Vietnamese – German University
Department of Electrical Engineering & Information Technology

Frankfurt University of Applied Science
Faculty 2: Computer Science and
Engineering

ADVANCED METHODS FOR FACE RECOGNITION

BY

Nguyễn Gia Khánh

Matriculation number: 1148563

BACHELOR THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor Engineering in study program
Electrical Engineering & Information Technology,
Vietnamese – German University

First supervisor: Dr. Nguyen Minh Hien
Second supervisor: M.Sc. Bien Minh Tri

Binh Duong, Vietnam, November 2018

Advanced Methods for Face Recognition

Approved by

_____

Dr. Nguyen Minh Hien, First Supervisor

_____

M.Sc. Bien Minh Tri, Second Supervisor

Thesis Committee

Disclaimer

I declare that this thesis is a product of my work under the supervision of Dr. Nguyen Minh Hien, unless otherwise referenced. I also declare that to my best knowledge, no part has been submitted here or elsewhere. All opinions, results, conclusions and recommendations are my own.

Signature 5th November 2018

……..……

Nguyễn Gia Khánh

Abstract

This thesis presents two popular techniques for detecting and recognizing of human faces which is totally able to setup and implement in real-time face recognition system. They could be used to identify a face of an object, determine if that is a human face or not and recognize the person by comparing main characteristics of his/her face to images of known people in the database. The main idea shared by these algorithms is that a face would be processed as a two-dimensional mathematical matrix before passing through some calculations to find its main features. All faces images are then projected on the feature space ("face space") to find the corresponding coordinators. The face space is composed of "Eigenfaces" or "Fisherfaces" which are actually eigenvectors found after doing a matrix composition - Eigendecomposition. At the heart of Eigenface method is the Principal Component Analysis (PCA) - one of the most popular unsupervised learning algorithms - while Fisherface is a better version of the previous one which makes use of both Principal Component Analysis and Linear Discrimination Analysis (LDA) to get more reliable results. Both methods would be examined deeply in their working principles as well as their potential applications in reality before coming to conclusion about the advantage and drawback of each one.

The algorithms were realized by Python 3.7 with a Graphical User Interface (GUI). Given initial images in the database, the program can detect and recognize the human faces in the provided pictures before saving them in the database to improve the calculation accuracy in the future. After evaluation, the recognition general results are exported on the screen with details included in the text files.

This thesis consists of four main parts:

1. Introduction to Face Recognition.

2. Eigenface Method.

3. Fisherface Method.

4. Implementation and Evaluation.

Acknowledgements

I would like to express my deep appreciation to Dr. Nguyen Minh Hien – lecturer of Vietnamese German University for her explanation and guidance. In addition, I also want to thank Mr. Bien Minh Tri for the permission to work in his lab and his very useful advices.

Nguyễn Gia Khánh

Table of Contents

List of Figures

List of Abbreviations

PCA	Principal Component Analysis
SVD	Single Value Decomposition
RGB	Red-Green-Blue
2-D	Two dimensional
3-D	Three dimensional
LDA	Linear Discriminant Analysis
SVM	Support Vector Machine
LBP	Local Binary Pattern
HOG	Histogram of the Oriented Gradients
SIFT	Scale Invariant Feature Transformation
SOM	Self-Organizing Maps
CNN	Convolution Neural Network
CAGR	Compound Annual Growth Rate
FRGC	Face Recognition Grand Challenge
FPBM	Fast Pixel Based Matching
EBGM	Elastic Bunch Graph Matching
LEM	Line Edge Map
IDA	Independent Component Analysis
KFA	Kernel Fisher Analysis

KPCA	Kernel Principle Component Analysis
LLE	Local Linear Embedding
AAM	Active Appearance Model

1. Introduction to Face Recognition

1.1 Introduction

In many past decades, developing an accurate computational model of face recognition seemed to be impossible due to the complexity of the face nature as well as countless emotions it possesses. It is natural and not the same as sine waveform or anything which belong to the "blocks world" we have ever known and researched in the computer vision field. Face recognition is a quite simple task for our brain to perform, however, too difficult for the computer to stimulate or imitate.

Today, with the advancement of technology and the existence of more and more powerful processors, we have empowered all conditions and capabilities to invent and develop our own face recognition systems. Face recognition is no longer something existing in fictional novels or movies but becomes more and more familiar with us in daily life. One of the advantages of Face Recognition compared to fingerprints, eye iris and other biometric modalities is its simplicity to establish and maintain the images database. In addition, besides being more natural and nonintrusive, faces can be captured at a distance and in a covert manner.

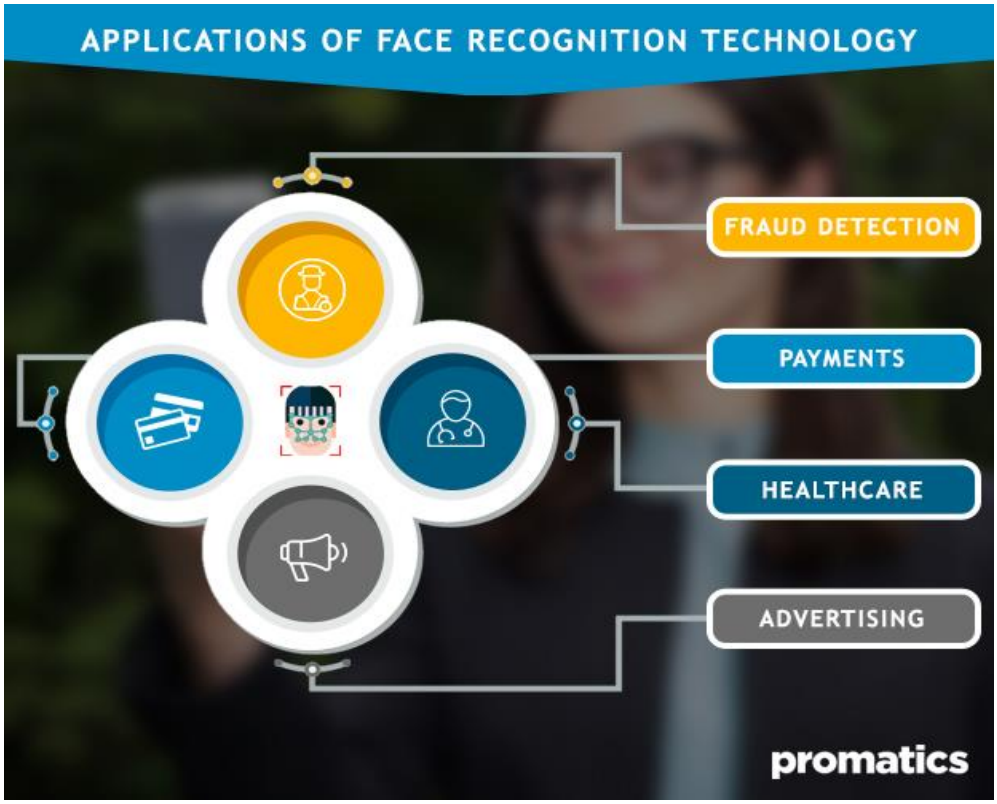

Figure 1.1. *Applications of Face Recognition [1].*

There are bunches of applications of this computer technique:

- Facebook People Tagging: tagging people by their faces.
- Microsoft Windows Hello: signing in to Windows by faces.
- Google Images Search: searching by images.
- Apple Face ID: unlocking iPhone by faces.
- Amazon Rekognition service: Amazon Face Recognition service
- Tesla Autopilot in the self-driving car: reducing accidents due to negligence and fatigue from long-term driving.
- Public camera surveillance: preventing and detecting criminals and illegal actions.
- Fraud detection.
- Digital camera: detecting and focusing on human faces for better photos.
- Human emotions analysis.
- Rescue robots: the capability to work and look for victims in extremely dangerous places or situations.
- Finding lost people.
- Payments, healthcare, security, advertising and marketing field.

Face recognition technology has now traveled a long way from the time when the Eigenface or Fisherface method was first presented. By 2022, the global market of facial recognition technology is expected to reach approximately $9.6 billion in revenue with 21.3% Compound Annual Growth Rate (CAGR) from 2016 to 2022. There are 3 segments in facial recognition industry: 3-D, 2-D and facial analytics. The 3-D facial recognition technology has become more and more popular to our lives with the existence of Face ID on Apple iPhone X and has been implemented in many modern buildings as well as secret military bases. This segment also holds the highest market shares and with the fastest growth rate in the next 10 years. Meanwhile, 2-D technology has still been ubiquitous owing to its simplicity and low installation/operation cost [2].

Figure 1.2. *Surveillance system used in Switzerland. It can recognize driver's face, identify vehicle model, color and read license plate [3].*

Today, the facial recognition technology could be carried out under widely varying conditions like (scaled) frontal or 45° view, subjects with spectacles or countless emotions on a face given a very limited range covered by the training data set (database). In the constrained situations, suppose that lighting, pose, stand-off, facial wear and facial expression are under controlled, an automated face recognition system can easily outperform human common recognition performance, provided that the database contains large enough faces images. However, the system still faces many obstacles when dealing with captured images under uncontrolled (non-ideal) environments.

The purpose of this thesis is to research and develop an efficient Python program for face recognition using Eigenface and Fisherface algorithm as well as perform many kinds of tests to study each technique's stability and accuracy in performance. These approaches are preferred to study because of their speed, simplicity and self-learning capability. The algorithms are also implemented to recognize faces from videos or real-time streaming, however, this is out of the scope of the current project. In the next parts, a brief overview of a general face recognition process would be presented, followed by working principles in details of Eigenface and Fisherface techniques as well as the comparison in performance between these methods.

1.2 Background and related works

Most of the earlier researches in computer face recognition field concentrated on identifying the specific features of a face such as the ear, scar, nose, mouth, head shape and trying to describe a definition for the model of a face specified by parameters like size, relative location, shape and relationships among those features.

Beginning with Bledsoe's [4] and Kanade's [5] early systems in 1964, many initial face recognition algorithms (automated or semi-automated) have been devised and developed which classify individual faces based on proportions and normalized distances among face feature points. Recently these approaches have been continued to maintain and enhance by Yuille et al. [6] which then have been considered the most popular techniques in the computer vision literature. However, they have shown many difficulties to deal with multiple views and have not always been trustworthy. The limitations of those methods imply that accounting for information about individual facial features as well as their relationships is not sufficient to perform a good recognition of standard adult faces whose results are accurate and reliable.

In 1980s, T.Kohonen [7] proposed a method using neural unsupervised learning process called Self-Organizing Maps (SOM), which maps the distribution of unknown class data points. This work was then developed by Lawrence et al. [8] based on Convolutional Neural Network (CNN). However, the serious problem this method has to cope with is its inability to handle high-dimensional space.

In 1987, Sirovich and Kirby [9] devised a method to represent a whole human face as a set of basic features. These features, known as Eigenfaces, have become to be popular in computer vision field due to their advantages: easy to establish and maintain the training data set, simple to understand and code, pretty fast and can be done in real-time with considerable large database, able to transform complex high-dimensional face images into the lower ones. On the other hand, the prices we had to pay are its heavy sensitivity to lighting, shifting and scaling of images, differences in facial looks (different angle, pose, glasses, emotions, hairstyle, makeup, mustache, beard, etc.) and aging.

In 1997, P.N. Belhumeur together with J.P. Hespanha and D.J. Kriegman [10] developed a new technique which is more insensitive to wide range variation in lighting direction and facial expression. This technique was named as "Fisherface" which is considered to be an "upgraded version" of Eigenface with better and more reliable results.

In 2006, Face Recognition Grand Challenge (FRGC) was commonly chosen to be a standard to evaluate many latest facial recognition algorithms which is sponsored by many departments of U.S. government. Iris images, 3-D face scan and high-resolution face images are used in this test. The final results at that time showed that new generation methods are 10 times

more accurate than the past ones of 2002 and 100 times than those of 1990s. Some of them could even surpass the performance of human's brain and correctly identify identical twins [11].

In 2017, Apple presented Face ID technology together with iPhone X, the first smartphone to integrate a fully-functional 3-D facial recognition system.

Figure 1.3. *Apple Face ID [12].*

1.3 Existing face recognition algorithms

A general flow chart of a typical biometric system (face recognition system) is proposed as below:

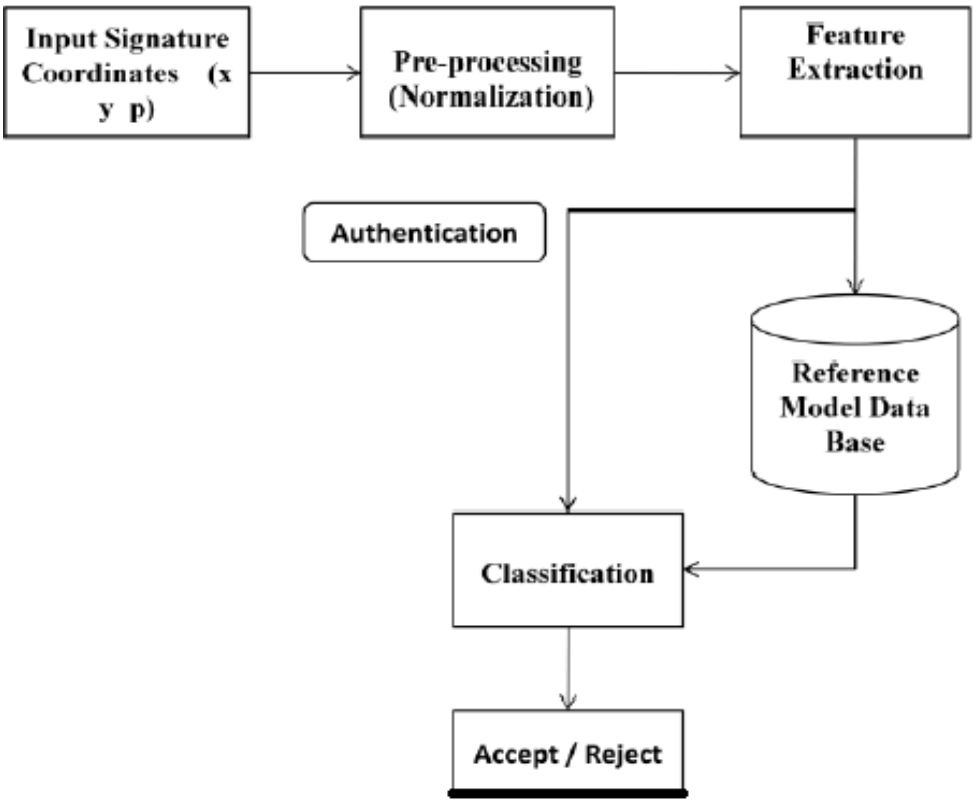

Figure 1.4. *Flow chart of a typical biometric system. [13].*

There are three main phases in a typical face recognition process: pre-processing (also called normalization), feature extraction and classification (by comparing the checking face to the existing face models in the database). In the pre-processing step, a face is detected and cropped from the given picture. Then the cropped face would be aligned properly with the ones in the database. The feature extraction step is responsible for extracting important features in form of geometric shapes or statistics. Finally, the classification step determines how much the checking image corresponds to faces in the database.

Above all, feature extraction plays the most important role which directly accounts for the processing time and total accuracy of the results. Many face recognition techniques differ from the others only in the algorithm used in this phase.

Until now, there are four main kinds of approaches for feature extraction phase: geometric feature-based and holistic-based methods [14].

Geometric feature-based suggests extracting only geometric face features as a dataset like eyes, eyebrows, mouth, ears or nose in order to form a set of feature vectors. On the other hand, in holistic or appearance-based approach, the whole face of a human has to be considered.

Based on these two approaches, many face feature extraction algorithms can be divided into four categories:

- Appearance-based using holistic texture features.
- Model-based considering shape and texture of the face, along with 3D depth information.
- Template-based face recognition.
- Algorithms using Convolution Neural Networks.

A summary of those algorithms could be illustrated in the below figure.

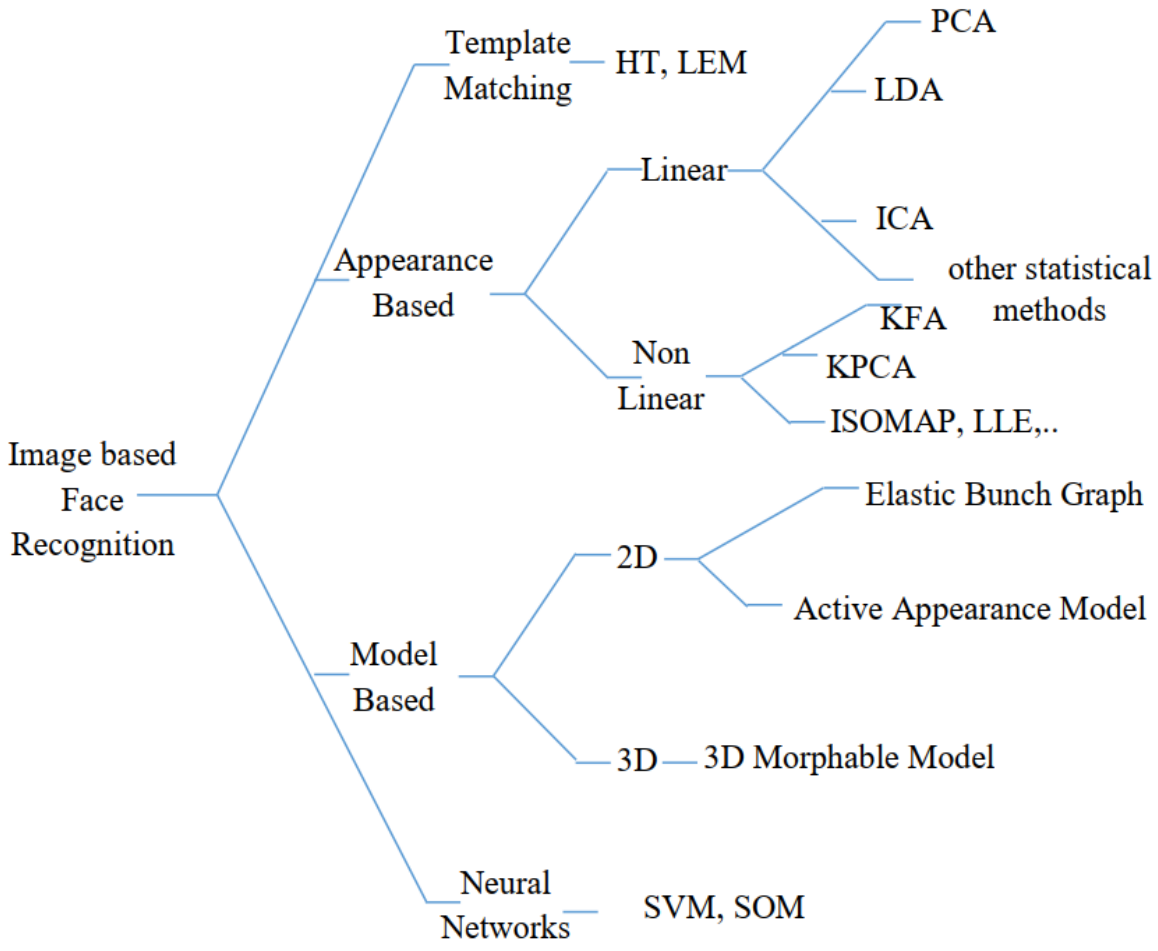

Figure 1.5. *Classification of face feature extraction algorithms [15].*

In the scope of this project, the Principal Component Analysis and Linear Discriminant Analysis will be thoroughly examined as the heart of the Eigenface and Fisherface methods. The research ultimate goal is to successfully answer three questions:

- Is the given image a known face or a completely new face?
- Eigenface and Fisherface, which one is better?
- What are the advantages and disadvantages of each method?

1.4 Digital image

A digital image is a matrix of pixels. Each pixel contains information about 2-D image represented by a series of code. They are arranged in order of a standard rectangular. Image size is defined to be the total number of rows and columns and is written in form of $M\ x\ N$. An image of size $M\ x\ N$ has M columns in width and N rows in height. The value held by a pixel is described as the intensity value of the image at that point (or brightness value). Standard digital image uses 8-bit to represent the intensity value, ranging from 0 to 255. Depending on the number of channels – normally the number of basic colors – used to represent the image, there are three main groups:

- One-channel image (black-white or grayscale image): The intensity value grows from black at the darkest (weakest) intensity to white at the lightest (strongest) intensity.
- Three-channel image (RGB, YUV, HSV): RGB uses 3 basic colors which are Red, Green and Blue. In YUV format, Y is the luminance (brightness) factor while U and V are the chrominance (color) factors of the whole image. HSV uses Hue Saturation Value to store information of the image. Hue and Saturation are responsible for the color components while Value holds the value of brightness.
- Four-channel image (RGBA, CMYK): RGBA is simply RGB with the fourth channel is Alpha describing the transparency. CMYK stands for Cyan, Magenta, Yellow and Key (black). CMYK is commonly used in the color printing industry.

