İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

NULL SPACE APPROACH OF FISHER DISCRIMINANT ANALYSIS FOR FACE RECOGNITION

M.Sc. Thesis by

Kürşat AĞIR, B.S.

Department : Computer Engineering

Programme : Computer Engineering

MAY 2006

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

NULL SPACE APPROACH OF FISHER DISCRIMINANT ANALYSIS FOR FACE RECOGNITION

M.Sc. Thesis by

Kürşat AĞIR, B.S.

(504021521)

Date of submission : 12 May 2006

Date of defence examination: 15 June 2006

Supervisor (Chairman): Prof. Dr. Muhittin GÖKMEN

Members of the Examining Committee: Assos. Prof. M. Ertuğrul ÇELEBİ

Asst. Prof. Zehra ÇATALTEPE

MAY 2006

PREFACE

First and foremost, I am grateful to my thesis supervisor Prof. Dr. Muhittin GÖKMEN for his continuous guidance, patience and support.

I would also like to thank educator Binnur KURT for his patience to answering my questions.

Finally, and above all, I would like to thank my parents, whose love and support I could always count on. This thesis is dedicated to them.

May, 2006 **Kürşat AĞIR**

CONTENTS

ABBREVIATIONS

PCA	**:** Principle Component Analysis
LDA	**:** Linear Discriminant Analysis
FLDA	**:** Fisher Linear Discriminant Analysis
DLDA	**:** Direct LDA
NLDA	**:** Null Space-based LDA
HEFLDA	**:** Histogram Equalization applied FLDA
HEDLDA	**:** Histogram Equalization applied DLDA
HENLDA	**:** Histogram Equalization applied NLDA
KFDA	**:** Kernel Fisher Discriminant Analysis
SSSP	**:** Small Sample Size Problem
HCI	**:** Human Computer Interface
AFIS	**:** Automated Finger Imaging Systems
CCTV	**:** Closed Circuit Television
COV	**:** Covariance
CORR	**:** Correlation
MAH	**:** Mahalanobis Distance

TABLE LIST

FIGURE LIST

SYMBOL LIST

$\mathbf{S_w}$	**:** Within class scatter matrix
$\mathbf{S_b}$	**:** Between class scatter matrix
$\mathbf{N}$	**:** Number of all samples
$\mathbf{N_j}$	**:** The number of samples in class C_i
$\mathbf{C}$	**:** Number of all classes
$\mathbf{C_i}$	**:** ith class
$\mathbf{m_i}$	**:** The mean of the samples in the class C_i
$\mathbf{m}$	**:** The overall mean of all samples
$\mathbf{S_t}$	**:** The covariance matrix
$\mathbf{\Phi_t}$	**:** The data matrix
$\mathbf{W}$	**:** Matrix constructed by the eigenvectors of $S_w^{-1} S_b$
$\mathbf{I}$	**:** Unit matrix
$\mathbf{\Lambda}$	**:** Diagonal matrix sorted in decreasing order
$\mathbf{D_w}$	**:** Diagonal matrix sorted in increasing order
$\mathbf{N(X)}$	**:** Null space of matrix X
$\mathbf{P}$	**:** The projection matrix
$\mathbf{V}$	**:** Eigenvector matrix
$\mathbf{Y}$	**:** The first c-1 columns of V
$\overline{X}$	**:** The mean of the set X
$\mathbf{s}$	**:** Standart Deviation
$\mathbf{s^2}$	**:** Variance
$\mathbf{L_1}$	**:** L_1 Norm
$\mathbf{L_2}$	**:** L_2 Norm
$\mathbf{\mu A}$	**:** The mean of A
λ_i	**:** The i-th eigenvalue
$\mathbf{e_i}$	**:** The i-th eigenvector
$\mathbf{y_i}$	**:** The eigenvector of S_b

NULL SPACE APPROACH OF FISHER DISCRIMINANT ANALYSIS FOR FACE RECOGNITION

SUMMARY

A wide variety of systems require reliable personal recognition schemes to either confirm or determine the identity of an individual requesting their services. The purpose of such schemes is to ensure that the rendered services are accessed only by a legitimate user, and not anyone else. Examples of such applications include secure access to buildings, computer systems, laptops, cellular phones and ATMs. Biometric recognition, or simply biometrics, refers to the automatic recognition of individuals based on their physiological and/or behavioral characteristics. Face recognition from images is a sub-area of the general object recognition problem. Identifying an individual from his or her face is one of the most nonintrusive modalities in biometrics. It is of particular interest in a wide variety of applications. PCA is a techique commonly used in dimension reduction in computer vision and particularly in face recognition. PCA techniques, also known as Karhunen-Loeve methods, choose a linear projection that reduces the dimensionality while maximizing the scatter of all projected samples. The null space of the within-class scatter matrix is found to express most discriminative information for the small sample size problem (SSSP). The null space-based LDA takes full advantage of the null space while the other methods remove the null space. It proves to be optimal in performance. From the theoretical analysis, we present the NLDA algorithm and the most suitable situation for NLDA. Our method is simpler than all other null space approaches, it saves the computational cost and maintains the performance simultaneously. Experiments are carried out on different face data sets, different facial expression, different class count and different eigenvalue count to demonstrate the effectiveness of the proposed methods.

Keywords: Face Recognition, PCA, LDA, Null Space, Eigenface, Fisherface, NLDA, DLDA

YÜZ TANIMA İÇİN FISHER DİSKRİMİNANT ANALİZİNE SIFIR UZAYI YAKLAŞIMI

ÖZET

Günümüzde birçok farklı sistem, insanların kendi servislerine erişimlerinde kimlik onaylamak veya belirlemek için güvenilir kimlik tanıma projelerine ihtiyaç duymaktadır. Bu tür projelerdeki asıl amaç sunulan servislere erişimlerin sadece yetkili kullanıcıya verilmesini garanti etmektir. Bu ve buna benzer birkaç uygulama binalara, bilgisayar sistemlerine, dizüstü bilgisayarlara, telefonlara, ATM'lere güvenli erişimlerdir. Biometrik tanıma veya sadece biometrik, insanların fiziksel ve davranışsal özelliklerinin otomatik tanınması anlamına gelmektedir. Bu çalışmada, yüz tanıma için Fisher Diskriminant Analizine sıfır uzay yaklaşımı gerçekleştirilmiştir. Yüz tanıma genel nesne tanıma problemlerinin bir alt alanıdır. Herhangi birini yüzünü baz alarak tanımak biometrik içerisinde yanıltılması güç bir yöntemdir. PCA ise görüntü işleme alanında boyut küçültmede sıkça kullanılan bir yöntemdir. Aynı zamanda Karhunen-Loeve olarak da bilenen bu metot, boyutları küçülten bir lineer izdüşüm seçerek tüm izdüşüm örnekleri arasındaki dağılımı en yüksek dereceye getirir. Sınıfiçi dağılım matrisinin sıfır uzayı küçük örnek boyutu probleminin en diskriminatif bilgisini göstermektedir. Diğer metotlar sıfır uzayını kaldırdığı halde, sıfır uzay tabanlı Lineer Diskriminant Analizi sıfır uzayının tüm avantajlarını kullanmaktadır. Bu yöntem performans için en uygun olduğunu kanıtlamaktadır. Sıfır Uzayı Lineer Diskriminant Analizi algoritması ve bunun için en uygun durum çalışmada gösterilmiştir. Yöntemimiz diğer bütün sıfır uzayı yaklaşımlarından daha basit, işlemsel maliyet ve performans açısından daha uygundur. Deneyler farklı yüz veritabanlarında, farklı yüz ifadeleri kullanılarak, farklı sınıf sayısı ve farklı özvektör sayısı baz alınarak gerçekleştirilmiş ve önerilen metotların etkinlikleri ölçülmüştür.

Anahtar Kelimeler: Yüz tanıma, PCA, LDA, özyüzler, Fisher yüzleri, sıfır uzayı, NLDA, DLDA

1. INTRODUCTION

In recent years face recognition has received substantial attention from researchers in biometrics, pattern recognition, and computer vision communities [1,3,4]. The machine learning and computer graphics communities are also increasingly involved in face recognition. This common interest among researchers working in diverse fields is motivated by our remarkable ability to recognize people and the fact that human activity is a primary concern both in everyday life and in cyberspace [5]. Besides, there are a large number of commercial, security, and forensic applications requiring the use of face recognition technologies. These applications include automated crowd surveillance, access control, mugshot identification (e.g., for issuing driver licenses), face reconstruction, design of human computer interface (HCI), multimedia communication (e.g., generation of synthetic faces), and content-based image database management. A number of commercial face recognition systems have been deployed.

1.1. Definition of Biometrics

A concise definition of biometrics is "the automatic recognition of a person using distinguishing traits." A more expansive definition of biometrics is "any automatically measurable, robust and distinctive physical characteristic or personal trait that can be used to identify an individual or verify the claimed identity of an individual." This definition requires elaboration [2].

Measurable means that the characteristic or trait can be easily presented to a sensor, located by it, and converted into a quantifiable, digital format [5]. This measurability allows for matching to occur in a matter of seconds and makes it an automated process.

The *robustness* of a biometric refers to the extent to which the characteristic or trait is subject to significant changes over time. These changes can occur as a result of

age, injury, illness, occupational use, or chemical exposure [4]. A highly robust biometric does not change significantly over time while a less robust biometric will change. For example, the iris, which changes very little over a person's lifetime, is more robust than one's voice.

Distinctiveness is a measure of the variations or differences in the biometric pattern among the general population [6]. The higher the degree of distinctiveness, the more individual is the identifier. A low degree of distinctiveness indicates a biometric pattern found frequently in the general population [8,9]. The iris and the retina have higher degrees of distinctiveness than hand or finger geometry.

Biometrics are used for human recognition which consists of *identification* and *verification.* The terms differ significantly. With *identification*, the biometric system asks and attempts to answer the question, "Who is X?" In an identification application, the biometric device reads a sample and compares that sample against every record or template in the database [1]. This type of comparison is called a "one-to-many" search (1:N). Depending on how the system is designed, it can make a "best" match, or it can score possible matches, ranking them in order of likelihood. Identification applications are common when the goal is to identify criminals, terrorists, or other "wolves in sheep's clothing," particularly through surveillance.

Verification occurs when the biometric system asks and attempts to answer the question, "Is this X?" after the user claims to be X. In a verification application, the biometric system requires input from the user, at which time the user claims his identity via a password, token, or user name (or any combination of the three). This user input points the system to a template in the database [13]. The system also requires a biometric sample from the user. It then compares the sample to or against the user-defined template. This is called a "one-to-one" search (1:1). The system will either find or fail to find a match between the two. Verification is commonly used for physical or computer access.

1.2. Examples of Biometrics

Biometric technologies may seem exotic, but their use is becoming increasingly common, and in 2001 *MIT Technology Review* named biometrics as one of the "top ten emerging technologies that will change the world." While this briefing focuses on facial recognition, there are many different types of biometrics [11]. Examples include:

Iris Scan

Iris scanning measures the iris pattern in the colored part of the eye, although the iris color has nothing to do with the biometric. Iris patterns are formed randomly. As a result, the iris patterns in a person's left and right eyes are different, and so are the iris patterns of identical twins. Iris scanning can be used quickly for both identification and verification applications because the iris is highly distinctive and robust.

Retinal Scan

Retinal scans measure the blood vessel patterns in the back of the eye. The device involves a light source shined into the eye of a user who must be standing very still within inches of the device. Because users perceive the technology to be somewhat intrusive, retinal scanning has not gained popularity; currently retinal scanning devices are not commercially available.

Facial Recognition

Facial recognition records the spatial geometry of distinguishing features of the face. Different vendors use different methods of facial recognition, however, all focus on measures of key features of the face. Because a person's face can be captured by a camera from some distance away, facial recognition has a clandestine or covert capability (*i.e.* the subject does not necessarily know he has been observed). For this reason, facial recognition has been used in projects to identify card counters or other undesirables in casinos, shoplifters in stores, criminals and terrorists in urban areas.

Speaker / Voice Recognition

Voice or speaker recognition uses vocal characteristics to identify individuals using a pass-phrase. A telephone or microphone can serve as a sensor, which makes it a relatively cheap and easily deployable technology. However, voice recognition can be affected by environmental factors such as background noise. This technology has been the focus of considerable efforts on the part of the telecommunications industry and the U.S. government's intelligence community, which continue to work on improving reliability.

Fingerprint

The fingerprint biometric is an automated digital version of the old ink and paper method used for more than a century for identification, primarily by law enforcement agencies. The biometric device involves users placing their finger on a platen for the print to be electronically read. The minutiae are then extracted by the vendor's algorithm, which also makes a fingerprint pattern analysis. Fingerprint biometrics currently have three main application arenas: large-scale Automated Finger Imaging Systems (AFIS) generally used for law enforcement purposes, fraud prevention in entitlement programs, and physical and computer access.

Hand/Finger Geometry

Hand or finger geometry is an automated measurement of many dimensions of the hand and fingers. Neither of these methods takes actual prints of the palm or fingers. Spatial geometry is examined as the user puts his hand on the sensor's surface and uses guiding poles between the fingers to properly place the hand and initiate the reading. Finger geometry usually measures two or three fingers. Hand geometry is a well-developed technology that has been
field-tested and is easily accepted by users. Because hand and finger geometry have a low degree of distinctiveness, the technology is not well suited for identification applications.

Dynamic Signature Verification

We have long used a written signature as a means to acknowledge our identity. Dynamic signature verification is an automated method of measuring an individual's signature. This technology examines such dynamics as speed, direction, and pressure

of writing; the time that the stylus is in and out of contact with the "paper," the total time taken to make the signature; and where the stylus is raised from and lowered onto the "paper."

Keystroke Dynamics

Keystroke dynamics is an automated method of examining an individual's keystrokes on a keyboard. This technology examines such dynamics as speed and pressure, the total time taken to type particular words, and the time elapsed between hitting certain keys. This technology's algorithms are still being developed to improve robustness and distinctiveness. One potentially useful application that may emerge is computer access, where this biometric could be used to verify the computer user's identity continuously.

1.3. Biometrics are used for Authentication

Authentication may be defined as "providing the right person with the right privileges the right access at the right time. In general, there are three approaches to authentication. In order of least secure and least convenient to most secure and most convenient, they are:

- Something you **have** - card, token, key.
- Something you **know** – PIN, password.
- Something you **are** - a biometric.

Any combination of these approaches further heightens security. Requiring all three for an application provides the highest form of security.

All three authentication mechanisms have drawbacks, so security experts routinely recommend using two separate mechanisms, a process called two-factor authentication. But implementing two-factor authentication requires expensive hardware and infrastructure changes. Therefore, security has most often been left to just a single authentication method.

