2014

Comparison of Computer-Based and Optical Face Recognition Paradigms

Abdulaziz A. Alorf
Michigan Technological University

COMPARISON OF COMPUTER-BASED AND OPTICAL FACE RECOGNITION PARADIGMS

By

Abdulaziz A. Alorf

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Electrical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2014

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Electrical Engineering.

Department of Electrical and Computer Engineering

Thesis Advisor:	*Professor, Michael C. Roggemann*
Committee Member:	*Associate Professor, Christopher T. Middlebrook*
Committee Member:	*Assistant Professor, Timothy Havens*
Committee Member:	*Associate Professor, Aleksandr V. Sergeyev*
Department Chair:	*Professor, Daniel R. Fuhrmann*

To my mother, father and wife.

Contents

List of Figures

List of Tables

Acknowledgements

There are a number of people without whom this thesis might not have been written and to whom I am greatly indebted. I would like to take this opportunity to look back and acknowledge people who have helped to shape me into the successful young man I am today.

First of all, I am grateful to the Almighty God for all the blessings that you have bestowed on me.

A very special thank you to my parents who continue to grow, learn and develop; and who have been a source of encouragement and inspiration to me throughout my life.

To my dear wife, Hajar, who remains willing to engage with the struggle and to tolerate my long busyness in research. Thank you so much for your practical and emotional support.

A special note of thanks to my sponsor, Qassim University in Saudi Arabia for granting me this scholarship to pursue my graduate studies. Thank you from the bottom of my heart for supporting me since I was one of your students at the College of Engineering.

I would like to express my deepest appreciation to Prof. Sulaiman A. Al-Yahya, the vice president of Qassim University for planning, development and quality; and Dr. Abdulrahman F. Almarshoud, the dean of Engineering College at Qassim University; for their continuous support. You are a source of encouragement and inspiration for me. Everytime I visit or call, you try your best to push me forward regardless of any difficulty.

My journey here at Michigan Technological University has been challenging. I have enjoyed all of the opportunities that have come my way. I have acquired such an immense wealth of knowledge and experience throughout my time here that I will likely owe most of my future professional success to the four years I have spent up here in the freezing cold. I want to take this opportunity to send my warmest thanks to this institution.

I wish to express my sincere thanks to my advisor Prof. Michael C. Roggemann

for his kindness, expertise, sincerity, valuable guidance and encouragement. You have actively supported me in my determination to find and realize my potential and to make this contribution to our world. I recall that when I faced a problem in my research, I would email you with a description of the problem; then you sometimes replied with "Let's chat about that tomorrow in my office" which means that it is not an easy problem. I want to tell you that I really hate the word "chat" because it means that I have to afford more effort and time. I am really proud that you have directed my thesis.

I would like to say thank you to my friends Mr. Mohamed Elsayed Hanafy, Mr. Mathkar Alawi Alharthi and Mr. Mansour Thuwaini Al-Shammari for providing me with images of their faces to use in my research. It is impossible to forget that when I call you to schedule a session for taking some photos, you were willing to come and offer any time I needed. I really appreciate your cooperation.

I would like to thank Dr. Jeffrey B. Burl for providing me with essential suggestions on different aspects of my research.

I also want to thank Dr. Glen E. Archer for teaching me some powerful research tools. Thank you very much for offering me time and lots of effort.

Thanks with all of my heart to Dr. Imran Aslam and Mr. Ehsan Taheri for indispensable advice, information and support throughout my master's degree.

I take this opportunity to record my sincere thanks to my committee members Dr. Christopher T. Middlebrook, Dr. Timothy Havens and Dr. Aleksandr V. Sergeyev for their willingness to be on the committee and taking the time to work on my thesis.

I am also very grateful to Mr. Jim LaBeske and Ms. Jamie Peryam who knowingly and unknowingly have made my and my family's stay in Houghton a lot easier than we thought it was going to be.

Finally, I also place on record my sense of gratitude to one or all who directly or indirectly have lent their helping hand in this thesis or have supported me.

Abstract

The main objectives of this thesis are to validate an improved principal components analysis (IPCA) algorithm on images; designing and simulating a digital model for image compression, face recognition and image detection by using a principal components analysis (PCA) algorithm and the IPCA algorithm; designing and simulating an optical model for face recognition and object detection by using the joint transform correlator (JTC); establishing detection and recognition thresholds for each model; comparing between the performance of the PCA algorithm and the performance of the IPCA algorithm in compression, recognition and, detection; and comparing between the performance of the digital model and the performance of the optical model in recognition and detection. The MATLAB © software was used for simulating the models.

PCA is a technique used for identifying patterns in data and representing the data in order to highlight any similarities or differences. The identification of patterns in data of high dimensions (more than three dimensions) is too difficult because the graphical representation of data is impossible. Therefore, PCA is a powerful method for analyzing data. IPCA is another statistical tool for identifying patterns in data. It uses the information theory for improving PCA. The joint transform correlator (JTC) is an optical correlator used for synthesizing a frequency plane filter for coherent optical systems.

The IPCA algorithm, in general, behaves better than the PCA algorithm in the most of the applications. It is better than the PCA algorithm in image compression because it obtains higher compression, more accurate reconstruction, and faster processing speed with acceptable errors; in addition, it is better than the PCA algorithm in real-time image detection due to the fact that it achieves the smallest error rate as well as remarkable speed. On the other hand, the PCA algorithm performs better than the IPCA algorithm in face recognition because it offers an acceptable error rate, easy calculation, and a reasonable speed. Finally, in detection and recognition, the performance of the digital model is better than the performance of the optical model.

Chapter 1

Introduction

1.1 Background

1.1.1 Digital Processing

Principal components analysis (PCA) [1] and improved principal components analysis (IPCA) [2] are statistical tools frequently used for analyzing data. Their main applications are pattern recognition such as face detection and recognition, and data compression such as image compression.

PCA is a technique used for identifying patterns in data and representing the data in such a way that their similarities and differences are highlighted. The identification of patterns in data of high dimensions (more than three dimensions) is too difficult because the graphical representation of data is impossible. Therefore, PCA is a powerful method for analyzing data. The PCA algorithm starts with the creation of a data set and ends with the projection of the data on the eigenspace. A covariance matrix is computed for the data; in addition, the eigenvectors and eigenvalues of the covariance matrix are obtained. Eigenvectors associated with the biggest eigenvalues of the covariance matrix follow the most significant patterns of the data. Those eigenvectors are called the principle components of the data set. Therefore, the eigenvalues of the covariance matrix work as measures of how much information is contained in each of the principal components. The principal components form a feature vector matrix. In order to select principal components that form the feature vector matrix, the variance contribution rate (VCR) and the total variance contribution rate (TVC) (they are proposed in the IEEE paper presented in Reference [2]) are computed. When the TVC is significantly high then q eigenvectors associated with the biggest q eigenvalues can be selected. The feature vector matrix is used for projecting the data on the eigenspace. Finally, by projecting the data on the eigenspace, the PCA algorithm is completed.

IPCA is another statistical tool for identifying patterns in data. It is similar to PCA except for the way that it selects eigenvectors that form the feature vector matrix. It affords a new accurate method to measure the information content of the principal components based on the information theory for improving PCA. For measuring the degree of information content of the eigenvectors, two new concepts are used; the first is the information rate (IR) and the second is the accumulated information rate (AIR) (they are proposed in the IEEE paper presented in Reference [2]). When the AIR is significantly high then q eigenvectors associated with the biggest q eigenvalues can be selected.

1.1.2 Optical Processing

Spatially coherent light is going to be used in the optical model. Coherent optical systems are linear in complex amplitude; therefore, filtering processes can be performed by direct manipulation of complex amplitude appearing in the back focal plane of a Fourier transforming lens. There are at least two methods for synthesizing the frequency plane filter for coherent optical systems. One of these methods is by using the joint transform correlator (JTC), Reference [3], Section 8.5.

The JTC is an optical correlator used for synthesizing the frequency plane filter for coherent optical systems. This correlator was invented by Weaver and Goodman [4]. The filter is divided into two stages: recording the filter, and getting the filter output. The transparencies of the desired impulse response h and the data g (here it is called the object) to be filtered are aligned simultaneously in the input plane. They are then Fourier transformed together. At that point, a spatial light modulator (SLM) captures the intensity distribution of the transformed field. The intensity is then Fourier transformed again for producing the cross-correlated field in the output plane. The output field is composed of four terms; two terms respectively represent the cross-correlation of the impulse response h and itself as well as the cross-correlation of the data g and itself; the third and fourth terms represent the cross-correlations of h and g. Lastly, the joint transform correlator has a great feature: its ability to change the filter impulse response quickly. Therefore, it is considered beneficial for real-time systems. On the other hand, its defect is that the input bandwidth of the data is reduced due to the filter impulse response being introduced simultaneously with the data to be filtered.

1.2 Problem Statement

The main objectives of this thesis are to validate the improved principal components analysis (IPCA) algorithm on images; designing and simulating a digital model for image compression, face recognition, and image detection by using the principal components analysis (PCA) algorithm and the IPCA algorithm; designing and simulating

an optical model for face recognition and object detection by using the joint transform correlator (JTC); establishing detection and recognition thresholds for each model; comparing between the performance of the PCA algorithm and the performance of the IPCA algorithm in compression, recognition and detection; and comparing between the performance of the digital model and the performance of the optical model in recognition and detection.

1.3 Technical Approach

1.3.1 Digital Model

1.3.1.1 Introduction

This subsection provides a general overview of technical approaches behind the application of the PCA and IPCA algorithms in image compression, face recognition, and image detection.

Here, the database for each algorithm is composed of some images of faces (training faces). The principal components that form the feature vector matrix are here called eigenfaces.

1.3.1.2 Image Compression

When some of the eigenvectors that are calculated from the covariance matrix for all training faces are selected to form the feature vector matrix then the dimensions of the reconstructed data set will be reduced. This implies that the PCA and IPCA algorithms work as compression. The algorithms are said to be lossy because a decompressed image is not exactly the same as the original one, but is generally worse.

Compression performance for each algorithm as analyzed from three points of view are the speed of compression and reconstruction, the quality of a reconstructed image, and the size of compression. The number of the eigenfaces that is used to compress and reconstruct the training faces mainly controls the processing speed of compression and reconstruction. When a small number of the eigenfaces is used to project and reconstruct the training faces then the processing speed will increase and vice versa. For measuring the quality of a reconstructed image, the mean squared error (MSE) between the image and its reconstruction can be computed. The size of compression can be measured in two ways: these are through the information rate and the mean squared error (MSE) of compressed images. The information rate measures how much information is present after compression compared with information present before compression; in other words, it measures the number of pixels after compression compared to before compression.

1.3.1.3 Face Recognition

The PCA and IPCA algorithms are used to recognize an unknown face image based on the database that contains the training faces. For doing face recognition, an unknown face image is taken. The training faces and the unknown face image are projected on the eigenspace by using the PCA or IPCA feature vector matrix. The Euclidean distance between the projected unknown face image and each projected training face is computed. Then the unknown face image is recognized as a training face, which has the minimum distance from the unknown face image.

Unfortunately, when the unknown face image does not have a similar training face then getting the minimum distance does not *always* mean that the unknown face image is recognized as a training face that has the minimum distance from the unknown face image. Therefore, a certain threshold must be used to increase the accuracy of recognition. For setting up a recognition threshold, the mean and standard deviation (the average distance from the mean to a point) are established for each training face. Then recognition can be updated as, when the obtained minimum distance between the unknown face image and a training face is less than or equal to the mean plus the standard deviation for the training face and bigger than or equal to the mean minus the standard deviation for the training face. At that point, the unknown face image is recognized as that training face; otherwise, it is an unknown face image.

Recognition performance can be analyzed from two points of view: these are the speed of recognition and the error rate. The number of selected eigenfaces that are used to recognize the unknown face image mainly controls the recognition speed. When the number of the selected eigenfaces decreases, the processing speed increases and vice versa. The error rate computes the percentage of error in recognition.

1.3.1.4 Image Detection

The PCA and IPCA algorithms are used to detect whether or not an unknown image contains a face based on a determined threshold for detection. Hence, in image detection, only a detection threshold is needed.

To obtain the detection threshold, the mean and the standard deviation are established for some images that contain faces. In regards to detection, an unknown image is taken. It is projected on the eigenspace and reconstructed again by using the PCA or IPCA feature vector matrix. Then, detection can be performed as if the Euclidean distance between the unknown image and its reconstruction is less than or equal to the computed mean plus standard deviation and bigger than or equal to the computed mean minus standard deviation then the unknown image is detected as a face image; otherwise, it is not a face image.

Detection performance can be analyzed from two points of view: these are the speed of detection and the error rate. The number of selected eigenfaces that are used to detect the unknown image mainly controls the detection speed. When the

number of selected eigenfaces decreases, the processing speed increases and vice versa. The error rate computes the percentage of error in detection.

1.3.2 Optical Model

1.3.2.1 Introduction

This subsection provides a general overview of technical approaches behind the application of the joint transform correlator (JTC) in face recognition and object detection.

1.3.2.2 Face Recognition

The joint transform correlator (JTC) is used to recognize an unknown face object based on a database of desired impulses. The database is composed of some images of faces (impulses). For face recognition, an unknown face object is picked. The cross-correlated field between the unknown face object and each impulse is obtained. Then, the unknown face object is recognized as an impulse, which has the biggest cross-correlation with the unknown face object among other impulses.

Unfortunately, when the unknown face object does not have a similar impulse response, getting the biggest cross-correlation does not *always* mean that the unknown face object is recognized as an impulse, which has the biggest cross-correlation with the unknown face object. Therefore, a certain threshold must be used to increase the accuracy of recognition. For setting up a recognition threshold, the mean and standard deviation (the average distance from the mean to a point) are established for each impulse. Then, recognition can be updated, when the biggest cross-correlation with an impulse is less than or equal to the mean plus the standard deviation for the impulse and bigger than or equal to the mean minus the standard deviation for the impulse then the unknown face object is recognized as that impulse response; otherwise, it is an unknown face object.

Recognition performance can be analyzed by calculating an error rate of recognition. The error rate computes the percentage of error in recognition.

1.3.2.3 Object Detection

The joint transform correlator (JTC) is used to detect whether or not an unknown object contains a face based on a determined threshold for detection. Hence, in object detection, only a detection threshold is needed.

To obtain the detection threshold, the mean and the standard deviation are established for some objects that contain faces. For doing detection, an unknown object is taken. The unknown object is cross-correlated with any impulse response. Then, detection can be performed as if the resulted cross-correlation is less than or equal to the computed mean plus standard deviation and bigger than or equal to the com-

puted mean minus standard deviation then the unknown object is detected as a face object; otherwise, it is not a face object.

Detection performance can be analyzed by calculating an error rate of detection. The error rate computes the percentage of error in detection.

1.4 Summary of Key Results

The IPCA algorithm, in general, behaves better than the PCA algorithm in the most of the applications. It is better than the PCA algorithm in image compression because it obtains higher compression, more accurate reconstruction, and faster processing speed with acceptable errors; in addition, it is better than the PCA algorithm in real-time image detection due to the fact that it achieves the smallest error rate as well as remarkable speed. On the other hand, the PCA algorithm performs better than the IPCA algorithm in face recognition because it offers an acceptable error rate, easy calculation, and a reasonable speed. Finally, in detection and recognition, the performance of the digital model is better than the performance of the optical model.

1.5 Organization

The remainder of this thesis is organized as follows:

- Chapter 1: provides a general overview of this thesis.
- Chapter 2: covers theoretical backgrounds behind the PCA and IPCA algorithms, their applications, and their performance in the applications. A comparison between the PCA and IPCA algorithms is also provided in this chapter. Finally, it shows a theoretical background for designing an optical model for object detection and face recognition; and theories behind the joint transform correlator (JTC), its applications, and its performance in the applications.
- Chapter 3: presents the simulations of the PCA and IPCA algorithms by using the MATLAB © software and comparison between the simulations. It also presents the simulations of the PCA and IPCA applications by using the MATLAB © software. Lastly, this chapter provides the simulations of the joint transform correlator (JTC) and its applications by using the MATLAB © software.
- Chapter 4: covers the performance results of the PCA and IPCA algorithms in their applications. Also, this chapter provides the results of JTC performance in its applications.
- Chapter 5: presents a conclusion of this thesis.

Chapter 2

Theoretical Background

2.1 Preview

Digital and optical image processing are areas used experimentally to establish solutions to given problems. In this chapter, theoretical backgrounds for a couple of digital and optical processing techniques are demonstrated.

2.2 Digital Processing

2.2.1 Introduction

Principal components analysis (PCA) and improved principal components analysis (IPCA) are statistical tools frequently used for analyzing data. Their main applications are pattern recognition such as face detection and recognition, and data compression such as image compression. It is found that IPCA acts better than PCA in the most of applications. The analysis of each one is covered in this section.

2.2.2 A Principal Components Analysis (PCA) Algorithm

2.2.2.1 Introduction

PCA is a technique used for identifying patterns in data and representing the data in such a way as to highlight their similarities and differences. The identification of patterns in data of high dimensions (more than three dimensions) is too difficult because the graphical representation of data is impossible. Therefore, PCA is a powerful method for analyzing data.

This subsection covers the steps that are needed for performing PCA on a set of data and reconstructing the data set along with examples; as well as the steps that

are needed for performing PCA on images and reconstructing the images back. How and why the technique works are explained as well as what is happening at each step is demonstrated.

2.2.2.2 Analysis of the PCA Algorithm

The PCA algorithm is built up based on the following steps:

Step 1: getting some data.

Step 2: computing the mean vector $\boldsymbol{m}_D$ of the data set as in Equation 2.1. Where $\boldsymbol{D}_k$ is a column vector contains one data item such that $\boldsymbol{D}_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}$; and n is the total number of the data items.

$$\boldsymbol{m}_D = \frac{1}{n} \sum_{k=1}^{n} \boldsymbol{D}_k \tag{2.1}$$

Step 3: subtracting the mean from each of the data dimensions as in Equation 2.2. This produces a data set whose mean is zero which means the data set is centered. This step is really an important step for decreasing the error rate of face recognition.

$$R = [\boldsymbol{D}_1 - \boldsymbol{m}_D, \ldots, \boldsymbol{D}_n - \boldsymbol{m}_D] \tag{2.2}$$

Step 4: calculating a covariance matrix as in Equation 2.3. The covariance matrix is real and symmetric. $\frac{1}{n-1}$ can be removed or left because it is just a normalization factor which affects all values by the same amount. The division on $n-1$ and not n because the data set is a sample of the population. It is found that gives an answer is very close to the answer that will result if the entire population is used. If the covariance matrix is calculated for the entire population then the division must be on n.

$$C = \frac{1}{n-1} \sum_{k=1}^{n} (\boldsymbol{D}_k - \boldsymbol{m}_D) (\boldsymbol{D}_k - \boldsymbol{m}_D)^T = \frac{1}{n-1} \times RR^T \tag{2.3}$$

Step 5: obtaining the eigenvectors and eigenvalues of the covariance matrix C as in Equation 2.4 and Equation 2.5 respectively. Where the columns of the matrix A are the eigenvectors of C; the diagonal of the matrix Λ contains the eigenvalues of C; and d is the number of the dimensions of the data set.

$$A = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_d] \tag{2.4}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{bmatrix}, \qquad \text{where } \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \tag{2.5}$$

Since the covariance matrix C is a real and symmetric (Reference [5]; Pages 207 and 208; Theorems 5.9, 5.10, 5.11 and 5.12) $d \times d$ matrix then its eigenvectors form an orthonormal basis. Therefore, the matrix A is an orthonormal matrix.

Step 6: choosing principal components and forming a feature vector matrix. Eigenvectors associated with the biggest eigenvalues of the covariance matrix C follow the most significant patterns of the data. Those eigenvectors are called the principle components of the data set. Therefore, the eigenvalues of the covariance matrix C work as measures of how much information is contained in the principal components.

The feature vector matrix A_q represented in Equation 2.6 is an $d \times q\,(q < d)$ matrix that contains only q eigenvectors (principal components) from the matrix of eigenvectors A.

$$A_q = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_q] \tag{2.6}$$

In order to select the principal components that form the feature vector matrix A_q, the variance contribution rate (VCR) and the total variance contribution rate (TVC) (they are proposed in the IEEE paper presented in Reference [2]) are computed as in Equation 2.7 and Equation 2.8 respectively. When the TVC is significantly high then q eigenvectors associated with the biggest q eigenvalues can be selected.

$$VCR_k\,(\%) = \frac{\lambda_k}{\sum_{k=1}^{d} \lambda_k} \times 100, \qquad k = 1, \ldots, d \tag{2.7}$$

$$TVC\,(\%) = \frac{\sum_{k=1}^{q} \lambda_k}{\sum_{k=1}^{d} \lambda_k} \times 100, \qquad q = 1, \ldots, d \tag{2.8}$$

Step 7: performing the principal components transform (also called the Hotelling or Karhunen-Loéve transform).
Equation 2.9 is used for projecting the data on the eigenspace. The columns of the matrix Y represent the coordinates of the projected data in the eigenspace.

$$Y = A_q^T R \tag{2.9}$$

The mean of the matrix Y is $\boldsymbol{m}_Y = E\left[A_q^T R\right] = A_q^T \underbrace{E\,[R]}_{\mathbf{0}} = \mathbf{0}$. This has important meaning in face recognition. In fact, Y gives the original centered data solely in terms of the selected principal components instead of the original axes. It is possible to express data in terms of any two perpendicular axes as shown in Reference [6], Page 167, Theorem 5.7.

Finally, by projecting the centered data on the eigenspace, the PCA algorithm is completely done.

2.2.2.3 An Example of the PCA Algorithm

The example moves simultaneously with the PCA steps illustrated in Sub-subsection 2.2.2.2 until a data set is transformed as follows:

Step 1: the two-dimensional data set D shown in Equation 2.10 is obtained for performing the PCA algorithm. The plot of the data is shown in Figure 2.1.

$$D = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 2 & 1 & 5 & 2 & 6 & 5 & 10 & 7 & 11 & 8 \end{bmatrix}. \tag{2.10}$$

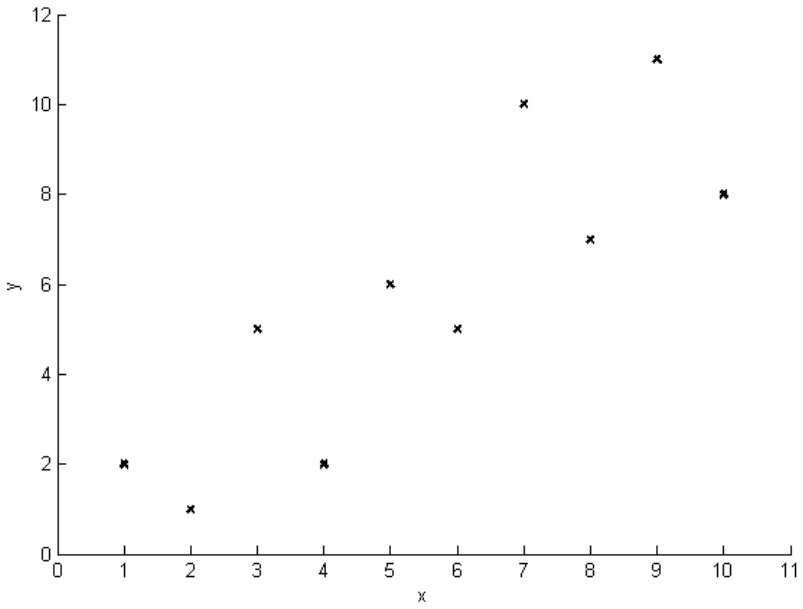

Figure 2.1: The plot of the data set D.

Step 2: the mean vector $\boldsymbol{m}_D$ of the data set is computed as in Equation 2.11.

$$\boldsymbol{m}_D = \frac{1}{10}\sum_{k=1}^{10} \boldsymbol{D}_k = \begin{bmatrix} 5.50 \\ 5.70 \end{bmatrix}. \tag{2.11}$$

Step 3: the mean is subtracted from each of the data dimensions then the centered data set R is obtained as in Equation 2.12. The plot of the centered data is shown in Figure 2.2.

$$R = [\boldsymbol{D}_1 - \boldsymbol{m}_D, \ldots, \boldsymbol{D}_{10} - \boldsymbol{m}_D]$$

$$= \begin{bmatrix} -4.50 & -3.50 & -2.50 & -1.50 & -0.50 & 0.50 & 1.50 & 2.50 & 3.50 & 4.50 \\ -3.70 & -4.70 & -0.70 & -3.70 & 0.30 & -0.70 & 4.30 & 1.30 & 5.30 & 2.30 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}. \tag{2.12}$$

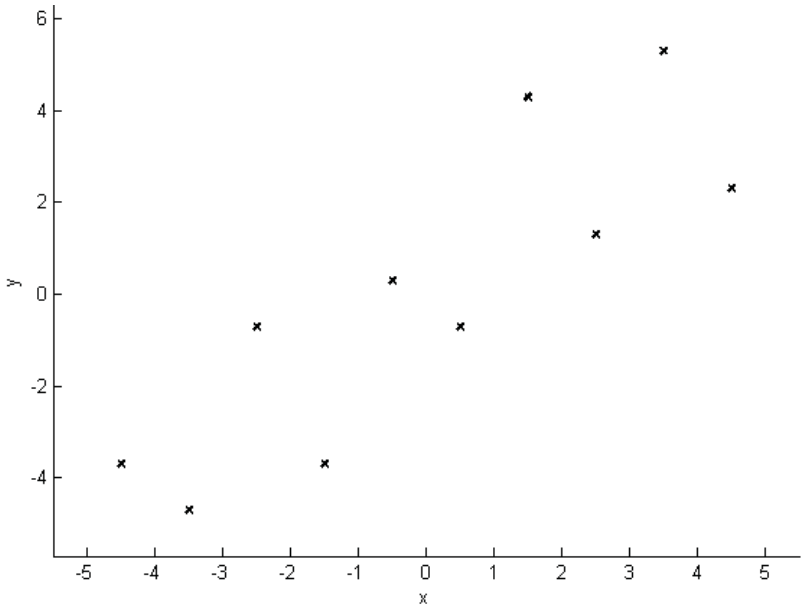

Figure 2.2: The plot of the centered data set R.

Step 4: the covariance matrix C is computed as in Equation 2.13. Since the non-diagonal elements of the covariance matrix are positive then both the x and y variables are expected to increase together.

$$\begin{aligned} C &= \frac{1}{10-1}\sum_{k=1}^{10}\left(\boldsymbol{D}_k - \boldsymbol{m}_D\right)\left(\boldsymbol{D}_k - \boldsymbol{m}_D\right)^T \\ &= \frac{1}{10-1} \times RR^T \\ &= \begin{bmatrix} 9.1667 & 8.7222 \\ 8.7222 & 11.5667 \end{bmatrix}. \end{aligned} \tag{2.13}$$

Step 5: the eigenvectors and eigenvalues of the covariance matrix C are computed

respectively as in Equation 2.14 and Equation 2.15.

$$A = [\boldsymbol{v}_1, \boldsymbol{v}_2]$$

$$= \begin{bmatrix} 0.6572 & -0.7538 \\ 0.7538 & 0.6572 \end{bmatrix}. \tag{2.14}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \qquad \text{where } \lambda_1 \geq \lambda_2$$

$$= \begin{bmatrix} 19.1711 & 0 \\ 0 & 1.5623 \end{bmatrix}. \tag{2.15}$$

The centered data as well as the orthonormal eigenvectors are plotted together in Figure 2.3. Figure 2.3 shows how the data have totally a noticed pattern; and as anticipated from the covariance matrix, the two variables are increasing together. The eigenvectors are plotted as diagonal dotted lines. As expected, they are perpendicular to each other; more importantly they highlight patterns in the data where the highly correlated eigenvector passes through the middle of the points. It divides the points to two sets, like drawing a line of the best fit; and it describes the most significant relationship between the data dimensions. The other eigenvector follows little patterns of the data.

Step 6: for choosing principal components and forming the feature vector matrix A_q, the variance contribution rate (VCR) and the total variance contribution rate (TVC) are calculated in Table 2.1. Based on Table 2.1, the feature vector matrix in Equation 2.16 only contains the eigenvector which is associated with the biggest eigenvalue (the highly correlated eigenvector).

Table 2.1: The calculations of the VCR and TVC.

$\boldsymbol{k}$	$\boldsymbol{\lambda_k}$	$\boldsymbol{VCR_k}$ (%)	$\boldsymbol{TVC}$ (%)
1	19.1711	92.4648	92.4649
2	1.5623	7.5352	100

$$A_q = \begin{bmatrix} 0.6576 \\ 0.7538 \end{bmatrix}. \tag{2.16}$$

Therefore, by computing the eigenvectors of the covariance matrix C and selecting the highest correlated ones then lines that describe the data are extracted. The rest

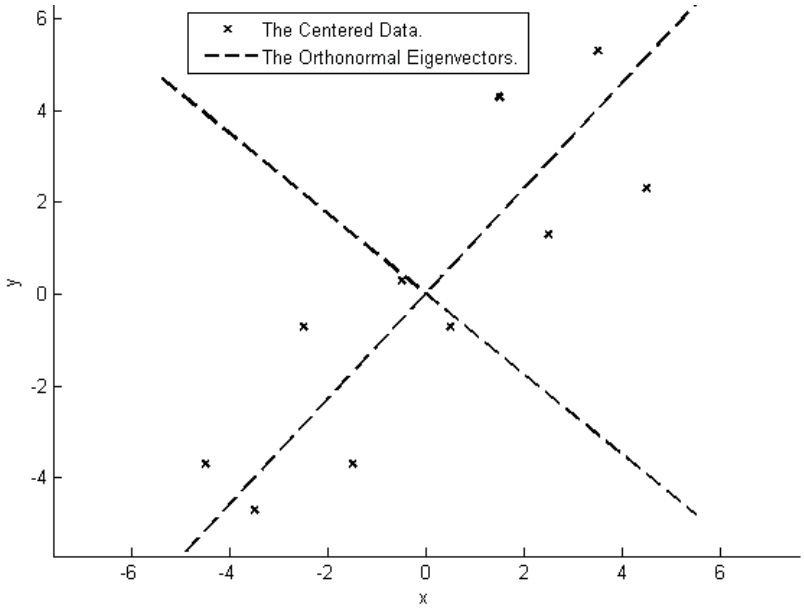

Figure 2.3: The plot of the centered data set R as well as the orthonormal eigenvectors.

of the steps involve transforming the data such that they are expressed in terms of the extracted lines.

Step 7: Equation 2.9 is used for performing the principal components transform; and the coordinates of the projected data in the eigenspace are shown in Equation 2.17.

$$Y^T = \begin{bmatrix} \text{Along } \boldsymbol{v}_1 \text{ axis} \\ -5.7461 \\ -5.8427 \\ -2.1705 \\ -3.7746 \\ -0.1025 \\ -0.1991 \\ 4.2269 \\ 2.6228 \\ 6.2950 \\ 4.6908 \end{bmatrix}. \tag{2.17}$$

In Equation 2.17, it can be seen that the dimensions of the projected data are reduced because the highest correlated eigenvector is only selected and the lowest one is neglected then some information is lost here. The projected centered data are plotted as in Figure 2.4. As shown in Figure 2.4, the projected centered data represent

a series of data items along the highest correlated eigenvector axis $\boldsymbol{v}_1$ without any information about the data along the axis of the lowest correlated eigenvector $\boldsymbol{v}_2$.

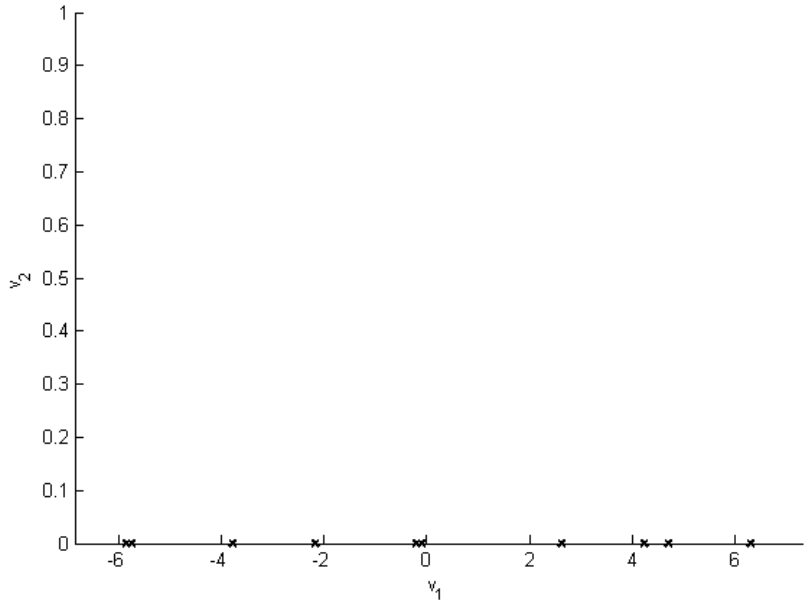

Figure 2.4: The projected centered data by using just the highest correlated eigenvector $\boldsymbol{v}_1$.

Therefore, in this step, the data are expressed in terms of the patterns between them where the patterns are the extracted lines that highly characterize the relationships between the data.

2.2.2.3.1 Taking All Eigenvectors as Principal Components for Doing PCA

We want to figure out what happens in the example presented in Sub-subsection 2.2.2.3 when all eigenvectors are selected as principal components?

The coordinates of the projected data in the eigenspace when all eigenvectors are taken to form the feature vector matrix A_q (i.e. $A = A_q$) are shown in Equation 2.18; and they are plotted in Figure 2.5. The projected data in Equation 2.17 is exactly equal to the first dimension of the projected data in Equation 2.18. The plot in Figure 2.5 and the plot of the original centered data in Figure 2.2 are typically the

same except that in Figure 2.5 the eigenvectors are the axes instead of x and y axes.

$$Y^T = \begin{bmatrix} \text{Along } \boldsymbol{v}_1 \text{ axis} & \text{Along } \boldsymbol{v}_2 \text{ axis} \\ -5.7461 & 0.9604 \\ -5.8427 & -0.4505 \\ -2.1705 & 1.4244 \\ -3.7746 & -1.3008 \\ -0.1025 & 0.5740 \\ -0.1991 & -0.8369 \\ 4.2269 & 1.6951 \\ 2.6228 & -1.0301 \\ 6.2950 & 0.8448 \\ 4.6908 & -1.8805 \end{bmatrix}. \tag{2.18}$$

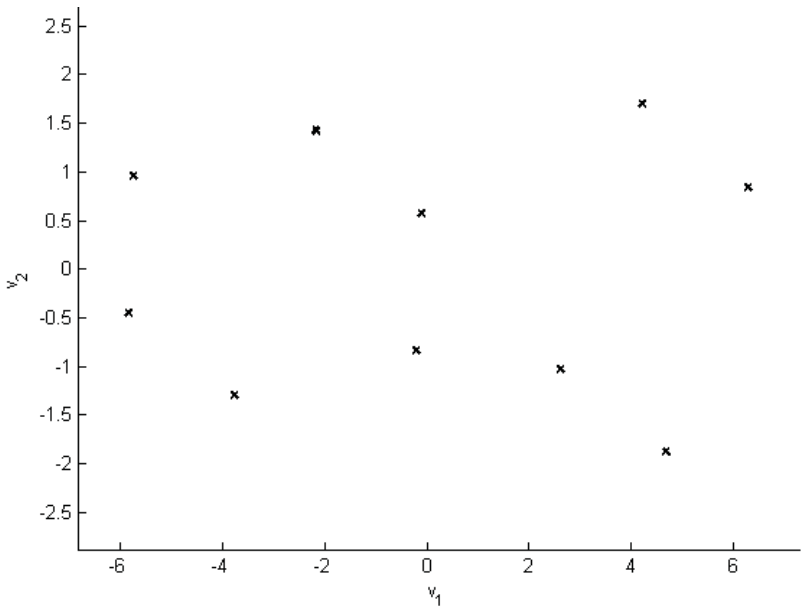

Figure 2.5: The projected centered data when all eigenvectors are used as principal components.

Therefore, there is no loss of information when all eigenvectors are selected as principal components for doing PCA.