In this project, only grayscale images are considered for the purpose of reducing the computation time.

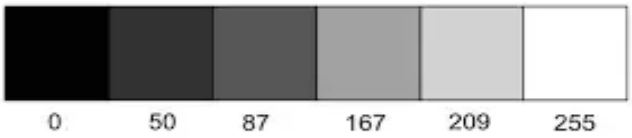

Figure 1.6. *Grayscale [16].*

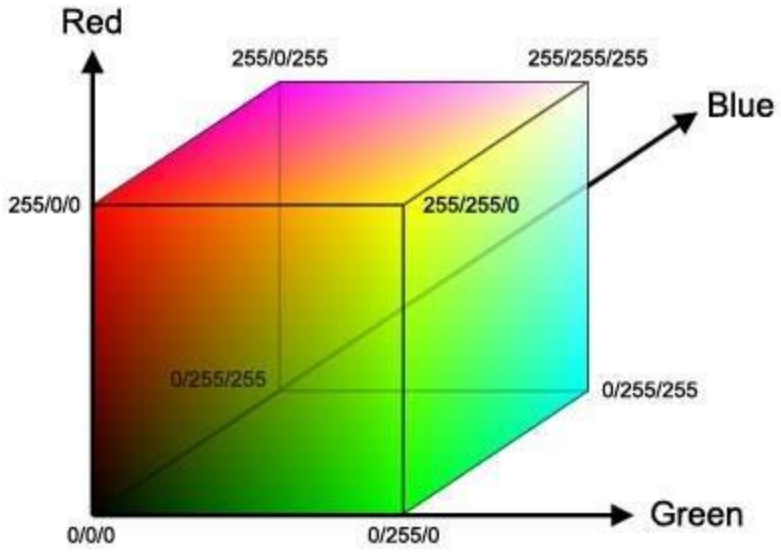

Figure 1.7. *3-D RGB Color Space [17].*

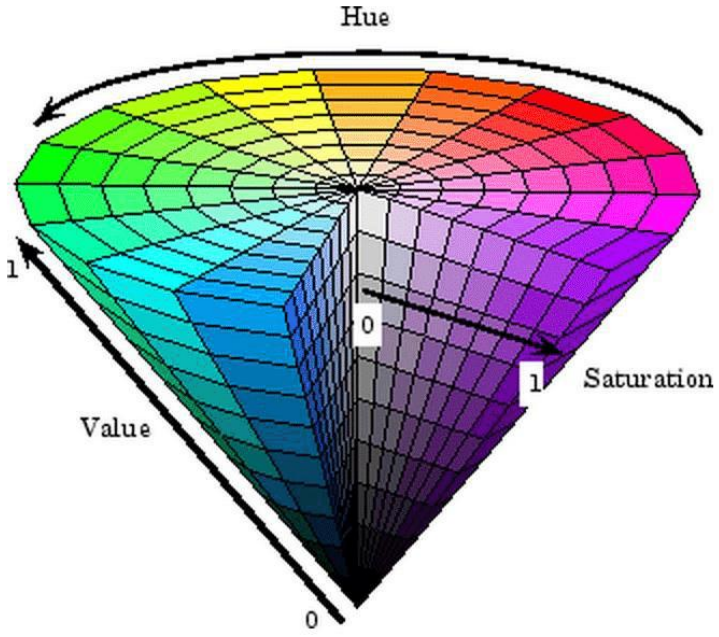

Figure 1.8. *3-D HSV Color Space [18].*

Figure 1.9. *Example of RGBA image with different colors and transparencies [19].*

2. Eigenface Method

2.1 Principal Component Analysis (PCA)

In mathematics, PCA is defined as an orthogonal linear transformation method which transforms data points from the original coordinate system to a new one in such a way that the largest data projection variance is on the first principal component (the first coordinate), the second largest variance on the second coordinate and so on.

When each repetition of an experiment yields the same number of distinct values p (such as the results from various sensors), the data can be arranged into a matrix $\mathbf{X}$. Its n rows record the experimental results (in any order). Its p columns correspond to the different values or variables.

Often, before applying PCA, $\mathbf{X}$ is modified by "centering" the variables: the mean of each column is subtracted from each value in that column.

PCA converts $\mathbf{X}$ into a matrix $\mathbf{T}$ of the same dimensions. It replaces each row of experimental results $\mathbf{x}(i) = (x_1, \dots, x_p)_{(i)}$ by a row of "principal component scores", which are the values of new variables $\mathbf{t}(i) = (t_1, \dots, t_p)_{(i)}$. Each score is a linear combination of the entries in its row, written as the dot product with a "loading vector" $\mathbf{w}(i) = (w_1, \dots, w_p)_{(i)}$,

$$t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}. \tag{1}$$

$\mathbf{w}_{(1)}$ is chosen to be any unit vector that maximizes the variance of the column of scores $t_{1(i)}$ it produces. (The choice is not unique). $\mathbf{X}$ is then modified by subtracting the score-scaled loading vectors $t_{1(i)}\mathbf{w}_{(1)}$ from the rows. This process is repeated until all p of the loading vectors and the *p*-associated scores have been found.

2.1.1 First principal component

To achieve the maximum variance, the first loading vector $\mathbf{w}_{(1)}$ must satisfy the following condition:

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1}\left\{\sum_{i=1}(t_1)^2_{(i)}\right\} = \arg\max_{\|\mathbf{w}\|=1}\left\{\sum_{i=1}\left(\mathbf{x}_{(i)} \cdot \mathbf{w}\right)^2_{(i)}\right\}. \tag{2}$$

Alternatively, we can write this in matrix form:

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1}\{\|\mathbf{X}\mathbf{w}\|^2\} = \arg\max_{\|\mathbf{w}\|=1}\{\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}\}. \tag{3}$$

Since $\mathbf{w}_{(1)}$ is a unit vector, therefore we have:

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \frac{\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}. \tag{4}$$

The above equation is possible to maximize using Rayleigh quotient. Following that, if $\mathbf{X}^T\mathbf{X}$ is a positive semidefinite matrix, its maximum possible value is the greatest eigenvalue of the $\mathbf{X}^T\mathbf{X}$ with corresponding eigenvector $\mathbf{w}$.

If we can find $\mathbf{w}_{(1)}$, then the first principal component of a data vector $\mathbf{x}_{(i)}$ can be therefore defined as $t_{1(i)} = \mathbf{x}_{(i)}.\mathbf{w}_{(1)}$in the new coordinate, or as the corresponding vector in the original variables, $\{\mathbf{x}_{(i)}.\mathbf{w}_{(1)}\}\mathbf{w}_{(1)}$.

2.1.2 K^{th} principal component

At first, we can calculate $\widehat{\mathbf{X}}_k$ by

$$\widehat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X}\mathbf{w}_{(s)}\mathbf{w}_{(s)}^T, \tag{5}$$

then its corresponding loading vector whose variance is maximum in the new coordinator matrix:

$$\mathbf{w}_{(k)} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \left\| \widehat{\mathbf{X}}_k \mathbf{w} \right\|^2 \right\} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \frac{\mathbf{w}^T \widehat{\mathbf{X}}_{(k)}^T \widehat{\mathbf{X}}_{(k)} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}. \tag{6}$$

Here, $\mathbf{X}^T\mathbf{X}$ takes an integral part in this maximum operation. The maximum possible values of quantity $\left\{ \frac{\mathbf{w}^T \widehat{\mathbf{X}}_{(k)}^T \widehat{\mathbf{X}}_{(k)} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}$ depends on the eigenvectors of $\mathbf{X}^T\mathbf{X}$ as well as their corresponding eigenvalues. Therefore, its eigenvectors are also the loading vectors we want to find.

After determining the value of $\mathbf{w}_{(k)}$, we would simply do the dot product matrix operation $t_{k(i)} = \mathbf{x}_{(i)}.\mathbf{w}_{(1)}$ to find out the k^{th} scalar of matrix $\mathbf{T}$, or as the corresponding vector in the space of the original variables, $\{\mathbf{x}_{(i)}.\ \mathbf{w}_{(k)}\}\mathbf{w}_{(k)}$, where $\mathbf{w}_{(k)}$is the k^{th} eigenvector of $\mathbf{X}^T\mathbf{X}$.

Overall, the full principal components decomposition of $\mathbf{X}$ can be presented as

$$\mathbf{T} = \mathbf{X}\mathbf{W}, \tag{7}$$

where $\mathbf{W}$ is a *p*-by-*p* matrix whose columns are the eigenvectors of $\mathbf{X}^T\mathbf{X}$. The transpose of $\mathbf{W}$, $\mathbf{W}^T$ is called the whitening or sphering transformation.

2.1.3 Covariance

$\mathbf{X}^T\mathbf{X}$ itself can be considered as proportional to the empirical sample covariance matrix of the dataset $\mathbf{X}$.

The sample covariance Q between two different principal components over the dataset is given by

$$\begin{aligned} Q\left(\mathrm{PC}_{(j)}\mathrm{PC}_{(k)}\right) &\propto \left(\mathbf{X}\mathbf{w}_{(j)}\right)^T\left(\mathbf{X}\mathbf{w}_{(k)}\right) \\ &= \mathbf{w}_{(j)}^T\mathbf{X}^T\mathbf{X}\mathbf{w}_{(k)} \\ &= \mathbf{w}_{(j)}^T\lambda_{(k)}\mathbf{w}_{(k)} \\ &= \lambda_{(k)}\mathbf{w}_{(j)}^T\mathbf{w}_{(k)}. \end{aligned} \tag{8}$$

where we use the eigenvalue property of $\mathbf{w}_{(k)}$to deduce $\mathbf{w}_{(j)}^T\lambda_{(k)}\mathbf{w}_{(k)}$ from $\mathbf{w}_{(j)}^T\mathbf{X}^T\mathbf{X}\mathbf{w}_{(k)}$. Notice that the eigenvectors $\mathbf{w}_{(j)}$ and $\mathbf{w}_{(k)}$ corresponding to eigenvalues of a symmetric matrix are orthogonal (if their eigenvalues are different), or possible to be orthogonalized (if the vectors both have an equal repeated value). The product of $\lambda_{(k)}\mathbf{w}_{(j)}^T\mathbf{w}_{(k)}$ is therefore zero, which explicitly means that there is no sample covariance between different principal components over a dataset.

2.2 Eigenface algorithm

2.2.1 Brief introduction

Standard Eigenface technique was proposed based on Principal Component Analysis. Eigenface method was first used by Sirovich and Kirby to represent a human face as a set of basic features. Beginning with a given dataset of some real human face images, they were successful in calculating the best vector system for image compression. After that, the technique was applied to computer vision field, specifically in face recognition problem by Turk and Pentland.

The main idea of the Eigenface algorithm is to determine the characteristic features of a face so that it can be described as a linear combination of the so-called “Eigenfaces” which is found during the process of face feature extraction.

The first step is applying the PCA to all face images in the training dataset. Then each of them as well as the checking image is projected into the space formed by the obtained eigenvectors (face space) to find its corresponding position. Finally, we compare the Euclidean distance of the checking image with those of images in the database to find the nearest pair (smallest error). If this

error is small enough (smaller than the given threshold), the person is successfully recognized. On the other hand, if the error is too large (larger than the given threshold), the image is considered to belong to an object that was unknown or even not a human.

2.2.2 Eigenface algorithm

2.2.2.1 Image transformation

Let a grayscale face image $\boldsymbol{\Gamma}(x, y)$ have the resolution of *M x N*. Due to the definition of the grayscale image, every pixel in $\boldsymbol{\Gamma}(x, y)$ contains a value from 0 to 255 which describes the intensity of that pixel. The image in this situation could also be treated as a vector of dimension *M* × *N*. For example, an image resolution of 100 × 200 would be considered as a 100 × 200 dimensional vector or simply a single point in a 20,000 high-dimensional space.

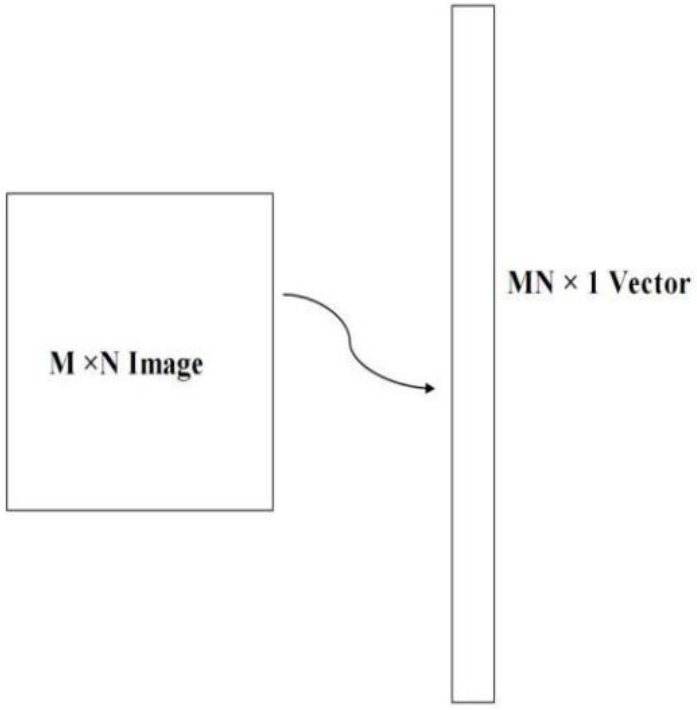

Figure 2.1. *Transformation of image vector [20].*

The main idea is to transform a face image vector of high dimension into the lower one. In the Eigenface technique, this process is done with PCA. PCA will extract the original image to be the total variation (difference) from the calculated mean face. Finally, it filters only the variables containing the highest information from the initial face image to proceed further. PCA plays a very important role in the entire algorithm because it directly reduces the used variables as well as corresponding necessary memory spaces which in turn increases the overall computational speed.

2.2.2.2 Computation of Eigenfaces

Step 1: Prepare the training images.

Figure 2.2. *Yale Face Database A.*

Obtain face images $\mathbf{\Gamma}_1, \mathbf{\Gamma}_2, \mathbf{\Gamma}_3, \mathbf{\Gamma}_4 \dots \mathbf{\Gamma}_M$ (training faces). The images in the database must be converted into the same size (for Yale Face Database A, there are total 165 images of 11 different people, each image has a size of 243 x 320).

Convert those images into *MN x 1* vectors.

Step 2: Prepare the training dataset.

All face images $\mathbf{\Gamma}_i$ in the database are then transformed into one single vector contained in the training dataset **D**:

$$\mathbf{D} = [\mathbf{\Gamma}_1\ \mathbf{\Gamma}_2\ \mathbf{\Gamma}_3\ \mathbf{\Gamma}_4 \dots \mathbf{\Gamma}_M]. \tag{9}$$

Step 3: Compute the average face.

The average face $\mathbf{\Psi}$ consisting information of all training images is obtained through the formulation:

$$\mathbf{\Psi} = \frac{1}{M} \sum_{i=1}^{M} \mathbf{\Gamma}_i. \tag{10}$$

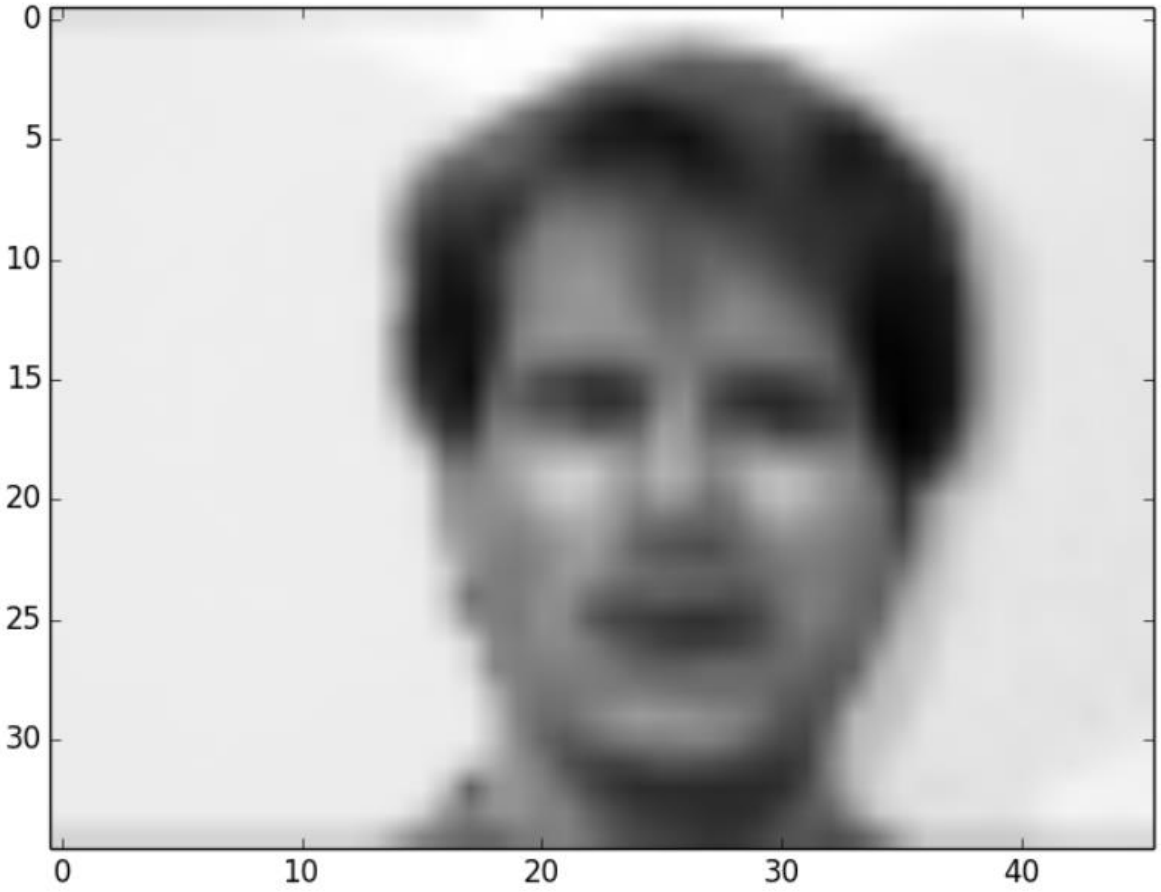

Figure 2.3. *Example of an average face [21].*

Step 4: Compute the mean face.

To obtain the mean face $\mathbf{\Phi}_i$, subtracting the average face $\mathbf{\Psi}$ from the original face $\mathbf{\Gamma}_i$:

$$\mathbf{\Phi}_i = \mathbf{\Gamma}_i - \mathbf{\Psi}. \tag{11}$$

where $i = 1, 2 \ldots M$.

Step 5: Calculate the covariance matrix of the mean face.

Let $\boldsymbol{C}$ be the covariance matrix the mean face. It is determined by the following formulation:

$$\boldsymbol{C} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^T = \boldsymbol{A}\boldsymbol{A}^T \ (MN\ x\ MN \text{ matrix}), \tag{12}$$

where $\boldsymbol{A} = [\boldsymbol{\Phi}_1 \boldsymbol{\Phi}_2 \boldsymbol{\Phi}_3 \ldots \boldsymbol{\Phi}_M]$ $(MN\ x\ M \text{ matrix})$.

Step 6: Calculate eigenvectors and corresponding eigenvalues of $\boldsymbol{A}\boldsymbol{A}^T$.

This is usually performed by applying Single Value Decomposition (SVD) or Eigendecomposition. However, the dimensions of the covariance matrix are too large $(MN\ x\ MN)$, which means the total number of eigenvectors we have to find out is $MN!$ Suppose that with a very bad quality picture (256 x 256), the algorithm has to calculate totally 65,536 eigenvectors as well as their corresponding eigenvalues too. It is clearly not a wise task for computing, especially in real-time applications. Therefore, it is a must to seek another way to go through this step.

One of the common "short-cuts" which are widely used nowadays is to obtain eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$ $(M\ x\ M \text{ matrix})$ before converting them all to our desired eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$.

