gives an average distance across dimensions. The effect of a single large difference in one variable is dampened.

$$dist(x,y) = \sum_{i=1}^{k} | x_i - y_i |$$

Squared Euclidean Distance (L2 norm): this is most common, and the original MIT Media Lab code uses only this distance measure. It is the geometric distance between the objects in the p-dimensional variable space. Squared Euclidean distance puts more weight on objects that are further apart.

$$dist(x,y) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

Angle Distance: given two vectors, this measure computes the angle between them. The vectors with the smallest angle between them are assumed to be the closest.

$$dist(x,y) = \frac{\sum_{i=1}^{k} x_i y_i}{\sqrt{\sum_{i=1}^{k} (x_i)^2 \sum_{i=1}^{k} (y_i)^2}}$$

The above three distance measures do not take into account the variance from the training data. A statistical distance measure called the Mahalanobis takes this into account. We implemented a Mahalanobis distance as:

$$dist(x,y,z) = -\sum_{i=1}^{k} x_i y_i z_i$$

Where vector z is computed from the training set by:

$$z_i = \sqrt{\frac{l_i}{l_i + a^2}} \approx \frac{1}{\sqrt{l_i}}, a = 0.25$$

Where l_i = eigenvalue of the *i*th vector.

In our attempt to cater for variance in the training set we combined the variance scored with the City-block and Euclidean measures. Moon and Phillips (1998) refer to these as L1 + Mahalanobis, L2 + Mahalanobis and Angle + Mahalanobis distance measures.

The L1 + Mahalanobis distance measure is defined as:

$$dist(x,y) = \sum_{i=1}^{k} | x_i - y_i | z_i$$

The L2 + Mahalanobis distance measure is defined as:

$$dist(x,y) = \sum_{i=1}^{k} (x_i - y_i)^2 z_i$$

The Angle + Mahalanobis distance measure is defined as:

$$dist(x,y) = \frac{\sum_{i=1}^{k} x_i y_i z_i}{\sqrt{\sum_{i=1}^{k} (x_i)^2 \sum_{i=1}^{k} (y_i)^2}}$$

Moon and Phillips (1998) simplified the above formulas in their study on the Analysis of PCA-based Face-Recognition Algorithms.

The image-signature-matcher web service is also implemented as an ASP.NET front end with a web service back end, allowing us, for example, to specifically create a tailored face space for e.g. a group who work in a given laboratory, or for those individuals authorised to use a photocopier. One can upload a gallery of signatures (enrolled signatures to match against a probe signature), a probe signature file, and the variance-scores files (drawn from the face space used to generate signatures), and get back a results file with matches that can use the ASP.NET interface. Alternatively, remote users need only integrate our image-signature matcher into their software to compute a closeness measure and do all the matching on their own machines. Preliminary findings and a more concise summary of this aspect of the work appear in the proceedings of SATNAC 2003 [Ndlangisa and Wentworth (2003)].

4.7 DRUBIS logical design – summary

The following table gives a high-level view of different implementations of DRUBIS.

DRUBIS - version	Description	Client Applications	Available to third party	Limitations on this version
Simple web-service-driven DRUBIS version 1	- A web service version of the MIT code, implemented using C# on a Microsoft .NET platform. - Exposes core phases of the face-recognition process.	- Simple Windows-based face-recognition demonstration applications. - No real-time recognition implemented.	- This service is NOT available to third-party software. - But it is consumed by Windows applications running on remote machines. Knowledge of files is needed to consume this service	-NO stored signatures. -Not accessible to third-party applications. -Very abstract and shows minimal outputs from each face-recognition phase.

Database-driven DRUBIS version 2	-Improves on the limitations of the first DRUBIS. -Image-handler web service supports the image-recogniser web service. -SQL database implemented to centrally manage images, with searchable categorisation, and named gallery and probe sets.	-Acts as a toolbox where different PCA-related tasks are performed using only the two web services. -Real-time face recognition using two client applications (Science Fest demo).	-No third-party software support available. - But it is consumed by Windows applications running on remote machines. Knowledge of files is needed to consume this service	-No third-party support. -Separates image data from the face-recognition functionality - But does not show the functional distribution of face-recognition phases
Fully distributed DRUBIS version 3	- This version of DRUBIS is made of three web services, each dealing with a unique key phase of the face-recognition process. -ASP.NET is used to enhance interfaces and aid in uploading Internet guest users. - Aimed at giving third-party software support and emphasizing the functional independency of the key phases of the face-recognition process.	- Clients are guest users interacting with the system through the Internet. - An advanced matcher from this implementation is consumed by the toolbox to compute sophisticated matching scores.	-This DRUBIS implementation is primarily aimed at giving easy third-party support. -Third-party applications are easily integrated into our system. -But we do not intend giving permanent, persistent image storage support to Internet guest users,	-The price paid by distributing the core face-recognition service is that, before computing a signature, you have to upload images and an image space to be used.

Table 4a: A summary of different DRUBIS implementations

4.8 Physical design - summary

Our framework does not aim to cater for a large-scale biometric system aimed at processing thousands of images. DRUBIS is a small-scale laboratory-like experiment. On the other hand, our goal is to generalise the framework enough for it to be of value to the designs and implementations of larger-scale systems.

Rhodes Face Recognition System is distributed over two back-end machines and two client machines, and this is enough as a proof-of-concept for our framework. The four machines communicate through XML via SOAP over HTTP. The system can be easily extended to include more machines. For example, extra machines can be added, one to host the SQL database and others to increase the number of client machines. This approach was not of current interest to us since our primary goal in distribution was not improving speed by e.g. distributing the workload.

We believe this framework can be easily extended to include multiple PCs to leverage the workload at each workstation. However, our primary interest was to investigate a framework utilizing web services technology to distribute a face-recognition task over more than one PC. DRUBIS has successfully met that goal and this framework can be easily generalised to a larger system.

4.9 Conclusion

This chapter looked at the design and implementation of a distributed framework for face recognition. Our implementation is informed by our unique system requirements. We tried to meet most of these requirements. DRUBIS is implemented iteratively as three versions, each fulfilling different requirements.

DRUBIS evidently demonstrates that, though targeted at a small-scale environment, it can be easily extended to cater for a larger system by distributing the system logic over a greater number of machines. Breaking down the face-recognition web service and spreading it across different machines as smaller web services achieves this, as was discussed above.

Our demonstration applications serve as our proof-of-concept. These are: simple static images on Windows clients; the web-service-driven toolbox; our real-time face-recognition “demo” and lastly our ASP.NET-enhanced DRUBIS with third-party software support.

Lastly, the role of Microsoft .NET XML web services in systems integration via SOAP over HTTP is highlighted. The setback of client/server request/response roundtrip cycles is mentioned. On the other hand, the flexibility, platform independency and the ability to raise computing power through integrating a number of web services are also observed.

Chapter 5

Improved Face Space from Visual Eigenface Information

5.1 Introduction

The quality of the performance of a PCA-driven face-recognition system is to a certain extent determined by the quality of the face space used [Turk and Pentland (1991a)]. A quality face space is a product of good training-set images and these, together with fine gallery and probe images, determine the matching performance. There is, however, no single face space that will cater for all image sets. A number of factors need to be considered when deciding on the training-set images. The significance of visual eigenface information is explored in this chapter.

The image space aims to capture the optimum variance within the training-set images. For effective results this variance must be drawn only from face differences in the training set. The problem is that there is always secondary noise not related to the image content (a face in an image), such as background information, illumination, face position in an image. With all these factors to take into account, the generation of a good quality image space becomes something of a black-magic art. Some image preprocessing before image-space creation is suggested in the literature. It is in fact common today for face-recognition performance surveys to be conducted only after some preprocessing has first been carried out on the images [Qahwaji et al (2002), Howell (1999), Turk and Pentland (1991a), Phillips et al (2000)].

The problem is that it is unclear to what extent each noise factor impacts the overall face-recognition performance. Neither is it obvious how one develops insight into which factors are critical. We aim to solve this problem of what constitutes a quality face space through a study of visual eigenface information.

This chapter uses DRUBIS (discussed in Chapter 4) to assess the effects of noise in the training set. Rhodes University face spaces are constructed and compared to the MIT face spaces. Matching scores from different image sets are presented.

5.2 Sample image sets

The MIT code comes with a corresponding database. The database contains images from 16 individuals with different poses and varying brightness levels. We, however, used only the full-frontal views with varying levels of brightness in our experiments. These images look so similar that it is sometimes hard to tell the difference with the naked eye. The following illustration gives a sample of the MIT images we used in our experiment.

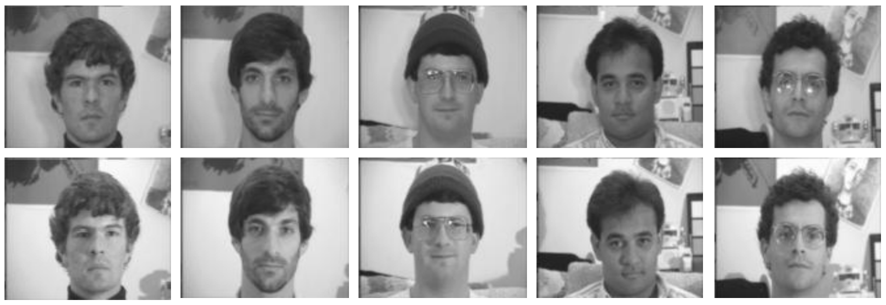


Figure 5a: A sample of the MIT highly-posed images

The top row shows the images from the gallery set with sample probe images in the row below. It is clear that the images were taken under a highly controlled environment. Judging from the relative positions of the background objects in corresponding gallery and probe images, it appears that they were taken directly after one another, with the subjects probably seated in an unmoving chair.

Figure 5a suggests that the MIT images can be accurately recognised by humans solely on background and head shape similarity that is a result of the poses. A manual experiment seemed to be the best way to try to confirm this. If the results are positive, we shall have highlighted an important concern, that face recognition here occurs as a result of the background instead of through, any recognition of the features from the actual face. Face recognition by background is not ideal for fairly obvious reasons: it is the face itself that must be significant. The elimination of as much as possible of the background information is already a feature of many face-recognition systems. It is unlikely that faces can be effectively recognised without first eliminating background since the PCA algorithm has no special adaptation that enables it to discriminate between the two but is solely based on pixel intensity values.

We gave 6 individuals (They were all members of the Computer Science Masters laboratory; this allowed effective monitoring of the experiment.) the task of attempting to match probe and gallery images without the face part so that they could match purely on background information. The results were very positive except for those from one individual, who failed to match even a single image, and we therefore excluded his results. Out of the remaining five individuals, an average of 12.4 over 16 images was achieved – i.e. a 78% successful match score. These results imply that there is a need either for background consistency across all image sets or for the complete elimination of background information. In this study, like many other researchers, we decided to eliminate most of the background information using our face-uprighter software (discussed in depth later).

As part of our study we took our own images from the third-year computer science class at Rhodes University. The images were taken two weeks apart. There were two sets of images: the training set and the gallery set were composed of the same images taken on the first week (a total of 53 images). The probe-set images were taken two weeks later.

There were no controlled factors. This face database closely resembles real-life conditions where minimum control of the environment is possible. Secondly a two-week period is good enough to test an algorithm as a real-life identification problem, which would not be the case had we simply taken both groups of images on the same day and tested the algorithm on these. The algorithm might perform well but the results will not be reflected in its performance outside a laboratory environment. Our gallery and probe images are illustrated below:

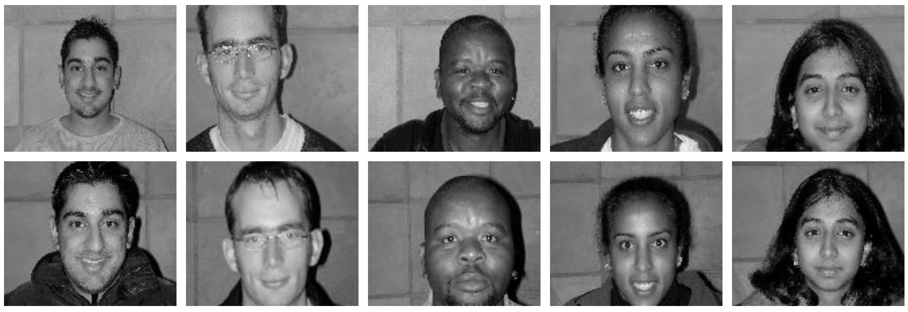


Figure 5b: A sample of the Rhodes University image sets

There is a huge difference between those we took and those chosen from the MIT database. The MIT images look much more similar to each other than do the Rhodes University images. We look at the effect of the image set quality later on.

5.3 Visual face-space information interpretation

The Principal Component Analysis extraction process is mathematical and hard to understand for non-mathematics graduates. On the other hand, research into face recognition is not confined only to the mathematics-enabled community. There is a need for simple and visual models to explain the PCA-extraction process. The visual impact of the eigenfaces as we vary aspects of the space generation can be a simple method of augmenting and improving our understanding of how the PCA approach works.

Because there is no obvious visual correspondence between the eigenfaces and the signature coefficients and data being encoded by them, we have taken an engineering approach to this problem: we process the data and changing variables; evaluate the results and draw our conclusions based on these results. We conducted the following experiments in trying to understand the information encoded by the image signature. The tests look at a broader perspective (through eigenfaces) in trying to understand the information that we can extract from a face space.

5.4 Moving-head position observation: vertical

In trying to understand the meaning of visual information from the eigenfaces, which are ghost images created from a face space, we generate different face spaces using one image. The following figure shows images that were used to create a face space.

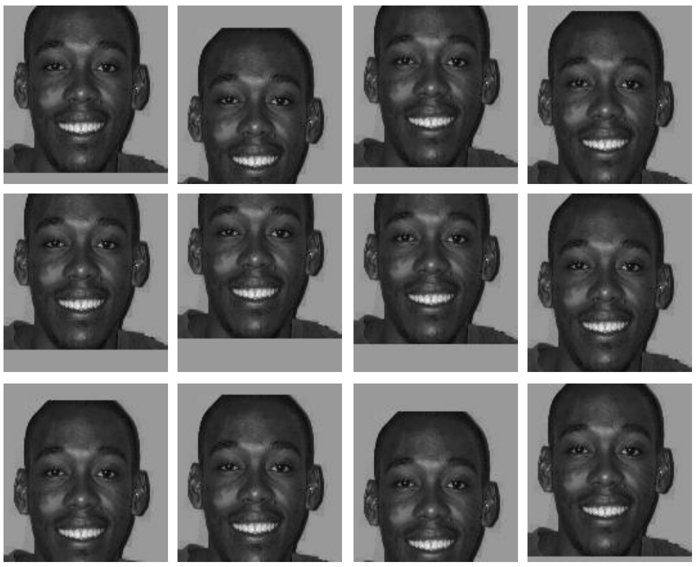


Figure 5c: Moving-head (vertical) training set

The above images are identical apart from their vertical positioning: the image has been moved in several increments of eight pixels up and then down, as may be seen from Figure 5c above. The aim was to see if we could get any visual clues from the eigenfaces about the nature of the training-set images. The eigenfaces produced were as follows:

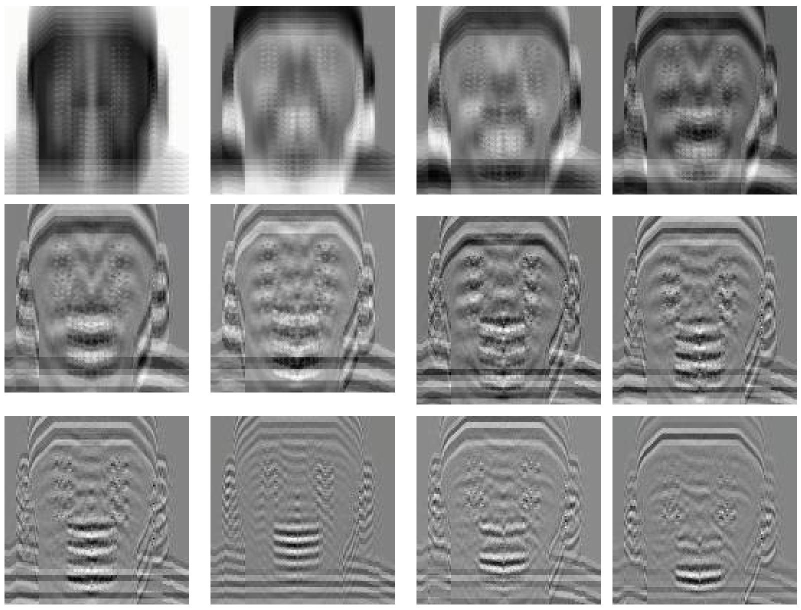
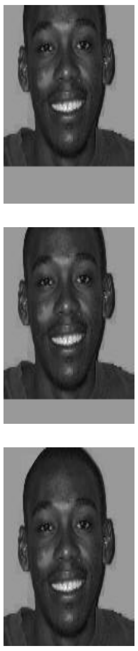


Figure 5d: A moving-head (vertical) face space

Here, the image in the first column of the first row at position (1, 1) is an image-space mean, computed by adding together all the training-set images and then dividing them by the number of images in the training set. Thus the mean image = (Img1 + Img2 + Img3 +…..ImgM)/ M where M is 12 in our case. The rest of the images are eigenfaces sorted by eigenvalues in descending order, according to the variance they represent in the training set. For example, the last eigenface at position (12, 12) accounts for minimal variance as opposed to eigenface at position (2, 12), which accounts for the maximum variance [Turk and Pentland (1991a)].

From the eigenfaces above (figure 5d) we can tell that much of the variance they account for is due to the differences in vertical positioning among the training-set images. The areas that account for the major variance can be clearly observed in the eigenfaces, where more light is visible around the ears and there are apparent changes to features like eyes and mouths (due to vertical displacement of the training-set images). Other noteworthy aspects concern the "energy" in the eigenfaces that falls into obvious vertical bands; the highest ranking eigenfaces clearly have lower-frequency bands. The mean image at position (1, 1) above can be hardly recognised as representing a face. In real life, we would expect the above face space to discriminate a vertical face position in an image very easily; in fact we would expect it to be better at classifying vertical face positions than at identifying faces. This suggests that vertical displacement in an image may have a far greater matching impact than the appearance of the face. It emphasises the point that PCA has no "structural knowledge" – there is nothing that inherently biases the algorithm towards recognising facial features rather than towards recognising vertical displacement.

The next Figure (5e) shows the variation on the image signature that comes from moving the image up and down. The first bold value in a signature represents a DFFS (Difference From Face Space) value. DFFS is discussed in depth in chapters 6 and 7 but it encodes the reconstruction error. The signature values are all rounded down to zero.



4260;2063;-2218;1179;-807;-996;-864;-103;255;-17;-231;-85

A

4029;2561;-1347;163;-43;454;205;-6;285;-231;-154;99

B

4322;-1068;2002;-345;-243;179;-34;-169;-62;-138;80;90

C

Figure 5e: The effect of modifying some image pixels on a signature

Figure 5e shows how displacement in an image can affect the image signature generated. Changes in head position result in major changes to signature co-efficients. The following table shows the distances (closeness) of the above images. The distance scores are computed from an Euclidian distance measure.

	Image A	Image B	Image C
Image A	0	2390	4771
Image B	2390	0	3443
Image C	4771	3443	0

Table 5a: Difference in closeness due to modifying image pixels

The matrix above shows the increase in distance between images as more pixels are modified; the distance remains unchanged for an image against itself. This observation highlights the need to normalise and preprocess images very carefully with respect to position, before projecting them into image space.

5.5 Moving-head position observation: horizontal

Diagram 5f represents another close look at some training-set images also computed from one image but in this case with varying *horizontal* positional properties. In this case the image has been moved left and right in increments of eight pixels:

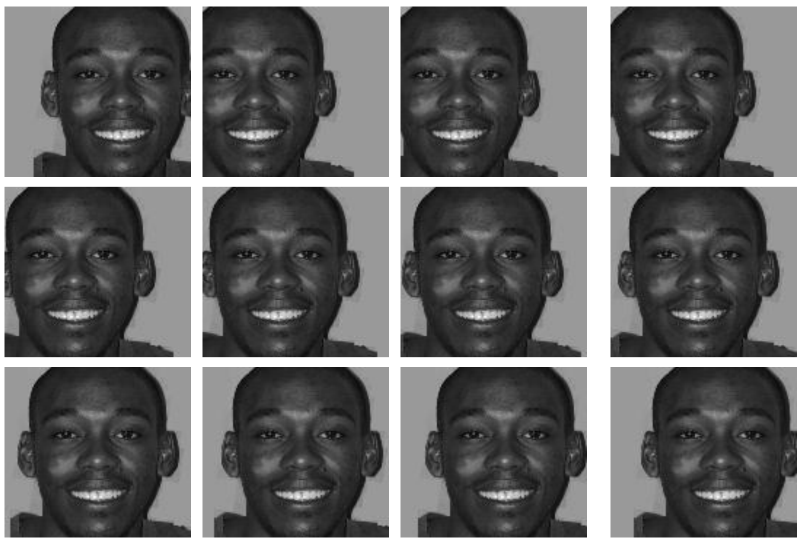


Figure 5f: Moving-head (horizontal) training set

The eigenfaces from this training set are expected to capture the horizontal variance. This can be still observed at prominent areas such as the head position, the ears and the mouth. The following eigenfaces from the face space created confirm that.

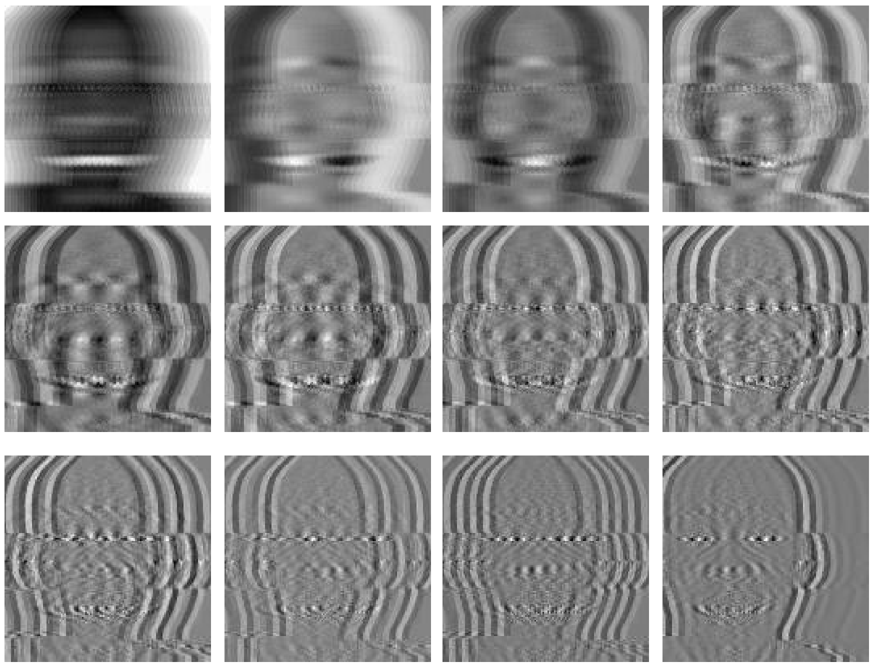


Figure 5g: Moving-head (horizontal) face space

Again the first eigenface captures most of the horizontal head variance, as is confirmed by the light bands. The rest of the eigenfaces still show the variance from a change in the horizontal positioning. This face space can also be expected to increase discrimination or the easy matching of the horizontal head position.

The above two examples attempt to illustrate that one can to a certain extent predict where most variance will lie. From the PCA theory we know that the face space captures and uses this variance for recognition purposes [Turk and Pentland (1991a)]. These experiments convince us that the head position is highly significant in recognising images from this image space.

This observation also suggests that one can look at the face space and tell if it captures the real face identifying factors, or whether it might be just representing horizontal or vertical misalignment.

5.6 Changing blocks of the image

To further understand how the eigenfaces capture the variance, we added arbitrary blocks of "noise" to replications of a single image. In the case illustrated in Figure 5h, the image is fixed at a central position and only the background is varied with a block of dark pixels which, for interest in observing, also covers the mouth in some instances.

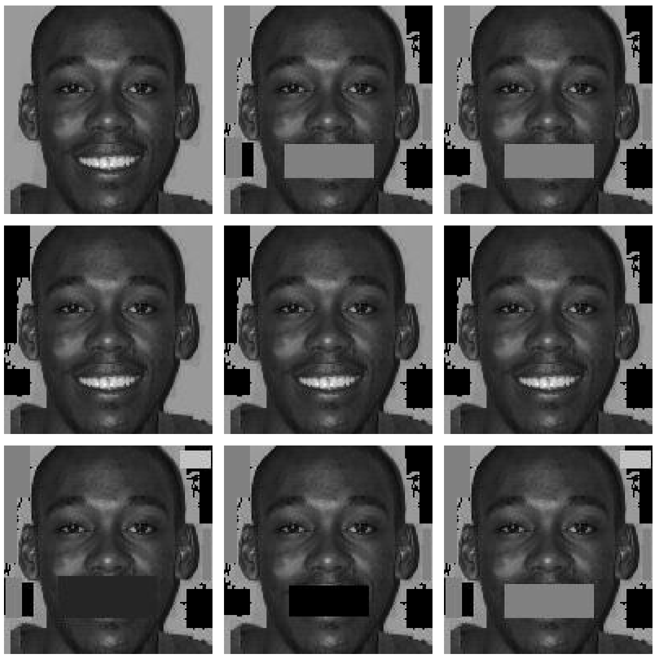


Figure 5h: Changing the background training set

The resulting eigenfaces are expected not to capture variance on ears, eyes, forehead chin and shoulders, where no changes have been made. The background and the mouth account for the major variance in this training set. (Recall that the first image below is the mean image, and that subsequent images show the "energy" in the eigenfaces.)