2.2.2.4 Reconstruction of the Data Set D

For reconstructing the data set, Equation 2.9 is turned around to get the centered data set R as in Equation 2.19. Then the mean vector $\boldsymbol{m}_D$ is added again to obtain

the reconstructed data set $\widehat{D}$ as in Equation 2.20.

$$R = \left(A_q^T\right)^{-1} \times Y \tag{2.19}$$

$$\widehat{D} = \left(A_q^T\right)^{-1} \times Y + [\boldsymbol{m}_D, \ldots, \boldsymbol{m}_D]_{d \times n} \tag{2.20}$$

Since A_q^T in Equation 2.20 is not a square matrix and it has orthonormal (implies orthogonality) column vectors then the left inverse (Reference [6], Page 21, Definition 1.11) can be used to obtain its inverse. Its inverse is found to be $\left(A_q^T\right)^{-1} = \left(A_q^T\right)^T = A_q$. Then Equation 2.20 can be simplified as in Equation 2.21.

$$\widehat{D} = A_q \times Y + [\boldsymbol{m}_D, \ldots, \boldsymbol{m}_D]_{d \times n} \tag{2.21}$$

2.2.2.5 Reconstruction of the Data Set D of the Example in Sub-subsection 2.2.2.3

Equation 2.21 is used to reconstruct the data set D of the example in Sub-subsection 2.2.2.3 as in Equation 2.22. The reconstructed data are plotted in Figure 2.6. As seen in Figure 2.6, some information is lost from the reconstructed data due to some of the eigenvectors are used as principal components in performing PCA transform.

$$\widehat{D} = \begin{bmatrix} \text{Along } \boldsymbol{x} \text{ axis} & \text{Along } \boldsymbol{y} \text{ axis} \\ 1.7239 & 1.3689 \\ 1.6605 & 1.2960 \\ 4.0736 & 4.0640 \\ 3.0195 & 2.8549 \\ 5.4327 & 5.6228 \\ 5.3692 & 5.5500 \\ 8.2777 & 8.8860 \\ 7.2236 & 7.6769 \\ 9.6368 & 10.4449 \\ 8.5826 & 9.2357 \end{bmatrix}. \tag{2.22}$$

2.2.2.5.1 Using All Eigenvectors for Reconstructing the Data Set D

When all eigenvectors are selected as principal components for performing PCA in the example in Sub-subsection 2.2.2.3 then the original data set D will be reconstructed perfectly (i.e. $\widehat{D} = D$) without loss of information.

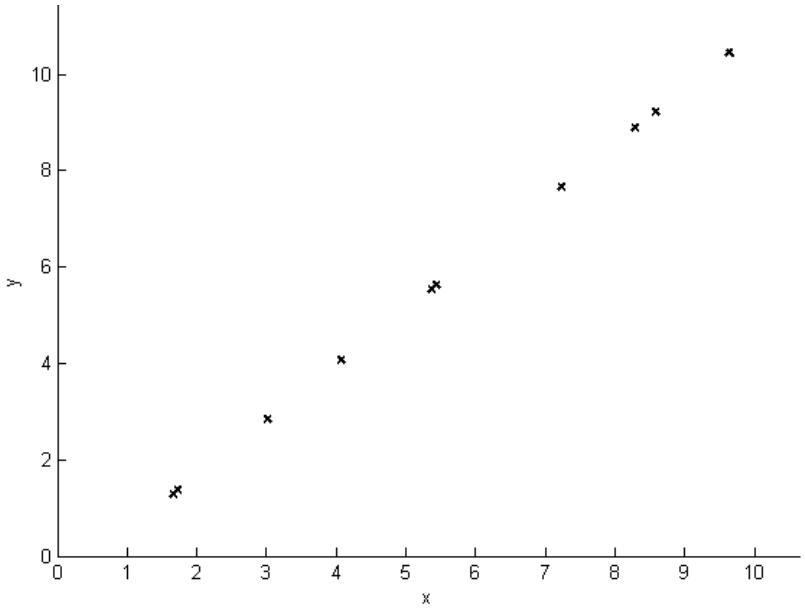

Figure 2.6: The reconstructed data by using just the highest correlated eigenvector $\boldsymbol{v}_1$.

2.2.2.6 Application of the PCA Algorithm to Images

The reason beyond performing the PCA algorithm on a simple database is to be able to provide plots of data for showing PCA behavior at each step. After demonstrating the PCA algorithm on a simple database, the idea can be generalized to see how the PCA algorithm works when the data set is composed of images. This idea is based on References [7] and [8]; and Reference [9], Section 12.5. The PCA algorithm can be applied to images as in the following steps:

Step 1: creating a database.
The database is composed of n, $N \times N$ images of faces (training faces) on black backgrounds such that I_k where $k = 1, \ldots, n$; and n is the total number of the training faces.

For decreasing the error rates of face detection and recognition, all face projections must be defined in the database; images must have the same size; images must have only faces and they are expanded to the boarders of the images; finally, images must have unified backgrounds in order to discriminate between the pixels occupying the backgrounds and the pixels occupying the faces.

Step 2: normalizing each training face I_k.
This normalization is for removing lighting effects on the training faces. It is very important to increase the accuracy of face recognition but it does not affect face

detection.

The normalization must be done just for the pixels occupying the faces to keep variations among the images just in the faces without the effects of the backgrounds. In order to block the pixels occupying the backgrounds for all training faces, a threshold must be picked to distinguish between the pixels occupying the faces and the pixels occupying the backgrounds. Note that, a number zero can not be taken to be the threshold although the training faces have black backgrounds because the MATLAB © software does not read a black color exactly zero then some error will occur.

According to the definition of an image histogram (Reference [10], Section 3.3), if a training face has a unified background then the biggest histogram of the intensity levels will be for the pixels occupying the backgrounds because pixels that have the same intensity levels are the pixels occupying the backgrounds of the training face. Therefore, the threshold can be selected based on the average histogram for all training faces.

Step 3: centering each training face I_k.
Since operations are performed on two-dimensional images then images must be centered before centering the whole database. This can be done by simply subtracting the mean of the pixels occupying the face for a training face from each pixel on the face as in Equation 2.23 where I_{C_k} is the k^{th} centered training face; and m_{f_k} is the mean of the pixels occupying the face for the k^{th} training face. By doing that the pixels on the face will have zero mean that means the face is centered.

$$I_{C_k} = I_k - m_{f_k}, \qquad \text{where } k = 1, \ldots, n \tag{2.23}$$

Step 4: representing each centered training face I_{C_k} as a column image vector $\mathbf{\Gamma}_k$.
Each $N \times N$ centered training face is represented as an N^2 column image vector by transposing the rows of pixels then stacking them one after another to form a column vector as in Equation 2.24.

$$\mathbf{\Gamma}_k = \begin{bmatrix} Row_1^T \\ Row_2^T \\ \vdots \\ Row_N^T \end{bmatrix} \tag{2.24}$$

Step 5: calculating the average training face vector $\mathbf{\Psi}$ as in Equation 2.25.

$$\mathbf{\Psi} = \frac{1}{n} \sum_{k=1}^{n} \mathbf{\Gamma}_k \tag{2.25}$$

Step 6: centering the set of the training faces.
The set of the training faces is centered by simply subtracting the mean training face vector $\mathbf{\Psi}$ from each centered training face vector $\mathbf{\Gamma}_k$ as in Equation 2.26 where $\mathbf{\Phi}_k$ is the k^{th} centered (with respect to the set of the training faces) training face. By doing

that the set of the training faces will have zero mean which means the database is centered.

$$\mathbf{\Phi}_k = \mathbf{\Gamma}_k - \mathbf{\Psi}, \qquad \text{where } k = 1, \ldots, n \tag{2.26}$$

Step 7: calculating a covariance matrix for all training faces as in Equation 2.27 where $R = [\mathbf{\Phi}_1, \mathbf{\Phi}_2, \ldots, \mathbf{\Phi}_n]$. The covariance matrix can be equal to RR^T due to $\frac{1}{n}$ can be removed or left because it is just a normalization factor which affects all values by the same amount.

$$C = \frac{1}{n} \sum_{k=1}^{n} \mathbf{\Phi}_k \mathbf{\Phi}_k^T = RR^T \tag{2.27}$$

Step 8: computing the eigenvectors and eigenvalues of the covariance matrix C.
The covariance matrix C is $N^2 \times N^2$ matrix where N^2 is the total number of pixels along one of its dimensions. The covariance matrix is usually too big which makes the calculation of the eigenvalues and eigenvectors is very difficult if not impossible. Hence, it is not practical to calculate the eigenvalues and eigenvectors for the such matrix but we will calculate them in this work for examining the performance of the PCA and IPCA algorithms.

The dimensions of the covariance matrix can be reduced to the number of the training faces. Let's suppose the $n \times n$ matrix $R^T R$; the eigenvectors and eigenvalues of this matrix are found as in Equation 2.28 where $\boldsymbol{u}_k$ is the k^{th} eigenvector of the matrix $R^T R$; and μ_k is the k^{th} eigenvalue.

$$R^T R \boldsymbol{u}_k = \mu_k \boldsymbol{u}_k, \qquad \text{where } k = 1, \ldots, n \tag{2.28}$$

The relationship between the eigenvector $\boldsymbol{v}_k$ of the matrix RR^T and the eigenvector $\boldsymbol{u}_k$ of the matrix $R^T R$ can be obtained as in Equation 2.29.

$$R^T R \boldsymbol{u}_k = \mu_k \boldsymbol{u}_k$$

$$RR^T R \boldsymbol{u}_k = \mu_k R \boldsymbol{u}_k$$

$$CR \boldsymbol{u}_k = \mu_k R \boldsymbol{u}_k$$

$$C \boldsymbol{v}_k = \mu_k \boldsymbol{v}_k, \qquad \text{where } \boldsymbol{v}_k = R \boldsymbol{u}_k \tag{2.29}$$

Equation 2.29 implies a couple of important notes are RR^T can have up to N^2 eigenvalues and eigenvectors; $R^T R$ can have up to n eigenvalues and eigenvectors; and n eigenvectors of the matrix RR^T associated with the biggest eigenvalues are exactly identical to the eigenvectors of the matrix $R^T R$ and they are related as, $\boldsymbol{v}_k = R \boldsymbol{u}_k$.

The generated eigenvectors by using the reduced covariance matrix must be normalized. The normalization can be performed by dividing the vector $\boldsymbol{v}_k$ by its length such that $\frac{\boldsymbol{v}_k}{\|\boldsymbol{v}_k\|}$ then the length of the normalized eigenvector $\boldsymbol{v}_k$ will be equal to one;

i.e., $\|\boldsymbol{v}_k\| = 1$. From now on, the reduced covariance matrix is going to be used for calculating desired eigenvectors and eigenvalues.

Step 9: selecting principal components and forming a feature vector matrix.
Here, principal components are called eigenfaces. They constitute the calculated eigenvectors associated with the biggest eigenvalues.

The feature vector matrix A_q represented in Equation 2.30 is an $N^2 \times q$ matrix that contains only q eigenvectors (principal components) such that $q << N^2$. Since the number of calculated eigenvectors by using the reduced covariance matrix $R^T R$ is equal to the total number of the training faces then $q \leq n$.

$$A_q = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_q], \qquad \text{where } q \leq n \tag{2.30}$$

In order to select the eigenfaces that form the feature vector matrix A_q, the variance contribution rate (VCR) and the total variance contribution rate (TVC) (they are proposed in the IEEE paper presented in Reference [2]) are computed as in Equation 2.31 and Equation 2.32 respectively. When the TVC is significantly high then q eigenvectors associated with the biggest q eigenvalues can be selected.

$$VCR_k\,(\%) = \frac{\lambda_k}{\sum_{k=1}^{n} \lambda_k} \times 100, \qquad k = 1, \ldots, n \tag{2.31}$$

$$TVC\,(\%) = \frac{\sum_{k=1}^{q} \lambda_k}{\sum_{k=1}^{n} \lambda_k} \times 100, \qquad q = 1, \ldots, n \tag{2.32}$$

Step 10: performing the principal components transform (also called the Hotelling or Karhunen-Loéve transform).
Equation 2.33 is used for projecting the training faces on the eigenspace. The columns of the matrix Y represent the coordinates of the projected training faces in the eigenspace; and $\boldsymbol{\Omega}^n$ contains the coordinates of the n^{th} projected training face.

$$Y = A_q^T R = \left[\boldsymbol{\Omega}^1 \ldots \boldsymbol{\Omega}^n\right], \qquad \text{where } \boldsymbol{\Omega}^n = \begin{bmatrix} w_1^n \\ \vdots \\ w_q^n \end{bmatrix} \tag{2.33}$$

Finally, by projecting the training faces on the eigenspace, the PCA algorithm is completely done.

2.2.2.7 Reconstruction of the Original Images

For reconstructing the training faces, Equation 2.33 is turned around to get the centered training faces matrix R as in Equation 2.34. The average training face vector $\boldsymbol{\Psi}$ is added again to obtain the reconstructed centered training faces vectors as in Equation 2.35 where $\widehat{\boldsymbol{\Gamma}}_n$ is the n^{th} reconstructed centered training face vector.

Also, the means of the pixels occupying the faces are added to get the reconstructed training faces vectors as in Equation 2.36 where $\widehat{\boldsymbol{I}}_n$ is the n^{th} reconstructed training face vector. Finally, $\widehat{\boldsymbol{I}}_n$ can be represented in the same manner as in Step 4, Subsubsection 2.2.2.6; to obtain the reconstructed training face $\widehat{I}_n$.

$$R = A_q Y \tag{2.34}$$

$$\left[\widehat{\boldsymbol{\Gamma}}_1 \dots \widehat{\boldsymbol{\Gamma}}_n\right] = A_q Y + \left[\boldsymbol{\Psi} \dots \boldsymbol{\Psi}\right]_{N^2 \times n} \tag{2.35}$$

$$\left[\widehat{\boldsymbol{I}}_1 \dots \widehat{\boldsymbol{I}}_n\right] = A_q Y + \left[\boldsymbol{\Psi} \dots \boldsymbol{\Psi}\right]_{N^2 \times n} + \left[m_{f_1} \dots m_{f_n}\right] \tag{2.36}$$

2.2.3 An Improved Principal Components Analysis (IPCA) Algorithm

2.2.3.1 Introduction

IPCA is another statistical tool for identifying patterns in data. It is typically like PCA except in the way of selecting eigenvectors that form the feature vector matrix A_q. It affords a new accurate method to measure the information content of principal components based on the information theory for improving PCA. IPCA acts better than PCA in the most of applications.

2.2.3.2 Analysis of the IPCA Algorithm

In order to estimate the degree of information content of eigenvectors, the concepts of Shannon information theory are fully used then two new concepts called the possibility information function (PIF) and the possibility information entropy (PIE) are obtained.

Eigenvalues can be transformed as in Equation 2.37 where d is the number of the dimensions of the data set. In Equation 2.37, it can be seen that $0 \leq \rho_k \leq 1$, where $k = 1, \dots, d$. Therefore, ρ_k has the numerical properties of probability. Being similar with the definition of entropy, the PIF and PIE can be defined respectively as in Equation 2.38 and Equation 2.39. In Equation 2.39, $H(T)$ reflects the unevenness of ρ_k. According to the PIF and PIE, it can be obtained that firstly $I(\lambda_k)$ denotes the information content included by λ_k where the bigger λ_k is associated with the bigger $I(\lambda_k)$; Secondly, when all ρ_k are equal (i.e. uniformly distributed) then the PIE reaches its maximum.

$$\rho_k = 1 - \frac{\lambda_k}{\sum_{k=1}^{d} \lambda_k}, \qquad k = 1, \dots, d \tag{2.37}$$

$$I(\lambda_k) = -\log_2 \rho_k, \qquad k = 1, \dots, d \tag{2.38}$$

$$H(T) = H(\rho_1, \dots, \rho_d) = -\sum_{k=1}^{d} \rho_k \log_2 \rho_k \tag{2.39}$$

For measuring the degree of information content of eigenvectors, two new concepts are used. The first one is the information rate (IR) shown in Equation 2.40; and the second one is the accumulated information rate (AIR) shown in Equation 2.41. When the AIR is significantly high then q eigenvectors associated with the biggest q eigenvalues can be selected.

$$IR_k\,(\%) = \frac{I\,(\lambda_k)}{\sum_{k=1}^{d} I\,(\lambda_k)} \times 100, \qquad k = 1, \ldots, d \tag{2.40}$$

$$AIR\,(\%) = \frac{\sum_{k=1}^{q} I\,(\lambda_k)}{\sum_{k=1}^{d} I\,(\lambda_k)} \times 100, \qquad q = 1, \ldots, d \tag{2.41}$$

2.2.3.3 Applying IPCA on the Example in Sub-subsection 2.2.2.3

The IR and AIR are computed for the eigenvectors of the covariance matrix C in the example in Sub-subsection 2.2.2.3; and the results are shown in Table 2.2. Based on Table 2.2, the feature vector matrix A_q in Equation 2.42 only contains the eigenvector which is associated with the biggest eigenvalue (the highly correlated eigenvector).

Table 2.2: The calculations of the IR and AIR.

$\boldsymbol{k}$	$\boldsymbol{\lambda_k}$	$\boldsymbol{\rho_k}$	$\boldsymbol{I\,(\lambda_k)}$	$\boldsymbol{IR_k\,(\%)}$	$\boldsymbol{AIR\,(\%)}$
1	19.1711	0.0754	3.7293	97.0616	97.0616
2	1.5623	0.9247	0.1129	2.9384	100

$$A_q = \begin{bmatrix} 0.6576 \\ 0.7538 \end{bmatrix}. \tag{2.42}$$

2.2.4 Comparison of the PCA and IPCA Algorithms

Comparison between the PCA and IPCA algorithms is based on the values of the TVC and AIR that determine the selected eigenvectors for the feature vector matrix A_q in Sub-subsection 2.2.2.3, Step 6; and in Sub-subsection 2.2.3.3.

The computed TVC and AIR are respectively equal to 92.4649% and 97.0616%. The AIR is big enough but the TVC is slightly small. Then the AIR is providing us with more confidence to pick the eigenvector which is associated with the biggest eigenvalue but the TVC is not. Therefore, the AIR tells us more about information contained in the eigenvectors of the covariance matrix C.

2.2.5 Applications of the PCA and IPCA Algorithms

2.2.5.1 Introduction

PCA and IPCA are applied in many fields. They have acceptable performance in many of them. In this subsection, the applications of the PCA and IPCA algorithms in image compression, face recognition and image detection are covered.

2.2.5.2 Image Compression

When some of the eigenvectors that are calculated from the covariance matrix C for all training faces are selected to form the feature vector matrix A_q then the dimensions of the reconstructed data set will be reduced. This implies that the PCA and IPCA algorithms work as compression. The algorithms are said to be lossy because a decompressed image is not exactly the same as the original one, but is generally worse.

2.2.5.3 Face Recognition

The PCA and IPCA algorithms are used to recognize an unknown face image based on the database which contains the training faces. For doing face recognition, an unknown face image is taken. It must have the same properties of the training faces. Hence, it must have the same size as the training faces; it has the same background; it has a projection as one of the training faces; finally, it has only face and it is expanded to the boarders of the image.

A couple of the operations explained in Sub-subsection 2.2.2.6 are applied to the unknown face image. The pixels occupying the face of the unknown face image are normalized and centered as in Step 2 and Step 3 respectively. The centered unknown face image is represented as a column image vector as in Step 4. The centered unknown image vector is centered in the set of the training faces by simply subtracting the mean training face vector $\mathbf{\Psi}$ from it as in Step 6. Then the unknown face image is projected on the eigenspace by using the PCA or IPCA feature vector matrix as in Step 10. The coordinates of the projected unknown face image in the eigenspace are shown in Equation 2.43.

$$\mathbf{\Omega}^{Unknown} = \begin{bmatrix} w_1^{Unknown} \\ \vdots \\ w_q^{Unknown} \end{bmatrix} \tag{2.43}$$

The Euclidean distance between the coordinates of the projected unknown face image and the coordinates of each training face is computed as in Equation 2.44. d_k is the distance between the coordinates of the projected unknown face image $\mathbf{\Omega}^{Unknown}$ and the coordinates of the k^{th} projected training face $\mathbf{\Omega}^k$; and n is the total number of the training faces. Note that, the distances between the unknown face image and

the training faces are measured along the new axes derived from the PCA algorithm but not along the original axes. It turns out that these axes work much better for recognizing faces because PCA has given us the original training faces in terms of the differences and similarities between them.

$$d_k = \left\| \mathbf{\Omega}^{Unknown} - \mathbf{\Omega}^k \right\|, \qquad \text{where } k = 1, \ldots, n \tag{2.44}$$

Then recognition can be performed by using Condition 2.1 where md_k is the minimum distance between the coordinates of the projected unknown face image $\mathbf{\Omega}^{Unknown}$ and the coordinates of the k^{th} projected training face $\mathbf{\Omega}^k$.

Condition 2.1. *If,*

$$md_k = min\left([d_1, \ldots, d_n]\right), \qquad \textit{where } k \textit{ can be any number between 1 to } n$$

Then the unknown face image is recognized as the k^{th} training face; otherwise, it is an unknown face image.

At this point, it can be answered why the normalization of the training faces as well as the centering of the database and faces increase the accuracy of face recognition. That because when the database and faces are centered, the set of the projected vectors in the eigenspace will have zero mean; that means the vectors begin from the same origin; then the distance between a training face and its corresponding unknown face image will be very small compared with other training faces. Regarding the normalization of the training faces, when a training face and its corresponding unknown face image are normalized (i.e. they have the same length), the distance between them will decrease as shown in Figure 2.7. On the other hand, if they are unnormalized (i.e. they do not have the same length), the distance between them will increase as shown in Figure 2.8.

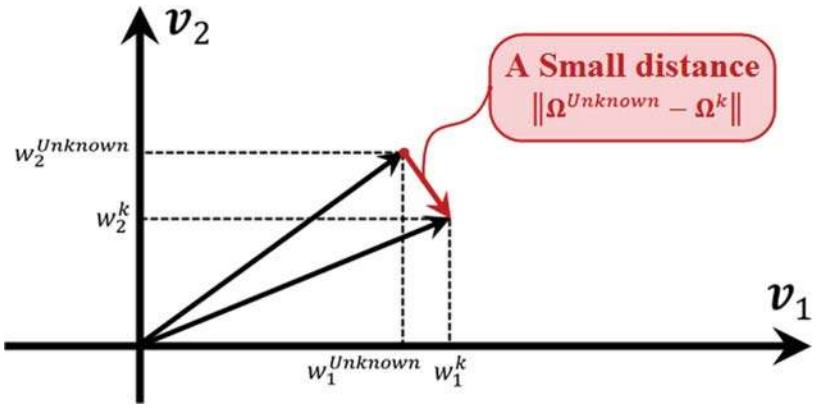

Figure 2.7: A normalized training face and its normalized corresponding unknown face image.

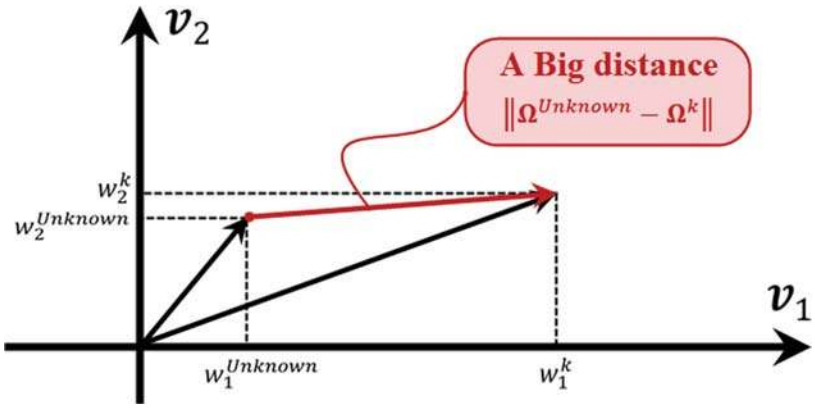

Figure 2.8: An unnormalized training face and its unnormalized corresponding unknown face image.

2.2.5.3.1 Setting up a Recognition Threshold

Unfortunately, when the unknown face image does not have a similar training face then getting the minimum distance md_k does not *always* mean that the unknown face image is recognized as the k^{th} training face. Therefore, a certain threshold must be used to increase the accuracy of recognition.

For setting up a recognition threshold, some different images are taken for each training face. These images are called tested images. They are known here just for picking the threshold. The distances between each training face and its *corresponding* tested images are obtained; and they are stacked in the row vectors $\boldsymbol{T}_1, \ldots, \boldsymbol{T}_k, \ldots, \boldsymbol{T}_n$, where $\boldsymbol{T}_k$ contains the smallest expected distances because the k^{th} training face and its *corresponding* tested images have the same person and the same face projection.

Thereafter, the mean m_k and standard deviation STD_k (the average distance from the mean to a point) are computed for each row vector $\boldsymbol{T}_k$ of the smallest distances. By calculating the means and the standard deviations, a certain threshold is established for each training face.

Then the recognition threshold can be applied to recognize the unknown face image by using Condition 2.2.

Condition 2.2. *If,*

$$m_k - STD_k \leq md_k \leq m_k + STD_k, \qquad \textit{where } k = 1, \ldots, n$$

Then the unknown face image is recognized as the k^{th} training face; otherwise, it is an unknown face image.

2.2.5.4 Image Detection

The PCA and IPCA algorithms are used to detect if an unknown image contains a face or not based on a determined threshold for detection. Hence, in image detection, only a detection threshold is needed.

To obtain the detection threshold, tested images are generated from the database of the training faces in the same manner as in Sub-sub-subsection 2.2.5.3.1. From Sub-subsection 2.2.2.6, the preprocessing operations illustrated in Step 2, Step 3, Step 4, Step 6 and Step 10 are respectively applied to the tested images. Then the projected centered (with respect to the set of the training faces) tested images are reconstructed again without adding the average training face neither the means of the pixels occupying the faces. Each reconstructed tested image is normalized and multiplied by the biggest intensity from its centered tested image; this is done in order to make each centered tested image and its reconstruction have approximately the same dynamic range.

The Euclidean distance between each centered tested image and its reconstruction is computed as in Equation 2.45 where a_k is the distance between the k^{th} centered tested image $\mathbf{\Phi}_k^{Tested\,Im}$ and its reconstruction $\widehat{\mathbf{\Phi}}_k^{Tested\,Im}$; as well as t is the total number of the tested images. Thereafter, the computed distances are placed in the row vector $\boldsymbol{S}$. The mean $m_{\boldsymbol{S}}$ and the standard deviation $STD_{\boldsymbol{S}}$ (the average distance from the mean to a point) are calculated for the vector $\boldsymbol{S}$. By calculating the mean and the standard deviation, the detection threshold is established.

$$a_k = \left\| \mathbf{\Phi}_k^{Tested\,Im} - \widehat{\mathbf{\Phi}}_k^{Tested\,Im} \right\|, \qquad \text{where } k = 1, \ldots, t \tag{2.45}$$

For applying the detection threshold, the unknown image $I^{Unknown}$ is picked for detection. The Euclidean distance $a^{Unknown}$ between the centered unknown image $\mathbf{\Phi}^{Unknown}$ and its reconstruction $\widehat{\mathbf{\Phi}}^{Unknown}$ is computed. Then the unknown image $I^{Unknown}$ can be detected by means of Condition 2.3.

Condition 2.3. *If,*

$$m_{\boldsymbol{S}} - STD_{\boldsymbol{S}} \leq a^{Unknown} \leq m_{\boldsymbol{S}} + STD_{\boldsymbol{S}}$$

Then the unknown image $I^{Unknown}$ is detected as a face image; otherwise, it is not a face image.

2.2.6 Performance Analysis of the PCA and IPCA Algorithms

2.2.6.1 Introduction

In studying performance, attention is paid to study how each application behaves when the eigenfaces that are generated by using the PCA algorithm are used; the

eigenfaces that are generated by using the IPCA algorithm are used; and when different eigenfaces are selected to form the feature vector matrix.

2.2.6.2 Analysis of Compression Performance

2.2.6.2.1 Introduction

Compression performance can be analyzed from three points of view are the speed of compression and reconstruction, the quality of a reconstructed image, and the size of compression. Each one is explained in details in this sub-subsection.

2.2.6.2.2 Speed of Compression and Reconstruction

The number of the eigenfaces that is used to compress and reconstruct the training faces mainly controls the processing speed of compression and reconstruction. When a small number of the eigenfaces is used to project and reconstruct the training faces then the processing speed will increase and vice versa.

2.2.6.2.3 Quality of a Reconstructed Image

For measuring the quality of a reconstructed image, the mean squared error (MSE) is computed as in Equation 2.46 to measure an error between the image I_k and its reconstruction $\widehat{I}_k$; where N is the number of rows and columns of the image I_k.

$$e_{MSE} = \frac{1}{N^2} \sum_{r=1}^{N} \sum_{c=1}^{N} \left[\widehat{I}_k (r, c) - I_k (r, c) \right]^2 \tag{2.46}$$

2.2.6.2.4 Size of Compression

2.2.6.2.4.1 Introduction

The size of compression can be measured in two ways are an information rate and the mean squared error (MSE) of compressed images.

2.2.6.2.4.2 Information Rate

An information rate measures how much information is after compression compared with information before compression; in other words, it measures the number of pixels after compression compared with before. This can be accomplished as in

Equation 2.47.

Before Compression : After Compression

$$\frac{\left(\underbrace{n}_{\substack{\text{The total}\\ \text{number of}\\ \text{the training}\\ \text{faces}}} \times \underbrace{N^2}_{\substack{\text{An image}\\ \text{size}}}\right)}{n \times N^2} \times 100 \;:\; \frac{\left(\underbrace{N^2 \times q}_{\substack{\text{The feature}\\ \text{vector matrix}\\ A_q}} + \underbrace{q \times n}_{\substack{\text{The coordinates}\\ \text{matrix } Y \text{ of}\\ \text{the projected}\\ \text{training faces}}} + \underbrace{N^2}_{\substack{\text{The average}\\ \text{training}\\ \text{face vector}\\ \boldsymbol{\Psi}}}\right)}{n \times N^2} \times 100 \tag{2.47}$$

In Equation 2.47 the rows and columns for the pixels occupying the faces as well as the means of the pixels on the faces are not considered in the overall size after compression because the operation of face centering is not really important in image compression; and it does not have any effect if it is done or not; but it has been done here because it is important for other applications. The normalization is done in Equation 2.47 to make the overall information before compression is equal to 100% all the time in order to make comparison easier.

2.2.6.2.4.3 Mean Squared Error (MSE) of Compressed Images

The mean squared error (MSE) between the exact and approximate reconstruction of the training face vector $\boldsymbol{I}_k$ is calculated as follows,

Equation 2.48 shows the exact reconstruction of the training face vector $\boldsymbol{I}_k$.

$$\boldsymbol{I}_k = \sum_{i=1}^{n} w_i^k \boldsymbol{v}_i + \boldsymbol{\Psi} \tag{2.48}$$

And, Equation 2.49 shows the approximate reconstruction of the training face vector $\boldsymbol{I}_k$.

$$\widehat{\boldsymbol{I}}_k = \sum_{i=1}^{q} w_i^k \boldsymbol{v}_i + \boldsymbol{\Psi} \tag{2.49}$$

The error between $\boldsymbol{I}_k$ and $\widehat{\boldsymbol{I}}_k$ can be computed as in Equation 2.50.

$$\begin{aligned}
\boldsymbol{e} &= \boldsymbol{I}_k - \widehat{\boldsymbol{I}}_k \\
&= \sum_{i=1}^{n} w_i^k \boldsymbol{v}_i + \boldsymbol{\Psi} - \sum_{i=1}^{q} w_i^k \boldsymbol{v}_i - \boldsymbol{\Psi} \\
&= \sum_{i=1}^{n} w_i^k \boldsymbol{v}_i - \sum_{i=1}^{q} w_i^k \boldsymbol{v}_i \\
&= \sum_{i=q+1}^{n} w_i^k \boldsymbol{v}_i
\end{aligned} \tag{2.50}$$

For computing the MSE of the linear estimate $\widehat{\boldsymbol{I}}_k$, Equation 2.51 can be used.

$$\begin{aligned}
E\left[\boldsymbol{e}^{\boldsymbol{T}}\boldsymbol{e}\right] &= E\left[\left(\sum_{i=q+1}^{n} w_i^k \boldsymbol{v}_i\right)^T \left(\sum_{m=q+1}^{n} w_m^k \boldsymbol{v}_m\right)\right] \\
&= E\left[\sum_{i=q+1}^{n} w_i^k \boldsymbol{v}_i^T \sum_{m=q+1}^{n} w_m^k \boldsymbol{v}_m\right] \\
&= E\left[\sum_{i=q+1}^{n} \sum_{m=q+1}^{n} w_i^k w_m^k \boldsymbol{v}_i^T \boldsymbol{v}_m\right] \\
&= \sum_{i=q+1}^{n} \sum_{m=q+1}^{n} E\left[w_i^k w_m^k\right] \boldsymbol{v}_i^T \boldsymbol{v}_m
\end{aligned} \tag{2.51}$$

To find $E\left[w_i^k w_m^k\right]$, a covariance matrix for the coordinates matrix Y of the projected

training faces must be computed as in Equation 2.52

$$\begin{aligned} C_Y &= YY^T \\ &= \left[A^T R\right]\left[A^T R\right]^T \\ &= \left[A^T R\right]\left[R^T A\right] \\ &= A^T \underbrace{RR^T}_{C} A \\ &= A^T C A \end{aligned} \tag{2.52}$$

From Reference [3], Page 169, Theorem A.1.30; since C is a real and symmetric matrix as well as A is an orthonormal matrix then C_Y can be written as in Equation 2.53 where I is the identity matrix.

$$\begin{aligned} C_Y &= A^T C A \\ &= \underbrace{A^T A}_{I} \Lambda \underbrace{A^T A}_{I} \\ &= I \Lambda I \\ &= \Lambda \\ &= \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & \lambda_n \end{bmatrix} \end{aligned} \tag{2.53}$$

Hence, from Equation 2.53, it can be concluded that,

$$E\left[w_i^k w_m^k\right] = \begin{cases} \lambda_i & \text{when } i = m \\ 0 & \text{when } i \neq m \end{cases}$$

And,

$$\boldsymbol{v}_i^T \boldsymbol{v}_m = \begin{cases} 1 & \text{when } i = m \\ 0 & \text{when } i \neq m \end{cases} \text{ Because } \boldsymbol{v}_i \text{ and } \boldsymbol{v}_m \text{ form an orthonormal basis.}$$

Therefore, when $i = m$, the MSE between the exact and approximate reconstruction of the training face vector $\boldsymbol{I}_k$ is obtained as in Equation 2.54.

$$E\left[\boldsymbol{e^T e}\right] = \sum_{i=q+1}^{n} \sum_{m=q+1}^{n} E\left[w_i^k w_m^k\right] \boldsymbol{v}_i^T \boldsymbol{v}_m$$

$$= \sum_{i=q+1}^{n} E\left[\left(w_i^k\right)^2\right] \cdot 1$$

$$= \sum_{i=q+1}^{n} \lambda_i \tag{2.54}$$

2.2.6.3 Analysis of Recognition Performance

2.2.6.3.1 Introduction

Recognition performance can be analyzed from two points of view are the speed of recognition and an error rate. Each one is explained in details in this sub-subsection.

2.2.6.3.2 Speed of Recognition

The number of the selected eigenfaces that is used to recognize the unknown face image mainly controls the recognition speed. When the number of the selected eigenfaces decreases, the processing speed increases and vice versa.

2.2.6.3.3 Error Rate

The error rate computes the percentage of error in recognition. It can be computed as in Equation 2.55 where L is the total number of the tested images; SR is the total number of successes in the recognition of the tested images; FR is the total number of failures in the recognition of the tested images; and ER is the error rate.

$$ER\,(\%) = \frac{FR}{L} \times 100 \tag{2.55}$$

2.2.6.4 Analysis of Detection Performance

2.2.6.4.1 Introduction

Detection performance can be analyzed from two points of view are the speed of detection and an error rate. Each one is explained in details in this sub-subsection.

2.2.6.4.2 Speed of Detection

The number of the selected eigenfaces that is used to detect the unknown image mainly controls the detection speed. When the number of the selected eigenfaces decreases, the processing speed increases and vice versa.

2.2.6.4.3 Error Rate

The error rate computes the percentage of error in detection. It can be computed as in Equation 2.56 where L is the total number of the tested images; SD is the total number of successes in the detection of the tested images; FD is the total number of failures in the detection of the tested images; and ER is the error rate.

$$ER\,(\%) = \frac{FD}{L} \times 100 \tag{2.56}$$

2.3 Analog Optical Information Processing

2.3.1 Introduction

Analog optical information processing is an important area which recalls the linearity concepts of imaging systems in order to synthesize an optical model that can perform one or multiple functions. The focus of this section is about providing a theoretical background for designing an optical model for object detection and face recognition. Concentration will be limited to coherent optical models for some reasons will be mentioned in the next subsection. This section is based on Reference [3], Chapter 8.

2.3.2 Coherent and Incoherent Optical Image Processing Systems

This subsection shows the difference between the usage of spatially incoherent light and the usage of spatially coherent light in optical information processing. Spatially incoherent light has some big advantages but on the other hand it has severe disadvantages. The advantages of spatially incoherent light are:

1. It is free from coherent artifacts such as dust specks on optical components and the speckle phenomenon.

2. Data can be introduced to a system by using incoherent light sources such as light-emitting diode (LED) arrays or cathode-ray tube (CRT) displays; but in coherent systems, complicated and costly spatial light modulators (SLMs) are used to introduce data.

3. In general, incoherent systems are easier than coherent systems in physical implementation.

On the other hand, spatially incoherent light has disadvantages that make us prefer to use spatially coherent light in our model. The disadvantages of spatially incoherent light are:

1. An incoherent optical system does not have a frequency plane but a coherent optical system has a plane at a distance f from a lens is called a frequency plane or a focal plane. The absence of this plane makes the manipulation of an input spectrum is very difficult rather than just the direct manipulation of a spectrum on a back focal plane.

2. Incoherent optical systems are linear in intensity. The manipulation of intensity in optical processing systems is very complex if it is not impossible because intensity is a positive and real physical quantity. For instance, there is no a normal optical method to subtract two intensity patterns; but coherent optical systems are linear in complex amplitude; consequently, if one wants to subtract two complex amplitude patterns then the patterns can be added together with a π radian phase shift between them.