Let $\boldsymbol{L} = \boldsymbol{A}^T\boldsymbol{A}$ with its related eigenvectors $\boldsymbol{v}_i$ and corresponding eigenvalues $\boldsymbol{\mu}_i$, $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{A}^T$ with its related eigenvectors $\boldsymbol{u}_i$. The below condition must be satisfied:

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{v}_i = \boldsymbol{\mu}_i\boldsymbol{v}_i \tag{13}$$

$$\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{v}_i = \boldsymbol{A}\boldsymbol{\mu}_i\boldsymbol{v}_i \tag{14}$$

$$\boldsymbol{C}\boldsymbol{A}\boldsymbol{v}_i = \boldsymbol{\mu}_i\boldsymbol{A}\boldsymbol{v}_i \tag{15}$$

$$\boldsymbol{C}\boldsymbol{u}_i = \boldsymbol{\mu}_i\boldsymbol{u}_{i,} \tag{16}$$

where $\boldsymbol{u}_i = \boldsymbol{A}\boldsymbol{v}_i$.

Following that, $\boldsymbol{L} = \boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{A}^T$ have the same eigenvalues and the relationship between their eigenvectors are $\boldsymbol{u}_i = \boldsymbol{A}\boldsymbol{v}_i$.

Again, notice that the total number of eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$ is MN while the corresponding number of $\boldsymbol{A}^T\boldsymbol{A}$ is only M. Another point worth pointing out is that those M eigenvalues correspond to the M largest eigenvalues of $\boldsymbol{A}^T\boldsymbol{A}$, along with their related eigenvectors.

Figure 2.4. *Example of M Eigenfaces (M = 11) [21].*

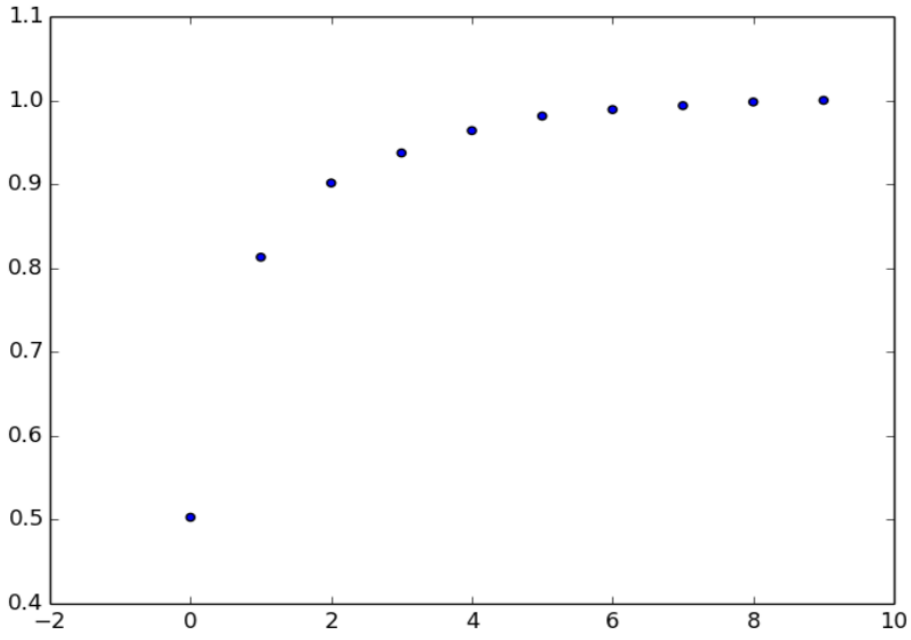

Figure 2.5. *Cumulative sum of the first 10 Eigenvalues [21].*

The M eigenvectors of $\boldsymbol{L} = \boldsymbol{A}^T\boldsymbol{A}$ are used to obtain M largest eigenvectors of $\boldsymbol{C}$ which in turn create our face space basis ($\boldsymbol{v}_i = [v_1\, v_2\, v_3 \ldots v_M]$):

$$\boldsymbol{u}_i = \sum_{l=1}^{M} v_l \boldsymbol{\Phi}_l. \quad (17)$$

Step 7: Choose to keep only K eigenvectors with K largest corresponding eigenvalues of $\boldsymbol{A}\boldsymbol{A}^T$.

Eigenvectors with smallest eigenvalues are permitted to discard due to the reason that they just contain inconsiderably small information about characteristics of a face.

In this step, we can also implement Kaiser Criterion [22] which will only pick the eigenvectors whose eigenvalues which have the absolute values greater than 1.

2.2.2.3 Projecting face images into the face space

After computing necessary eigenvectors as well as their corresponding eigenvalues, the only thing we have to do is projecting all face images $\boldsymbol{\Gamma}_i$ on the database into the face space formed by obtained eigenvectors (or "Eigenfaces") and find the face which has the nearest Euclidean distance to the checking one.

Image projection task can be done by simple calculation:

$$\boldsymbol{\omega}_i = \boldsymbol{u}_i^T(\boldsymbol{\Gamma}_i - \boldsymbol{\Psi}) = \boldsymbol{u}_i^T\boldsymbol{\Phi}_i, \quad (18)$$

where $i = 1, 2 \ldots M$.

The received result is the position of the projecting image on the face space, which is a series of coefficients of the linear combination of eigenvectors forming that image.

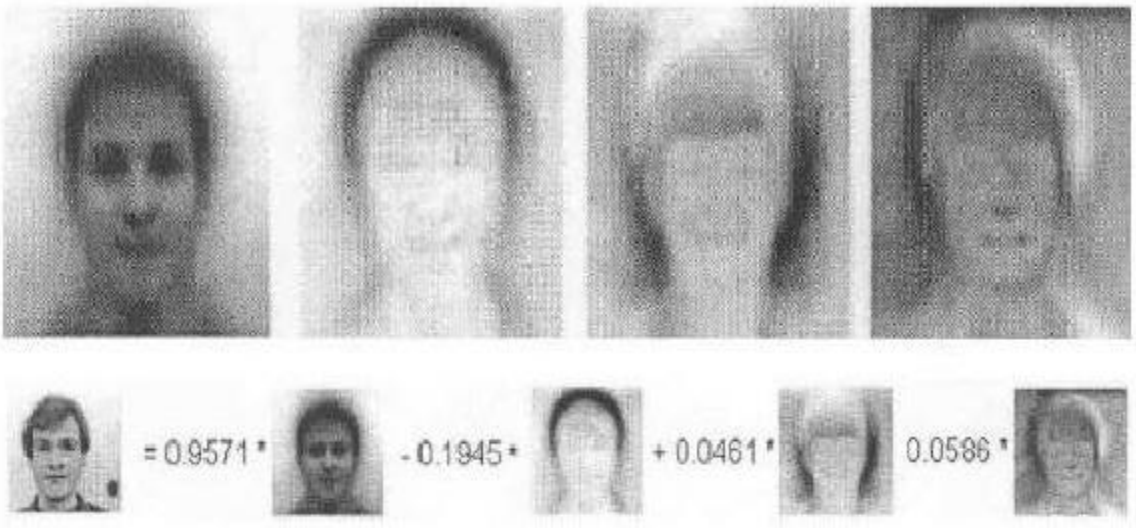

Figure 2.6. *A linear combination of Eigenfaces [23].*

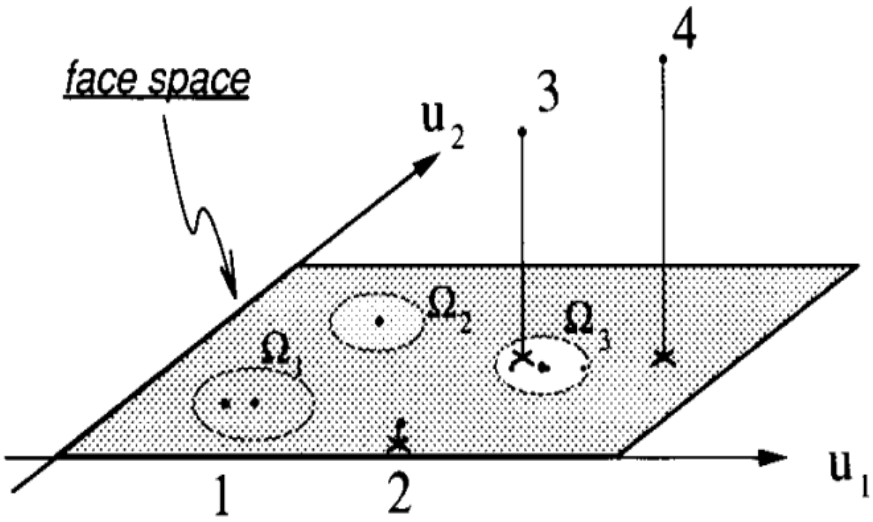

Figure 2.7. *Image projection on face space [24].*

Let $\boldsymbol{\Omega}^T$ be a vector set containing the coordinates of a face in the face space. Hence, $\boldsymbol{\Omega}^T$ of a face image could be defined as

$$\boldsymbol{\Omega}^T = [\boldsymbol{\omega}_1\ \boldsymbol{\omega}_2\ \boldsymbol{\omega}_3 \dots \boldsymbol{\omega}_M]. \tag{19}$$

Keep repeating the above formulation *M* times to determine all faces coordinates $\boldsymbol{\Omega}_i^T$ of face images in the database and one more time for $\boldsymbol{\Omega}_c^T$ of the checking face. Now we can do an

estimation of recognizing the checking face image by finding its closest positioned face in the database in terms of Euclidean distance:

$$\varepsilon_r = min\left\|\boldsymbol{\Omega}_c^T - \boldsymbol{\Omega}_i^T\right\|, \tag{20}$$

where $i = 1, 2 \ldots M$.

For obtain the best result, it is necessary to define spectacular threshold values $\theta_{\varepsilon 1} < \theta_{\varepsilon 2}$. If the closest distance computed above ε_r is greater than the value of threshold $\theta_{\varepsilon 2}$, we can discard the result to conclude that the input face is not a human face. On the other hand, if it is smaller or equal to the value of threshold $\theta_{\varepsilon 2}$ and greater than $\theta_{\varepsilon 1}$, we can say that it is a human face but unknown or not yet trained. Otherwise, the result would point to the person described in the nearest image.

2.2.2.4 Summary

In general, the computation of the Eigenface algorithm can be done via these steps:

1) Prepare and convert the training images $\boldsymbol{\Gamma}_1, \boldsymbol{\Gamma}_2, \boldsymbol{\Gamma}_3, \boldsymbol{\Gamma}_4 \ldots \boldsymbol{\Gamma}_M$.
2) Prepare the training dataset $\mathbf{D} = [\boldsymbol{\Gamma}_1\ \boldsymbol{\Gamma}_2\ \boldsymbol{\Gamma}_3\ \boldsymbol{\Gamma}_4 \ldots \boldsymbol{\Gamma}_M]$.
3) Compute the average face $\boldsymbol{\Psi} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\Gamma}_i$.
4) Compute the mean face $\boldsymbol{\Phi}_i = \boldsymbol{\Gamma}_i - \boldsymbol{\Psi}$.
5) Calculate the covariance matrix $\boldsymbol{C} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^T = \boldsymbol{A}\boldsymbol{A}^T$ of the mean face.
6) Calculate the eigenvectors with corresponding eigenvalues of $\boldsymbol{C}$ via $\boldsymbol{L} = \boldsymbol{A}^T\boldsymbol{A}$ and the relationship $\boldsymbol{u}_i = \boldsymbol{A}\boldsymbol{v}_i$.
7) Choose to keep only *K* eigenvectors with *K* largest corresponding eigenvalues of $\boldsymbol{A}\boldsymbol{A}^T$.
8) Projecting all face images in the database and the input face into the face space $\boldsymbol{\omega}_i = \boldsymbol{u}_i^T(\boldsymbol{\Gamma}_i - \boldsymbol{\Psi}) = \boldsymbol{u}_i^T\boldsymbol{\Phi}_i$.
9) Find the nearest Euclidean distance $\varepsilon_r = min\left\|\boldsymbol{\Omega}_c^T - \boldsymbol{\Omega}_i^T\right\|$ where $\boldsymbol{\Omega}^T = [\boldsymbol{\omega}_1\ \boldsymbol{\omega}_2\ \boldsymbol{\omega}_3 \ldots \boldsymbol{\omega}_M]$, $\boldsymbol{\Omega}_c^T$ is the checking image projection and $i = 1, 2, \ldots, N'$.
10) Define and compare to threshold values $\theta_{\varepsilon 1} < \theta_{\varepsilon 2}$ to determine whether it is recognized, a new person or not a human face.

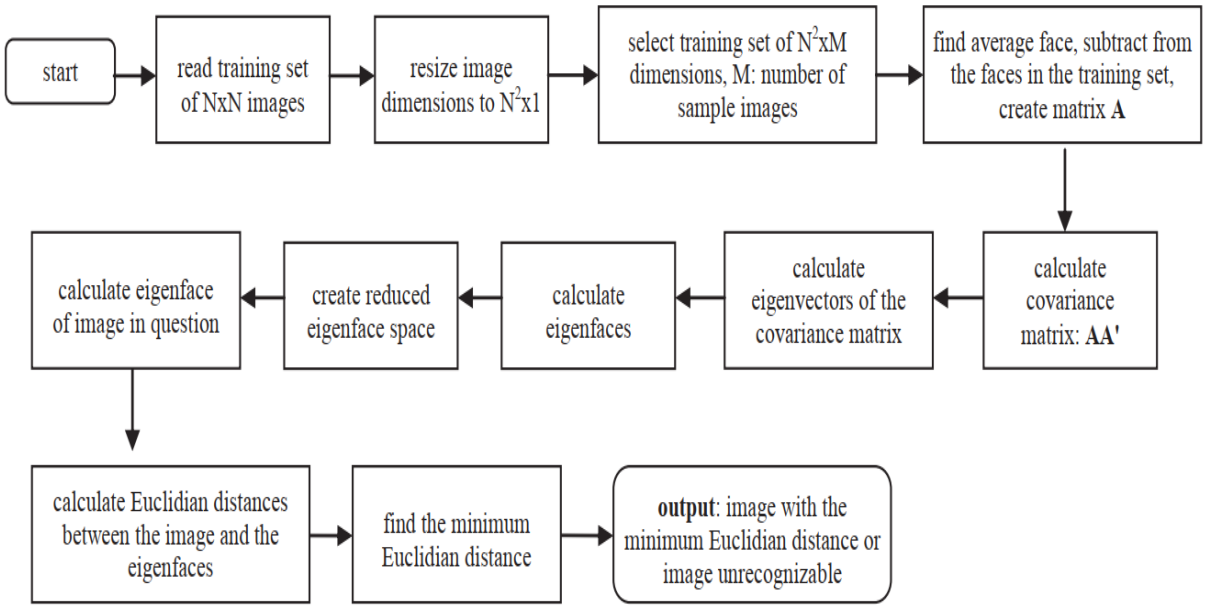

Figure 2.8. *Flow chart of the entire Eigenface algorithm [25].*

3. Fisherface Method

3.1 Linear Discriminant Analysis (LDA)

First proposed by Ronald A. Fisher as “Linear Discriminant” in 1936 [26], the method quickly showed its potential in applying to solve many practical problems in positioning and product management such as bankruptcy prediction, marketing, biomedical studies, earth science and face recognition. The original method only aimed to a 2-class problem, however, it was expanded later on to be called as “multi-class Linear Discriminant Analysis” or “Multiple Discriminant Analysis” by C. R. Rao in 1948 [27].

The general idea of LDA is very similar to PCA. They are both used to reduce the number of dimensions of input data. However, PCA is classified as “unsupervised learning” algorithm because it discards the information about the class labels and its goal is simply to maximize the total variance of the whole dataset. By contrast, LDA is a “supervised learning” approach which takes into account the class labels to maximize the separation among the classes and minimize the distance among samples within a class.

While LDA has proved its superior performance to PCA in most of the multi-class problems, this is not always the case. LDA still has its own issue about singular within-class scatter matrix. Therefore, in practice, it is not uncommon to use PCA followed by LDA in combination to overcome this obstacle.

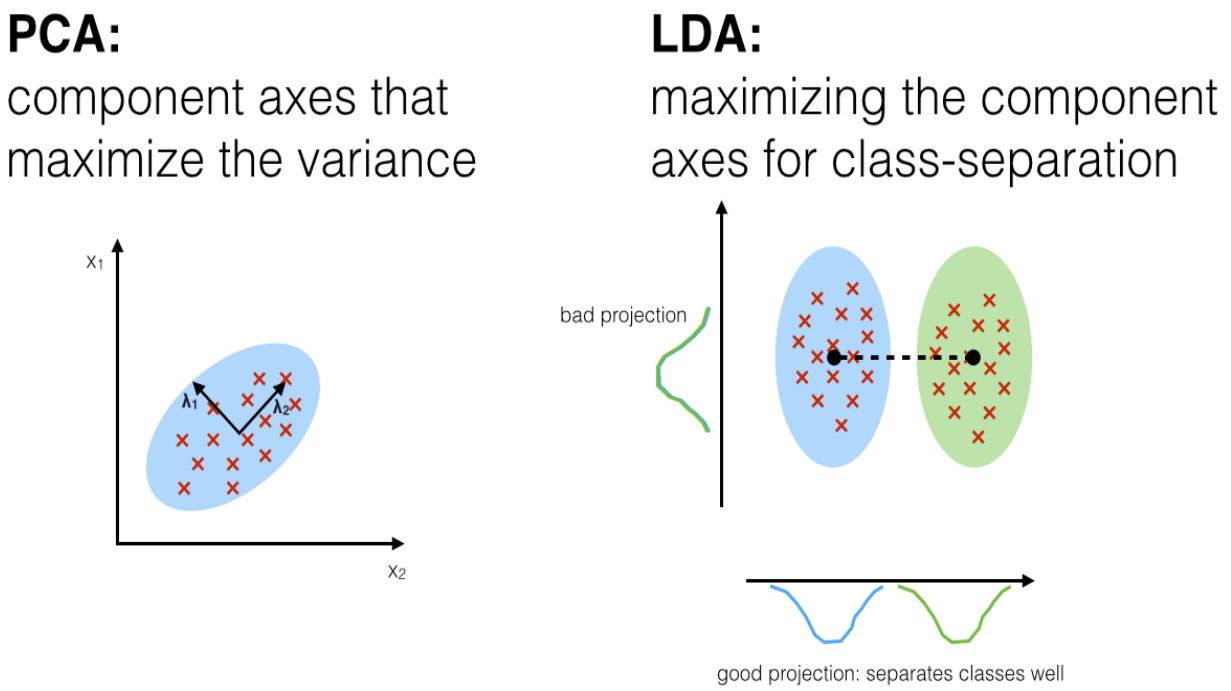

Figure 3.1. *PCA versus LDA [28].*

3.2 Fisherface algorithm

3.2.1 Multi-class LDA

Fisherface technique was proposed by Peter N. Belhumeur, João P. Hespanha and David J. Kriegman in 1997 by making use of the multi-class LDA method.

Suppose that there are C classes where each class has mean $\boldsymbol{\mu}_i$. Then the between-class scatter matrix $\boldsymbol{\Sigma}_b$ and within-class scatter matrix $\boldsymbol{\Sigma}_w$ could be devised as

$$\boldsymbol{\Sigma}_b = \sum_{i=1}^{C} N_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T, \tag{21}$$

$$\boldsymbol{\Sigma}_w = \sum_{i=1}^{C} \sum_{k \in i} (\mathbf{x}_k - \boldsymbol{\mu}_i)(\mathbf{x}_k - \boldsymbol{\mu}_i)^T, \tag{22}$$

where $\boldsymbol{\mu}$ is the mean of all class means and $\mathbf{x}_k$ is the column matrix of each sample in class i. If we call $\vec{w}$ as the direction of the separation among classes, then

$$\boldsymbol{S} = \frac{\vec{w}^T \boldsymbol{\Sigma}_b \vec{w}}{\vec{w}^T \boldsymbol{\Sigma}_w \vec{w}}. \tag{23}$$

The meaning of the above equation is that if $\vec{w}$ is an eigenvector of $\boldsymbol{\Sigma}_w^{-1}\boldsymbol{\Sigma}_b$, the class separation equals to the corresponding eigenvectors.

In addition, if $\boldsymbol{\Sigma_w^{-1}\Sigma_b}$ is a diagonalized matrix, the distribution among features can be represented in a space spanned by the eigenvectors relating to the $C-1$ largest eigenvalues (since it was proven that the maximum rank of $\boldsymbol{\Sigma_b}$ is $C-1$ [10]).

3.2.2 Computation of Fisherfaces

Step 1: Use Eigenface method to calculate the coordinates of each face image on the eigenface space in the database, mean $\boldsymbol{\mu_i}$ of each class and choose K largest eigenvectors to be at most $N'-C$ where C is the total number of classes and N' is the total number of face images in the database. It is a compulsory step to deal with singular matrix problem which will be discussed later on.

Step 2: Arrange K largest eigenvectors as a matrix $\boldsymbol{W}_{\text{pca}}$ where the number of columns equals to K.