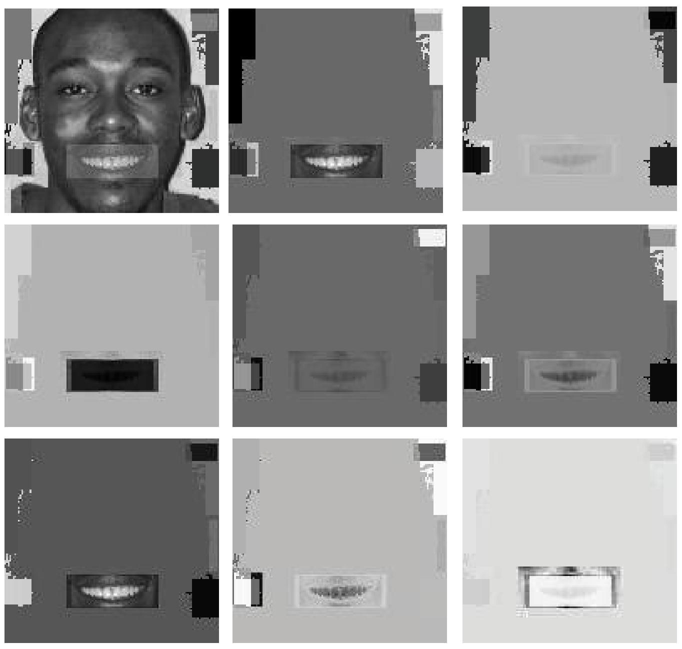


Figure 5i: Changing the background in the face space

As expected, the background and mouth information is the more evident in this face space, which suggests that background and mouth data are more significant in the recognition process for this face space.

Having looked at the above three examples, it is evident that it is critical to eliminate background information as far as possible and also to avoid changing face positions in an image. The distance between the face and the camera might, for instance, cause an apparent difference in face positions. Such discrepancies can be easily controlled during the face-image capturing stages, where posing for the image is done. In the next section we expand on observations drawn from the above face spaces by testing the claims which we have made.

5.7 Image classification using the moving-head-position face space

We have asserted that one can tell from eigenfaces the information that has been encoded in the face space for image discrimination. We have shown three face spaces that we deemed poor and incapable of effectively discriminating images from our training set. The first two face spaces have been deemed capable of better classifying face images through the positioning of the face, than of finding its identity. In this section we experimentally investigate the above assertions.

We used the first face space above to register two more images and we adjusted the images as in Figure 5c above (which shows vertical adjustments in increments of 8 pixels) to get 12 images from each new person. These, together with the 12 images in Figure 5c, made up our gallery set. The probe set was generated using a different set of images from the same individuals, taken two weeks later. Our gallery and probe-set original images were as follows:

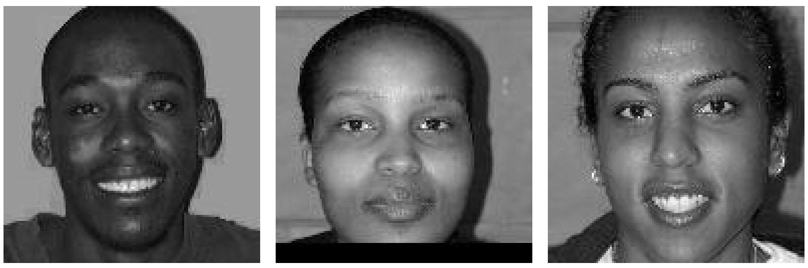


Figure 5j: Gallery set used for testing image classification

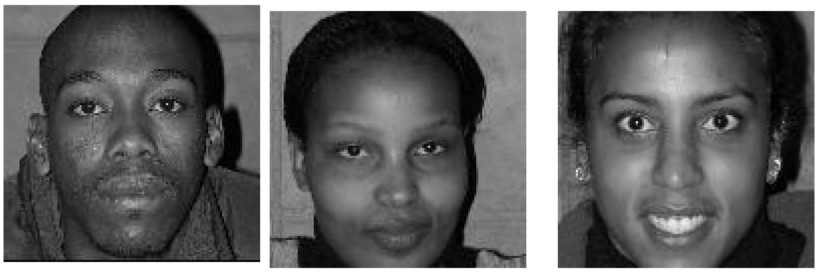


Figure5k: The probe set used for testing image classification

The face-uprighter application that we developed to help with centring and improving pose for a face also straightened the above images. Overall, this gave us 36 gallery images and 36 probe images. Out of the 36 gallery images, 12 were also used for training the face space: those can be fully reconstructed from the face space i.e. they have a Difference From Face Space (DFFS) of zero.

5.7.1 The results from the experiment

The following table of results originated in two tests, one of the ability of the system effectively to identify the owner of the image and, independently, the ability of the system to classify the position of the image. In the latter category an image that is wrongly identified in terms of its owner but correctly identified in terms of its position is recorded as a match. These two tests are aimed at confirming or rejecting the claim that one can predict the most discriminating information that will be captured by a face space – or what information in a given face space is likely to be more discriminating.

(there were 36 probe and gallery set images)	Pos 1	Pos 2	Pos 3	Pos 4	Pos 5	Best Match %	Within Top 5 Match %
Face Matching (individual identified)	11	8	3	1	2	31%	69%
Face Position (displacement identified)	13	10	6	1	1	36%	86%

Table 5b: Face-matching versus face-position classification results

Recall that for each probe we find the closest five matches in the training set. So the numbers in the columns Pos 2 /Pos 3 are the number of times the system found the “correct” image, but placed it as the second or later choice. The above table shows that *displacement* tends to have a stronger effect than facial characteristics. Out of 36 images, 11 were successfully matched to the correct face image and in the right position, leaving only two images that were well classified but matched the wrong people, thus giving a score of 13 for well-classified images at position 1. The rest of the matches follow the same pattern. Interestingly, in Position 5, 2 images matched the right person but one was in the wrong position and that was the only instance where the face matcher outperformed the position classifier. Overall we conclude that the PCA matcher is better at recognising head position than it is at recognising individuals.

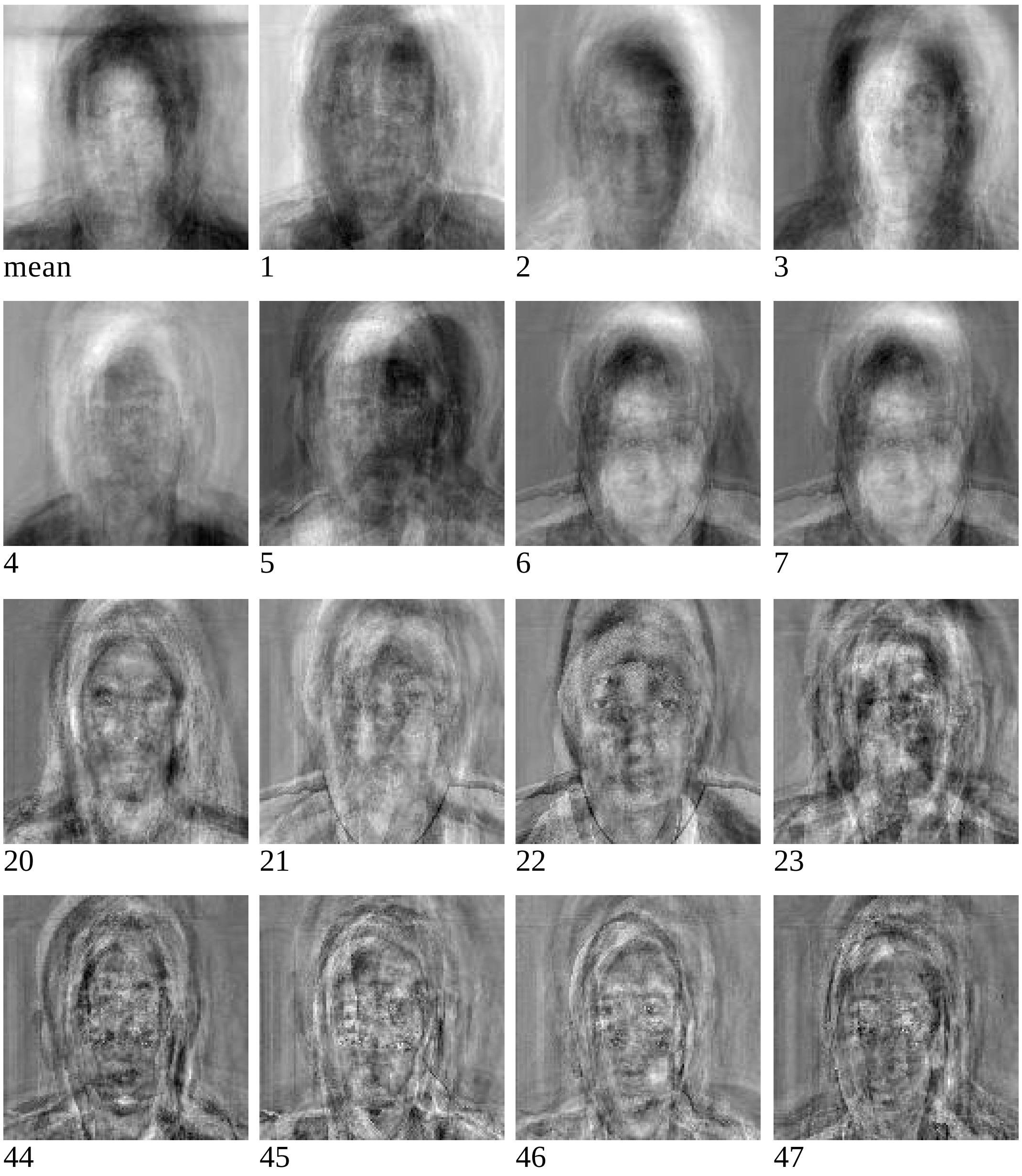
This section has tried to show how one may visually interpret face-space information. It emphasises how the information from eigenfaces can then be used to decide on the quality of a face space. Yambor et al (2000) say that this information can be used to decide which eigenfaces are suitable for use in recognition trials. They claim that it can be easily observed from the eigenfaces which ones are relevant for recognition i.e. the ones capturing the variance necessary to discriminate a face. This claim contradicts the popular approach of using eigenvalues to rank the importance of the eigenfaces. Interestingly, Yambor et al (2000) claim an improved performance after removing the first few eigenfaces, which they claim mainly encode lighting information. We shall examine this later in this chapter.

5.8 Improved face space by normalising head position in an image

In this section we present a face space from our image database and corresponding results from its use. These images are not pre-processed, and part of the evaluation is to investigate the effect of preprocessing. In most cases the face-recognition algorithms perform very badly without preprocessing. Preprocessing can take many forms, including the posing of images, control of the light source and removal of the background.

5.8.1 Rhodes University Face Database eigenfaces

Our image set is very diverse in terms of ethnicity and gender. Before any normalisation of head size, position, or tilt, the eigenfaces were as illustrated below:



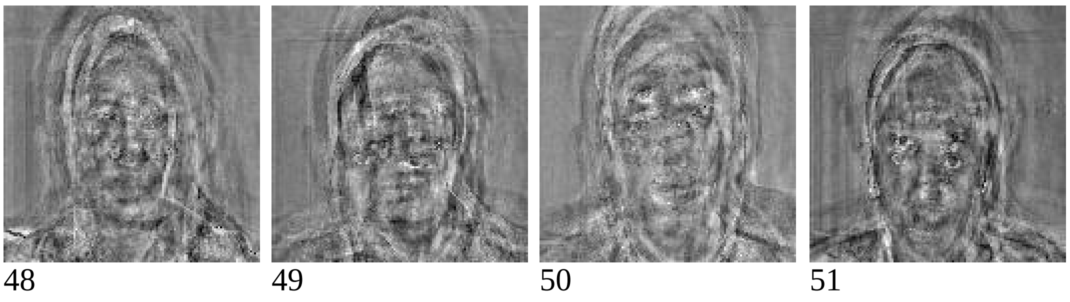
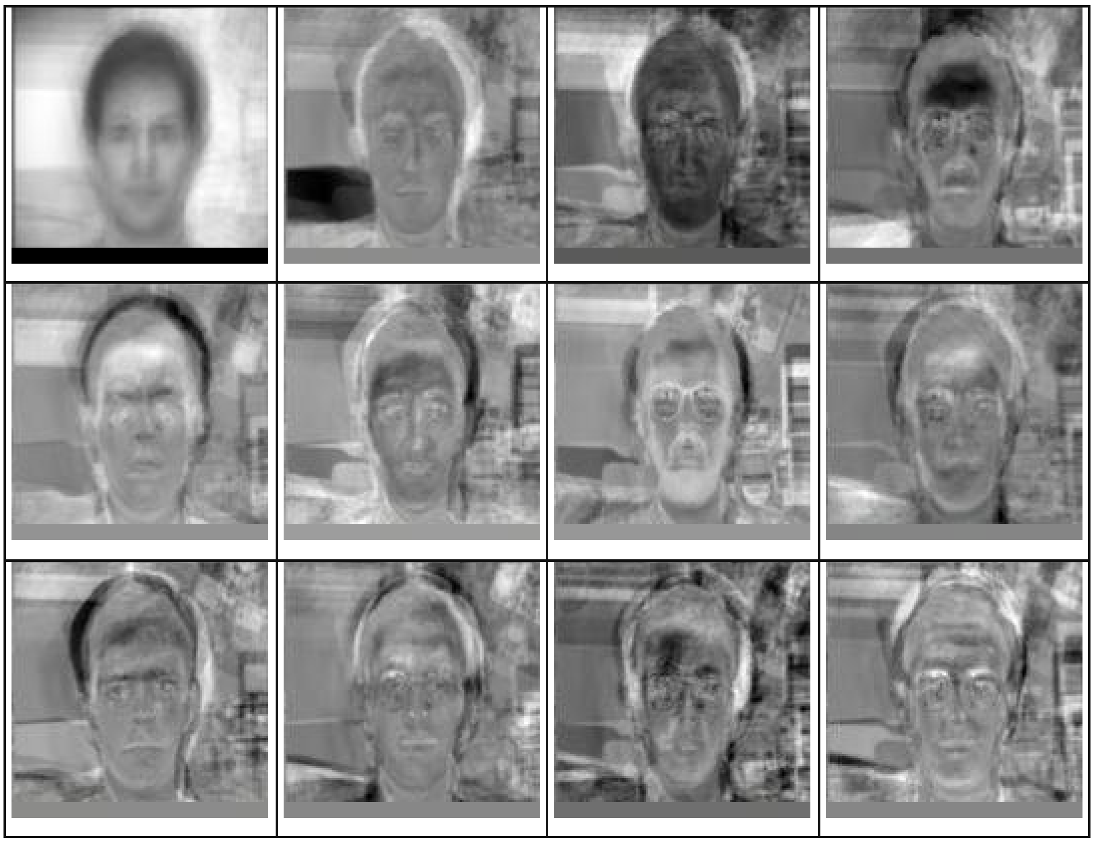


Figure 5l: Rhodes face images in a face space

It is quite clear from these images that the information from the eigenfaces is very fuzzy. The information in the eigenfaces tells us very little about the kinds of information which differentiates subjects. Such a space is of poor quality and less likely to produce quality results.

5.8.2 A good face space – the MIT example

An example of a good face space is the face space from the MIT image database. A quality face space together with high-quality training and probe sets determine the matching performance. The MIT face space produced a 95% matching rate using the MIT image database. This space is represented in the following eigenimages:



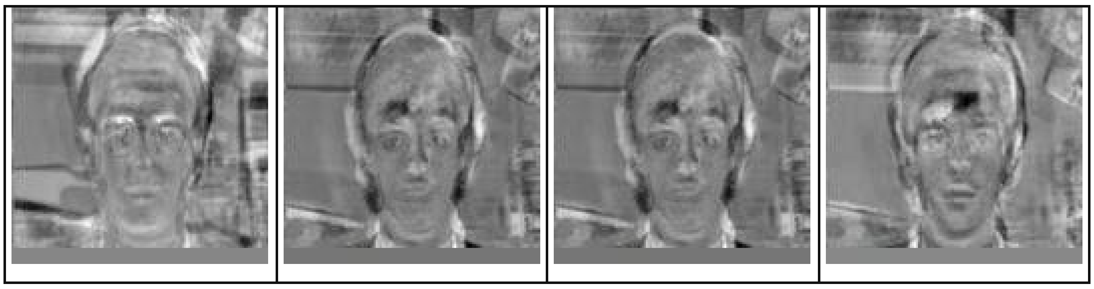


Figure 5m: MIT face images in a face space

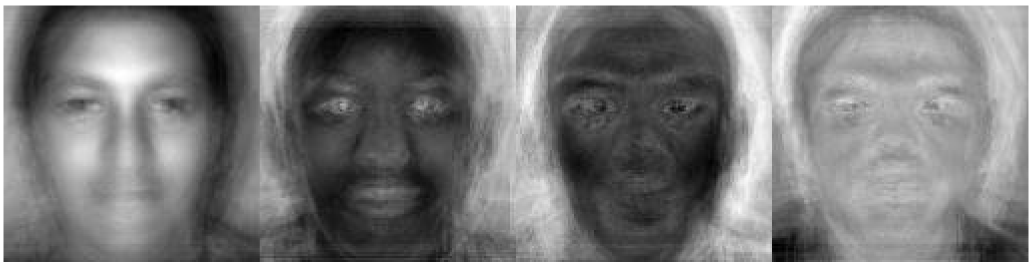
A quality face space is not composed only of good-looking training-set images. Quality can only be judged in relation to the gallery and probe-sets images used. This face space has good quality in relation to the MIT gallery and probe sets.

There is a huge difference between the MIT face space and our face space. The difference originates from a number of factors such as posing, controlled lighting and a highly controlled environment. Our training images were not created to be this artificially ideal. In most real-life applications, such a highly controlled environment is hardly ever obtained and almost impossible to achieve in live face-recognition applications.

5.8.3 Face preprocessing through the face-uprighter

In our attempt to improve our face space we designed an application (the face-uprighter) that scales and normalises the face using only the eye positions in an image. This is achieved by manual intervention – in our case, dragging a mouse from one eye to the other. The new image is then produced with the eyes at the centre and the process re-scales the images and crops them to 128 by 128 pixels. The face uprighter proved to be very successful in normalising the eye position and gave an almost fully consistent head position across the image set.

The new face space generated after normalising the head position through the face uprighter in our training set looked as follows:



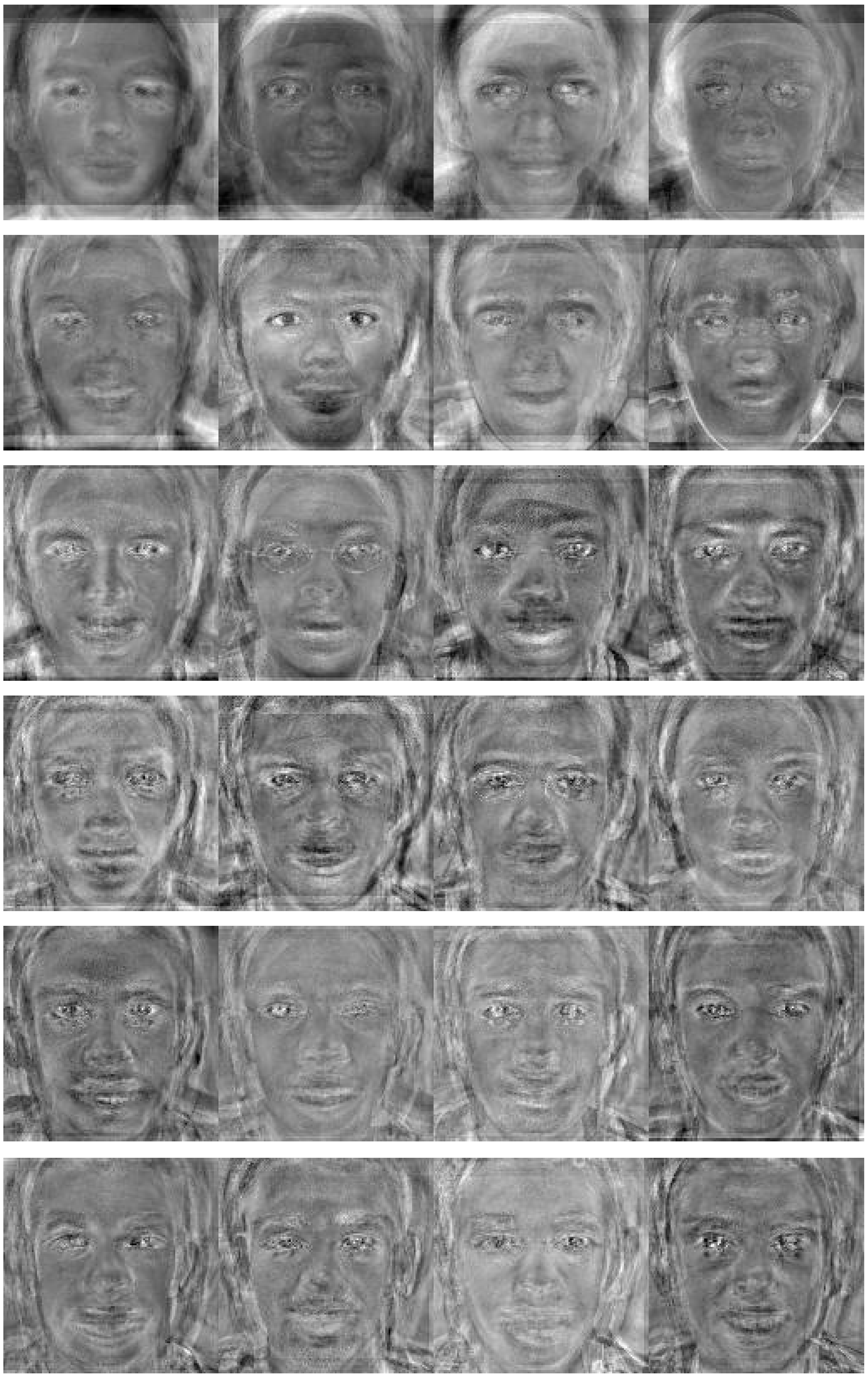


Figure 5n: Scaled Rhodes face images in a face space

The above eigenfaces exhibit an improvement over the face-space originals even without normalised head positions. The eyes are in the same position across all eigenfaces because it was these that were used to normalise the head position in an image. One problem that may be encountered is that one of the features that could be a strong discriminating factor for humans - the distance between the eyes – has been normalised by the preprocessing.

5.8.4 Matching scores from preprocessed images

Based on our earlier observations, one would expect improved matching rates from the latter face space as compared to the first one. Our next step is therefore to compute matching results using our ad hoc image set and compare these with the results generated from the face space with the normalised head positions. We normalised the heads in both the gallery and probe sets so that they were consistent.

The matching scores are computed using the squared Euclidean distance as used by the MIT media Lab open-source code. An in-depth analysis of the performance of different distance measures is covered in the next chapter.

The following graph gives the matching results from the face space using an ad hoc image set with no head-position normalisation, as opposed to the face space with the normalised head position. The training set images are the same images as the gallery set images. The results are categorised by the different number of eigenfaces (the signature size) starting from a face space with 10 eigenfaces up to a face space with 52 eigenfaces.