Passwords are cheap, but most implementations offer little real security. Managing multiple passwords for different systems is a nightmare, requiring users to maintain lists of passwords and systems that are inevitably written down because they can't

remember them. The short answer, talked about for decades but rarely achieved in practice, is the idea of single sign-on.

Using security tokens or smart cards requires more expense, more infrastructure support and specialized hardware. Still, these used to be a lot cheaper than biometric devices and, when used with a PIN or password, offer acceptable levels of security, if not always convenience.

Biometric authentication has been widely regarded as the most foolproof - or at least the hardest to forge or spoof. Since the early 1980s, systems of identification and authentication based on physical characteristics have been available to enterprise IT. These biometric systems were slow, intrusive and expensive, but because they were mainly used for guarding mainframe access or restricting physical entry to relatively few users, they proved workable in some high-security situations. Twenty years later, computers are much faster and cheaper than ever. This, plus new, inexpensive hardware, has renewed interest in biometrics.

1.4. A Brief History of Face Recognition

It has been over a decade since the “Eigenfaces” approach to automatic face recognition, and other appearance-based methods, made an impression on the computer vision research community and helped spur interest in vision systems being used to support biometrics and human-computer interface [1] . Appearance-based approaches to recognition complement feature-or shape-based approaches, and a practical face recognition system should have elements of both [17]. Eigenfaces is not a general approach to recognition, but rather one tool out many to be applied and evaluated int the appropriate context.

It is often observed that the human ability to recognize faces is remarkable [10]. Faces are complex visual stimuli, not easily described by simple shapes or patterns; yet people have the ability to recognize familiar faces at a glance after years of separation. Lest we marvel too much at human performance, it should be noted that the inability to recognize a face is sometimes remarkable as well [5,6,9]. Quite often we strain to see the resemblance between a picture (e.g., a driver’s license photo) and the real person, and sometimes we are greeted in a friendly, familiar manner by someone we do not remember ever seeing before [14]. Although face recognition in humans may be impressive, it is far from perfect. Yet there is something about the

perception of faces that is very fundamental to the human experience. Early in life we learn to associate faces with pleasure, fulfillment, and security. As we get older, the subtleties of facial expression enhance our explicit communication in myriad ways. The face is our primary focus of attention in social intercourse; this can be observed in interaction among animals as well as between humans and animals [2]. The face, more than any other part of the body, communicates identity, emotion, race, and age, and is also quite useful for judging gender, size, and perhaps even character.

The subject of visual processing of human faces has received attention from philosophers and scientists such as Aristotle and Darwin for centuries. The ability of a person to recognize another person (e.g., a mate, a child, or an enemy) is important for many reasons. Recognition is not only visual; it may occur through a variety of sensory modalities, including sound, touch, and even smell. For people, however, the most reliable and accessible modality for recognition is the sense of sight. Using vision, a person may be recognized by one's face, but also by one's clothing, hairstyle, gait, silhouette, hands, etc. People often distinguish animals not by their faces but by characteristic markings on their bodies. Similarly, the human face is not the only, and may not even be the primary, visual characteristic used for person identification. For example, in a home or office setting, the person's face may be used merely in verifying identity, after identity has already been established based on other factors such as clothing, hairstyle, or a distinctive moustache. Indeed, the identification of humans may be viewed as a Bayesian classification system, with prior probabilities on several relevant random variables. For example, a parent is predisposed to recognize his child if, immediately prior to contact, he sees a school bus drive by and then hears yelling and familiar light footsteps [15]. Nevertheless, because faces are so important in human interaction, no other avenue to person identification is as compelling as face recognition.

There has been a good deal of investigation into human face recognition performance, seeking to understand and characterize the representations and processes involved [18]. Face-specific cells (cells that appear to respond selectively to the presence of faces) have been found in monkeys and sheep. Prosopagnosia, the specific inability to recognize faces, has been identified and studied in human patients. There have been many interesting studies in experimental and

developmental psychology that have probed the limits human face recognition, suggesting models and constraints on representation and processing. Nevertheless, it is still the case that a through understanding of how humans (and animals) represent, process, and recognize faces remains a distant goal [20]. Although studies of face recognition in physiology, neurology, and psychology provide insight into the problem of face recognition, they have yet to provide insight into the problem of face recognition, they have yet to provide substantial practical guidance for computer vision systems in this area.

What does it mean to recognize a face? There are several aspects of recognizing human identity and processing facial information that make the problem somewhat ill-defined. As mentioned above, recognition of a person's identity is not necessarily (and perhaps rarely) a function of viewing the person's face in isolation. In addition, face recognition is closely related to face (and head and body) detection, face tracking, and facial expression analysis. Figure 1.a, b, c shows a few typical engineering approaches to the overall problem. In the first example, a face is initially detected, then recognized. In the second example, detection and recognition are performed in tandem; detection is merely a successful recognition. In the third example, facial feature tracking is performed and expression analysis occurs before attempting to recognize the normalized (expressionless) face. There are, of course, many additional variations possible [8].

Just as the human task of face recognition is neither clearly defined nor clearly differentiated from related tasks, automatic face recognition by computers is not a single defined problem. Face recognition systems may be useful in several contexts, for example:

- Given a database of standard face images (e.g., criminal mug shots), determine whether or not a new mug shot is of one of the people in the database.
- In the same situation, determine possible identity when the new image originates from a completely different source (e.g., a surveillance camera at a bank), with different (and probably unknown) imaging conditions.
- Identify the new computer user as one of the registered users in order to allow login access.

- Determine that a face is present in an image, at a particular location and scale, in order to correctly color balance the image, or to compress the image properly.

"Face recognition" and "face identification" describe the same task. That is, given an image of a human face, classify that face as one of the individuals whose identity is already known by the system, or perhaps as an unknown face. " Face detection" means detecting the presence of any face, regardless of identity. "Face location" is specifying the 2D position (and perhaps orientation) of a face in the image [14]. "Face tracking" is updating the (2D or 3D) location of the face. "Facial feature tracking" is updating the (2D or 3D) locations, and perhaps the parameterized descriptions, of individual facial features. "Face pose estimation" is determining the position and orientation of a face. "Facial expression analysis" is computing parametric, and perhaps also symbolic, descriptions of facial deformations.

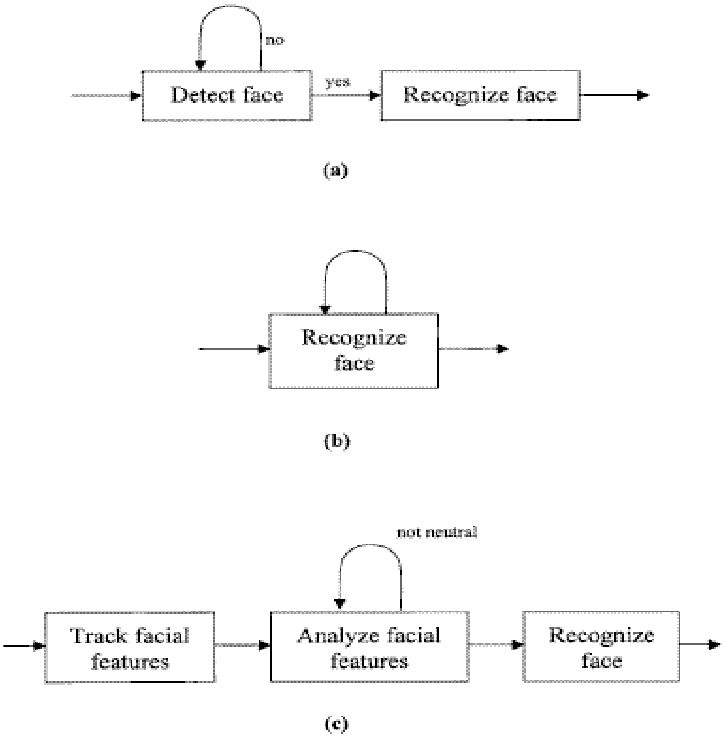

Figure 1.1: Typical approaches to face recognition

1.5. Face Recognition Also Provides a Surveillance Capability

Although the concept of recognizing someone from facial features is intuitive, facial recognition, as a biometric, makes human recognition a more automated, computerized process [7,8]. What sets apart facial recognition from other biometrics is that it can be used for surveillance purposes. For example, public safety authorities want to locate certain individuals such as wanted criminals, suspected terrorists, and missing children. Facial recognition may have the potential to help the authorities with this mission.

Facial recognition offers several advantages. The system captures faces of people in public areas, which minimizes legal concerns for reasons explained below. Moreover, since faces can be captured from some distance away, facial recognition can be done without any physical contact. This feature also gives facial recognition a clandestine or covert capability.

For any biometric system to operate, it must have records in its database against which it can search for matches [12]. Facial recognition is able to leverage existing databases in many cases. For example, there are high quality mugshots of criminals readily available to law enforcement. Similarly, facial recognition is often able to leverage existing surveillance systems such as surveillance cameras or closed circuit television (CCTV).

1.6. Five Steps to Face Recognition

As a biometric, facial recognition is a form of computer vision that uses faces to attempt to identify a person or verify a person's claimed identity. Regardless of specific method used, facial recognition is accomplished in a five step process.

- First, an image of the face is acquired. This acquisition can be accomplished by digitally scanning an existing photograph or by using an electro-optical camera to acquire a live picture of a subject. As video is a rapid sequence of individual still images, it can also be used as a source of facial images.
- Second, software is employed to detect the location of any faces in the acquired image. This task is difficult, and often generalized patterns of what a face "looks like" (two eyes and a mouth set in an oval shape) are employed to pick out the faces.

- Once the facial detection software has targeted a face, it can be analyzed. As noted in slide three, facial recognition analyzes the spatial geometry of distinguishing features of the face. Different vendors use different methods to extract the identifying features of a face. Thus, specific details on the methods are proprietary. The most popular method is called Principle Components Analysis (PCA), which is commonly referred to as the eigenface method. PCA has also been combined with neural networks and local feature analysis in efforts to enhance its performance. Template generation is the result of the feature extraction process. A template is a reduced set of data that represents the unique features of an enrollee's face. It is important to note that because the systems use spatial geometry of distinguishing facial features, they do not use hairstyle, facial hair, or other similar factors.
- The fourth step is to compare the template generated in step three with those in a database of known faces. In an identification application, this process yields scores that indicate how closely the generated template matches each of those in the database. In a verification application, the generated template is only compared with one template in the database – that of the claimed identity.
- The final step is determining whether any scores produced in step four are high enough to declare a match. The rules governing the declaration of a match are often configurable by the end user, so that he or she can determine how the facial recognition system should behave based on security and operational considerations.

1.7. Human Difficulties with Face Recognition Surveillance

People are generally very good at recognizing faces that they know. However, people experience difficulties when they perform facial recognition in a surveillance or watch post scenario [9,10]. Several factors account for these difficulties: most notably, humans have a hard time recognizing unfamiliar faces. Combined with relatively short attention spans, it is difficult for humans to pick out unfamiliar faces.

Considerable evidence supports this claim. For example, in a British study, trained supermarket cashiers were tested on their ability to screen shoppers using credit cards that included a photograph of the card owner [16]. Each shopper was issued

four cards: one with a recent picture of the shopper, one that included minor modifications to the shopper's hairstyle, facial hair or accessories (*e.g.*, glasses, hat), another card with a photograph of a person similar in appearance to the shopper, and the last card with a photograph of a person who was only of the same sex and race as the shopper. When the various cards were presented to the checkout clerks, more than half of the fraudulent cards were accepted. The breakdown was as follows: 34 percent of the cards that did not look like the shopper were accepted, 14 percent of the cards where the appearance had been altered were accepted, and 7 percent of the unchanged cards were rejected by the clerks.

In addition to unfamiliar face recognition problems, the ability of human beings to detect critical signals drops rapidly from the start of a task, and stabilizes at a significantly lower level within 25 to 35 minutes. Thus the ability of people to focus their attention drops significantly after only half an hour.

1.8. Technical Difficulties with Face Recognition Surveillance

Machines also experience difficulties when they perform facial recognition in a surveillance or watch post scenario. Performing face recognition processes with relatively high fidelity and at long distances remains technically challenging for automated systems [17]. At the most basic level, detecting whether a face is present in a given electronic photograph is a difficult technical problem. It is noted that subjects should ideally be photographed under tightly controlled conditions. For example, each subject should look directly into the camera and fill the area of the photo for an automated system to reliably identify the individual or even detect his face in the photograph. Thus, while the technology for facial recognition systems shows promise, it is not yet considered fully mature.

2. PRINCIPAL COMPONENT ANALYSIS (PCA)

2.1. Introduction

This chapter is designed to give the reader an understanding of Principal Components Analysis (PCA). PCA is a useful statistical technique that has found application in fields such as face recognition and image compression, and is a common technique for finding patterns in data of high dimension [19,20].

Before getting to a description of PCA, this chapter first introduces mathematical concepts that will be used in PCA. It covers standard deviation, covariance, eigenvectors and eigenvalues [20,21]. This background knowledge is meant to make the PCA section very straightforward, but can be skipped if the concepts are already familiar.There are examples all the way through this tutorial that are meant to illustrate the concepts being discussed.

2.2. Background Mathematics

This section will attempt to give some elementary background mathematical skills that will be required to understand the process of Principal Components Analysis. The topics are covered independently of each other, and examples given. It is less important to remember the exact mechanics of a mathematical technique than it is to understand the reason why such a technique may be used, and what the result of the operation tells us about our data. Not all of these techniques are used in PCA, but the ones that are not explicitly required do provide the grounding on which the most important techniques are based.

I have included a section on Statistics, which looks at distribution measurements, or, how the data is spread out. The other section is on Matrix Algebra and looks at eigenvectors and eigenvalues, important properties of matrices that are fundamental to PCA.

2.2.1. Statistics

The entire subject of statistics is based around the idea that you have this big set of data, and you want to analyse that set in terms of the relationships between the individual points in that data set. I am going to look at a few of the measures you can do on a set of data, and what they tell you about the data itself.

2.2.1.1. Standart Deviation

To understand standard deviation, we need a data set. Statisticians are usually concerned with taking a *sample* of a *population*. To use election polls as an example, the population is all the people in the country, whereas a sample is a subset of the population that the statisticians measure. The great thing about statistics is that by only measuring (in this case by doing a phone survey or similar) a sample of the population, you can work out what is most likely to be the measurement if you used the entire population.

In this statistics section, I am going to assume that our data sets are samples of some bigger population. There is a reference later in this section pointing to more information about samples and populations.