3. The spectrum of an incoherent image that is generated from an incoherent optical system always has the biggest spectral component at the origin. This makes a produced incoherent image has low contrast. Therefore, incoherent optical systems need a huge use of electronics in order to enhance an output incoherent image and makes it comparable to an output coherent image.

Due to these serious disadvantages, spatially coherent light is going to be used in our model.

2.3.3 Coherent Optical Information Processing Systems

We now present the coherent optical information processing model used for object detection and face recognition. From the last subsection, it is known that coherent optical systems are linear in complex amplitude; therefore, filtering processes can be performed by direct manipulation of complex amplitude appearing in the back focal plane of a Fourier transforming lens. There are a large number of system architectures that can do frequency domain filtering but a pretty conceptually straightforward system shown in Figure 2.9 is implemented. This model for coherent optical information processing is called $4f$ model because a distance that separates the input plane P_1 form the output plane P_3 is composed of four separate distances of length f. The length of this model from the point source S until the output plane is $5f$.

The collimating lens L_1 is used to collimate light from the point source S. The input transparency is placed in the input plane P_1 against the collimating lens L_1.

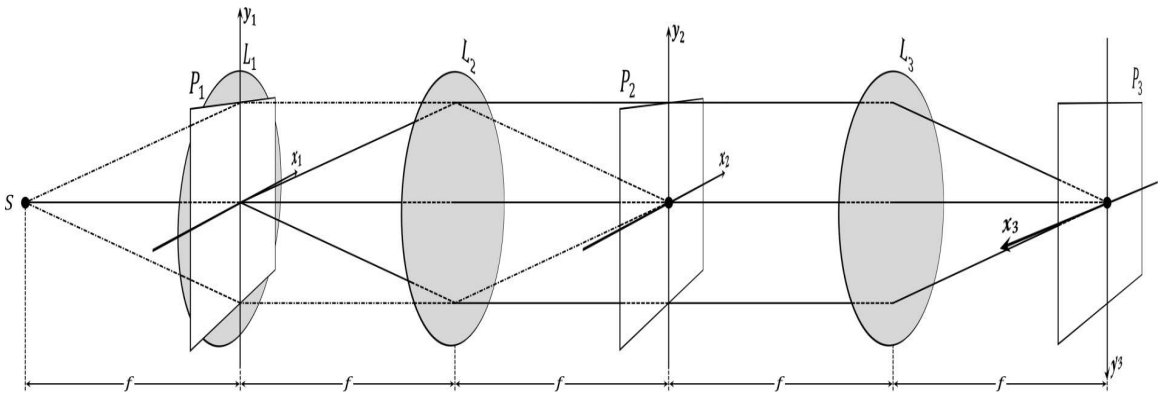

Figure 2.9: The architecture for coherent optical information processing.

The input plane is placed against the collimating lens in order to decrease the total length of the coherent model. The amplitude transmittance of the input transparency is $g(x_1, y_1)$. The input transparency is illuminated by a uniform normally incident plane (or spherical) wave of amplitude A. Then the complex amplitude distribution of the field just behind the input transparency (i.e. the field transmitted by the input transparency) is $U_1(x_1, y_1) = Ag(x_1, y_1)$. The Fourier transforming lens L_2 is Fourier transforming the illuminated input transparency in its back focal plane P_2. A transparency is inserted in the back focal plane to modulate the amplitude transmittance over that plane. Then the complex amplitude distribution $U_2(x_2, y_2)$ of the Fourier transformed field can be found as in Equation 2.57.

$$
\begin{aligned}
U_2(x_2, y_2) &= \frac{1}{j\lambda f} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U_1(x_1, y_1) \exp^{-j\frac{2\pi}{\lambda f}(x_1 x_2 + y_1 y_2)} dx_1 dy_1 \\
&= \frac{A}{j\lambda f} \underbrace{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x_1, y_1) \exp^{-j\frac{2\pi}{\lambda f}(x_1 x_2 + y_1 y_2)} dx_1 dy_1}_{f_X = \frac{x_2}{\lambda f} \;\&\; f_Y = \frac{y_2}{\lambda f}} \\
&= \frac{A}{j\lambda f} G\left(\frac{x_2}{\lambda f}, \frac{y_2}{\lambda f}\right) \\
&= k_1 G\left(\frac{x_2}{\lambda f}, \frac{y_2}{\lambda f}\right) \qquad (2.57)
\end{aligned}
$$

Where k_1 is a complex constant; λ is the light wavelength in meter (m); and $G\left(\frac{x_2}{\lambda f}, \frac{y_2}{\lambda f}\right)$ is the Fourier spectrum of the Fourier transformed field.

A desired filter can be synthesized and placed in the plane P_2 in order to manipulate $G\left(\frac{x_2}{\lambda f}, \frac{y_2}{\lambda f}\right)$. Let the transfer function of a synthesized filter be represented by H then the complex amplitude distribution in the back focal plane of the filter should be as in Equation 2.58 where k_2 is a complex constant.

$$U_f(x_2, y_2) = k_2 H\left(\frac{x_2}{\lambda f}, \frac{y_2}{\lambda f}\right) \tag{2.58}$$

The spectrum of the field just behind the transparency of the plane P_2 is $G\left(\frac{x_2}{\lambda f}, \frac{y_2}{\lambda f}\right) H\left(\frac{x_2}{\lambda f}, \frac{y_2}{\lambda f}\right)$. The Fourier transforming lens L_3 Fourier transforms the altered spectrum in its back focal plane P_3. There is no an optical component does inverse Fourier transform; therefore, another Fourier transforming lens is used to produce the final complex amplitude distribution on the output plane P_3. The usage of the two consecutive Fourier transforming lenses makes the coordinates of the output plane P_3 inverted (Reference [11], Page 25, "Application of the Fourier Transform"). The coordinates inversion problem can be overcome by inverting the coordinates of the output plane P_3.

This coherent model has a disadvantage that vignetting can happen through performing the first Fourier transform. In order to overcome this problem, the input plane $P1$ can be placed against the Fourier transforming lens L_2.

There are at least two methods for synthesizing the frequency plane filter for coherent optical systems. One of these methods is by using the joint transform correlator (JTC). This method is discussed in the next subsection.

2.3.4 The Joint Transform Correlator (JTC)

This correlator is used to synthesize the frequency plane filter in order to manipulate the spectrum on the back focal plane of the Fourier transforming lens L_2. This method was invented by Weaver and Goodman [4] and called as the joint transform correlator. The filter architecture is shown in Figure 2.10.

The filter is divided to two stages: recording the filter and getting the filter output. In the recording process, the lens L_1 collimates light from the point source S. Two input transparencies are placed in the input plane P_1. The first transparency is for the desired impulse response h and centered at the coordinate $\left(0, \frac{Y}{2}\right)$. The other transparency is for the data g to be filtered and centered at the coordinate $\left(0, -\frac{Y}{2}\right)$. Hence, their centers are separated by the distance Y. The input transparencies are illuminated by a uniform normally incident plane (or spherical) wave of amplitude A. Then the complex amplitude distribution of the field just behind the input transparencies (i.e. the field transmitted by the input transparencies) is obtained as in Equation 2.59.

$$U_1(x_1, y_1) = A\left[h\left(x_1, y_1 - \frac{Y}{2}\right) + g\left(x_1, y_1 + \frac{Y}{2}\right)\right] \tag{2.59}$$

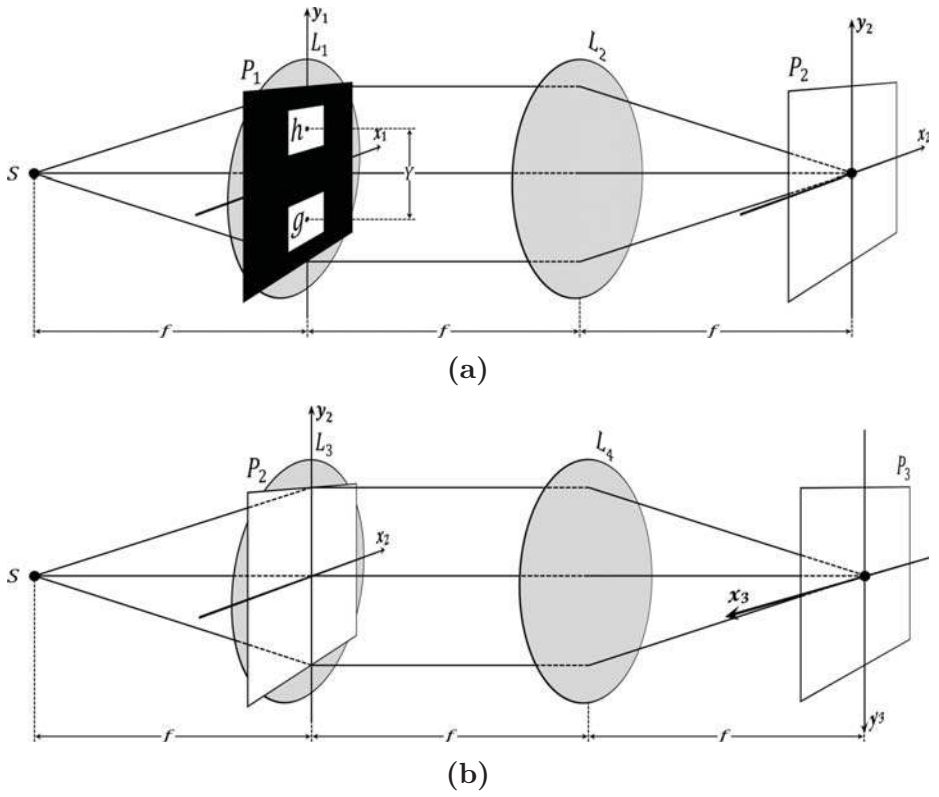

Figure 2.10: The joint transform correlator: **(a)** Recording the filter; and **(b)** getting the filter output.

The Fourier transforming lens L_2 is Fourier transforming the field transmitted by the input transparencies in its back focal plane P_2. The complex amplitude distribution $U_2(x_2, y_2)$ of the Fourier transformed field is obtained as in Equation 2.60. The derivation of $U_2(x_2, y_2)$ is shown in Appendix E.

$$U_2(x_2, y_2) = \frac{A}{j\lambda f} \exp^{-j\frac{\pi Y}{\lambda f} y_2} H\left(\frac{x_2}{\lambda f}, \frac{y_2}{\lambda f}\right) + \frac{A}{j\lambda f} \exp^{j\frac{\pi Y}{\lambda f} y_2} G\left(\frac{x_2}{\lambda f}, \frac{y_2}{\lambda f}\right) \qquad (2.60)$$

From linear algebra, if Z_1 and Z_2 are complex numbers; as well as Z_1^* and Z_2^* are their conjugates respectively then $|Z_1 + Z_2|^2 = |Z_1|^2 + |Z_2|^2 + (Z_1 Z_2^* + Z_2 Z_1^*)$. Using this operation on complex numbers and another operation is that the conjugate of the exponential function $\exp^{jx}$ is equal to $\exp^{-jx}$ (conversely, the conjugate of the exponential function $\exp^{-jx}$ is equal to $\exp^{jx}$) then the incident intensity on the back

focal plane of the lens L_2 is computed as in Equation 2.61.

$$\begin{aligned} I(x_2,y_2) =& \frac{A^2}{\lambda^2 f^2}\left[\left|H\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right)\right|^2+\left|G\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right)\right|^2+\right. \\ &+H\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right)G^*\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right)\exp^{-j\frac{2\pi Y}{\lambda f}y_2}+ \\ &\left.+H^*\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right)G\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right)\exp^{j\frac{2\pi Y}{\lambda f}y_2}\right] \end{aligned} \tag{2.61}$$

Note that, the recorded transparency in the plane P_2 is supposed to have an amplitude transmittance that is proportional to the intensity $I(x_2,y_2)$. To obtain the output of the filter, the recorded transparency is illuminated by a uniform normally incident plane (or spherical) wave of amplitude B. The complex amplitude distribution of the field just behind the recorded transparency (i.e. the field transmitted by the recorded transparency) is $U_r(x_2,y_2) = BI(x_2,y_2)$. The lens L_4 is Fourier transforming the transmitted field in its back focal plane P_3. The complex amplitude distribution of the Fourier transformed field in the output plane P_3 is computed in Equation 2.62.

$$\begin{aligned} U_3(x_3,y_3) &= \mathfrak{F}\{U_r(x_2,y_2)\} \\ &= B\,\mathfrak{F}\{I(x_2,y_2)\} \end{aligned} \tag{2.62}$$

From linear algebra, if Z_1 is a complex number and Z_1^* is its conjugate then $|Z_1|^2 = Z_1 Z_1^*$. Using this operation on complex numbers; and using another properties and theorems of Fourier transform are the convolution theorem (Reference [12], Page 37, Property 12), the complex conjugation property (Reference [12], Page 28, Property 3), and the property that the Fourier transform of the shifted impulse response $\delta(t-t_o)$ is equal to $\exp^{-j2\pi f t_o}$ where t_o is the amount of the shift; then the field in the back focal plane P_3 is found as in Equation 2.63.

$$\begin{aligned} U_3(x_3,y_3) =& B\frac{A^2}{\lambda^2 f^2}\left[h(x_3,y_3)\otimes h^*(-x_3,-y_3)+g(x_3,y_3)\otimes g^*(-x_3,-y_3)+\right. \\ &+h(x_3,y_3)\otimes g^*(-x_3,-y_3)\otimes\delta(x_3,y_3-Y)+ \\ &\left.+h^*(-x_3,-y_3)\otimes g(x_3,y_3)\otimes\delta(x_3,y_3+Y)\right] \end{aligned} \tag{2.63}$$

The complex distribution is composed of four terms. The first two terms respectively represent the cross-correlation of the impulse response h and itself as well as the

cross-correlation of the data g and itself. The third and fourth terms represent the cross-correlations of h and g. The third and fourth terms are typically the same except that the third cross-correlation is centered at $(0, Y)$ and the fourth term is centered at $(0, -Y)$. There is not too much thing to do with the first and second terms but the third and fourth terms are of interest. The cross-correlations of h and g can be written as in Equation 2.64 and Equation 2.65.

$$h(x_3, y_3) \otimes g^*(-x_3, -y_3) \otimes \delta(x_3, y_3 - Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(\xi, \eta)\, g^*\left(\xi - x_3, \eta - \left(y_3 - Y\right)\right) d\xi d\eta$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(\xi, \eta)\, g^*(\xi - x_3, \eta - y_3 + Y)\, d\xi d\eta \quad (2.64)$$

Note that,

$$h(x_3, y_3) \otimes g^*(-x_3, -y_3) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(\xi, \eta)\, g^*(\xi - x_3, \eta - y_3)\, d\xi d\eta$$

Similarly,

$$h^*(-x_3, -y_3) \otimes g(x_3, y_3) \otimes \delta(x_3, y_3 + Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(\xi, \eta)\, h^*(\xi - x_3, \eta - y_3 - Y)\, d\xi d\eta \quad (2.65)$$

From digital signal processing (DSP), the main difference between cross-correlation and convolution is that in cross-correlation the functions $g^*(-x_3, -y_3)$ and $h^*(-x_3, -y_3)$ are not rotated by 180^o before doing the cross-correlation process but in convolution they must be rotated by 180^o before doing a convolution process. To get the convolution of the impulse response h and the data g, the input transparency of h or g (just one of them) in the recording stage must be rotated by 180^o along the spatial coordinate x_1 as well as along the spatial coordinate y_1. If the input transparency of the impulse response h is rotated then it will be $h\left(-x_1, -y_1 + \frac{Y}{2}\right)$ instead of $h\left(x_1, y_1 - \frac{Y}{2}\right)$; similarly, if the input transparency of the data g is rotated then it will be $g\left(-x_1, -y_1 - \frac{Y}{2}\right)$ instead of $g\left(x_1, y_1 + \frac{Y}{2}\right)$.

From digital signal processing (DSP), the bandwidth of a resulting function from the convolution, correlation or cross-correlation of two functions is equal to the sum of their bandwidths [13]. For example, if the bandwidths of the functions $m(t)$ and $f(t)$ are three and one respectively; then the bandwidth of their convolution is equal to four. Therefore, the bandwidths of the patterns of the cross-correlated field in the output plane P_3 along the spatial coordinates x_3 and y_3 are illustrated in Figure 2.11.

Obviously from Figure 2.11, in order to prevent overlapping between the patterns of the cross-correlated field that are centered at $(0, 0)$, $(0, Y)$ and $(0, -Y)$ (i.e. they are fully separated); the distance between the centers of the input transparencies must satisfy Relation 2.1 where W_g and W_h are the widths of g and h respectively in the

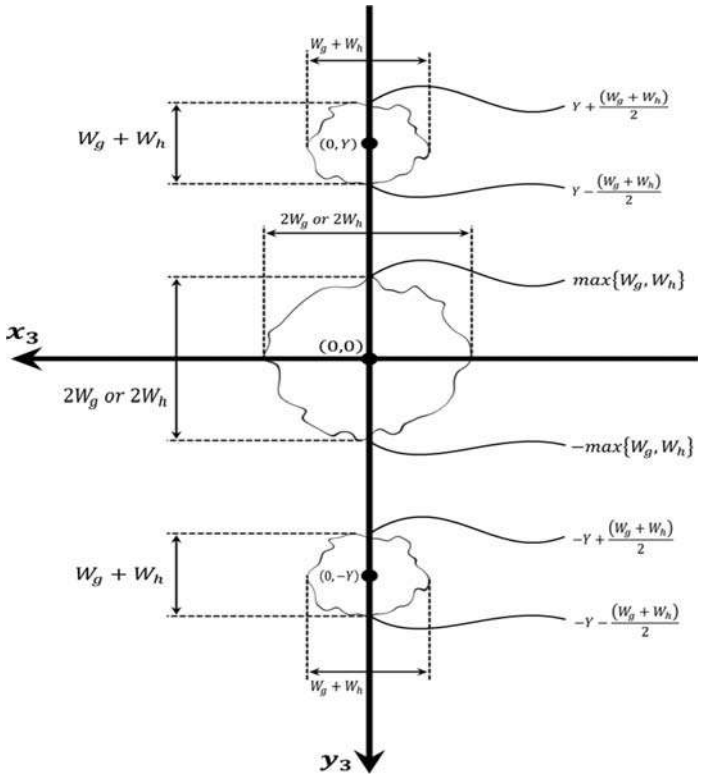

Figure 2.11: The bandwidths of the patterns of the cross-correlated field in the output plane P_3 along the spatial coordinates x_3 and y_3.

direction of the y-coordinate. It is really important to be noticed that in Figure 2.11, the highest cross-correlation exists in the center of each cross-correlated pattern and decreases by moving away from the center.

Relation 2.1.

$$Y > max\left\{W_h, W_g\right\} + \frac{W_g}{2} + \frac{W_h}{2}$$

The constant c is added to the distance Y for realizing Relation 2.1. In order to confine the patterns of the cross-correlations of the impulse response h and the data g in the output plane P_3, the distance d_1 which is from the horizontal axis x_1 to the top edge of the input plane P_1 shown in Figure 2.12 must be bigger than or equal to the distance D obtained in Equation 2.66 which is from the horizontal axis x_3 in the output plane P_3 to the top edge of the pattern centered at $(0, Y)$ or the bottom edge of the pattern centered at $(0, -Y)$ shown in Figure 2.13; as well as the distance d_2 which is from the horizontal axis x_1 to the bottom edge of the input plane P_1 shown in Figure 2.12 must be bigger than or equal to the distance D. For making d_1 is equal to D, the amount $D - r_1$ has to be added to the distance r_1; where the distance r_1 obtained in Equation 2.67 is from the horizontal axis x_1 to the top edge of h as shown

in Figure 2.12. Similarly, to make d_2 is equal to D, the amount $D - r_2$ has to be added to the distance r_2; where the distance r_2 obtained in Equation 2.68 is from the horizontal axis x_1 to the bottom edge of g as shown in Figure 2.12. The distances $D - r_1$ and $D - r_2$ are respectively obtained as in Equation 2.69 and Equation 2.70. Finally, the distance d_1 will be equal to the distance D as in Equation 2.71; similarly, the distance d_2 will also be equal to D as in Equation 2.72.

$$D = Y + c + \frac{W_g}{2} + \frac{W_h}{2} = max\left\{W_h, W_g\right\} + W_h + W_g + c \tag{2.66}$$

$$r_1 = \frac{Y+c}{2} + \frac{W_h}{2} = \left[\frac{max\{W_h,W_g\}}{2} + \frac{W_h}{4} + \frac{W_g}{4} + \frac{c}{2}\right] + \frac{W_h}{2} = \frac{max\{W_h,W_g\}}{2} + \frac{3}{4}W_h + \frac{1}{4}W_g + \frac{c}{2} \tag{2.67}$$

$$r_2 = \frac{Y+c}{2} + \frac{W_g}{2} = \left[\frac{max\{W_h,W_g\}}{2} + \frac{W_h}{4} + \frac{W_g}{4} + \frac{c}{2}\right] + \frac{W_g}{2} = \frac{max\{W_h,W_g\}}{2} + \frac{1}{4}W_h + \frac{3}{4}W_g + \frac{c}{2} \tag{2.68}$$

$$D - r_1 = \frac{1}{2}max\left\{W_h, W_g\right\} + \frac{1}{4}W_h + \frac{3}{4}W_g + \frac{c}{2} \tag{2.69}$$

$$D - r_2 = \frac{1}{2}max\left\{W_h, W_g\right\} + \frac{3}{4}W_h + \frac{1}{4}W_g + \frac{c}{2} \tag{2.70}$$

$$d_1 = r_1 + (D - r_1) = D \tag{2.71}$$

$$d_2 = r_2 + (D - r_2) = D \tag{2.72}$$

Lastly, the joint transform correlator has a great feature is that its ability to change the filter impulse response quickly; therefore, it is considered beneficial for real-time systems. On the other hand, it has a defect is that the input bandwidth of the data is reduced due to the filter impulse response is introduced simultaneously with the data to be filtered.

2.3.4.1 Sampling Issues

2.3.4.1.1 Introduction

Sampling is considered the first important step for simulating optical models. In order to generate the cross-correlated field in the back focal plane P_3; then the input plane P_1, the back focal plane of the lens L_2 and the back focal plane of the lens L_4 must be sampled properly. Sampling for each one of these is completely discussed in this sub-subsection.

2.3.4.1.2 Sampling of the Input Plane P_1

The input plane P_1 is located in a rectangular array has N samples along the spatial space coordinate x_1 and M samples along the spatial space coordinate y_1. L_{x_1} is the physical side length in meter (m) of the array in the x_1 direction; similarly, L_{y_1} is

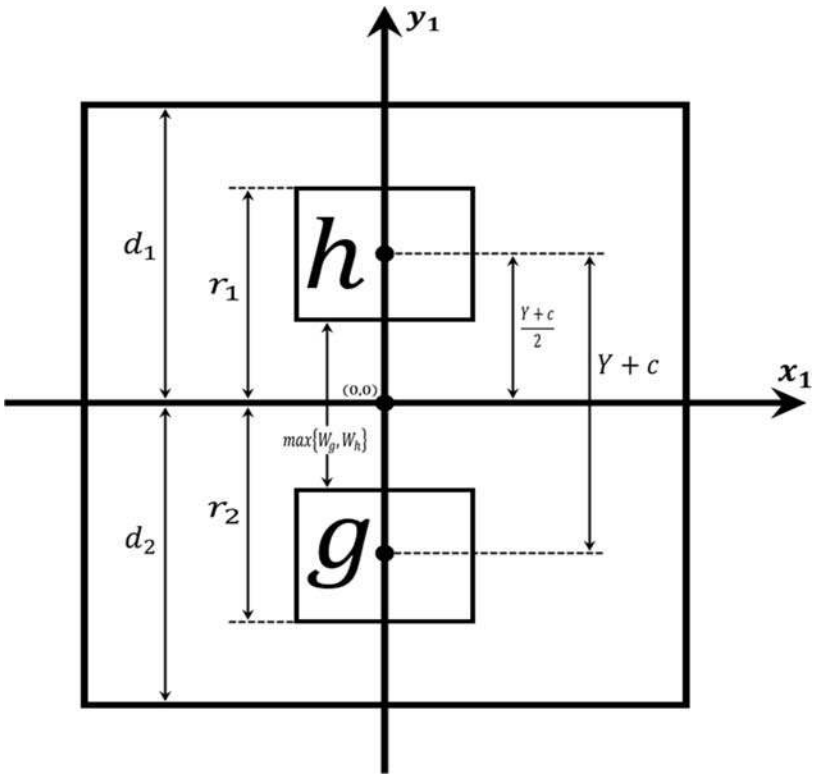

Figure 2.12: The alignment of the input transparencies in the input plane P_1.

the physical side length in meter of the array in the y_1 direction. Then the sample spacing Δx_1 along the x_1-coordinate in meter is equal to $\frac{L_{x_1}}{N}$; and the sample spacing Δy_1 along the y_1-coordinate in meter is equal to $\frac{L_{y_1}}{M}$.

2.3.4.1.3 Sampling of the Back Focal Plane P_2

From Fourier transform, the spatial frequency coordinate f_{X_2} of the back focal plane P_2 in cycles per meter $\left(\frac{cyc}{m}\right)$ is obtained as in Equation 2.73.

$$f_{X_2} = \frac{x_2}{\lambda f} \tag{2.73}$$

By turning around Equation 2.73, the spatial space coordinate x_2 can be expressed as in Equation 2.74.

$$x_2 = \lambda f f_{X_2} \tag{2.74}$$

Note that, x_2 is in meter (m) because the units of the variables in Equation 2.74 can be concluded as $m \times m \times \frac{cyc}{m} = m$. Then the spatial space sampling interval Δx_2 in

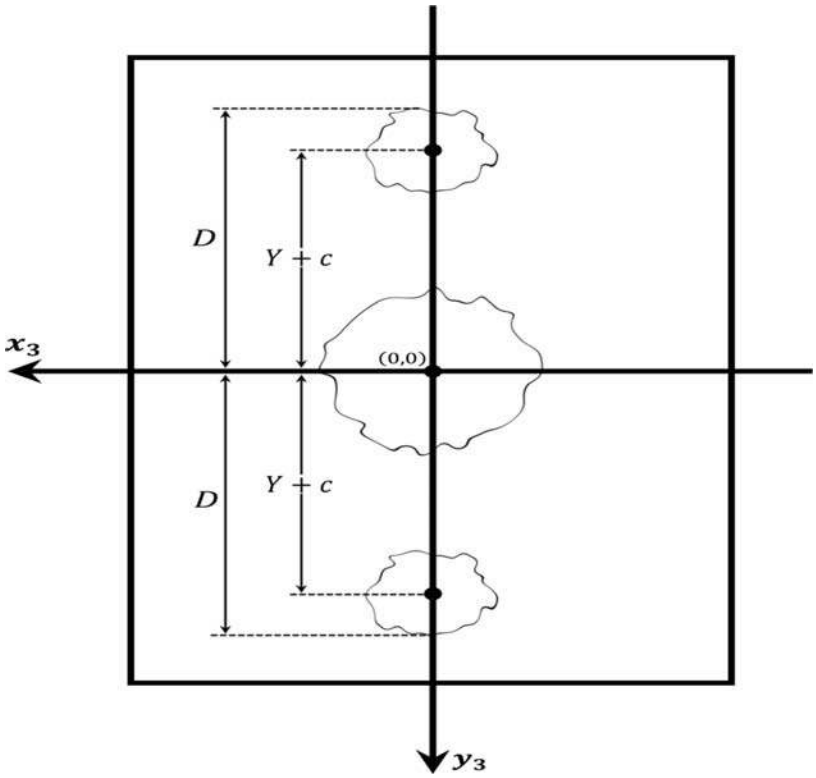

Figure 2.13: The alignment of the patterns of the cross-correlated field in the output plane P_3.

meter (m) along the spatial space coordinate x_2 is computed as in Equation 2.75.

$$\Delta x_2 = \lambda f \Delta f_{X_2} \tag{2.75}$$

From discrete Fourier transform (DFT), the relationship between the spatial space sampling interval Δx_1 in the input plane and the spatial frequency sampling interval Δf_{X_2} is shown in Equation 2.76.

$$\Delta f_{X_2} = \frac{1}{N \Delta x_1} \tag{2.76}$$

For more information about how the relation in Equation 2.76 is obtained; Reference [14], Subsection 4.4.2 can be consulted. By substitution from Equation 2.76 in Equation 2.75 then the spatial space sampling interval Δx_2 in meter (m) along the spatial space coordinate x_2 is obtained as in Equation 2.77.

$$\Delta x_2 = \frac{\lambda f}{N \Delta x_1} \tag{2.77}$$

Similarly, the spatial space sampling interval Δy_2 in meter (m) along the spatial

space coordinate y_2 can be computed as in Equation 2.78.

$$\Delta y_2 = \frac{\lambda f}{M \Delta y_1} \tag{2.78}$$

2.3.4.1.4 Sampling of the Back Focal Plane P_3

From Fourier transform, the spatial frequency coordinate f_{X_3} of the back focal plane P_3 in cycles per meter $\left(\frac{cyc}{m}\right)$ is obtained as in Equation 2.79.

$$f_{X_3} = \frac{x_3}{\lambda f} \tag{2.79}$$

By turning around Equation 2.79, the spatial space coordinate x_3 can be expressed as in Equation 2.80.

$$x_3 = \lambda f f_{X_3} \tag{2.80}$$

Then the spatial space sampling interval Δx_3 in meter (m) along the spatial space coordinate x_3 is computed as in Equation 2.81.

$$\Delta x_3 = \lambda f \Delta f_{X_3} \tag{2.81}$$

From discrete Fourier transform (DFT), the relationship between the spatial space sampling interval Δx_2 in the back focal plane P_2 and the spatial frequency sampling interval Δf_{X_3} is shown in Equation 2.82.

$$\Delta f_{X_3} = \frac{1}{N \Delta x_2} \tag{2.82}$$

By substitution from Equation 2.82 in Equation 2.81 then the spatial space sampling interval Δx_3 in meter (m) along the spatial space coordinate x_3 is obtained as in Equation 2.83.

$$\Delta x_3 = \frac{\lambda f}{N \Delta x_2} = \Delta x_1 \tag{2.83}$$

Similarly, the spatial space sampling interval Δy_3 in meter (m) along the spatial space coordinate y_3 can be computed as in Equation 2.84.

$$\Delta y_3 = \frac{\lambda f}{M \Delta y_2} = \Delta y_1 \tag{2.84}$$

The spatial space sampling intervals in the input plane P_1 and the back focal plane P_3 are equal because the focal lengths of the lenses L_2 and L_4 are equal; but if the focal lengths are not identical then the sampling intervals will not be equal anymore.

2.3.4.2 Applications of the JTC

2.3.4.2.1 Introduction

The joint transform correlator (JTC) is one of the techniques frequently used in the field of optical pattern identification and classification. It plays an important role in object detection, face recognition, fingerprint recognition, and many other areas. In this sub-subsection, the usage of the JTC for object detection and face recognition is fully covered.

2.3.4.2.2 Face Recognition

The joint transform correlator (JTC) is used to recognize an unknown face object based on a database of desired impulses. The database is composed of n, $N \times N$ images of faces (impulses) on black backgrounds such that I_k where $k = 1, \ldots, n$; and n is the total number of the impulses. For decreasing the error rates of face recognition and object detection, all face projections must be defined in the database; impulses must have the same size; impulses must have only faces and they are expanded to the boarders of the images; finally, impulses must have unified backgrounds to discriminate between the pixels occupying the backgrounds and the pixels on the faces. For doing face recognition, an unknown face object have to be picked. It must have the same properties of the impulses. Hence, it must have the same size as the impulses; it has a black background; it has a projection as one of the impulses; and it has only face that is expanded to the boarders of the image. The pixels occupying the faces of the impulses and the unknown face object are normalized in the same manner as the normalization of the pixels occupying the faces of the training faces in Step 2 in Sub-subsection 2.2.2.6.

The cross-correlated field between the unknown face object and each impulse is obtained. An adaptive filtering mask is used to produce the cross-correlated patterns that are centered at $(0, Y)$. The maximum values of the cross-correlated patterns are computed; such that c_1 is the maximum value of the cross-correlated pattern between the unknown face object and the first impulse; similarly, c_n is the maximum value of the cross-correlated pattern between the unknown face object and the n^{th} impulse.

Then recognition can be performed by using Condition 2.4 where bc_k is the biggest cross-correlation among the maximum values of the cross-correlated patterns between the unknown face object and impulses.

Condition 2.4. *If,*

$$bc_k = max\left([c_1, \ldots, c_n]\right), \qquad \textit{where } k \textit{ can be any number between 1 to } n$$

Then the unknown face object is recognized as the k^{th} impulse response; otherwise, it is an unknown face object.

2.3.4.2.2.1 Setting up a Recognition Threshold

Unfortunately, when the unknown face object does not have a similar impulse response then getting the biggest cross-correlation bc_k among the maximum values of the cross-correlated patterns does not *always* mean that the unknown face object is recognized as the k^{th} impulse response. Therefore, a certain threshold must be used to increase the accuracy of recognition.

For setting up a recognition threshold, some different objects are taken for each impulse; consequently, the objects are known here just for picking the threshold. The cross-correlated patterns that are centered at $(0, Y)$ resulting from the cross-correlations between each impulse and its *corresponding* objects are produced. After that, the maximum values of the cross-correlated patterns between the impulses and their *corresponding* objects are obtained; and they are stacked in the row vectors $\boldsymbol{T}_1, \ldots, \boldsymbol{T}_k, \ldots, \boldsymbol{T}_n$, where $\boldsymbol{T}_k$ contains the biggest expected cross-correlations because the k^{th} impulse response and its *corresponding* objects have the same person and the same face position.

Thereafter, the mean m_k and the standard deviation STD_k (the average distance from the mean to a point) are computed for each row vector $\boldsymbol{T}_k$ of the biggest cross-correlations. By calculating the means and the standard deviations, a certain threshold is established for each impulse response.

Then the recognition threshold can be applied to recognize the unknown face object by using Condition 2.5.

Condition 2.5. *If,*

$$m_k - STD_k \leq bc_k \leq m_k + STD_k, \qquad \textit{where } k = 1, \ldots, n$$

Then the unknown face object is recognized as the k^{th} impulse response; otherwise, it is an unknown face object.

2.3.4.2.3 Object Detection

The joint transform correlator (JTC) is used to detect if an unknown object contains a face or not based on a determined threshold for detection. Therefore, in object detection, only a detection threshold is needed.

To obtain the detection threshold, objects are generated from the database of the impulses in the same manner as in Sub-sub-sub-subsection 2.3.4.2.2.1. The cross-correlated patterns that are centered at $(0, Y)$ resulting from the cross-correlations between each impulse and all objects are produced. The maximum values of the cross-correlated patterns between the impulses and all objects are computed and placed in the row vector $\boldsymbol{S}$. After that, the mean $m_{\boldsymbol{S}}$ and the standard deviation $STD_{\boldsymbol{S}}$ (the average distance from the mean to a point) are computed for the vector $\boldsymbol{S}$. By calculating the mean and the standard deviation, the detection threshold is established.

For applying the detection threshold, the unknown object $I^{Unknown}$ is picked for detection. The maximum value $c^{Unknown}$ of the cross-correlated pattern between the unknown object and any impulse response is computed. Then the unknown object $I^{Unknown}$ can be detected by means of Condition 2.6.

Condition 2.6. *If,*

$$m_{\boldsymbol{S}} - STD_{\boldsymbol{S}} \leq c^{Unknown} \leq m_{\boldsymbol{S}} + STD_{\boldsymbol{S}}$$

Then the unknown object $I^{Unknown}$ is detected as a face object; otherwise, it is not a face object.

2.3.4.3 Analysis of JTC Performance

2.3.4.3.1 Introduction

Joint transform correlator (JTC) performance is analyzed through analyzing the performance of its applications. In order to study the performance of face recognition and object detection, their error rates are computed and analyzed. Hence, in this sub-subsection, attention is paid for calculating the error rates of face recognition and object detection.

2.3.4.3.2 Analysis of Recognition Performance

The error rate computes the percentage of error in recognition. For computing the error rate of face recognition, let L is the total number of objects; SR is the total number of successes in the recognition of the objects; and FR is the total number of failures in the recognition of the objects. Then the error rate can be computed as in Equation 2.85.

$$ER\,(\%) = \frac{FR}{L} \times 100 \tag{2.85}$$

2.3.4.3.2.1 Improvement of Recognition Performance

The error rate of face recognition is usually big then it has to be improved. One of the techniques for decreasing the error rate is optimizing the database of the impulses. The optimization of the database means trying to find the best combination of impulses that ensures the smallest error rate.

Some different objects are taken for each impulse response; consequently, the objects are known here just for optimizing the database of the impulses. Then this technique can be simulated by trying different combinations out of the generated objects until finding a combination that produces the lowest error rate of face recognition then the combination can be used to form the database of the impulses.