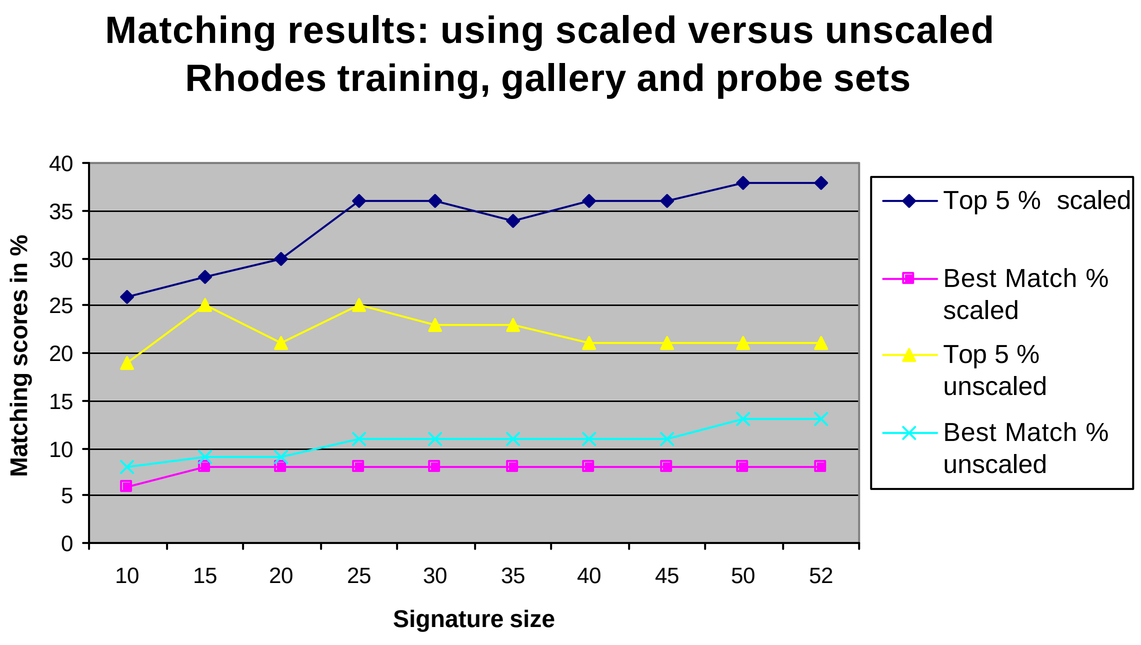


Figure 5o: Preprocessed vs. unprocessed images using the Euclidian distance measure

Above, figure 5o clearly illustrates an improvement over the top 5 matching results. On the other hand it failed completely to improve results on first-position matching. These were unexpected and somewhat disturbing results. They imply that scaling has made it much harder to distinguish some images beyond a reasonable doubt. One possible explanation is that the normalisation of the eye positions has resulted in a loss of information about absolute face size. A person with smaller features or eyes closer together might be recognizable precisely because of a characteristic that we might have discarded. In the absence of knowing absolute measurements, we do not have an answer to this issue, and it could well be a fruitful source of future research. The results also raise the question of the influence of the gallery set size. For example, it is easier to distinguish 10 images than it is to distinguish more than 50. The following section considers this factor.

5.9 Using cropped face images

The use of only partial face images (cropped face images) has been studied by a number of researchers [Haddadnia et al (2002), Lin (2000), Jiang (1996)]. There is no consistency concerning which parts of the face should be cropped. Some crop the whole face (excluding the hair) while others crop e.g. the eyes only. In our experiment with cropped images we cropped the face without a mouth. We chose this because the positional variation of the mouth is great: it can appear very differently when talking, laughing, or smiling, and it may also appear open or closed.

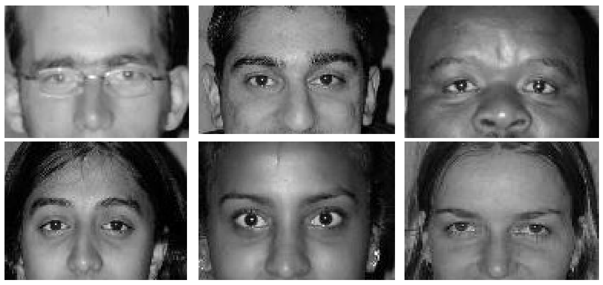


Figure 5p: Sample of cropped Rhodes University images

We computed some matching scores using different distance measures and compared these with the un-cropped, scaled Rhodes University images. We used 26 eigenfaces for each case; the gallery and probe sets are made up of 53 images and each image in the probe set has a corresponding image in the gallery set. The cropped images are of size 64 by 64 pixels, which is half of the original uncropped image size (128 by 128 pixels). The images were also preprocessed through the face-uprighter. We used a top 5 rank match. The following table shows the matching scores that were achieved against the different distance measures:

	L1 norm	Euclidean (L2 norm)	Angle	Mahalanobis	L1 + Mahalanobis	L2 + Mahalanobis	Angle + Mahalanobis
Cropped-frontal	32%	26%	38%	43%	38%	28%	36%
Full frontal	30%	35%	38%	42%	21%	26%	45%

Table 5c: Matching scores using cropped, frontal face images against uncropped images

Table 5c shows that different distance measures gave a wide variety of different performance scores for the cropped images, except for the Angle distance measure which returned the same 38% for both image sets. Euclidean and Angle + Mahalanobis gave improved performance (32% and 45%) when full frontal images were used. The rest (L1, Mahalanobis, L1 + Mahalanobis, L2 + Mahalanobis) of the distance measures showed improved performance with cropped frontal images. This left us with no clear trend as to whether cropping the mouth improved results. We hope that future study will examine this in depth and that our experimentation framework may be able to facilitate such research.

5.10 Sorting eigenvectors according to eigenvalues, versus using visual eigenface information

Selecting the most significant eigenfaces on the basis of eigenvalue information has become a de facto standard in PCA face-recognition algorithms. Yambor et al (2000) explore the use of eigenvalue information together with some visual eigenface information. They claim an ability to eliminate eigenfaces that encode irrelevant information. This approach is shared by Moon and Phillips (1998), who claim that the low-order eigenfaces encode gross differences among the training set. They conclude that performance might be improved by excluding low-order eigenfaces from the presentation. Moon and Phillips (1998) performed their experiments using four different image sets from the FERET database:

- **duplicate I** (images taken days apart),
- **duplicate II** (taken from months up to years apart),

- **FB** (probe images taken from the same session, i.e. on the same day and at the same time) and
- **fc** (images taken using a different camera and different lighting, but during the same session).

The following results were reported in their study:

Number of low order eigenfaces removed	**Duplicate I**	**Duplicate 2**	**FB probe**	**fc probe**
0(baseline)	0.35	0.13	0.77	0.26
1	0.35	0.15	0.75	0.38
2	0.34	0.14	0.74	0.36
3	031	0.14	0.72	0.37
4	0.20	0.09	0.50	0.22

Table 5d: Moon and Phillips's (1998) performance scores with low-order eigenvectors removed. Performance scores – success rates using the first matcher choice only.

Above, a higher score gives a better recognition rate: 0.77 indicate a 77% recognition rate using the top n matches (n is one in this experiment). The above table only shows a slight improvement in **fc** probe scores when low-order eigenfaces are removed (the most significant eigenface seems to account for the lowest frequency changes in the image, which in this case seems to be the lighting conditions.)

We repeated similar experiments on our data. We computed our performance scores using the top 5 matches with none, the first, and then the first three eigenfaces removed. We used the Euclidian distance measure to compute our matching scores. Our performance scores are shown in the following table:

Number of low-order eigenfaces removed	**Rhodes University Face Database**
0(baseline)	0.35
1	0.19
3	0.13

Table 5e: Rhodes University Face Database performance scores using Euclidian distance.

Our experiment performance scores were unable to support the claims by Yambor et al (2000) and Moon and Phillips (1998). We found the recognition rate to decrease each time a lowest-order eigenface is removed from the face space. This is also evidenced by Moon and Phillips's (1998) performance scores listed in the table above. Eliminating the first eigenface failed to cause any significant change. The rapid decline in matching performance scores when the low-order eigenfaces were removed from face space is enough to conclude that low-order eigenfaces are indeed significant for matching. We therefore failed to replicate the findings of Yambor et al (2000). Our conclusion is that we ought to retain the orthodox practice of using the eigenvalue ordering to select the eigenfaces.

5.11 Conclusion

In conclusion, this chapter has looked at ways of building an intuitive understanding of the information captured by a face space. This was achieved by varying the background and face position in an image and monitoring any resulting changes in the face space. From these observations it has become apparent that visual eigenface information is significant in determining face-space quality. This quality appears only in relation to the quality of the corresponding gallery and probe sets. Quality face space on its own is not enough to guarantee good performance.

The observations from this chapter led to the development of our face-preprocessing tool "face-uprighter" which scales, rotates and normalise eye positions in a face image. The Rhodes face space was improved by normalising eye positions and scaling individual images. Using this, we achieved slightly improved recognition rates, but we were unsure about its overall validity as a tool, since it loses information about absolute face sizes.

Lastly, we looked at eliminating the first eigenfaces to improve matching performance. This action is suggested in the literature but is not standard. Our findings yielded poor results. Our evidence is that the standard eigenvalue-ordering selection method for eigenfaces is better, and we opted for this approach in the rest of this project.

Chapter 6

Primary Considerations in PCA Face-Recognition Applications

6.1 Introduction

Chapter 2 looked at key factors to consider when choosing a face-recognition algorithm. In this project we opted for a Principal Component Analysis (PCA) baseline algorithm. This chapter evaluates different performance aspects of PCA algorithms. Some of the work we cover has already been done and published by various research groups. Our interest in this work is to verify that our framework supports such work, to deepen our appreciation of PCA, and to compare their findings against ours.

There are a number of experimental tests that can add value to our investigation of the performance of the PCA algorithm. Some of these tests are aimed at assessing the capabilities of PCA in handling a diverse population like ours in South Africa. Others are aimed at introducing some guidelines on how to configure signatures and face spaces for optimum performance. We suspect that the diversity inherent in our population is very poorly captured by most face databases in that they are not fully representative of female, African or Asian communities, and we wish to investigate the impact of this diversity on face recognition. We further investigate the possibility of using the rich variety of facial information types at our disposal as a means of improving performance.

There are a number of possible methodologies that can be used to assess the performance of a face-recognition algorithm. In this project we follow some of the steps used in the FERET (FacE REcognition Technology) evaluation methodology.

6.2 Evaluation methodology

Our images in this trial were captured over a period of three weeks. We took approximately 70 images per week, using a third-year Computer Science class as our subjects. We matched week-to-week images by hand and we discarded both some that we could match by hand, and those from people who attended only one photography session. In the end we were left with 53 images to use in our experiment. These were sufficiently diverse and comprised 31 males, 22 females, 7 Asians, 24 Africans, 22 Caucasians.

6.2.1 FERET evaluation protocol

6.2.1.1 Identification versus verification

We attempted to follow the FERET evaluation methodology as closely as possible in our experiments. This methodology and the FERET face database have become the de facto standards in the evaluation of face-recognition algorithms. The FERET methodology makes a very clear distinction between identification and verification [Phillips et al (2000)].

Identification addresses the problem of identifying a face against the top *n* matches. A “success” is recorded if the system returns a correct match within its top *n* candidates. The whole gallery set is searched and a similarity measure is computed for all gallery images; only the top *n* images with the highest similarity measure are returned as a match. In the following experiments we look only at the identification aspect.

Identification differs from verification, in that it involves a “claimed identity” by the person to be identified by the system; the system’s role here is to accept or reject the person’s claim. In verification *two* images are involved. A similarity measure is computed upon which is based the decision either to accept or reject the claim.

6.2.1.2 Target vs. query images

FERET protocol test images are made up of target and query images. Target images are images known to the system and query images are images to be identified. A sub-group of target images make a virtual gallery and a sub group of query images make a virtual probe set. The virtual gallery and the probe-set technique allow us to characterise the algorithm's performance using subsets of target and query images [Phillips et al (2000)]. FERET computes the similarity measure between the gallery and probe sets. The similarity measure aids in grouping images into gallery and probe sets by giving an idea of the similarity scores between target and query images. Images falling within a particular similarity-score range can then be grouped together because they share the same level of recognition difficulty. The similarity measure subsequently becomes a measure of how difficult it is to obtain good scores on given query and target sets.

The different categories may include duplicates taken within a week of the gallery image, galleries containing one image per person, galleries containing duplicate images of the same person. A gallery of *n* people can be created and used for probing, to give an estimate of an algorithm's performance. Our Image Handler web service encapsulates similar functionality. A single collection of images is stored in our database, but these can be organised into a number of different subset galleries or probe sets based on gender, race, or other criteria.

6.2.1.3 The closed-universe model versus the open-universe model

We use a "closed universe" in evaluating the algorithm's performance against our images. In a closed universe, every probe image is in the gallery set. In an "open universe" some probes are not in the gallery set. These two models are different and different performance results are reported from the FERET tests [Phillips et al (2000)].

An open-universe problem is a bit harder to solve than a closed-universe problem. A specific similarity-score benchmark must be set so that scores falling below the benchmark are regarded as indicating that the person may be not found in the gallery; if the score is higher than the benchmark, a person is taken not to be in the gallery. A good example for an open-universe model would be a police system where criminals are already known, a bit like having e.g. a target set of 100 known criminals or terrorists, and using it for comparison to a probe set of one million people coming into an airport. The open-universe model with small target sets and big probe sets has useful applications, but they have issues of their own, and we have not pursued this subject further here.

6.3 PCA-algorithm evaluation experiments

This section covers some evaluation experiments and primarily aims to highlight the critical factors to watch in implementing a PCA face-recognition algorithm. Some of the experiments overlap and each experiment addresses a specific performance question in the PCA-based face-recognition system. For example, similarity scores are influenced by the dimensions of the face-space but there are also a number of factors affecting the way training-set images are chosen which may skew results. We investigate one factor at a time with an aim of highlighting the influence of different factors on performance scores.

6.3.1 What is a good face-space size or face-signature size?

The PCA algorithm is based on the generation of a face space which is later used in recognition. Fundamental to PCA is the computation of the covariance matrix. It is not clear how much variance needs to be captured in order to get reasonable results: i.e. if we use 30 images for training, how many eigenfaces must be used in a signature – 5 or 10 or 15 or more? The goal is to find a rule of thumb such as 20% or 85%.

Computing how much variance each eigenvector accounts for is not a hard problem, although this issue appears to be overlooked in the literature. The real question concerns the ideal eigenface signature size for a given user population e.g.: “for a system to be used by 5000 people is 15 eigenfaces a good signature size or should the signature size vary with the anticipated user population?” Such findings might imply different or specialised designs for bigger target populations.

The findings from this section are crucial as there is a trade-off between an increase in the number of eigenfaces and the computation time taken for matching. Bigger signatures slow down the system. On the other hand, if bigger signatures improve the recognition rate and therefore the overall effectiveness of the system, the trade-off can be justified. It is hoped that the findings will help establish guidelines on choosing a signature size.

The face-space quality used in matching the performance of PCA-driven algorithms is one of the critical factors. Moon and Phillips (1998) and Yambor et al (2000) investigate the influence of face-space size, i.e. the number of components in a face signature. They report interesting findings. Yambor et al (2000) conducted their experiment on Mahalanobis, City-Block (L1), Euclidean (L2) and Angle distance measures. They used the top 20 eigenfaces from 215 possible components (roughly 9% of the eigenfaces) against the top 127 eigenfaces (60%) over ten trials. The Mahalanobis distance measure outperformed the rest when 60% of the eigenfaces were used. There was no significant difference in matching scores between using 20 or 60 components for any of the other measures.

6.3.1.1 Matching scores against varying face-space size

In our experiments we varied the face-space size from 10 eigenfaces (19% of the total) to 52 eigenfaces (100% of the total). We compared matching scores from the different distance measures. This experiment is treated as independent of the experiment performed by Yambor et al (2000) using FERET face database. We use the entire 53 gallery-set images against 53 probe-set images using a "within the top 5 candidates" success metric.

Signature Size	L1	L2	Angle	Mahalanobis	L1+ Mahalanobis	L2+ Mahalanobis	Angle+ Mahalanobis
10(19%)	32%	26%	32%	38%	19%	28%	34%
20(38%)	26%	30%	36%	40%	21%	28%	40%
30(58%)	28%	36%	42%	43%	21%	30%	47%
40(77%)	36%	30%	38%	47%	25%	34%	42%
50(96%)	34%	38%	40%	45%	30%	32%	45%
52(100%)	32%	38%	40%	45%	32%	32%	45%

Table 6a: Matching scores from varying face-space sizes

Table 6c shows matching scores from different distance measures when the face-space size is varied. The Mahalanobis distance measure did well in most of the categories. It was outperformed only by the Angle+ Mahalanobis distance measure when the top 30 eigenfaces were used. The following graph gives a comparison of the performance of the different matching scores from the above table:

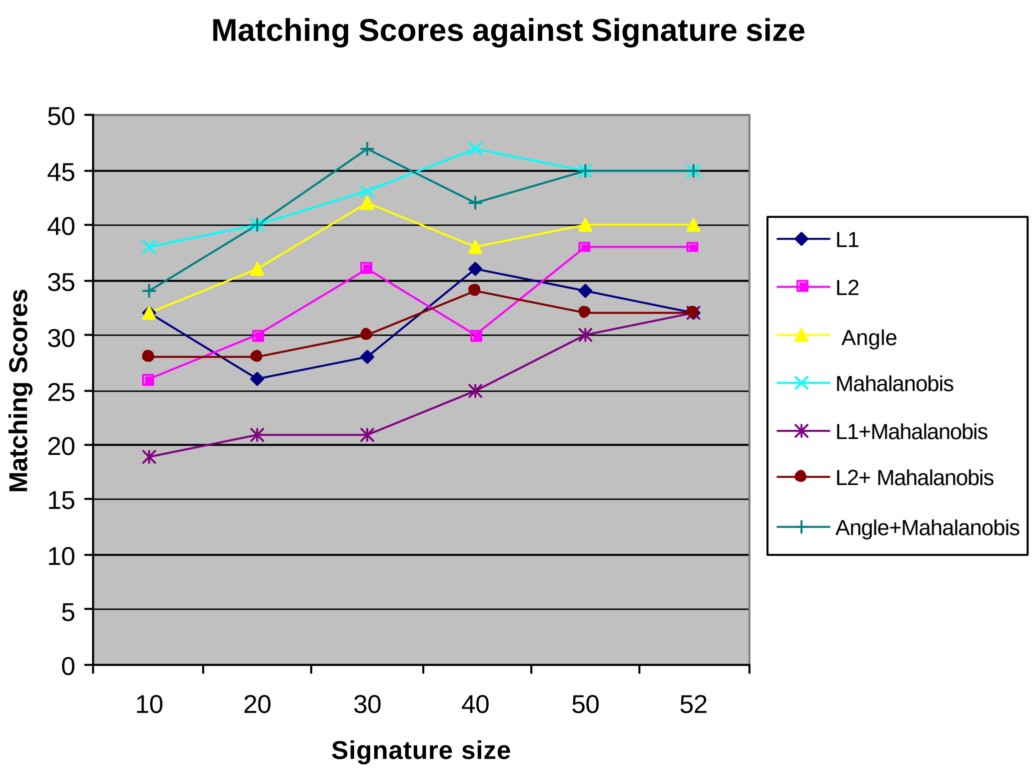


Figure 6a: Matching scores against face-signature sizes

The first observation is that the Angle + Mahalanobis, Angle and L2 (Euclidian distance) give their best performances when the most significant 30 eigenfaces (58% of the total eigenfaces) are used. The Mahalanobis, L1 (City-Block) and L2 + Mahalanobis achieve their best when 40 eigenfaces (77% of the total eigenfaces) are used. The L1 + Mahalanobis gave the worse performance in general.

The disparity in matching scores may be attributed to the way different distance measures handle noisy data. The last eigenfaces in the face space contain information that is less critical for recognition. These eigenfaces contribute mostly noise to the signature information. The Mahalanobis distance measure compensates for this by weighting eigenfaces with decreasing weights based on the eigenvalues when computing matches. This can be observed by comparing the way other distance measures dropped their performance when more eigenfaces were added i.e. when noisy data is added. Mahalanobis dropped only from 47% to 45% after its best score. It is not surprising that Mahalanobis is one of the overall best performers in our experiment.

The Angle distance measure also performed well. It outperformed the L1 and L2 distance measures i.e. where the eigenfaces were not weighted. The Angle + Mahalanobis distance measures shared the honours with the Mahalanobis for a best score of 47% when 30 eigenfaces were used.

The L1 + Mahalanobis and L2 + Mahalanobis failed to outperform L1 and L2 respectively. This implies that the weighting from eigenvalues simply added noise, rather than compensating for it in these distance measures. This is different from using the Angle + Mahalanobis measures combined, where performance improved when they were merged.

Lastly, 6 of 7 distance measures dropped their performance when all eigenfaces were used. Only the L1 + Mahalanobis report improved performance when all eigenfaces were used. This suggests that using all the eigenfaces does more harm than good.

In summary, though our study was based on a limited population size, this did not prevent a number of issues coming to the fore. Using all the eigenfaces does not improve the overall matching performance of a distance measure. For our data we obtained best results using between 50% and 80% of the total eigenfaces from a face space, depending on the distance measure used. Incorporating variance information to L1 and L2 distance measures failed to improve performance but instead decreased it. We anticipate that other experiments with different noise characteristics in the data may show different results.

6.3.1.2 Visual effects of images projected onto face space with some eigenfaces discarded

The above section has looked at the effects of varying image-space size on matching scores. In this section we look at projecting an image onto face space with some eigenfaces discarded. We do this in an attempt to answer the question: "How much variance must be kept to be able effectively to reconstruct an image that is in the training set?" The literature suggests that if all eigenfaces from the face space are used, the projected image can be fully reconstructed. In this section we hope to improve an understanding of the effects of discarding some eigenfaces, and give a reader a visual understanding of how images are compared when generating matches.

We generated a face space of thirteen eigenfaces i.e. from 14 training-set images. We then projected one of the image-set images onto the face space with varying (1, 3, 5, 7, 9, 11, 13) eigenfaces. We expected to see the projected images improve as more eigenfaces were used and eventually to get almost exactly the same image when all 13 eigenfaces were used. The following diagram shows our results.

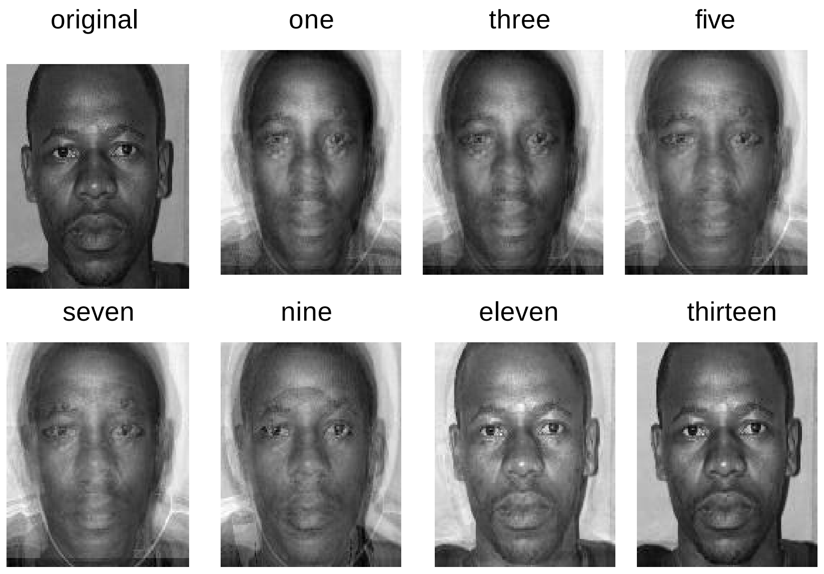


Figure 6b: The influence of independent training-set images in face-space generation

Figure 6b shows the original image projected onto face spaces with varying eigenfaces, together with the resulting reconstructed images. The number of eigenfaces used in each projection is labelled above each reconstructed image. The images above include a mean image which is added to the reconstructed images. It may clearly be seen from this that images reconstructed from face spaces with fewer than nine eigenfaces (69%) yielded very poor images compared to the rest. The reconstructed image with all its eigenfaces (thirteen), gave the best reconstructed image, as expected.

In summary this section has visually explored the effects of projecting a training set image onto a face space. The image signatures that are used in matching are generated from the reconstructed images. It is clear that, if few eigenfaces are used, the resulting images are very poor and can hardly be matched by a human matcher. This understanding is critical in deciding on an optimum face-space size.

6.4 Optimum face space

In this section we look at two different scenarios. The first one looks at the effects of having identical, as opposed to different, training-set and gallery-set images. The second scenario investigates the way a face-space signature is generated: does it matter, for instance, if you use all your training images from your training set or use only *n* +1 to get *n* eigenfaces for the signature?

6.4.1 Using the same training and gallery sets vs. using sets independent of each other

The big difference between using the same, as opposed to independent, gallery and training-set images is that an image used in training can be fully reconstructed from the face space: if it is projected onto the face space, it occupies a certain point within the space (with a DFFS of zero). The opposite applies for a face that was not in the training set: it can still be reconstructed, but not nearly as accurately. This section investigates the effects of such a difference on matching scores: the questions are, firstly, "what is the significance of using the same images for face-space generation and for enrolment?", and secondly, "does a face space with gallery-set images yielding a zero DFFS gives better matching scores?" We hope to understand better the significance of using a face space that is independent of the training set.

There are three groups of images involved:

- The training set: *this is the set of images used for face-space generation.*
- The enrolled images or the gallery set: *these are the registered images that are used to match against the probe image* ***and these can be the same images as the training set.***
- The probe set: these *are images presented to the system for recognition or classification.*

There are three possible outcomes from this section:

- If the recognition rates are better with identical training- and gallery-set images, this would imply that PCA is more suited to batch-processing applications (e.g. in police applications or in access control, where people are all known to the system prior to any computer-aided recognition process). This might force the regeneration of the face space each time new subjects are registered on the system.