Here's an example set:

$$X=[1\ \ 2\ \ 4\ \ 6\ \ 12\ \ 15\ \ 25\ \ 45\ \ 68\ \ 67\ \ 65\ \ 98]$$

I could simply use the symbol X to refer to this entire set of numbers. If I want to refer to an individual number in this data set, I will use subscripts on the symbol X to indicate a specific number. Eg. X_3 refers to the 3rd number in X, namely the number 4. Note that X_1 is the first number in the sequence, not X_0 like you may see in some textbooks. Also, the symbol *n* will be used to refer to the number of elements in the set X.

There are a number of things that we can calculate about a data set. For example, we can calculate the mean of the sample. I assume that the reader understands what the mean of a sample is, and will only give the formula:

$$\overline{X} = \frac{\sum_{i=1}^{n} Xi}{n} \qquad \textbf{(2.1)}$$

Notice the symbol $\overline{X}$ (said "X bar") to indicate the mean of the set X. All this formula says is "Add up all the numbers and then divide by how many there are".

Unfortunately, the mean doesn't tell us a lot about the data except for a sort of middle point. For example, these two data sets have exactly the same mean (10), but are obviously quite different:

[0 8 12 20] and [8 9 11 12]

So what is different about these two sets? It is the *spread* of the data that is different. The Standard Deviation (SD) of a data set is a measure of how spread out the data is. How do we calculate it? The English definition of the SD is: "The average distance from the mean of the data set to a point". The way to calculate it is to compute the squares of the distance from each data point to the mean of the set, add them all up, divide by n-1, and take the positive square root. As a formula:

$$s=\sqrt{\frac{\sum_{i=1}^{n}(Xi-\overline{X})^2}{(n-1)}} \qquad \textbf{(2.2)}$$

Where "s" is the usual symbol for standard deviation of a sample. I hear you asking "Why are you using (n-1) and not n?". Well, the answer is a bit complicated, but in general, if your data set is a *sample* data set, ie. you have taken a subset of the real-world (like surveying 500 people about the election) then you must use (n-1) because it turns out that this gives you an answer that is closer to the standard deviation that would result if you had used the *entire* population, than if you'd used "n" . If, however, you are not calculating the standard deviation for a sample, but for an entire population, then you should divide by "n" instead of "(n-1)".

Table 2.1: Calculation of Standart Deviation Set 1

X	**(X- $\overline{X}$)**	**(X- $\overline{X}$)²**
0	-10	100
8	-2	4
12	2	4
20	10	100
Total		**208**
Divide by (n-1)		**69,333**
Square Root		**8,3266**

Table 2.2: Calculation of Standart Deviation Set 2

X_i	**(X_i- $\overline{X}$)**	**(X_i- $\overline{X}$)²**
8	-2	4
9	-1	1
11	1	1
12	2	4
Total		**10**
Divide by (n-1)		**3,333**
Square Root		**1,8257**

So, for our two data sets above, the calculations of standard deviation are in Table 2.1. And so, as expected, the first set has a much larger standard deviation due to the fact that the data is much more spread out from the mean. Just as another example, the data set:

[10 10 10 10]

also has a mean of 10, but its standard deviation is 0, because all the numbers are the same. None of them deviate from the mean.

2.2.1.2. Variance

Variance is another measure of the spread of data in a data set. In fact it is almost identical to the standard deviation. The formula is this:

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{(n-1)} \qquad \textbf{(2.3)}$$

You will notice that this is simply the standard deviation squared, in both the symbol (s^2) and the formula (there is no square root in the formula for variance). s^2 is the usual symbol for variance of a sample. Both these measurements are measures of the spread of the data. Standard deviation is the most common measure, but variance is also used. The reason why I have introduced variance in addition to standard deviation is to provide a solid platform from which the next section, covariance, can launch from.

2.2.1.3. Covariance

The last two measures we have looked at are purely 1-dimensional. Data sets like this could be: heights of all the people in the room, marks for the last COMP101 exam etc. However many data sets have more than one dimension, and the aim of the statistical analysis of these data sets is usually to see if there is any relationship between the dimensions. For example, we might have as our data set both the height of all the students in a class, and the mark they received for that paper. We could then perform statistical analysis to see if the height of a student has any effect on their mark.

Standard deviation and variance only operate on 1 dimension, so that you could only calculate the standard deviation for each dimension of the data set *independently* of the other dimensions. However, it is useful to have a similar measure to find out how much the dimensions vary from the mean *with respect to each other*.

Covariance is such a measure. Covariance is always measured *between* 2 dimensions. If you calculate the covariance between one dimension and *itself*, you get the variance. So, if you had a 3-dimensional data set (x, y, z), then you could measure the covariance between the x and y dimensions, the x and z dimensions, and the y and z dimensions. Measuring the covariance between x and x, or y and y, or z and z would give you the variance of the x, y and z dimensions respectively.

The formula for covariance is very similar to the formula for variance. The formula for variance could also be written like this:

$$Var\,(\mathrm{X}) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(X_i - \overline{X})}{(n-1)} \qquad \textbf{(2.4)}$$

where I have simply expanded the square term to show both parts. So given that knowledge, here is the formula for covariance:

$$\text{Cov (X,Y)} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{(n-1)} \qquad \textbf{(2.5)}$$

It is exactly the same except that in the second set of brackets, the X's are replaced by Y's. This says, in English, "For each data item, multiply the difference between the x value and the mean of x , by the the difference between the y value and the mean of y . Add all these up, and divide by (n-1)".

How does this work? Lets use some example data. Imagine we have gone into the world and collected some 2-dimensional data, say, we have asked a bunch of students how many hours in total that they spent studying COSC241, and the mark that they received. So we have two dimensions, the first is the *H* dimension, the hours studied, and the second is the *M* dimension, the mark received. The calculation of *cov(H,M)*, the covariance between the Hours of study done and the Mark received.

So what does it tell us? The exact value is not as important as it's sign (ie. positive or negative). If the value is positive, as it is here, then that indicates that both dimensions *increase together*, meaning that, in general, as the number of hours of study increased, so did the final mark.

If the value is negative, then as one dimension increases, the other decreases. If we had ended up with a negative covariance here, then that would have said the opposite, that as the number of hours of study increased the final mark *decreased*. In the last case, if the covariance is zero, it indicates that the two dimensions are independent of each other.

The result that mark given increases as the number of hours studied increases can be easily seen by drawing a graph of the data. However, the luxury of being able to visualize data is only available at 2 and 3 dimensions. Since the covariance value can be calculated between any 2 dimensions in a data set, this technique is often used to find relationships between dimensions in high-dimensional data sets where visualisation is difficult.

You might ask "is *cov(X,Y)* equal to *cov(Y,X)*"? Well, a quick look at the formula for covariance tells us that yes, they are exactly the same since the only difference between *cov(X,Y)* and *cov(Y,X)* is that (Xi- $\overline{X}$)(Yi- $\overline{Y}$) is replaced by (Yi- $\overline{Y}$)(Xi- $\overline{X}$). And since multiplication is commutative, which means that it doesn't matter which way around I multiply two numbers, I always get the same number, these two equations give the same answer.

2.2.1.4. The Covariance Matrix

Recall that covariance is always measured between 2 dimensions. If we have a data set with more than 2 dimensions, there is more than one covariance measurement that can be calculated. For example, from a 3 dimensional data set (dimensions *x, y, z*) you could calculate *cov(x,y), cov(x,z)* and *cov(y,z)*. In fact, for an *n*-dimensional data set, you can calculate $\frac{n!}{(n-2)*2}$ different covariance values.

Table 2.3: Two dimensional data set and covariance calculation

	Hours (H)	**Mark (M)**	**(H_i- $\overline{H}$)**	**(M_i- $\overline{M}$)**	**(H_i- $\overline{H}$)(M_i- $\overline{M}$)**
Data	9	39	-4.92	-23.42	115.23
	15	56	1.08	-6.42	-6.93
	25	93	11.08	30.58	338.83
	14	61	0.08	-1.42	-0.11
	10	50	-3.92	-12.42	48.69
	18	75	4.08	12.58	51.33
	0	32	-13.92	-30.42	423.45
	16	85	2.08	22.58	46.97
	5	42	-8.92	-20.42	182.15
	19	70	5.08	7.58	38.51
	16	66	2.08	3.58	7.45
	20	80	6.08	17.58	106.89
Totals	**167**	**749**			**1149.89**
Averages	**13.92**	**62.42**			**104.54**

A useful way to get all the possible covariance values between all the different dimensions is to calculate them all and put them in a matrix. I assume in this tutorial that you are familiar with matrices, and how they can be defined. So, the definition for the covariance matrix for a set of data with *n* dimensions is:

$$C^{nxn} = (c_{i,j}\ ,\ c_{i,j} = \text{Cov}\ (\ Dim_i,\ Dim_j\ )\ ), \qquad \textbf{(2.6)}$$

Where C^{nxn} is a matrix with n rows and n coloumns, and Dim_x is the xth dimension. All that this ugly looking formula says is that if you have an n-dimensional data set, then the matrix has n rows and coloumns (so is square) and each entry in the matrix is the result of calculating the covariancebetween two separate dimensions. Eg. the entry on row 2, coloumn 3, is the covariance value calculated between the 2nd dimension and the 3rd dimension.

An example. We'll make up the covariance matrix for an imaginary 3 dimensional data set, using the usual dimensions x, y and z. Then, the covariance matrix has 3 rows and 3 columns, and the values are this:

$$C = \begin{pmatrix} \text{cov}(x,x) & \text{cov}(x,y) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{pmatrix} \quad \textbf{(2.7)}$$

Some points to note: Down the main diagonal, you see that the covariance value is between one of the dimensions and itself. These are the variances for that dimension. The other point is that since *cov(a,b) = cov(b,a)*, the matrix is symmetrical about the main diagonal.

2.3.2. Matrix Algebra

This section serves to provide a background for the matrix algebra required in PCA. Specifically I will be looking at eigenvectors and eigenvalues of a given matrix. Again, I assume a basic knowledge of matrices.

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} x \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} x \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \text{ x } \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Figure 2.1: Example of one non-eigenvector and one eigenvector

$$2 \text{ x } \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} x \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \text{ x } \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

Figure 2.2: Example of how a scaled eigenvector is still and eigenvector

2.3.2.1. Eigenvectors

As you know, you can multiply two matrices together, provided they are compatible sizes. Eigenvectors are a special case of this. Consider the two multiplications between a matrix and a vector in Figure 2.1.

In the first example, the resulting vector is not an integer multiple of the original vector, whereas in the second example, the example is exactly 4 times the vector we began with. Why is this? Well, the vector is a vector in 2 dimensional space. The vector $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ (from the second example multiplication) represents an arrow pointing from the origin, (0,0), to the point (3,2). The other matrix, the square one, can be thought of as a transformation matrix. If you multiply this matrix on the left of a vector, the answer is another vector that is transformed from it's original position.

It is the nature of the transformation that the eigenvectors arise from. Imagine a transformation matrix that, when multiplied on the left, reflected vectors in the line $y = x$. Then you can see that if there were a vector that lay *on* the line $y = x$, it's reflection it *itself*. This vector (and all multiples of it, because it wouldn't matter how long the vector was), would be an eigenvector of that transformation matrix.

What properties do these eigenvectors have? You should first know that eigenvectors can only be found for *square* matrices. And, not every square matrix has eigenvectors. And, given an *nxn* matrix that does have eigenvectors, there are *n* of them. Given *3x3* matrix, there are 3 eigenvectors.

Another property of eigenvectors is that even if I scale the vector by some amount before I multiply it, I still get the same multiple of it as a result. This is because if you scale a vector by some amount, all you are doing is making it longer, not changing it's direction. Lastly, all the eigenvectors of a matrix are *perpendicular*, ie. at right angles to each other, no matter how many dimensions you have. By the way, another word for perpendicular, in maths talk, is *orthogonal*. This is important because it means that you can express the data in terms of these perpendicular eigenvectors, instead of expressing them in terms of the *x* and *y* axes. We will be doing this later in the section on PCA.

Another important thing to know is that when mathematicians find eigenvectors, they like to find the eigenvectors whose length is exactly one. This is because, as you

know, the length of a vector doesn't affect whether it's an eigenvector or not, whereas the direction does. So, in order to keep eigenvectors standard, whenever we find an eigenvector we usually scale it to make it have a length of 1, so that all eigenvectors have the same length. Here's a demonstration from our example above.

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

is an eigenvector, and the length of that vector is

$$\sqrt{(3^2 + 2^2)} = \sqrt{13}$$

so we divide the original vector by this much to make it have a length of 1.

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \div \sqrt{13} = \begin{pmatrix} 3 / \sqrt{13} \\ 2 / \sqrt{13} \end{pmatrix}$$

How does one go about finding these mystical eigenvectors? Unfortunately, it's only easy(ish) if you have a rather small matrix, like no bigger than about 3x3. After that, the usual way to find the eigenvectors is by some complicated iterative method which is beyond the scope of this tutorial (and this author). If you ever need to find the eigenvectors of a matrix in a program, just find a maths library that does it all for you. A useful maths package, called newmat, is available.

2.3.2.2. Eigenvalues

Eigenvalues are closely related to eigenvectors, in fact, we saw an eigenvalue in Figure 2.1. Notice how, in both those examples, the amount by which the original vector was scaled after multiplication by the square matrix was the same? In that example, the value was 4. 4 is the *eigenvalue* associated with that eigenvector. No matter what multiple of the eigenvector we took before we multiplied it by the square matrix, we would always get 4 times the scaled vector as our result (as in Figure 2.2)

So you can see that eigenvectors and eigenvalues always come in pairs. When you get a fancy programming library to calculate your eigenvectors for you, you usually get the eigenvalues as well.

2.4. Principal Component Analysis

Finally we come to Principal Components Analysis (PCA). What is it? It is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analysing data.

The other main advantage of PCA is that once you have found these patterns in the data, and you compress the data, ie. by reducing the number of dimensions, without much loss of information. This technique used in image compression, as we will see in a later section.

This chapter will take you through the steps you needed to perform a Principal Components Analysis on a set of data. I am not going to describe exactly *why* the technique works, but I will try to provide an explanation of what is happening at each point so that you can make informed decisions when you try to use this technique yourself.

2.4.1. Method

Step 1. Get some data

In my simple example, I am going to use my own made-up data set. It's only got 2 dimensions, and the reason why I have chosen this is so that I can provide plots of the data to show what the PCA analysis is doing at each step.

The data I have used is found in Figure 2.3, along with a plot of that data.