2.3.4.3.3 Analysis of Detection Performance

The error rate computes the percentage of error in detection. For computing the error rate of object detection, let L is the total number of objects; SD is the total number of successes in the detection of the objects; and FD is the total number of failures in the detection of the objects. Then the error rate can be computed as in Equation 2.86.

$$ER\,(\%) = \frac{FD}{L} \times 100 \tag{2.86}$$

2.3.4.3.3.1 Improvement of Detection Performance

The error rate of object detection is usually big then it has to be improved. The error rate is improved by optimizing the database of the impulses in the same manner as the optimization of the error rate of face recognition in Sub-sub-sub-subsection 2.3.4.3.2.1.

Chapter 3

Modelling

3.1 Digital Modelling

3.1.1 Introduction

In this section, PCA and IPCA algorithms are simulated by using the MATLAB © software when the data set is composed of images; as well as comparison between two algorithms is illustrated. The simulation begins from generating the database of the training faces until projecting the training faces on the eigenspace. The applications of the PCA and IPCA algorithms are simulated too. The complete MATLAB © code that had been written to simulate this work is shown in Appendix A.

3.1.2 Application of the PCA Algorithm to Images

The steps for analyzing the PCA algorithm to images presented in Sub-subsection 2.2.2.6 are simulated as follows:

Step 1: the database is created to be composed of nine, 50×50 images of faces (training faces) on black backgrounds such that I_k where $k = 1, \ldots, 9$. The training faces are taken for three people where each person has three training faces with different projections as shown in Figure 3.1.

Step 2: the average histogram for all training faces is obtained as in Figure 3.2. From Figure 3.2, it turns out all intensity levels below the threshold represent the backgrounds of images because these levels have the biggest histogram; and all intensity levels above the threshold represent the faces; therefore, the threshold is equal to eight. The threshold is applied on the training faces; Figure 3.3 shows how good the applied threshold is on the training faces. As seen from Figure 3.3, the applied

Figure 3.1: The database of the training faces.

threshold is doing pretty well; and it can be used for normalizing the training faces as in Figure 3.4.

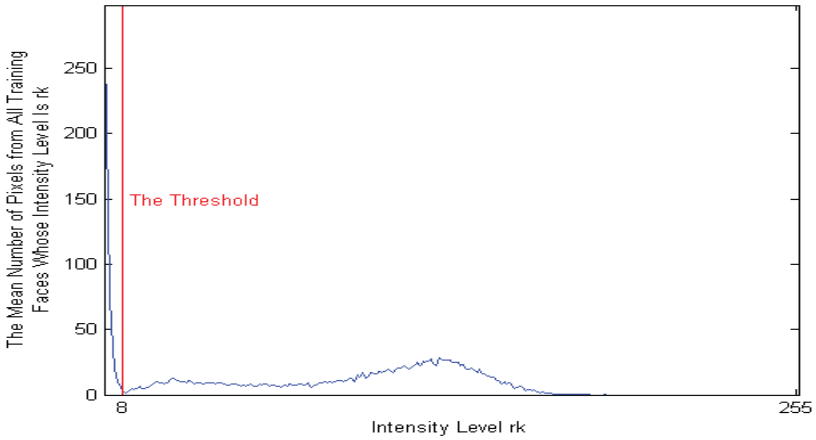

Figure 3.2: The average histogram for all training faces.

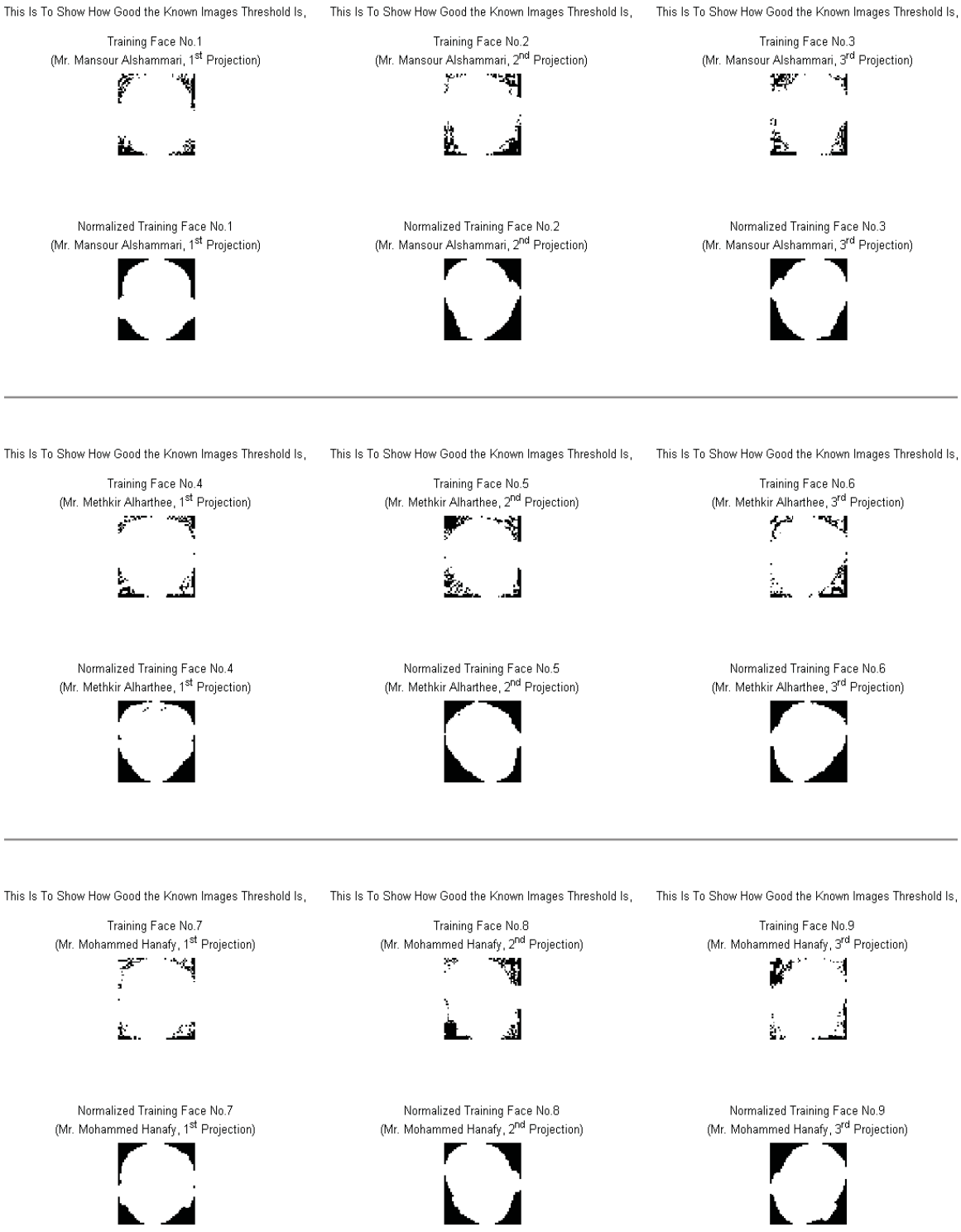

Figure 3.3: The application of the threshold on the training faces.

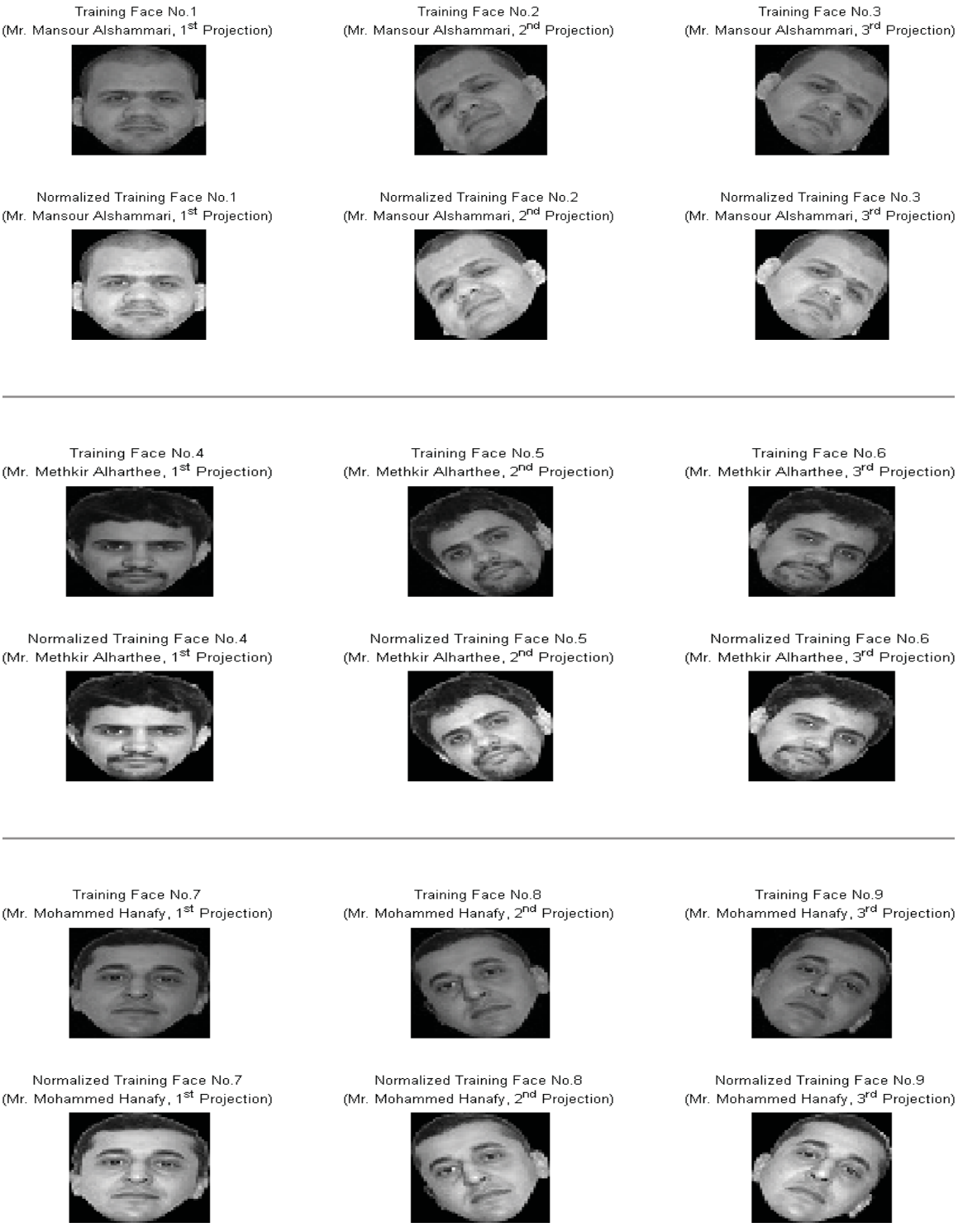

Figure 3.4: The normalization of all training faces by means of the selected threshold.

Step 3: all training faces are centered as shown in Figure 3.5.

Figure 3.5: The centered training faces.

Step 4: all centered training faces are represented as 50^2 column vectors.

Step 5: The average training face vector $\mathbf{\Psi}$ is computed and the average training face is shown in Figure 3.6.

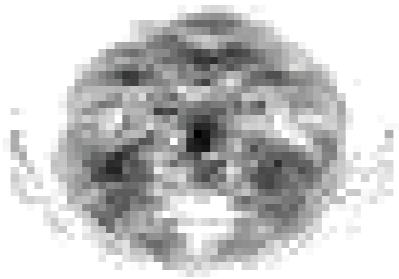

Figure 3.6: The average training face. Note that, this is a negative image.

Step 6: the set of the training faces is centered; and the centered (with respect to the set of the training faces) training faces are presented in Figure 3.7.

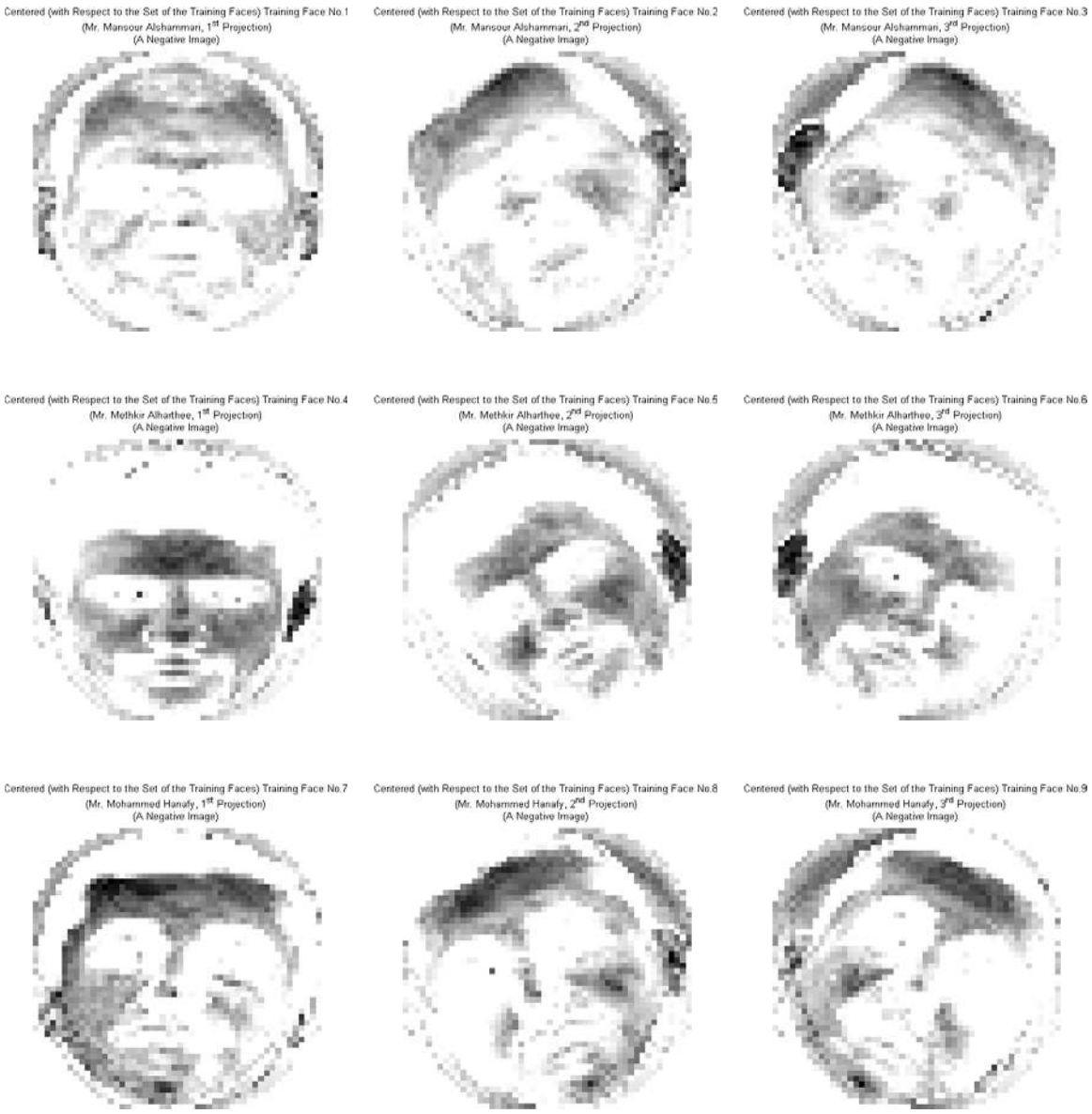

Figure 3.7: The centered (with respect to the set of the training faces) training faces. Note that, these are negative images.

Step 7: a 2500×2500 covariance matrix for all training faces is calculated.

Step 8: the eigenvectors and eigenvalues of the covariance matrix C are computed.

Step 9: for choosing principal components and forming the feature vector matrix A_q, the variance contribution rate (VCR) and the total variance contribution rate (TVC) are computed in Table 3.1. When the TVC is over 95.5% then q eigenvectors associated with the biggest q eigenvalues are selected. Hence, Based on Table 3.1, the biggest eight eigenvectors associated with the biggest eigenvalues are selected to form the feature vector matrix. The selected eigenfaces (the highly correlated eigenvectors) are shown in Figure 3.8.

Table 3.1: The calculations of the VCR and TVC.

$\boldsymbol{k}$	$\boldsymbol{\lambda_k}$	$\boldsymbol{VCR_k}$ (%)	$\boldsymbol{TVC}$ (%)
1	9732585.1016	29.0361	29.0361
2	6578765.9806	19.6270	48.6631
3	4576184.9630	13.6525	62.3157
4	4054857.8169	12.0972	74.4129
5	3000248.2414	8.9509	83.3638
6	2204809.6081	6.5778	89.9416
7	1816000.6101	5.4178	95.3595
8	1555452.9118	4.6405	100
9	0	0	100

Step 10: the principal components transform is performed and the training faces are projected on the eigenspace.

3.1.3 Application of the IPCA Algorithm to Images

The IPCA algorithm shown in Sub-subsection 2.2.3.2 is used for calculating the IR and AIR for the computed eigenvectors of the covariance matrix C in Subsection 3.1.2 where d here is equal to the total number of the training faces n; and the results are shown in Table 3.2. When the AIR is over 95.5% then q eigenvectors associated with the biggest q eigenvalues are selected. Hence, Based on Table 3.2, the biggest seven eigenvectors associated with the biggest eigenvalues are selected to form the feature vector matrix A_q. The selected eigenfaces (the highly correlated eigenvectors) are the first seven selected eigenfaces by using the PCA algorithm shown in Figure 3.8.

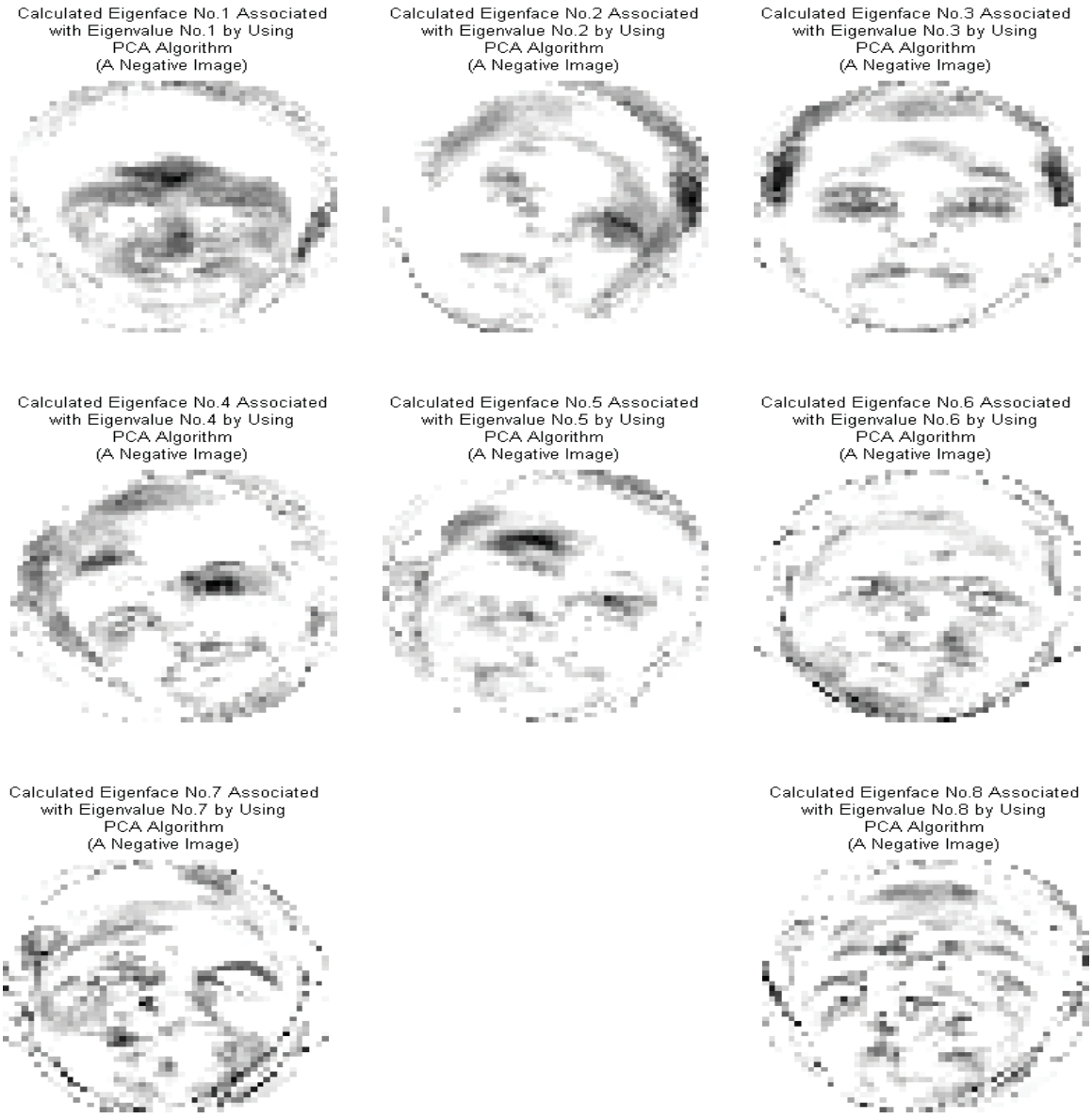

Figure 3.8: The eigenfaces. Note that, these are negative images.

3.1.4 Comparison of the PCA and IPCA Algorithms

Comparison between the PCA and IPCA algorithms is based on the values of the TVC and AIR that determine the selected eigenfaces for the feature vector matrix A_q in Subsection 3.1.2, Step 9 and in Subsection 3.1.3.

When $k = 7$ in Table 3.2, the AIR is equal to 95.6891% which is bigger than 95.5% but the TVC is slightly small; consequently, if the eigenvectors are selected

Table 3.2: The calculations of the IR and AIR.

k	λ_k	ρ_k	$I(\lambda_k)$	IR_k (%)	AIR (%)
1	9732585.1016	0.7096	0.4948	31.1180	31.1180
2	6578765.9806	0.8037	0.3152	19.8224	50.9404
3	4576184.9630	0.8635	0.2118	13.3174	64.2578
4	4054857.8169	0.8790	0.1860	11.6978	75.9555
5	3000248.2414	0.9105	0.1353	8.5073	84.4628
6	2204809.6081	0.9342	0.0982	6.1729	90.6357
7	1816000.6101	0.9458	0.0804	5.0534	95.6891
8	1555452.9118	0.9536	0.0686	4.3109	100
9	0	1	0	0	100

based on the TVC then the first eight eigenvectors must be taken in order to make sure that the TVC is big enough. Therefore, the AIR tells us more about the information contained in the eigenfaces. Figure 3.9 shows a comparison between the biggest calculated eigenvalues by using the PCA and IPCA algorithms as well as the calculated eigenvalues from the covariance matrix C for all training faces.

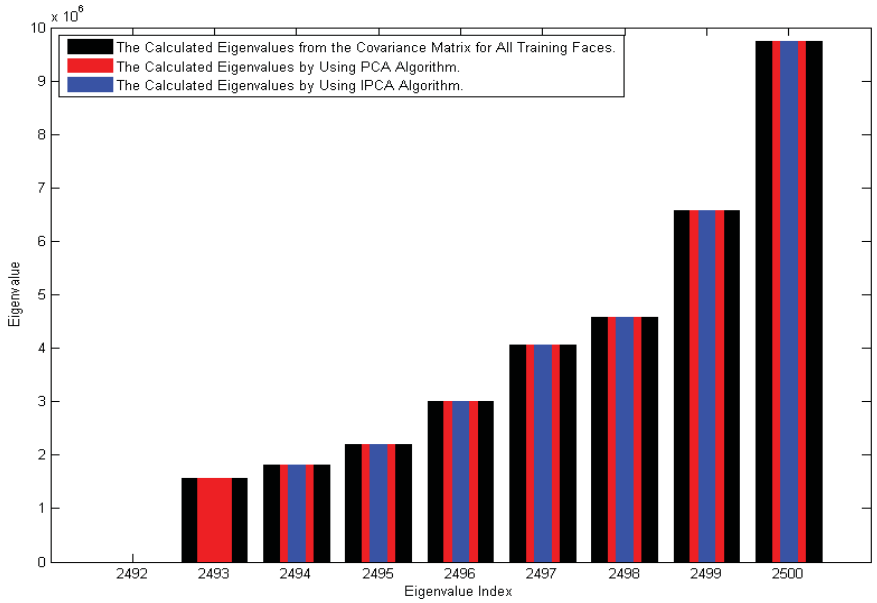

Figure 3.9: A comparison between the biggest calculated eigenvalues by using the PCA and IPCA algorithms as well as the calculated eigenvalues from the covariance matrix for all training faces.

3.1.5 Image Compression

In image compression modelling, the training faces in Figure 3.1 are projected on the eigenspace and recostructed again by using the PCA and IPCA algorithms; as well as by using different selected eigenfaces to form the feature vector matrix A_q.

3.1.6 Face Recognition

The database of the training faces created in Step 1 in Subsection 3.1.2 is used for recognition. Some tested images are taken out of the database for setting up a recognition threshold. Thirty-six tested images are taken for the first projection of each person; twelve tested images are taken for the second projection of each person; and twelve tested images are taken for third projection of each person; therefore, the total number of the tested images is equal to 180. Some samples of the tested images are shown in Figure 3.10; the full database of the tested images is shown in Appendix B.

By using the PCA algorithm, a recognition threshold is specified for each training face by means of the method explained in Sub-sub-subsection 2.2.5.3.1. The tested images are processed as unknown images of faces as illustrated in Sub-subsection 2.2.5.3. Then Condition 2.2 is used for recognizing the tested images. The recognition results of the tested images in Figure 3.10 by using the PCA algorithm are presented in Table 3.3. The recognition of all 180 tested images by using the PCA algorithm is shown in Appendix C. Finally, the tested images can be recognized in the same way when the IPCA algorithm is used; or different eigenfaces are selected to form the feature vector matrix.

Table 3.3: The recognition of the tested images in Figure 3.10.

Tested Image No.	Input Face	Recognized Output Face	Status
4	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
15	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
38	Mr. Mansour Alshammari	Unknown Image	Failure
62	Mr. Methkir Alharthee	Unknown Image	Failure
72	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
120	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
124	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
160	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
179	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success

Figure 3.10: Some samples of the tested images.

3.1.7 Image Detection

The generated database of the tested images in Subsection 3.1.6 is used for detection. By using the PCA algorithm, a detection threshold is specified by means of the method explained in Sub-subsection 2.2.5.4. Then Condition 2.3 is used for detecting

the tested images. The detection results of the tested images in Figure 3.10 by using the PCA algorithm are presented in Table 3.4. The detection of all 180 tested images by using the PCA algorithm is shown in Appendix D. Finally, the tested images can be detected in the same way when the IPCA algorithm is used; or different eigenfaces are selected to form the feature vector matrix.

Table 3.4: The detection of the tested images in Figure 3.10.

Tested Image No.	Input Image	Detected Output Image	Status
4	a face	a face	Success
15	a face	a face	Success
38	a face	not a face	Failure
62	a face	a face	Success
72	a face	a face	Success
120	a face	not a face	Failure
124	a face	a face	Success
160	a face	a face	Success
179	a face	a face	Success

3.2 Optical Modelling

3.2.1 Introduction

In this section, the joint transform correlator (JTC) is fully simulated by using the MATLAB © software. The simulation begins from scratch until generating the desired pattern of the cross-correlated field in the back focal plane P_3 by using an adaptive filtering mask designed for that. The JTC applications are simulated too. The complete MATLAB © code that had been written to simulate this work is shown in Appendix F.

In the simulation, lenses are assumed to be ideally focused (i.e. the model is an aberration-free system) as in Figure 3.11; where f in the figure is the focal length.

3.2.2 Simulation of the JTC

We are at level that we can start in simulating the joint transform correlator (JTC). The desired impulse response h and the data g (here it is called the object) are 100×100 images of faces for two people. They are respectively shown in Figure 3.12 and Figure 3.13. The impulse response and the object are normalized in order to remove lighting effects on them then increasing the accuracy of cross-correlation. To

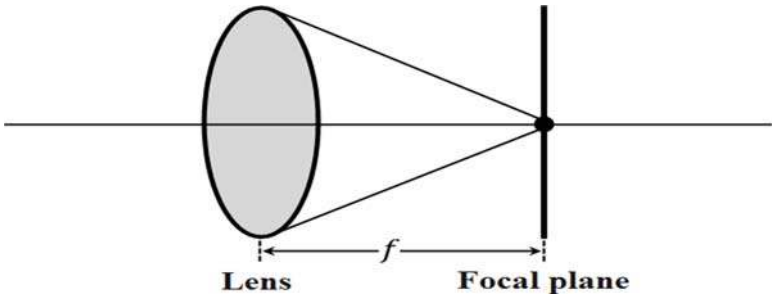

Figure 3.11: A focused lens.

keep variations among them just in the faces without the effects of the backgrounds, the normalization is performed on the pixels occupying the faces in the same manner as the normalization of the pixels on the faces of the training faces in Step 2 in Sub-subsection 2.2.2.6.

Figure 3.12: The desired impulse response h.

Figure 3.13: The object g.

The transparencies of the impulse response h and the object g are located in a square array in the input plane P_1. The width W_g of g in the direction of the y-coordinate is 100 pixels; the width W_h of h in the direction of the y-coordinate is 100

pixels; then the distance Y that separates the centers of h and g is equal to 200 pixels. The constant c is selected to be 10 pixels then Relation 2.1 is satisfied. The distance D obtained in Equation 2.66 is 130 pixels; the distance r_1 obtained in Equation 2.67 is 155 pixels; the distance r_2 obtained in Equation 2.68 is 155 pixels; $D - r_1$ and $D - r_2$ are respectively equal to 155 pixels and 155 pixels; then the distances d_1 and d_2 obtained respectively in Equation 2.71 and Equation 2.72 are respectively equal to 310 pixels and 310 pixels. Since the distance d_1 is equal to the distance D as well as the distance d_2 is also equal to the distance D then the input transparencies of h and g are aligned properly in the input plane P_1.

The number of samples N along the spatial space coordinate x_1 in the input plane P_1 is 630; and the number of samples M along the spatial space coordinate y_1 is 630. The physical side length L_{x_1} of the array in the x_1 direction is $10\,(m)$; and the physical side length L_{y_1} of the array in the y_1 direction is $10\,(m)$. Then the sample spacing Δx_1 along the x_1-coordinate is equal to $\frac{10}{630} = 0.0159\,(m)$; and the sample spacing Δy_1 along the y_1-coordinate is equal to $\frac{10}{630} = 0.0159\,(m)$. The transmitted field $U_1\,(x_1, y_1)$ from the input plane P_1 is shown in Figure 3.14.

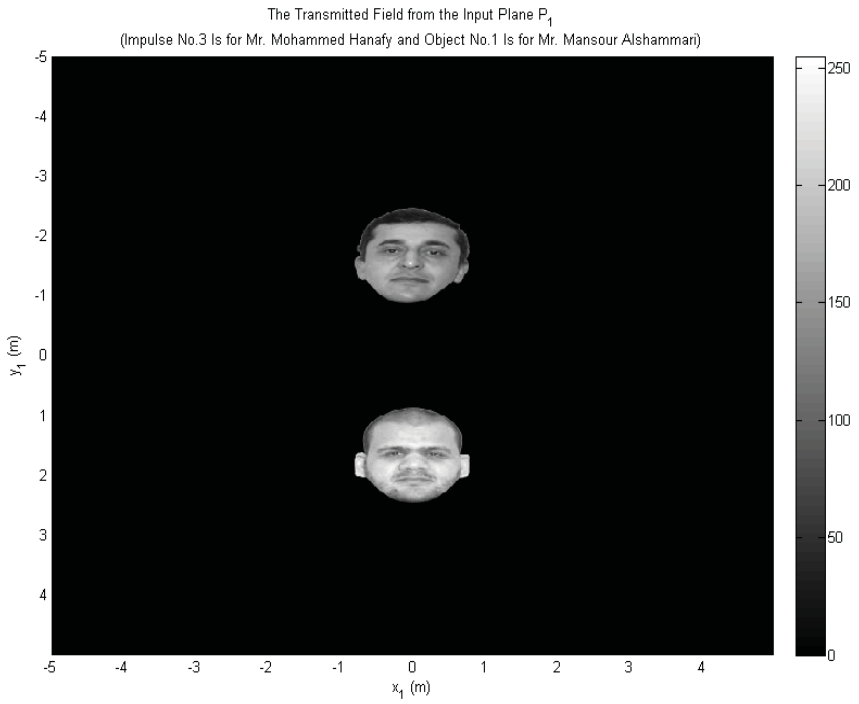

Figure 3.14: The transmitted field $U_1\,(x_1, y_1)$ from the input plane P_1.

The light wavelength λ is $550 \times 10^{-9}\,(m)$; and the focal length f is $0.055\,(m)$. Then the spatial space sampling interval Δx_2 along the spatial space coordinate x_2 is equal to $\frac{\lambda f}{N \Delta x_1} = \frac{550 \times 10^{-9} \times 0.055}{630 \times 0.0159} = 3.0250 \times 10^{-9}\,(m)$; similarly, the spatial space sampling

interval Δy_2 along the spatial space coordinate y_2 is equal to $\frac{\lambda f}{M \Delta y_1} = \frac{550 \times 10^{-9} \times 0.055}{630 \times 0.0159} = 3.0250 \times 10^{-9} (m)$. The incident intensity $I(x_2, y_2)$ on the back focal plane P_2 is shown in Figure 3.15.

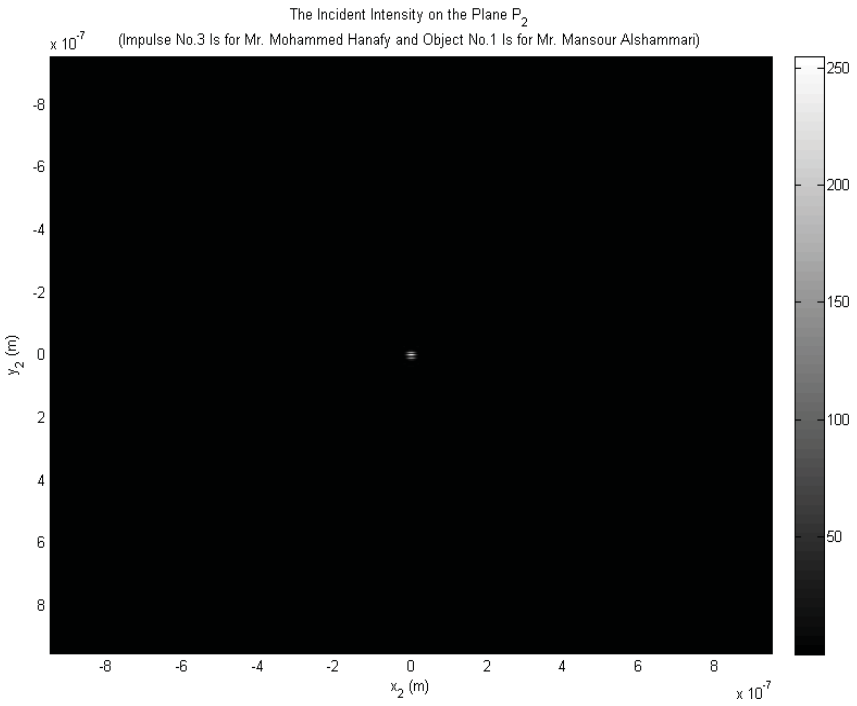

Figure 3.15: The incident intensity $I(x_2, y_2)$ on the back focal plane P_2.

Since the focal lengths of the lenses L_2 and L_4 are equal, the spatial space sampling intervals Δx_3 and Δy_3 in the back focal plane P_3 are respectively equal to the spatial space sampling intervals Δx_1 and Δy_1 in the input plane P_1. The cross-correlated field $U_3(x_3, y_3)$ in the back focal plane P_3 is obtained as in Figure 3.16 by calculating the inverse Fourier transform for the incident intensity $I(x_2, y_2)$ on the back focal plane P_2.

Figure 3.17 shows the designed adaptive mask for obtaining the desired pattern of the cross-correlations of the impulse response h and the object g in the back focal plane P_3. The mask produces the cross-correlated pattern that is centered at $(0, Y)$. It is multiplied by the cross-correlated field in the plane P_3 in order to obtain the filtered filed as in Figure 3.18.