- On the other hand, if recognition rates improve when an independent training set is used, this would suggest that we should avoid reusing the same images for training and enrolment, since an independent training set captures variance better.

- If there is no difference in matching performance scores, we shall know that there is no significant advantage in either coupling or decoupling the training and enrolment images.

In summary, this section compares the performance of PCA in Incremental Enrolment Mode **(IEM),** where the face space is independent of the subjects enrolled, against the Batch Enrolment Mode **(BEM)** where the subjects are all known from prior face-space generations.

In our experiment to evaluate the effects of using Incremental Enrolment Mode as opposed to Batch Enrolment Mode, we used the face spaces trained from the MIT and the RU images respectively. We enrolled RU and MIT images against MIT and RU face spaces, resulting in four groups of experiments namely:

- RU training against MIT enrolment,
- MIT training against RU enrolment,
- RU training against RU enrolment,
- MIT training against MIT enrolment.

The RU face database is made up of 53 images and the MIT face database is made up of 16 images. To normalise this database size difference, we randomly selected 15 images from the RU face database. These images were divided into three groups of 15 images where each image belongs to a single group. We computed matching scores for each group and kept an average for each category. We computed matching scores from 15 images from the MIT face database. In our experiments each gallery image has one corresponding probe image and both MIT and RU face spaces use the top ten eigenfaces (each signature has 10 components). We counted successes only if the system identified the correct individual within its top candidates (the top 5).

The following matching scores were recorded.

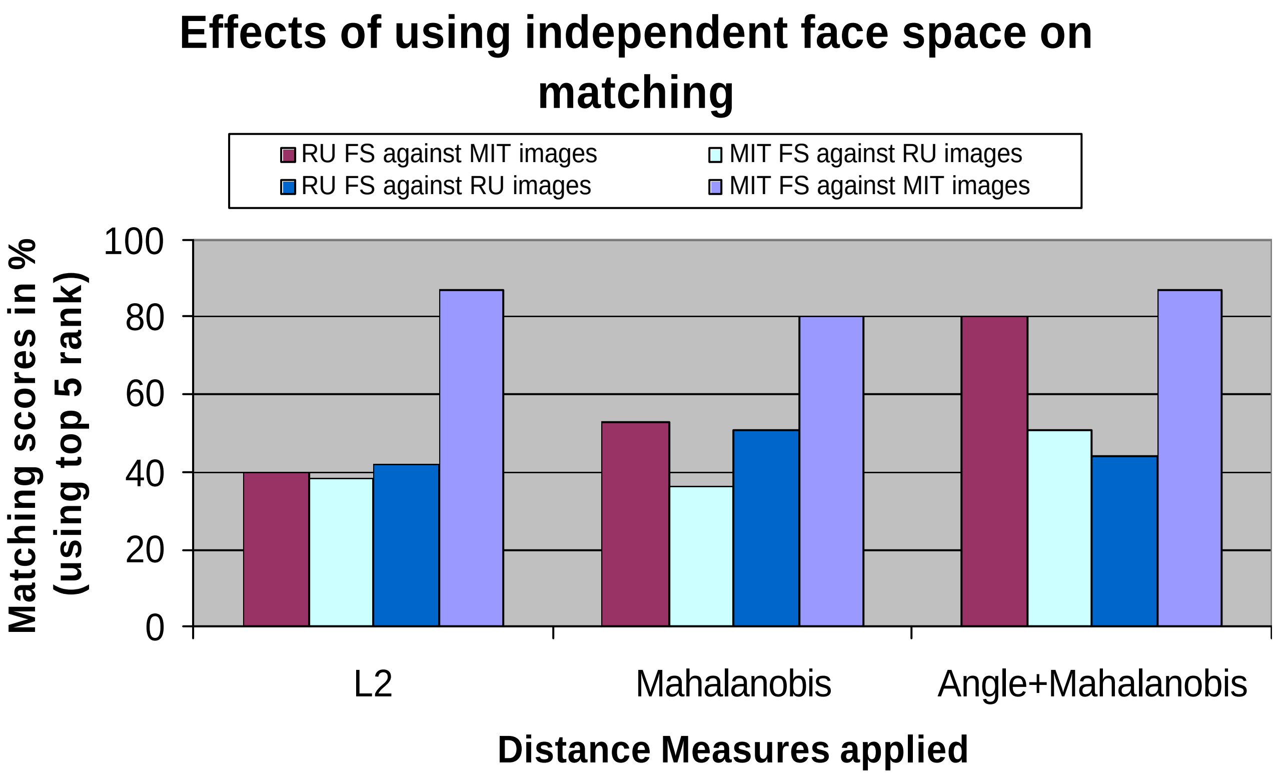


Figure 6c: The influence of independent training-set images in face-space generation

The above graph shows an interesting trend. The MIT images against the MIT face space achieved an outstanding performance across all distance measures, where the worst score was 68% and the best 87%. This is in contrast to the performance of all other experiments. The second best performer was the MIT images using RU face space. There was a slight improvement (42 % and 51% using L2 and Mahalanobis distance measures) in matching scores when the RU images and RU face space were used against that of using RU images with the MIT face space.

Considering the best-performing metric, Angle + Mahalanobis, it seems that the underlying training set made very little difference: the MIT images return around 80% matching, whether or not they were enrolled and recognised on either the MIT-trained space, or the RU-trained space. Similarly, the RU image-recognition rates show around 45% success, more or less independently of which training set was used. What the experiment does suggest is that the MIT enrolments and probes are very alike and easy to recognise, whereas the RU enrolments and probes are inherently less controlled and more difficult to recognise. This is evident from a visual inspection in figures 5a and 5b, and is supported by our work in section 5.2.

In summary, though a marginally improved matching performance was recorded when using dependent training-set images, the effects of using good quality gallery and probe sets far supersedes good training-set images. This finding might not generally apply if all image sets are taken under highly controlled environments. We need a more sophisticated image normalisation process, prior to using the images in a PCA-driven face-recognition application, before we can generally conclude on the effects of using independent training-set images. This leaves us with the assertion that a PCA-based system can give a slight improvement in recognition scores when operating in Batch Enrolment Mode as opposed to operating in Incremental Enrolment Mode (using highly normalised images). Quality gallery and probe-set images will always give better matching scores independently of the training set, but bad gallery and probe-set images will always yield poor matching results,.

6.4.2 How to train if you want to retain only *n* eigenfaces?

Above, we investigated the coupling of a training set with matching performance scores. A closely related issue is how to get an optimum representation of a face space. Suppose we want to retain only 20 eigenfaces, does it matter if we train with 100 images and take only the 20 top eigenfaces, or can we simply use 21 images to generate a 20-eigenface facespace?

Findings from this section will assist us in understanding how best to capture variance in face space. Through this understanding a rule of thumb may be produced on how best to generate a training set for optimum results.

In this experiment face spaces of size 10, 20 up to 50 eigenfaces were used. The difference of adding ten extra eigenfaces was monitored from 10 to 50. Angle + Mahalanobis was the only distance measure used in this experiment (since it gave best results in Figure 6c above). The following matching scores were recorded:

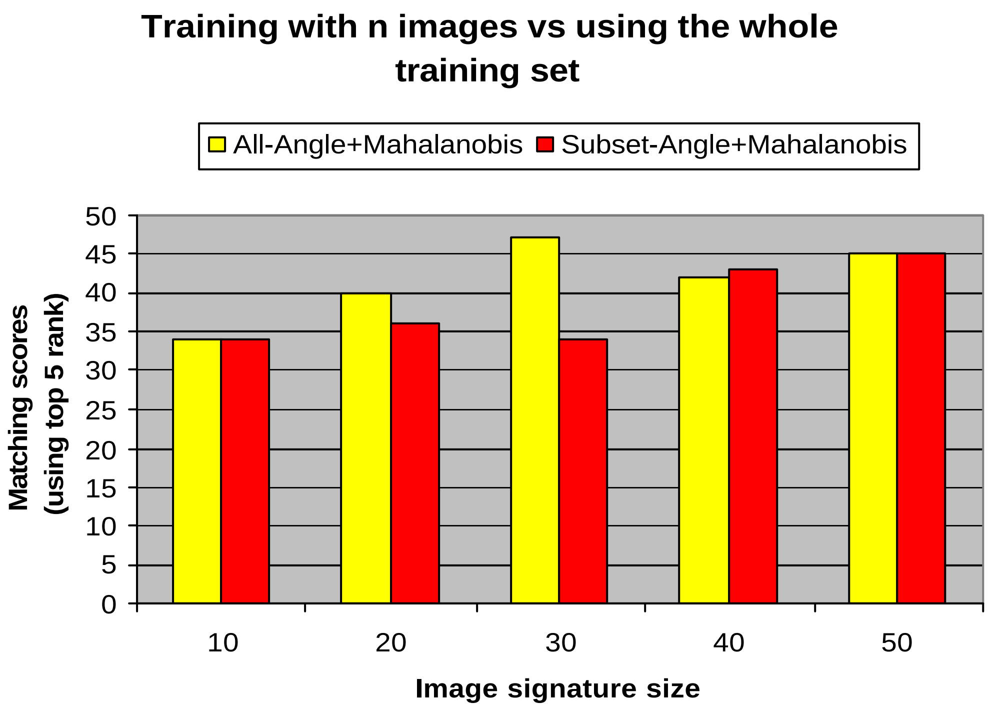


Figure 6d: Matching scores using a subset of the training images versus using all training-set images

The above graph shows some interesting observations. First, there was no difference in matching scores when too many or too few training-set images were used. When a few training-set images are used (10 in our case), there is very little variance captured and not enough for proper discrimination. This results in poorer matching performance. The eigenface selection method had no effect on matching scores (at least not with our images). Note, too, that for those experiments where we used only some subset of images for training, the questions can validly be asked "which subset, and does this matter?" Our subsets were chosen randomly and we did not investigate how much the right-hand bars would vary if experiments were repeated with other subsets.

The best matching performance for both eigenface selection methods was achieved when the most eigenfaces from the training set were used (the top 50). This was to be expected, since our training set is made up of 53 images: there is very little difference between taking the top 50 eigenfaces from 52 eigenfaces against training with 51 images and keeping 50 eigenfaces!

A good rule of thumb is that, if fewer eigenfaces are needed, it is better to train with the whole training set and keep only the top *n* eigenfaces. The Angle + Mahalanobis measure performs better if the signature size is not very small and if all the training-set images are used, so that there enough variance is available. In figure 6c above, training with the whole training set shows a slight improvement in matching performance scores.

6.5 Conclusion

This chapter outlined a number of experiments that our framework has facilitated. These have all improved our understanding of PCA. We have examined important methods for determining the quality of images to be used in a face-recognition system. Determining image quality (i.e. intra versus inter difference) was covered in depth. This gave high-level guidelines concerning the overall performance of the system against specific image sets.

The problem of an optimum face-space size and an optimum face space (how best to choose eigenfaces) has been addressed in some detail. A rough guideline is that too few eigenfaces compromises performance, and using too many eigenfaces lowers system performance and can add noise to the face space.

Chapter 7

Group Classification Using DFFS

7.1 Introduction

The need for of a computer system that can classify images by gender, race and age, or can distinguish whether the image is even a face, is unquestionable. Information from such system can help enhance current Human-Computer Interaction (HCI) technologies. It can serve as a basis for passive surveillance and control in "smart buildings", as a means for restricting access to certain parts of a building and can also collect valuable demographics such as the number of women entering a retail store everyday or the number of Africans or Whites observed to be buying a specific product [Moghaddam and Yang (2002)].

Additionally, a successful gender or race-classification approach can boost the performance of many other applications including face identification, smart human-computer interfaces, etc [Sun et al (2002)]. Gender or race information may help, for example, with the decision about which face space to use for matching when using a PCA algorithm. We investigate this possibility in the current chapter. Our goal is to establish whether the use of multiple face spaces improves performance.

There are a number of face-image classification algorithms. In this work we have primarily focused on PCA-based face-recognition algorithms, using traditional distance measures like Euclidean distance or City-Block for matching individuals. As a natural extension to this work, we explore aspects of PCA-based work that show promise for group classification. The distance measure we use, DFFS (Difference From Face Space), again comes in more than one flavour, and can be computed using Euclidean distance or City-block similarity measures.

It is worth revisiting our initial premise: there are useful applications of identification, verification, and classification technology even though we might not achieve 100% accuracy. This chapter now addresses the classification problem in the context of relaxed accuracy requirements.

7.2 Difference From Face Space (DFFS)

DFFS is defined as the difference between the original image and the reconstructed image after it has been projected onto a face space. DFFS represents the "reconstruction error" or "residual description error" which is left unexplained by our signature representation. The following figure illustrates this concept.

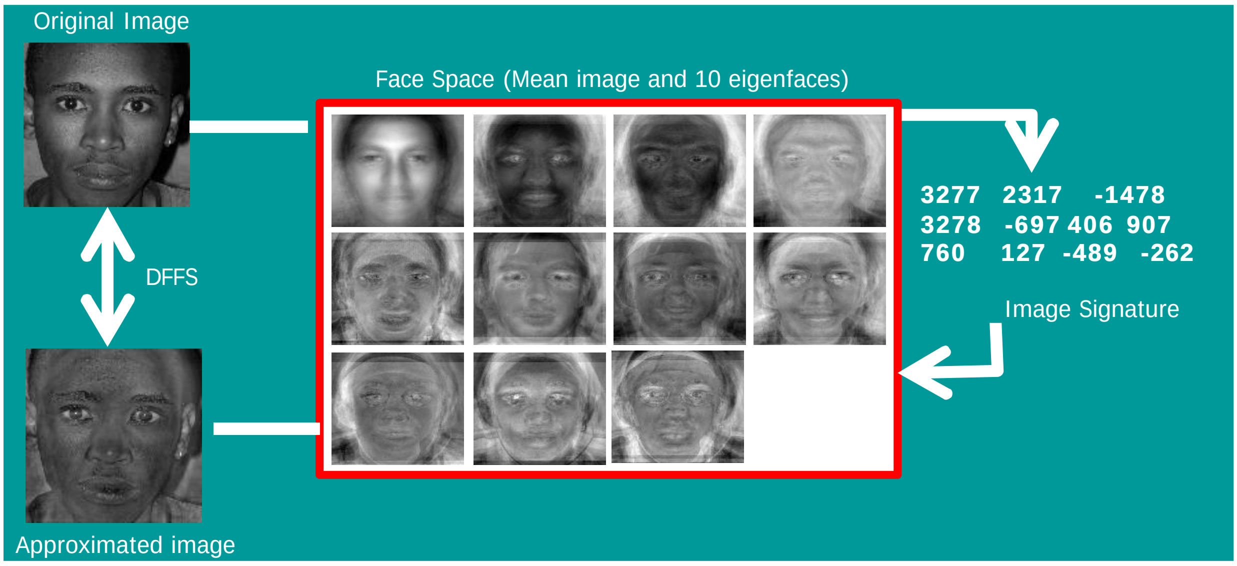


Figure 7a: An illustration of the DFFS generation process

As discussed in earlier chapters, a face image is projected onto a face space to produce an image feature, or simply a unique signature representing an image. Figure 7a illustrates this, where a face space of ten eigenfaces is used to produce a signature made up of ten coefficients. During reconstruction, the signature multiplied by the eigenfaces (from the face space), plus the face-space mean, produces an approximation of the original image. The closeness of this approximated image to the original image will depend on the quality of a face space, whether the projected image is part of the training set or not, and the number of eigenfaces that were discarded as being "insignificant" for the purposes of producing good approximations. If all of the eigenfaces are kept, an image that is used during training should project exactly onto face space and will give a zero DFFS value. (In practice, small computational variances might occur.) But our interest is not in working with images that were known during the training phase, but in working with new images that are subsequently presented.

7.2.1 The DFFS value

The DFFS value is the similarity measure between the original image projected onto face space and the approximate image generated using a signature plus the face space. In Figure 7a above, the DFFS value is the similarity measure of closeness between the two images outside the face space. This similarity measure can be computed using distance measures such as Euclidean distance and City-Block. This measure computes pixel-by-pixel Euclidian differences. The DFFS value computed using Euclidian distance from the above two (original and approximated) images is **3112.596**. The DFFS can also be visually exposed as an image, a DFFS image.

7.2.2 The DFFS image

A DFFS image is a face image resulting from pixel differences when an approximated image is subtracted from the original image projected onto face space. The following figure shows three DFFS images, one computed from an original image that is part of the training set and the other two with independent images.

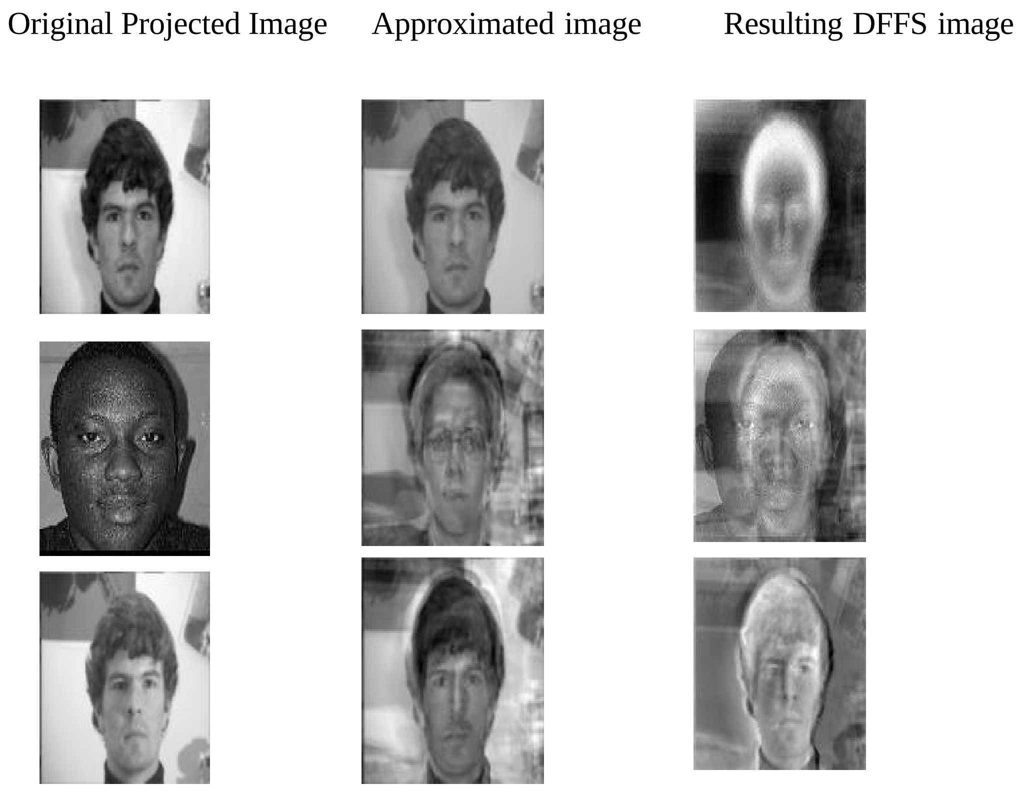


Figure 7b: DFFS images

The original image in the first row is part of the training set and the ones in the second and third rows are not. The DFFS image is computed by subtracting each pixel value in the corresponding approximated image from the original image. The new image is then reconstructed from the resulting pixel values.

In Figure 7b, the image in row one has been better reconstructed than the ones in rows two and three. A DFFS value of **284.76** is computed for the images in row one. The DFFS image shown has no mean face added to it. This difference is a result of the original image in row one being in the training set. The original image in row two is badly approximated, with a DFFS value of **9221.51**. Firstly the image is not in the training set and secondly it does not resemble the training images. The image in the third row is of the same person as in row one. The difference is that this image is not in the training set and has different illumination levels. The images in row three were taken in the same environment as the ones in row one, but with different lighting conditions. As with the image in row two, the approximated image in row three is not as good as its counterpart in row one, and there is a considerable difference in the DFFS images. The DFFS value for this image is **4519.64**. We can see from this that an image that has little correspondence with the training set can be easily spotted using either the DFFS value or visually through DFFS images.

An original image that is from quite another class of objects e.g. a car or a plane, will generate a huge DFFS value. The approximated image will also be very far from the original image projected onto face space since it is from a different class of objects and can nor be effectively represented by face-space. Since our interpretation of DFFS is the residual unexplained variance after projection into face space, a small DFFS value implies a close "fit" in the signature space that approximates it [Turk and Pentland (1991b)]. Based on this, the DFFS value can be used to classify images as belonging to one group or the other i.e. faces or non-faces. It is less obvious that DFFS would be able to make finer distinctions between, say, male or female, or images with faces oriented in one direction rather than in another. We investigate the feasibility of using DFFS to classify our images and the impact this will have on the general matching performance of the PCA algorithm.

Since our primary focus in this work was our framework, we will illustrate some of the algorithms and the flavour of our tools in this section by providing concrete code. We had the following web-service module to generate approximated images and DFFS images. The module was programmed in C# running on a Microsoft .NET platform.

```
[WebMethod (Description="Returns original, the reconstructed, and the
difference.")]
//web method with a description of the module
            public byte [] [] get3Images(string spaceName, string picName)
            //space and path of the image to be projected in space
               {
                  byte [] [] result = new byte [3][];
                  //array to hold the three images

                  //return the original image as a feature image
                  fImage orig = new fImage(fImage.imagePath+picName);
                  result[0] = orig.toJpegByteStream(true);

                  //load face space and generate a feature signature
                  ImageSpace ims = theCache.getSpace(spaceName);
                  Feature f = new Feature(orig, ims);
                  //reconstruct the fimage from the feature signature
                  fImage reconstruct = f.reconstructImage(ims);
                  result[1] = reconstruct.toJpegByteStream(true);


                  //subtract the reconstructed image from the original image
                  orig.mix(-1, reconstruct);
                  //add the mean to the new difference image
                  //orig.mix(1,ims.theMean);
                  result [2] = orig.toJpegByteStream(true);

                  // return the 3 images as a byte array
                  return result;
            }
```

The above module takes an image, creates a feature image, and loads it into an array. The second step projects the original image onto the face space by generating its feature signature and reconstructs this image to get an approximation of the original image; this image is loaded at the second position in an array. Thirdly, the reconstructed image is subtracted from the original image. This is done through the **`orig.mix(-1, reconstruct)`**subroutine which takes two images and adds corresponding pixels. The implementation of the **`mix(float,fimage)`** subroutine is:

```
public void mix(float alpha, fImage image)
{
      System.Diagnostics.Debug.Assert(image.imgSize == imgSize);
      for (int i = 0; i < imgSize; i++)
      {     //processing the pixels of the 2 images
            data[i] += alpha*image.data[i];
      }
}
```

The above subroutine simply multiplies all pixel intensity values and adds them together. An assert construct validates that the two images involved are of the same size. The original image then replaces this processed image.

7.2.3 DFFS values weighed against matching scores

As explained above, DFFS represents the "reconstruction error" or "residual description error" which is left unexplained by our signature representation of an image. It is not clear if probe images that have a small DFFS are likely to match well. When probe images yield a small DFFS, we can infer that they are well represented by a linear combination of the current eigenfaces. Since our training set and gallery set are composed of the same images, all enrolled images have a DFFS close to zero. This section attempts to answer the question, "Do probe images that have good representation in the space, i.e. small DFFS values, give more accurate matching results?"

The results from this section are critical in the sense that they will improve our understanding of the usefulness of DFFS as an additional measure of confidence when we match. If good matching scores correlate with small DFFS values, it means that it is very critical for probe images to be accurately correlated with training and gallery-set images. DFFS values can be precomputed before matching, because confidence intervals (the chances of the system picking a correct match) can be known before the actual matching. If there is no difference in matching scores with small or large DFFS, we can infer that matching scores have little to do with how well a probe image is reconstructed in a face space. This will mean that there no reason to compute DFFS values from probe images before matching.

For each probe image we return the exact matching position from the whole gallery set. Matches range from position 0 for the best match to 52 for the worst matching position. We use 53 gallery and probe-set images and the combined Angle + Mahalanobis distance measure is used to compute matching results. The following figure shows our results: points close to the X axis, (e.g. Y=5 or less), mean that the matcher identified the correct gallery image in its best 5 candidates. DFFS values close to the Y axis imply a good representation in face space. The query we need to make of the graph is whether the points have an obvious correlation or regression line.