Table 2.4: Data and Data Adjust

	x	y		x	y
Data	2.5	2.4	Data Adjust	0.69	0.49
	0.5	0.7		-1.31	-1.21
	2.2	2.9		0.39	0.99
	1.9	2.2		0.09	0.29
	3.1	3.0		1.29	1.09
	2.3	2.7		0.49	0.79
	2	1.6		0.19	-0.31
	1	1.1		-0.81	-0.81
	1.5	1.6		-0.31	-0.31
	1.1	0.9		-0.71	-1.01

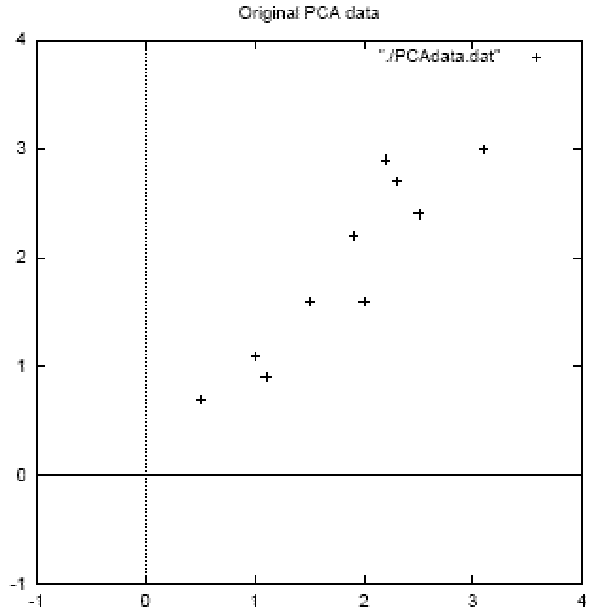

Figure 2.3: PCA example data, original data on the left, data with the means subtracted on the right, and a plot of the data

Step 2: Substract the mean

For PCA to work properly, you have to subtract the mean from each of the data dimensions. The mean subtracted is the average across each dimension. So, all the x values have $\bar{x}$ (the mean of the x values of all the data points) subtracted, and all the y values have $\bar{y}$ subtracted from them. This produces a data set whose mean is zero.

Step 3: Calculate the covariance matrix

This is done in exactly the same way as was discussed before. Since the data is 2 dimensional, the covariance matrix will be 2x2. There are no surprises here, so I will just give you the result:

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

So, since the non-diagonal elements in this covariance matrix are positive, we should expect that both the *x* and *y* variable increase together.

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

Since the covariance matrix is square, we can calculate the eigenvectors and eigenvalues for this matrix. These are rather important, as they tell us useful information about our data. I will show you why soon. In the meantime, here are the eigenvectors and eigenvalues:

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

It is important to notice that these eigenvectors are both *unit* eigenvectors ie. their lengths are both 1. This is very important for PCA, but luckily, most maths packages, when asked for eigenvectors, will give you unit eigenvectors.

So what do they mean? If you look at the plot of the data in Figure 2.4 then you can see how the data has quite a strong pattern. As expected from the covariance matrix, they two variables do indeed increase together. On top of the data I have plotted both the eigenvectors as well. They appear as diagonal dotted lines on the plot. As stated in the eigenvector section, they are perpendicular to each other. But, more importantly, they provide us with information about the patterns in the data. See how one of the eigenvectors goes through the middle of the points, like drawing a line of best fit? That eigenvector is showing us how these two data sets are related along that line. The second eigenvector gives us the other, less important, pattern in the data,

that all the points follow the main line, but are off to the side of the main line by some amount.

So, by this process of taking the eigenvectors of the covariance matrix, we have been able to extract lines that characterise the data. The rest of the steps involve transforming the data so that it is expressed in terms of them lines.

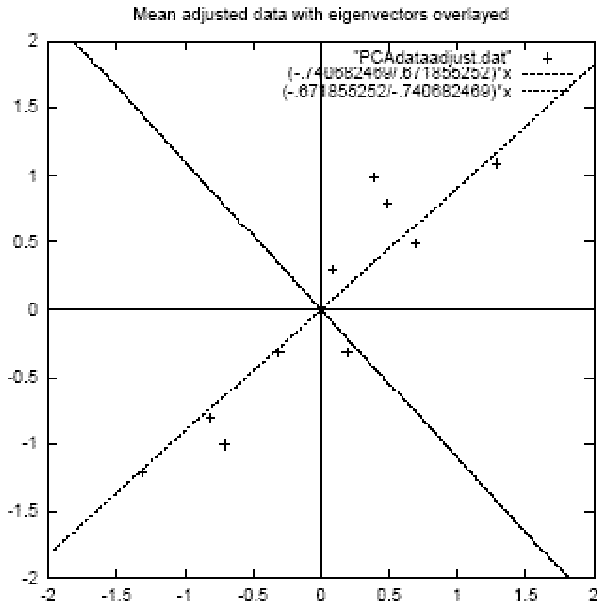

Figure 2.4: A plot of the normalized data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

Step 5: Choosing components and forming a feature vector

Here is where the notion of data compression and reduced dimensionality comes into it. If you look at the eigenvectors and eigenvalues from the previous section, you will notice that the eigenvalues are quite different values. In fact, it turns out that the eigenvector with the *highest* eigenvalue is the *principle component* of the data set. In our example, the eigenvector with the larges eigenvalue was the one that pointed down the middle of the data. It is the most significant relationship between the data dimensions.

In general, once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest. This gives you the components in order of significance. Now, if you like, you can decide to *ignore* the components of lesser significance. You do lose some information, but if the eigenvalues are small, you don't lose much. If you leave out some components, the final data set will have less dimensions than the original. To be precise, if you originally have *n* dimensions

in your data, and so you calculate *n* eigenvectors and eigenvalues, and then you choose only the first *p* eigenvectors, then the final data set has only *p* dimensions.

What needs to be done now is you need to form a *feature vector*, which is just a fancy name for a matrix of vectors. This is constructed by taking the eigenvectors that you want to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns.

$$FeatureVector = (eig_1\ eig_2\ eig_3\ \ldots\ldots\ldots eig_n) \quad \textbf{(2.8)}$$

Given our example set of data, and the fact that we have 2 eigenvectors, we have two choices. We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

or, we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

We shall see the result of each of these in the next section.

Step 5: Deriving the new data set

This the final step in PCA, and is also the easiest. Once we have chosen the components (eigenvectors) that we wish to keep in our data and formed a feature vector, we simply take the transpose of the vector and multiply it on the left of the original data set, transposed.

$$FinalData = RowFeatureVector \times RowDataAdjust \quad \textbf{(2.9)}$$

where *RowFeatureVector* is the matrix with the eigenvectors in the columns *transposed* so that the eigenvectors are now in the rows, with the most significant eigenvector at the top, and *RowDataAdjust* is the mean-adjusted data *transposed*, ie. the data items are in each column, with each row holding a separate dimension. I'm sorry if this sudden transpose of all our data confuses you, but the equations from here on are easier if we take the transpose of the feature vector and the data first,

rather that having a little T symbol above their names from now on. *FinalData* is the final data set, with data items in columns, and dimensions along rows.

What will this give us? It will give us the original data *solely in terms of the vectors we chose*. Our original data set had two axes, x and y , so our data was in terms of them. It is possible to express data in terms of any two axes that you like. If these axes are perpendicular, then the expression is the most efficient. This was why it was important that eigenvectors are always perpendicular to each other. We have changed our data from being in terms of the axes x and y, and now they are in terms of our 2 eigenvectors. In the case of when the new data set has reduced dimensionality, ie. we have left some of the eigenvectors out, the new data is only in terms of the vectors that we decided to keep.

To show this on our data, I have done the final transformation with each of the possible feature vectors. I have taken the transpose of the result in each case to bring the data back to the nice table-like format. I have also plotted the final points to show how they relate to the components.

In the case of keeping both eigenvectors for the transformation, we get the data and the plot found in Figure 2.5. This plot is basically the original data, rotated so that the eigenvectors are the axes. This is understandable since we have lost no information in this decomposition.

The other transformation we can make is by taking only the eigenvector with the largest eigenvalue. The table of data resulting from that is found in Table 2.6. As expected, it only has a single dimension. If you compare this data set with the one resulting from using both eigenvectors, you will notice that this data set is exactly the first column of the other. So, if you were to plot this data, it would be 1 dimensional, and would be points on a line in exactly the x positions of the points in the plot in Figure 2.5. We have effectively thrown away the whole other axis, which is the other eigenvector.

So what have we done here? Basically we have transformed our data so that is expressed in terms of the patterns between them, where the patterns are the lines that most closely describe the relationships between the data. This is helpful because we have now classified our data point as a combination of the contributions from each of those lines. Initially we had the simple x and y axes. This is fine, but the x and y

values of each data point don't really tell us exactly how that point relates to the rest of the data. Now, the values of the data points tell us exactly where (ie. above/below) the trend lines the data point sits. In the case of the transformation using *both* eigenvectors, we have simply altered the data so that it is in terms of those eigenvectors instead of the usual axes. But the single-eigenvector decomposition has removed the contribution due to the smaller eigenvector and left us with data that is only in terms of the other.

2.4.1.1. Getting the old data back

Wanting to get the original data back is obviously of great concern if you are using the PCA transform for data compression (an example of which to will see in the next section).

Table 2.5: Transformed Data

	X	Y
Transformed Data	-.827970186	-.175115307
	1.77758033	.142857227
	-.992197494	.384374989
	-.274210416	.130417207
	-1.67580142	-.209498461
	-.912949103	.175282444
	.0991094375	-.349824698
	1.14457216	.0464172582
	.438046137	.0177646297
	1.22382056	-.162675287

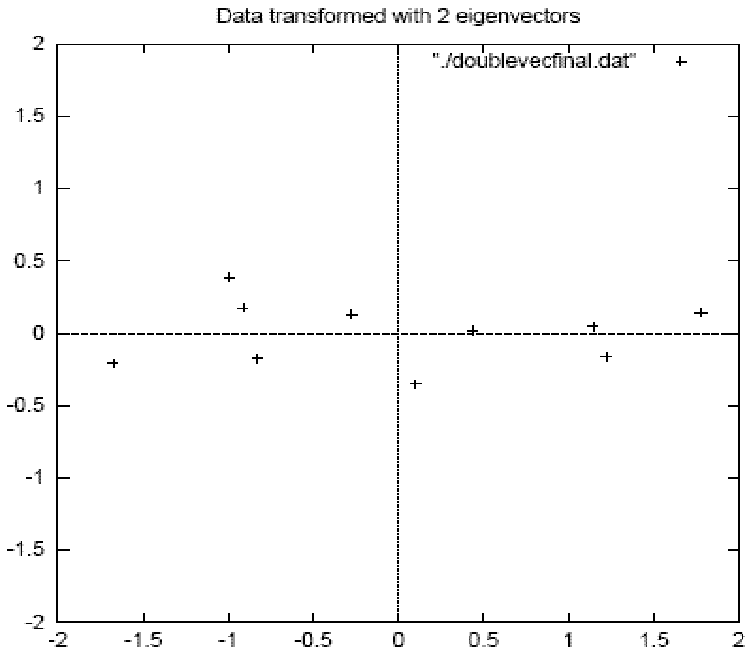

Figure 2.5: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

Table 2.6: The data after transforming using only the most significant eigenvector

X
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

So, how do we get the original data back? Before we do that, remember that only if we took *all* the eigenvectors in our transformation will we get *exactly* the original data back. If we have reduced the number of eigenvectors in the final transformation, then the retrieved data has lost some information.

Recall that the final transform is this:

$$FinalData = RowFeatureVector \text{ x } RowDataAdjust \quad \textbf{(2.10)}$$

which can be turned around so that, to get the original data back,

$$RowDataAdjust = RowFeatureVector^{-1} \text{ x } FinalData \quad \textbf{(2.11)}$$

where $RowFeatureVector^{-1}$ is the inverse of RowFeatureVector . However, when we take *all* the eigenvectors in our feature vector, it turns out that the inverse of our feature vector is actually equal to the transpose of our feature vector. This is only true because the elements of the matrix are all the unit eigenvectors of our data set. This makes the return trip to our data easier, because the equation becomes

$$RowDataAdjust = RowFeatureVector^{T} \text{ x } FinalData \quad \textbf{(2.12)}$$

But, to get the actual original data back, we need to add on the mean of that original data (remember we subtracted it right at the start). So, for completeness,

$$RowOriginalData = (RowFeatureVector^{T} \text{ x } FinalData) + OriginalMean \quad \textbf{(2.13)}$$

This formula also applies to when you do not have all the eigenvectors in the feature vector. So even when you leave out some eigenvectors, the above equation still makes the correct transform.

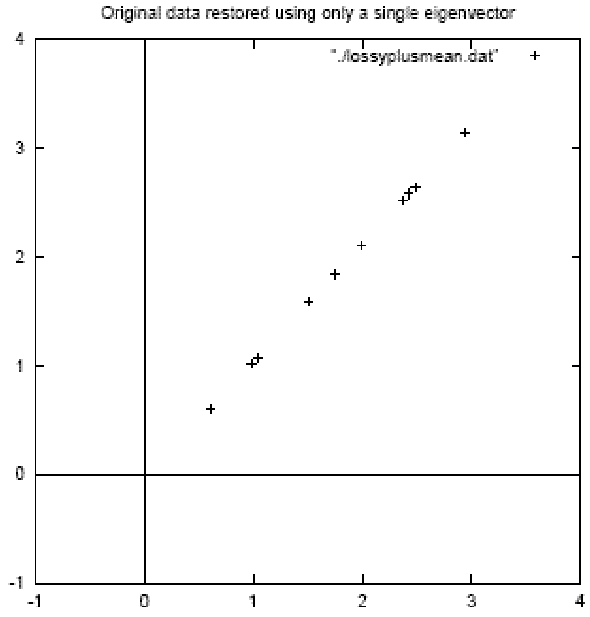

Figure 2.6: The reconstruction from the data that was derived using only a single eigenvector

I will not perform the data re-creation using the *complete* feature vector, because the result is exactly the data we started with. However, I will do it with the reduced feature vector to show you how information has been lost. Figure 2.6 show this plot. Compare it to the original data plot in Figure 2.3 and you will notice how, while the variation along the principle eigenvector (see Figure 2.4 for the eigenvector overlayed on top of the mean-adjusted data) has been kept, the variation along the other component (the other eigenvector that we left out) has gone.

3. LINEAR DISCRIMINANT ANALYSIS

3.1. Introduction

There are many possible techniques for classification of data. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are two commonly used techniques for data classification and dimensionality reduction. Linear Discriminant Analysis easily handles the case where the within-class frequencies are unequal and their performances has been examined on randomly generated test data. This method maximizes the ratio of between-class variance to the within-class variance in any particular data set thereby guaranteeing maximal separability. The prime difference between LDA and PCA is that PCA does more of feature classification and LDA does data classification. In PCA, the shape and location of the original data sets changes when transformed to a different space whereas LDA doesn't change the location but only tries to provide more class separability and draw a decision region between the given classes. This method also helps to better understand the distribution of the feature data.