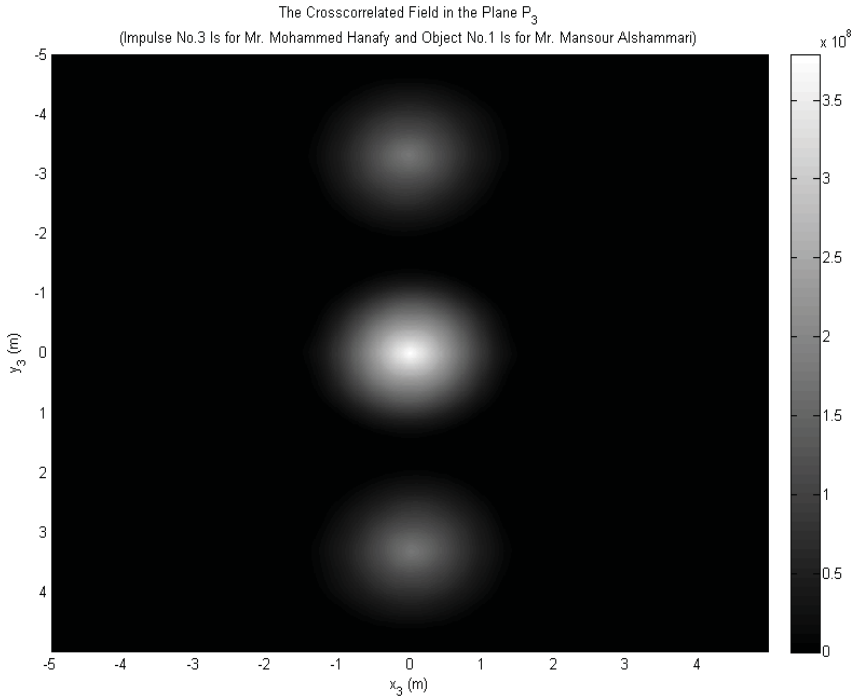

Figure 3.16: The cross-correlated field $U_3(x_3, y_3)$ in the back focal plane P_3.

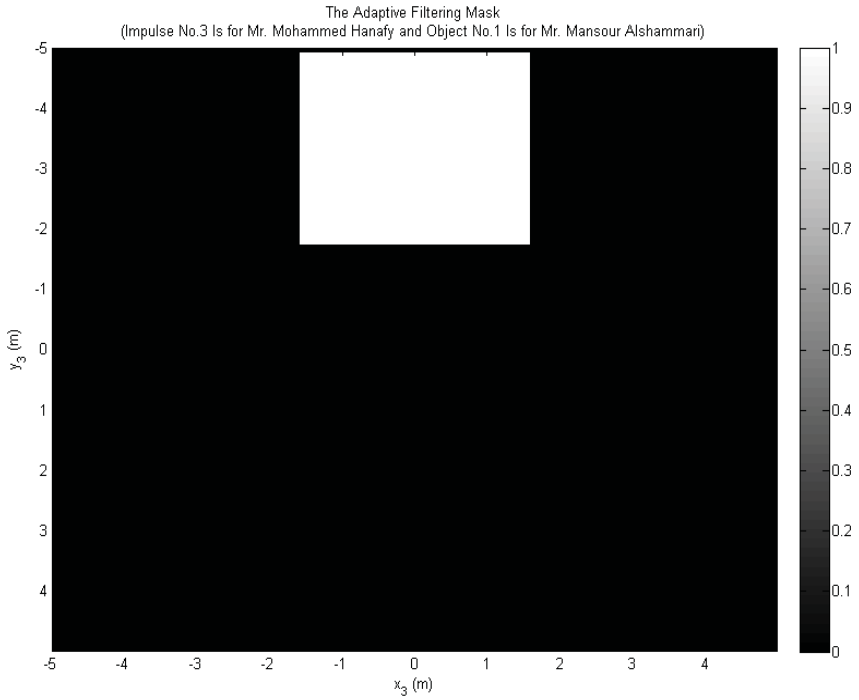

Figure 3.17: The adaptive filtering mask.

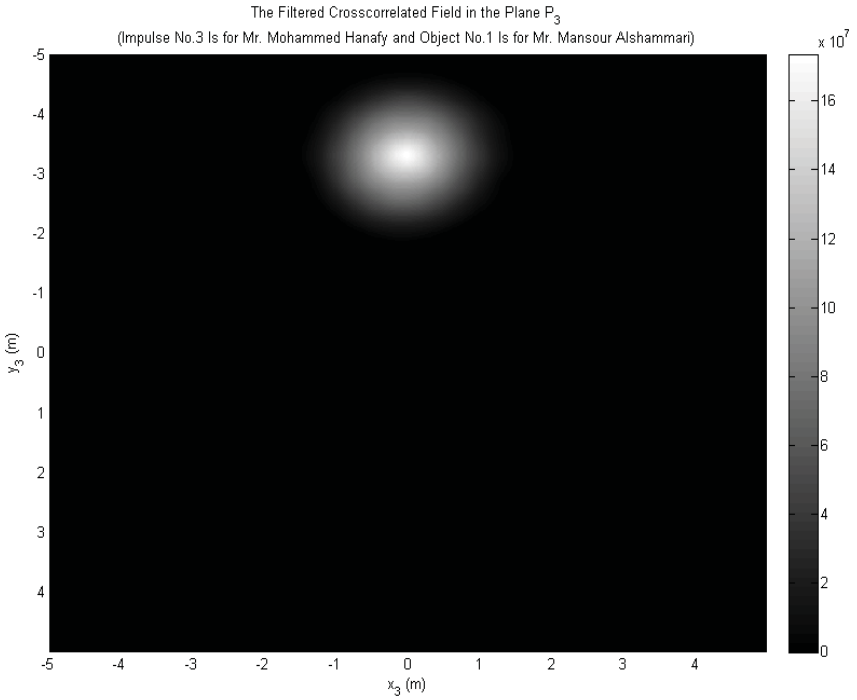

Figure 3.18: The filtered cross-correlated field in the back focal plane P_3.

3.2.3 Face Recognition

The database of the impulses for face recognition is picked to contain three, 100×100 impulses (images of faces) on black backgrounds. The impulses are taken for vertical faces to people's shoulders where oblique faces are ignored as shown in Figure 3.19. Taking just vertical faces will simplify the optimization of the database of the impulses. Some objects are taken out of the database for setting up a recognition threshold. Thirty-six objects are taken for each impulse then the total number of the objects is equal to 108. Some samples of the objects are shown in Figure 3.20; the full database of the objects is shown in Appendix A.

A recognition threshold is specified for each impulse by means of the method explained in Sub-sub-sub-subsection 2.3.4.2.2.1. The objects are processed as unknown images of faces as illustrated in Sub-sub-subsection 2.3.4.2.2. Then Condition 2.5 is used for recognizing the objects. The recognition results of the objects in Figure 3.20 are presented in Table 3.5. The recognition of all 108 objects is shown in Appendix G.

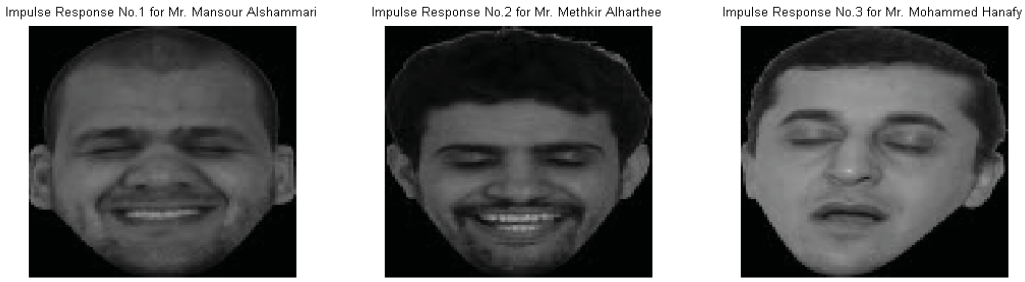

Figure 3.19: The database of the impulses.

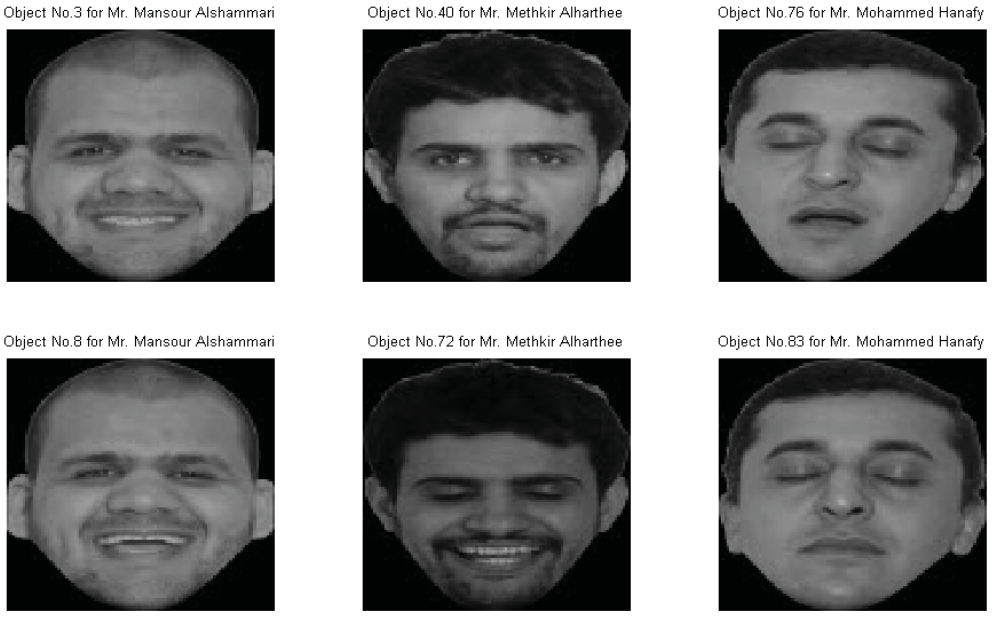

Figure 3.20: Some samples of the objects.

Table 3.5: The recognition of the objects in Figure 3.20.

Object No.	Input Face	Recognized Output Face	Status
3	Mr. Mansour Alshammari	Unknown Object	Failure
8	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
40	Mr. Methkir Alharthee	Unknown Object	Failure
72	Mr. Methkir Alharthee	Unknown Object	Failure
76	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
83	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success

3.2.4 Object Detection

The database of the generated objects in Subsection 3.2.3 is used for detection. A detection threshold is specified by means of the method explained in Sub-sub-subsection 2.3.4.2.3. Then Condition 2.6 is used for detecting the objects. The detection results of the objects in Figure 3.20 are presented in Table 3.6. The detection of all 108 objects is shown in Appendix H.

Table 3.6: The detection of the objects in Figure 3.20.

Object No.	Input Object	Detected Output Object	Status
3	a face	not a face	Failure
8	a face	not a face	Failure
40	a face	a face	Success
72	a face	not a face	Failure
76	a face	a face	Success
83	a face	not a face	Failure

Chapter 4

Results of Performance Analysis

4.1 Performance Results of the PCA and IPCA Algorithms

4.1.1 Introduction

After modelling the applications of the PCA and IPCA algorithms, the performance results for each application are obtained in this subsection. The results show the behavior of each application when the eigenfaces that are generated by using the PCA algorithm are used; the eigenfaces that are generated by using the IPCA algorithm are used; and when different eigenfaces are selected to form the feature vector matrix.

4.1.2 Results of Compression Performance

4.1.2.1 Speed of Compression and Reconstruction

When a small number of the eigenfaces is used to project and reconstruct the training faces then the processing speed will increase and vice versa. Therefore, the IPCA algorithm is the fastest one; then the PCA algorithm comes second; finally, the smallest processing speed occurs when all calculated eigenvectors from the covariance matrix for all training faces are used as eigenfaces.

4.1.2.2 Quality of a Reconstructed Image

When a small number of the eigenfaces is used to project and reconstruct the training faces then the training faces will have bad quality. Therefore, the highest error in reconstruction occurs when the IPCA algorithm is used; then the PCA algorithm

comes second; finally, the usage of all eigenvectors as eigenfaces produces the samllest reconstruction error.

For measuring the quality of the reconstructed training faces, Equation 2.46 is used for computing the mean squared errors (MSEs) between the training faces and their reconstructions. Figure 4.1 shows the plot of the MSEs of reconstructing training face number five for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used. The plots of the MSEs of reconstructing other training faces are shown in Appendix I. Figure 4.2 shows the reconstruction of training face number five by using the highest correlated eigenface ($q = 1$), the PCA eigenfaces ($q = 8$), the IPCA eigenfaces ($q = 7$), and all eigenvectors as eigenfaces ($q = 2500$); along with mean squared errors resulted from reconstructing the training face by using those eigenfaces.

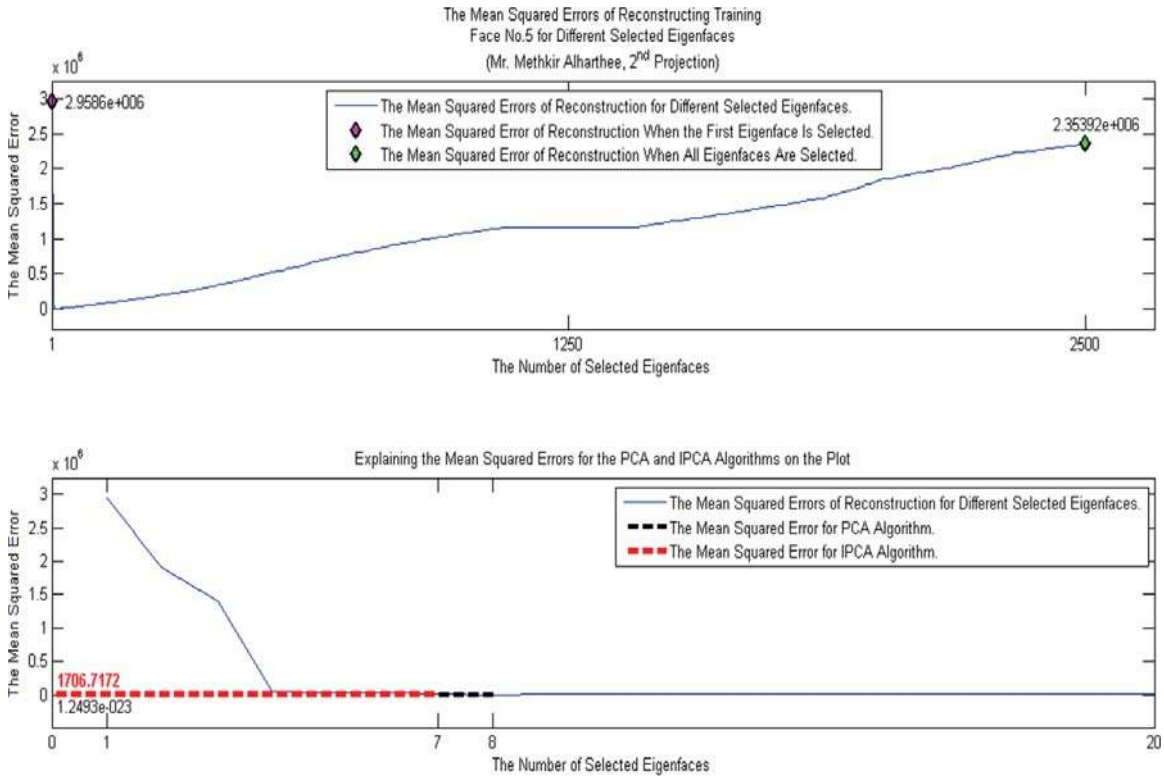

Figure 4.1: The plot of the mean squared errors (MSEs) of reconstructing training face number five for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

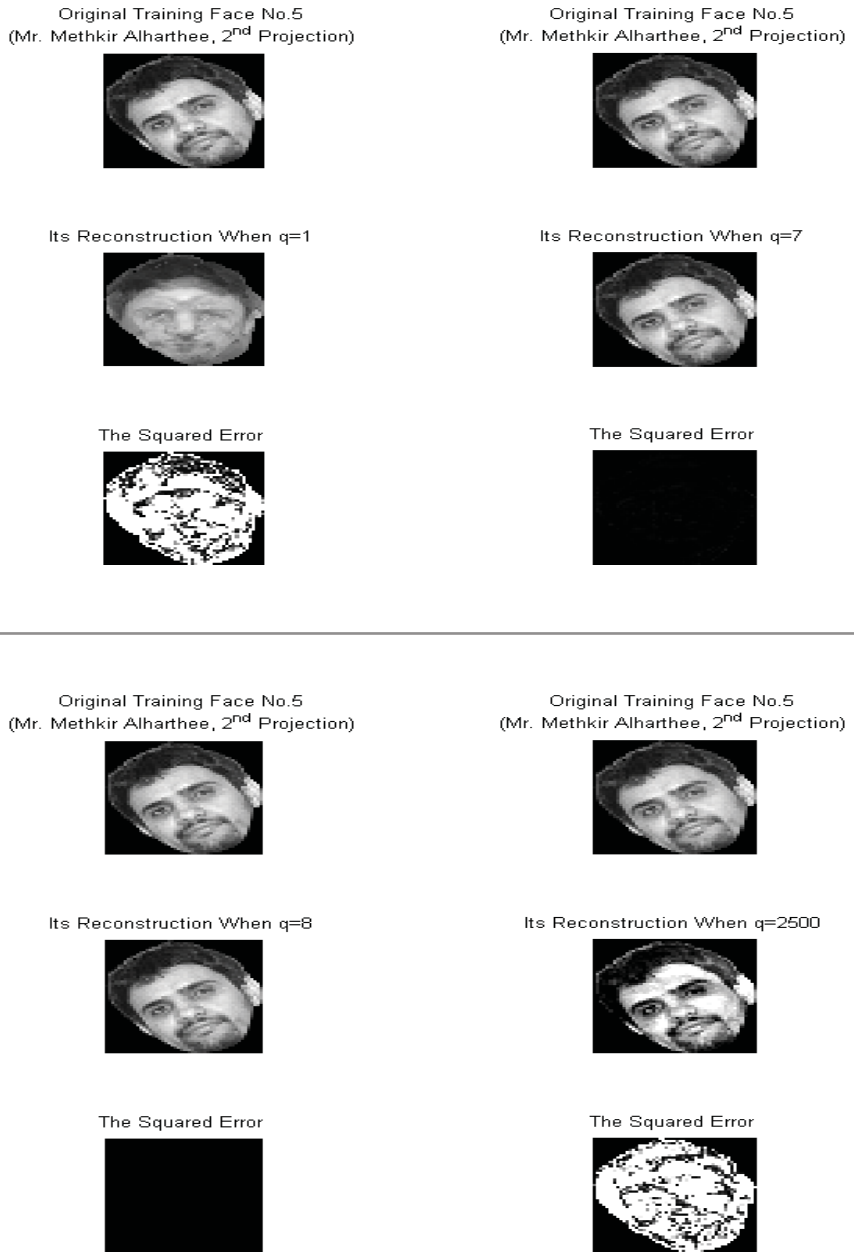

Figure 4.2: The reconstruction of training face number five by using the highest correlated eigenface ($q = 1$), the PCA eigenfaces ($q = 8$), the IPCA eigenfaces ($q = 7$), and all eigenvectors as eigenfaces ($q = 2500$); along with mean squared errors resulted from reconstructing the training face by using those eigenfaces.

From Figure 4.1 and Figure 4.2, it can be noticed that when all eigenvectors

are used as eigenfaces then the reconstructed training faces have the worst resolution; that because the covariance matrix is too big then the calculation of 2500 eigenvectors by using the MATLAB © software leads to some round-off errors in the eigenvectors associated with the smallest eigenvalues. Hence, it is not recommended to select the lowest correlated eigenvectors for reconstructing training faces due to they add some noise to the reconstructed training faces. In addition, due to round-off error, the MATLAB © software makes the smallest eigenvalues negative while they must be positive because they are calculated from a positive definite covariance matrix. Those negative eigenvectors must be set to zero.

4.1.2.3 Size of Compression

4.1.2.3.1 Information Rate

Obviously, from Equation 2.47, an information rate depends on the number of the selected eigenfaces q. Note that, when all eigenvectors are picked to form the feature vector matrix then there will not be any compression; and the overall size when there is no any compression method is used will be the optimum one. Consequently, when the number of the selected eigenfaces decreases then an information rate will decrease (i.e. compression will increase) and vice versa. Therefore, the IPCA algorithm offers the highest compression with the highest loss of information; after that, when the PCA algorithm is used, there will not be information lost (i.e. there is no compression); lastly, the usage of all eigenvectors as eigenfaces add some information (i.e. there is no compression).

For measuring how much information is after compression compared with information before compression, Equation 2.47 is used. Figure 4.3 shows the rates of information for different selected eigenfaces compared with resulted information rates when the PCA and IPCA algorithms are used.

4.1.2.3.2 Mean Squared Error (MSE) of Compressed Images

Lost information increases when a small number of eigenfaces is picked and vice versa. Therefore, the IPCA algorithm offers the highest compression with the highest error; then the PCA algorithm comes second; lastly, the usage of all eigenvectors as eigenfaces produce the lowest compression with the lowest error.

Equation 2.54 is used for computing the mean squared error (MSE) of compression. Figure 4.4 shows the mean squared errors (MSEs) of compression for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

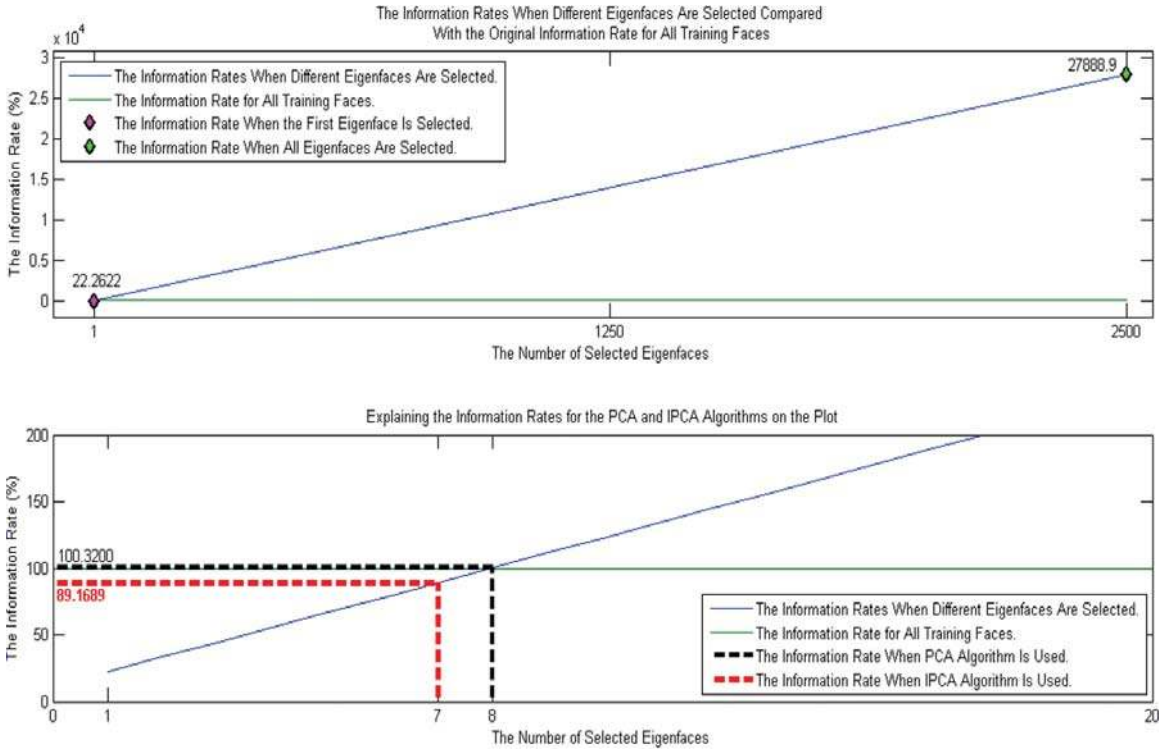

Figure 4.3: The rates of information for different selected eigenfaces compared with resulted information rates when the PCA and IPCA algorithms are used.

4.1.3 Results of Recognition Performance

4.1.3.1 Speed of Recognition

When the number of the selected eigenfaces decreases, the processing speed increases and vice versa. Therefore, the usage of all calculated eigenvectors as eigenfaces leads to the biggest processing time; then the usage of the calculated eigenfaces by using the PCA algorithm comes second; finally, the usage of the calculated eigenfaces by using the IPCA algorithm leads to the smallest processing time.

4.1.3.2 Error Rate

When the number of the selected eigenfaces increases, the error rate decreases; that because the unknown face image will be projected precisely next to its corresponding training face. On the other hand, when the selected eigenfaces decreases, the error rate increases. Therefore, the usage of all calculated eigenvectors as eigenfaces leads to the smallest error rate; then the usage of the calculated eigenfaces by using the PCA algorithm comes second; finally, the usage of the calculated eigenfaces by using

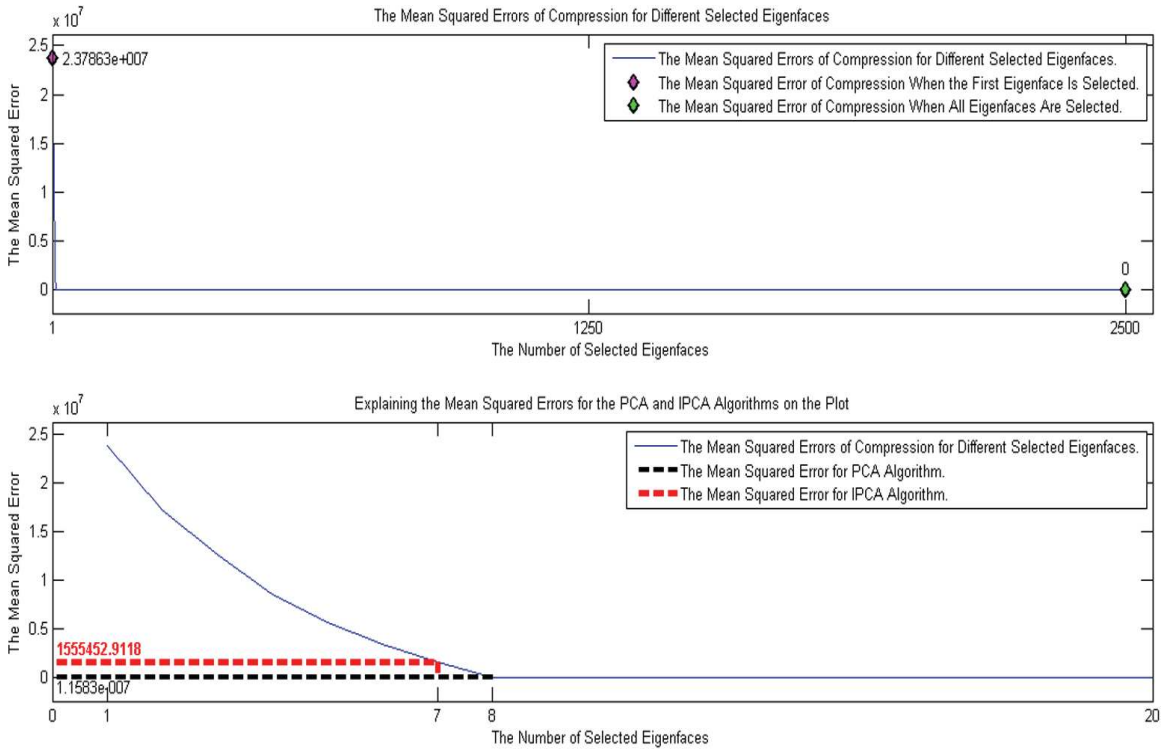

Figure 4.4: The mean squared errors (MSEs) of compression for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

the IPCA algorithm leads to the biggest error rate.

In the recognition of all 180 tested images in Subsection 3.1.6, $L = 180$; $SR = 133$; and $FR = 47$. Then by using Equation 2.55, the error rate $ER\,(\%)$ is equal to $\frac{47}{180} \times 100 = 26.1111\%$. Figure 4.5 shows the error rates of recognition for different selected eigenfaces compared with resulted error rates when the PCA and IPCA algorithms are used.

4.1.4 Results of Detection Performance

4.1.4.1 Speed of Detection

When the number of the selected eigenfaces decreases, the processing speed increases and vice versa. Therefore, the IPCA algorithm is the fastest one; then the PCA algorithm comes second; finally, the smallest processing speed occurs when all calculated eigenvectors from the covariance matrix for all training faces are used as eigenfaces.

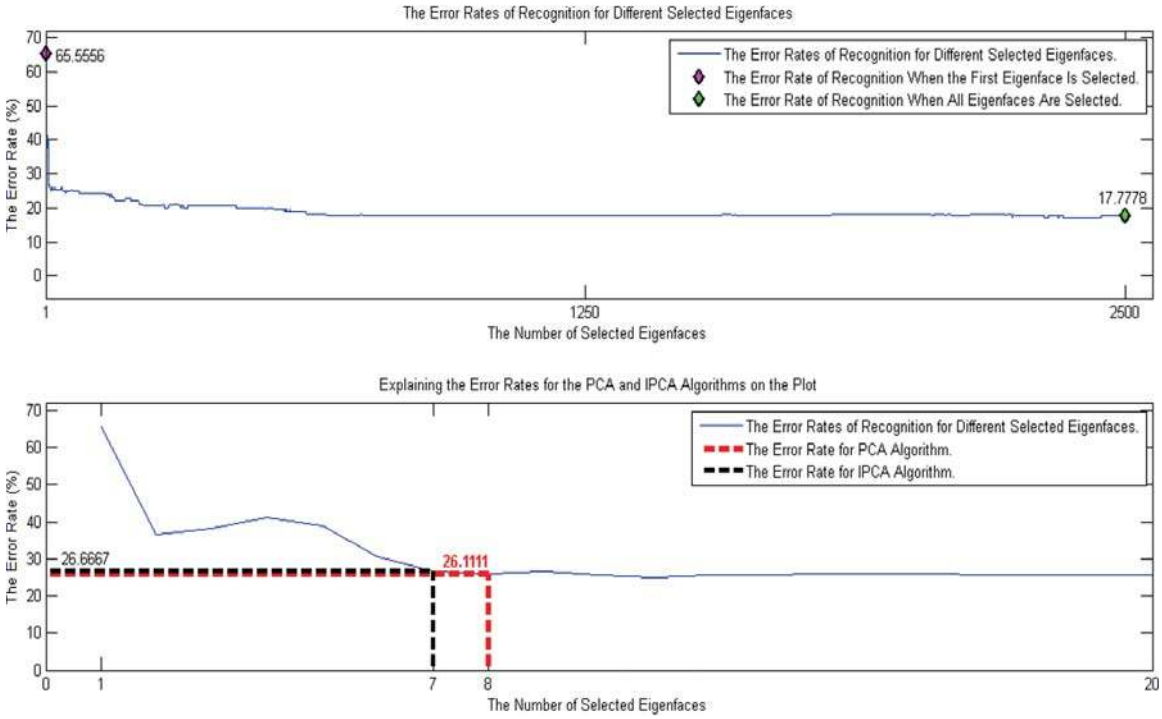

Figure 4.5: The error rates of recognition for different selected eigenfaces compared with resulted error rates when the PCA and IPCA algorithms are used.

4.1.4.2 Error Rate

When the eigenfaces that contain the most significant patterns from the correlated training faces (highly correlated eigenvectors) are only selected then the accuracy of face detection increases. That because in face detection, distance calculation is between the centered unknown image and its reconstruction; consequently, if the unknown image is not a face then the distance will be big; therefore, the unknown image will not be detected as a face image. As a result of that, the usage of the calculated eigenfaces by using the IPCA algorithm produces the smallest error rate; then the usage of the calculated eigenfaces by using the PCA algorithm comes second; finally, the usage of all calculated eigenvectors as eigenfaces obtains the biggest error rate.

In the detection of all 180 tested images in Subsection 3.1.7, $L = 180$; $SD = 147$; and $FD = 33$. Then by using Equation 2.56, the error rate $ER(\%)$ is equal to $\frac{33}{180} \times 100 = 18.3333\%$. Figure 4.6 shows the error rates of detection for different selected eigenfaces compared with resulted error rates when the PCA and IPCA algorithms are used.

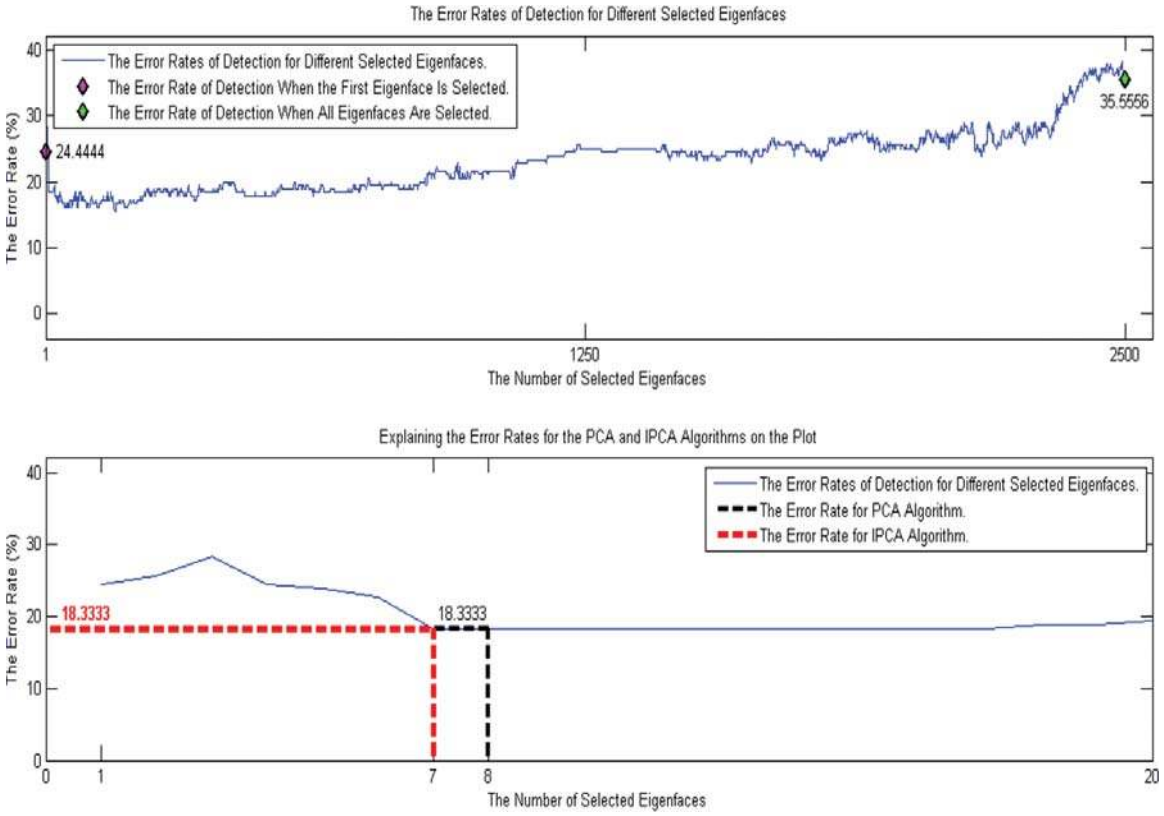

Figure 4.6: The error rates of detection for different selected eigenfaces compared with resulted error rates when the PCA and IPCA algorithms are used.

4.2 Results of JTC Performance

4.2.1 Results of Recognition Performance

In the recognition of all 108 objects in Subsection 3.2.3, $L = 108$; $SR = 20$; and $FR = 88$. Then by using Equation 2.85, the error rate $ER\,(\%)$ is equal to $\frac{88}{108} \times 100 = 81.4815\%$.

4.2.1.1 Improvement of Recognition Performance

For finding the optimal combination out of the 108 objects in Subsection 3.2.3, 46656 $(36 \times 36 \times 36)$ iterations are performed until the database of the optimal impulses for face recognition is obtained as in Figure 4.7. The complete MATLAB © code that had been written to optimize the database of the impulses for face recognition is shown in Appendix J. When the database of the optimal impulses is used for recognizing all 108 objects then the total number of successes SR becomes 71;

and the total number of failures FR becomes 37. Therefore, the error rate $ER\,(\%)$ is equal to $\frac{37}{108} \times 100 = 34.2593\%$ which is much less than the resulted error rate when the database of the impulses is not optimized.

Figure 4.7: The database of the optimal impulses for face recognition.

4.2.2 Results of Detection Performance

In the detection of all 108 objects in Subsection 3.2.4, $L = 108$; $SD = 58$; and $FD = 50$. Then by using Equation 2.86, the error rate $ER\,(\%)$ is equal to $\frac{50}{108} \times 100 = 46.2963\%$.

4.2.2.1 Improvement of Detection Performance

For finding the optimal combination out of the 108 objects in Subsection 3.2.4, 46656 $(36 \times 36 \times 36)$ iterations are performed until the database of the optimal impulses for object detection is obtained as in Figure 4.8. The complete MATLAB © code that had been written to optimize the database of the impulses for object detection is shown in Appendix J. When the database of the optimal impulses is used for detecting all 108 objects then the total number of successes SD becomes 79; and the total number of failures FD becomes 29. Therefore, the error rate $ER\,(\%)$ is equal to $\frac{29}{108} \times 100 = 26.8519\%$ which is much less than the resulted error rate when the database of the impulses is not optimized.

Figure 4.8: The database of the optimal impulses for object detection.

Chapter 5

Conclusion

5.1 Discussion of Results

5.1.1 Introduction

In fact, the results of the joint transform correlator (JTC) applications are obviously shown in Section 4.2; but the results of the PCA and IPCA applications are not summarized yet. Hence, in this section, these results are fully discussed and summarized.

5.1.2 Comparison of the PCA and IPCA Algorithms

5.1.2.1 Introduction

In this sub-subsection, it is determined which algorithm behaves better in each application. It concludes that the IPCA algorithm, in general, behaves better than the PCA algorithm in the most of the applications.

It is very important to be noticed that the calculation of all eigenvectors from the covariance matrix for all training faces is too difficult because the covariance matrix is too big as explained in Step 8 in Sub-subsection 2.2.2.6. Therefore, it is impractical to use all eigenvectors as eigenfaces; but they are computed for comparison purposes with the PCA and IPCA algorithms.