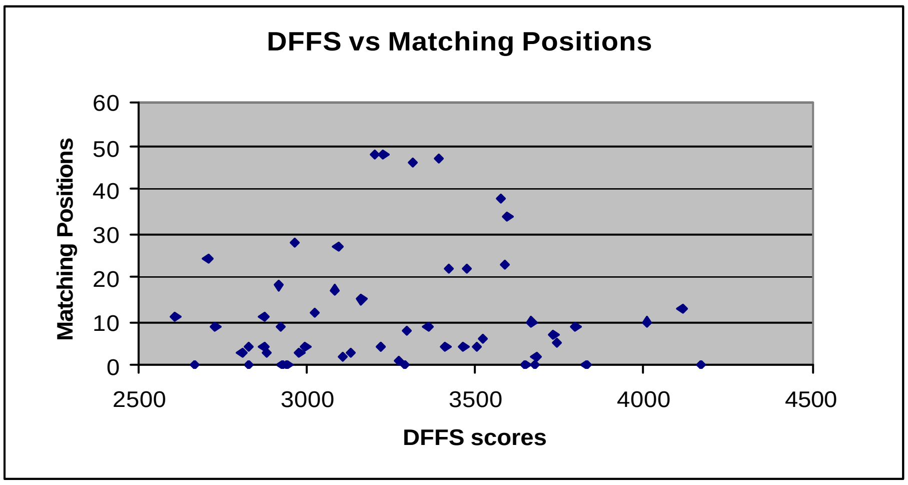


Figure 7c: DFFS values versus matching positions

The above matching scores show no obvious correlation between the matching scores and the DFFS values. This suggests that knowing DFFS values before matching may not be necessary. Next we look at the 4 images that gave the best matches.

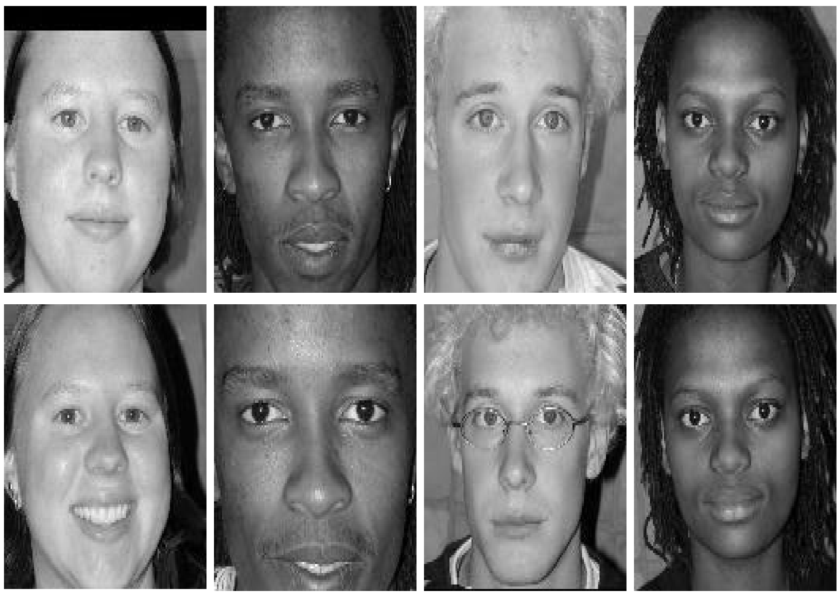


Figure 7d: Best matched images

The above images yielded the best matches in that the system correctly located each gallery image as its first choice. Though the images are very different from each other, the corresponding gallery and probe images in each case are remarkably alike, especially with respect to lighting aspects. This is one of the characteristics that the MIT images posses: they are highly posed and well scaled. The gallery images are in the first row and the probes in the second row. It is worth adding that the above images had probe images with DFFS values ranging from 2942 to 4169.

The next figure shows our worst matched images. Look carefully at the posing and the scale.

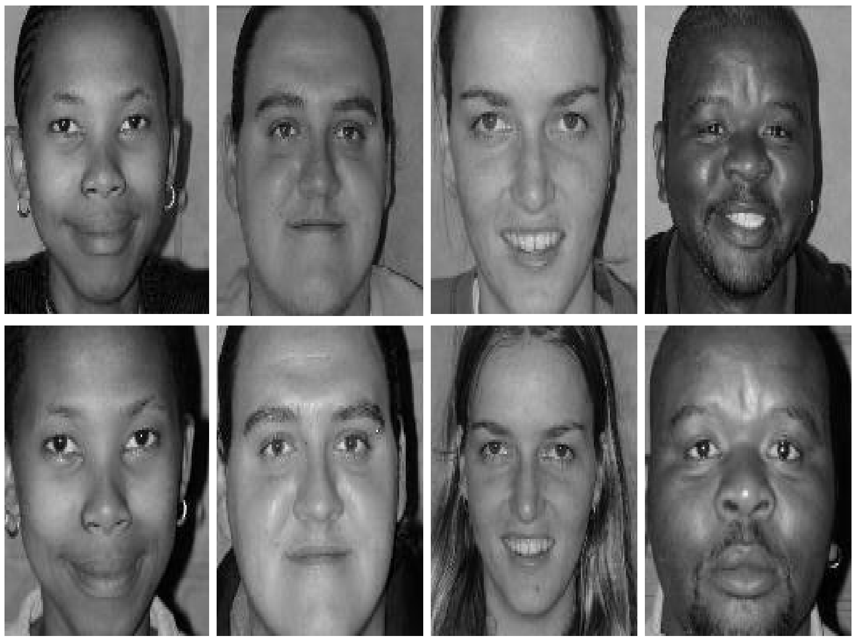


Figure 7e: Worst matched images

The above image pairs have different scales and lighting. Some images have background information that their corresponding match does not share. The proportion of the image taken up by the face seems to be a major difference between the image pairs. In general this experiment reveals that scaling, face position in an image, and illumination levels are very crucial in matching. This helps to explain why our face images performed so badly compared to the highly posed and scaled MIT images. It is, therefore, scaling that needs special attention for better matching performance, and not the probe DFFS va lues.

7.3 Related DFFS classification work

Studies have suggested that the PCA algorithm is capable of classifying images. The classification is done by determining the decision boundary between e.g. the male and female class (or the Africans and Asians class) from the training-set images. In these studies the DFFS (Difference From Face Space) value [Turk and Pentland (1991a), Pentland et al (1994)] and the eigenface information [Abdi et al (1995)] are used for image classification in that DFFS is used to determine if the image falls within or outside a specific class.

The PCA information has been used, with varying degrees of success, in automatic face recognition to classify images according to orientation, or to detect eigenfeatures like eyes and nose [Pentland et al (1994)], or to differentiate face images from non-face images [Turk (2001) and Turk and Pentland(1991b)], and even to classify images by gender [Sun et al (2002), Abdi et al (1995), O'Toole et al (1998) and Moghaddam et al (2002)].

Sun et al (2002) use the PCA algorithm to represent each image as a feature vector (a signature made up of projection coefficients) in a low-dimensional space. Genetic Algorithms (GAs) are then employed to select a subset of features (eigenfaces) from the low-dimensional representation by disregarding certain eigenfaces that do not seem to encode important gender information. The image classification is performed using any one of a number of techniques: a Bayesian classifier, a Neural Network (NN), Support Vector Machines (SVM), or a classifier based on Linear Discriminant Analysis. The best gender image classification score reported in their work is 91.1% from a Support Vector Machine (SVM) classifier. Moghaddam et al (2002) also confirm the superiority of the SVMs by reporting a 96.6% correct gender image classification rate (i.e. only a 3.4% error rate) from their experiments.

Support Vector Machines are learning machines that can perform binary classification (pattern recognition) and real valued-function approximation (regression estimation) tasks. The main idea is that, given a set of data points which belong to either of two classes (male or female signatures in our case), an SVM finds a hyperplane that leaves the largest possible fraction of points of the same class on the same side and maximises the distance of either class from the hyperplane.

Abdi et al (1995) suggest a need for an optimised face space for image classification. Their approach is similar to that of Sun et al (2002), where a Genetic Algorithm is used to pick those eigenfaces that encode significant gender specific information from a face space. Abdi et al (1995) optimise their face space by computing the eigen-decomposition of a set of faces. Their process is similar to PCA, but is based on neural networks. Unlike PCA, in which all images contribute equally in the construction of face space, and matching weightings are based on eigenvalues, the neural network approach learns weightings so that some training faces become more important than others in their contribution to the eventual recognition. They report results ranging from 85% (with new images) to 100% (with training-set images), using this approach with different image sets.

Pentland et al (1994), in their view-based and modular eigenspaces, considered faces from different views: frontal, left profile, right profile, etc. Their approach is to create separate face spaces for each category of view or each part of the face (eyes, nose, and mouth) and to use the DFFS information to determine which space to use before trying to identify the individual within that space. They report an average of 94% correct feature detection (a false alarm rate of 6%) using a signature size of 10 elements.

The success of the above implementation from Pentland et al (1994) attracted our interest in investigating the use of DFFS information to classify images by sex or demographics and simply to establish whether a given image fits better with cluster X rather than with cluster Y.

7.4 The DFFS research question

The DFFS value tells how well a projected image is represented in the face space. An image that closely resembles the training-set images, or one that is accurately expressible in terms of a linear sum of the base eigenface values, gives a small DFFS, and an image from a different class (very different from the training-set images or face-space images) gives a huge DFFS. This suggests that DFFS information can be used to classify images where images with a small DFFS closely resemble the training set and that a huge DFFS implies an image from an unknown class. A good test of this will be to generate a face space using female images and project both male and female images onto this space. The expected result is that female images will yield smaller DFFS values than male images. Section 7.6 below attempts to explore the use of the DFFS value.

7.5 The link between DFFS scores and matching scores

A DFFS value tells how well the image is represented by the face space. This does not necessarily imply that a face space that best captures variance amongst the training-set images will give better recognition results. The problem of matching entails computing a similarity score between an approximated probe image and the approximated gallery images in the face space. In short, matching is about how best the gallery images and probe images are distributed on the face space. An ideal setup will have images belonging to the same individual making a cluster (close together in space). The following diagram attempts to clarify this concept by giving a picture of the different approximated images after projection.

The flat surface represents the face space and every face-space signature (the image signature generated against the space) is represented by a point on the surface. Gallery and probe images are all represented as lying somewhere above the surface in a higher-dimensional space. When an image is reduced to a signature, it is projected onto the surface. The DFFS error is then visualised as the difference between the original image and its "shadow" projection on the surface.

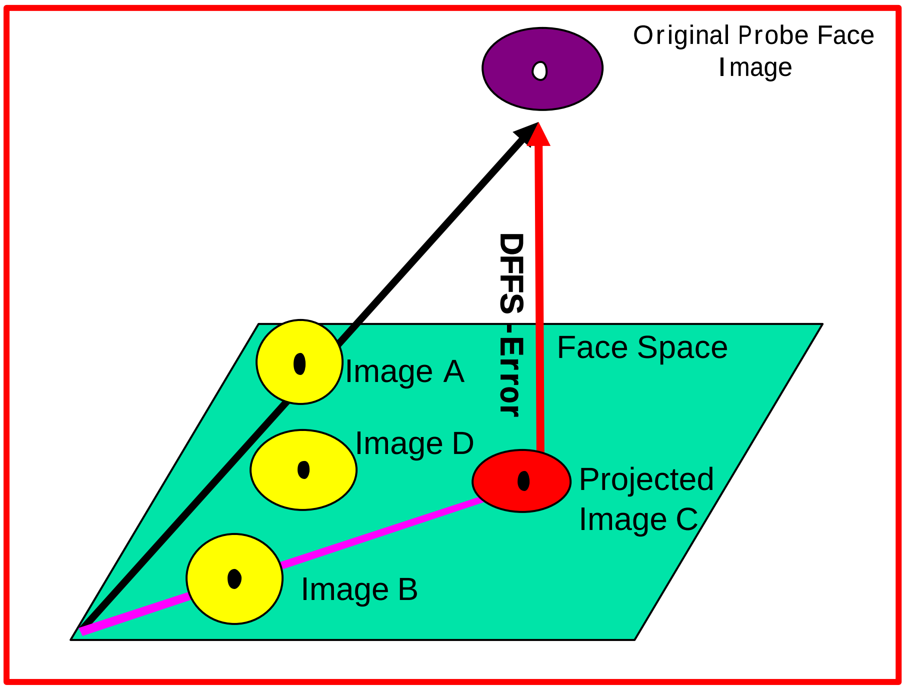


Figure 7f: A pictorial representation of a DFFS and a computation of a match

Figure 7f attempts to clarify the difference between generating DFFS information and the matching process. In the figure above, there are training-set images that are on the face space, i.e. images A, B and D. These images form both the gallery and the training set. Since the training set was used to determine where to put the surface in the first place, these images can be accurately represented with zero DFFS. Image C is a projected probe image: the original image is not on the space – only its projection is there. Image C is represented as a linear combination of the face-space eigenfaces. The DFFS, as already mentioned, is arrived at by computing a similarity measure between an original image and a reconstructed (projected) image.

A match is generated by computing the distance between the reconstructed gallery and probe-set projections. Referring again to the figure above, the match will be generated by computing similarity scores or image distances from C, i.e. the distance between C and A, C and B, and C and D. The smallest distance is interpreted as a match in that it has the fewest differences from the probe image.

Knowledge about different DFFS does not directly help to improve matching performance. But DFFS information is critical if we want to *classify* information i.e. to decide on which face space the face image best fits. The rest of this chapter is dedicated to investigating this question.

7.6 Gender classification from DFFS

In this section we look at 3 different approaches to using DFFS information for gender classification. At first we use all the eigenfaces from a face space. Secondly, we use only the top *n* eigenfaces. Lastly we use the last *n* eigenfaces, in the manner discussed by Abdi et al (1995).

7.6.1 A simple DFFS classification with no pre-excluded eigenfaces

We divided our image set into a group of 21 males and 21 females. We generated two face spaces of 20 eigenfaces each, using all male images in one and all female images in the other. We then used 42 probe images such that each image in the training set has one corresponding image in the probe set. We project each image onto both face spaces to get the DFFS values from each. The DFFS values are then compared and the face space that generates the smaller value is taken as a correct classification.

Using this simple DFFS classification method, with no priori information about the gender-specific face features, and using all eigenfaces from the corresponding face spaces, we achieved 76% correct classifications on male face images and only 52% on female images. (Note that 52% correct classification is hardly better than a random guess, since flipping a coin may be predicted correctly in 50% of the cases.) Overall, our scores are not as good as reported in some other studies. One possible reason is that we have no method for pruning our space to use only those eigenfaces that somehow capture gender. We return to this point shortly.

The difference between male and female picture classification scores is hard to explain, but it is a feature that has also been noted in other work. Abdi et al (1995) also had uneven distribution of successful gender classification scores, 87% for males and 66% for females. They attributed this inconsistency to the fact that some female faces (especially females with short hair) looked like males and most male faces were more similar to each other. This might also be features shared by our own image set. Supporting evidence occurs in Phillips et al (2003) in the Face Recognition Vendor Test (FRVT) report, where they also have better recognition results for male face images than for female face images.

7.6.2 DFFS classification with pre-excluded eigenfaces

The traditional PCA algorithm is based on the notion that eigenvectors with high eigenvalues are "more significant" for recognition, since they capture more variance critical for individual face representation [Turk and Pentland (1991b), Pentland et al (1994), Moon et al (1999), Howell (1999)]. Good performance is expected from a PCA algorithm, even if only the few most significant eigenfaces are used for a face space. This standard of success has stood the test of time and has become a de facto standard for PCA-based algorithms.

Abdi et al (1995), Yambor et al (2000), Moon and Phillips (1998) and Sun et al (2002) claim that one can visually tell what different eigenfaces represent e.g. illumination, faces with glasses. Moon et al (1999) and Yambor et al (2000) even claim an improved performance by removing some high-eigenvalue eigenfaces that encode glasses and moustaches. We attempted to verify these claims by knocking out some of our eigenfaces, but were unable to reproduce their results.

We used exactly the same set of images in the above experiment except that we generated four face spaces, two with the top 10 eigenfaces (the top half of the face space used in the above experiment) and two with 10 low-end eigenfaces (the bottom half of the face space used in the above experiment). This follows the suggestion of Yambor et al (2000) that "high-order, fine features" were more significant for gender differentiation. The following graph presents our classification scores.

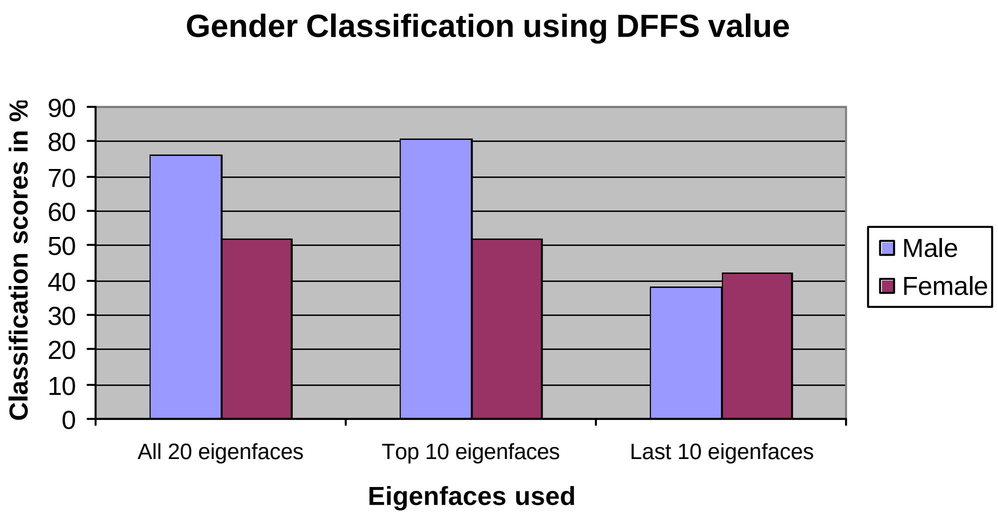


Figure 7g: Gender classification (from Rhodes University Face Database) using DFFS

The graph shows that the low-eigenvalue eigenfaces are not good enough for the system to successfully classify the images. The top 10 eigenvectors achieved the best classification in male faces (81%) and the worse male classification was 38% from the low-end eigenfaces.

The female images proved to be harder to classify, achieving a best score of 52% for all eigenfaces and for the top 10 respectively. The worse classification score for females was 42% which was better than the 38% worst score from males. There is more here than meets the eye. A totally random coin flip should score 50%. What these results seem to show is that low-end eigenfaces perform worse than random – e.g. they influence the classification negatively. While we cannot explain why this is, it does explain the improvement when they are excluded.

7.7 Race classification using DFFS

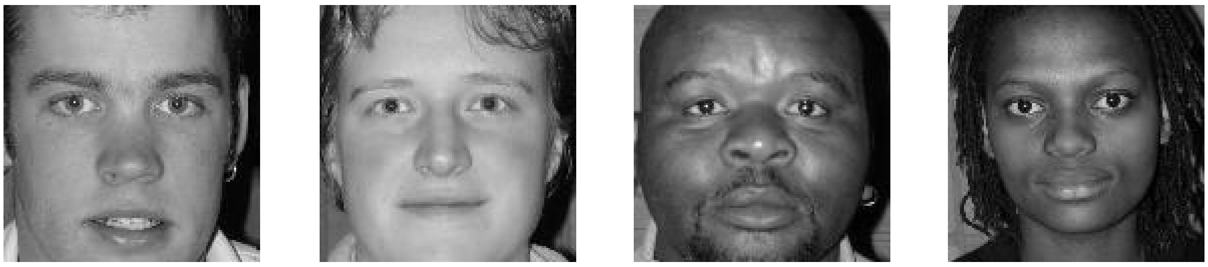
Race classification is of value for a number of fair discrimination purposes such as tracking the preferences of one race for specific products in a retail store or to aid (or enhance) identification.

An identification system that can classify an image based on its race or gender might improve identification performance since it can establish identity only against a limited subset of known face images. In this project we are interested in race classification with the aim of improving our face-identification scores.

We separated our images into two groups, Africans (i.e. Blacks) and Caucasians (i.e. Whites). We did not have enough face images to produce comparable experimental results for Asians and Indians. The training and probe sets are made up of 22 images such that each image in the training set has exactly one corresponding image in the probe set. Our training and probe sets are made up of both male and female images.

We employed the same technique as in (section 7.6.1) our gender classification experiments, where two face spaces were generated. The face spaces are made up of 20 eigenfaces (i.e. all the eigenfaces) and the top 10 eigenfaces have been sorted using higher eigenvalues. We project an image onto both face spaces and the face space that returns the smallest DFFS is regarded as the correct classification. For example, if projecting image A onto the Caucasian face space returns a DFFS value of 100 and A's projection onto the African face space returns a DFFS value of 60, then the system classifies image A as an African.

In this category all different face spaces gave very good results, with 100% accurate classification for Caucasian images and 95% (21 out of 22 images were correctly classified) classification for African images (using all 20 eigenfaces and using only the top 10 eigenfaces). This performance might have been achieved because there is such a huge difference between the African images and the Caucasian images. The figure below displays some of the probe images:



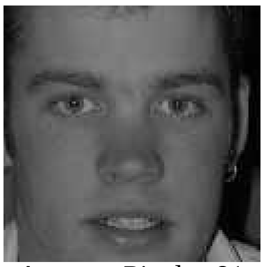
Caucasian FS=2265.88	Caucasian FS=1090.23	African FS =3615.54	African FS =3183.36
African FS =5217.10	African FS =6072.05	Caucasian FS=4106.58	Caucasian FS=4322.69
Average Pixel = 127	Average Pixel = 140	Average Pixel = 92	Average Pixel = 69

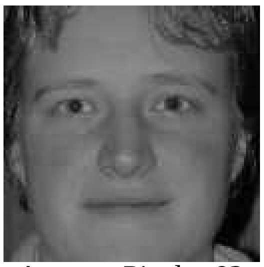
Figure 7h: Sample of probe images and their corresponding DFFS values

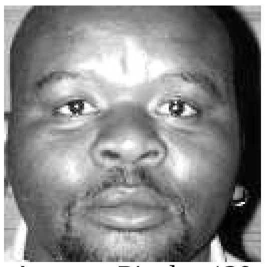
The above probe images, supported by DFFS values (shown under each image against the African or Caucasian face space), show that it is much simpler to classify an image based on race than on gender. In an attempt to understand the reason behind good race-classification scores we computed average pixel intensity values. Average pixel intensity values give the overall brightness of the image. Our aim is not to classify images based on brightness but on race-specific characteristics.

The next experiment is aimed at testing whether PCA classifies images on the basis of race-specific image characteristics or whether PCA really uses brightness information as a means of race classification. If the classification scores worsen after changing the brightness, we shall be able to infer that the above findings will be very limited in real-life applications. A good system should be able to classify race based on race-specific characteristics not only on the basis of facial colour and reflectiveness.

We added more brightness to the African images and computed the classification scores against this set. At first we brightened the images only slightly, giving just a small variance in average pixel intensity values. We also darkened the Caucasian images and computed the classification scores again. The resulting images were as follows:


Average Pixel = 81


Average Pixel = 92


Average Pixel = 129

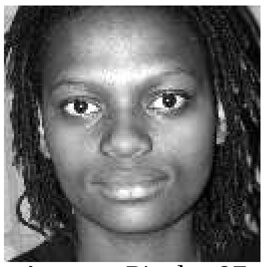

Average Pixel = 97

Figure 7i: A sample of slightly brightened and darkened probe images with average pixel intensity values

The classification scores for both Africans and Caucasians remained very high at 91% correct classifications i.e. 20 out of 22 images. This suggests that brightness has a negligible impact on race classification. We repeated the same experiment with greater darkness for Caucasian images and brightness for the African images. The goal here was to verify our results in extreme cases. A sample of the images was as follows:

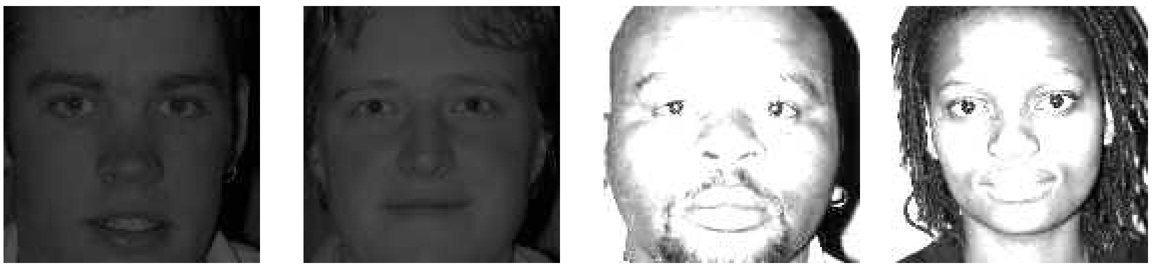

Average Pixel = 43 Average Pixel = 48 Average Pixel = 198 Average Pixel = 159

Figure 7j: Sample of greatly brightened and darkened probe images with average pixel intensity values

The classification scores remained unchanged at 91% for both Caucasians and Africans. This experiment made us finally able to conclude that brightness has very little to do with race classification and that PCA does indeed classify images on the basis of race-specific facial characteristics. In summary, PCA can be effectively used to add value in race-classification problems.