Step 3: Calculate the mean of class means $\boldsymbol{\mu}$ (total mean):

$$\boldsymbol{\mu} = \frac{\mathbf{1}}{C}\sum_{i=1}^{C}\boldsymbol{\mu_i}. \tag{24}$$

Step 4: Calculate the between-class scatter matrix $\boldsymbol{S_B}$:

$$\boldsymbol{S_B} = \sum_{i=1}^{C} N_i(\boldsymbol{\mu_i} - \boldsymbol{\mu})\,(\boldsymbol{\mu_i} - \boldsymbol{\mu})^T, \tag{25}$$

where N_i is the number of samples in class i.

Step 5: Calculate the within-class scatter matrix $\boldsymbol{S_W}$:

$$\boldsymbol{S_w} = \sum_{i=1}^{C}\sum_{k\in i}(\mathbf{x}_k - \boldsymbol{\mu_i})\,(\mathbf{x}_k - \boldsymbol{\mu_i})^T, \tag{26}$$

where $\mathbf{x}_k$ is the column matrix of each image in class i which we have found out in the first step of calculating Eigenface method.

Step 6: If $\boldsymbol{S_w}$ is a nonsingular matrix, the LDA projection matrix $\boldsymbol{W}_{\text{lda}}$ could be defined as an orthonormal column matrix which maximizes the proportion of the determinant of the between-class scatter matrix to the determinant of the within-class scatter matrix:

$$\begin{aligned}\boldsymbol{W}_{\text{lda}} &= arg\max_{W}\frac{|W^T\boldsymbol{S_B}W|}{|W^T\boldsymbol{S_w}W|}\\ &= [\mathbf{w}_1\ \mathbf{w}_2\ \dots\ \mathbf{w}_m],\end{aligned} \tag{27}$$

where $[\mathbf{w}_1 \ \mathbf{w}_2 \ \dots \ \mathbf{w}_m]$ is the set of m largest eigenvectors of $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ with corresponding eigenvalues $[\lambda_1 \ \lambda_2 \ \dots \ \lambda_m]$ found via the formula:

$$\boldsymbol{S}_B\mathbf{w}_i = \lambda_i\boldsymbol{S}_w\mathbf{w}_i$$

$$\boldsymbol{S}_W^{-1}\boldsymbol{S}_B\mathbf{w}_i = \lambda_i\mathbf{w}_i, \tag{28}$$

where $i = 1, 2, \dots, m$.

We also note that there are at most $C - 1$ nonzero generalized eigenvectors indicating that the maximum number of Fisherfaces is always $C - 1$.

Although the calculation is quite complicated compared to PCA, the results are a lot better.

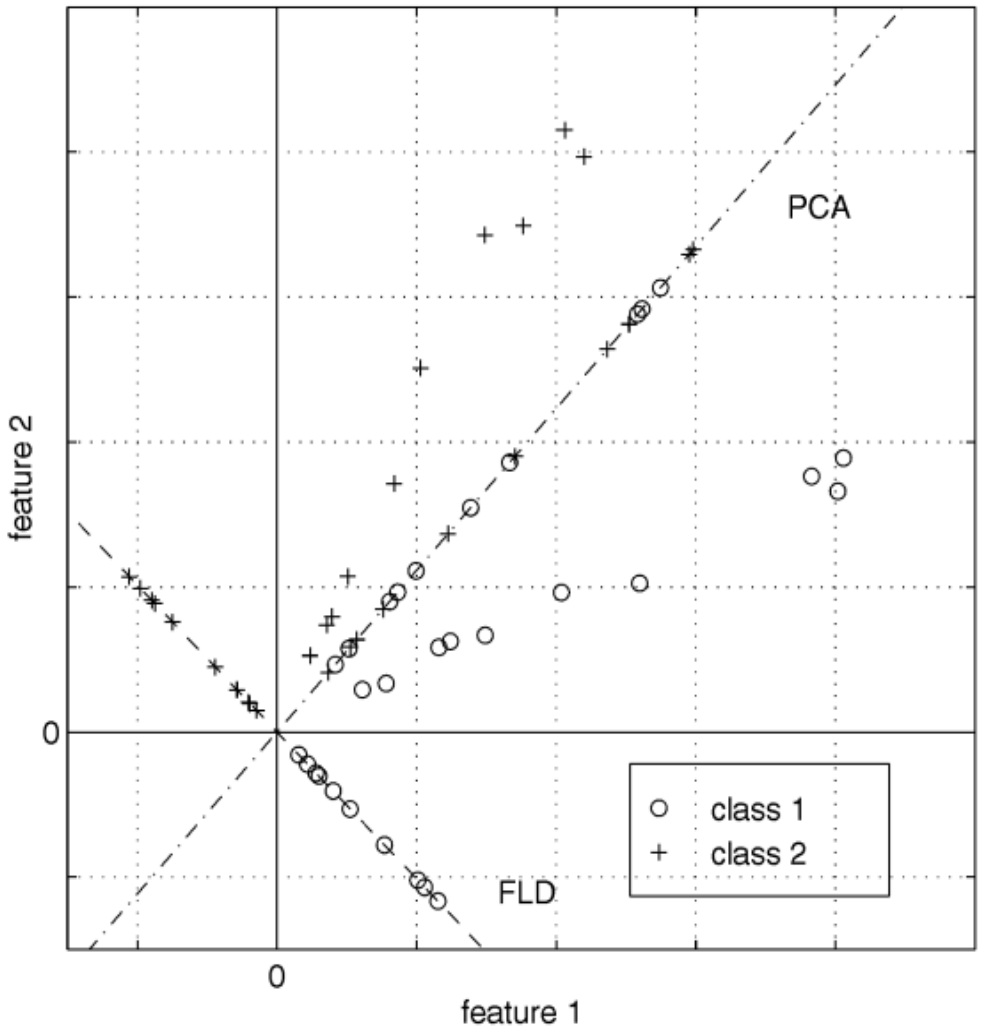

Figure 3.2. *Comparison of PCA and FLD (LDA) [10].*

In the above figure, there are two classes with 20 samples for each one. All samples lie randomly near (not on) the line passing through the origin point of the 2-D space. PCA and LDA are both used to reduce the number of dimensions from 2-D to 1-D. It finally turns out that LDA does a better job in classifying between class 1 and 2 where samples in each class are all pulled together while the distance is lengthened between two different classes. PCA, on the other hand, only focuses on preserving the total variance of the whole range of dataset. Therefore, it is obvious that PCA has a larger total variance while LDA achieves larger between-class scatter and smaller within-class scatter which results in better classification performance.

A drawback of LDA method occurs when $\boldsymbol{S}_w$ is a singular matrix which we cannot calculate $\boldsymbol{W}_{\text{lda}}$ using the presented formula. In practice, $\boldsymbol{S}_w$ is always singular due to the fact that the number of pixels XY in each XxY image sample is always much larger than the maximum rank of $\boldsymbol{S}_w$ which is $N' - C$. To overcome this obstacle, Fisherface suggests that we have to implement PCA first to reduce the number of dimensions of feature space from XY to $N' - C$. After that, we can apply the standard LDA to reduce the number of dimensions further to $C - 1$ without having any worries about the singular matrix problem.

Step 7: Ultimately, $\boldsymbol{W}_{\text{opt}}$ is defined to be:

$$\boldsymbol{W}_{\text{opt}} = \boldsymbol{W}_{\text{pca}}\boldsymbol{W}_{\text{lda}}, \tag{29}$$

where $\boldsymbol{W}_{\text{pca}}$ is a $XY\ x\ (N' - C)$ matrix which contains $N' - C$ eigenvectors after performing PCA and $\boldsymbol{W}_{\text{pca}}$ is a $(N' - C)\ x\ m$ matrix with m eigenvectors found by implementing LDA.

It can be devised from the above equation that our final m Fisherfaces are contained in $\boldsymbol{W}_{\text{opt}}$ which now is a $XY\ x\ m$ matrix. This solution to solve the singular matrix problem of LDA is obviously not the unique one. There are certainly many other alternative ways to deal with that problem, however, it is the most common way which is ubiquitous in a lot of machine learning systems nowadays.

3.2.3 Projecting face images into the face space

The projection task of Fisherface method is the same as Eigenface except we do not have to normalize the face image:

$$\boldsymbol{\Omega}_i = \boldsymbol{W}_{opt}^T\boldsymbol{\Gamma}_i, \tag{30}$$

where $i = 1, 2, \ldots, N'$.

Again, projecting all face images in the database as well as checking face into the face space before doing the comparison to determine which one in the training set has the nearest Euclidean distance to our checking face.

We can also define both two threshold values $\theta_{\varepsilon 1} < \theta_{\varepsilon 2}$ or only one of them depending on the goal of our face recognition system.

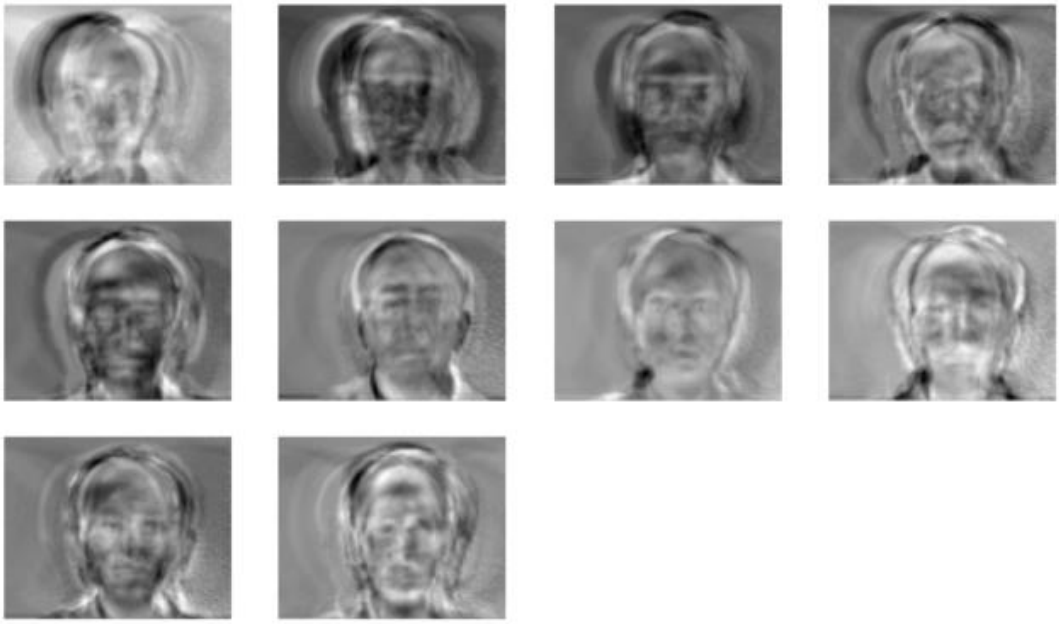

Figure 3.3. *Example of Fisherfaces.*

3.2.4 Summary

In general, the computation of the Fisherface technique can be done via these steps:

1) Implement the Eigenface algorithm with $K = N' - C$.
2) Arrange K largest eigenvectors as a matrix $\boldsymbol{W}_{\text{pca}}$ where the number of columns equals to K.
3) Calculate the mean of class means $\boldsymbol{\mu} = \frac{\mathbf{1}}{C}\sum_{i=1}^{C}\boldsymbol{\mu}_i$.
4) Calculate the between-class scatter matrix $\boldsymbol{S}_B = \sum_{i=1}^{C} N_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})\,(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$.
5) Calculate the within-class scatter matrix $\boldsymbol{S}_w = \sum_{i=1}^{C}\sum_{k\in i}(\mathbf{x}_k - \boldsymbol{\mu}_i)\,(\mathbf{x}_k - \boldsymbol{\mu}_i)^T$.
6) Find $m = C - 1$ eigenvectors of $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ with largest corresponding eigenvalues. Arrange m largest eigenvectors as a matrix $\boldsymbol{W}_{\text{lda}}$ where the number of columns equals to m.
7) Calculate $\boldsymbol{W}_{\text{opt}} = \boldsymbol{W}_{\text{pca}}\boldsymbol{W}_{\text{lda}}$.
8) Projecting all face images in the database and input face into the face space $\boldsymbol{\Omega}_i = \boldsymbol{W}_{opt}^T\boldsymbol{\Gamma}_i$.
9) Find the nearest Euclidean distance $\varepsilon_r = min\left\|\boldsymbol{\Omega}_c^T - \boldsymbol{\Omega}_i^T\right\|$ where $\boldsymbol{\Omega}_c^T$ is the checking image projection and $i = 1, 2, \ldots, N'$.

10) Define and compare to threshold values $\theta_{\varepsilon 1} < \theta_{\varepsilon 2}$ to determine whether it is recognized, a new person or not a human face.

4. Implementation and evaluation

4.1 Basic system implementation

Integrated Development Environment (IDE) used in this thesis was Microsoft Visual Studio Code. The chosen database was the extended Yale Face Database B [29] (which could be found via http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html). Originally, there are 16128 images in total with 28 different human faces taken under 9 poses and 64 illumination conditions. Each image is grayscale and has a size of 640 x 480.

In this project, we did not use the entire original images but only part of their cropped version. In the cropped database, there are 38 human faces with about 61 images with different lighting angles for each. Cropping and aligning the face from given image is a necessary step before applying any face recognition method since face in every image has different size and position. Doing this also reduces the effect coming from background such as the wall or scene behind the face which could improve the overall performance.

Based on the represented theory, a program was created using Python 3.7 programming language with self-written code with a convenient Graphical User Interface (GUI). External libraries used are

- Numpy 1.15.2: http://www.numpy.org/ (for matrix operations)
- Matplotlib 3.0.0: https://matplotlib.org/ (for graphs plotting)
- Pillow 5.2.0: https://pillow.readthedocs.io/en/5.2.x/ (for images reading)
- Scikit-image 0.14.0: http://scikit-image.org/ (for images down-sampling)
- Tkinter: https://docs.python.org/3/library/tk.html/ (for GUI programming)

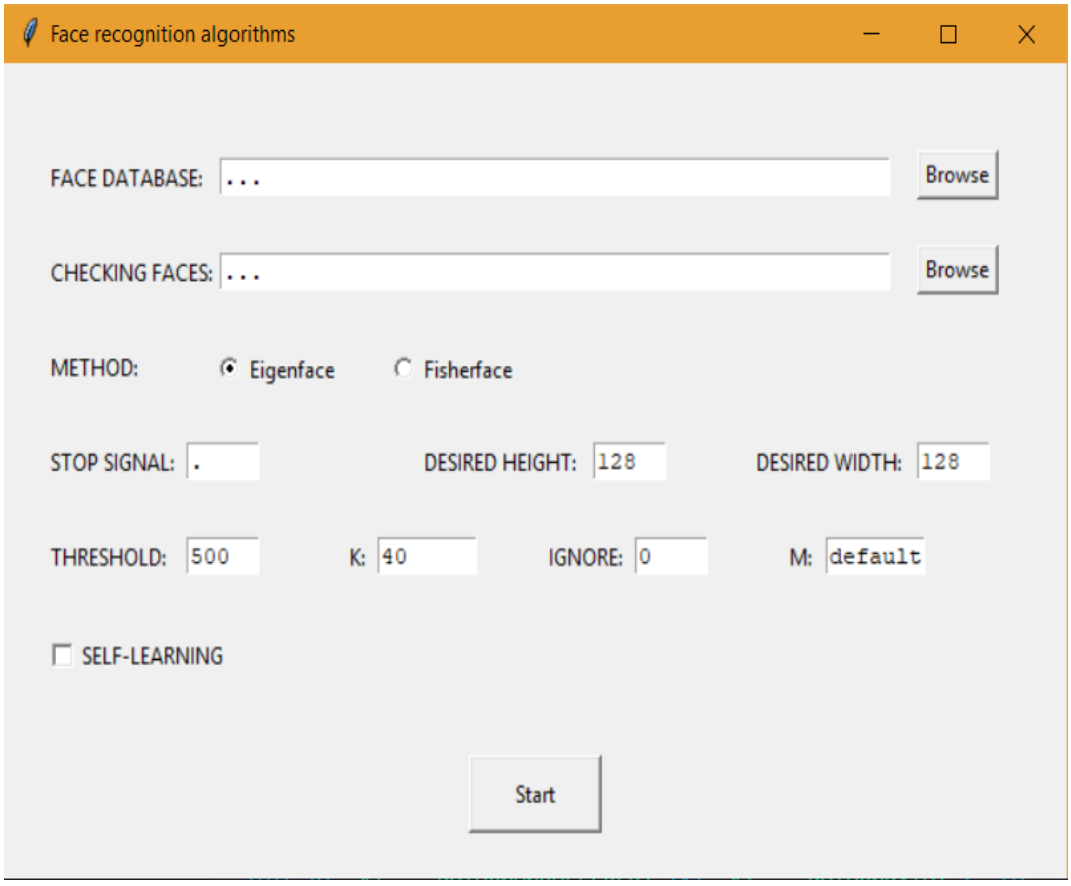

Figure 4.1. *Input interface.*

At first, images we want to check will be browsed to "CHECKING FACES" line, while the database is input via "FACE DATABASE" line. Next, we will choose which method we want to perform the recognition task. "STOP SIGNAL" is used for the program to get the owner of the picture. We specify our desired height and width of a standard image for the purpose of downscaling. "IGNORE" input is only used for Eigenface to ignore the first N components (depending on what we typed). On the other hand, M is the number of Eigenfaces which is kept after implementing LDA step in the Fisherface method (default means the maximum one). We can also tick to "SELF-LEARNING" box if we want the program automatically copy the image which it has just recognized to the database to enhance the upcoming results.

In those experiments, we only made use of threshold $\theta_{\varepsilon 1}$. The nearest distance from the face image in the database in terms of Euclidean distance to the checking face would be computed before applying previously defined threshold $\theta_{\varepsilon 1}$ to get the final result.

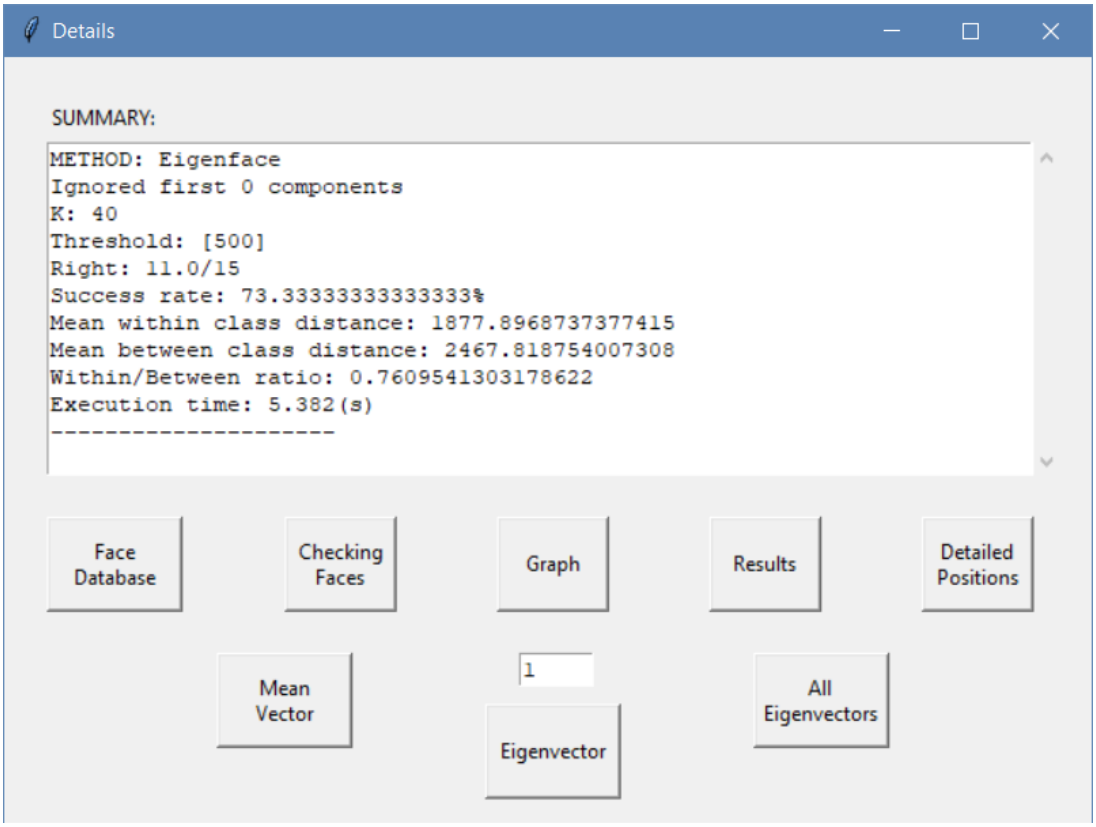

Figure 4.2. *Output interface.*

After doing the face recognition, a result window will be shown as the above figure. "SUMMARY" part will list all the important input information of the recognition method like its name, the number of the first components to ignore (if any), K and threshold value $\theta_{\varepsilon 1}$ as well. Then the Success Rate, Mean Within Class Distance, Mean Between Class Distance and Within/Between Ratio which will be defined later on would be displayed on the screen. The final line shows the calculated execution time of the operation for comparison purpose.