5.1.2.2 Results of Image Compression

The results of image compression are summarized in Table 5.1. From Table 5.1, the IPCA algorithm behaves better than any other algorithm or technique. It offers wonderful compression, reconstruction and processing speed with acceptable errors.

Table 5.1: The results of image compression.

		The Mean Squared Error (MSE) of Reconstructing Training Face No. 5	An information rate (%) Before Compression : After Compression	The MSE of Compression	Processing Speed
The Number of Selected Eigenfaces q	$q = 1$	2.96×10^6	100 : 22.26	2.38×10^7	The fastest
	$q = 7$ (IPCA Algorithm)	1706.72	100 : 89.17	1.56×10^6	Second
	$q = 8$ (PCA Algorithm)	1.25×10^{-23}	100 : 100.32	1.16×10^{-7}	Third
	$q = 2500$ (All Eigenvectors)	2.35×10^6	100 : 27888.9	0	The slowest

5.1.2.3 Results of Face Recognition

The results of face recognition are summarized in Table 5.2. From Table 5.2, the PCA algorithm behaves better than any other algorithm or technique. It offers an acceptable error rate, easy calculation and the speed is not bad.

Table 5.2: The results of face recognition.

		An Error Rate (%)	Processing Speed
The Number of Selected Eigenfaces q	$q = 1$	65.56	The fastest
	$q = 7$ (IPCA Algorithm)	26.67	Second
	$q = 8$ (PCA Algorithm)	26.11	Third
	$q = 2500$ (All Eigenvectors)	17.78	The slowest

5.1.2.4 Results of Face Detection

The results of face detection are summarized in Table 5.3. From Table 5.3, the IPCA algorithm behaves better than any other algorithm or technique. It offers the smallest error rate as well as remarkable speed.

Table 5.3: The results of face detection.

		An Error Rate (%)	**Processing Speed**
The Number of Selected Eigenfaces q	**$q = 1$**	24.44	The fastest
	$q = 7$ (IPCA Algorithm)	18.33	Second
	$q = 8$ (PCA Algorithm)	18.33	Third
	$q = 2500$ (All Eigenvectors)	35.56	The slowest

5.2 Methods to Improve the Digital and the Optical Models

5.2.1 Introduction

In fact, the discussed models are not in their final stage where they can be optimized. Some ideas are presented in this section for each model that are going to help in enhancing their performance.

5.2.2 Improvement of the Digital and the Optical Models

The performance of face recognition and image detection of the digital model can be improved by increasing the size of the training faces and the detected or recognized unknown image. Also, increasing the size of the impulses and the detected or recognized unknown object of the optical model improves its performance in detection and recognition.

The performance of digital and optical recognition can be improved by obtaining a good way for blocking the pixels occupying the background of a face especially if the background is not black. If it is not black such as white, the error rate of recognition will increase because the intensities on the face will be close to 255 (i.e. they will be close to the intensities on the background) then discrimination between the intensities occupying the background and the intensities occupying the face becomes too hard.

For decreasing the error rates of face recognition and image detection of the digital model, the database of the training faces can be optimized in the same manner as the optimization of the database of the impulses for the optical model that is presented in Sub-sub-sub-subsection 2.3.4.3.2.1 and Sub-sub-sub-subsection 2.3.4.3.3.1.

The models performance can be improved by using another technique for enhancing the optimization speed of the databases. This technique measures information contained in the 180 tested images (or in the 108 objects) then picking the tested images (or the objects) that contain the highest information than the others to form the database of the training faces (or the database of the impulses).

Finally, the models performance can be improved by trying different detection and recognition thresholds such as thresholds generated by receiver operating characteristics (ROC).

5.3 The Digital Model Versus the Optical Model

In this section, we are going to compare between the digital model and the optical model in detection and recognition based on a couple of criteria. The comparison is summarized in Table 5.4.

Table 5.4: The comparison between the digital and the optical models.

A Comparison Criterion	The Digital Model	The Optical Model
The Database	It is not necessarily to be optimized	It must be optimized
Implementation	Easier	Harder
Speed	Slower	Faster because it uses the speed of light
Detection and Recognition Performance	Better	Good

5.4 Future Work

For developing this work in future, the proposed ideas for improving the digital and the optical models presented in Section 5.2 are going to be achieved. Also, there is another idea that is considered to be performed in future is testing the models performance under various types of noises.

Bibliography

[1] L. I. Smith, "A tutorial on principal components analysis." http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf, February 2002. 1

[2] S.-F. Ding, Z.-Z. Shi, Y. Liang, and F.-X. Jin, "Information feature analysis and improved algorithm of pca," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 3, pp. 1756–1761 Vol. 3, Aug 2005. doi: 10.1109/ICMLC.2005.1527229. 1, 2, 9, 20

[3] J. Goodman, *Introduction to Fourier Optics*. McGraw-Hill physical and quantum electronics series, Roberts & Company, 2005. http://books.google.com/books?id=ow5xs_Rtt9AC, lccn: 2004023213, isbn: 9780974707723. 2, 30, 32

[4] C. S. Weaver and J. W. Goodman, "A technique for optically convolving two functions," *Appl. Opt.*, vol. 5, pp. 1248–1249, Jul 1966. publisher: OSA, http://ao.osa.org/abstract.cfm?URI=ao-5-7-1248, number: 7, doi: 10.1364/AO.5.001248. 2, 35

[5] *Matrix Methods and Vector Spaces in Physics*. Prentice-Hall Of India Pvt. Limited. http://books.google.com/books?id=jF-08gk-BcsC, isbn: 9788120338661. 9

[6] J. Kwak and S. Hong, *Linear Algebra*. Birkhäuser Boston, 2004. http://books.google.com/books?id=27zJ7zJCiAIC, isbn: 9780817642945, lccn: 2004043751. 9, 16

[7] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pp. 586–591, Jun 1991. doi: 10.1109/CVPR.1991.139758, ISSN: 1063-6919. 17

[8] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, pp. 71–86, Jan. 1991. publisher: MIT Press, address: Cambridge, MA, USA, http://dx.doi.org/10.1162/jocn.1991.3.1.71, number: 1, numpages: 16, acmid: 1326894, doi: 10.1162/jocn.1991.3.1.71, issn: 0898-929X. 17

[9] R. Gonzalez, R. Woods, and S. Eddins, *Digital Image Processing Using MATLAB*. Gatesmark, LLC, 2009. http://www.imageprocessingplace.com/DIPUM-2E/dipum2e_main_page.htm, isbn: 978-0-9820854-0-0. 17

[10] M. Fiedler, *Matrices and Graphs in Geometry*. Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2011. http://books.google.com/books?id=otbzeAqL63kC, lccn: 2010046601, isbn: 9780521461931. 18

[11] B. Buttkus, *Spectral Analysis and Filter Theory in Applied Geophysics: With 23 Tables*. Springer Berlin Heidelberg, 2000. http://books.google.com/books?id=n6a6BjoGzwwC, lccn: 00026538, isbn: 9783540626749. 35

[12] S. Haykin and M. Moher, "Properties of the fourier transform," in *An Introduction to Analog and Digital Communications, 2nd Edition*, Wiley Global Education, 2006. http://books.google.com/books?id=2RUcAAAAQBAJ, isbn: 9781118313107. 37

[13] A. Kumar, *Signals and Systems*. PHI Learning Pvt. Ltd. http://books.google.com/books?id=FGGa6BXhy3kC, isbn: 9788120343108. 38

[14] R. Gonzalez and R. Woods, *Digital Image Processing*. Pearson/Prentice Hall, 2008. http://books.google.com/books?id=8uGOnjRGEzoC, lccn: 2009289249, isbn: 9780131687288. 42