The following code shows our concrete implementation of the ideas discussed above, again to give the reader some sense of the flavour of our tools. When we run the exclusively male (or female) probe images through this algorithm, it tells us how many are correctly (or incorrectly) classified.

```
public string ClassifyUsingDFFS(string imagesPath,string spaceNameMale,string
spaceNameFemale)
   {  // image names to be used i.e. male or female image names
      string [] imageNames = getallimageNames(imagesPath,"*.JPG");
      int maleCount = 0; //counter for correct classifications
      //loop through the image list, computing DFFS values
      for (int i = 0; i< imageNames.Length; i++)
      {   //project an image onto the male face space
            Feature male = getFeature(spaceNameMale,imageNames[i]);
            //project an image onto the female face space
            Feature female = getFeature(spaceNameFemale,imageNames[i]);
            //compare signatures
            if ( male.diffFromImageSpace < female.diffFromImageSpace)
            {
                   maleCount++;  // increase counter
            }
      }
      return maleCount.ToString()+ " correctly classified from "+
      imageNames.Length.ToString();
```

}

7.8 Using multiple face spaces to improve recognition performance

The use of multiple face spaces can help improve both the image classification and identification success rates. The good results from race classification can be used as a foundation for improved performance in other areas. For example the gallery-set images can be grouped by race so that there is a gallery for Africans, another for Caucasians and a further one for those who are neither African nor Caucasian. When a new probe image is to be identified, its race is established first and it is matched against only the corresponding gallery group. The same applies for gender classification: gender can be established after race.

7.8.1 Gender classification after race classification

The gender classification problem using PCA (and the nearest neighbour algorithm) proves to be more difficult than race. Kim (2002) reports 61% correct gender classification and 80% correct race classification from 200 training and probe-set images.

In our experiment we used 22 training and 22 probe-set images belonging to male and female Africans and Caucasians. In each category there were 8 female images and 14 male images.

The overall matching scores did not improve. The best classification score came from male Africans with 71% (10 out of 14 images), followed by Caucasian males with 57% (8 out of 10 images). The female matching score was 38% (3 out of 8 images) for both Africans and Caucasians. In short, gender classification did not improve after dividing images by race (see Figure 7g above for gender classification without pre-determined race).

7.8.2 Image recognition after race classification

As explained above, gender classification did not become more successful after categorising images by race. This section investigates the effects of categorising images by race before identification. We are looking at more than just improved performance due to a smaller gallery set. We use the same set of images as above, except that they are not categorised by gender. The top 3 matches are used for correct identification (giving a 14% error rate) from 22 images from each category. We generate matching scores using Euclidean (L1) distance, City-Block (L2), Cosine Angle, Mahalanobis, L1+Mahalanobis, L2+Mahalanobis and Angle + Mahalanobis distance measures. A face space of 20 eigenfaces, generated from 20 African and Caucasian images respectively, is used.

The correct identification scores failed to convince us that race classification prior to identification improves identification performance. The Angle + Mahalanobis distance measure gave the best performance at 45% (10 out of 22 images) against African images and a 41% (9 out of 22 images) best performance against Caucasian images. This performance is not very different from matching scores achieved without any race classification, as was discussed in the previous chapter. Given that our classifier also makes some errors even before the misidentification in the reduced space, it appears that the two-tier approach of classifying before identification is not as effective as simple identification without prior classification.

In summary, categorising images based on race failed to improve either gender classification or image identification. This does not necessarily imply that automatic race classification is of no use in the face recognition process. It can still be used to categorise images automatically, splitting them into smaller subcategories which can then reduce the dimensions of the gallery set. Automated race-classification information can still be used to collect race-related statistics for other purposes such as marketing.

7.9 DFFS classification of faces from non-faces

The problem of classifying faces from non-faces is covered under face detection: given an image, can you find a face, or is there a face in a given image? The only difference with using a PCA approach is that the target image (or the section of the target image) and the training-set images must be of the same size. In this section we are looking at using the DFFS measure for deciding if a given image is a face image or not.

7.9.1 Related work

Face detection has been an active field of research for the past decade. It is impossible to cover all the developments in this field since then. A detailed view of face-detection research can be found at **Robert Frischholz's face-detection home**[9] page; the Hjelmas and Low (2001) **face-detection survey**; and in a list of face-publication readings from the **Machine Perception Laboratory (MPLAB[10])** at the Institute for Neural Computation.

Jung et al (2002) implemented a PCA-based real-time face detection and tracking system where the 'faceness' of an image is computed using the DFFS technique. They report correct classification rates of 95% for faces and 92% for non-faces. Lu and Sun (2003) report 71% to 100% correct face detection in their PCA and neural network based colour-face-image detection. They repetitively used groups of seven images in their experiments. The images had a complex background in that they contained more than one face and the faces themselves were randomly distributed over the image.

The more recent implementations of PCA-aided algorithms in face detection reflect the maturity of the PCA algorithm in face-processing applications. In our experiments we use a purely PCA and Euclidean distance-driven approach to face detection. Our goal is to decide whether a given image represents a face or not.

7.9.2 Face-detection experiments using a DFFS measure

We projected 10 face, car and military airplane images onto a face space of 10 eigenfaces. The following figure shows a sample of images projected onto the face space.

[9] Available online at: http://home.t-online.de/home/Robert.Frischholz/face.htm

[10] Available online at: http://mplab.ucsd.edu/FaceDetectionReadings.html

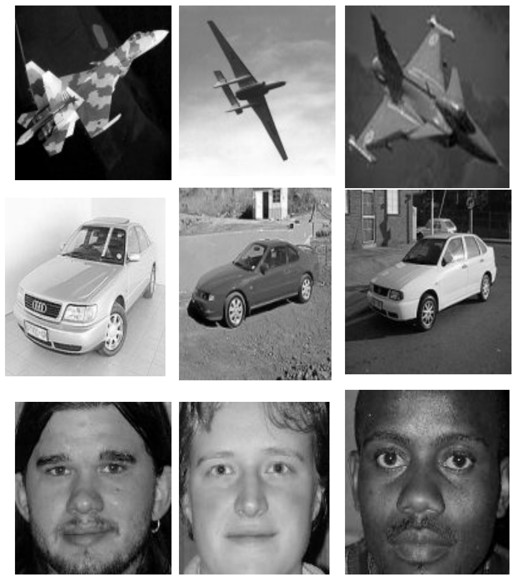


Figure 7k: A sample of images projected onto the face space

DFFS values from these images were computed and are graphed in the next figure. DFFS from different images were given numbers from 1 to 10.

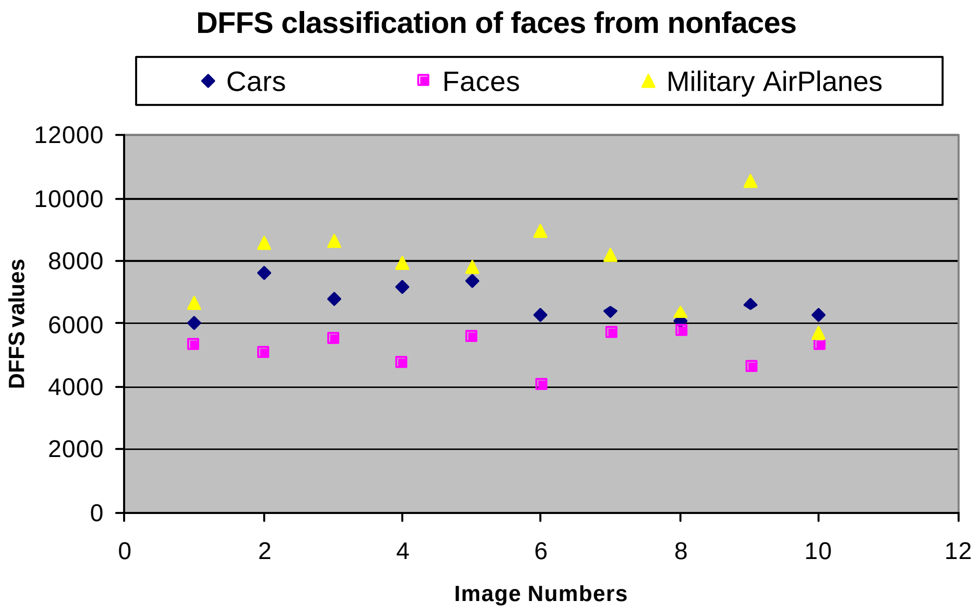


Figure 7l: DFFS classification of faces from non-faces

All face images generated DFFS values that were less than 6000. The DFFS values from cars and military airplanes were mostly higher than 6000. In this case 6000 can be used as a threshold. If a given image generates a DFFS value of more than 6000 points, it will be regarded as a non-face image. It will then be rejected by the system. The opposite also applies: if an image generates a DFFS value less than the 6000 threshold, it will be accepted as a face image by the system.

An interesting observation to note is that the DFFS values from military airplane images are generally distributed above those for the car and the face. The graph simply shows that the DFFS values from a single category are closely distributed together. Faces images generated DFFS values less than 6000 points, while most car images generated DFFS values between 6000 and 8000 points. Lastly most military airplane images generated DFFS values greater than 8000 points.

The DFFS information enables us to tell with greater confidence if an image is a face or a non-face. This suggests a much more valuable use of DFFS information in class detection rather than in specific image identification. Such an application may add more value to content-based image classification or retrieval.

7.10 Conclusion

The primary purpose of this chapter was to provide some evidence that our framework is indeed useful for helping to answer questions about face recognition. As a specific validation case, this chapter has explored the different secondary roles that DFFS information can play in image processing. It built on chapter five where the visual interpretation of face-space information is discussed.

Gender classification proves to be a difficult problem. The fact that our images are scaled with eyes positioned at the same place across all images could have contributed to our poor results. Traditional geometric-based face algorithms can classify gender based on distances between different face features. Our scaling obviously removes any advantages that may be gleaned from knowing the absolute size of the face.

DFFS classification did very well in our small samples of classifying information according to race. But this prior classification ahead of other matching failed to improve subsequent gender classification or face recognition in general. On the other hand, the DFFS measure performed well in classifying faces from non-faces.

Although the DFFS information failed to improve face-recognition performance, there are application domains where the technique could be of great value given its simplicity. It seems feasible to exploit the DFFS classifier for marketing purposes, for automatic categorisation or for detecting faces from non-faces i.e. in an image-content-driven retrieval domain.

Chapter 8

Conclusion and Future Work

8.1 Introduction

This treatise is targeted at two different types of reader. The first is a researcher familiar with the progress being made in face-recognition systems but who has not had hands-on experience with the technology. The work has covered aspects of the low-level processes involved, how they interact with one another, and most importantly, how these processes can be distributed across the Internet via SOAP/HTTP and XML standards through web services. DRUBIS is of value to this class of users.

The second target audience is those who are not familiar with the different technologies and techniques involved in face recognition. This work hopes to balance some of the media hype about the performance and promise of face-recognition systems by providing an exposure to the issues that need to be overcome in the development of a production-quality face-recognition system. This project has also served as a detailed summary of the issues involved and the progress being made in the face-recognition field in general. We managed to combine assertions from the literature and ran practical tests on some of these.

8.2 Thesis contribution

Our design and implementation of DRUBIS serves as a proof-of-concept of our biometric framework. We have looked at a number of the factors surrounding proprietary implementations of PCA-driven face-recognition systems. The analysis and in-depth investigation of how a PCA algorithm for face recognition works, what factors influence identification performance and, most importantly, the extended role of DFFS in image classification serves as the core of our work. A more concise summary of this aspect of the work appear in [Ndlangisa and Wentworth (2003)].

8.2.1 DRUBIS Software

Our implementation of a distributed biometric framework has served as a small-scale deployment of an Internet- and XML-driven biometric system. The concept can be easily extended to cater for different biometric systems such as those based on palm-prints or fingerprints. The concept can also be of value to large-scale identification systems distributed nationally (even internationally) utilising a common database.

Our face-recognition demonstration applications at the 2003 National SASOL Science Festival, in Grahamstown, South Africa, gave us an opportunity to test the real-life performance of our XML web-service-based distributed face-recognition system. This gave us an extra opportunity to assess the performance of our face-recognition system against live data. Such performances are not usually published in the literature and even de facto standards such as FERET or FRVT (Face Recognition Vendor Test) only report face-recognition performances on still images. In this respect, our research is ground-breaking.

8.2.2 PCA-driven face-recognition systems

When we started this project we had visions of implementing a system that could recognise people entering the Rhodes University Computer Science Department and greet them by their names. That was before getting hands-on experience with the technologies and exploring the literature in depth. Face-recognition techniques appear reasonably simple, yet it has proven difficult to build robust, good quality recognisers.

On the other hand, the mass deployment of the technology suggests reasonable maturity. This project has explored the issues involved in deploying proprietary PCA-driven face-recognition applications. In contrast to the media hype and some laboratory experiments, we have proven that, though the PCA technique is mature, its accurate application to real-life face-recognition problems demands high investments in terms of image preprocessing and also a high degree of control before images can be fed into the system.

Our experience confirms the claims of Turk and Pentland (1991a) and Turk (2001) that PCA is not particularly suitable for use in unconstrained situations. For some applications, such as the use of police mugshots of criminals, which are constrained, fixed-distance and blank-background shots, it may be more useful. However, it is unlikely to perform well unless other factors (such as different hair lengths, beards, or the wearing of spectacles) are also well controlled.

8.2.3 Visual eigenface information interpretation

The use of visual eigenface information is well published in the literature. This project has confirmed some of the claims that have been made and casts doubt on others. The idea of throwing out some of the eigenfaces, based on what they represent (e.g. beard or glasses) did not work in our trials. On the other hand, we confirmed that eigenface information could distinguish the quality of the face space: how well the space captures the training set or how different the training set images are from each other. We demonstrated this in Chapter 5 of this thesis.

8.2.4 DFFS in image classification

We have applied existing knowledge from the literature about the role of DFFS in image classification. We looked at different ways of applying a DFFS value to improve the overall face-recognition process. Our good results in race classification serve as a basis for further analysis of DFFS-based face detection. The good rate of differentiation between face images and non-face images possibly opens the door to a new domain of DFFS application i.e. in content-based image retrieval.

8.3 Discussion

Huge progress has been made in the face-recognition field over recent years. The use of the PCA algorithm and information-theory-based algorithms for face recognition has grown enormously. The issues now are mostly whether a better performance can be gained from using the neural network nearest-neighbour approach, as opposed to the various distance measures such as Euclidian distance, City-Block; or whether the neural network approach is to be preferred when making decisions concerning image matching.

The computational expenses involved in preprocessing images before feeding them to the algorithm are prohibitive in a real-life environment. There are good face detectors and images can be scaled automatically online but factors such as illumination and other pixel-level operations are more demanding in terms of time. These can be best achieved in an offline mode and they are not easy to automate.

In short, though PCA has stood the test of time, it is still not a simple matter to implement efficient, proprietary face-processing systems.

The other puzzle from the literature is how to decide which eigenfaces to use for different application domains. There seem to be a number of different approaches. The de facto standard championed by Turk and Pentland (1991a) and many others says that the top eigenfaces capture more information critical for identification and that the low eigenfaces should be ignored because they capture less information. Yambor et al (2000) and Moon and Phillips (1998) claim that, within the top eigenfaces, there are eigenfaces containing noisy data such as information about glasses and lighting, and that these should be removed. They report improved results after removing some of these eigenfaces. We failed to confirm these claims in our experiments.

O'Toole et al (1998) and Abdi et al (1995) claim that, while the top eigenfaces tell more about the low-level frequency of an image (they are good for identification), low-eigenvalue eigenfaces represent more information about the high-level frequency (they are good for gender or race). They demonstrate this claim using some of their images but we also failed to replicate this unpopular approach. Abdi et al (1995) also claim that a second top eigenface contains more gender information than others. They base this information on data gathered from their tests.

In summary, there is a need for a formal study to clear up the mystery surrounding eigenface selection. Without such a study, there will be more and more claims that are hard (if not impossible) to replicate, or that are face-database specific, and there will be no means of verifying them. We believe frameworks such as the one we have constructed will be useful for promoting studies in these areas.

8.3.1 The FERET Database

The motive behind a FERET database was to create a medium where different face-recognition algorithms could be compared [Phillips et al (1996)]. The aim was to formalise an assessment of the progress being made in the field. To achieve this goal, the FERET evaluations took neither very complex nor very easy image sets for evaluations: the intention was to assess progress in the field by using a fair sample of images.

The most published results to date use Fa and Fb or FB images (images taken on the same day, one after the other or very similar) from the FERET database. These image categories are not very different from images from the ORL, MIT, and other face databases. The Duplicate 2 image set (images taken days apart) from FERET has produced very poor results [Moon and Phillips (1998), Phillips et al (2003)].

In retrospect, our own images are more like the Duplicate 2 category in FERET, for which very low matching scores are the norm. These results are comparable to our matching scores using images taken two weeks apart.

Perhaps the most important lesson that we learnt is that publishing rates of good recognition is more or less meaningless unless we also have significant evidence about the image sets.

The converse is also true: the results and conclusions that we have presented in this work may be valid only in limited situations. For example, if we were to redo all our experiments with a highly posed and controlled set of images, we might well find that DFFS is a good discriminator for gender classification.

8.4 The limitations of this project

When we started this project we were new in the field of face recognition. It took us some time to grasp concepts and as a result we ended with an in-depth study of only PCA-based techniques. We had envisioned having more components, perhaps a neural-network-based component encapsulated as a web service and integrated into DRUBIS. In spite of the fact that we did not achieve this, DRUBIS lays a good foundation for such extensions to be implemented in future. This project also demonstrates the feasibility of making all face-recognition systems readily available via the web. This would greatly assist researchers who could perform comparative experiments without having to own or deploy the software themselves.

We had 53 images taken two weeks apart. Images spread over time are the usual case in the real world. In retrospect, it would have been interesting to have had images taken on the same day, one after the other, or images taken only few days apart. This would have helped us in our study of how recognition performance declines over e.g. a day or a week. Storing a new image whenever a user is authenticated, and discarding the old enrolled image in an access-control face-recognition system might prove very interesting and useful.

Since we had only two images per user it was impossible for us to examine whether DFFS information can be used to classify images of the same user. This can only be achieved by generating an independent face space for each user, using a neural network on PCA-generated signatures or by applying discriminant analysis techniques to classify images.

8.5 Future Work

We have clearly demonstrated from our experimentation framework that a biometric application can be distributed across the Internet. More work would have to be done if we were to adapt our framework for fingerprint recognition. Extending our framework so that it might be easily configured to support multiple biometrics would be a helpful and useful extension to it. This would also be in line with the vision of an XML-driven Common Biometric Information Interchange Format (CBIIF) for different biometric systems.

This project has highlighted a number of areas where PCA-based image-classification algorithms can be of help, for example in marketing (where the number of Africans or Caucasians buying a specific product might be ascertained) or in content-based image-retrieval applications. A detailed (or a comparative study with different techniques) study of this application area might open doors to the new world for PCA in image processing.

An investigation into what information different eigenfaces encapsulate would clarify much misunderstanding in this area. Such a study could help improve recognition and classification rates. Eigenfaces corresponding to noisy data could be filtered out from the face space. Alternatively, image-signature coefficients generated from these eigenfaces could be removed before computing similarity scores.

8.6 Conclusion

Face recognition is a fascinating field that has drawn much attention lately, as is evidenced by publications over the past 10 years. This thesis is our contribution to this pool of knowledge. Our main contributions are in highlighting the role that can be played by XML web services in face-recognition systems. We developed a PCA experimentation framework; we raised and answered some of the PCA face-recognition questions e.g. what are the optimum face spaces and how training images should be selected. Lastly we looked at group face-image classification. In addition, we have provided a platform which we hope will facilitate future work in this area in our institution, or by outsiders.

References

[Abdi et al (1995)]	H. Abdi, D. Valentin, B. Edelman and A.J. O'Toole. **More about the difference between men and women: evidence from linear neural networks and the principal component approach**. Perception, 1995, Volume 24, pp. 539-562.
[Achermann and Bunke (1996)]	B. Achermann and H. Bunke. **Combination of Face Classifiers for Person Identification**. In Proceedings 13th IAPR International Conference on Pattern Recognition (ICPR), Vienna, Austria, Vol. III, pp. 416-420, August 1996.
[Beveridge and Draper (2003)]	R. Beveridge and B. Draper. *Evaluation of Face Recognition Algorithms web site***.** Hosted by Colorado State University. Online: http://www.cs.colostate.edu/evalfacerec/ .
[Brunelli and Poggio (1993)]	R. Brunelli and T. Poggio. **Face recognition: feature versus templates**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(10):1042-1052, 1993.
[Campos et al (2000)]	T.E. de Campos, R.S. Feris and R.M Cesar Junior. **Eigenfaces versus Eigeneyes – First step Towards Performance Assessment of Representations for Face Recognition**, Lecture Notes in Artificial Intelligence, vol.1793, pp. 197-206, April 2000, Springer Verlag.
[Chaudhary et al (2002)]	A.S. Chaudhary, M.A. Saleem, and H.Z. Bukhari. **Web Services in Distributed Applications -Advantages and Problems**, IEE conference at Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Pakistan in April 2002.
[Curbera et al (2001)]	F. Curbera,W. Nagy and S. Weerawarana. **Web Services: Why and How**, IBM TJ Watson Research Center. Online: http:// www.research.ibm.com/people/b/bth/OOWS2001/curbera.pdf
[Czyz et al (2002)]	J. Czyz, J. Kittler and L. Vandendorpe. **Combining Face Verification Experts**, 16th International Conference on Pattern Recognition (ICPR'02)

Volume 2 August 11 - 15, 2002, Quebec City, QC, Canada.

[Egmont-Petersen et al (2002)] M. Egmont-Petersen, D. de Ridder, H. Handels. **Image processing with neural networks - a review**, *Pattern Recognition*, Vol. 35, No. 10, pp. 2279-2301, 2002

[Fang et al (2002a)] Y. Fang, T. Tan, Y. Wang. **Fusion of Global and Local Feature for Face Verification**, IEEE International Conference on Pattern Recognition (ICPR), 2002.

[Fang et al (2002b)] Y. Fang, T. Tan, Y. Wang. **Personalizsed Feature Combination for Face Recognition**, IEEE Region 10 Technical Conference on Computers, Communication, Control and Power Engineering, 2002.

[Flurry (2001)] G. Flurry. **Applying web services to the application service provider environment: An example of web services applied to e-businnes**. IBM DeveloperWorks. January 2001.

[Frischholz (2003)] R. Frischholz, **Face Detection Home Page**.
Online: http://home.t-online.de/home/Robert.Frischholz/face.htm.

[Glass (2000)] G. Glass. **The web services (r) evolution: Applying web services to applications**. IBM DeveloperWorks. November 2000.

[Gross et al (2001)] R. Gross, J. Shi, J. Cohn, "Quo Vadis Face Recognition**",**
Third Workshop on Empirical Evaluation Methods in Computer Vision, Kauai, HI, December 10, 2001.

[Haas and Brown (2003)] H. Haas and A. Brown. **Web Services Architecture Working Group**, W3C Working Draft 8 August 2003.
Online: http://www.w3.org/TR/2003/WD-ws-gloss-20030808/.

[Haddadnia et al (2002)] J. Haddadnia, K. Faez, and M. Ahmadi. "N-Feature Neural Network Human Face Recognition," Proceedings of the 15th International Conference on Vision Interface, May 27-29, 2002, Calgary, Canada.

[Hjelmas and Low (2001)] E. Hjelmås and B.K. Low. **Face Detection: A Survey**. *Computer Vision and Image Understanding*. 83, pp. 236-274, 2001.

[Hong et al (1999)] L. Hong, A. Jain and S. Pankanti, "Can Multibiometrics Improve Performance?", *Proceedings AutoID'99*, Summit, NJ, Oct 1999, PP. 59-64.

[Howell (1999)] A.J. Howell, **Introduction to Face Recognition** in L.C. Gain, U. Halici, I. Hayashi, S.B. Lee and S. Tsutsui, *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, CRC Press,1999.

[Jiang (1996)] Principal Component Analysis and Neural Network-based face-recognition. Degree thesis, University of Chicago. Available: http://people.cs.uchicago.edu/~qingj/ThesisHtml/ .

[Jung et al (2002)] D.J. Jung, Y. Bak, C.W. Lee, Y.C. Lee, J.B. Kim, H. Kang, H.J. Kim, "PCA-Base Real-Time Face Detection and Tracking", *International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC'02)*, Jul, Phuket, Thailand, Vol. 1 , pp. 615-618. 2002.

[Kim (2002)] S. Kim. "Face Classification with Eigenfaces", Pattern Recognition and Analysis, Fall 2002 Class Project.
Online : http://courses.media.mit.edu/2002fall/mas622j/proj/students/kim/

[King and Xu (1997)] I. King and L. Xu. Localised principal component analysis learning for face feature extraction and recognition. In *Proceedings to the Workshop on 3D Computer Vision '97*, pp. 124-128, Shatin, Hong Kong, 1997. The Chinese University of Hong Kong.

[Kotropoulus et al (1999)] C. Kotropoulos, A. Tefas, I. Pitas. "Morphological Elastic Graph Matching Applied to Frontal Face Authentication Under Optimal and Real Conditions". IEEE International Conference on Multimedia Computing and Systems Volume II-Volume 2, June 07 - 11, 1999, Florence, Italy.