We also designed some useful buttons which is at the bottom of the result window. Face Database and Checking Faces buttons will open the folder containing the database and checking images respectively. Graph button will show the 2-D chart which is built based on taking the first two components to give the overall visual perspective of the results. Results and Detailed Position buttons will lead the user to the detailed results for each checking image as well as the specific position of each database image stored in two separated .txt files.

In addition, the user can click to Mean Vector button for observing the calculated mean face of all face images in the database. All Eigenvectors button shows all the previously defined number of eigenvectors visually, while the Eigenvector gives us a choice to see only the desired eigenvector.

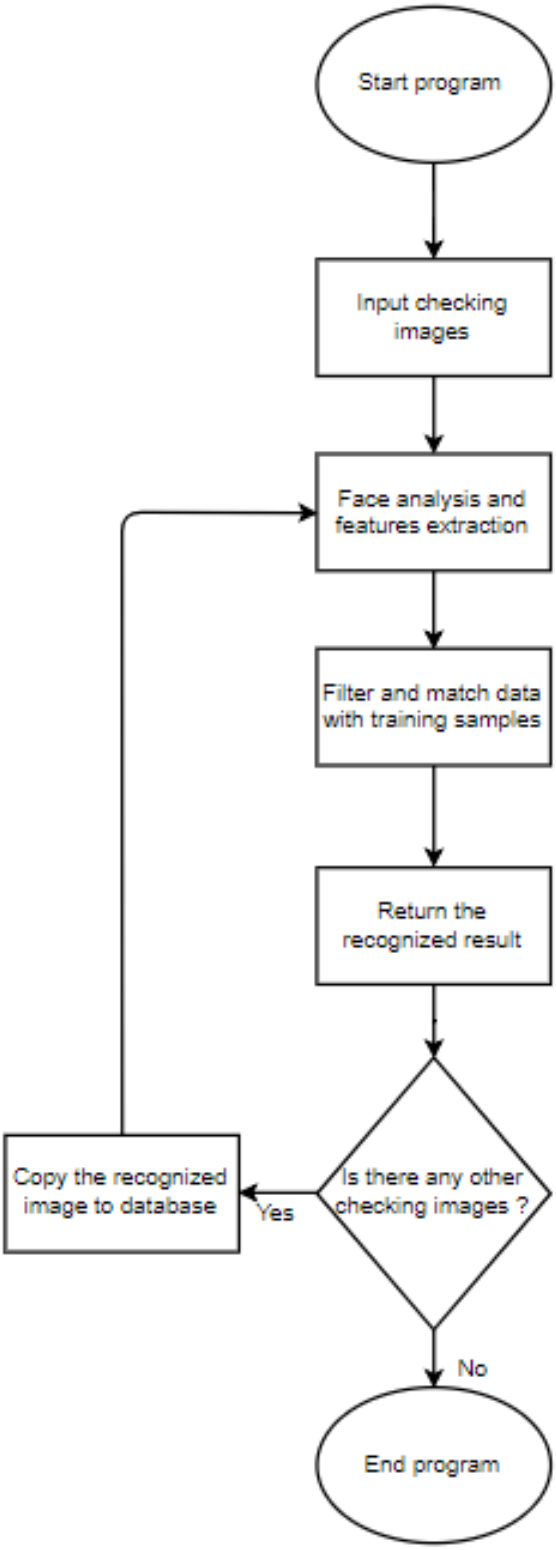

Figure 4.3. *System flowchart.*

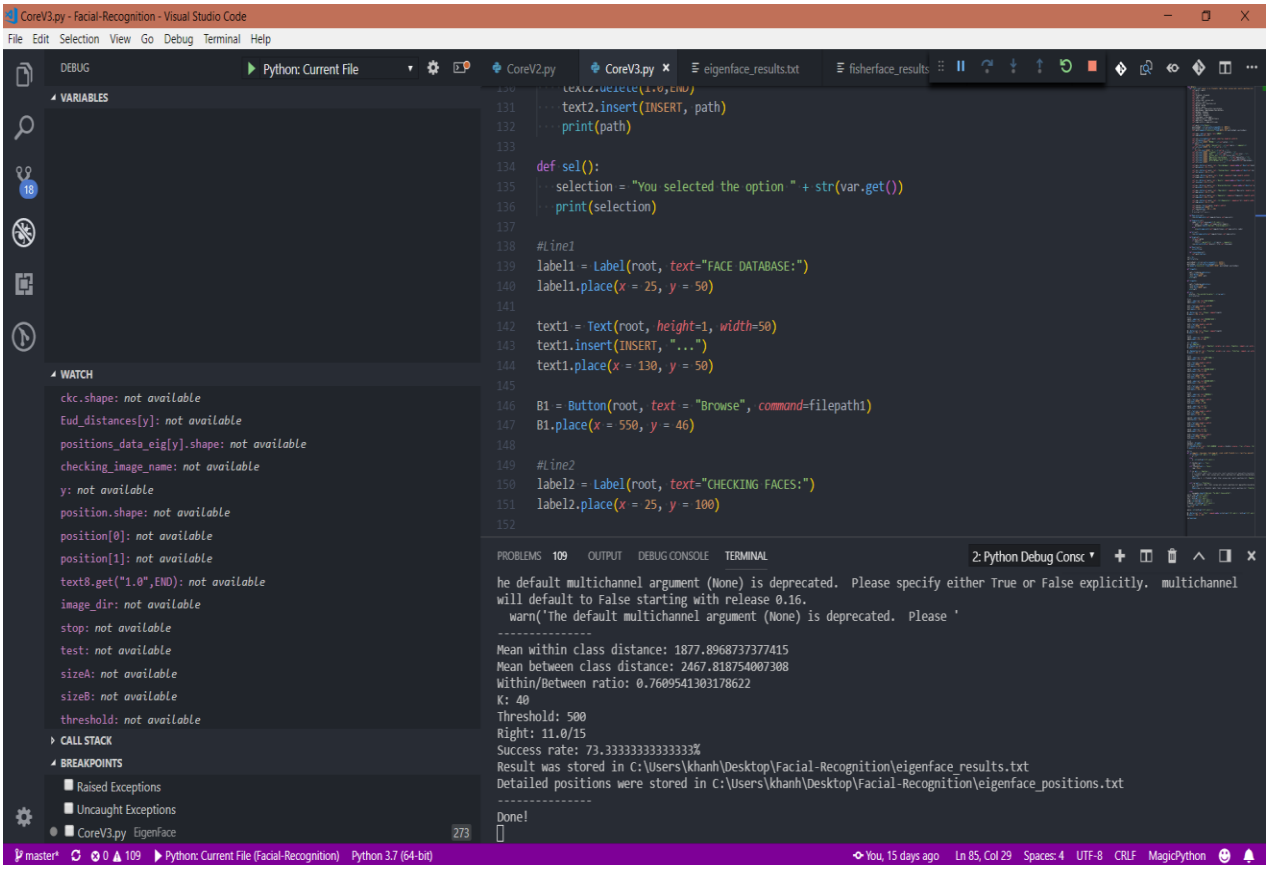

Figure 4.4. *Visual Studio IDE.*

4.2 Comparison between Eigenface and Fisherface method

4.2.1 Performance criteria

In this thesis, the performance of both two methods was thoroughly examined based on two main criteria: Success Rate and Within/Between Ratio.

4.2.1.1 Success Rate

The definition of Success Rate used here is the ratio of the number of successful attempts to the total number of attempts in a specific experiment in percentage:

$$\frac{\sum successful\ attempts}{\sum total\ attempts}\ x\ 100\%. \tag{31}$$

4.2.1.2 Within/Between Ratio

Within/Between Ratio is defined as the ratio of the mean of all mean Euclidean distances of samples in one class to the mean of Euclidean distances between classes mean. Suppose that a specific class S_i has P samples and there are L class in total, then Within/Between Ratio is

$$\frac{\frac{1}{L}\sum_{i=1}^{L}(\frac{1}{P}\sum_{k=1}^{P}\boldsymbol{\Omega}_k)}{\frac{1}{C_2^L}\sum_{i\neq j}(\boldsymbol{\Upsilon}_i - \boldsymbol{\Upsilon}_j)}, \tag{32}$$

where $\boldsymbol{\Omega}_k$: projection of a sample of class S_i in face space,

$\boldsymbol{\Upsilon}_i$, $\boldsymbol{\Upsilon}_j$: mean projection of all samples within class S_i, S_j,

C_2^L : number of combinations of choosing 2 different classes out of L classes.

We want this ratio to be as small as possible, implying that the numerator has to be very small compared to denominator. This means the distances between samples within one class are relatively small while they are far away from those belonging to the other classes, which in turn results in better classification.

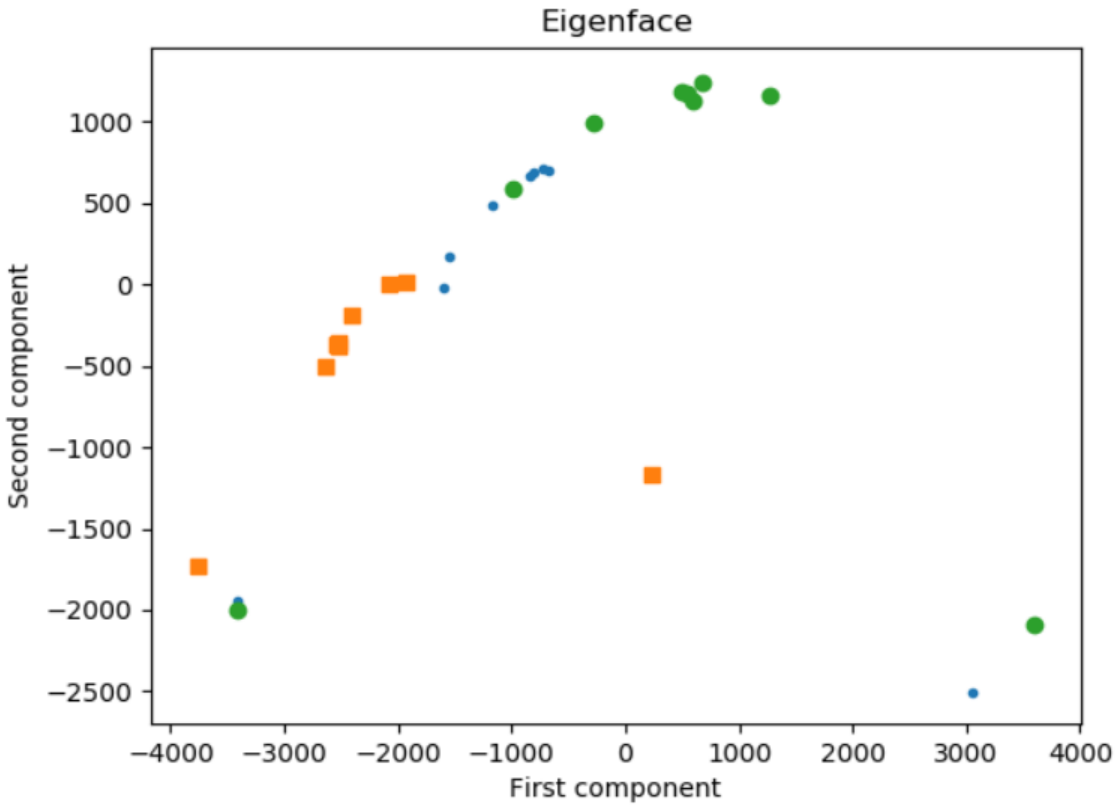

Figure 4.5. *Example of 0.76 Within/Between Ratio.*

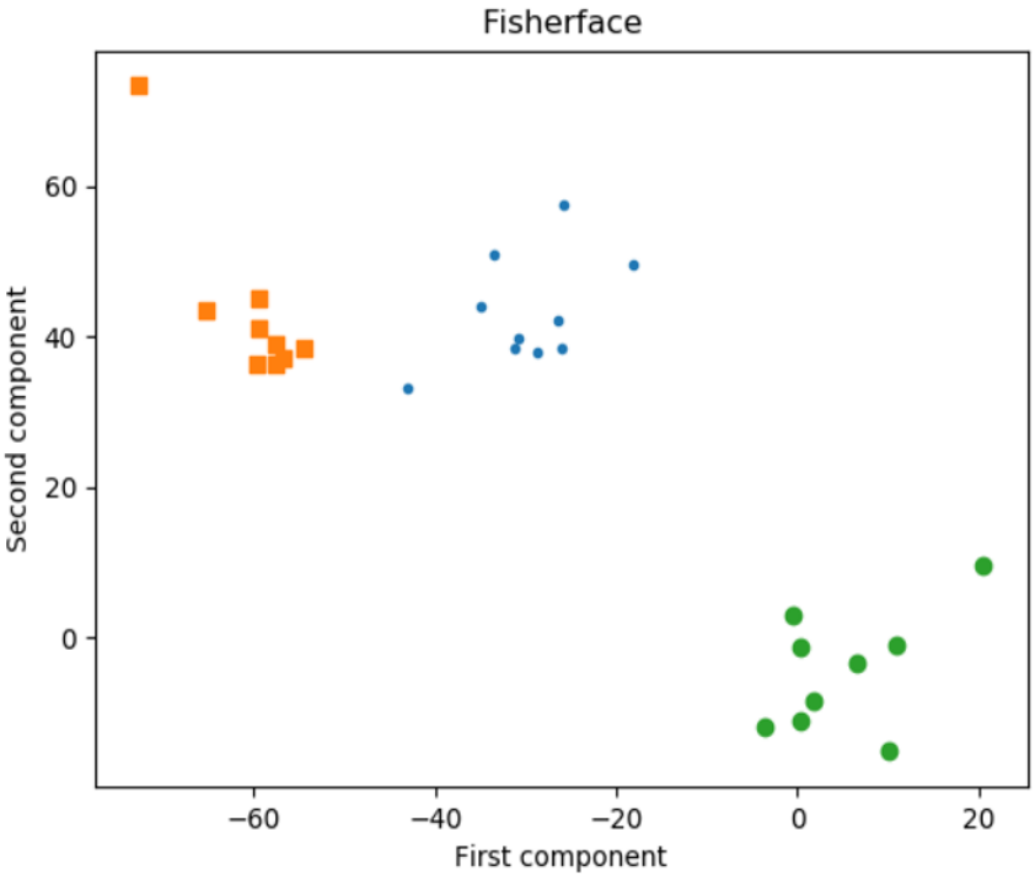

Figure 4.6. *Example of 0.22 Within/Between Ratio.*

4.2.2 Experiments

Experiment 1: Recognizing ideal input images with only ideal training images.

In database, there were totally 152 ideal images taken from 8 different people. Each one had about 17 - 20 images. Test images were 16 random ideal images of those 8 people (2 for each). $K = 40$ for Eigenface and $m = 7$ for Fisherface. $\theta_{\varepsilon 1} = 500$ and the dimensions of each image were $168\ x\ 192$. Down sample factor was 2.