Appendix A

A Code for the Digital Model

This code is for testing face reconstruction, detection and recognition processes as well as the process of image compression by using principal components analysis (PCA) and improved principal components analysis (IPCA) algorithms. In addition to that this code is for setting up recognition and detection thresholds.

```
1
2  % This Code Is for Testing Face Reconstruction, Detection and
3  % Recognition Processes as Well as the Process of Image Compression
4  % by Using Principal Components Analysis (PCA) and Improved
5  % Principal Components Analysis (IPCA) Algorithms. In Addition to
6  % That This Code Is for Setting up Recognition and Detection
7  % Thresholds.
8
9
10 clc
11 clear all
12 close all
13 format long
14
15
16 % Faces images are NxN images.
17 N=size(imread('Mr. Mansour Alshammari.jpg'),1); % This N is the
18                                                 % number of pixels.
19
20
21 % Training faces.
22 Total_No_of_Known_Im=9; % The total number of the training faces.
23
24 All_Known_Im_V=zeros(N*N,Total_No_of_Known_Im); % An N^2xP, 2D
25                                                 % matrix where P is
26                                                 % the total number
```

```
                                                    % of the training
                                                    % faces. Each
                                                    % training face is
                                                    % vectorized and
                                                    % placed in one of
                                                    % the columns of
                                                    % the 2D matrix.
                                                    % The size of each
                                                    % training face
                                                    % vector is N^2.

Known_Images_Folder=..............
    [cd '/The Known Images of Black Backgrounds']; % The folder of
                                                     % the black
                                                     % background
                                                     % training
                                                     % faces.
% Known_Images_Folder=.........
%     [cd '/The Known Images of White Backgrounds']; % The folder
                                                       % of the white
                                                       % background
                                                       % training
                                                       % faces.

if isdir(Known_Images_Folder)==0
    Error_Message=sprintf(...................
        'Error: The following folder does not exist\n%s', .......
        Known_Images_Folder);
    warndlg(Error_Message);
end

Known_Images=dir(fullfile(Known_Images_Folder,'*.jpg'));
for k=1:length(Known_Images)
    Known_Image=Known_Images(k).name;
    Known_Image_Location=fullfile(Known_Images_Folder,Known_Image);
    All_Known_Im_V(:,k)=.................
        reshape(double(rgb2gray(imread(Known_Image_Location))).....
        ,N*N,1);

%     figure('units','centimeters','position',[16 7 7.5 8.5])
%     subplot(1,1,1)
%     imshow(uint8(reshape(All_Known_Im_V(:,k),N,N)))
%     if k==1
%         title({['Training Face No.' num2str(k)];.............
%             ['(' Known_Image(1:length(Known_Image)-6) .........
%             ', 1^{st} Projection)']})
%     elseif k==2
%         title({['Training Face No.' num2str(k)];.............
%             ['(' Known_Image(1:length(Known_Image)-6) ........
%             ', 2^{nd} Projection)']})
```

```
77 %     elseif k==3
78 %         title({['Training Face No.' num2str(k)];.............
79 %             ['(' Known_Image(1:length(Known_Image)-6) ........
80 %             ', 3^{rd} Projection)']})
81 %     elseif k==4
82 %         title({['Training Face No.' num2str(k)];.............
83 %             ['(' Known_Image(1:length(Known_Image)-6) ......
84 %             ', 1^{st} Projection)']})
85 %     elseif k==5
86 %         title({['Training Face No.' num2str(k)];.............
87 %             ['(' Known_Image(1:length(Known_Image)-6) ......
88 %             ', 2^{nd} Projection)']})
89 %     elseif k==6
90 %         title({['Training Face No.' num2str(k)];.............
91 %             ['(' Known_Image(1:length(Known_Image)-6) .....
92 %             ', 3^{rd} Projection)']})
93 %     elseif k==7
94 %         title({['Training Face No.' num2str(k)];.............
95 %             ['(' Known_Image(1:length(Known_Image)-6) ......
96 %             ', 1^{st} Projection)']})
97 %     elseif k==8
98 %         title({['Training Face No.' num2str(k)];.............
99 %             ['(' Known_Image(1:length(Known_Image)-6) .....
100 %            ', 2^{nd} Projection)']})
101 %    elseif k==9
102 %        title({['Training Face No.' num2str(k)];.............
103 %            ['(' Known_Image(1:length(Known_Image)-6) ......
104 %            ', 3^{rd} Projection)']})
105 %    end
106 %    disp(['Please, press any keyboard button to explore '......
107 %        'the remaining training faces >>>>>>>'])
108 %    pause
109 %    close all
110 %    clc
111 end
112 All_Known_Im_V;
113
114 % Imhist for setting up a threshold to work on just the pixels of a
115 % face and throwing the background pixels. Imhist calculates the
116 % number of pixels in an image that have the same intensity levels.
117 % So, if a training face has a unified background then the biggest
118 % histogram of the intensity levels will be for the backgroubd
119 % pixels because the total number of pixels that have the same
120 % intensity levels are the background pixels of the training face.
121 % Note that, the histogram of a digital image is defined as the
122 % discrete function, h(rk)=nk, where rk is the kth intensity level
123 % and nk is the number of pixels in the image whose intensity level
124 % is rk.
125 hist_Known_Im=zeros(Total_No_of_Known_Im,256);
126 for A=1:Total_No_of_Known_Im
```

```
127     % Note that, a training face has to be scaled between 0 to 255
128     % before using imhist. For doing that, uint8 can be used for
129     % converting the training face class form double to uint8.
130     hist_Known_Im(A,:)=.............
131         imhist(uint8(reshape(All_Known_Im_V(:,A),N,N)));
132
133 %     Known_Image=Known_Images(A).name;
134 %     plot(hist_Known_Im(A,:))
135 %     if A==1
136 %         title({['The Histogram of Training Face No.' .......
137 %             num2str(A)];.......
138 %             ['(' Known_Image(1:length(Known_Image)-6) .........
139 %             ', 1^{st} Projection)']})
140 %     elseif A==2
141 %         title({['The Histogram of Training Face No.' ......
142 %             num2str(A)];......
143 %             ['(' Known_Image(1:length(Known_Image)-6) ........
144 %             ', 2^{nd} Projection)']})
145 %     elseif A==3
146 %         title({['The Histogram of Training Face No.' ......
147 %             num2str(A)];......
148 %             ['(' Known_Image(1:length(Known_Image)-6) ........
149 %             ', 3^{rd} Projection)']})
150 %     elseif A==4
151 %         title({['The Histogram of Training Face No.' .....
152 %             num2str(A)];......
153 %             ['(' Known_Image(1:length(Known_Image)-6) ......
154 %             ', 1^{st} Projection)']})
155 %     elseif A==5
156 %         title({['The Histogram of Training Face No.' .......
157 %             num2str(A)];......
158 %             ['(' Known_Image(1:length(Known_Image)-6) ......
159 %             ', 2^{nd} Projection)']})
160 %     elseif A==6
161 %         title({['The Histogram of Training Face No.' ......
162 %             num2str(A)];....
163 %             ['(' Known_Image(1:length(Known_Image)-6) .....
164 %             ', 3^{rd} Projection)']})
165 %     elseif A==7
166 %         title({['The Histogram of Training Face No.' .....
167 %             num2str(A)];.....
168 %             ['(' Known_Image(1:length(Known_Image)-6) ......
169 %             ', 1^{st} Projection)']})
170 %     elseif A==8
171 %         title({['The Histogram of Training Face No.' .....
172 %             num2str(A)];.....
173 %             ['(' Known_Image(1:length(Known_Image)-6) .....
174 %             ', 2^{nd} Projection)']})
175 %     elseif A==9
176 %         title({['The Histogram of Training Face No.' ....
```

```
177 %             num2str(A)];......
178 %             ['(' Known_Image(1:length(Known_Image)-6) ......
179 %             ', 3^{rd} Projection)']})
180 %     end
181 %     xlabel('Intensity Level rk')
182 %     ylabel({'The Number of Pixels in the Training Face'.........
183 %         ' Whose Intensity Level Is rk Where h(rk)=nk'})
184 %     axis tight
185 %     disp(['Please, press any keyboard button to explore '......
186 %         'the remaining histograms >>>>>>>'])
187 %     pause
188 %     clc
189 end
190 hist_Known_Im;
191
192 Mean_hist_Known_Im=sum(hist_Known_Im,1)/......
193     Total_No_of_Known_Im; % The average histogram
194                           % for all training faces.
195 Threshold_Known_Im=8; % The picked threshold is based on the
196                       % average histogram for all training faces
197                       % when the training faces have black
198                       % backgrounds. Note that, all intesity levels
199                       % below the threshold represent the images
200                       % backgrounds because these levels have the
201                       % biggest histogram.
202 % Threshold_Known_Im=180; % The Picked threshold is based on the
203                           % average histogram for all training
204                           % faces when the training faces have
205                           % white backgrounds. Note that, all
206                           % intesity levels above the threshold
207                           % represent the images backgrounds
208                           % because these levels have the biggest
209                           % histogram.
210 % plot(Mean_hist_Known_Im)
211 % line([Threshold_Known_Im Threshold_Known_Im],......
212 %     [0 max(Mean_hist_Known_Im)],'Color','r')
213 % text(Threshold_Known_Im+0.5,max(Mean_hist_Known_Im)/2,.....
214 %     '{\color{red} The Threshold}')
215 % title('The Mean Histogram of All Training Faces')
216 % xlabel('Intensity Level rk')
217 % set(gca,'XTick',[0 Threshold_Known_Im 255])
218 % ylabel({'The Mean Number of Pixels from All Training' .........
219 %     'Faces Whose Intensity Level Is rk'})
220 % axis tight
221 % pause
222
223 % Normalizing all training faces for removing the lightening
224 % effects on them and to increase the resolution of face detection
225 % and recognition. Note that, the normalization will be done just
226 % for face pixels for keeping the variations among the images just
```

```
% in the faces without the backgrounds effects.
Threshold_Known_Image=.......
    zeros(N,N,Total_No_of_Known_Im); % The training faces after
                                     % applying the threshold.
Normalized_Known_Im_V=......
    zeros(N*N,Total_No_of_Known_Im); % An N^2xP, 2D matrix where
                                     % each column represents a
                                     % normalized training face
                                     % vector.
for M=1:Total_No_of_Known_Im
    t=reshape(All_Known_Im_V(:,M),N,N);
    T=t>Threshold_Known_Im; % The pixels bigger than the threshold
                            % are of interest because they
                            % represent the pixels of a face.
%     T=t<Threshold_Known_Im; % The pixels smaller than the
%                             % threshold are of interest because
%                             % they represent the pixels of
%                             % a face.
    for R=1:N
        for C=1:N
            if T(R,C)==1
                Threshold_Known_Image(R,C,M)=.......
                    floor(255*(double(t(R,C))/........
                    max(max(double(t))))); % The normalization of a
                                           % training face. This is
                                           % done to increase the
                                           % dynamic range of the
                                           % training face for
                                           % visualization by
                                           % scaling the
                                           % intensities of the
                                           % training face from 0
                                           % to 255.
            end;
        end;
    end;
    Normalized_Known_Im_V(:,M)=.........
        reshape(Threshold_Known_Image(:,:,M),N*N,1);

%     Known_Image=Known_Images(M).name;
%     figure
%     subplot(2,1,1)
%     imshow(t)
%     if M==1
%         title({['This Is To Show How Good the Known '.......
%             'Images Threshold Is,'];.....
%             blanks(1);['Training Face No.' num2str(M)];
%             ['(' Known_Image(1:length(Known_Image)-6) ......
%             ', 1^{st} Projection)']})
%     elseif M==2
```

```
277 %         title({['This Is To Show How Good the Known '......
278 %             'Images Threshold Is,'];.....
279 %             blanks(1);['Training Face No.' num2str(M)];
280 %             ['(' Known_Image(1:length(Known_Image)-6) .....
281 %             ', 2^{nd} Projection)']})
282 %     elseif M==3
283 %         title({['This Is To Show How Good the Known '......
284 %             'Images Threshold Is,'];.....
285 %             blanks(1);['Training Face No.' num2str(M)];
286 %             ['(' Known_Image(1:length(Known_Image)-6) .....
287 %             ', 3^{rd} Projection)']})
288 %     elseif M==4
289 %         title({['This Is To Show How Good the Known '......
290 %             'Images Threshold Is,'];.....
291 %             blanks(1);['Training Face No.' num2str(M)];
292 %             ['(' Known_Image(1:length(Known_Image)-6) ....
293 %             ', 1^{st} Projection)']})
294 %     elseif M==5
295 %         title({['This Is To Show How Good the Known '.....
296 %             'Images Threshold Is,'];.....
297 %             blanks(1);['Training Face No.' num2str(M)];
298 %             ['(' Known_Image(1:length(Known_Image)-6) .....
299 %             ', 2^{nd} Projection)']})
300 %     elseif M==6
301 %         title({['This Is To Show How Good the Known '.....
302 %             'Images Threshold Is,'];.....
303 %             blanks(1);['Training Face No.' num2str(M)];
304 %             ['(' Known_Image(1:length(Known_Image)-6) ......
305 %             ', 3^{rd} Projection)']})
306 %     elseif M==7
307 %         title({['This Is To Show How Good the Known '......
308 %             'Images Threshold Is,'];.....
309 %             blanks(1);['Training Face No.' num2str(M)];
310 %             ['(' Known_Image(1:length(Known_Image)-6) .....
311 %             ', 1^{st} Projection)']})
312 %     elseif M==8
313 %         title({['This Is To Show How Good the Known '.....
314 %             'Images Threshold Is,'];.....
315 %             blanks(1);['Training Face No.' num2str(M)];
316 %             ['(' Known_Image(1:length(Known_Image)-6) ....
317 %             ', 2^{nd} Projection)']})
318 %     elseif M==9
319 %         title({['This Is To Show How Good the Known '......
320 %             'Images Threshold Is,'];.....
321 %             blanks(1);['Training Face No.' num2str(M)];
322 %             ['(' Known_Image(1:length(Known_Image)-6) .....
323 %             ', 3^{rd} Projection)']})
324 %     end
325 %     subplot(2,1,2)
326 %     imshow(reshape(Normalized_Known_Im_V(:,M),N,N))
```

```
%     if M==1
%         title({['Normalized Training Face No.' num2str(M)];....
%             ['(' Known_Image(1:length(Known_Image)-6) .......
%             ', 1^{st} Projection)']})
%     elseif M==2
%         title({['Normalized Training Face No.' num2str(M)];....
%             ['(' Known_Image(1:length(Known_Image)-6) ......
%             ', 2^{nd} Projection)']})
%     elseif M==3
%         title({['Normalized Training Face No.' num2str(M)];....
%             ['(' Known_Image(1:length(Known_Image)-6) ......
%             ', 3^{rd} Projection)']})
%     elseif M==4
%         title({['Normalized Training Face No.' num2str(M)];....
%             ['(' Known_Image(1:length(Known_Image)-6) ......
%             ', 1^{st} Projection)']})
%     elseif M==5
%         title({['Normalized Training Face No.' num2str(M)];....
%             ['(' Known_Image(1:length(Known_Image)-6) ......
%             ', 2^{nd} Projection)']})
%     elseif M==6
%         title({['Normalized Training Face No.' num2str(M)];....
%             ['(' Known_Image(1:length(Known_Image)-6) .....
%             ', 3^{rd} Projection)']})
%     elseif M==7
%         title({['Normalized Training Face No.' num2str(M)];....
%             ['(' Known_Image(1:length(Known_Image)-6) ......
%             ', 1^{st} Projection)']})
%     elseif M==8
%         title({['Normalized Training Face No.' num2str(M)];....
%             ['(' Known_Image(1:length(Known_Image)-6) .......
%             ', 2^{nd} Projection)']})
%     elseif M==9
%         title({['Normalized Training Face No.' num2str(M)];....
%             ['(' Known_Image(1:length(Known_Image)-6) .......
%             ', 3^{rd} Projection)']})
%     end
%     disp(['Please, press any keyboard button to see how '.....
%         'good the applied'])
%     disp('threshold on the normalized training faces is >>>>>>>')
%     pause
%     close all
%     clc
%
%     figure
%     subplot(2,1,1)
%     imshow(uint8(t))
%     if M==1
%         title({['Training Face No.' num2str(M)];...........
%             ['(' Known_Image(1:length(Known_Image)-6) ......
```

```
377 %              ', 1^{st} Projection)']})
378 %      elseif M==2
379 %          title({['Training Face No.' num2str(M)];.......
380 %              ['(' Known_Image(1:length(Known_Image)-6) .....
381 %              ', 2^{nd} Projection)']})
382 %      elseif M==3
383 %          title({['Training Face No.' num2str(M)];.......
384 %              ['(' Known_Image(1:length(Known_Image)-6) ......
385 %              ', 3^{rd} Projection)']})
386 %      elseif M==4
387 %          title({['Training Face No.' num2str(M)];..........
388 %              ['(' Known_Image(1:length(Known_Image)-6) ......
389 %              ', 1^{st} Projection)']})
390 %      elseif M==5
391 %          title({['Training Face No.' num2str(M)];........
392 %              ['(' Known_Image(1:length(Known_Image)-6) ......
393 %              ', 2^{nd} Projection)']})
394 %      elseif M==6
395 %          title({['Training Face No.' num2str(M)];......
396 %              ['(' Known_Image(1:length(Known_Image)-6) ......
397 %              ', 3^{rd} Projection)']})
398 %      elseif M==7
399 %          title({['Training Face No.' num2str(M)];......
400 %              ['(' Known_Image(1:length(Known_Image)-6) .......
401 %              ', 1^{st} Projection)']})
402 %      elseif M==8
403 %          title({['Training Face No.' num2str(M)];......
404 %              ['(' Known_Image(1:length(Known_Image)-6) ......
405 %              ', 2^{nd} Projection)']})
406 %      elseif M==9
407 %          title({['Training Face No.' num2str(M)];......
408 %              ['(' Known_Image(1:length(Known_Image)-6) .......
409 %              ', 3^{rd} Projection)']})
410 %      end
411 %      subplot(2,1,2)
412 %      imshow(uint8(reshape(Normalized_Known_Im_V(:,M),N,N)))
413 %      if M==1
414 %          title({['Normalized Training Face No.' num2str(M)];....
415 %              ['(' Known_Image(1:length(Known_Image)-6) ......
416 %              ', 1^{st} Projection)']})
417 %      elseif M==2
418 %          title({['Normalized Training Face No.' num2str(M)];....
419 %              ['(' Known_Image(1:length(Known_Image)-6) ......
420 %              ', 2^{nd} Projection)']})
421 %      elseif M==3
422 %          title({['Normalized Training Face No.' num2str(M)];....
423 %              ['(' Known_Image(1:length(Known_Image)-6) .......
424 %              ', 3^{rd} Projection)']})
425 %      elseif M==4
426 %          title({['Normalized Training Face No.' num2str(M)];....
```

```
427 %             ['(' Known_Image(1:length(Known_Image)-6) .....
428 %             ', 1^{st} Projection)']})
429 %     elseif M==5
430 %         title({['Normalized Training Face No.' num2str(M)];....
431 %             ['(' Known_Image(1:length(Known_Image)-6) ......
432 %             ', 2^{nd} Projection)']})
433 %     elseif M==6
434 %         title({['Normalized Training Face No.' num2str(M)];....
435 %             ['(' Known_Image(1:length(Known_Image)-6) .......
436 %             ', 3^{rd} Projection)']})
437 %     elseif M==7
438 %         title({['Normalized Training Face No.' num2str(M)];....
439 %             ['(' Known_Image(1:length(Known_Image)-6) ......
440 %             ', 1^{st} Projection)']})
441 %     elseif M==8
442 %         title({['Normalized Training Face No.' num2str(M)];....
443 %             ['(' Known_Image(1:length(Known_Image)-6) .....
444 %             ', 2^{nd} Projection)']})
445 %     elseif M==9
446 %         title({['Normalized Training Face No.' num2str(M)];....
447 %             ['(' Known_Image(1:length(Known_Image)-6) .......
448 %             ', 3^{rd} Projection)']})
449 %     end
450 %     disp(['Please, press any keyboard button to explore '......
451 %         'the remaining normalized training faces >>>>>>>'])
452 %     pause
453 %     close all
454 %     clc
455 end;
456 Normalized_Known_Im_V;
457
458 % Centering each face by simply subtracting the mean of the face
459 % pixels from each pixel in the face. By doing that the new face
460 % pixels will have zero mean that means the face is centered.
461 Rows_Columns=zeros(1,1,...........
462     Total_No_of_Known_Im); % The rows and columns for the pixels
463                            % of the faces. Note that, the training
464                            % faces are not similar so the rows and
465                            % columns of the faces pixels will not
466                            % be equal. Therefore, MATLAB will add
467                            % zero rows and columns to make the
468                            % matrices of the rows and columns of
469                            % the faces pixels are equal.
470 Means=zeros(1,Total_No_of_Known_Im); % The means of the
471                                      % faces pixels.
472 Centered_Known_Im=zeros(N*N,.................
473     Total_No_of_Known_Im); % An N^2xP, 2D matrix where each
474                            % column represents a training
475                            % face vector with a centered
476                            % face.
```

```
Centered_Known_Image=zeros(N,N,........
    Total_No_of_Known_Im); % The training faces after
                           % centering the faces.
for j=1:Total_No_of_Known_Im
    x=reshape(Normalized_Known_Im_V(:,j),N,N);
    [rr cc]=find(x>0); % The pixels bigger than zero are of
                       % interest because they represent the
                       % pixels of a face.
    Rows_Columns(1,1:size(rr,1),j)=rr.';
    Rows_Columns(2,1:size(rr,1),j)=cc.';
    Sum=0;
    No=0;
    for RR=1:size(rr,1)
        Sum=Sum+x(rr(RR),cc(RR));
        No=No+1;
    end
    Means(1,j)=Sum/No;

    for RR1=1:size(rr,1)
        Centered_Known_Image(rr(RR1),cc(RR1),j)=........
            x(rr(RR1),cc(RR1))-Means(1,j);
    end
    Centered_Known_Im(:,j)=.........
        reshape(Centered_Known_Image(:,:,j),N*N,1);

%     figure('units','centimeters','position',[16 7 7 8.5])
%     subplot(1,1,1)
%     Known_Image=Known_Images(j).name;
%     imshow(uint8(reshape(Centered_Known_Im(:,j),N,N)))
%     if j==1
%         title({['Centered Training Face No.' num2str(j)];....
%             ['(' Known_Image(1:length(Known_Image)-6) .....
%             ', 1^{st} Projection)']})
%     elseif j==2
%         title({['Centered Training Face No.' num2str(j)];....
%             ['(' Known_Image(1:length(Known_Image)-6) ......
%             ', 2^{nd} Projection)']})
%     elseif j==3
%         title({['Centered Training Face No.' num2str(j)];....
%             ['(' Known_Image(1:length(Known_Image)-6) ......
%             ', 3^{rd} Projection)']})
%     elseif j==4
%         title({['Centered Training Face No.' num2str(j)];....
%             ['(' Known_Image(1:length(Known_Image)-6) ......
%             ', 1^{st} Projection)']})
%     elseif j==5
%         title({['Centered Training Face No.' num2str(j)];....
%             ['(' Known_Image(1:length(Known_Image)-6) .....
%             ', 2^{nd} Projection)']})
%     elseif j==6
```

```
527 %         title({['Centered Training Face No.' num2str(j)];....
528 %             ['(' Known_Image(1:length(Known_Image)-6) .....
529 %             ', 3^{rd} Projection)']})
530 %     elseif j==7
531 %         title({['Centered Training Face No.' num2str(j)];....
532 %             ['(' Known_Image(1:length(Known_Image)-6) .....
533 %             ', 1^{st} Projection)']})
534 %     elseif j==8
535 %         title({['Centered Training Face No.' num2str(j)];....
536 %             ['(' Known_Image(1:length(Known_Image)-6) ......
537 %             ', 2^{nd} Projection)']})
538 %     elseif j==9
539 %         title({['Centered Training Face No.' num2str(j)];....
540 %             ['(' Known_Image(1:length(Known_Image)-6) ......
541 %             ', 3^{rd} Projection)']})
542 %     end
543 %     disp(['Please, press any keyboard button to explore '......
544 %         'the remaining centered training faces >>>>>>>'])
545 %     pause
546 %     close all
547 %     clc
548 end
549 Centered_Known_Im;
550
551
552 % Tested images are supposed to be unknown but here multiple
553 % images for each training face are taken for testing the face
554 % reconstruction, recognition and detection processes as well
555 % as selecting a decision threshold for face detection and
556 % recognition.
557 Total_No_of_Tested_Im=180; % Total number of the
558                            % tested images.
559
560 Im_P=[36 12 12 36 12 12 36 12 12]; % Each element in this vector
561                                    % represents the total number of
562                                    % the taken images for each
563                                    % training face.
564
565 L1=Im_P(1);
566 L2=L1+Im_P(2);
567 L3=L2+Im_P(3); % L3=60 is the total number of the tested
568                % images for Mr. Mansour Alshammari.
569 L4=L3+Im_P(4);
570 L5=L4+Im_P(5);
571 L6=L5+Im_P(6); % L6=120 is the total number of the tested images
572                % for Mr. Methkir Alharthee.
573 L7=L6+Im_P(7);
574 L8=L7+Im_P(8);
575 L9=L8+Im_P(9); % L9=180 is the total number of the tested images
576                % for Mr. Mohammed Hanafy.
```

```
577
578 All_Tested_Im_V=zeros(N*N,.....
579     Total_No_of_Tested_Im); % An N^2xP1, 2D matrix where P1 is the
580                             % total number of the tested images.
581                             % Each tested image is vectorized and
582                             % placed in one of the columns of the
583                             % 2D matrix. The size of each tested
584                             % image vector is N^2.
585
586 Tested_Images_Folder=.....
587     [cd '/The Tested Images of Black Backgrounds']; % The folder of
588                                                     % the black
589                                                     % background
590                                                     % tested
591                                                     % images.
592 % Tested_Images_Folder=.....
593 %     [cd '/The Tested Images of White Backgrounds']; % The folder
594                                                     % of the white
595                                                     % background
596                                                     % tested
597                                                     % images.
598
599 if isdir(Tested_Images_Folder)==0
600     Error_Message1=sprintf(.....
601         'Error: The following folder does not exist\n%s'.....
602         ,Tested_Images_Folder);
603     warndlg(Error_Message1);
604 end
605
606 Tested_Images=dir(fullfile(Tested_Images_Folder,'*.jpg'));
607 for k1=1:length(Tested_Images)
608     Tested_Image_Number=[num2str(k1) '.jpg'];
609     Tested_Image_Location=.....
610         fullfile(Tested_Images_Folder,Tested_Image_Number);
611     All_Tested_Im_V(:,k1)=reshape.............
612         (double(rgb2gray(imread(Tested_Image_Location))),N*N,1);
613
614 %     figure('units','centimeters','position',[16 7 7.5 8.5])
615 %     subplot(1,1,1)
616 %     imshow(uint8(reshape(All_Tested_Im_V(:,k1),N,N)))
617 %     if k1<=L1
618 %         title({['Tested Image No.' num2str(k1)];.....
619 %             '(Mr. Mansour Alshammari, 1^{st} Projection)'})
620 %     elseif k1>L1 && k1<=L2
621 %         title({['Tested Image No.' num2str(k1)];.....
622 %             '(Mr. Mansour Alshammari, 2^{nd} Projection)'})
623 %     elseif k1>L2 && k1<=L3
624 %         title({['Tested Image No.' num2str(k1)];.....
625 %             '(Mr. Mansour Alshammari, 3^{rd} Projection)'})
626 %     elseif k1>L3 && k1<=L4
```

```
627 %         title({['Tested Image No.' num2str(k1)];...........
628 %             '(Mr. Methkir Alharthee, 1^{st} Projection)'})
629 %     elseif k1>L4 && k1<=L5
630 %         title({['Tested Image No.' num2str(k1)];...........
631 %             '(Mr. Methkir Alharthee, 2^{nd} Projection)'})
632 %     elseif k1>L5 && k1<=L6
633 %         title({['Tested Image No.' num2str(k1)];...........
634 %             '(Mr. Methkir Alharthee, 3^{rd} Projection)'})
635 %     elseif k1>L6 && k1<=L7
636 %         title({['Tested Image No.' num2str(k1)];...........
637 %             '(Mr. Mohammed Hanafy, 1^{st} Projection)'})
638 %     elseif k1>L7 && k1<=L8
639 %         title({['Tested Image No.' num2str(k1)];...........
640 %             '(Mr. Mohammed Hanafy, 2^{nd} Projection)'})
641 %     elseif k1>L8 && k1<=L9
642 %         title({['Tested Image No.' num2str(k1)];...........
643 %             '(Mr. Mohammed Hanafy, 3^{rd} Projection)'})
644 %     end
645 %     disp(['Please, press any keyboard button to explore '......
646 %         'the remaining tested images >>>>>>>'])
647 %     pause
648 %     close all
649 %     clc
650 end
651 All_Tested_Im_V;
652
653 % Imhist for setting up a threshold to work on just the pixels of a
654 % face and throwing the background pixels. Imhist calculates the
655 % number of pixels in an image that have the same intensity levels.
656 % So, if a tested image has a unified background then the biggest
657 % histogram of the intensity levels will be for the background
658 % pixels because the total number of pixels that have the same
659 % intensity levels are the background pixels of the tested image.
660 % Note that, the histogram of a digital image is defined as the
661 % discrete function, h(rk)=nk, where rk is the kth intensity level
662 % and nk is the number of pixels in the image whose intensity level
663 % is rk.
664 hist_Tested_Im=zeros(Total_No_of_Tested_Im,256);
665 for A1=1:Total_No_of_Tested_Im
666     % Note that, a tested image has to be scaled between 0 to 255
667     % before using imhist. For doing that, uint8 can be used for
668     % converting the tested image class form double to uint8.
669     hist_Tested_Im(A1,:)=.......
670         imhist(uint8(reshape(All_Tested_Im_V(:,A1),N,N)));
671
672 %     plot(hist_Tested_Im(A1,:))
673 %     if A1<=L1
674 %         title({['The Histogram of Training Face No.' ......
675 %             num2str(A1)];.......
676 %             '(Mr. Mansour Alshammari, 1^{st} Projection)'})
```

```
677 %     elseif A1>L1 && A1<=L2
678 %         title({['The Histogram of Training Face No.' ......
679 %             num2str(A1)];.......
680 %             '(Mr. Mansour Alshammari, 2^{nd} Projection)'})
681 %     elseif A1>L2 && A1<=L3
682 %         title({['The Histogram of Training Face No.' .......
683 %             num2str(A1)];.......
684 %             '(Mr. Mansour Alshammari, 3^{rd} Projection)'})
685 %     elseif A1>L3 && A1<=L4
686 %         title({['The Histogram of Training Face No.' ......
687 %             num2str(A1)];.......
688 %             '(Mr. Methkir Alharthee, 1^{st} Projection)'})
689 %     elseif A1>L4 && A1<=L5
690 %         title({['The Histogram of Training Face No.' ......
691 %             num2str(A1)];.......
692 %             '(Mr. Methkir Alharthee, 2^{nd} Projection)'})
693 %     elseif A1>L5 && A1<=L6
694 %         title({['The Histogram of Training Face No.' .....
695 %             num2str(A1)];.......
696 %             '(Mr. Methkir Alharthee, 3^{rd} Projection)'})
697 %     elseif A1>L6 && A1<=L7
698 %         title({['The Histogram of Training Face No.' ......
699 %             num2str(A1)];.......
700 %             '(Mr. Mohammed Hanafy, 1^{st} Projection)'})
701 %     elseif A1>L7 && A1<=L8
702 %         title({['The Histogram of Training Face No.' ......
703 %             num2str(A1)];.......
704 %             '(Mr. Mohammed Hanafy, 2^{nd} Projection)'})
705 %     elseif A1>L8 && A1<=L9
706 %         title({['The Histogram of Training Face No.' .....
707 %             num2str(A1)];.......
708 %             '(Mr. Mohammed Hanafy, 3^{rd} Projection)'})
709 %     end
710 %     xlabel('Intensity Level rk')
711 %     ylabel({'The Number of Pixels in the Tested Image' ........
712 %         ' Whose Intensity Level Is rk Where h(rk)=nk'})
713 %     axis tight
714 %     disp(['Please, press any keyboard button to explore '......
715 %         'the remaining histograms >>>>>>>'])
716 %     pause
717 %     clc
718 end
719 hist_Tested_Im;
720
721 Mean_hist_Tested_Im=...........
722     sum(hist_Tested_Im,1)/Total_No_of_Tested_Im; % The average
723                                                  % histogram for
724                                                  % all tested
725                                                  % images.
726 Threshold_Tested_Im=8; % The picked threshold is based on the
```

```
727                          % average histogram for all tested images
728                          % when the tested images have black
729                          % backgrounds. Note that, all intesity
730                          % levels below the threshold represent the
731                          % images backgrounds because these levels
732                          % have the biggest histogram.
733 % Threshold_Tested_Im=180; % The picked threshold is based on the
734                            % average histogram for all tested
735                            % images when the tested images have
736                            % white backgrounds. Note that, all
737                            % intesity levels above the threshold
738                            % represent the images backgrounds
739                            % because these levels have the biggest
740                            % histogram.
741 % plot(Mean_hist_Tested_Im)
742 % line([Threshold_Tested_Im Threshold_Tested_Im],..............
743 %     [0 max(Mean_hist_Tested_Im)],'Color','r')
744 % text(Threshold_Tested_Im+0.5,max(Mean_hist_Tested_Im)/2,........
745 %     '{\color{red} The Threshold}')
746 % title('The Mean Histogram of All Tested Images')
747 % xlabel('Intensity Level rk')
748 % set(gca,'XTick',[0 Threshold_Tested_Im 255])
749 % ylabel({'The Mean Number of Pixels from All Tested'......
750 %     'Images Whose Intensity Level Is rk'})
751 % axis tight
752 % pause
753
754 % Normalizing all the tested images for removing the lightening
755 % effects on them and to increase the resolution of face detection
756 % and recognition. Note that, the normalization will be done just
757 % for face pixels for keeping the variations among the images just
758 % in the faces without the backgrounds effects.
759 Threshold_Tested_Image=.........
760     zeros(N,N,Total_No_of_Tested_Im); % The tested images after
761                                       % applying the threshold.
762 Normalized_Tested_Im_V=zeros(N*N,...........
763     Total_No_of_Tested_Im); % An N^2xP1, 2D matrix where each
764                             % column represents a normalized
765                             % tested image vector.
766 for M1=1:Total_No_of_Tested_Im
767     t1=reshape(All_Tested_Im_V(:,M1),N,N);
768     T1=t1>Threshold_Tested_Im; % The pixels bigger than the
769                               % threshold are of interest because
770                               % they represent the pixels of
771                               % a face.
772 %     T1=t1<Threshold_Tested_Im; % The pixels smaller than the
773 %                               % threshold are of interest
774 %                               % because they represent the
775 %                               % pixels of a face.
776     for R1=1:N
```

```
777          for C1=1:N
778              if T1(R1,C1)==1
779                  Threshold_Tested_Image(R1,C1,M1)=.........
780                      floor(255*(double(t1(R1,C1))/.......
781                      max(max(double(t1))));  % The normalization of
782                                              % a tested image. This
783                                              % is done to increase
784                                              % the dynamic range of
785                                              % the tested image for
786                                              % visualization by
787                                              % scaling the
788                                              % intensities of the
789                                              % tested image from 0
790                                              % to 255.
791              end;
792          end;
793      end;
794      Normalized_Tested_Im_V(:,M1)=.............
795          reshape(Threshold_Tested_Image(:,:,M1),N*N,1);
796
797  %     figure
798  %     subplot(2,1,1)
799  %     imshow(t1)
800  %     if M1<=L1
801  %         title({['This Is To Show How Good the Tested '....
802  %             'Images Threshold Is,'];.....
803  %             blanks(1);['Tested Image No.' num2str(M1)];....
804  %             '(Mr. Mansour Alshammari, 1^{st} Projection)'})
805  %     elseif M1>L1 && M1<=L2
806  %         title({['This Is To Show How Good the Tested '....
807  %             'Images Threshold Is,'];.....
808  %             blanks(1);['Tested Image No.' num2str(M1)];....
809  %             '(Mr. Mansour Alshammari, 2^{nd} Projection)'})
810  %     elseif M1>L2 && M1<=L3
811  %         title({['This Is To Show How Good the Tested '....
812  %             'Images Threshold Is,'];.....
813  %             blanks(1);['Tested Image No.' num2str(M1)];....
814  %             '(Mr. Mansour Alshammari, 3^{rd} Projection)'})
815  %     elseif M1>L3 && M1<=L4
816  %         title({['This Is To Show How Good the Tested '.....
817  %             'Images Threshold Is,'];.....
818  %             blanks(1);['Tested Image No.' num2str(M1)];....
819  %             '(Mr. Methkir Alharthee, 1^{st} Projection)'})
820  %     elseif M1>L4 && M1<=L5
821  %         title({['This Is To Show How Good the Tested '.....
822  %             'Images Threshold Is,'];.....
823  %             blanks(1);['Tested Image No.' num2str(M1)];.....
824  %             '(Mr. Methkir Alharthee, 2^{nd} Projection)'})
825  %     elseif M1>L5 && M1<=L6
826  %         title({['This Is To Show How Good the Tested '.....
```

```
827 %             'Images Threshold Is,'];.....
828 %             blanks(1);['Tested Image No.' num2str(M1)];.....
829 %             '(Mr. Methkir Alharthee, 3^{rd} Projection)'})
830 %     elseif M1>L6 && M1<=L7
831 %         title({['This Is To Show How Good the Tested '....
832 %             'Images Threshold Is,'];.....
833 %             blanks(1);['Tested Image No.' num2str(M1)];....
834 %             '(Mr. Mohammed Hanafy, 1^{st} Projection)'})
835 %     elseif M1>L7 && M1<=L8
836 %         title({['This Is To Show How Good the Tested '.....
837 %             'Images Threshold Is,'];.....
838 %             blanks(1);['Tested Image No.' num2str(M1)];.....
839 %             '(Mr. Mohammed Hanafy, 2^{nd} Projection)'})
840 %     elseif M1>L8 && M1<=L9
841 %         title({['This Is To Show How Good the Tested '....
842 %             'Images Threshold Is,'];.....
843 %             blanks(1);['Tested Image No.' num2str(M1)];......
844 %             '(Mr. Mohammed Hanafy, 3^{rd} Projection)'})
845 %     end
846 %     subplot(2,1,2)
847 %     imshow(reshape(Normalized_Tested_Im_V(:,M1),N,N))
848 %     if M1<=L1
849 %         title({['Normalized Tested Image No.' num2str(M1)];....
850 %             '(Mr. Mansour Alshammari, 1^{st} Projection)'})
851 %     elseif M1>L1 && M1<=L2
852 %         title({['Normalized Tested Image No.' num2str(M1)];....
853 %             '(Mr. Mansour Alshammari, 2^{nd} Projection)'})
854 %     elseif M1>L2 && M1<=L3
855 %         title({['Normalized Tested Image No.' num2str(M1)];....
856 %             '(Mr. Mansour Alshammari, 3^{rd} Projection)'})
857 %     elseif M1>L3 && M1<=L4
858 %         title({['Normalized Tested Image No.' num2str(M1)];....
859 %             '(Mr. Methkir Alharthee, 1^{st} Projection)'})
860 %     elseif M1>L4 && M1<=L5
861 %         title({['Normalized Tested Image No.' num2str(M1)];....
862 %             '(Mr. Methkir Alharthee, 2^{nd} Projection)'})
863 %     elseif M1>L5 && M1<=L6
864 %         title({['Normalized Tested Image No.' num2str(M1)];....
865 %             '(Mr. Methkir Alharthee, 3^{rd} Projection)'})
866 %     elseif M1>L6 && M1<=L7
867 %         title({['Normalized Tested Image No.' num2str(M1)];....
868 %             '(Mr. Mohammed Hanafy, 1^{st} Projection)'})
869 %     elseif M1>L7 && M1<=L8
870 %         title({['Normalized Tested Image No.' num2str(M1)];....
871 %             '(Mr. Mohammed Hanafy, 2^{nd} Projection)'})
872 %     elseif M1>L8 && M1<=L9
873 %         title({['Normalized Tested Image No.' num2str(M1)];....
874 %             '(Mr. Mohammed Hanafy, 3^{rd} Projection)'})
875 %     end
876 %     disp(['Please, press any keyboard button to '......
```

```
877 %         'see how good the applied'])
878 %     disp('threshold on the normalized tested images is >>>>>>>')
879 %     pause
880 %     close all
881 %     clc
882 %
883 %     figure
884 %     subplot(2,1,1)
885 %     imshow(uint8(t1))
886 %     if M1<=L1
887 %         title({['Tested Image No.' num2str(M1)];.........
888 %             '(Mr. Mansour Alshammari, 1^{st} Projection)'})
889 %     elseif M1>L1 && M1<=L2
890 %         title({['Tested Image No.' num2str(M1)];.......
891 %             '(Mr. Mansour Alshammari, 2^{nd} Projection)'})
892 %     elseif M1>L2 && M1<=L3
893 %         title({['Tested Image No.' num2str(M1)];......
894 %             '(Mr. Mansour Alshammari, 3^{rd} Projection)'})
895 %     elseif M1>L3 && M1<=L4
896 %         title({['Tested Image No.' num2str(M1)];.......
897 %             '(Mr. Methkir Alharthee, 1^{st} Projection)'})
898 %     elseif M1>L4 && M1<=L5
899 %         title({['Tested Image No.' num2str(M1)];......
900 %             '(Mr. Methkir Alharthee, 2^{nd} Projection)'})
901 %     elseif M1>L5 && M1<=L6
902 %         title({['Tested Image No.' num2str(M1)];.....
903 %             '(Mr. Methkir Alharthee, 3^{rd} Projection)'})
904 %     elseif M1>L6 && M1<=L7
905 %         title({['Tested Image No.' num2str(M1)];.......
906 %             '(Mr. Mohammed Hanafy, 1^{st} Projection)'})
907 %     elseif M1>L7 && M1<=L8
908 %         title({['Tested Image No.' num2str(M1)];.....
909 %             '(Mr. Mohammed Hanafy, 2^{nd} Projection)'})
910 %     elseif M1>L8 && M1<=L9
911 %         title({['Tested Image No.' num2str(M1)];.......
912 %             '(Mr. Mohammed Hanafy, 3^{rd} Projection)'})
913 %     end
914 %     subplot(2,1,2)
915 %     imshow(uint8(reshape(Normalized_Tested_Im_V(:,M1),N,N)))
916 %     if M1<=L1
917 %         title({['Normalized Tested Image No.' num2str(M1)];....
918 %             '(Mr. Mansour Alshammari, 1^{st} Projection)'})
919 %     elseif M1>L1 && M1<=L2
920 %         title({['Normalized Tested Image No.' num2str(M1)];....
921 %             '(Mr. Mansour Alshammari, 2^{nd} Projection)'})
922 %     elseif M1>L2 && M1<=L3
923 %         title({['Normalized Tested Image No.' num2str(M1)];....
924 %             '(Mr. Mansour Alshammari, 3^{rd} Projection)'})
925 %     elseif M1>L3 && M1<=L4
926 %         title({['Normalized Tested Image No.' num2str(M1)];....
```

```
927 %             '(Mr. Methkir Alharthee, 1^{st} Projection)'})
928 %     elseif M1>L4 && M1<=L5
929 %         title({['Normalized Tested Image No.' num2str(M1)];....
930 %             '(Mr. Methkir Alharthee, 2^{nd} Projection)'})
931 %     elseif M1>L5 && M1<=L6
932 %         title({['Normalized Tested Image No.' num2str(M1)];....
933 %             '(Mr. Methkir Alharthee, 3^{rd} Projection)'})
934 %     elseif M1>L6 && M1<=L7
935 %         title({['Normalized Tested Image No.' num2str(M1)];....
936 %             '(Mr. Mohammed Hanafy, 1^{st} Projection)'})
937 %     elseif M1>L7 && M1<=L8
938 %         title({['Normalized Tested Image No.' num2str(M1)];....
939 %             '(Mr. Mohammed Hanafy, 2^{nd} Projection)'})
940 %     elseif M1>L8 && M1<=L9
941 %         title({['Normalized Tested Image No.' num2str(M1)];....
942 %             '(Mr. Mohammed Hanafy, 3^{rd} Projection)'})
943 %     end
944 %     disp(['Please, press any keyboard button to explore '....
945 %         'the remaining normalized tested images >>>>>>>'])
946 %     pause
947 %     close all
948 %     clc
949 end
950 Normalized_Tested_Im_V;
951
952 % Centering each face by simply subtracting the mean of the face
953 % pixels from each pixel in the face. By doing that the new face
954 % pixels will have zero mean that means the face is centered.
955 Rows_Columns1=zeros(1,1,.......
956     Total_No_of_Tested_Im); % The rows and columns for the
957                             % pixels of the faces. Note that,
958                             % the tested images are not similar
959                             % so the rows and columns of the
960                             % faces pixels will not be equal.
961                             % Therefore, MATLAB will add zero
962                             % rows and columns to make the
963                             % matrices of the rows and columns
964                             % of the faces pixels are equal.
965 Means1=zeros(1,Total_No_of_Tested_Im);
966 Centered_Tested_Im=.........
967     zeros(N*N,Total_No_of_Tested_Im); % An N^2xP1, 2D matrix where
968                                       % each column represents a
969                                       % tested image vector with a
970                                       % centered face.
971 Centered_Tested_Image=..........
972     zeros(N,N,Total_No_of_Tested_Im); % The tested images after
973                                       % centering the faces.
974 for j1=1:Total_No_of_Tested_Im
975     x1=reshape(Normalized_Tested_Im_V(:,j1),N,N);
976     [rr1 cc1]=find(x1>0); % The pixels bigger than zero are of
```

```
977                             % interest because they represent the
978                             % pixels of the faces.
979       Rows_Columns1(1,1:size(rr1,1),j1)=rr1.';
980       Rows_Columns1(2,1:size(rr1,1),j1)=cc1.';
981       Sum1=0;
982       No1=0;
983       for RR2=1:size(rr1,1)
984           Sum1=Sum1+x1(rr1(RR2),cc1(RR2));
985           No1=No1+1;
986       end
987
988       Means1(1,j1)=Sum1/No1;
989       for RR3=1:size(rr1,1)
990           Centered_Tested_Image(rr1(RR3),cc1(RR3),j1)=.......
991               x1(rr1(RR3),cc1(RR3))-Means1(1,j1);
992       end
993       Centered_Tested_Im(:,j1)=..........
994           reshape(Centered_Tested_Image(:,:,j1),N*N,1);
995
996   %     figure('units','centimeters','position',[16 7 7 8.5])
997   %     subplot(1,1,1)
998   %     imshow(uint8(reshape(Centered_Tested_Im(:,j1),N,N)))
999   %     if j1<=L1
1000  %         title({['Centered Tested Image No.' num2str(j1)];....
1001  %             '(Mr. Mansour Alshammari, 1^{st} Projection)'})
1002  %     elseif j1>L1 && j1<=L2
1003  %         title({['Centered Tested Image No.' num2str(j1)];.....
1004  %             '(Mr. Mansour Alshammari, 2^{nd} Projection)'})
1005  %     elseif j1>L2 && j1<=L3
1006  %         title({['Centered Tested Image No.' num2str(j1)];....
1007  %             '(Mr. Mansour Alshammari, 3^{rd} Projection)'})
1008  %     elseif j1>L3 && j1<=L4
1009  %         title({['Centered Tested Image No.' num2str(j1)];....
1010  %             '(Mr. Methkir Alharthee, 1^{st} Projection)'})
1011  %     elseif j1>L4 && j1<=L5
1012  %         title({['Centered Tested Image No.' num2str(j1)];....
1013  %             '(Mr. Methkir Alharthee, 2^{nd} Projection)'})
1014  %     elseif j1>L5 && j1<=L6
1015  %         title({['Centered Tested Image No.' num2str(j1)];....
1016  %             '(Mr. Methkir Alharthee, 3^{rd} Projection)'})
1017  %     elseif j1>L6 && j1<=L7
1018  %         title({['Centered Tested Image No.' num2str(j1)];.....
1019  %             '(Mr. Mohammed Hanafy, 1^{st} Projection)'})
1020  %     elseif j1>L7 && j1<=L8
1021  %         title({['Centered Tested Image No.' num2str(j1)];....
1022  %             '(Mr. Mohammed Hanafy, 2^{nd} Projection)'})
1023  %     elseif j1>L8 && j1<=L9
1024  %         title({['Centered Tested Image No.' num2str(j1)];....
1025  %             '(Mr. Mohammed Hanafy, 3^{rd} Projection)'})
1026  %     end
```

```
%     disp(['Please, press any keyboard button to explore '......
%         'the remaining centered tested images >>>>>>>'])
%     pause
%     close all
%     clc
end
Centered_Tested_Im;


% Centering the set of the training faces by simply subtracting the
% mean training face from each training face in the set. By doing
% that the new set of the training faces will have zero mean which
% means the set is centered.
Av_Image=sum(Centered_Known_Im,2)./......
    Total_No_of_Known_Im; % The average training face.
% figure('units','centimeters','position',[12 4 12.5 13])
% subplot(1,1,1)
% Reshaped_Av_Image=reshape(Av_Image,N,N);
% Negative_Av_Image=255*ones(N,N)-255*(Reshaped_Av_Image/.....
%     max(max(Reshaped_Av_Image))); % Obtaining a negative image
%                                   % for the mean training face
%                                   % in order to enhance its
%                                   % appearance.
% imshow(uint8(Negative_Av_Image))
% title('The Average Training Face')
% pause
% close all

Known_Im_Subt_Mean=......
    zeros(N*N,Total_No_of_Known_Im); % An N^2xP, 2D matrix where
                                     % each column represents a
                                     % centered (with respect to
                                     % the set of the training
                                     % faces) training face vector.
for J=1:Total_No_of_Known_Im
    Known_Im_Subt_Mean(:,J)=Centered_Known_Im(:,J)-Av_Image;
    Reshaped_Known_Im_Subt_Mean=........
        reshape(Known_Im_Subt_Mean(:,J),N,N);
    Negative_Known_Im_Subt_Mean=255*ones(N,N)-.......
        255*(Reshaped_Known_Im_Subt_Mean/max(max(.......
        Reshaped_Known_Im_Subt_Mean))); % Obtaining a negative
                                        % image for the centered
                                        % training face in order
                                        % to enhance its
                                        % appearance.

    Known_Image=Known_Images(J).name;
%     figure('units','centimeters','position',[12 4 12.5 14])
%     subplot(1,1,1)
%     imshow(uint8(Negative_Known_Im_Subt_Mean))
```

```
1077 %     if J==1
1078 %         title({['Centered (with Respect to the Set of the '.....
1079 %             'Training Faces) Training Face No.' num2str(J)];....
1080 %             ['(' Known_Image(1:length(Known_Image)-6) .......
1081 %             ', 1^{st} Projection)'];'(A Negative Image)'})
1082 %     elseif J==2
1083 %         title({['Centered (with Respect to the Set of the '....
1084 %             'Training Faces) Training Face No.' num2str(J)];....
1085 %             ['(' Known_Image(1:length(Known_Image)-6) ........
1086 %             ', 2^{nd} Projection)'];'(A Negative Image)'})
1087 %     elseif J==3
1088 %         title({['Centered (with Respect to the Set of the '.....
1089 %             'Training Faces) Training Face No.' num2str(J)];....
1090 %             ['(' Known_Image(1:length(Known_Image)-6) .......