Linear Discriminant Analysis (LDA) has been successfully applied to face recognition. The objective of LDA is to seek a linear projection from the image space onto a low dimensional space by maximizing the between-class scatter and minimizing the within-class scatter simultaneously [16]. Belhumeur compared Fisherface with Eigenface on the *HARVARD* and *YALE* face databases, and showed that LDA was better than PCA, especially under illumination variation. LDA was also evaluated favorably under the *FERET* testing framework.

In many practical face recognition tasks, there are not enough samples to make the within-class scatter matrix S_w nonsingular, this is called a small sample size problem [17,18,19]. Different solutions have been proposed to deal with it in using LDA for face recognition.

The most widely used methods (Fisherface) applies PCA firstly to reduce the dimension of the samples to an intermediate dimension, which must be guaranteed not more than the rank of S_w so as to obtain a full-rank within-class scatter matrix [6,15,17]. Then standard LDA is used to extract and represent facial features. All these methods above do not consider the importance of null space of the within-class scatter matrix, and remove the null space to make the resulting within-class scatter full-rank.

Yang et al. proposed a new algorithm which incorporates the concept of null space. It first removes the null space of the between-class scatter matrix S_b and seeks a projection to minimize the within-class scatter (called Direct LDA / DLDA) [7,19]. Because the rank of S_b is smaller than that of S_w, removing the null space of S_b may lose part of or the entire null space of S_w, which is very likely to be full-rank after the removing operation.

Chen et al. proposed a more straightforward method that makes use of the null space of S_w. The basic idea is to project all the samples onto the null space of S_w, where the resulting within-class scatter is zero, and then maximize the between-class scatter [3,5,11]. This method involves computing eigenvalue in a very large dimension since S_w is an $n \times n$ matrix. To avoid the great computational cost, pixel grouping method is used in advance to artificially extract features and to reduce the dimension of the original samples.

Huang et al. introduced a more efficient null space approach. The basic notion behind the algorithm is that the null space of S_w is particularly useful in discriminating ability, whereas, that of S_b is useless [8,12,13]. They proved that the null space of the total scatter matrix S_t is the common null space of both S_w and S_b. Hence the algorithm firstly removes the null space of S_t and projects the samples onto the null space of S_w . Then it removes the null space of the between-class scatter in the subspace to get the optimal discriminant vectors.

Although null space-based LDA seems to be more efficient than other linear subspace analysis methods for face recognition, it is still a linear technique in nature [6]. Hence it is inadequate to describe the complexity of real face images because of illumination, facial expression and pose variations. The kernel technique has been extensively demonstrated to be capable of efficiently representing complex nonlinear

relations of the input data. Kernel Fisher Discriminant Analysis (KFDA) is an efficient nonlinear subspace analysis method, which combines the kernel technique with LDA [15]. After the input data are mapped into an implicit feature space, LDA is performed to yield nonlinear discriminating features of the input data.

In this paper, some elements of state-of-the-art null space techniques will be looked at in more depth and our null space approach is proposed to save the computational cost and maintain the performance simultaneously. Furthermore, we concentrate on the advantages of both the null space approach and the kernel technique [18,19]. A kernel mapping based on an efficient kernel function, called Cosine kernel, is performed on all the samples firstly. In kernel space, we can find that the total scatter matrix is full rank, so the procedure of the null space approach is greatly simplified and more stable in numerical computation.

3.2. Previous Work

Some assumptions and definitions in mathematics are provided at first. Let n denote the dimension of the original sample space, and c is the number of classes. The between-class scatter matrix S_b and the within-class scatter S_w are defined as below [17,18,19]:

$$S_b = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T = \Phi_b \Phi_b^T , \quad \textbf{(3.1)}$$

$$S_w = \sum_{i=1}^{c} \sum_{k \in ci} (x_k - m_i)(x_k - m_i)^T = \Phi_w \Phi_w^T , \quad \textbf{(3.2)}$$

where N_j is the number of samples in class C_i $(i=1,2,....,c)$, N is the number of all samples, m_j is the mean of the samples in the class C_i, and m is the overall mean of all samples [17]. The total scatter matrix i.e. the covariance matrix of all the samples is defined as:

$$S_t = S_b + S_w = \sum_{i=1}^{N} (x_i - m)(x_i - m)^T = \Phi_t \Phi_t^T , \quad \textbf{(3.3)}$$

LDA tries to find an optimal projection: $W = [w_1, w_2, w_3, ……., w_{c-1}]$,which satisfies

$$J(W) = \arg\max_{w} \frac{\left| W^T S_b W \right|}{\left| W^T S_w W \right|}, \quad \textbf{(3.4)}$$

that is just Fisher criterion function.

3.2.1. Standart LDA

As well known, W can be constructed by the eigenvectors of $S_w^{-1} S_b$. But this method is numerically unstable because it involves the direct inversion of a likely high-dimensional matrix [15,16]. The most frequently used LDA algorithm in practice is based on simultaneous diagonalization. The basic idea of the algorithm is to find a matrix W that can simultaneously diagonalize both S_w and S_b, i.e.,

$$W^T S_W W = I \,,\; W^T S_b W = \Lambda. \quad \textbf{(3.5)}$$

Most algorithms require that S_w be non-singular, because the algorithms diagonalize S_w first. The above procedure will break down when S_w becomes singular [12,15]. It surely happens when the number of training samples is smaller than the dimension of the sample vector, i.e. the small sample size problem (SSSP). The singularity exists for most face recognition tasks.

An available solution to this problem is to perform PCA to project the n-dimensional image space onto a lower dimensional subspace. The PCA step essentially removes null space from both S_w and S_b. Therefore, this step potentially loses useful information.

In fact, the null space of S_w contains the most discriminative information especially when the projection of S_b is not zero in that direction. The Direct LDA (DLDA) algorithm is presented to keep the null space of S_w.

DLDA removes the null space of S_b firstly by performing eigen analysis on S_b , then a simultaneous procedure is used to seek the optimal discriminant vectors in the subspace of S_b, i.e.

$$W^T S_b W = I, \;\; W^T S_w W = D_w. \quad \textbf{(3.6)}$$

Because the rank of S_b is smaller than that of S_w in majority, removing the null space of S_b may lose part of or the entire null space of S_w, which is very likely to be full-

rank after the removing operation. So, DLDA does not make full use of the null space [10].

3.2.2. Direct LDA

D-LDA method proposed, which draws on a variant of discriminant analysis criterion and exploits the strength of the D-LDA algorithm.

3.2.2.1 A Variant Criterion Function

Most LDA based algorithms including Fisherfaces and D-LDA utilize the conventional Fisher criterion defined in (3.7), while some authors use the alternative given in (3.8) proposed by Liu.

$$\mathbf{\Phi} = \arg\max_{\Phi} \frac{\left|\Phi^T S_b \Phi\right|}{\left|\Phi^T S_w \Phi\right|} \tag{3.7}$$

$$\mathbf{\Phi} = \arg\max_{\Phi} \frac{\left|\Phi^T S_b \Phi\right|}{\left|\Phi^T S_t \Phi\right|} \tag{3.8}$$

where S_b, S_w and S_t are the between-class scatter matrix, within-class scatter matrix and population scatter matrix respectively.

We propose the use of a variant of Fisher criterion expressed as follows:

$$\mathbf{\Phi} = \arg\max_{\Phi} \frac{\left|\Phi^T S_t \Phi\right|}{\left|\Phi^T S_w \Phi\right|} \tag{3.9}$$

3.2.2.2. A Variant D-LDA

The between-class scatter matrix, which is of size *NxN*, can be expressed as $S_b = \Phi_b \Phi_b^T$, where $\Phi_b = \left[\sqrt{n_1}(\mu_1 - \mu), \ldots \sqrt{n_c}(\mu_c - \mu)\right]$, N is the dimensionality of the samples, n_i is the number of samples of *i-th* class, μ_i is the mean of *i-th* class, μ is the total mean vector of all samples, and C is the number of classes. Turk and Pentland proposed an indirect method to find the eigenvectors of $S_b = \Phi_b \Phi_b^T$ which can be indirectly derived from the eigenvectors of the matrix, $\Phi_b \Phi_b^T$, of size *CxC*. Let λ_i and e_i be the *i-th* eigenvalue and its corresponding eigenvector of $\Phi_b \Phi_b^T$,

where $i= 1,2,\ldots,C,$ and let λ_i be sorted in the decreasing order of magnitude. Since $(\Phi_b\Phi_b^T)(\Phi_b e_i) = \lambda_i \Phi_b e_i$, $y_i = \Phi_b e_i$ is the eigenvector of S_b. As the rank, m, of S_b satisfies $m= rank\ (S_b) = \min(N, C\text{-}1)$ there is no discriminative information in the null space of S_b . The first m eigenvectors, $Y = [y_1,y_2, \ldots , y_m] = \Phi_b E_m$, whose corresponding eigenvalues are greater than 0, are used, where $E_m= [e_1,e_2,\ldots,e_m]$.

Let $D_b = diag\ [\lambda_1, \lambda_2, \ldots , \lambda_m]$, and further let $Z=Y D_b^{-\frac{1}{2}}$ denote a projection matrix. Projecting S_w into the subspace spanned by Z, we have

$$\tilde{S}_w = Z^T S_w Z$$

Let u_i be the i-th eigenvector of $\tilde{S}_w$, where i=1,2,…,m , corresponds to the i-th eigenvalue λ_i'. The diagonal matrix of eigenvalues is denoted as $D_w = diag\ [\lambda_1', \ldots, \lambda_m']$, which corresponds to the matrix, U, of eigenvectors of $\tilde{S}_w$.

Let $P = ZU$ and further let $Q = P D_w^{-\frac{1}{2}}$ be a projection matrix. Projecting S_t and S_w into the subspace spanned by Q, we have

$$Q^T S_w Q = I$$

$$Q^T S_t Q = \tilde{S}_t$$

In order to maximize Eq.(3.9), we need only to select the eigenvectors of $\tilde{S}_t$. Let us denote the selected eigenvectors as V =[v_1, v_2,…, v_m], the corresponding diagonal matrix of eigenvalues as D_t. Note that D_t is an m'xm' diagonal matrix. The optimal discriminant feature extractor can be derived through $A=V^TQ^T$.

According to the above analysis, the steps of the new D-LDA algorithm can be summarized as follows:

Input: A set of training face images $\{x_i, i=1,2, \ldots, n_t\}$, each of which is represented as an N-dimensional vector.

Output: A low-dimensional representation **x*** of **x** with enhanced discriminatory power obtained by transformation **x*=Ax**.

Algorithm:

Step1. Determine the set E_m =[e_1, … ,e_m] of eigenvectors of $\Phi_b^T \Phi_b$ associated with the

$m \leq c\text{-}1$ non-zero eigenvalues.

Step 2. Calculate the first m most significant eigenvectors and the corresponding eigenvalues of S_b by $Y = \Phi_b E_m$ and D_b

Step 3. Let $Z=Y D_b^{-\frac{1}{2}}$. Calculate the eigenvectors U and the corresponding eigenvalues D_w of $Z^T S_W Z$

Step 4. Let P=ZU, Q= $P D_w^{-\frac{1}{2}}$. Calculate the first n most significant eigenvectors V and the corresponding eigenvalues D_t of $Q^T S_t Q$, *ie.* $V^T Q^T S_t QV=D_t$

Step 5. Let $A=V^T Q^T=(QV)^T$. The low dimensional transformed vector x* for each testing sample x is

$$x^* = Ax$$

3.2.3. Null Space-Based LDA

From Fisher's criterion that is objective function (3.4), we can find that: In standard LDA, W is seeked such that (3.5), so the form of the optimal solution provided by standard LDA is

$$\underset{LDA}{optimum} = \max_W \frac{\left| W^T S_b W \right|}{\left| W^T S_w W \right|} = \left| \Lambda \right| = \text{opt max/1.} \qquad \textbf{(3.10)}$$

In DLDA, W is seeked such that (3.6), so the form of the optimal solution provided by DLDA is

$$\underset{DLDA}{optimum} = \max_W \frac{\left| W^T S_b W \right|}{\left| W^T S_w W \right|} = \frac{1}{\left| D_w \right|} = \frac{1}{opt \min} \quad . \qquad \textbf{(3.11)}$$

Compared with above LDA approaches, a more reasonable method (Chen), we called Null Space-based LDA, has been presented. In Chen's theory, null space-based LDA should reach below:

$$\underset{NULL}{optimum} = \max_{W} \frac{\left| W^T S_b W \right|}{\left| W^T S_w W \right|} = \frac{opt \max}{0} \quad . \tag{3.12}$$

That means the optimal projection W should satisfy

$$W^T S_w W = 0, \;\; W^T S_b W = \Lambda, \tag{3.13}$$

i.e. the optimal discriminant vectors must exist in the null space of S_w.

In a performance benchmark, we can conclude that null space-based LDA generally outperforms LDA (Fisherface) or DLDA since

$$\underset{NULL}{optimum} = \infty \geq \underset{DLDA}{optimum} \geq \underset{LDA}{optimum} . \tag{3.14}$$

Because the computational complexity of extracting the null space of S is very high because of the high dimension of S. So a pixel grouping operation is used in advance to extract geometric features and to reduce the dimension of the samples [12,16]. However, the pixel grouping preprocess is irresponsible and may arouse a loss of useful facial features.

3.3. Suggested Null Space Method (NLDA)

In this section, the essence of null space-based LDA in the SSSP is revealed by theoretical justification, and the most suitable situation of null space methods is discovered [6]. Next, we propose the NLDA algorithm, which is conceptually simple yet powerful in performance.