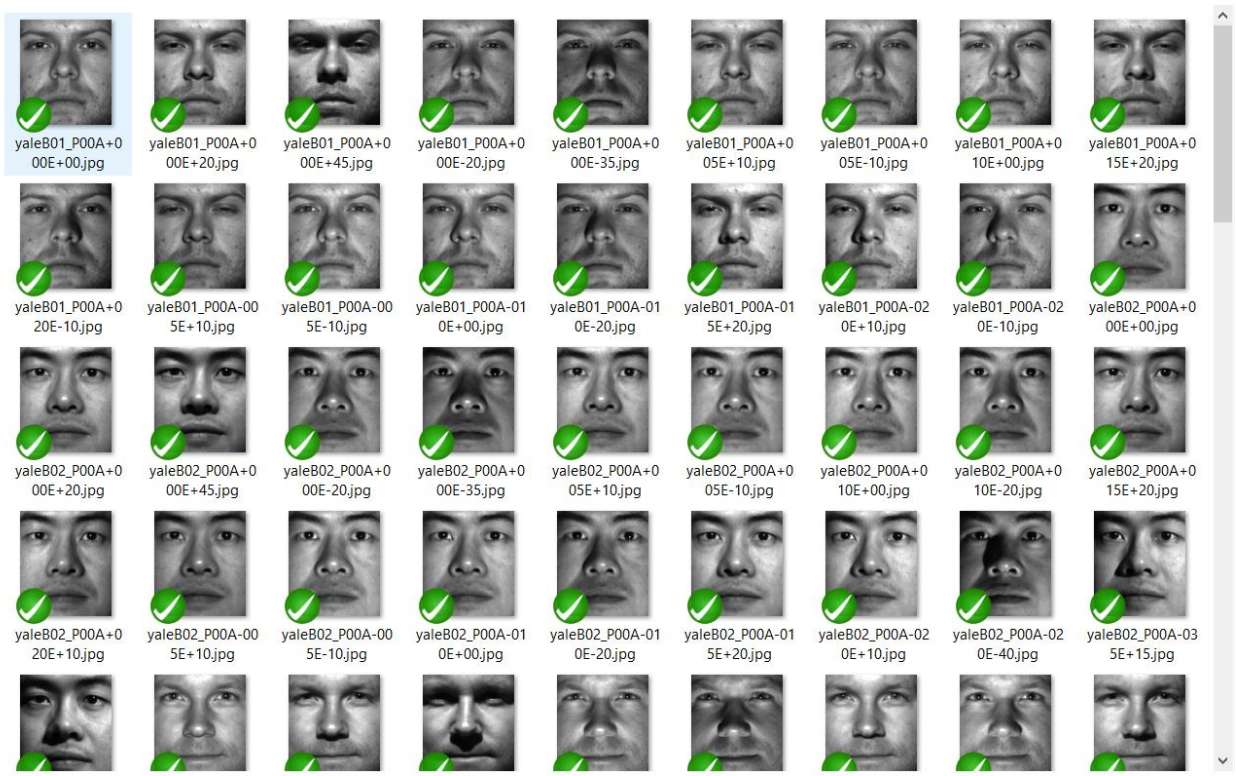

Figure 4.7. *Experiment 1: 152 ideal training images.*

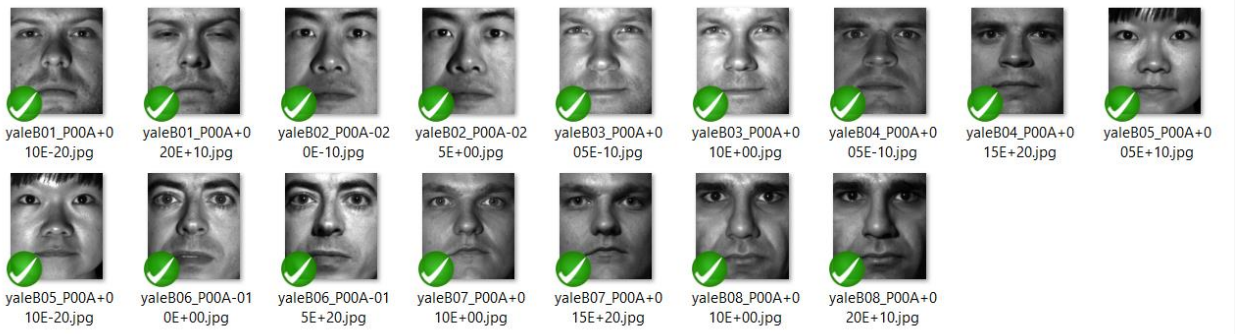

Figure 4.8. *Experiment 1: 16 ideal input images.*

Figure 4.9. *Experiment 1: Eigenfaces.*

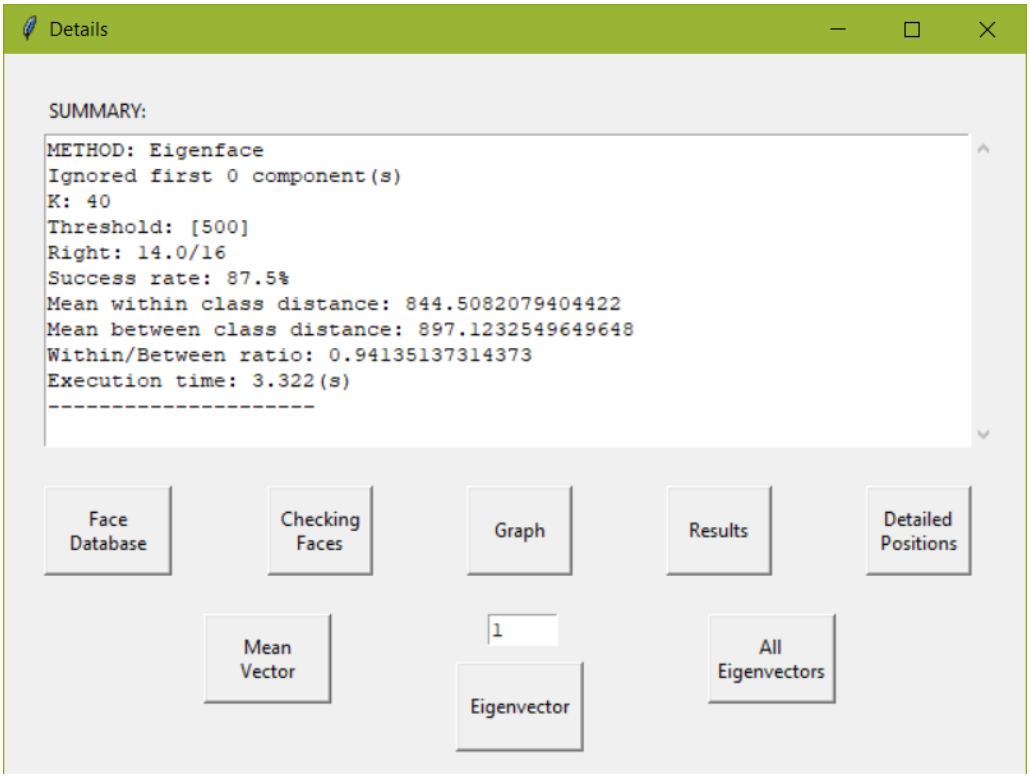

Figure 4.10. *Experiment 1: Eigenface results.*

Figure 4.11. *Experiment 1: Fisherfaces.*

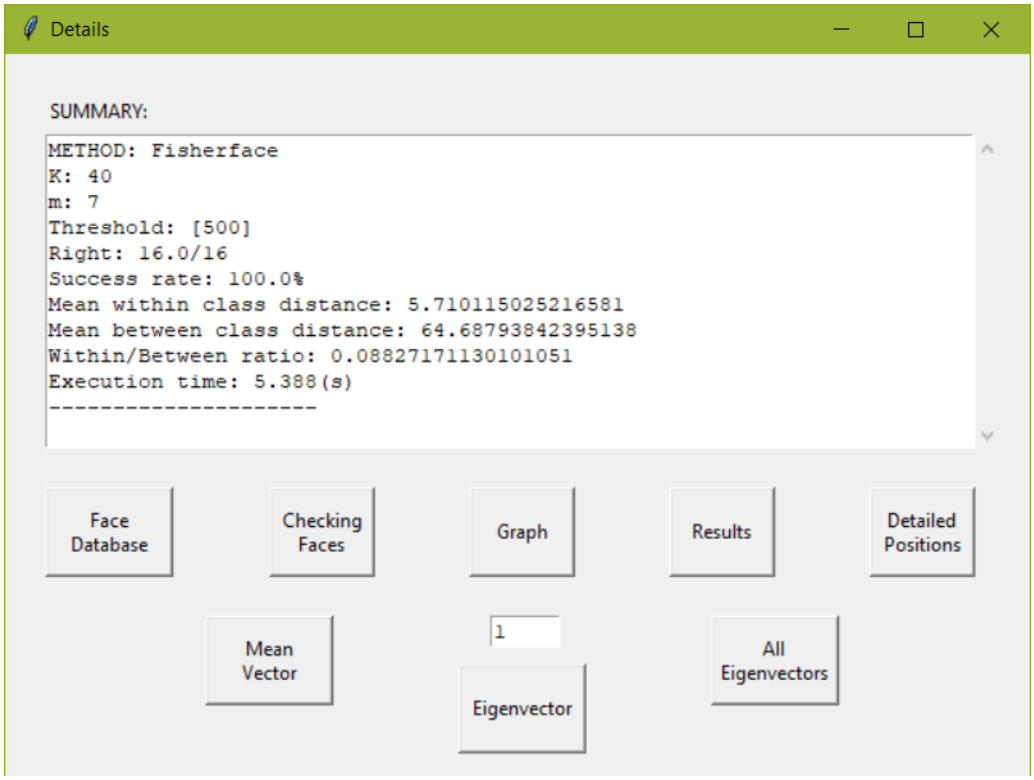

Figure 4.12. *Experiment 1: Fisherface results.*

Discussion:

As we can see, Fisherface had better results than Eigenface with 100% compared to 87.5% Success Rate. However, the trade-off was the execution time. Fisherface required approximately 2.5 times slower than the Eigenface (8.995 vs 3.447).

Taking a deeper look, the performance of two methods can be explained thoroughly via Within/Between Ratio. Eigenface statistic was 0.94 while the corresponding one of Fisherface was 10 times smaller (0.088). This huge different distance between them had led to two different scenarios, which could be illustrated more clearly by plotting the position of each training image in 2-D face space taking the first two principal components.

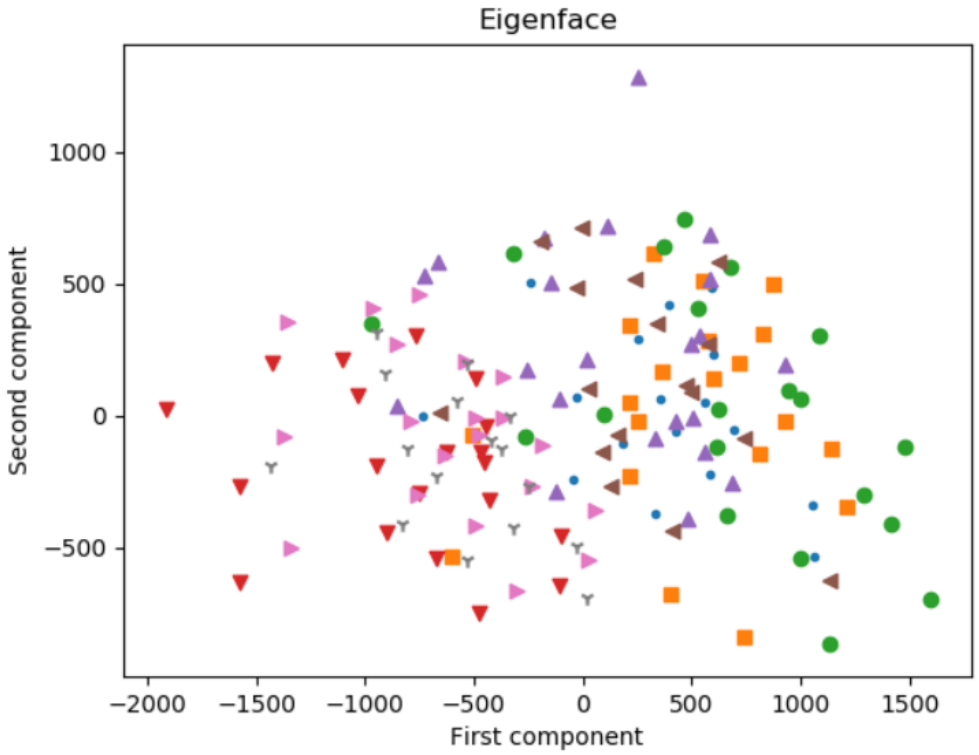

Figure 4.13. *Experiment 1: 2-D Eigenface space.*

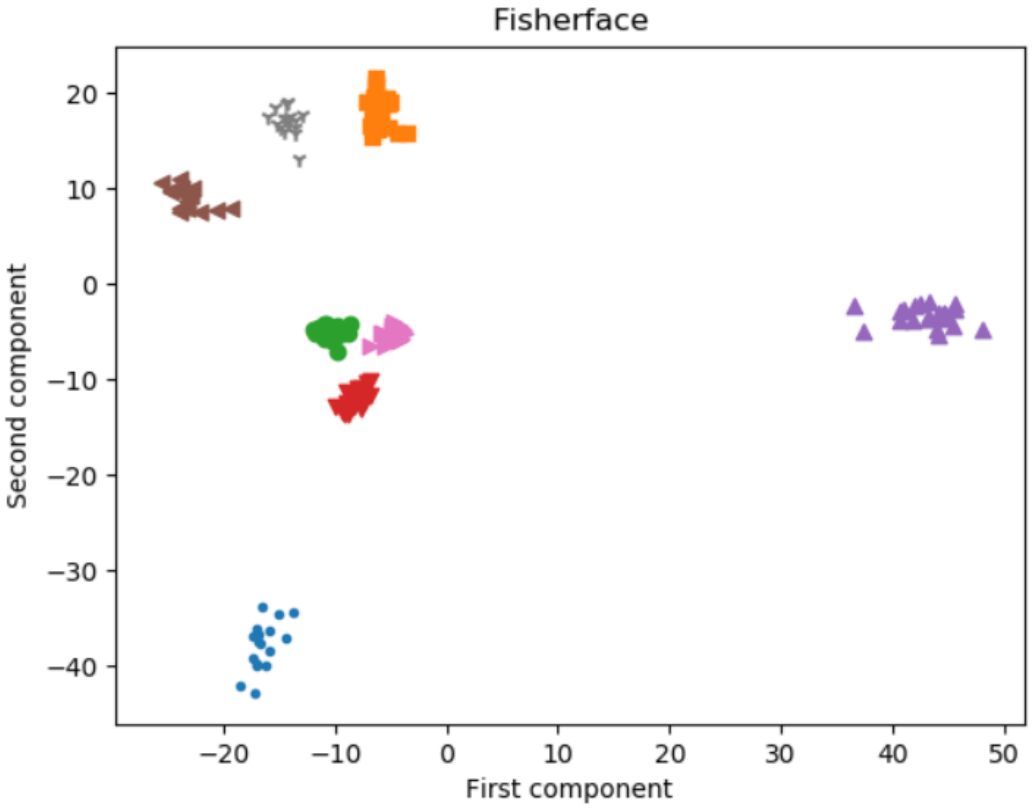

Figure 4.14. *Experiment 1: 2-D Fisherface space.*

For Eigenface's graph, it looks no more than a mess. The positions of images distributed everywhere randomly in the graph. People whose faces were completely different had some images which were very close. Meanwhile, the images belonged to the same person might stay too far from each other's, which in turn resulted in a higher wrong recognition rate.

On the other hand, Fisherface method showed us a more orderly and neatly arranged graph where images within a class lied near to each other's. Moreover, the classes were separated clearly by large distances. This advantage, undoubtedly, is the main reason behind the success of this method. The only drawback, as mentioned above, is the longer execution time due to the higher amount of calculation required to perform.

Another point which is also worth indicating that the performance of Eigenface could be improved by ignoring some first principal components [10]. However, doing this way may result in losing important features which is necessary for recognizing faces.

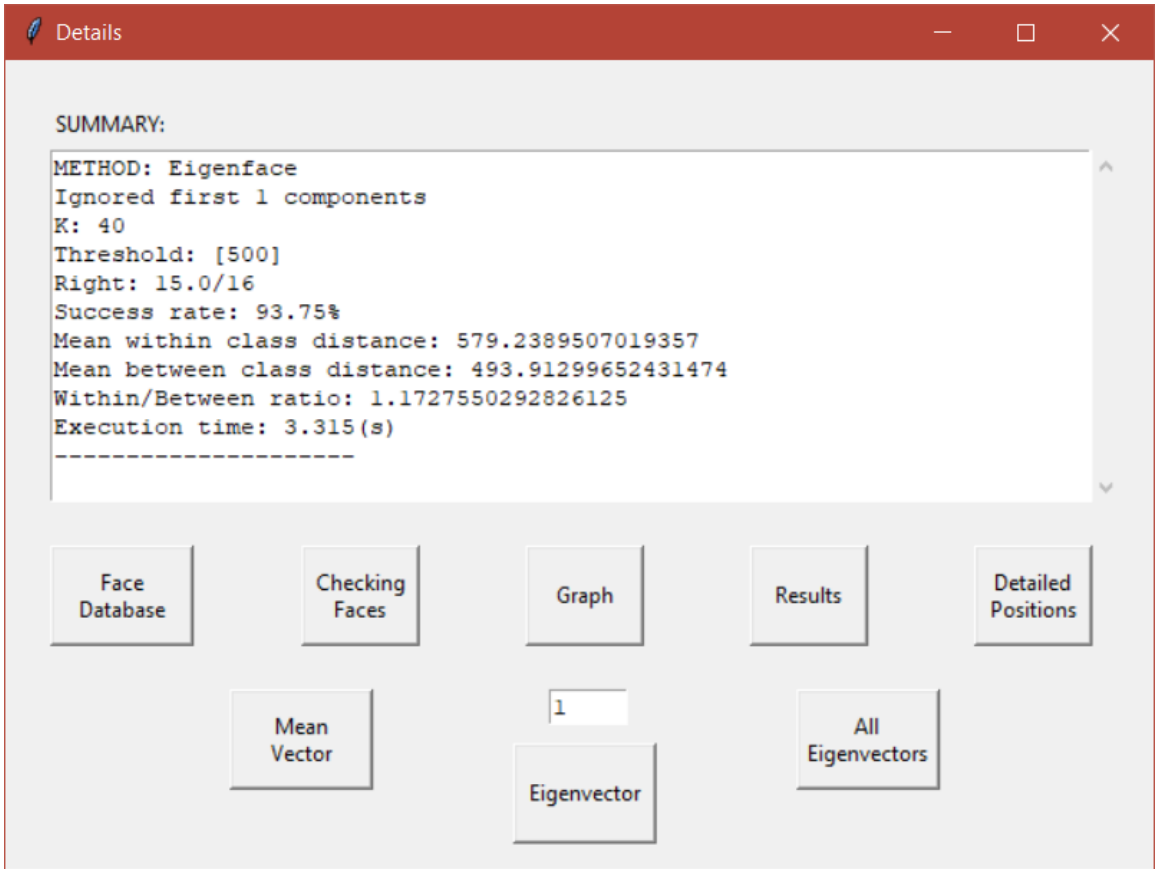

Figure 4.15. *Experiment 1: Eigenface ignoring the first component results.*

Experiment 2: Recognizing ideal input images with not only ideal training images.

In database, there were totally 496 images taken from 8 different people in different lighting angles. Each one had about 61 - 63 images. Test images were 16 images used in the experiment 1. $K = 40$ for Eigenface and $m = 7$ for Fisherface. $\theta_{\varepsilon 1} = 500$ and the dimensions of each image were $168\ x\ 192$. Down sample factor was 2.

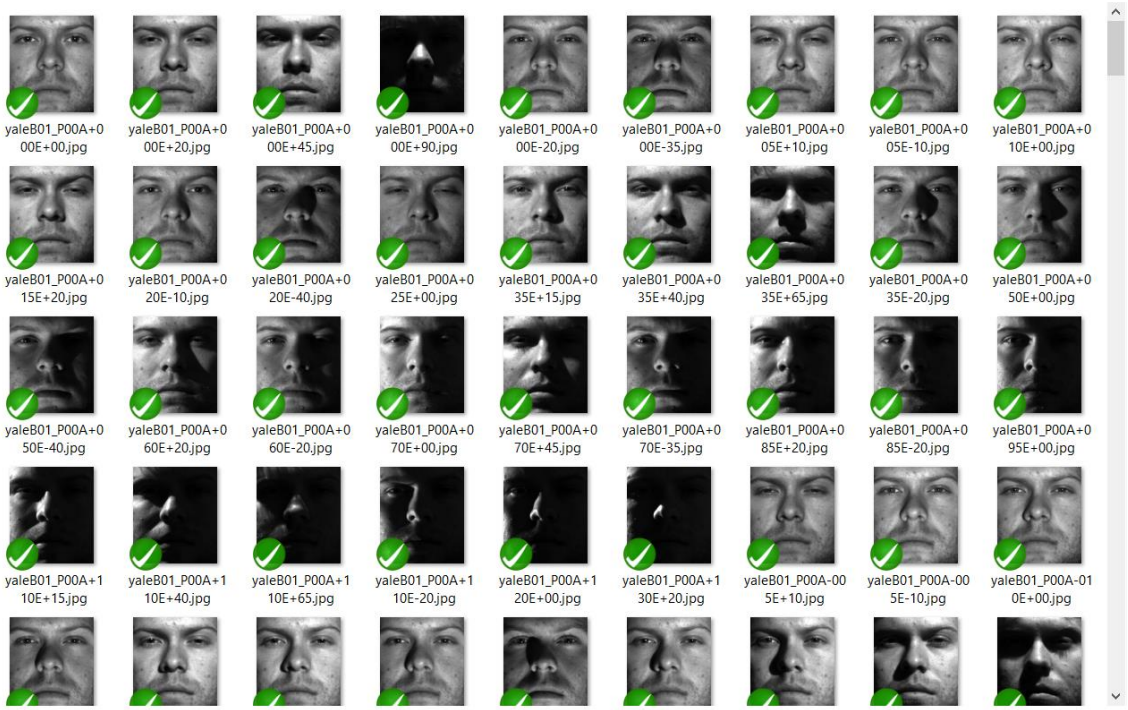

Figure 4.16. *Experiment 2: 496 training images.*

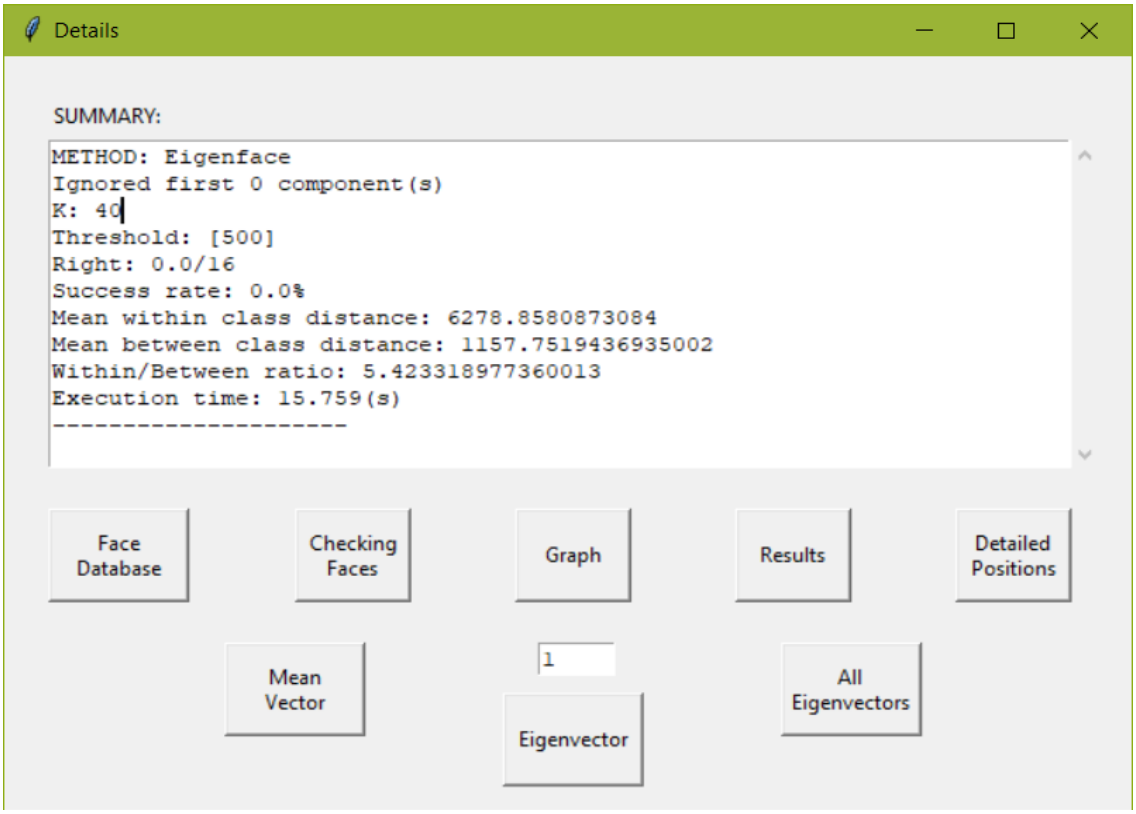

Figure 4.17. *Experiment 2: Eigenface results.*

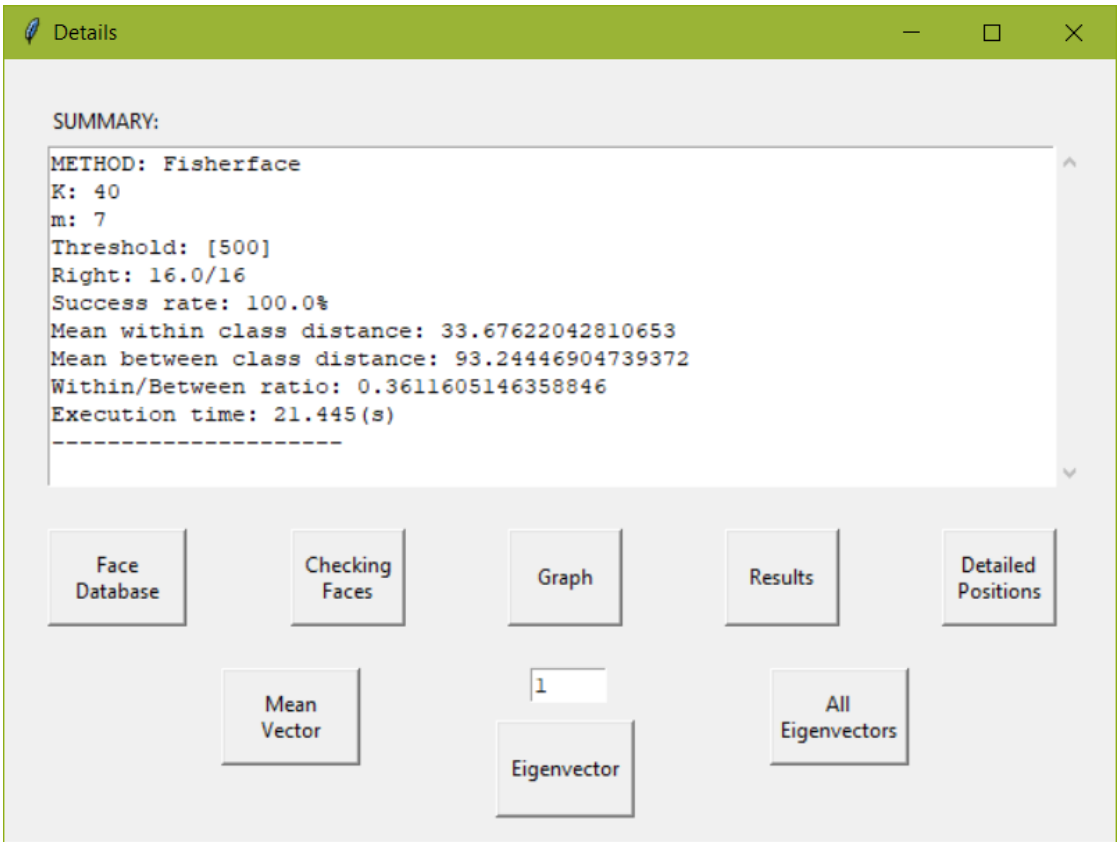

Figure 4.18. *Experiment 2: Fisherface results.*

Discussion:

The Fisherface method continued to dominate this competition with 100% compared to 0% Success Rate. The surprising point here was that the Eigenface had completely failed to recognize any test images. In addition, it is noticeable that the Within/Between Ratio of Eigenface was more 17 times higher than corresponding ratio of Fisherface and almost 6 times higher than its own previous ratio in experiment 1. This would be illustrated more obviously by comparing two face space graphs:

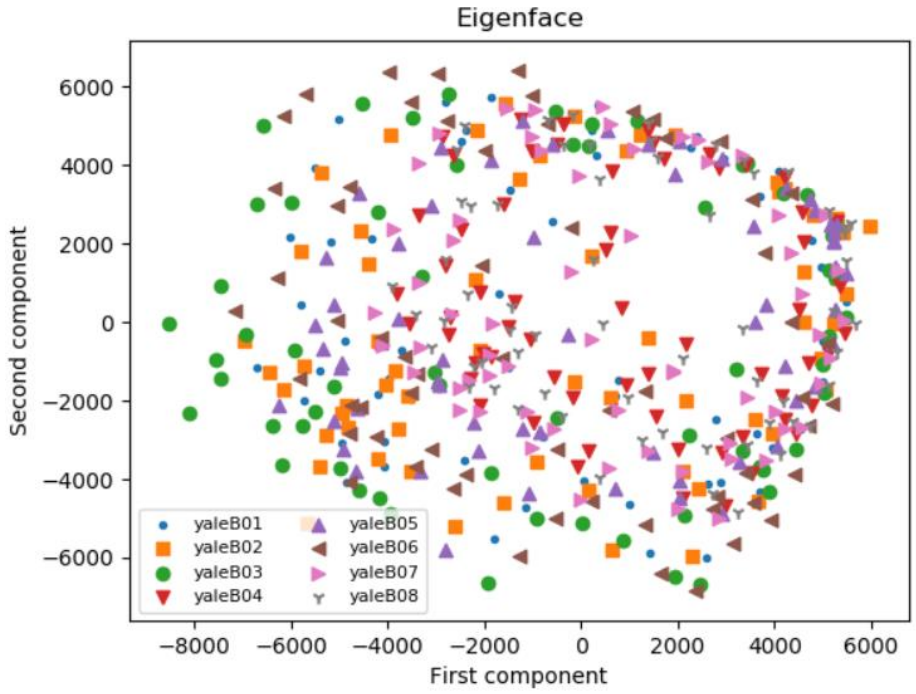

Figure 4.19. *Experiment 2: 2-D Eigenface space.*

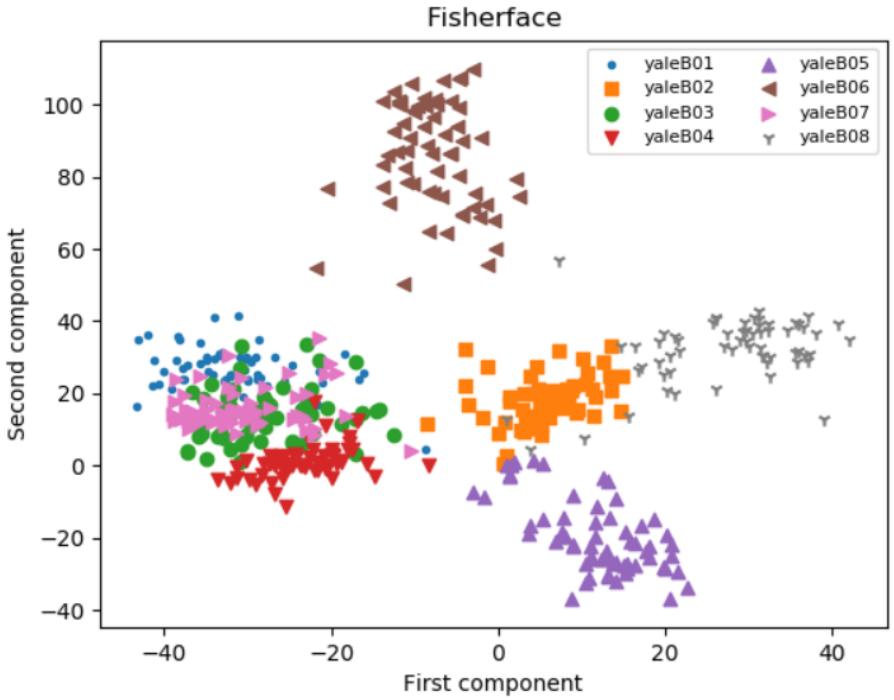

Figure 4.20. *Experiment 2: 2-D Fisherface space.*

The graph of Eigenface now looks much worse than before with the existence of lighting images in the database. The Fisherface's graph would not be as good as we expected, however, it is somehow acceptable and still be a lot better than the one of Eigenface. Some images from different classes seem to be mixed together in Fisherface's graph, but this is just a 2-D graph while the images are actually represented in 7-D space. Therefore, some images are very close to each other's in a 2-D space might be very far when being represented fully in 7-D space.

In general, from two conducted experiments, we could come to the conclusion that Fisherface outperforms Eigenface in all situations with the price of execution time and computer resources. Besides, when testing with ideal input images, the lighting images included in the database may cause the downgrade in the performance of both methods, which is significant for the Eigenface and considerable for the Fisherface method.

Again, we can enhance the Eigenface method performance by ignoring the first principal component.

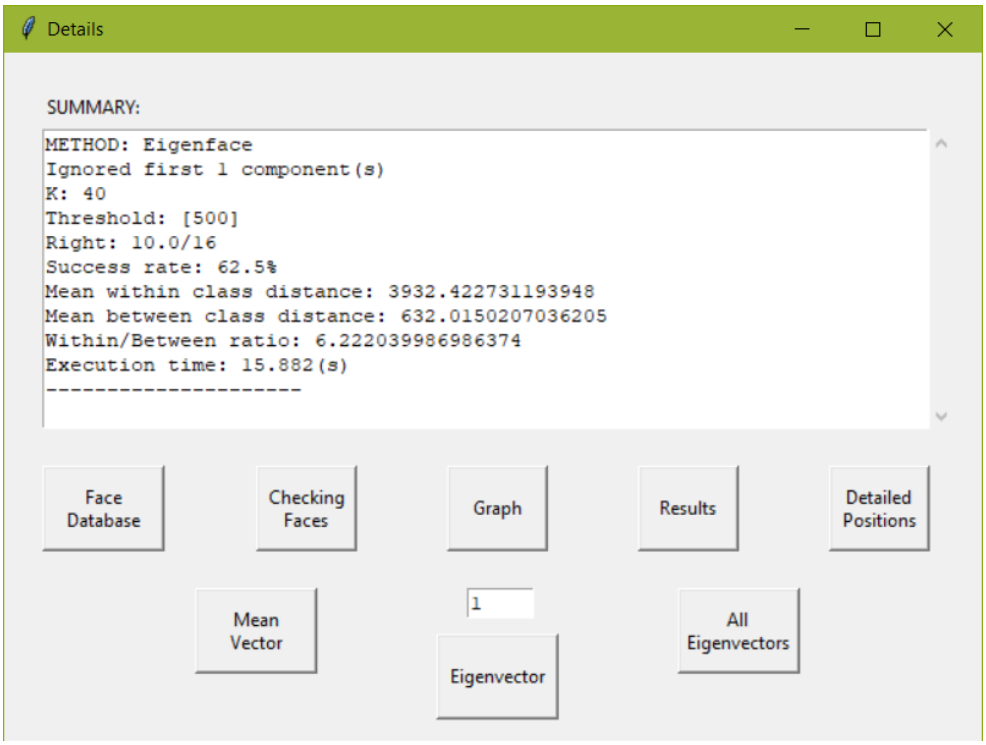

Figure 4.21. *Experiment 2: Eigenface ignoring the first component results.*

Experiment 3: Recognizing not only ideal input images with only ideal training images.

In database, there were totally 168 ideal images taken from 8 different people (almost the same as the one used in experiment 1). Each one had about 19 - 21 images. Test images were 16 random images with different lighting angles of those 8 people (2 for each). $K = 40$ for

Eigenface and $m = 7$ for Fisherface. $\theta_{\varepsilon 1} = 500$ and the dimensions of each image were $168\ x\ 192$. Down sample factor was 2. And here are the results.

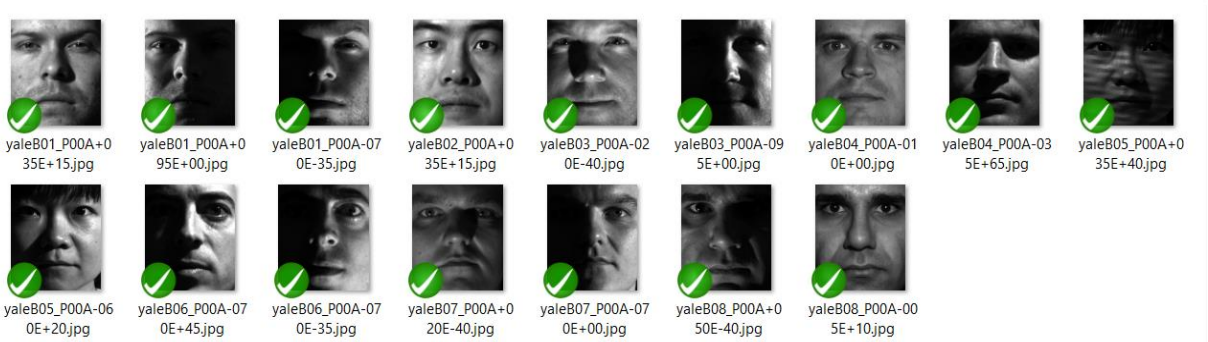

Figure 4.22. *Experiment 3: 16 test images with different angles of light.*

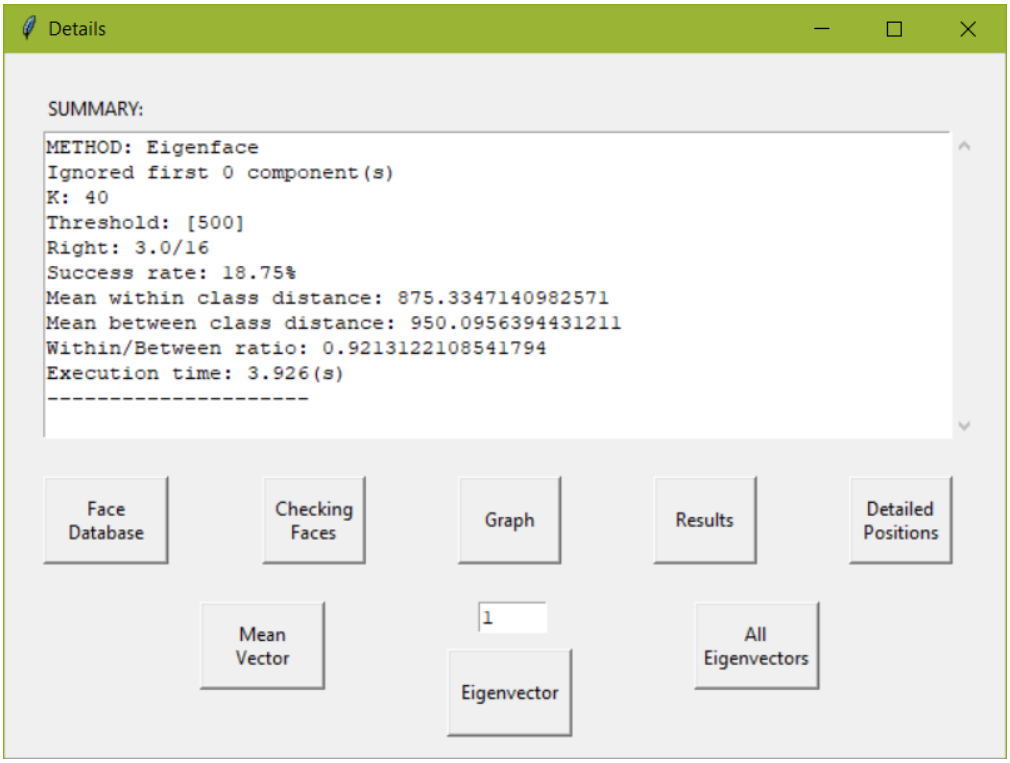

Figure 4.23. *Experiment 3: Eigenface results.*

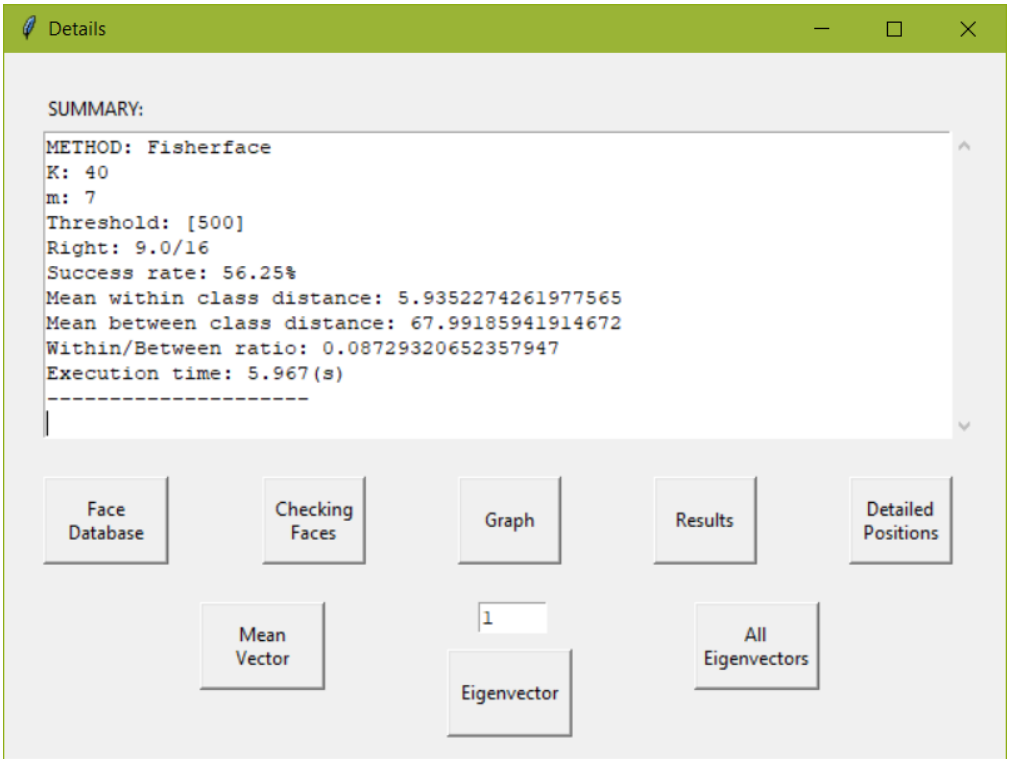

Figure 4.24. *Experiment 3: Fisherface results.*

Discussion:

It is not surprising that Fisherface's Success Rate and Within/Between Ratio were 3 times and 11 times higher than the corresponding ones of Eigenface. However, when comparing to experiment 1, the overall results of both methods had been declined significantly (87.5% to 18.75% for Eigenface and 100% to 56.25% for Fisherface) although the Within/Between Ratio remained nearly unchanged which means that the graphs would not be too different from themselves in experiment 1.

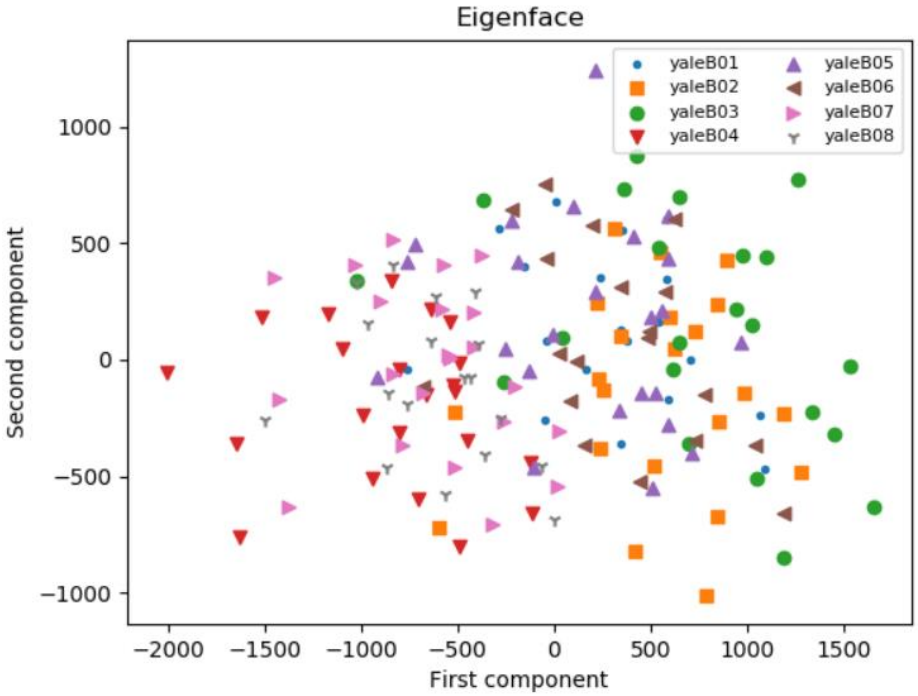

Figure 4.25. *Experiment 3: 2-D Eigenface space.*

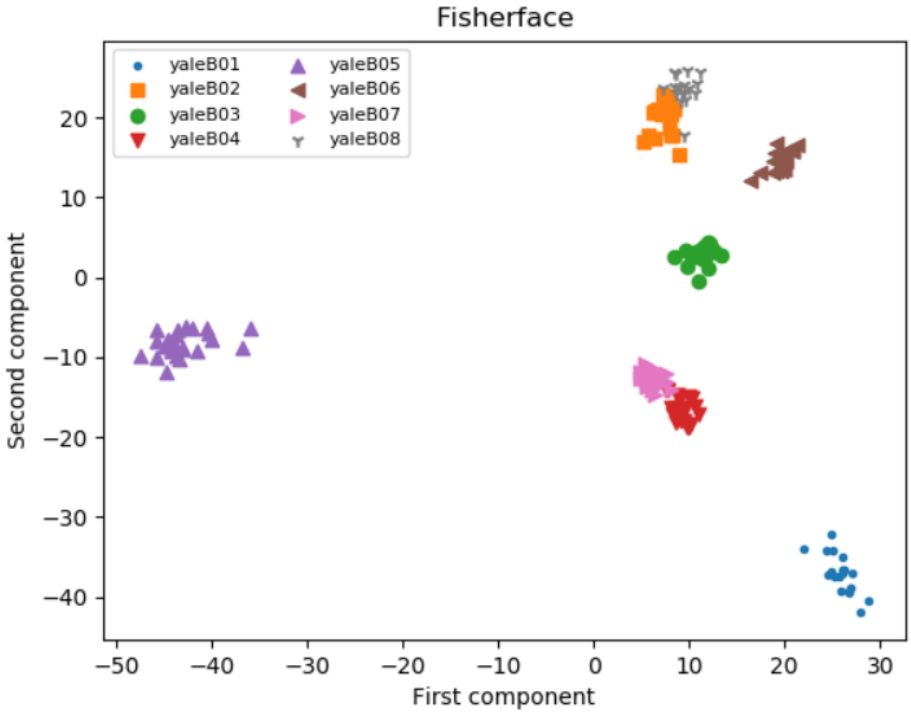

Figure 4.26. *Experiment 3: 2-D Fisherface space.*

It is not something strange for Eigenface that including test images with different lighting angles makes the classification more difficult, while for Fisherface, the explanation may be quite different. Fisherface still accounted for the label (owner) of each image, however, in this circumstance, because the database only stored ideal images (without lighting angles), when applying the algorithm, Fisherface just looked for a new coordinate system which is best for classifying those ideal images – not for lighting images. Therefore, when we projected lighting input images on an unsuitable face space, bad overall results were inevitable.