[Krueger (2001)] M. Krueger, SharpZipLib written entirely in C# for the .NET platform. Available:

http://msd2d.com/FreeCode_view.aspx?section=dotNet&category=Libraries

[Lam and Yan (1998)] K. Lam and H. Yan, "An Analytic-to-Holistic Approach for Face Recognition Based on a Single Frontal View", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 20, July 1998.

[Leung et al (1995)] T.K. Leung, M.C. Burl, and P. Perona. **Finding faces in cluttered scenes using random labelled graph matching**. In Fifth International Conference on Computer Vision, pp. 637-644, Cambridge, Massachusetts, June 1995. IEEE Computer Society Press.

[Lin (2000)] S. Lin, **An Introduction to Face Recognition Technology**, Informing Science Special Issue on Informing Systems and Multimedia Technologies-Part 2, Volume 3 No 1, 2000. Available:http://inform.nu/Articles/Vol3/v3n1p01-07.pdf

[Lu and Sun (2003)] W. Lu and S. Sun, "Face Detection in Color images," EE 368 Project. Available: http://www.stanford.edu/class/ee368/Project/reports/ee368group03.pdf

[Marcialis and Roli (2002)] G.L. Marcialis and F. Roli, "Fusion of LDA and PCA for Face Recognition", *Proc. of the Workshop on Machine Vision and Perception*, held in the context of the 8th Meeting of the Italian Association of Artificial Intelligence (AI*IA), Siena, Italy, September 10-13, 2002.

[Micheals et al (2003)] R.J. Micheals, P.J. Grother, and P.J. Phillips, "The NIST HumanID Evaluation Framework", Proceedings 4th International Conference on Audio Visual Based Person Authentication, 2003.

[Moghaddam and Yang (2002)] B. Moghaddam, M. Yang. **Learning Gender with Support Faces**. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 24, No. 7, July 2002.

[Moghaddam et al B. Moghaddam, T. Jebara and A. Pentland, "Bayesian Modeling of Facial

(1999)] Similarity”, Advances in Neural Information Processing Systems 11, MIT Press, 1999.

[Moon and Phillips (1998)] H. Moon and P.J. Phillips. **Analysis of PCA-based Face Recognition Algorithms in Emperical Evaluation Techniques in Computer Vision**, K.W.Bowyer and P. J. Phillips, 1998.

[Navarrete and Ruiz-del-Solar (2002)] P. Navarrete and J. Ruiz-del-Solar. **Comparative Study between Different Eigenspace-Based Approaches for Face Recognition**, AFSS 2002: 178-184

[Ndlangisa and Wentworth (2003)] M. Ndlangisa and E.P. Wentworth. **DRUBIS: Distributed XML Driven Biometric Identification System.** Proceedings of SATNAC (Southern African Telecommunication Networks & Applications Conference) 2003, George, South Africa, September 2003.

[O’Toole et al (1998)] A.J. O’Toole, K.A. Deffenbacher, D. Valentin, K. McKee, D. Huff, H. Abdi, (1998, in press). **The perception of face gender: The role of stimulus structure in recognition and classification**. *Memory and Cognition*, volume 26, pp.146-160.

[Pandya and Szabo (1999)] A.S. Pandya and R.R. Szabo. **Neural Networks for Face Recognition**. In L.C. Jain, U.Halici, I. Hayashi, S.B. Lee and S. Tsutsui, Intelligent Biometric Techniques in Fingerprint and Face Recognition,CRC Press,1999.

[Pentland et al (1994)] A. Pentland, B. Moghaddam and T. Starner, “View-Based and Modular Eigenspaces for Face Recognition” , *IEEE Conference on Computer Vision & Pattern Recognition*, Seattle, WA, July 1994.

[Phillips et al (2003)] P.J. Phillips, P.G. Grother, R.J. Micheals, D.M. Blackburn, E. Tabassi and M. Bone, **Face Recognition Vendor Test 2002**. Available: www.frvt.org.

[Phillips et al (1996)] P.J. Phillips, P. Rauss and S. Der, **FERET (FacE REcognition Technology) Recognition Algorithm Development and Test Report,** ARL-TR-995, U.S. Army Research Laboratory, 1996.

[Phillips et al (2000)] P.J. Phillips, H. Moon, P.J. Rauss, and S. Rizvi, "The FERET evaluation methodology for face recognition algorithms," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 10, October 2000.

[Qahwaji et al (2002)] R. Qahwaji and R. Green, “Improving the Recognition Performance of PCA,” IEE CATEE 2002, Amman, pp. 378-381, 2002.

[Raj (1998)] G.S. Raj, A Detailed Comparison of CORBA, DCOM and Java/RMI. Online: http://www.hirstlogics.net/scraps/expli/compare.html.

[Rzeszutek (2002)] K. Rzeszutek, Dynamic Scalable Distributed Face Recognition System Security Framework. Available Online: http://darnok.com/~konrad/presentations/dist-face-reco-framework/.

[Short (2002)] S. Short, **Building XML Web Services for the Microsoft® .NET Platform**, Microsoft ® Press, 2002.

[Sun et al (2002)] Z. Sun, G. Bebis, X. Yuan, and S. Louis, "Genetic Feature Subset Selection for Gender Classification: A Comparison Study", **IEEE Workshop on Applications of Computer Vision**, pp. 165-170, Orlando, December 2002.

[Tang et al (2003)] H. Tang, M.R. Lyu and I. King,”Face Recognition Committee Machines: Dynamic vs Static Structures”, 12th International Conference on Image Analysis and Processing (ICIAP'03), September 17 - 19, 2003, Mantova, Italy.

[Turk (2001)] M. Turk, “A random walk through eigenspace,” *IEICE Trans Information and Systems*, Vol.E84-D, No.12, pp. 1586-1595, Dec. 2001.

[Turk and Pentland (1991b)] M. Turk and A. Pentland. Eigenfaces for recognition. Journal of Cognitive Science Neouroscience, Volume 3(1), pp. 71-86, 1991.

[Turk and Pentland (1991a)] M. Turk and A. Pentland. "Face recognition using eigenfaces," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Maui, Hawaii, 1991.

[Viisage (2002)] **Viisage Face Recognition Technology.** Online: http://www.viisage.com/technology.htm.

[VISMOD (2002)] An Automatic System for Detection, Recognition & Coding of Faces, MIT Vision and Modelling Laboratory. Online: http://whitechapel.media.mit.edu/vismod/demos/facerec/system.html.

[Wiskott et al (1995)] L. Wiskott, J.M. Fellous, N. Krüger, and C. von der Malsburg. **Face Recognition and Gender Determination.** Proc. *Int'l Workshop on Automatic Face- and Gesture- Recognition, IWAFGR'95,* Zurich, June 26-28, ed. Martin Bichsel, publ. MultiMedia Laboratory, University of Zurich, pp. 92-97.

[Wiskott et al (1997)] L. Wiskott, J. Fellous, N. Kruger and C. von der Malsburg. **Face recognition by Elastic Bunch graph matching**. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 19, No. 7, July 1997.

[Wiskott et al (1999)] L. Wiskott, J. Fellous, N. Kruger and C. von der Malsburg. **Face recognition by Elastic Bunch graph matching**. In L.C. Jain, U. Halici, I. Hayashi, S.B. Lee and S. Tsutsui, *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, CRC Press,1999

[Wiskott (2003)] L. Wiskott. **Face-Recognition Bibliography**. Online: http://www.cnl.salk.edu/~wiskott/Bibliographies/FaceRecognition.html .

[Yambor et al (2000)] W. S. Yambor, B. A. Draper and J. R. Beveridge, "Analyzing PCA-based Face Recognition Algorithms: Eigenvector Selection and Distance Measures", 2nd Workshop on Empirical Evaluation in Computer Vision, Dublin, Ireland, July 1, 2000.

[Yang et al (2000)] J. Yang, H. Yu, W. Kunz. "An Efficient LDA Algorithm for Face Recognition", ICARCV 2000, Singapore.

[Yang et al (2002)] X. Yang, Y. Xing and A. Nguyen. **A Web-Based Face Recognition System**

	Using Mobile Agent Technology , Proceedings of the Eighth Australian World Wide Web Conference 2002, Queensland, Australia.
[Yoshino et al (2001)]	M. Yoshino, H. Matsuda, S. Kubota, K. Imaizumi and S. Miyasaka, "Computer-Assisted Facial Image Identification System", *Forensic Science Communications,* January 2001, Volume 3, NO. 1. Available: http://www.fbi.gov/hq/lab/fsc/backissu/jan2001/yoshino.htm.
[Zhang (2000)]	D.D. Zhang, **Automated Biometrics Technologies and Systems**, Kluwer Academic Publishers, 2000.
[Zhao and Chellapa (2002)]	W. Zhao and R. Chellappa, "Image Based Face Recognition, Issues and Methods", Image *Recognition and Classification,* Ed. B. Javidi, Mercel Dekker, pp. 375-402, 2002.
[Zhao et al (1998)]	W. Zhao, A. Krishnaswamy, R. Chellappa, D. Swets, and J. Weng. "Discriminant Analysis of Principal Components for Face Recognition," Face *Recognition: From Theory to Applications*, Eds. H. Wechsler, P.J. Phillips, V. Bruce, F.F. Soulie and T.S. Huang, Springer-Verlag, pp. 73-85, 1998.
[Zhao et al (2000)]	W. Zhao, R. Chellappa, A. Rosenfeld, and J. Phillips, "Face Recognition: A Literature Survey," Technical Report, CS-TR4167, Univ. of Maryland, 2000.

Appendices

Appendix A: An image-recogniser web service

This web service serves as an experimentation module where all our experiments are carried out. It is the first service that we implemented and built incrementally as our experimentation needs dictated.

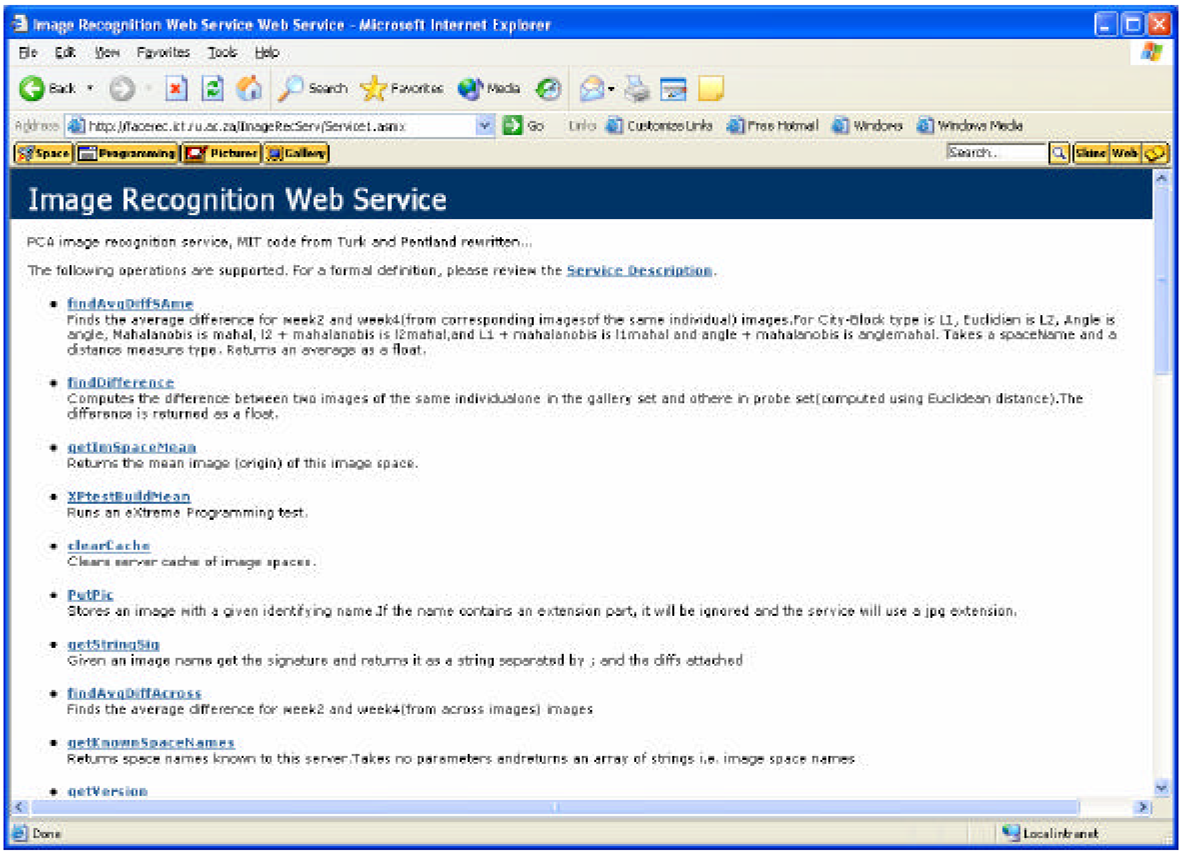


Image Recognition Web Service Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://facerec.ict.ru.ac.za/ImageRecServ/Service1.asmx
Go Links Customize Links Free Hotmail Windows Windows Media
Space Programming Pictures Gallery Search.. Skins Web
Done
Local intranet

Image Recognition Web Service

PCA image recognition service, MIT code from Turk and Pentland rewritten...

The following operations are supported. For a formal definition, please review the Service Description.

- findAvgDiffSAme
  Finds the average difference for week2 and week4(from corresponding imagesof the same individual) images.For City-Block type is L1, Euclidian is L2, Angle is angle, Mahalanobis is mahal, l2 + mahalanobis is l2mahal,and L1 + mahalanobis is l1mahal and angle + mahalanobis is anglemahal. Takes a spaceName and a distance measure type. Returns an average as a float.
- findDifference
  Computes the difference between two images of the same individualone in the gallery set and othere in probe set(computed using Euclidean distance).The difference is returned as a float.
- getImSpaceMean
  Returns the mean image (origin) of this image space.
- XPtestBuildMean
  Runs an eXtreme Programming test.
- clearCache
  Clears server cache of image spaces.
- PutPic
  Stores an image with a given identifying name.If the name contains an extension part, it will be ignored and the service will use a jpg extension.
- getStringSig
  Given an image name get the signature and returns it as a string separated by ; and the diffs attached
- findAvgDiffAcross
  Finds the average difference for week2 and week4(from across images) images
- getKnownSpaceNames
  Returns space names known to this server.Takes no parameters andreturns an array of strings i.e. image space names
- getVersion

A screenshot of the image-recognition service

An overview of some of the methods implemented by the web service

The web service consumes the following web services: the image handler and image signature matcher modules. The following code declares these web services, for their web methods, as accessible to the current application.

```
za.ac.ru.ict.netserv.ImageHandler imh = new
za.ac.ru.ict.netserv.ImageHandler();

za.ac.ru.ict.netserv1.ImageSignatureMatcher ism = new
za.ac.ru.ict.netserv1.ImageSignatureMatcher()
```

Web methods – methods accessible from a web service via the Internet.

`public string getVersion()` Returns a web service version that changes on each build. The returned string gives the version number and the date and time the program was last built. This method takes no parameters and returns the version information as a string.

`public string[] getKnownSpaceNames()` Returns space names known to the server. Takes no parameters and returns an array of strings i.e. image- space names.

`public Matching [] findMatches(string spaceName, string imageNameUrl, int howMany)` Gets closest matching images from an image space. The method takes a spaceName, the imageName and the number of matches to be returned and returns a list of matches from a given face space.

`public float findDifference(string probe Image, string spaceName)` Computes the difference between two images of the same individual, one in the gallery set and the other in the probe set (computed using the Euclidean distance measure). The difference is returned as a float.

`public float findAvgDiffSAme(string spaceName,string DistMeasureType)` Finds the average difference from week2 and week4 (from corresponding images of the same individual) images. For City-Block, the type is L1; Euclidian is L2; Angle is angle; Mahalanobis is mahal; L2 + Mahalanobis is l2mahal, and L1 + Mahalanobis is l1mahal and Angle + Mahalanobis is anglemahal. Takes a spaceName and a distance measure type.

public string clearCache() Clears server cache of image spaces.

public string createImageSpace(string spaceName,string [] imageNames) Creates a new image space, with eigenfaces not explicitly specified, i.e. a space of size n-1 where n is the total number of images in the TS.

public string createImageSpacewithNeigF(string fullpath,string spaceName,int numEigenfaces) This method generates an image space with a given number of eigenfaces and the images path is supplied by the user.

public byte [] getImSpaceMean(string spaceName) Returns the mean image (origin) of this image space as an array of bytes.

public byte [] [] getBaseImages(string spaceName, int numWanted) Returns the most significant base images defining this image space i.e. eigenfaces.

public string storeBaseImages(string spaceName,int numWanted) Stores base images (eigenfaces) in a temporary folder in the local c:drive.

public byte [] [] get3Images(string spaceName, string picName) Returns three images: the original projected onto image space, the reconstructed image from image space, and the difference image computed by subtracting the reconstructed image from the original image.

public string PutPic(string Picname, byte[] Picture) Stores an image with a given identifying name. If the name contains an extension part, it will be ignored and the web method will use a .jpg extension.

public void EnrolSignature (string spaceName, string existingImageName) Enrols a given feature-signature into a given spaceName. This is not persistent across different sessions. You will need to save individuals to persist this info.

public int saveKnownIndividuals(string spaceName) This methods saves known individuals (in this image space) to a file on the hard drive of the server. The method returns the number of individuals known to the space. This method was implemented before the introduction of an image-handler web service i.e. when all the image information was kept on the local machine.

`public string  computeRes(string spaceName, string MatchType, int TopNmatches, int spaceSize)` Generates the recognition results using gallery and probe images registered against a given face space. Matching results are returned as a string.

`public string getStringSig(string imageNameUrl,string spaceName)` Given an image name, gets the signature and returns it as a string separated by ";" and with the DFFS attached.

`public string [] ComputeAvgQuality(string filePathInterDiff,string filePathIntraDiff)` Computes a similarity measure by dividing an inter-difference similarity by an intra-difference similarity measure. The Inter and Intra differences from different images are stored in text files to avoid the intensive re-computation of these each time they are needed.

`public Feature getFeature(string spaceName, string picName)` Computes the features (signature) of this picture in this image space. Returns a Feature signature which is a signature with DFFS and the number of elements in a signature (image-space size).

`public Feature [] getFeatures(string spaceName, string [] picNames)` Computes the features for these pictures in this image space. Returns a list of Feature signatures.

`public string ClassifyUsingDFFS(string imagesPath,string spaceNameMales,string spaceNameFemales)` Uses DFFS to classify images according to gender.

Appendix B: the image-handler web service

This service handles the processing, retrieval and storage of image information. As part of our experimentation framework we decided to pre-compute image signatures and store them in an SQL database. This web service acts as an interface to this database. It consumes the image-recognition web service i.e. signatures are computed by the image-recognition web service and this service accepts them and stores them in the database. The same applies on matching: this web service provides only the requested signatures to the image-matcher web service.

Some web methods defined by image handler web service

`public int compAndStoreGallarySigs(string spaceName)` Computes and stores gallery image signatures in a gallery-image signatures table. The signatures are passed to this method from the image recognition web service. They are computed against an image space registered with the image recognition server.

`public int compAndStoreProbeSigs(string spaceName)` Computes and stores probe image signatures in a probe-set table. The signatures are passed to this method from the image recognition web service. They are computed against an image space registered with the image recognition server.

`public string getProbeSig(string imageName,string spaceName)` Returns a string signature for a given image against a given image space from the probe-set table.

`public string getGallarySig(string imageName,string spaceName)`Returns a string signature for a given image against a given image space from the gallery image-set table

`public string []ReturnAllweek2ImgNames()`Returns all the image names in the database, gallery images from week2.

`public string [] ReturnAllweek4ImgNames()` Returns all the image names in the database, probe images from week4.

`public string MakeGroup( string[] imageNames, string groupName)` Makes a grouping, i.e. tells which images belong to which group. The group information is stored in a groups table.

`public string RemoveGroup(string GroupName)` Removes all image names belonging to a specific group and a group name.

`public string [] RetriveGroup (string GroupName)` Retrieves image names associated with a group.

`public int GetNoOrecordsfromGallary()` Retrieves the number of existing records in the gallery database.

`public int GetNoOrecordsfromProbe()`Retrieves the number of existing records in the probe database.

`public string [] RetNamesfromGender(string gender, string type)` Returns all male or female names for a specified image type. An image type can be a thumbnail or a colour image etc.

`public int ReturnTotalNoGender(string gender, string type)` Counts the number of male or female images belonging to a type.

`public byte [][]ReturnGivenImages(string []imageNames, string Type)` Returns a list of images given the image names and the type.

Web methods that were implemented to facilitate the Science Festival demo

`public string StoreScifestImg(string picName, byte[] imageContent)`
Stores an image with a given identifying name. The client application running on a remote machine with a web camera passes the image and a computer-generated name to this web method which stores the image name and content in a database.

`public string StoreScifestSig(string picName,string picSig)` Stores a signature in a signature database table

`public string GetlastScifestSig()` Returns a the last string signature to be stored for a given image name from the Science Festival signatures table.

`public string []ReturnScifestImgNames()` Gets all the registered image names from the Science Festival table.

`public string getScifestSig(string imageName)` Returns a string signature for a given image from Science Festival table.

`public string [] getlastSigs(int startPos)` Retrieves image signatures given a starting position e.g. one can ask for the last ten signatures.

`public string [] getlastNames(int startPos)` Retrieves image names from the Science Festival database given a starting position.

`public byte [] getScifestImg(string imageName)` Returns an image for a given image name from Science Festival images table.

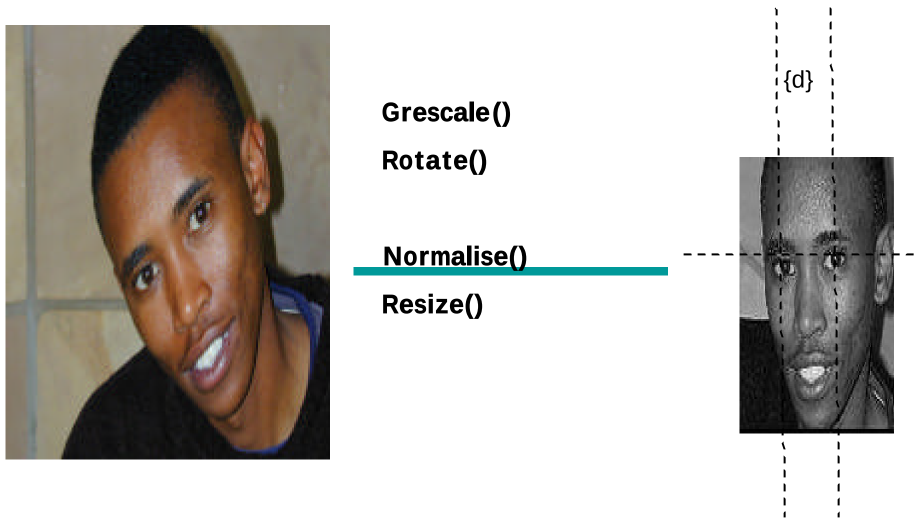
`public string ClearScifestImages()` Cleans the Science Festival images database by deleting all the images and associated signatures from the database.

`public int getScifesTableSize()`Retrieves the number of records in the Science Festival images database table.

`public int getScifesSigsTableSize()` Retrieves the number of records in the Signatures Science Festival signatures table.

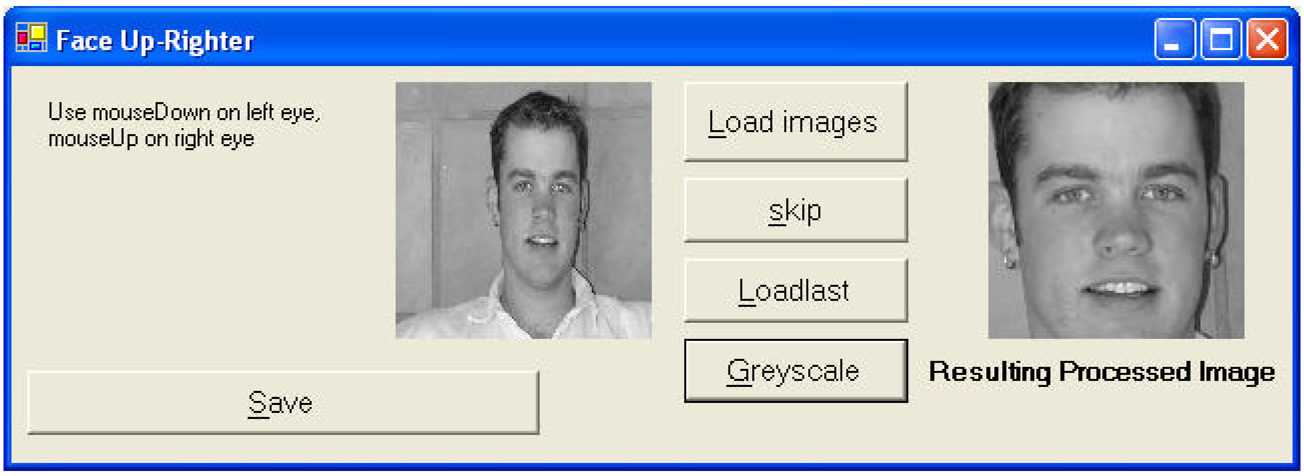
Appendix C: face-image preprocessing

The overall goal of our preprocessing software (face-uprighter) is to convert a colour image into a greyscale image, rotate the image using eye positions, put the eyes at a fixed position, normalising the whole image set.