3.3.1. Most Suitable Situation

For the small sample size problem (SSSP) in which $n>N$, the dimension of null space of S_w is very large, and not all null space contributes to the discriminative power. Since both S_b and S_w are symmetric and semi-positive, we can prove, as mentioned in [6], that

$$N(S_t) = N(S_b) \cap (S_w) \tag{3.15}$$

From the statistical perspective, the null space of S_b is of no use in its contribution to discriminative ability. Therefore, the useful subspace of null space of S_w is

$$\overset{\Lambda}{N}(S_w) = N(S_w) - N(S_t) = N(S_w) \cap \overline{N(S_t)} \quad \textbf{(3.16)}$$

The sufficient and necessary condition so that null space methods work is

$$\overset{\Lambda}{N}(S_w) \neq \Phi \Rightarrow N(S_w) \supset N(S_t) \Rightarrow \dim N(S_w) > \dim N(S_t) \Rightarrow$$
$$\operatorname{rank}(S_t) > \operatorname{rank}(S_w) \quad \textbf{(3.17)}$$

In many cases,

$$\operatorname{rank}(S_t) = \min\{n, N-1\}, \operatorname{rank}(S_w) = \min\{n, N-c\} \quad \textbf{(3.18)}$$

the dimension of discriminative null space of S_w can be evaluated from (3.15):

$$\dim \overset{\Lambda}{N}(S_w) = \operatorname{rank}(S_t) - \operatorname{rank}(S_w). \quad \textbf{(3.19)}$$

If $n \leq N-c$, due to $rank(S_t) = n \leq rank(S_w) = N-c$, the necessary condition (3.17) is not satisfied so that we can not extract any null space. That means any null space-based method does not work in the large sample size case.

If $N-c < n < N-1$, due to $rank(S_t) = n > rank(S_w) = N-c$, the dimension of effective null space can be evaluated fron (3.19): $\dim \overset{\Lambda}{N}(S_w) = n-N+c < c-1$. Hence, the number of discriminant vectors would be less than $c-1$, and some discriminatory information maybe lost.

Only when $n \geq N-1$ (SSSP), for $rank(S_t) = N-1 > rank(S_w) = N-c$, we derive dim $\overset{\Lambda}{N}(S_w) = c-1$. The dimension of extracted null space is just c-1, which coincides with the number of ideal features for classification [6,7,8]. Therefore, we can conclude that null space methods are always applicable to any small sample size problem.

Especially when n is equal to N-1, S_t is full-rank and $N(St)$ is null. By (3.16) we have $\overset{\Lambda}{N}(S_w) = N(S_w)$, it follows all null space of S_w contributes to the discriminative power. Hence, we conclude the most suitable situation for null space-based methods:

$$n = N-1. \quad \textbf{(3.20)}$$

3.3.2. NLDA

Combining (3.15)-(3.19), we develop our null space method [24].

1. Remove the null space of S_t

Perform PCA to project the *n*-dimensional image space onto a low dimensional subspace, i.e. perform eigen-analysis on S_t., the dimension of the extracted subspace is usually *N*-1 [17]. The projection *P*, whose columns are all the eigenvectors of S_t corresponding to the nonzero eigenvalues, are calculated firstly, and then the within-class scatter and between-class scatter in the resulting subspace are obtained.

$$P^T S_t P = D_t,\ P^T S_w P = S'_w,\ P^T S_b P = S'_b. \qquad \textbf{(3.21)}$$

2. Extract the null space of S_w'.

Diagonalize S_w', we have

$$V^T\ S_w'\ V = D_w \qquad \textbf{(3.22)}$$

Where $V^TV = I$, D_w is diagonal matrix sorted in increasing order. Discard those with eigenvalues sufficiently far from 0, keep *c*-1 eigenvectors of S_w' int most cases. Let *Y* be the first *c*-1 columns of *V*, which is the null space of S_w', we have

$$Y^T S_w' Y = 0,\ Y^T S_b'\ Y =\ S''_b\ . \qquad \textbf{(3.23)}$$

3. Diagonalize S''_b (usually a (c-1)x(c-1) matrix) which is full-rank.

Perform eigen-analysis:

$$U^T S_b'\ U = \Lambda, \qquad \textbf{(3.24)}$$

Where $U^TU=I$, Λ is diagonal matrix sorted in decreasing order.

The final projection matrix is:

$$W = PYU, \qquad \textbf{(3.25)}$$

W is usually an *n*x*(c-1)* matrix, which diagonalizes both the numerator and the denominator of Fisher's criterion to *(c-1)x(c-1)* matrices as (3.13), especially leads to a denominator of 0 matrix.

It is notable that the third step of Huang' algorithm is used to remove the null space of S''_b . In fact, we are able to prove that it is full-rank once through the previous two steps.

Lemmas S_b'' is full-rank, S_b'' is defined in step2 of algorithm I.

Proof:

From step1 and 2, we derive that $S_b'' = Y^T S'_b Y + Y^T S'_w Y = Y^T (S'_b + S'_w) Y = Y^T P^T (S_b + S_w) P Y = Y^T P^T S_t P Y = Y^T D_t Y$, for any vector α whose dimension is equal to that of S_b'', $\alpha^T S_b'' \alpha = \alpha^T Y^T D_t Y \alpha = (D_t^{1/2} Y \alpha)^T (D_t^{1/2} Y \alpha) \geq 0$, so S_b' 'is semi-positive. Suppose there exists α such that $\alpha^T S_b'' \alpha = 0$, then $D_t^{1/2} Y \alpha = 0$. By step1, we know D_t is full-rank, thus $Y\alpha = 0$. And by step2, we derive that Y is full-rank in columns since it is the extracted null space. Hence $\alpha = 0$, *iff.* $\alpha^T S_b'' \alpha = 0$. Therefore S_b'' is a positive matrix which is of course full-rank.

The third step is optional. Although it maximizes the between-class scatter in the null subspace, which appears to achieve best discriminative ability, it may incur overfitting. Because projecting all samples onto the null space of S_w is powerful enough in its clustering ability to archieve good generalization performance, step3 of algorithm I should be eliminated in order to avoid possible overfitting.

NLDA algorithm:

1. Remove the null space of S_t, i.e.

$$P^T S_t P = D_t,\ P^T S_w P = S'_w,$$

   P is usually $n \times (N-1)$.

2. Extract the null space of S_w', i.e.

$$Y^T S_w' Y = 0,$$

   Y is the null space, and is usually $(N-1) \times (c-1)$.

   The final NLDA projection matrix is:

$$W = PY,$$

PY is the discriminative subspace of the whole null space of S_w and is really useful for discrimination. The number of the optimal discriminant vectors is usually c-1, which just coincides with the number of ideal discriminant vectors [10,11]. Therefore, removing step3 is a feasible strategy against overfitting.

Under situation, S_t is full-rank and step1 of the NLDA algorithm is skipped. The NLDA projection can be extracted by performing eigen-analysis on S_w directly. The procedure of NLDA under this situation is most straightforward and only requires

one eigen-analysis. We can discover that NLDA will save much computational cost under the most suitable situation it is applicable to [24].

4. SIMILARITY & DISTANCE MEASURES

Once images are projected into a subspace, there is the task of determining which images are most like one another. There are two ways in general to determine how alike images are. One is to measure how similar two images are. When measuring distance, one wishes to minimize distance, so two images that are alike produce a small distance. When measuring similarity, one wishes to maximize similarity, so that two like images produce a high similarity value [10]. There are many possible similarity and distance measures; I will discuss five.

L_1 norm: The L_1 norm is also known as the city block norm or the sum norm. It sums up te absolute difference between pixels [10]. The L_1 norm of an image A and an image B is:

$$L_1(A,B) = \sum_{i=1}^{N} \left| A_i - B_i \right| \qquad \textbf{(4.1)}$$

The L1 norm is a distance measure.

L_2 norm: The L_2 norm is also known as the Euclidean norm or the Euclidean distance when its square root is calculated. It sums up the squared difference between pixels [10]. The L_2 norm of an image A and an image B is:

$$L_2(A,B) = \sum_{i=1}^{N} (A_i - B_i)(A_i - B_i) \qquad \textbf{(4.2)}$$

The L_2 norm is a distance measure.

Covariance: Covariance is also known as the angle measure. It calculates the angle between two normalized vectors [10,7,13]. Taking the dot product of the normalized vectors performs this calculation. The covariance between images A and B is:

$$\text{cov}(A,B) = \frac{A}{\|A\|} \bullet \frac{B}{\|B\|} \qquad \textbf{(4.3)}$$

Covariance is a similarity measure. By negating the covariance value, it becomes a distance measure.

Mahalanobis distance: The Mahanolobis distance calculates the product of the pixels and the eigenvalue of a specific dimension and sums all these products [10].

$$C_i = \frac{1}{\sqrt{\lambda_i}}$$

The Mahanolobis distance between an image A and an image B is:

$$Mah(A,B) = -\sum_{i=1}^{N} A_i B_i C_i \qquad \textbf{(4.4)}$$

Mahalanobis distance is a distance measure.

Correlation: Correlation measures the rate of change between the pixels of two images. It produces a value ranging from –1 to 1, where a value of –1 indicates the images are opposites of each other and a value of 1 indicates that the images are identical [10]. The correlation between an image A and an image B is:

$$corr(A,B) = \sum_{i=1}^{N} \frac{(A_i - \mu_A)(B_i - \mu_B)}{\sigma_A \sigma_B} \qquad \textbf{(4.5)}$$

where μ_A is the mean of A and σ_A is the standart deviation of A.

5. EXPERIMENTS

The YALE Face Database contains 165 grayscale images in GIF format of 15 individuals. There are 11 images per subject, one per different facial expression or configuration: center-light, w/glasses, happy, left-light, w/no glasses, normal, right-light, sad, sleepy, surprised and wink.

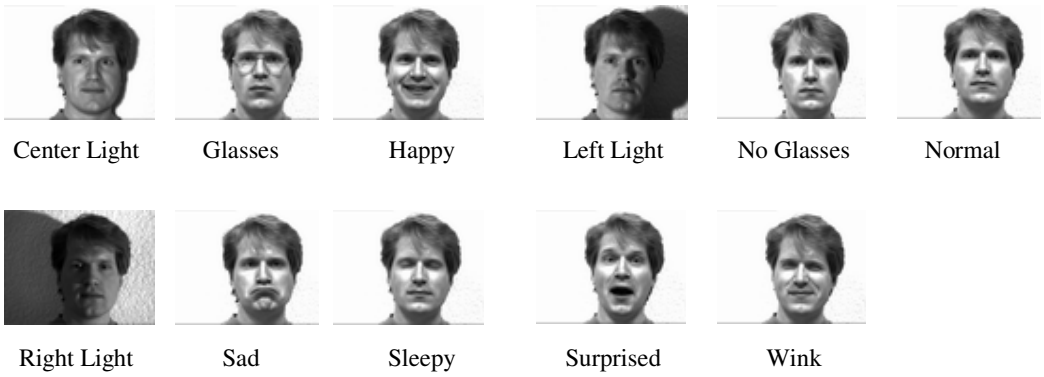

Figure 5.1: An example from the YALE Face Database

The ORL Face Database contains 400 images, 10 pictures of 40 people.

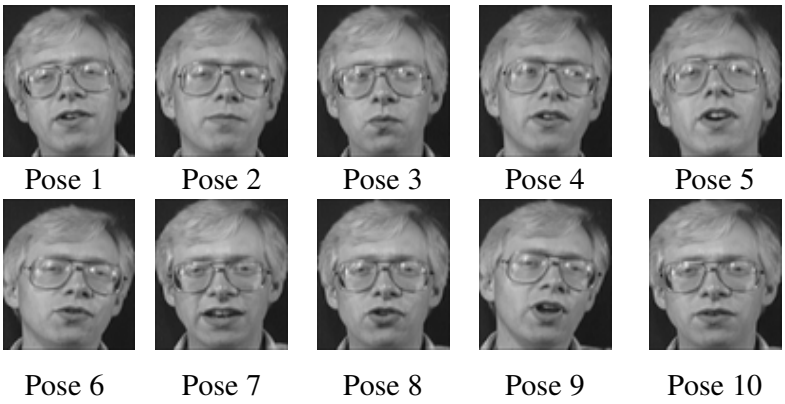

Figure 5.2: An example from the ORL Face Database

FERET Database 1 contains 160 pictures of 40 people. For each people there are 4 pictures.

Figure 5.3: An example from the FERET Database 1 (Normalized FERET)

FERET Database 2 contains 90 pictures of 15 people. For each people there are 6 pictures.

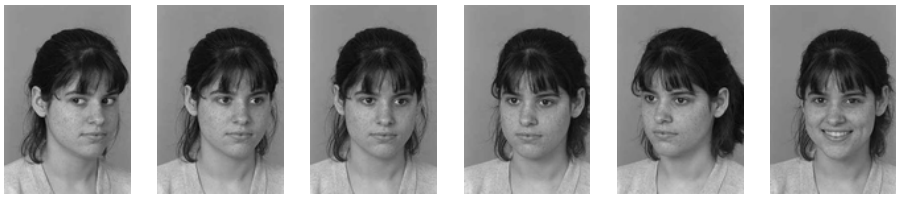

Figure 5.4: An example from the FERET Database 2

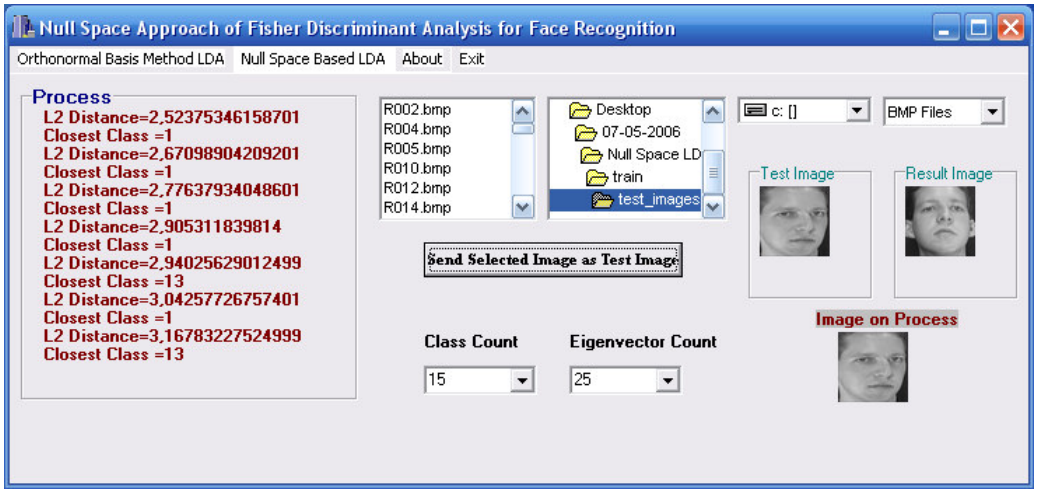

Figure 5.5: The Application

Experiment results for the YALE, ORL, FERET Database1 and FERET Database 2 are shown in tables Table 5.1 to Table 5.13. Each database is tested for different subject counts such as 10, 15 , 20 peoples. Tests are made on different methods such as FLDA, DLDA, NLDA, HEFLDA, HEDLDA and HENLDA. The results below show the success rates obtained by these tests.