Experiment 4: Recognizing not only ideal input images with not only ideal training images.

In the database, there were totally 496 images taken from 8 different people in different lighting angles (almost the same as the one used in experiment 2). Each one had about 61 - 63 images. Test images were 16 random images with different lighting angles of those 8 people (2 for each). $K = 40$ for Eigenface and $m = 7$ for Fisherface. $\theta_{\varepsilon 1} = 500$ and the dimensions of each image were $168\ x\ 192$. Down sample factor was 2. The results are shown below.

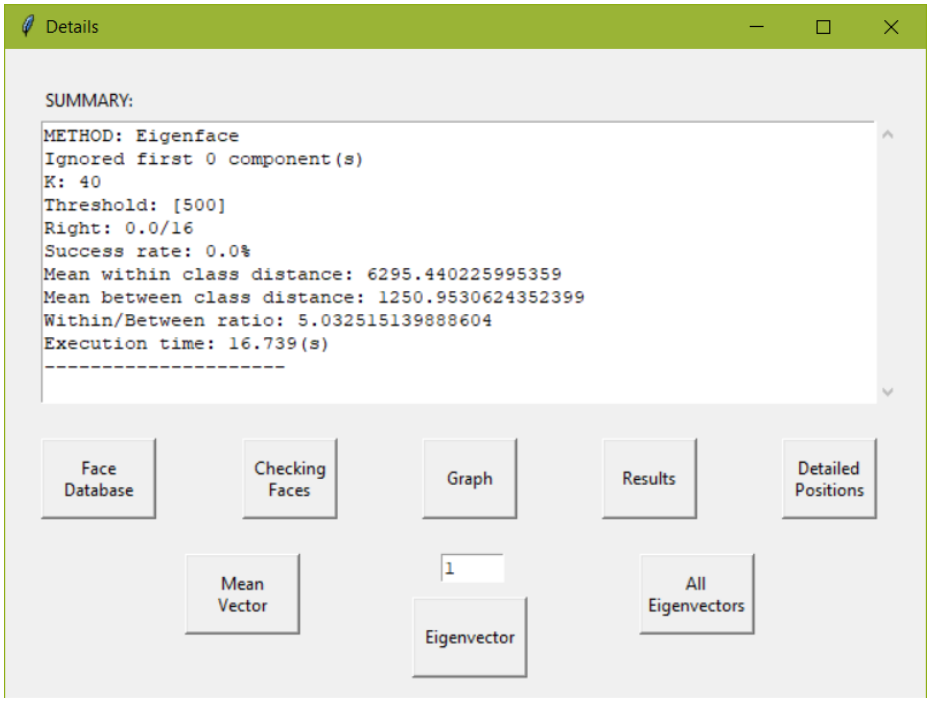

Figure 4.27. *Experiment 4: Eigenface results.*

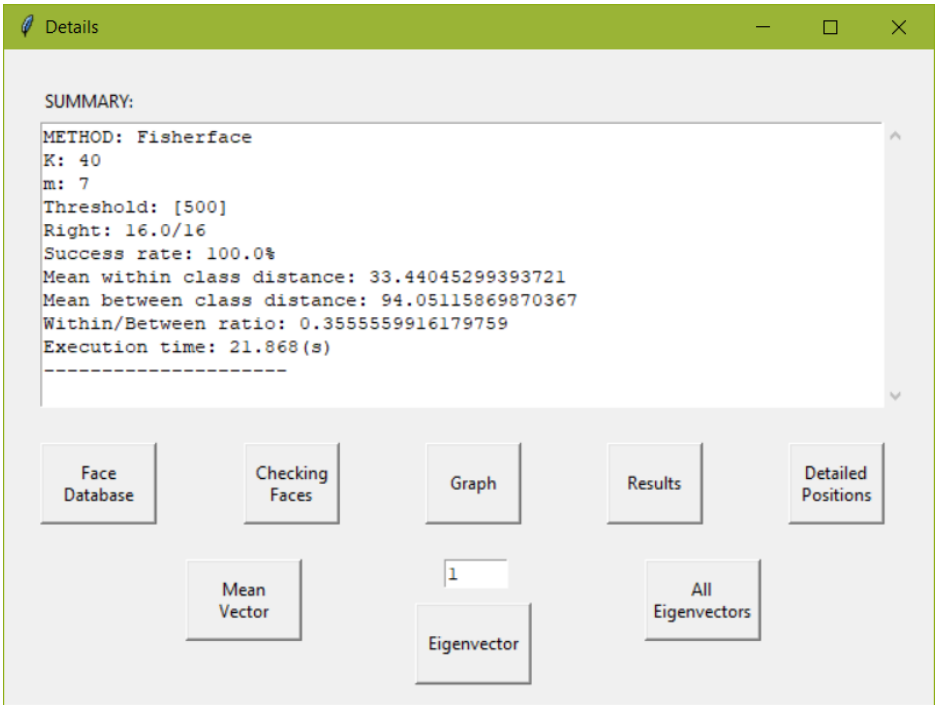

Figure 4.28. *Experiment 4: Fisherface results.*

The results of Eigenface were the same as in experiment 2. This is understandable because, from experiment 2 and 3, we have already known that unideal input and training images both make a severe impact on its performance. Hence, it is not strange for 0% Success Rate of Eigenface. Fisherface, on the other hand, came back to its original 100% in this experiment. This result, to some extent, could be expected from the last experiment. This time, not only ideal images but also images with different lighting angles were trained in our system. Therefore, Fisherface would have enough information to decide the best coordinate system which could handle ideal and unideal images as well.

As usual, when it comes to graphs, Fisherface's graph is a lot better than the one of Eigenface. These two graphs look almost the same as themselves in experiment 2.

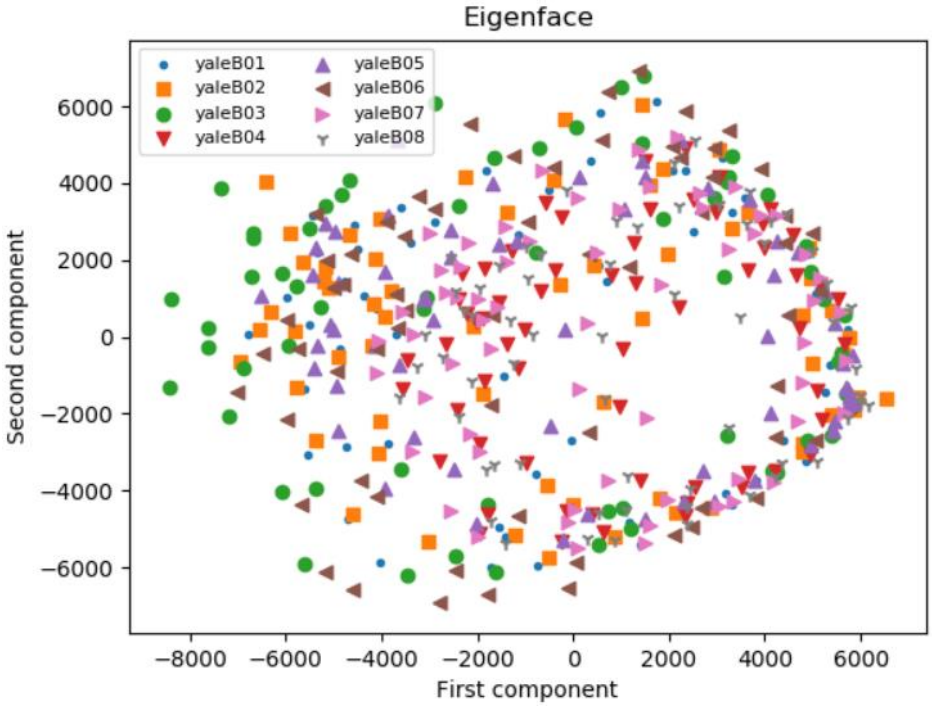

Figure 4.29. *Experiment 4: 2-D Eigenface space.*

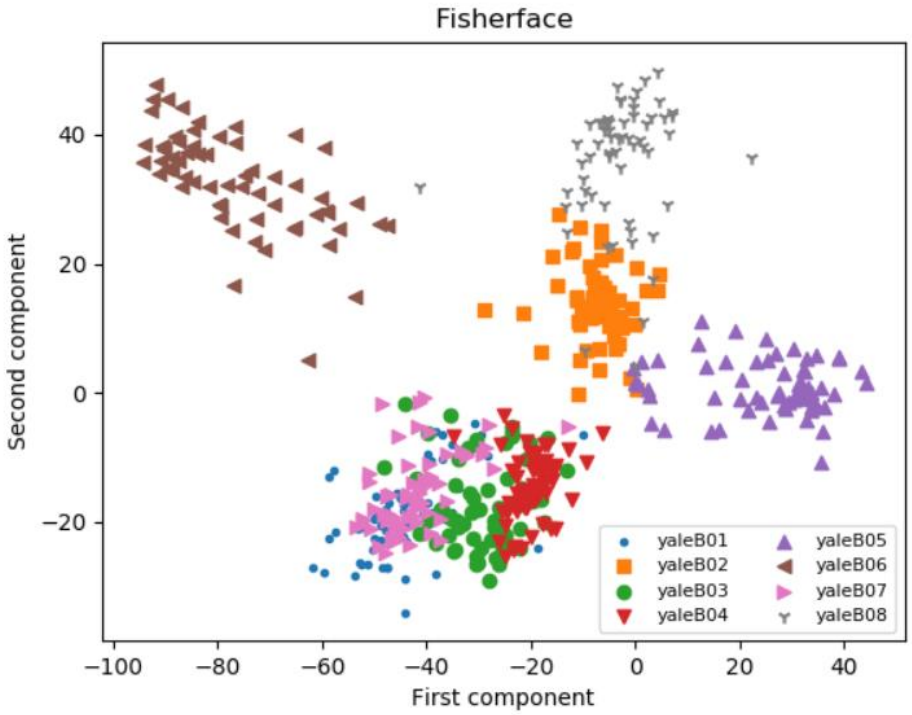

Figure 4.30. *Experiment 4: 2-D Fisherface space.*

5. Conclusion and future works

For many past decades, face recognition remained to be one of the most fabulous dreams of humankind. However, with countless efforts and non-stop workings of many great scientists, it has become such a quite common thing in our society nowadays. A significant number of methods have been come up with in order to enhance its efficiency day-by-day.

Through the project, Eigenface and Fisherface – two of the most popular techniques – were thoroughly examined and compared in many aspects like execution time, algorithm calculation, graph, Success Rate and Within/Between Ratio. For Eigenface, thanks to the Principal Component Analysis method, many important characteristics of a human face are extracted successfully via Eigenfaces (eigenvector), which will then be filtered by their corresponding eigenvalues. Fisherface is an upgraded version of Eigenface taking advantage of Linear Discriminant Analysis to achieve better results despite requiring more calculation as well as execution time.

The only thing Eigenface method cares is finding out a new coordinate system which maximizes the total variance of all training images in the database. As an unsupervised learning algorithm, it does not pay any attention to data information such as the number of classes in a dataset or the number of images within each class. Meanwhile, those pieces of information are all taken care of by Fisherface. Fisherface knows how many images belonging to a class and how many classes in total, which in turn leads to much better performance in all situations.

Both techniques, especially Eigenface, were exposed to have problem when dealing with untrained faces with different lighting angles, which is also a limitation of all 2-D face recognition methods nowadays. For the Fisherface method, it is possible to achieve a better recognition rate by putting more unideal images in the database, while in practice, one can enhance the performance of Eigenface by discarding some first principal components. However, doing this trick for Eigenface may lead to the loss of important useful features which are necessary for further classification and examination such as face reconstruction.

In the future, the thesis could be extended in many directions:

- Establishing a larger and more diverse database.
- Directly doing a comparison with many other 2-D face recognition techniques.
- Optimizing the codes to improve the processing rate.
- Combining with a real-time camera system.

References

[1] Gagandeep Sethi, “Facial recognition technology is gaining prominence in mobile apps: A reflection on the current state of affairs,” *Promatics Blog*, February 24, 2018. [Online]. Available: https://www.promaticsindia.com/blog/facial-recognition-technology-is-gaining-prominence-in-mobile-apps-a-reflection-on-the-current-state-of-affairs/. [Accessed October 07, 2018].

[2] Rachna Singh, “Facial Recognition Market by Technology (2D Facial Recognition, 3D Facial Recognition and Facial Analytics), Component (Hardware and Software) and Application (Homeland Security, Criminal Investigation, ID Management, Physical Security, Intelligent Signage, Photo Indexing and Sorting, Business Intelligence and Photo Indexing and Sorting) - Global Opportunity Analysis and Industry Forecast, 2015 - 2022,” Allied Market Research, June 2016.

[3] Maraparacc, “Swiss European surveillance: facial recognition and vehicle make, model, color and license plate reader. Used in Germany and Switzerland for tracking you and logging your movement for future reference,” January 07, 2008, 2,410 × 1,620 pixels. [Online]. Available: https://en.wikipedia.org/wiki/File:Surveillance_equipment_5413.jpg. [Accessed October 07, 2018].

[4] W. W. Bledsoe, “The model method in facial recognition,” Panoramic Research Inc., Palo Alto, CA, Rep. PRI:15, August 1966.

[5] T. Kanade, “Picture processing system by computer complex and recognition of human faces,” Dept. of Information Science, Kyoto University, November 1973.

[6] A. L. Yuille, D. S. Cohen and P. W. Hallinan, “Feature extraction from faces using deformable templates,” In Proc. CVPR , San Diego, CA, June 1989.

[7] Kohonen Teuvo, “Self-Organized Formation of Topologically Correct Feature Maps,” Department of Technical Physics, Helsinki University of Technology, Espoo, Finland, 1982.

[8] S. Lawrence, C.L. Giles, Ah Chung Tsoi and A.D. Back, “Face recognition: a convolutional neural-network approach,” *IEEE Transactions on Neural Networks*, Vol. 8, No. 1, January 1997.

[9] L. Sirovich and M. Kirby, “Low-dimensional procedure for the characterization of human faces,” *Journal of the Optical Society of America A*, Vol. 4, No. 3, pp. 519-524, March 1987.

[10] Peter N. Belhumeur, João P. Hespanha and David J. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, July 1997.

[11] Mark Williams Pontin, “Better Face-Recognition Software,” Intelligent Machines, MIT Technology Review, May 2007. [Online]. Available: https://www.technologyreview.com/s/407976/better-face-recognition-software/. [Accessed October 07, 2018].

[12] David Paul Morris, “Phil Schiller, senior vice president of worldwide marketing at Apple Inc., speaks about the iPhone X and the new FaceID feature on September 12, 2017,” September 12, 2017, Bloomberg via Getty Images. [Online]. Available: https://www.wired.com/story/iphone-x-faceid-security/. [Accessed October 07, 2018].

[13] Fahad Layth Malallah, Sharifah Mumtazah Syed Ahmad, Wan Azizun Wan Adnan, Olasimbo Ayodeji Arigbabu, Vahab Iranmanesh and Salman Yussof, “Online Handwritten Signature Recognition by Length Normalization using Up-Sampling and Down-Sampling,” *International Journal of Cyber-Security and Digital Forensics*, Vol. 4, No. 1, The Society of Digital Information and Wireless Communications, 2015.

[14] Zahraddeen Sufyanu, Fatma Susilawati Mohamad, Abdulganiyu Abdu Yusuf, Abdulbasit Nuhu Musa and Rabi’u Abdulkadir, “Feature Extraction Methods for Face Recognition,” *International Review of Applied Engineering Research (IRAER) ISSN 2248-9967*, Vol. 5, No. 3, 2016, pp. 5658-5668.

[15] D. Mohamed, “Trends and Challenges in Mono and Multi Biometrics”, *Image Processing Theory, Tools & Applications*, 2008.

[16] Bibek Joshi, comment on “What is the difference between a black and white and grayscale image?,” posted on November 09, 2017. [Online]. Available: https://www.quora.com/What-is-the-difference-between-a-black-and-white-and-grayscale-image. [Accessed October 07, 2018].

[17] Cindy Zapata, MODELO RGB, February 15, 2013. [Online]. Available: https://cindy2906.blogspot.com/2013/02/modelo-rgb.html. [Accessed October 07, 2018].

[18] Kemal Erdoğan and Nihat Yilmaz, “Shifting Colors to Overcome not Realizing Objects Problem due to Color Vision Deficiency,” 2nd Int. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2014, Kuala Lumpur, Malaysia, December 2014.

[19] Spitzak, “The RGBA sample image composited atop checkerboard, as most browsers do not do this themselves any more,” April 25, 2018, 240 × 240 pixels. [Online]. Available: https://commons.wikimedia.org/wiki/File:RGBA_comp.png. [Accessed October 07, 2018].

[20] Liton Chandra Paul and Abdulla Al Sumam, "Face Recognition Using Principal Component Analysis Method," *International Journal of Advanced Research in Computer Engineering & Technology*, Vol. 1, No. 9, November 2012.

[21] Bugra Akyildiz, “PCA, EigenFace and All That,” July 27, 2013. [Online]. Available: http://bugra.github.io/work/notes/2013-07-27/PCA-EigenFace-And-All-That/. [Accessed July 22, 2018].

[22] Braeken, Johan and Marcel A. L. M. van Assen. “An empirical Kaiser criterion,” *Psychological methods,* Vol. 22, No. 3, 2017.

[23] Marco Querini and Giuseppe F.Italiano, “Facial recognition with 2D color barcodes,” *International Journal of Computer Science and Applications,* Vol. 10, No. 1, pp. 78-97.

[24] Matthew A. Turk and Alex P. Pentland, “Face Recognition Using Eigenfaces,” Vision and Modeling Group, The Media Laboratory, Massachusetts Institute of Technology, 1991.

[25] Müge Çarıkçı and Figen Özen, “A Face Recognition System Based on Eigenfaces Method,” Haliç University, Electrical and Electronics Engineering Department, Şişli, Istanbul, Turkey.

[26] Ronald A. Fisher, “The Use Of Multiple Measurements In Taxonomic Problems,” *Annals of Human Genetics*, Vol. 7, Issue 2, pp. 179-188, September 1936.

[27] C. Radhakrishna Rao, “The Utilization of Multiple Measurements in Problems of Biological Classification,” *Journal of the Royal Statistical Society. Series B (Methodological),* Vol. 10, No. 2, 1948, pp. 159-203.

[28] Sebastian Raschka, “Linear Discriminant Analysis – Bit by Bit,” August 3, 2014. [Online]. Available: https://sebastianraschka.com/Articles/2014_python_lda.html. [Accessed October 07, 2018].

[29] Athinodoros Georghiades, Peter N. Belhumeur and David J. Kriegman, "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose," *IEEE Trans. Pattern Anal. Mach. Intelligence,* Vol. 23, No. 6, 2001, pp. 643-660.