An illustration of our preprocessing feature

Interface Screen Shot


Face Up-Righter

Use mouseDown on left eye, mouseUp on right eye

Screen shot of "Face –upritghter"

Code developed - Face uprighter implementations

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Drawing.Drawing2D;
using System.IO;


namespace faceUprighter
{
  /// <summary>
  /// Face Uprighter scales and positions eyes at a fixed position
  /// this is achieved by clicking and dragging the mouse from the left eye to
the right eye.
  /// </summary>
  public class Form1 : System.Windows.Forms.Form
  {
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Button cmdRefresh;
    private System.Windows.Forms.PictureBox pb2;
    private System.Windows.Forms.OpenFileDialog openFileDialog1;
    private System.Windows.Forms.OpenFileDialog openFileDialog2;
    private System.Windows.Forms.Button button2;
    private System.Windows.Forms.Button button3;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.PictureBox pb1;
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.PictureBox pb3;
    private System.Windows.Forms.Button button4;
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.Container components = null;

    public Form1()
    {

      InitializeComponent();
    }

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
      if( disposing )
      {
        if (components != null)
        {
          components.Dispose();
        }
      }
      base.Dispose( disposing );
    }

    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
      Application.Run(new Form1());
    }
  /// <summary>
  ///computes degrees by taking a double and computes a degree representation of
it
  /// </summary>

    private double toDegrees(double radians)
    {
      return( (radians / (2.0 * Math.PI))*360.0);
    }


    void showProgress(Bitmap bm, Point p, string msg, float xstretch, float
ystretch)
    {
      return;
      pb3.Image = bm;
      pb3.Refresh();
      Graphics g = pb3.CreateGraphics();
```

```
g.DrawLine(System.Drawing.Pens.Red,p.X/xstretch,p.Y/ystretch,(p.X+40)/xstretch,p
.Y/ystretch);
      if (msg.Length>0) MessageBox.Show(msg);
    }

    private Bitmap normalize(Image original, Point leftEye, Point rightEye)
      // Given a source image and two points (i.e. the person's eyes)
      // rotate the image to get the eyes horizontal, scale to get them exactly
60      //pixels apart,
      // and crop to a 128x128 image so that the left eye is at position (30,30)
    {

      float xstretch=1,  ystretch=1;

      if (pb3.SizeMode == PictureBoxSizeMode.StretchImage)
      {  // if the image is stretched, the eye coordinates are not the same as
bitmap //coords....
        //MessageBox.Show(pb3.Size.ToString() + "   " +
pb3.Image.Size.ToString());
        xstretch = (float) pb3.Image.Size.Width / (float) pb3.Size.Width;
        ystretch = (float) pb3.Image.Size.Height / (float) pb3.Size.Height;
        leftEye.X = (int) ((float)leftEye.X * xstretch);
        leftEye.Y = (int) ((float)leftEye.Y * ystretch);
        rightEye.Y = (int) ((float)rightEye.Y * ystretch);
        rightEye.X = (int) ((float)rightEye.X * xstretch);
      }

      Point [] theEyes = {leftEye, rightEye};
     // This creates copies of these two points.
     // leftEye and rightEye will always be relative to the original image,
untransformed,
      // but this array will be transformed as we fiddle with the image, so we
know where
      // the eyes are in the new bitmaps at any time...

      int x1 = leftEye.X,  y1=leftEye.Y,   x2=rightEye.X,  y2=rightEye.Y;
      if (x1==x2) { return (Bitmap)original; }


      // Compute the rotation matrix, and rotate the image about the
      // left eye (by that we mean the leftmost eye in the picture!)
      // so that the eyes are lined up horizontally.

      int angle = Convert.ToInt32(toDegrees(Math.Atan2(y2-y1,x2-x1)));
      Matrix myMatrix = new Matrix();
      myMatrix.RotateAt(-angle, leftEye);
      myMatrix.TransformPoints(theEyes);

      Bitmap bm1 = new Bitmap(original);
      Graphics g = Graphics.FromImage(bm1);
      g.Transform = myMatrix;

      Rectangle where = new Rectangle(0,0,bm1.Width,bm1.Height);

      g.DrawImage(original,where,where,GraphicsUnit.Pixel);



      // Having rotated the image, scale it to get the eyes a fixed distance
apart...
      float scale = 40.0f / (float) (theEyes[1].X-theEyes[0].X);

      showProgress(bm1, theEyes[0], "Now will scale by " + scale.ToString(),
xstretch, ystretch);

      myMatrix = new Matrix();
      myMatrix.Scale(scale, scale, System.Drawing.Drawing2D.MatrixOrder.Append);
      myMatrix.TransformPoints(theEyes);

      int newWidth = Convert.ToInt32(((float)bm1.Width)*scale);
      int newHeight = Convert.ToInt32(((float)bm1.Width)*scale);
      Bitmap bm2 = new Bitmap(newWidth, newHeight);
      g = Graphics.FromImage(bm2);
      g.Transform = myMatrix;
      g.DrawImage(bm1, where, where, GraphicsUnit.Pixel);
      showProgress(bm2, theEyes[0], "Now will clip to 128x128", xstretch,
ystretch);


      // Now cut out the head  ....
      Bitmap bm3 = new Bitmap(128,128);
      g = Graphics.FromImage(bm3);
      g.DrawImage(bm2,new Rectangle(0,0,128,128),
        new Rectangle(theEyes[0].X-40,theEyes[0].Y-50,128,128),
        GraphicsUnit.Pixel);
      showProgress(bm3,new Point(40,50),"", xstretch, ystretch);
      return bm3;
    }

    Point leftEye, rightEye;
    private void pb3_MouseDown(object sender,
System.Windows.Forms.MouseEventArgs e)
    {
      leftEye = new Point(e.X,e.Y);
    }

    string [] fileEntries;
    int pos = -1;
    string path;

    private void pb3_MouseUp(object sender, System.Windows.Forms.MouseEventArgs
e)
    {
      rightEye = new Point(e.X, e.Y);
      pb2.Image = normalize(pb3.Image, leftEye, rightEye);
    }

    private void Refresh_Click(object sender, System.EventArgs e)
    {
      path = "c:\\junk\\";
      fileEntries = Directory.GetFiles(path,"*.JPG");
      pos = -1;
      loadNext();
    }


    // Process all files in the directory passed in, and recurse on any
//subdirectories
    // that are found to process the files they contain
    public void loadNext()
    {


      if (pos < fileEntries.Length-1)
      { pos++;
        pb3.Image = new Bitmap(fileEntries[pos]);
      }
      else
      {
        MessageBox.Show("All files done");
      }
    }
/* The following code loads the last image in the list of images from a
specified   *directory   */
        public void loadlast()
        {


              if (pos < fileEntries.Length-1&& pos != 0)
              {
                          pos--;
                    pb3.Image = new Bitmap(fileEntries[pos]);

              }
              else
              {
                    MessageBox.Show("All files done");
              }
        }
        /* Loads the next image to the picture box from the images list
        */
    private void button1_Click(object sender, System.EventArgs e)
    {
       loadNext();
    }
/* The following method converts a colour image from a picture box into a
greyscale image  *and store it back to the  * same picture box. */
    public void Greyscale()
        {
              Bitmap bm = (Bitmap) pb3.Image;
              for (int y=0; y < bm.Height; y++)
              {
                    for (int x=0; x < bm.Width; x++)
                    {
                          Color c = bm.GetPixel(x,y);
                          Color g = Color.FromArgb(c.G, c.G, c.G);
                          bm.SetPixel(x,y,g);
                    }
              }
```

```
              pb3.Refresh();
        }
        /*Stores the processed image at a given path in a local directory
        */
      private void button2_Click(object sender, System.EventArgs e)
        {
            string newName = "c:\\junk\\me\\" +
fileEntries[pos].Substring(path.Length);
            Bitmap b = (Bitmap)pb2.Image;

            b.Save(newName,System.Drawing.Imaging.ImageFormat.Jpeg);
            loadNext();
        }
      /* Loads the last image to the picture box.
       */
        private void button3_Click(object sender, System.EventArgs e)
        {
            loadlast();
        }

/*The following code converts a colour image into a greyscale image. This is
done on a
* picture box i.e. not saved in memory. Saving is done only after positioning
eyes at  *fixed position */

        private void button4_Click(object sender, System.EventArgs e)
        {
            Greyscale();
        }



        }

  }
```

Appendix D: DRUBIS – third-party-support web services

Signature-matcher service

The signature-matcher web service takes two signatures and decides how close they are to each other. We have implemented quite a number of distance measures as part of this service. Some of the web methods implemented by this service are listed next.

`public float [] processStringSig(string StringSig)` Takes a string signature and generates a signature as an array of floats and removes the first element (DFFS in our case) since we do not use it for recognition. This method prepares a string signature to be used by a matcher which takes signatures as an array of floats.

`public float EuclidianMatchL2(float [] gallarySig, float [] probeSig)` Computes closeness given a gallery signature and a probe using Euclidian distance measure. Returns a closeness distance as a float.

`public float CityBlockL1(float [] gallarySig, float [] probeSig)` Computes closeness, given a gallery and probe signatures using L1 (City-Block) norm.

`public float L2AndMahalanobis(float [] gallarySig, float [] probeSig, float []theZed)` Computes closeness given a gallery and probe signatures using Euclidian distance and Mahalanobis distance; theZed array contains the eigenvalues from the face space that are needed for the Mahalanobis distance measure. Uses the Moon and Phillips's (1998) simplified formula.

`public float AngleBtnSigsMatch(float [] gallarySig, float [] probeSig)` Computes closeness given a gallery and probe signatures using the angle between signature vectors

`public float AngleBtnSigsAndMahalanobis(float [] gallarySig, float [] probeSig, float [] theZed)` Computes closeness given a gallery and a probe signature using the cosine angle between signature vectors and a variance vector (eigenvalues from space) for the Mahalanobis measure.

`public float Mahalanobis(float [] gallarySig, float [] probeSig, float [] theZed)`
Computes closeness given a gallery and probe signatures using the Mahalanobis distance. A variance vector (eigenvalues from space) theZed are also passed as a parameter.

`public float MahalL1and2andAngle(float [] gallarySig, float [] probeSig, float [] theZed)` Computes closeness given a gallery and a probe signatures using the Mahalanobis,L1,L2 and Angle distances. Adds all the closeness measures to a single sum.

`public float L2andAngle(float [] gallarySig, float [] probeSig, float [] theZed)`Computes closeness given a gallery and a probe signature using the L2(Euclidean) and Angle distances. Adds both closeness measures to a single sum.

`public float L1andL2(float [] gallarySig, float [] probeSig)`
Computes closeness, given a gallery and a probe signature using the L1 and L2 Distance. Adds both closeness measures to a single sum.

`public float L1AndMahalanobis(float [] gallarySig, float [] probeSig,float [] theZed)` Computes closeness given a gallery and probe signature using L1 norm and Mahalanobis distance measures.

Image signature generator service

This web service takes only the signature generation component of the image-recognition service discussed above. It is designed with the aim of offering support to third-party users and (or) applications. ASP.Net is used to improve the interface and enable face space and image files to be uploaded to the server.

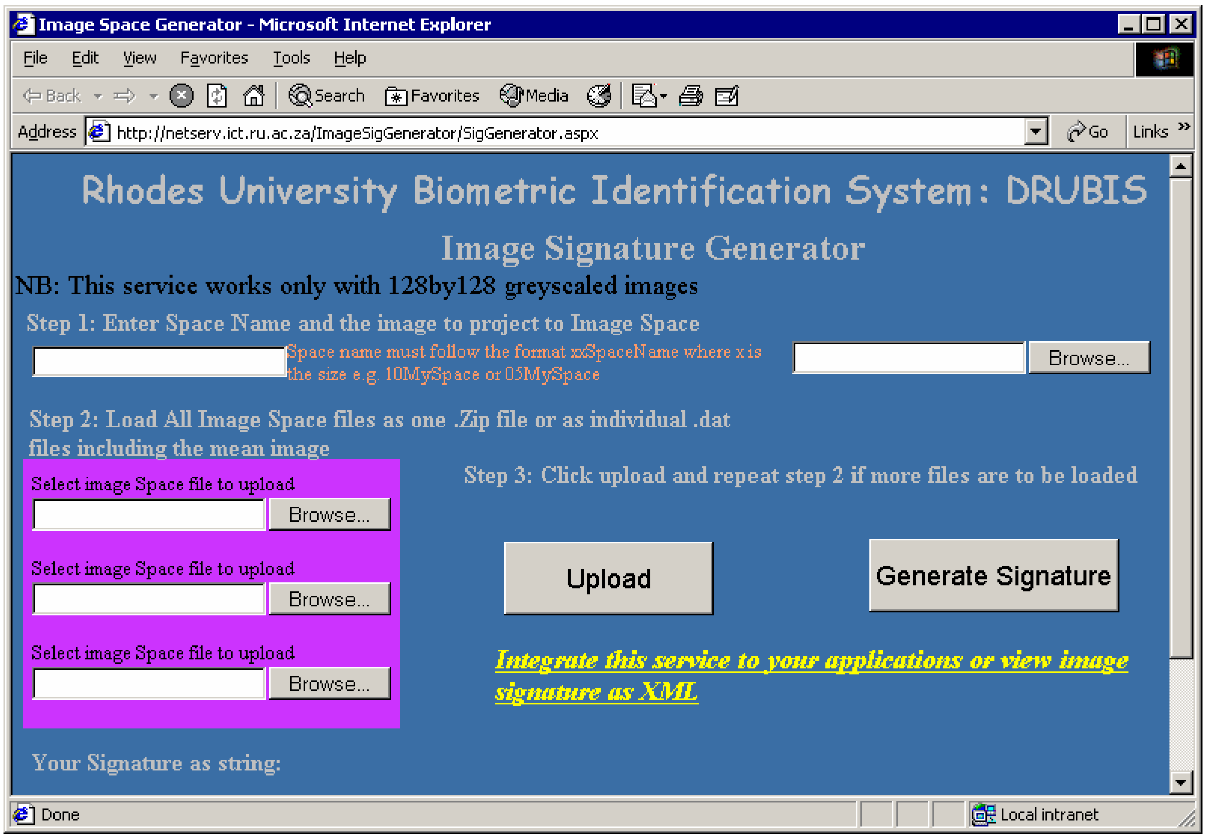


Image Space Generator - Microsoft Internet Explorer

http://netserv.ict.ru.ac.za/ImageSigGenerator/SigGenerator.aspx

Rhodes University Biometric Identification System: DRUBIS

Image Signature Generator

NB: This service works only with 128by128 greyscaled images

Step 1: Enter Space Name and the image to project to Image Space

Space name must follow the format xxSpaceName where x is the size e.g. 10MySpace or 05MySpace

Step 2: Load All Image Space files as one .Zip file or as individual .dat files including the mean image

Select image Space file to upload

Select image Space file to upload

Select image Space file to upload

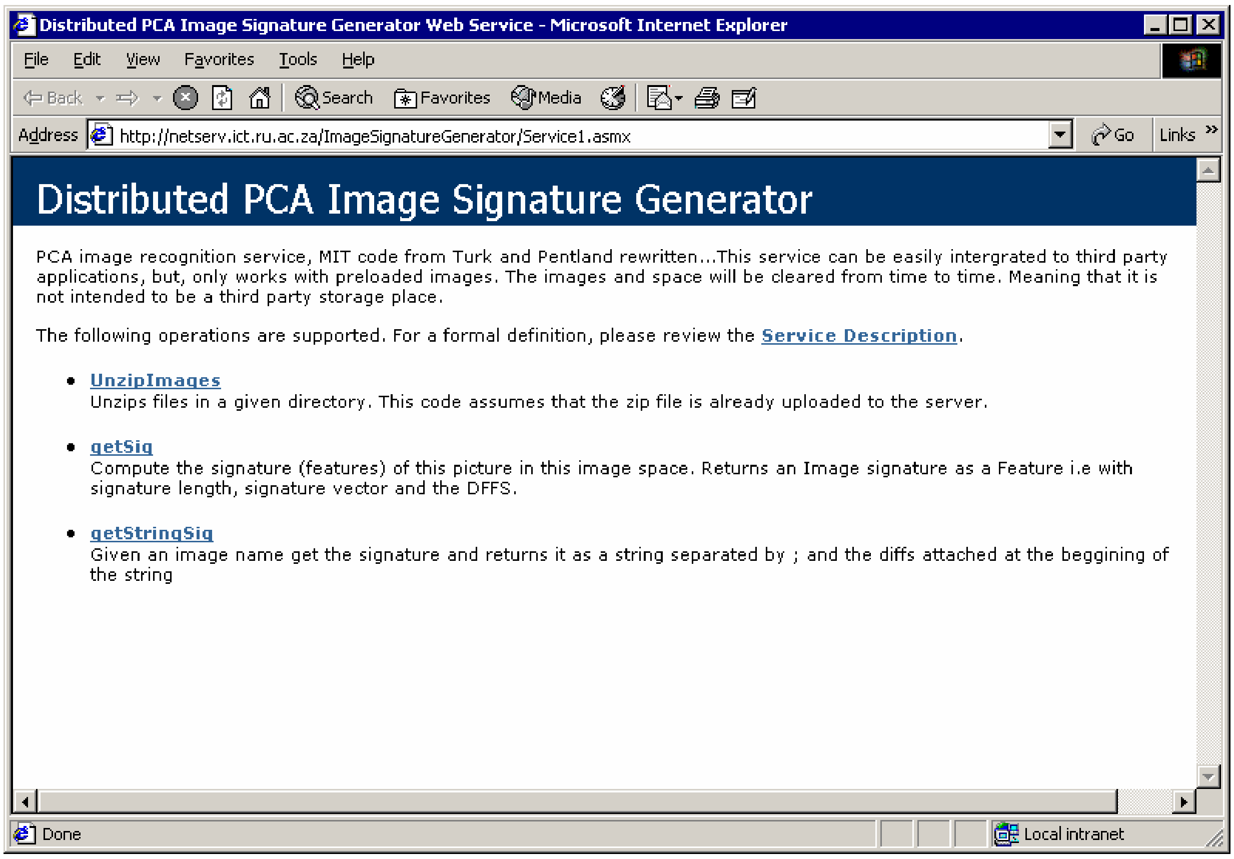
Step 3: Click upload and repeat step 2 if more files are to be loaded

Integrate this service to your applications or view image signature as XML

Your Signature as string:

A screenshot of the image-signature generator ASP.Net interface for uploading files

Above, image space files can be loaded as individual files or they can be zipped (using SharpZib open-source library from Krueger (2001)) into a single folder and uploaded at once. After the files have been uploaded, the “Generate Signature” button is pressed and the web service in the background unzips the files and generates a signature. The user can then copy a signature shown on the web page or can access it directly as XML data by clicking on the relevant link.


Distributed PCA Image Signature Generator Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://netserv.ict.ru.ac.za/ImageSignatureGenerator/Service1.asmx
Go Links
Done
Local intranet

Distributed PCA Image Signature Generator

PCA image recognition service, MIT code from Turk and Pentland rewritten...This service can be easily intergrated to third party applications, but, only works with preloaded images. The images and space will be cleared from time to time. Meaning that it is not intended to be a third party storage place.

The following operations are supported. For a formal definition, please review the **Service Description**.

- **UnzipImages**
  Unzips files in a given directory. This code assumes that the zip file is already uploaded to the server.
- **getSig**
  Compute the signature (features) of this picture in this image space. Returns an Image signature as a Feature i.e with signature length, signature vector and the DFFS.
- **getStringSig**
  Given an image name get the signature and returns it as a string separated by ; and the diffs attached at the beggining of the string

A screenshot of the image-signature-generator web service with the web methods that it implements

Image-space generator service

The image-space generator web service generates an image face space. This service is consumed by an ASP.Net interface that allows the user to upload and download image space files. The web service web methods are clearly visible in the next screen shot.

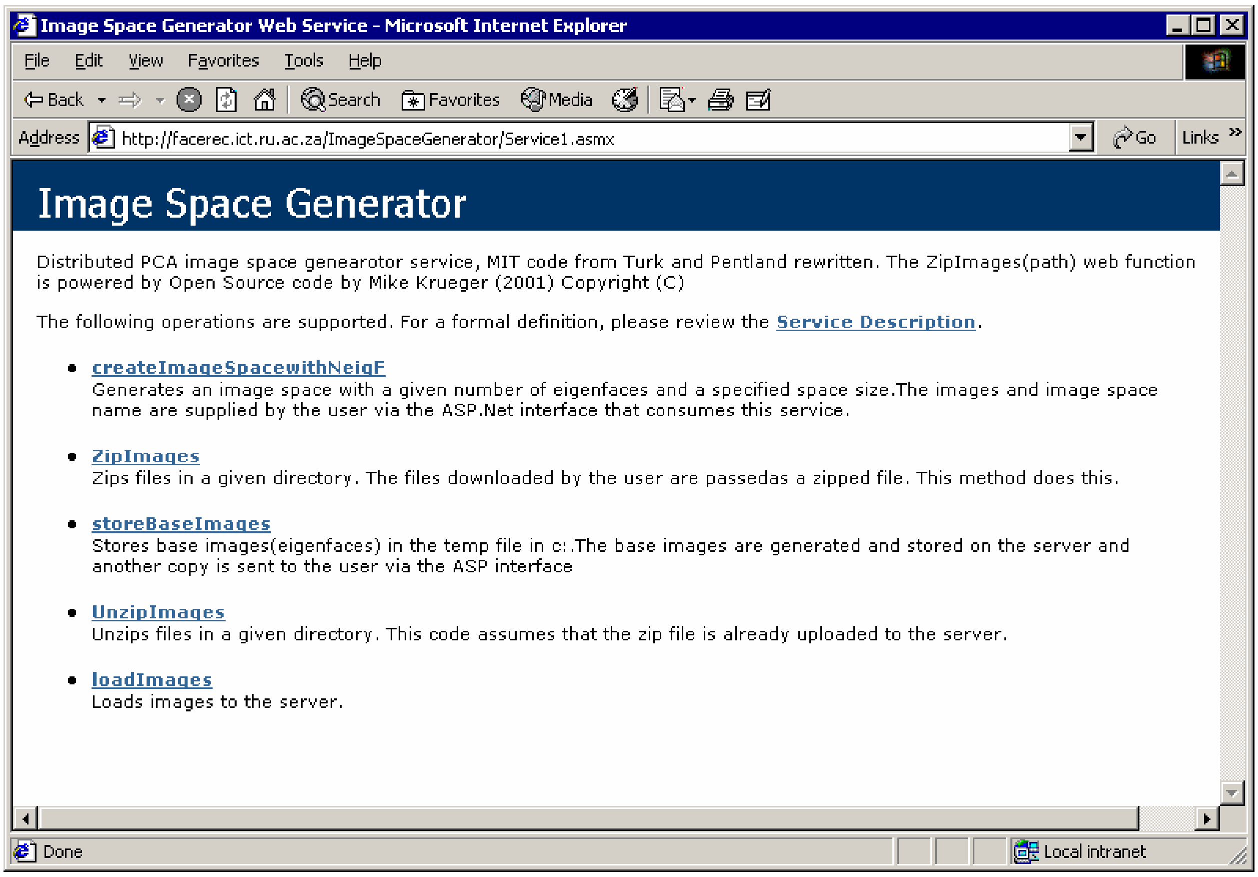


Image Space Generator Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://facerec.ict.ru.ac.za/ImageSpaceGenerator/Service1.asmx
Go Links
Done
Local intranet

Image Space Generator

Distributed PCA image space genearotor service, MIT code from Turk and Pentland rewritten. The ZipImages(path) web function is powered by Open Source code by Mike Krueger (2001) Copyright (C)

The following operations are supported. For a formal definition, please review the **Service Description**.

- **createImageSpacewithNeigF**
  Generates an image space with a given number of eigenfaces and a specified space size.The images and image space name are supplied by the user via the ASP.Net interface that consumes this service.
- **ZipImages**
  Zips files in a given directory. The files downloaded by the user are passedas a zipped file. This method does this.
- **storeBaseImages**
  Stores base images(eigenfaces) in the temp file in c:.The base images are generated and stored on the server and another copy is sent to the user via the ASP interface
- **UnzipImages**
  Unzips files in a given directory. This code assumes that the zip file is already uploaded to the server.
- **loadImages**
  Loads images to the server.

Screenshot of the image generator web service.