Table 5.1: HEFLDA : YALE Face Database Success Rate for 15 People
(For each people 10 train 1 test images)

	HEFLDA: YALE EXPERIMENT										
Eigenvector Count	**center-light**	**w/glasses**	**happy**	**leftlight**	**w/no glassess**	**normal**	**right-light**	**sad**	**sleepy**	**surprised**	**wink**
15	53,33%	73,33%	86,67%	60%	93,33%	80%	53,33%	93,33%	86,67%	80%	86,67%
20	53,33%	73,33%	86,67%	60%	93,33%	93,33%	53,33%	93,33%	86,67%	80%	86,67%
25	86,67%	86,67%	93,33%	80%	93,33%	93,33%	60%	93,33%	93,33%	93,33%	100%
30	86,67%	86,67%	93,33%	80%	93,33%	100%	80%	93,33%	93,33%	93,33%	100%

Table 5.2: HEDLDA: YALE Face Database Success Rate for 15 People
(For each people 10 train 1 test images)

	HEDLDA : YALE EXPERIMENT										
Eigenvector Count	**center-Light**	**w/glasses**	**happy**	**leftlight**	**w/no glassess**	**normal**	**right-light**	**sad**	**sleepy**	**surprised**	**wink**
15	46,7%	60%	86,67%	46,7%	80%	80%	53,33%	86,67%	93,33%	80%	86,67%
20	53,3%	60%	93,33%	46,7%	93,33%	93,33%	53,33%	86,67%	93,33%	93,33%	93,33%
25	86,67%	73,33%	93,33%	73,33%	93,33%	100%	73,33%	93,33%	100%	93,33%	100%
30	86,67%	86,67%	93,33%	73,33%	93,33%	100%	73,33%	93,33%	100%	93,33%	100%

Table 5.3: HENLDA : YALE Face Database Success Rate for 15 People
(For each people 10 train 1 test images)

	HENLDA : YALE EXPERIMENT										
Eigenvector Count	**center-light**	**w/glasses**	**happy**	**leftlight**	**w/no glassess**	**normal**	**right-light**	**sad**	**sleepy**	**surprised**	**wink**
15	46,7%	60%	86,67%	60%	80%	80%	53,33%	86,67%	100%	80%	86,67%
20	53,3%	60%	100%	60%	93,33%	100%	60%	93,33%	100%	100%	100%
25	86,67%	73,33%	100%	80%	93,33%	100%	80%	93,33%	100%	93,33%	100%
30	93,33%	86,67%	100%	86,67%	93,33%	100%	86,67%	93,33%	100%	93,33%	100%

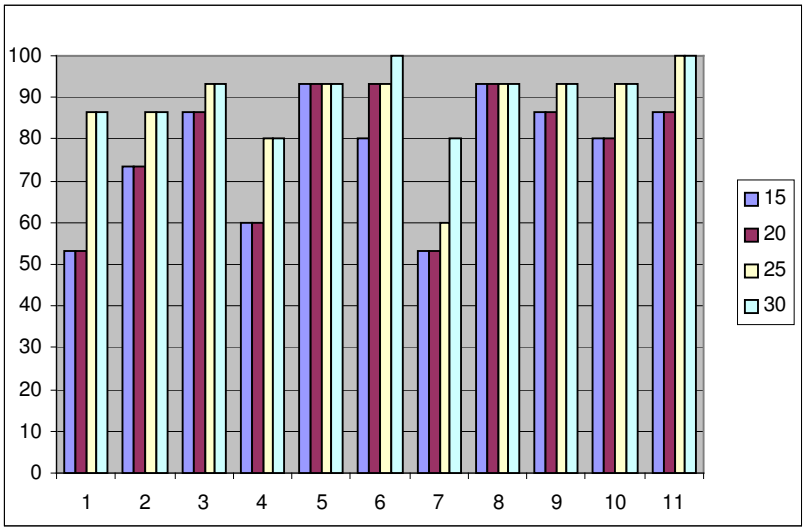

Figure 5.6: HEFLDA: YALE Face Database Success Rate for 15 People

In Figure 5.6 histogram equalization applied Fisher LDA results for the YALE Face Database in Table 5.1 are shown. The "leave one out" method is used for this experiment. For the illimunation changes such as center-light, right-light, left-light the results are good enough if we want to compare them without histogram equalization. When we choose more eigenvectors the success rate obtained is increases.

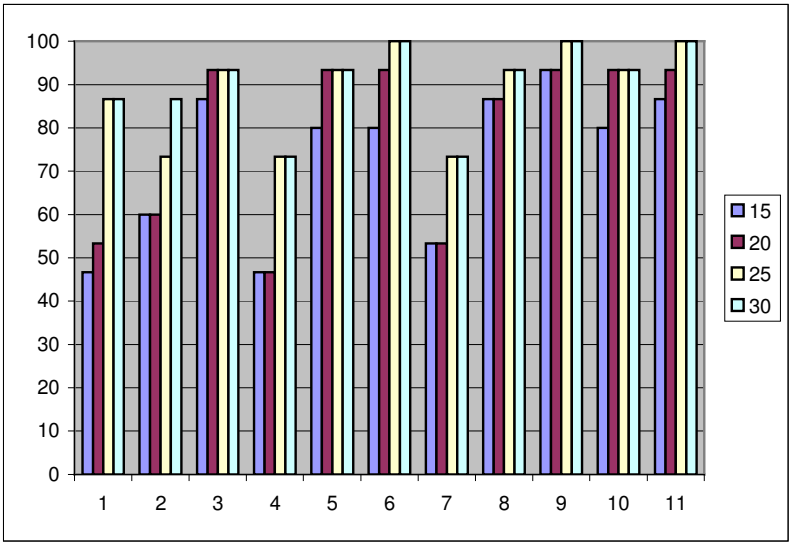

Figure 5.7: HEDLDA: YALE Face Database Success Rate for 15 People

In Figure 5.7 histogram equalization applied Direct LDA method results for the YALE Face Database in Table 5.2 are given. When we compare the results for the

Figure 5.6 we can say that Direct LDA performs better than Fisher LDA on YALE Face Database. The results for the illimunation changes are not good enough if we want to compare them for the face expressions such as sad, happy, wink.

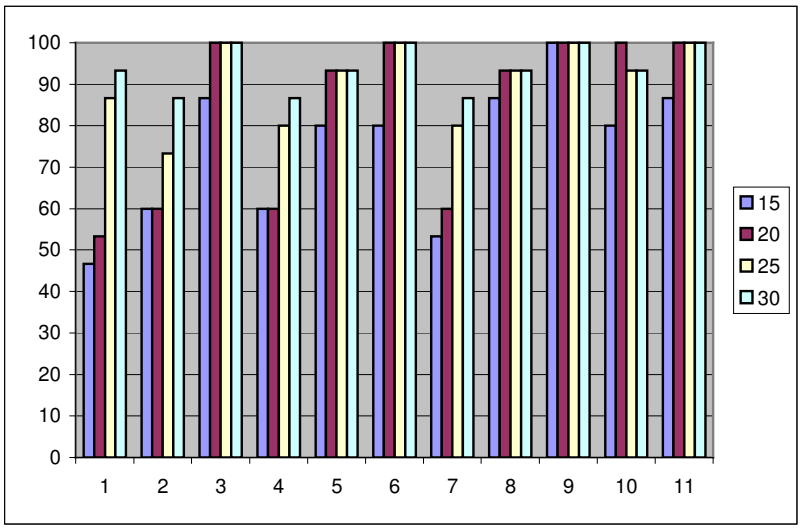

Figure 5.8: HENLDA: YALE Face Database Success Rate for 15 People

In Figure 5.8 histogram equalization applied Null Space LDA method results for the YALE Face Database in Table 5.3 are given. When we compare the results in Table 5.1 and in Table 5.2 we can conclude that Null Space LDA performs best. The success rates for especially illimunation changes increased. For the face expressions such as happy, normal, sleepy and wink the success rate reaches to top.

Table 5.4: HEFLDA : ORL Face Database Success Rate for 15 People
(For each people 9 train 1 test images)

	HEFLDA: ORL EXPERIMENT									
Eigenvector Count	**Pose 1**	**Pose 2**	**Pose 3**	**Pose 4**	**Pose 5**	**Pose 6**	**Pose 7**	**Pose 8**	**Pose 9**	**Pose 10**
15	100%	93,33%	93,33%	93,33%	100%	93,33%	100%	100%	100%	93,33%
20	100%	93,33%	93,33%	100%	100%	93,33%	100%	100%	100%	93,33%
25	100%	100%	100%	100%	100%	100%	100%	100%	100%	86,67%
30	100%	100%	100%	100%	100%	100%	100%	100%	100%	86,67%

Table 5.5: HEDLDA: ORL Face Database Success Rate for 15 People
(For each people 9 train 1 test images)

	HEDLDA: ORL EXPERIMENT									
Eigenvector Count	**Pose 1**	**Pose 2**	**Pose 3**	**Pose 4**	**Pose 5**	**Pose 6**	**Pose 7**	**Pose 8**	**Pose 9**	**Pose 10**
15	100%	100%	93,33%	100%	93,33%	100%	93,33%	100%	100%	93,33%
20	100%	93,33%	93,33%	100%	93,33%	100%	93,33%	100%	100%	93,33%
25	100%	100%	100%	100%	93,33%	100%	93,33%	100%	100%	86,67%
30	100%	100%	100%	100%	93,33%	100%	93,33%	100%	100%	86,67%

Table 5.6: HENLDA : ORL Face Database Success Rate for 15 People
(For each people 9 train 1 test images)

	HENLDA: ORL EXPERIMENT									
Eigenvector Count	**Pose 1**	**Pose 2**	**Pose 3**	**Pose 4**	**Pose 5**	**Pose 6**	**Pose 7**	**Pose 8**	**Pose 9**	**Pose 10**
15	100%	100%	93,33%	100%	100%	100%	100%	100%	100%	100%
20	100%	93,33%	100%	100%	100%	100%	100%	100%	100%	86,67%
25	100%	100%	100%	100%	100%	100%	100%	100%	100%	86,67%
30	100%	100%	100%	100%	100%	100%	100%	100%	100%	93,33%

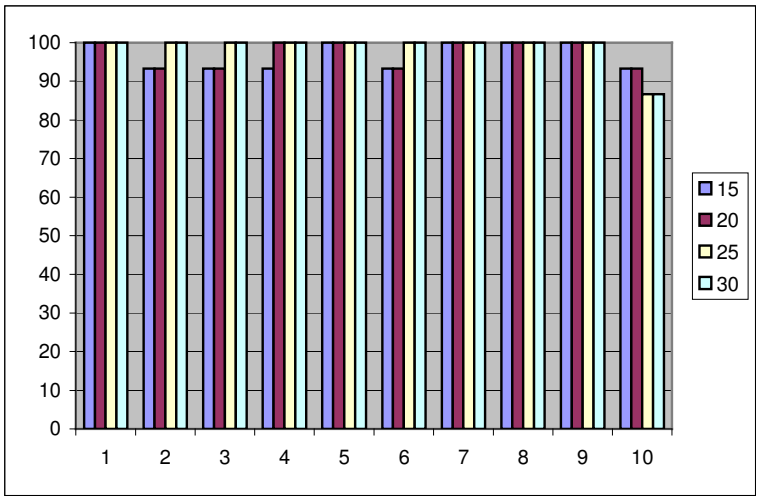

Figure 5.9: HEFLDA: ORL Face Database Success Rate for 15 People

In Figure 5.9 histogram equalization applied Fisher LDA method results for the ORL Face Database in Table 5.4 are given. “Leave one out” method is used for the experiments. When we compare the results in Figure 5.6 and Figure 5.9, we can easily say that HEFLDA method on ORL Face Database performs better for than YALE Face Database. For 30 eigenvectors selected nearly all results reaches 100 percent.

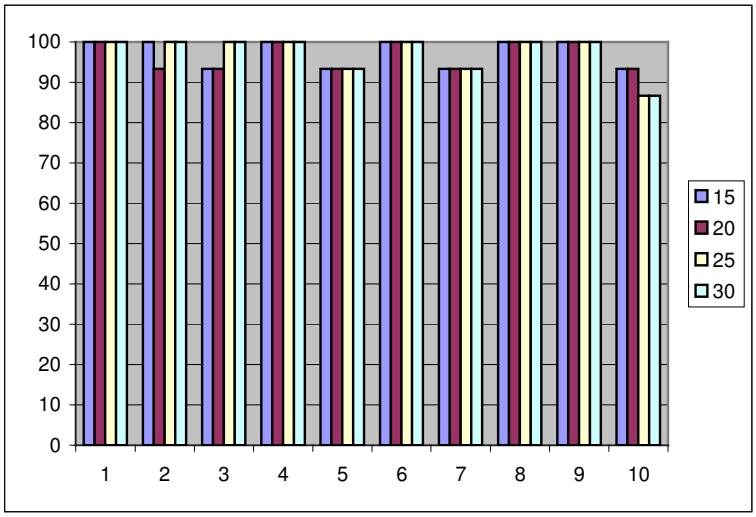

Figure 5.10: HEDLDA: ORL Face Database Success Rate for 15 People

In Figure 5.10 histogram equalization applied Direct LDA method results for the ORL Face Database in Table 5.5 are given. When we compare the results in Figure 5.7 and Figure 5.10, we can easily say that HEDLDA method on ORL Face

Database performs better than YALE Face Database. If we want to compare the results in Figure 5.9 and Figure 5.10, it can be seen that HEFLDA performs better than HEDLDA on ORL Face Database.

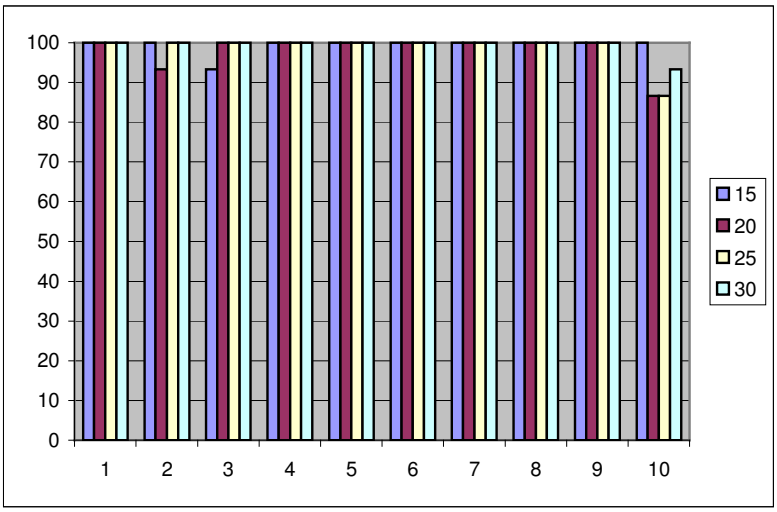

Figure 5.11: HENLDA: ORL Face Database Success Rate for 15 People

In Figure 5.10, histogram equalization applied Null Space LDA method results for the ORL Face Database in Table 5.6 are given. When we compare the results in Figure 5.8 and Figure 5.11 we can easily say that HENLDA method on ORL Face Database performs better than YALE Face Database. If we want to compare the results in Figure 5.9, Figure 5.10 and Figure 5.11 we can conclude that HENLDA and HEFLDA performs better than HEDLDA on ORL Face Database. Furthermore the success rates obtained from ORL Database are better than the results obtained from YALE Database.

Table 5.7: FERET Database 1 Success Rates for 15 People
(For each people 3 train, 1 test images)

	Eigenvector Count			
	15	**20**	**25**	**30**
FLDA	86,67%	86,67%	93,33%	93,33%
DLDA	86,67%	86,67%	86,67%	86,67%
NLDA	86,67%	93,33%	93,33%	93,33%
HEFLDA	86,67%	93,33%	86,67%	93,33%
HEDLDA	86,67%	86,67%	86,67%	93,33%
HENLDA	93,33%	86,67%	93,33%	93,33%

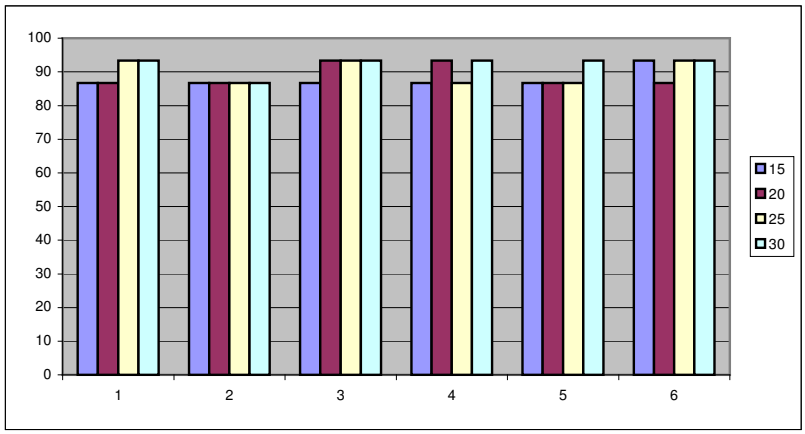

Figure 5.12: FERET Database 1 Success Rates for 15 People

Table 5.8: FERET Database 1 Success Rates for 20 People
(For each people 3 train, 1 test images)

	Eigenvector Count			
	15	**20**	**25**	**30**
FLDA	90%	90%	95%	95%
DLDA	85%	85%	90%	90%
NLDA	90%	95%	90%	95%
HEFLDA	90%	95%	90%	95%
HEDLDA	85%	85%	90%	95%
HENLDA	90%	90%	95%	95%

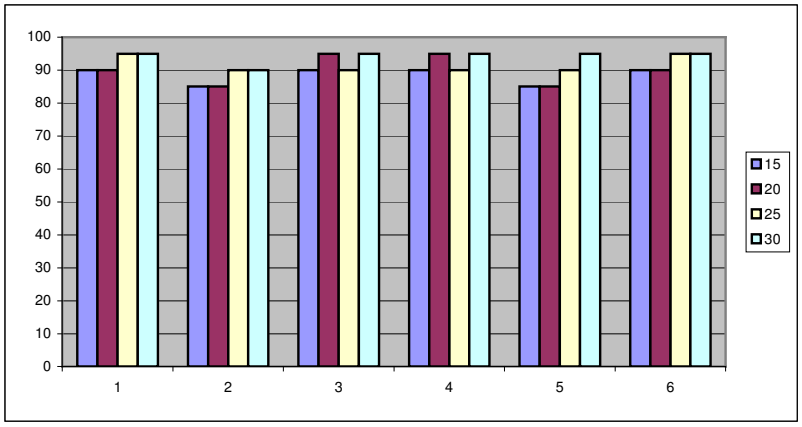

Figure 5.13: FERET Database 1 Success Rates for 20 People

Table 5.9: FERET Database 1 Success Rates for 25 People
(For each people 3 train, 1 test images)

	Eigenvector Count			
	15	**20**	**25**	**30**
FLDA	88%	88%	92%	92%
DLDA	84%	84%	88%	88%
NLDA	88%	88%	92%	92%
HEFLDA	88%	92%	92%	92%
HEDLDA	84%	88%	88%	88%
HENLDA	92%	92%	96%	92%

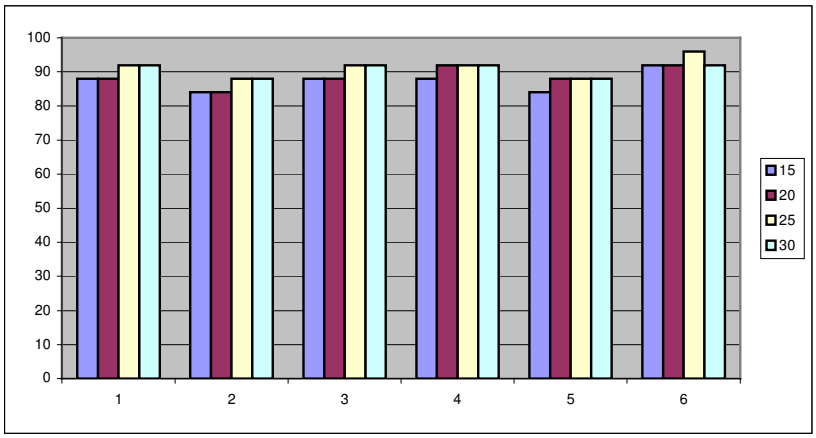

Figure 5.14: FERET Database 1 Success Rates for 25 People

In Figure 5.12 to Figure 5.14 there are different numbers of samples (15, 20, 25). From the figures the numbers corresponds the test results for the below given.

1) FLDA, 2) DLDA, 3) NLDA, 4) HEFLDA, 5) HEDLDA, 6) HENLDA

Generally success rates decreases when we select more number of samples in training set. When we carefully analize the results obtained from these figures we can conclude that histogram equalization applied training samples performs better success rates than the others. Histogram equalization increases success rate nearly 3 percent. HENLDA performs best when we want to compare it with the others. However DLDA performs worst. The success rates obtained from these figures are better than YALE Face Database because the training images (samples) are normalized before the test is done. However the results are not good enough when compared with the results obtained from ORL Face Database. Finally we can conclude that the more eigenvector selected the more success rate obtained.

Table 5.10: FERET Database 2 Success Rates for 15 People
(For each people 4 train, 2 test images)

	Eigenvector Count			
	15	**20**	**25**	**30**
FLDA	84,85%	84,85%	87,88%	90,09%
NLDA	90,09%	84,85%	90,09%	90,09%
HEFLDA	84,85%	87,88%	87,88%	90,09%
HENLDA	90,09%	93,33%	93,33%	93,33%

Table 5.11: YALE Database Success Rates for Different Face Expressions for 15 People
For each people 6 train (glasses, no-glasses, normal, center-light, left-light, right-light)
5 test (happy, sad, sleepy, surprised, wink) images

	Eigenvector Count			
	15	**20**	**25**	**30**
FLDA	82,66%	82,66%	88%	92%
NLDA	82,66%	88%	92%	92%
HEFLDA	92%	92%	93%	93%
HENLDA	92%	92%	92%	92%

Table 5.12: YALE Database Success Rates for Illumination Changes for 15 People
For each people 8 train, 3 test (center-light, left-light, right-light) images
Train images are happy, sad, sleepy, surprised, wink, with-glasses, no-glasses and normal

	Eigenvector Count			
	15	**20**	**25**	**30**
FLDA	66,67%	66,67%	66,67%	71,10%
NLDA	71,10%	66,67%	71,10%	71,10%
HEFLDA	73,30%	73,30%	77,70%	77,70%
HENLDA	66,67%	86,60%	86,60%	77,70%

Table 5.13: ORL Database success rates for different poses for 15 People
For each people 6 train, 4 test images
Pose 1, Pose 3, Pose 6, Pose 7, Pose 8 and Pose 9 are train images
Pose 2, Pose4, Pose 5 and Pose 10 are test images

	Eigenvector Count			
	15	**20**	**25**	**30**
FLDA	88,33%	88,33%	88,33%	90%
NLDA	88,33%	88,33%	90%	90%
HEFLDA	88,33%	90%	90%	91,67%
HENLDA	90%	90%	91,67%	91,67%

6. CONCLUSION

Face recognition is one of the several techniques for recognizing people. There are several methods that can be used for that purpose. Some of the most common are using PCA or eigenfaces. There are other new techniques more simple to understand use and implement but also with very good performance. The null space-based LDA takes full advantage of the null space. It proves to be optimal in performance.

In this thesis the PCA and LDA methods are used for face recognition. Null-space approach is also used to improve recognition performance. The YALE, ORL and FERET face databases are used for test and train images. Histogram equalization applied train and test images before FLDA and NLDA experiments to overcome the illumination problems. Finally from the result of experiments when we choose the more number of eigenvectors, the more success rate obtained. When we take the more number of the classes the less success rate obtained. Furthermore face expression and illumination changes successfully recognized by my application.

REFERENCES

[1] **Belhumeur P., Hespanha J., and Kriegman D**., 1997. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, *IEEE Trans. PAMI*, 19(7):711-720.

[2] **Duda R. and Hart P.**, 1973. *Pattern Classification and Scene Analysis*, New York: John Wiley & Sons.

[3] **Fisher, R.A.,** 1936. The use of Multiple Measures in Taxonomic Problems, *Ann. Eugenics*, 7:179-188.

[4] **Horn R. and Johnson C.**, 1985. *Matrix Analysis,* New York: Cambridge University.

[5] **Kirby M.,** 2000. *Dimensionally of Reduction and Pattern Analysis: an empirical approach.* Under contract with Wiley.

[6] **Kirby M. and Sirovich L.,** 1990. Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces, *IEEE Trans. PAMI*, 12(1):103-108.

[7] **Moon H. and Phillips J.**, 1998. Analysis of PCA-based Faced Recognition Algorithms. In Boyer K. and Phillips J., editors, *Empirical Evaluation Techniques In Computer Vision*, IEEE Computer Society Press, Los Alamitos, CA.

[8] **Phillips J., Moon H., Rizvi S., and Rauss P.**, 1998. The FERET Evaluation. In Wechslet H., Phillips P., Bruse, V., Fogeliman Soulie F., and Hauhg T., editors, *FaceRecognition: From Theory to Application*, Springer-Verlag, Berlin.

[9] **Phillips J., Moon H., Rizvi S., and Rauss P.,** 1999. The FERET Evaluation Methodology for Face-Recognition Algorithms. *NIST Technical Report NISTIR 6264*.

[10] **Press W., Teukolsky S., Vetterling W., and Flannery B.**,1988. *Numerical Recipes in C, The Art of Scientific Computing,* New York: Cambridge University Press, Inc.

[11] **Stevens M.,** 1999. Reasoning about Object Appearance in terms of a Scene Context. *Ph.D. thesis*.

[12] **Swets D. and Weng J.,** 1996. Using Discriminant Eigenfeatures for Image Retrieval. *IEEE Trans. PAMI,* 18(8):831-836.

[13] **Turk M. A. and Pentland P.**, 1991. Face Recognition Using Eigenfaces, *Proc. Of IEEE Conference on Computer Vision and Pattern Recognition,* 586-591.

[14] **Yambor W., Draper B., and Beveridge J.R.**, 2000. Analysis of PCA-based Face Recognition Algorithms: Eigenvector Selection and Distance Measures. *Second Workshop on Empirical Evaluation Methods in Computer Vision*.

[15] **Zhao W., Krishnaswamy A., Chellappa R., Swets D., and Weng J.**, 1998. Discriminant Analysis of Principle Components for face Recognition. *3rd International Conference on Automatic Face and Gesture Recognition*, 336-341.

[16] **Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.,** 1997. "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection". IEEE Trans. Pattern Analysis and Machine Intelligence. Volume 19. Issue 7. 711-720.

[17] **Swets, D.L., Weng, J.,** 1996. "Using Discriminant Eigenfeatures for Image Retrieval". IEEE Trans. Pattern Analysis and Machine Intelligence. Volume 18. Issue 8. 831-836.

[18] **Zhao, W., Chellappa, R., Phillips, P.J.,** 1999. "Subspace Linear Discriminant Analysis for Face Recognition". Technical Report CAR-TR-914, Center for Automation Research, University of Maryland.

[19] **Yang, J., Yu, H., Kunz, W.,** 2000. "An Efficient LDA Algorithm for Face Recognition". In: Proceedings of the Sixth International Conference on Control, Automation, Robotics and Vision.

[20] **Chen, L.F., Liao, H.Y.M., Lin, J.C., Ko, M.T., Yu, G.J.,** 2000. "A New LDA-based Face Recognition System Which Can Solve the Small Sample Size Problem". Pattern Recognition. Volume 33. Issue 10. 1713-1726.

[21] **Huang, R., Liu, Q.S., Lu, H.Q., Ma, S.D.,** 2002. "Solving the Small Sample Size Problem of LDA". In: Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02), Quebec, Canada.

[22] **Phillips, P.J., Moon, H., Rizvi, S., Rauss, P.,** 2000. "The FERET Evaluation Methodology for Face Recognition Algorithms". IEEE Trans.

Pattern Analysis and Machine Intelligence. Volume 22. Issue 10. (2000) 1090—1104.

[23] **Baudat, G., Anouar, F.,** 2000. "Generalized Discriminant Analysis Using a Kernel Approach". Neural Computation. Volume 12. Issue 10. 2385-2404.

[24] **Wei Liu, Yunhong Wang, Stan Z. Li, Tieniu Tan.,** 2001. "Null Space-based Kernel Fisher Discriminant Analysis for Face Recognition". National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, P.R. China.

[25] **Dake Zhou and Xin Yang.,** 2001. "Face Recognition Using Direct-Weighed LDA". Institute of Image Processing & Pattern Recognition, Shanghai Jiaotong University.

[26] **Juwei Lu, Kostantinos N. Plataniotis, and Anastasios N.Venetsanopoulos.,** 2003. "Face Recognition Using LDA-Based Algorithms" IEEE Transactions on Neural Networks, Vol 14. No 1.

RESUME

Kürşat Ağır was born in 1979, in Kars. He was graduated from Kars Anatolian High School. In 2002, he was gratuated from Istanbul Technical University Control & Computer Engineering Department and he started the master program in Computer Engineering Department at the same university. He is now working at TUBITAK UEKAE as a computer engineer.