1091 %             ', 3^{rd} Projection)'];'(A Negative Image)'})
1092 %     elseif J==4
1093 %         title({['Centered (with Respect to the Set of the '.....
1094 %             'Training Faces) Training Face No.' num2str(J)];....
1095 %             ['(' Known_Image(1:length(Known_Image)-6) .........
1096 %             ', 1^{st} Projection)'];'(A Negative Image)'})
1097 %     elseif J==5
1098 %         title({['Centered (with Respect to the Set of the '.....
1099 %             'Training Faces) Training Face No.' num2str(J)];....
1100 %             ['(' Known_Image(1:length(Known_Image)-6) ......
1101 %             ', 2^{nd} Projection)'];'(A Negative Image)'})
1102 %     elseif J==6
1103 %         title({['Centered (with Respect to the Set of the '.....
1104 %             'Training Faces) Training Face No.' num2str(J)];....
1105 %             ['(' Known_Image(1:length(Known_Image)-6) .......
1106 %             ', 3^{rd} Projection)'];'(A Negative Image)'})
1107 %     elseif J==7
1108 %         title({['Centered (with Respect to the Set of the '.....
1109 %             'Training Faces) Training Face No.' num2str(J)];....
1110 %             ['(' Known_Image(1:length(Known_Image)-6) ......
1111 %             ', 1^{st} Projection)'];'(A Negative Image)'})
1112 %     elseif J==8
1113 %         title({['Centered (with Respect to the Set of the '.....
1114 %             'Training Faces) Training Face No.' num2str(J)];....
1115 %             ['(' Known_Image(1:length(Known_Image)-6) .......
1116 %             ', 2^{nd} Projection)'];'(A Negative Image)'})
1117 %     elseif J==9
1118 %         title({['Centered (with Respect to the Set of the '.....
1119 %             'Training Faces) Training Face No.' num2str(J)];....
1120 %             ['(' Known_Image(1:length(Known_Image)-6) .......
1121 %             ', 3^{rd} Projection)'];'(A Negative Image)'})
1122 %     end
1123 %     disp(['Please, press any keyboard button to explore '......
1124 %         'the remaining centered training faces >>>>>>>'])
1125 %     pause
1126 %     close all
```

```
%     clc
end
Known_Im_Subt_Mean;


Tested_Im_Subt_Mean=.......
    zeros(N*N,Total_No_of_Tested_Im); % An N^2xP1, 2D matrix where
                                      % each column represents a
                                      % tested image vector after
                                      % subtracting the average
                                      % training face.
for J1=1:Total_No_of_Tested_Im
    Tested_Im_Subt_Mean(:,J1)=Centered_Tested_Im(:,J1)-Av_Image;
    Reshaped_Tested_Im_Subt_Mean=........
        reshape(Tested_Im_Subt_Mean(:,J1),N,N);
    Negative_Tested_Im_Subt_Mean=255*ones(N,N)-255*.......
        (Reshaped_Tested_Im_Subt_Mean/max(max(.......
        Reshaped_Tested_Im_Subt_Mean))); % Obtaining a negative
                                         % image for the tested
                                         % image in order to
                                         % enhance its appearance.

%     figure('units','centimeters','position',[12 4 12.5 14])
%     subplot(1,1,1)
%     imshow(uint8(Negative_Tested_Im_Subt_Mean))
%     if J1<=L1
%         title({['Tested Image No.' num2str(J1) .........
%             ' (Mr. Mansour Alshammari, '.....
%             '1^{st} Projection)'];........
%             'After Subtracting the Average Training Face';.....
%             '(A Negative Image)'})
%     elseif J1>L1 && J1<=L2
%         title({['Tested Image No.' num2str(J1) .......
%             ' (Mr. Mansour Alshammari, '.....
%             '2^{nd} Projection)'];.........
%             'After Subtracting the Average Training Face';.....
%             '(A Negative Image)'})
%     elseif J1>L2 && J1<=L3
%         title({['Tested Image No.' num2str(J1) .....
%             ' (Mr. Mansour Alshammari, '.....
%             '3^{rd} Projection)'];.......
%             'After Subtracting the Average Training Face';.....
%             '(A Negative Image)'})
%     elseif J1>L3 && J1<=L4
%         title({['Tested Image No.' num2str(J1) .....
%             ' (Mr. Methkir Alharthee, '.......
%             '1^{st} Projection)'];.....
%             'After Subtracting the Average Training Face';....
%             '(A Negative Image)'})
%     elseif J1>L4 && J1<=L5
```

```
%         title({['Tested Image No.' num2str(J1) .....
%             ' (Mr. Methkir Alharthee, '.......
%             '2^{nd} Projection)'];.....
%             'After Subtracting the Average Training Face';.....
%             '(A Negative Image)'})
%     elseif J1>L5 && J1<=L6
%         title({['Tested Image No.' num2str(J1) .....
%             ' (Mr. Methkir Alharthee, '.......
%             '3^{rd} Projection)'];....
%             'After Subtracting the Average Training Face';....
%             '(A Negative Image)'})
%     elseif J1>L6 && J1<=L7
%         title({['Tested Image No.' num2str(J1) .....
%             ' (Mr. Mohammed Hanafy, '.......
%             '1^{st} Projection)'];.....
%             'After Subtracting the Average Training Face';....
%             '(A Negative Image)'})
%     elseif J1>L7 && J1<=L8
%         title({['Tested Image No.' num2str(J1) .....
%             ' (Mr. Mohammed Hanafy, '.......
%             '2^{nd} Projection)'];.....
%             'After Subtracting the Average Training Face';.....
%             '(A Negative Image)'})
%     elseif J1>L8 && J1<=L9
%         title({['Tested Image No.' num2str(J1) .....
%             ' (Mr. Mohammed Hanafy, '.......
%             '3^{rd} Projection)'];.....
%             'After Subtracting the Average Training Face';....
%             '(A Negative Image)'})
%     end
%     disp(['Please, press any keyboard button to explore '.....
%         'the remaining tested images'])
%     disp(['after subtracting the average training face '.....
%         'from them >>>>>>>'])
%     pause
%     close all
%     clc
end
Tested_Im_Subt_Mean;


% The calculation of a covariance matrix for all training faces.
Cov_Matrix=Known_Im_Subt_Mean*......
    transpose(Known_Im_Subt_Mean); % The covariance matrix
                                   % for all training faces.

% Another way for calculating the covariance matrix for all
% training faces.
% m=zeros(N*N,N*N);
% for i=1:Total_No_of_Known_Im
```

```
%     C=Known_Im_Subt_Mean(:,i)*transpose(Known_Im_Subt_Mean(:,i));
%     m=m+C;
% end
% Cov_Matrix=(1/Total_No_of_Known_Im)*m; % The covariance matrix
                                         % for all training faces.


% The calculation of eigenvalues and eigenvectors for the
% covariance matrix.
[Eigenvectors Eigenvalues]=.......
    eig(Cov_Matrix); % Note that, The calculated covariance
                     % matrix is usually too big which makes
                     % the calculation of eigenvalues and
                     % eigenvectors is very difficult if not
                     % impossible. So, it is not practical to
                     % calculate the eigenvalues and eigenvectors
                     % for the such matrix. The dimensions of the
                     % covariance matrix can be reduced to the
                     % number of the training faces as will be
                     % proved shortly.
Eigenvalues=(diag(Eigenvalues)).';

% Eigenvectors must be positive because the covariance matrix is
% positive definite. Due to round-off error, MATLAB makes the
% smallest eigenvectors negative. Then those negative eigenvectors
% must be set to zero.
for jjj=1:length(Eigenvalues)
    if Eigenvalues(1,jjj)<0
        Eigenvalues(1,jjj)=0;
    end
end


% The calculation of a more practical covariance matrix.
New_Cov_Matrix=transpose(Known_Im_Subt_Mean)*Known_Im_Subt_Mean;


[Eigvect Eigval]=eig(New_Cov_Matrix); % The calculation of
                                      % eigenvalues and
                                      % eigenvectors for the
                                      % reduced covariance matrix.

Eigval_Reduced_Cov=(diag(Eigval)).';
Eigvect_Reduced_Cov=Known_Im_Subt_Mean*.........
    Eigvect; % The columns of this matrix represent unnormalized
             % eigenvectors that are calculated based on the more
             % practical covariance matrix.

% Ordering the calculated eigenvalues from the reduced covariance
% matrix along with their eigenvectors in descending order as well
```

```
% as normalizing the eigenvectors.
Eigenvalues_Reduced_Cov=sort(Eigval_Reduced_Cov,'descend');
Eigenvectors_Reduced_Cov=zeros(size(Eigvect_Reduced_Cov,1),.....
    size(Eigvect_Reduced_Cov,2));
for ii=1:length(Eigenvalues_Reduced_Cov)
    for pp=1:length(Eigval_Reduced_Cov)
        if Eigenvalues_Reduced_Cov(1,ii)==Eigval_Reduced_Cov(1,pp)
            Eigenvectors_Reduced_Cov(:,ii)=........
                Eigvect_Reduced_Cov(:,pp)/.....
                norm(Eigvect_Reduced_Cov(:,pp));
            break
        end
    end
end


% PCA and IPCA algoritms for calculating a feature vector matrix.
% A feature vector matrix is a matrix that is composed of a couple
% of eigenvectors that follow the most significant patterns of the
% correlated faces. These eigenvectors are called eigenfaces. In
% fact, the eigenvalues associated with those eigenfaces are
% corresponding to the biggest calculated eigenvalues. Note that,
% the PCA is approximately similar to the IPCA. The main difference
% between them is that the way of selecting eigenvectors which form
% the feature vector matrix.

fid=fopen('PCA vs. IPCA.txt','w'); % A text file for typing the
                                   % required results to select the
                                   % desired eigenvectors for the
                                   % feature vector matrix.
fprintf(fid,['\n  ***** The Results of PCA and IPCA Used to'.....
    ' Select the Desired Eigenvectors *****\r\n']);
fprintf(fid,['    ***** for the Feature Vector'......
    ' Matrix. These Results Are Obtained from *****\r\n']);
fprintf(fid,['\t ***** the PCA and IPCA Code for Testing '......
    'and Setting up Thresholds *****\r\n\n']);
fprintf(fid,'\t\t\t\t      Transformed\r\n');
fprintf(fid,['No.    Eigenvalues    Eigenvalues   PIF'......
    '       IR(%%)\t    AIR(%%)     VCR(%%)    TVC(%%)\r\n']);
fprintf(fid,['====   ============   ===========   ======='......
    '   =======   ========   =======   ========\r\n']);

% The calculation of the transformed eigenvalues.
ro=zeros(1,length(Eigenvalues_Reduced_Cov));
for v=1:length(Eigenvalues_Reduced_Cov)
    ro(1,v)=1-(Eigenvalues_Reduced_Cov(1,v)/......
        sum(Eigenvalues_Reduced_Cov));
end

% The calculation of the possibility information function (PIF).
```

```
PIF=zeros(1,length(Eigenvalues_Reduced_Cov));
for nn=1:length(Eigenvalues_Reduced_Cov)
    PIF(1,nn)=-log2(ro(1,nn));
end

% The calculation of the possibility information entropy (PIE).
PIE=0;
for n1=1:length(Eigenvalues_Reduced_Cov)
    PIE=PIE-ro(1,n1)*log2(ro(1,n1));
end
PIE;

% The calculation of the information rate (IR) and accumulated
% information rate (AIR).
IR=zeros(1,length(Eigenvalues_Reduced_Cov));
AIR=zeros(1,length(Eigenvalues_Reduced_Cov));
PIF1=0;
for u=1:length(Eigenvalues_Reduced_Cov);
    IR(1,u)=PIF(1,u)/sum(PIF);
    PIF1=PIF(1,u)+PIF1;
    AIR(1,u)=PIF1/sum(PIF);
end

% The calculation of the variance contribution rate (VCR) and total
% variance contribution rate (TVC).
VCR=zeros(1,length(Eigenvalues_Reduced_Cov));
TVC=zeros(1,length(Eigenvalues_Reduced_Cov));
TVC1=0;
for Q=1:length(Eigenvalues_Reduced_Cov);
    VCR(1,Q)=Eigenvalues_Reduced_Cov(1,Q)/......
        sum(Eigenvalues_Reduced_Cov);
    TVC1=Eigenvalues_Reduced_Cov(1,Q)+TVC1;
    TVC(1,Q)=TVC1/sum(Eigenvalues_Reduced_Cov);
    fprintf(fid,['%-4.0f   %-12.4f   %-6.4f\t    %-7.4f   '....
        '%-7.4f   %-8.4f   %-7.4f   %-8.4f \r\n\n'],Q,.....
        Eigenvalues_Reduced_Cov(1,Q),ro(1,Q),PIF(1,Q),....
        IR(1,Q)*100,AIR(1,Q)*100,VCR(1,Q)*100,TVC(1,Q)*100);
end

fprintf(fid,['=============================================='.....
    '=====================================\r\n']);
fprintf(fid,['** Notice that, in IPCA we can take the first '.....
    'seven eigenvectors associated with the\r\n']);
fprintf(fid,['   biggest eigenvalues to form the feature '.....
    'vector matrix but in PCA we must take the\r\n']);
fprintf(fid,['   first eight eigenvectors. That because when '....
    'm=7, the AIR=95.6891%% which is good\r\n']);
fprintf(fid,['   enough but the TVC is slightly small. So, '.....
    'if we want to pick the eigenvectors\r\n']);
fprintf(fid,['   based on the TVC then we have to take the '.....
```

```
1377      'first eight eigenvectors in order to\r\n']);
1378  fprintf(fid,['   make sure that the TVC is big enough. '.....
1379      'Therefore, the AIR tells us more about the\r\n']);
1380  fprintf(fid,['   information contained in the eigenfaces. '.....
1381      '\r\n\n\n']);
1382  fclose(fid);
1383
1384  disp(['Please see the documented results of PCA and IPCA in '....
1385      'the open text file. Then select the'])
1386  disp(['number of the eigenvectors for each algorithm required '....
1387      'to form the feature vector matrix.'])
1388  disp(['After that, press any keyboard button to resume the '.....
1389      'code >>>>>>>>>>>>'])
1390  Text='PCA vs. IPCA.txt';
1391  open(Text) % Opening the text file which contains
1392             % the PCA and IPCA results.
1393  pause
1394  clc
1395  open('PCA_IPCA_Testing_and_Setting_up_Thresholds.m')
1396
1397  % Customizing and fixing the number of eigenfaces for PCA and IPCA.
1398  Prompt={['1 to m, enter the value of m for PCA, where m is '.....
1399      'the lower bound of eigenfaces. It can take up to ' ......
1400      num2str(length(Eigenvalues_Reduced_Cov)) ':'],.....
1401      ['1 to m, enter the value of m for IPCA, where m is the '.....
1402      'lower bound of eigenfaces. It can take up to ' .......
1403      num2str(length(Eigenvalues_Reduced_Cov)) ':']};
1404  Dlg_Title='The Lower Bound of Eigenfaces for PCA and IPCA';
1405  Num_Lines=1;
1406  Def={'8','7'};
1407  L=inputdlg(Prompt,Dlg_Title,Num_Lines,Def,'on');
1408  V=str2num(char(L));
1409  for xx=1:2
1410      if xx==1
1411          if V(xx)<1 || V(xx)>length(Eigenvalues_Reduced_Cov) || ....
1412                  mod(V(xx),1)~=0 || V(xx)<=V(2)
1413              if V(xx)<1
1414                  W=errordlg(['m for PCA Is Invalid Because It '.....
1415                      'Is Less Than One'],'An Error Dialog');
1416                  return
1417              elseif V(xx)>length(Eigenvalues_Reduced_Cov)
1418                  W=errordlg(['m for PCA Is Invalid Because It '....
1419                      'Is Bigger Than the Upper Bound of '......
1420                      'Eigenfaces'],'An Error Dialog');
1421                  return
1422              elseif mod(V(xx),1)~=0
1423                  W=errordlg(['m for PCA Is Invalid Because It '....
1424                      'Is Not an Integer'],'An Error Dialog');
1425                  return
1426              elseif V(xx)<=V(2)
```

```
1427                 W=errordlg(['m for PCA Is Invalid Because It '....
1428                     'Must Be Bigger Than IPCA']);
1429                 return
1430             end
1431         end
1432     else
1433         if V(xx)<1 || mod(V(xx),1)~=0
1434             if V(xx)<1
1435                 W=errordlg(['m for IPCA Is Invalid Because It '....
1436                     'Is Less Than One'],'An Error Dialog');
1437                 return
1438             elseif mod(V(xx),1)~=0
1439                 W=errordlg(['m for IPCA Is Invalid Because It '....
1440                     'Is Not an Integer'],'An Error Dialog');
1441                 return
1442             end
1443         end
1444     end
1445 end
1446
1447 Best_Eigenvalues_PCA=Eigenvalues_Reduced_Cov(1,1:V(1));
1448 Best_Eigenvectors_PCA=........
1449     Eigenvectors_Reduced_Cov(:,1:V(1)); % The columns of this
1450                                         % matrix represent the
1451                                         % eigenfaces that are
1452                                         % calculated based on
1453                                         % PCA algorithm.
1454 Best_Eigenvalues_IPCA=Eigenvalues_Reduced_Cov(1,1:V(2));
1455 Best_Eigenvectors_IPCA=..........
1456     Eigenvectors_Reduced_Cov(:,1:V(2)); % The columns of this
1457                                         % matrix represent the
1458                                         % eigenfaces that are
1459                                         % calculated based on
1460                                         % IPCA algorithm.
1461
1462
1463 % Plotting the eigenvalues for each explained algorithm compared
1464 % with the calculated eigenvalues from the covariance matrix for
1465 % all training faces.
1466 Threshold=0.1; % This threshold is for picking
1467                % up the biggest eigenvalues.
1468 [WW LL]=find(Eigenvalues>Threshold);
1469 Flipped_Eigenvalues_Reduced_Cov=fliplr(Eigenvalues_Reduced_Cov);
1470 if length(LL)<length(Flipped_Eigenvalues_Reduced_Cov)
1471     LL1=LL;
1472     LL=[LL(1)-ones(1,length(Flipped_Eigenvalues_Reduced_Cov)-....
1473         length(LL)) LL];
1474 end
1475 figure('units','centimeters','position',[6 1.2 25 16.9])
1476 subplot(1,1,1)
```

```
bar(LL,[zeros(length(LL)-length(LL1)) Eigenvalues(LL1)],.......
    0.8,'FaceColor','k','EdgeColor','k')
hold on
bar(LL,[zeros(1,length(LL)-length(Best_Eigenvalues_PCA)) ........
    fliplr(Best_Eigenvalues_PCA)],0.8/2,'FaceColor',......
    'r','EdgeColor','r')
hold on
bar(LL,[zeros(1,length(LL)-length(Best_Eigenvalues_IPCA)) .......
    fliplr(Best_Eigenvalues_IPCA)],0.8/4,.........
    'FaceColor','b','EdgeColor','b')
hold off
title({['The Calculated Eigenvalues for Each Proposed '.....
    'Algorithm Compared '];['with the Calculated Eigenvalues '....
    'form the Covariance Matrix for All Training Faces']})
xlabel('Eigenvalue Index')
ylabel('Eigenvalue')
legend(['The Calculated Eigenvalues from the Covariance '.....
    'Matrix for All Training Faces.'],['The Calculated '.....
    'Eigenvalues by Using PCA Algorithm.'],['The Calculated '.....
    'Eigenvalues by Using IPCA Algorithm.'],'Location','NorthWest')
disp('Please, press any keyboard button to resume the code >>>>')
pause
clc
close all


% Displaying the calculated eigenfaces from the covariance matrix
% for all training faces.
for I=1:length(LL1)
    Eigenvalues1=fliplr(LL1);
    Eigenvectors1=......
        flipdim(Eigenvectors(:,LL1),2); % This flipping just
                                        % to make the highest
                                        % correlated eigenface
                                        % is associated with the
                                        % first eigenvalue and
                                        % so on. This is done to
                                        % be compatable with the
                                        % generated document
                                        % "PCA vs. IPCA".
    X=floor(255*(double(Eigenvectors1(:,I))/.......
        max(max(Eigenvectors1(:,I))))); % Normalized eigenface.
                                        % This is done to increase
                                        % the dynamic range of the
                                        % eigenface for
                                        % visualization by scaling
                                        % the intensities of the
                                        % eigenface from 0 to 255.
    Eigenface=reshape(X,N,N);
    Negative_Eigenface=255*ones(N,N)-255*.....
```

```
1527          (Eigenface/max(max(Eigenface))); % Obtaining a negative
1528                                           % image for the eigenface
1529                                           % in order to enhance its
1530                                           % appearance.
1531
1532      figure('units','centimeters','position',[15.5 5.5 9 11.5])
1533      subplot(1,1,1)
1534      imshow(uint8(Negative_Eigenface))
1535      title({['Eigenface No.' num2str(I) ..............
1536          ' Associated with Eigenvalue No.'........
1537          num2str(Eigenvalues1(I)) '.'];......
1538          ['It Is Calculated from the '.......
1539          'Covariance Matrix'];'for All Training Faces.';......
1540          '(A Negative Image)'})
1541
1542      disp(['Please, press any keyboard button to explore '......
1543          'the remaining calculated eigenfaces'])
1544      disp(['from the covariance matrix for all training '......
1545          'faces >>>>>>>'])
1546      pause
1547      clc
1548      close all
1549  end
1550
1551  % Displaying the calculated eigenfaces by using PCA algorithm.
1552  for I=1:length(Best_Eigenvalues_PCA)
1553      X=floor(255*(double(Best_Eigenvectors_PCA(:,I))/max(max(.....
1554          Best_Eigenvectors_PCA(:,I))))); % Normalized eigenface.
1555                                         % This is done to increase
1556                                         % the dynamic range of the
1557                                         % eigenface for
1558                                         % visualization by scaling
1559                                         % the intensities of the
1560                                         % eigenface from 0 to 255.
1561      Eigenface=reshape(X,N,N);
1562      Negative_Eigenface=255*ones(N,N)-255*.....
1563          (Eigenface/max(max(Eigenface))); % Obtaining a negative
1564                                           % image for the eigenface
1565                                           % in order to enhance its
1566                                           % appearance.
1567
1568      figure('units','centimeters','position',[15.5 5.5 8 10.5])
1569      subplot(1,1,1)
1570      imshow(uint8(Negative_Eigenface))
1571      title({['Calculated Eigenface No.' num2str(I) .......
1572          ' Associated'];['with Eigenvalue No.' num2str(I) .....
1573          ' by Using'];'PCA Algorithm';'(A Negative Image)'})
1574
1575      disp(['Please, press any keyboard button to explore the '.....
1576          'remaining calculated'])
```

```
1577      disp('eigenfaces by using PCA algorithm >>>>>>>')
1578      pause
1579      clc
1580      close all
1581  end
1582
1583  % Displaying the calculated eigenfaces by using IPCA algorithm.
1584  for I=1:length(Best_Eigenvalues_IPCA)
1585      X=floor(255*(double(Best_Eigenvectors_IPCA(:,I))/max(max(.....
1586          Best_Eigenvectors_IPCA(:,I))))); % Normalized eigenface.
1587                                           % This is done to increase
1588                                           % the dynamic range of the
1589                                           % eigenface for
1590                                           % visualization by scaling
1591                                           % the intensities of the
1592                                           % eigenface from 0 to 255.
1593      Eigenface=reshape(X,N,N);
1594      Negative_Eigenface=255*ones(N,N)-255*.....
1595          (Eigenface/max(max(Eigenface))); % Obtaining a negative
1596                                           % image for the eigenface
1597                                           % in order to enhance its
1598                                           % appearance.
1599
1600      figure('units','centimeters','position',[15.5 5.5 8 10.5])
1601      subplot(1,1,1)
1602      imshow(uint8(Negative_Eigenface))
1603      title({['Calculated Eigenface No.' num2str(I) .......
1604          ' Associated'];['with Eigenvalue No.' num2str(I) .......
1605          ' by Using'];'IPCA Algorithm';'(A Negative Image)'})
1606
1607      disp(['Please, press any keyboard button to explore the '.....
1608          'remaining calculated'])
1609      disp('eigenfaces by using IPCA algorithm >>>>>>>')
1610      pause
1611      clc
1612      close all
1613  end
1614
1615
1616  save('Eigenvectors','Eigenvectors')
1617  save('Best_Eigenvectors_PCA','Best_Eigenvectors_PCA')
1618  save('Best_Eigenvectors_IPCA','Best_Eigenvectors_IPCA')
1619  break
1620
1621
1622  %%
1623  %%%%% The Compression and Reconstruction of the Training Faces %%%%
1624
1625
1626  clc
```

```
close all


%%%%% When a small number of eigenfaces is used to compress and
%%%%% reconstruct the training faces then the processing speed
%%%%% will increase so the IPCA algorithm is the fastest one then
%%%%% PCA algorithm finally the smallest processing speed occurs
%%%%% when all calculated eigenvectors from the covariance matrix
%%%%% for all training faces are used as eigenfaces. When a small
%%%%% number of the eigenfaces is used to project and reconstruct
%%%%% the training faces then the training faces will have bad
%%%%% quality. Therefore, the highest error in reconstruction
%%%%% occurs when the IPCA algorithm is used; then the PCA
%%%%% algorithm comes second; finally, the usage of all
%%%%% eigenvectors as eigenfaces produces the samllest
%%%%% reconstruction error.

% Best_Eigenvectors=Eigenvectors; % Just try it to see how affects
                                  % on the reconstructed training
                                  % faces.
% Best_Eigenvectors=Best_Eigenvectors_PCA; % Just try it to see how
                                           % affects on the
                                           % reconstructed training
                                           % faces.
Best_Eigenvectors=Best_Eigenvectors_IPCA; % Just try it to see how
                                          % affects on the
                                          % reconstructed training
                                          % faces.


% Projecting each training face on the eigenspace. It is just
% expressing each training face in terms of the eigenfaces. This is
% called principal components transform (also called the Hotelling
% or Karhunen–Loéve transform)
All_Known_Transformed_Im=.............
    zeros(size(Best_Eigenvectors,2),........
    Total_No_of_Known_Im); % A 2D matrix where each column
                           % represents the  coordinates of a
                           % projected training face in the
                           % eigenspace.
for r=1:Total_No_of_Known_Im
    All_Known_Transformed_Im(:,r)=Best_Eigenvectors.'*.......
        Known_Im_Subt_Mean(:,r);
end
All_Known_Transformed_Im;


% The reconstruction of the projected training faces.
All_Known_Reconstructed_Im_V=...........
    zeros(N*N,Total_No_of_Known_Im); % An N^2xP, 2D matrix where
```

```
                                        % each column represents a
                                        % reconstructed training face
                                        % vector.
All_Known_Reconstructed_Im=zeros(N,N,Total_No_of_Known_Im);
for a=1:Total_No_of_Known_Im
    Pre=reshape(Best_Eigenvectors*.............
        All_Known_Transformed_Im(:,a),N,N)+.............
        reshape(Av_Image,N,N); % Adding the average training face.

    RC1=Rows_Columns(:,:,a);
    F=RC1(1,:);
    F1=RC1(2,:);
    D=F(F>0);
    D1=F1(F1>0);
    RC=[D;D1]; % The rows and columns for the pixels of a face
               % after removing the added zero rows and columns.
               % The added zero rows and columns are added by
               % MATLAB for making the matrices of the rows and
               % columns of the faces pixels are equal.
    for QQ=1:size(RC,2)
        All_Known_Reconstructed_Im(RC(1,QQ),RC(2,QQ),a)=.......
            Pre(RC(1,QQ),RC(2,QQ))+Means(1,a); % Adding the mean
                                               % of the pixels of
                                               % a face.
    end

    Y=All_Known_Reconstructed_Im(:,:,a); % Removing any pixel less
                                         % than zero in a training
                                         % face because the image
                                         % can not be negative.
    [UU NN]=find(Y<0);
    for CC=1:size(UU,1)
        Y(UU(CC),NN(CC))=0;
    end

    All_Known_Reconstructed_Im_V(:,a)=reshape(Y,N*N,1);

    Known_Image=Known_Images(a).name;
    figure('units','centimeters','position',[16 7 7.5 8.5])
    subplot(1,1,1)
    imshow(uint8(reshape(All_Known_Reconstructed_Im_V(:,a),N,N)))
    if a==1
        title({['Reconstructed Training Face No.' num2str(a)];....
            ['(' Known_Image(1:length(Known_Image)-6) .........
            ', 1^{st} Projection)']})
    elseif a==2
        title({['Reconstructed Training Face No.' num2str(a)];....
            ['(' Known_Image(1:length(Known_Image)-6) ........
            ', 2^{nd} Projection)']})
    elseif a==3
```

```
1727          title({['Reconstructed Training Face No.' num2str(a)];....
1728              ['(' Known_Image(1:length(Known_Image)-6) ........
1729              ', 3^{rd} Projection)']})
1730      elseif a==4
1731          title({['Reconstructed Training Face No.' num2str(a)];....
1732              ['(' Known_Image(1:length(Known_Image)-6) ......
1733              ', 1^{st} Projection)']})
1734      elseif a==5
1735          title({['Reconstructed Training Face No.' num2str(a)];....
1736              ['(' Known_Image(1:length(Known_Image)-6) ......
1737              ', 2^{nd} Projection)']})
1738      elseif a==6
1739          title({['Reconstructed Training Face No.' num2str(a)];....
1740              ['(' Known_Image(1:length(Known_Image)-6) .....
1741              ', 3^{rd} Projection)']})
1742      elseif a==7
1743          title({['Reconstructed Training Face No.' num2str(a)];....
1744              ['(' Known_Image(1:length(Known_Image)-6) ......
1745              ', 1^{st} Projection)']})
1746      elseif a==8
1747          title({['Reconstructed Training Face No.' num2str(a)];....
1748              ['(' Known_Image(1:length(Known_Image)-6) .....
1749              ', 2^{nd} Projection)']})
1750      elseif a==9
1751          title({['Reconstructed Training Face No.' num2str(a)];....
1752              ['(' Known_Image(1:length(Known_Image)-6) ......
1753              ', 3^{rd} Projection)']})
1754      end
1755      disp(['Please, press any keyboard button to explore '......
1756          'the remaining'])
1757      disp('reconstructed training faces >>>>>>>')
1758      pause
1759      close all
1760      clc
1761  end
1762  All_Known_Reconstructed_Im_V;
1763
1764
1765  %%
1766  %%%%%%%% The Calculation of Compression and Reconstruction %%%%%%%%
1767  %%%%%%%%                      Performance                  %%%%%%%%
1768
1769  clc
1770  close all
1771
1772
1773  After_Compression_Normalization=zeros(1,...................
1774      length(Eigenvalues)); % Measuring an information rate after
1775                            % compression. The elements of this
1776                            % vector represent the information
```

```
1777                            % rates after compression with respect
1778                            % to the information rates before
1779                            % compression.
1780 MSE_Compression=.......
1781     zeros(1,length(Eigenvalues)); % Measuring compression
1782                                   % performance or how much
1783                                   % compressed information is.
1784                                   % The elements of this vector
1785                                   % represent the mean squared
1786                                   % errors in compression when
1787                                   % different eigenfaces are
1788                                   % selected.
1789 MSE_Reconstruction=zeros(length(Eigenvalues),..........
1790     Total_No_of_Known_Im); % Measuring reconstruction performance
1791                            % or the quality of a reconstructed
1792                            % training face. The elements of each
1793                            % column of this matrix represent the
1794                            % mean squared errors between a
1795                            % projected training face and its
1796                            % reconstruction when different
1797                            % eigenfaces are selected.
1798 Eigenvalues1=fliplr(Eigenvalues);
1799 Eigenvectors1=flipdim(Eigenvectors,2); % This flipping just to make
1800                                        % the highest correlated
1801                                        % eigenface is associated
1802                                        % with the first eigenvalue
1803                                        % and so on. This is done to
1804                                        % be compatable with the
1805                                        % generated document
1806                                        % "PCA vs. IPCA".
1807 for kk=1:length(Eigenvalues)
1808     ['Iteration No.: ' num2str(kk) ' Out of ' ..........
1809         num2str(length(Eigenvalues))]
1810     Selected_Eigenvectors=Eigenvectors1(:,1:kk);
1811
1812     % Projecting each training face on the eigenspace. It is just
1813     % expressing each training face in terms of the eigenfaces.
1814     % This is called principal components transform (also called
1815     % the Hotelling or Karhunen–Loéve transform)
1816     All_Known_Transformed_Im=.............
1817         zeros(size(Selected_Eigenvectors,2),........
1818         Total_No_of_Known_Im); % A 2D matrix where each column
1819                                % represents the coordinates of a
1820                                % projected training face in the
1821                                % eigenspace.
1822     for r=1:Total_No_of_Known_Im
1823         All_Known_Transformed_Im(:,r)=Selected_Eigenvectors.'*.....
1824             Known_Im_Subt_Mean(:,r);
1825     end
1826     All_Known_Transformed_Im;
```

```
1827
1828      % The calculation of an information ratio before and after
1829      % compression.
1830      Before_Compression=Total_No_of_Known_Im*.....
1831          N*N; % The overall information when there is
1832               % no any compression technique is used.
1833      After_Compression=N*N*kk+kk*Total_No_of_Known_Im+.............
1834          N*N; % The overall information when a compression technique
1835               % is used. The overall information here is controlled
1836               % by the selected number of eigenvectors. When the
1837               % selected number is small then the information will
1838               % be samll and vice versa. It is very important to
1839               % notice that when all eigenvectors are used then
1840               % there will not be any compression and the overall
1841               % information when there is no any compression
1842               % technique is used will be the optimum one. Also, it
1843               % is really important to notice that the rows and
1844               % columns for the pixels of each face as well as the
1845               % means of the faces pixels are not added to the
1846               % overall information after compression that because
1847               % the face centering operation is not really important
1848               % for the image compression process and will not have
1849               % any effect if it is done or not but it has been done
1850               % here because it is important for other processes.
1851      Before_Compression_Normalization=(Before_Compression/.......
1852          Before_Compression)*100; % Note that, the normalization is
1853                                   % done to make the overall
1854                                   % information before compression
1855                                   % is equal to 100 all the time
1856                                   % in order to make comparison
1857                                   % easier.
1858      After_Compression_Normalization(1,kk)=(After_Compression/.....
1859          Before_Compression)*100; % Note that, the normalization is
1860                                   % done to make the overall
1861                                   % information before compression
1862                                   % is equal to 100 all the time
1863                                   % in order to make comparison
1864                                   % easier.
1865  %     The_Information_Ratio=[..............
1866  %         num2str(Before_Compression_Normalization) ............
1867  %         '% (Before Compression) : ' ..............
1868  %         num2str(After_Compression_Normalization(1,kk))....
1869  %         '% (After Compression)'] % This is just to make the
1870  %                                  % information ratio is readable
1871  %                                  % on the command window.
1872
1873      % The reconstruction of the projected training faces.
1874      All_Known_Reconstructed_Im=zeros(N,N,Total_No_of_Known_Im);
1875      for a=1:Total_No_of_Known_Im
1876          Pre=reshape(Selected_Eigenvectors*...........
```

```
1877                All_Known_Transformed_Im(:,a),N,N)+.........
1878                reshape(Av_Image,N,N); % Adding the average
1879                                       % training face.
1880
1881            RC1=Rows_Columns(:,:,a);
1882            F=RC1(1,:);
1883            F1=RC1(2,:);
1884            D=F(F>0);
1885            D1=F1(F1>0);
1886            RC=[D;D1]; % The rows and columns for the pixels of a
1887                       % face after removing the added zero rows and
1888                       % columns. The added zero rows and columns are
1889                       % added by MATLAB for making the matrices of
1890                       % the rows and columns of the faces pixels
1891                       % are equal.
1892            for QQ=1:size(RC,2)
1893                All_Known_Reconstructed_Im(RC(1,QQ),RC(2,QQ),a)=......
1894                    Pre(RC(1,QQ),RC(2,QQ))+..........
1895                    Means(1,a); % Adding the mean of the
1896                                % pixels of a face.
1897            end
1898
1899            Y=All_Known_Reconstructed_Im(:,:,a); % Removing any pixel
1900                                                 % less than zero in
1901                                                 % a training face
1902                                                 % because the image
1903                                                 % can not be negative.
1904            [UU NN]=find(Y<0);
1905            for CC=1:size(UU,1)
1906                Y(UU(CC),NN(CC))=0;
1907            end
1908
1909            MSE_Compression(1,kk)=sum(Eigenvalues1(1,kk+1:end));
1910            MSE_Reconstruction(kk,a)=.........
1911                sum(sum(...........
1912                (reshape(Normalized_Known_Im_V(:,a),N,N)-Y).^2))/N*N;
1913
1914  %           % Displaying the effect of the selected eigenvectors on
1915  %           % the reconstructed training faces.
1916  %           Known_Image=Known_Images(a).name;
1917  %           figure('units','centimeters','position',[16 7 7.5 8.5])
1918  %           subplot(3,1,1)
1919  %           imshow(uint8(reshape(Normalized_Known_Im_V(:,a),N,N)))
1920  %           if a==1
1921  %               title({['Original Training Face No.' num2str(a)];....
1922  %                   ['(' Known_Image(1:length(Known_Image)-6) .....
1923  %                   ', 1^{st} Projection)']})
1924  %           elseif a==2
1925  %               title({['Original Training Face No.' num2str(a)];....
1926  %                   ['(' Known_Image(1:length(Known_Image)-6) ....
```

```
%                 ', 2^{nd} Projection)']})
%         elseif a==3
%             title({['Original Training Face No.' num2str(a)];....
%                 ['(' Known_Image(1:length(Known_Image)-6) .....
%                 ', 3^{rd} Projection)']})
%         elseif a==4
%             title({['Original Training Face No.' num2str(a)];....
%                 ['(' Known_Image(1:length(Known_Image)-6) ......
%                 ', 1^{st} Projection)']})
%         elseif a==5
%             title({['Original Training Face No.' num2str(a)];....
%                 ['(' Known_Image(1:length(Known_Image)-6) ......
%                 ', 2^{nd} Projection)']})
%         elseif a==6
%             title({['Original Training Face No.' num2str(a)];....
%                 ['(' Known_Image(1:length(Known_Image)-6) .....
%                 ', 3^{rd} Projection)']})
%         elseif a==7
%             title({['Original Training Face No.' num2str(a)];....
%                 ['(' Known_Image(1:length(Known_Image)-6) ......
%                 ', 1^{st} Projection)']})
%         elseif a==8
%             title({['Original Training Face No.' num2str(a)];....
%                 ['(' Known_Image(1:length(Known_Image)-6) .....
%                 ', 2^{nd} Projection)']})
%         elseif a==9
%             title({['Original Training Face No.' num2str(a)];....
%                 ['(' Known_Image(1:length(Known_Image)-6) ......
%                 ', 3^{rd} Projection)']})
%         end
%         subplot(3,1,2)
%         imshow(uint8(Y))
%         if a==1
%             title(['Its Reconstruction When q=' num2str(kk)])
%         elseif a==2
%             title(['Its Reconstruction When q=' num2str(kk)])
%         elseif a==3
%             title(['Its Reconstruction When q=' num2str(kk)])
%         elseif a==4
%             title(['Its Reconstruction When q=' num2str(kk)])
%         elseif a==5
%             title(['Its Reconstruction When q=' num2str(kk)])
%         elseif a==6
%             title(['Its Reconstruction When q=' num2str(kk)])
%         elseif a==7
%             title(['Its Reconstruction When q=' num2str(kk)])
%         elseif a==8
%             title(['Its Reconstruction When q=' num2str(kk)])
%         elseif a==9
%             title(['Its Reconstruction When q=' num2str(kk)])
```

```
1977 %          end
1978 %          subplot(3,1,3)
1979 %          imshow(uint8(.........
1980 %              (reshape(Normalized_Known_Im_V(:,a),N,N)-Y).^2))
1981 %          if a==1
1982 %              title('The Squared Error')
1983 %          elseif a==2
1984 %              title('The Squared Error')
1985 %          elseif a==3
1986 %              title('The Squared Error')
1987 %          elseif a==4
1988 %              title('The Squared Error')
1989 %          elseif a==5
1990 %              title('The Squared Error')
1991 %          elseif a==6
1992 %              title('The Squared Error')
1993 %          elseif a==7
1994 %              title('The Squared Error')
1995 %          elseif a==8
1996 %              title('The Squared Error')
1997 %          elseif a==9
1998 %              title('The Squared Error')
1999 %          end
2000 %
2001 %          kk
2002 %          pause
2003 %          close all
2004     end
2005 end
2006
2007
2008 % Plotting the information rates when different eigenfaces are
2009 % selected compared with the original information rate for all
2010 % training faces and explaining the information rates for the PCA
2011 % and IPCA algorithms on the plot.
2012 clc
2013 figure('units','centimeters','position',[0.15 1.2 35.8 16.9])
2014 subplot(2,1,1)
2015 Leg1=plot(1:length(Eigenvalues),After_Compression_Normalization);
2016 Leg2=line([0 length(Eigenvalues)],[100 100],'Color',[0 102/255 0]);
2017 hold on
2018 Leg3=plot(1,After_Compression_Normalization(1),'kd',.....
2019     'LineWidth',1.5,'MarkerEdgeColor','k','MarkerFaceColor',.....
2020     'm','MarkerSize',8);
2021 text(-50,After_Compression_Normalization(1)+3650,.....
2022     {'\fontsize{10} \color{black}' .......
2023     After_Compression_Normalization(1)})
2024 hold on
2025 Leg4=plot(length(Eigenvalues),.........
2026     After_Compression_Normalization(length(Eigenvalues)),'kd',.....
```

```
2027     'LineWidth',1.5,'MarkerEdgeColor','k','MarkerFaceColor',.....
2028     'g','MarkerSize',8);
2029 text(length(Eigenvalues)-140,............
2030     After_Compression_Normalization(length(Eigenvalues))+2350,....
2031     {'\fontsize{10} \color{black}' .........
2032     After_Compression_Normalization(length(Eigenvalues))})
2033 hold off
2034 axis([-100 length(Eigenvalues)+(length(Eigenvalues)/40) -2000 .....
2035     max(After_Compression_Normalization)+........
2036     (max(After_Compression_Normalization)/10)])
2037 set(gca,'XTick',[1 length(Eigenvalues)/2 length(Eigenvalues)])
2038 title({['The Information Rates When Different Eigenfaces '......
2039     'Are Selected Compared '];.......
2040     'With the Original Information Rate for All Training Faces'})
2041 xlabel('The Number of Selected Eigenfaces')
2042 ylabel('The Information Rate (%)')
2043 legend([Leg1 Leg2 Leg3 Leg4],['The Information Rates When '.......
2044     'Different Eigenfaces Are Selected.'],['The Information '.....
2045     'Rate for All Training Faces.'],['The Information Rate '.....
2046     'When the First Eigenface Is Selected.'],['The '.......
2047     'Information Rate When All Eigenfaces Are Selected.'],.....
2048     'Location','NorthWest');
2049 subplot(2,1,2)
2050 Leg1=plot(1:length(Eigenvalues),After_Compression_Normalization);
2051 axis([0 20 0 200])
2052 Leg2=line([0 length(Eigenvalues)],[100 100],'Color',[0 102/255 0]);
2053 Leg3=line([length(Best_Eigenvalues_PCA) ......
2054     length(Best_Eigenvalues_PCA)],[0 .......
2055     After_Compression_Normalization(length(.......
2056     Best_Eigenvalues_PCA))],'LineStyle','--','Color','k',....
2057     'LineWidth',3);
2058 line([0 length(Best_Eigenvalues_PCA)],.....
2059     [After_Compression_Normalization(.......
2060     length(Best_Eigenvalues_PCA)) ......
2061     After_Compression_Normalization(......
2062     length(Best_Eigenvalues_PCA))],......
2063     'LineStyle','--','Color','k','LineWidth',3)
2064 text(0.1,After_Compression_Normalization(.........
2065     length(Best_Eigenvalues_PCA))+16,......
2066     {'\fontsize{10} \color{black}' ........
2067     num2str(After_Compression_Normalization(.......
2068     length(Best_Eigenvalues_PCA)),'%6.4f')})
2069 Leg4=line([length(Best_Eigenvalues_IPCA) ........
2070     length(Best_Eigenvalues_IPCA)],[0 ........
2071     After_Compression_Normalization(.......
2072     length(Best_Eigenvalues_IPCA))],'LineStyle','--','Color',....
2073     'r','LineWidth',4);
2074 line([0 length(Best_Eigenvalues_IPCA)],......
2075     [After_Compression_Normalization(.....
2076     length(Best_Eigenvalues_IPCA)) .....
```

```
2077      After_Compression_Normalization(.....
2078      length(Best_Eigenvalues_IPCA))],'LineStyle','--','Color',....
2079      'r','LineWidth',4)
2080  text(0.08,After_Compression_Normalization(.....
2081      length(Best_Eigenvalues_IPCA))-1.7,....
2082      {'\fontsize{10} \color{red} \bf' .......
2083      num2str(After_Compression_Normalization(.....
2084      length(Best_Eigenvalues_IPCA)),'%6.4f')})
2085  vv=[length(Best_Eigenvalues_IPCA) length(Best_Eigenvalues_PCA)];
2086  if vv(1)==1 || vv(2)==1
2087      set(gca,'XTick',[0 sort(vv) 20])
2088  else set(gca,'XTick',[0 1 sort(vv) 20])
2089  end
2090  title(['Explaining the Information Rates for the PCA and '.....
2091      'IPCA Algorithms on the Plot'])
2092  xlabel('The Number of Selected Eigenfaces')
2093  ylabel('The Information Rate (%)')
2094  legend([Leg1 Leg2 Leg3 Leg4],['The Information Rates When '.....
2095      'Different Eigenfaces Are Selected.'],['The Information '....
2096      'Rate for All Training Faces.'],['The Information Rate '......
2097      'When PCA Algorithm Is Used.'],['The Information '.....
2098      'Rate When IPCA Algorithm Is Used.'],'Location','SouthEast');
2099  disp('Please, press any keyboard button to resume the code >>>>')
2100  pause
2101  clc
2102  close all
2103
2104
2105  % Plotting the mean squared errors of compression for different
2106  % selected eigenfaces compared with the mean squared errors for
2107  % the PCA and IPCA algorithms.
2108  figure('units','centimeters','position',[0.15 1.2 35.8 16.9])
2109  subplot(2,1,1)
2110  Leg1=plot(1:length(Eigenvalues),MSE_Compression);
2111  hold on
2112  Leg2=plot(1,MSE_Compression(1),'kd','LineWidth',1.5,......
2113      'MarkerEdgeColor','k','MarkerFaceColor','m','MarkerSize',8);
2114  text(25,MSE_Compression(1)+(10^7/11),.......
2115      {'\fontsize{10} \color{black}' MSE_Compression(1)})
2116  hold on
2117  Leg3=plot(length(Eigenvalues),......
2118      MSE_Compression(length(Eigenvalues)),'kd','LineWidth',1.5,....
2119      'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSize',8);
2120  text(length(Eigenvalues)-8,........
2121      MSE_Compression(length(Eigenvalues))+10^7/3.2,.......
2122      {'\fontsize{10} \color{black}' ........
2123      MSE_Compression(length(Eigenvalues))})
2124  hold off
2125  axis([0 length(Eigenvalues)+(length(Eigenvalues)/40) ......
2126      -max(MSE_Compression)/10 .....
```

```
2127     max(MSE_Compression)+(max(MSE_Compression)/10)])
2128 set(gca,'XTick',[1 length(Eigenvalues)/2 length(Eigenvalues)])
2129 title(['The Mean Squared Errors of Compression for Different '....
2130     'Selected Eigenfaces'])
2131 xlabel('The Number of Selected Eigenfaces')
2132 ylabel('The Mean Squared Error')
2133 legend([Leg1 Leg2 Leg3],['The Mean Squared Errors of '.....
2134     'Compression for Different Selected Eigenfaces.'],....
2135     ['The Mean Squared Error of Compression When the First '.....
2136     'Eigenface Is Selected.'],['The Mean Squared Error of '.....
2137     'Compression When All Eigenfaces Are Selected.'],......
2138     'Location','NorthEast');
2139 subplot(2,1,2)
2140 Leg1=plot(1:length(Eigenvalues),MSE_Compression);
2141 axis([0 20 -10^7/4 max(MSE_Compression)+(10^7/4)])
2142 Leg2=line([length(Best_Eigenvalues_PCA) ........
2143     length(Best_Eigenvalues_PCA)],.......
2144     [0 MSE_Compression(length(Best_Eigenvalues_PCA))],.....
2145     'LineStyle','--','Color','k','LineWidth',3);
2146 line([0 length(Best_Eigenvalues_PCA)],......
2147     [MSE_Compression(length(Best_Eigenvalues_PCA)) ......
2148     MSE_Compression(length(Best_Eigenvalues_PCA))],.......
2149     'LineStyle','--','Color','k','LineWidth',3)
2150 text(0.08,MSE_Compression(length(Best_Eigenvalues_PCA))+.....
2151     (-0.008*10^7),{'\fontsize{10} \color{black}' .....
2152     num2str(MSE_Compression(length(Best_Eigenvalues_PCA)))})
2153 Leg3=line([length(Best_Eigenvalues_IPCA) .......
2154     length(Best_Eigenvalues_IPCA)],[0 .....
2155     MSE_Compression(length(Best_Eigenvalues_IPCA))],....
2156     'LineStyle','--','Color','r','LineWidth',4);
2157 line([0 length(Best_Eigenvalues_IPCA)],.....
2158     [MSE_Compression(length(Best_Eigenvalues_IPCA)) .....
2159     MSE_Compression(length(Best_Eigenvalues_IPCA))],....
2160     'LineStyle','--','Color','r','LineWidth',4)
2161 text(0.08,MSE_Compression(length(Best_Eigenvalues_IPCA))+.....
2162     0.24*10^7,{'\fontsize{10} \color{red} \bf' .......
2163     num2str(MSE_Compression(length(Best_Eigenvalues_IPCA)))})
2164 if vv(1)==1 || vv(2)==1
2165     set(gca,'XTick',[0 sort(vv) 20])
2166 else set(gca,'XTick',[0 1 sort(vv) 20])
2167 end
2168 title(['Explaining the Mean Squared Errors for the PCA and '....
2169     'IPCA Algorithms on the Plot'])
2170 xlabel('The Number of Selected Eigenfaces')
2171 ylabel('The Mean Squared Error')
2172 legend([Leg1 Leg2 Leg3],['The Mean Squared Errors of '.....
2173     'Compression for Different Selected Eigenfaces.'],.....
2174     'The Mean Squared Error for PCA Algorithm.',.....
2175     'The Mean Squared Error for IPCA Algorithm.',.....
2176     'Location','NorthEast');
```

```
disp('Please, press any keyboard button to resume the code >>>>')
pause
clc
close all


% Plotting the mean squared errors between each training face and
% its reconstruction for different selected eigenfaces compared
% with the mean squared errors between each training face and its
% reconstruction for the PCA and IPCA Algorithms.
figure('units','centimeters','position',[0.15 1.2 35.8 16.9])
for jj=1:Total_No_of_Known_Im
    subplot(2,1,1)
    Leg1=plot(1:length(Eigenvalues),(MSE_Reconstruction(:,jj)).');
    hold on
    Leg2=plot(1,MSE_Reconstruction(1,jj),'kd','LineWidth',1.5,....
        'MarkerEdgeColor','k','MarkerFaceColor','m',.....
        'MarkerSize',8);
    text(35,MSE_Reconstruction(1,jj)+........
        max(MSE_Reconstruction(:,jj))/22,.........
        {'\fontsize{10} \color{black}' MSE_Reconstruction(1,jj)})
    hold on
    Leg3=plot(length(Eigenvalues),.......
        MSE_Reconstruction(length(Eigenvalues),jj),'kd',.....
        'LineWidth',1.5,'MarkerEdgeColor','k',.....
        'MarkerFaceColor','g','MarkerSize',8);
    text(length(Eigenvalues)-75,......
        MSE_Reconstruction(length(Eigenvalues),jj)+.....
        max(MSE_Reconstruction(:,jj))/7,.......
        {'\fontsize{10} \color{black}' .....
        MSE_Reconstruction(length(Eigenvalues),jj)})
    hold off
    axis([0 length(Eigenvalues)+(length(Eigenvalues)/15) .....
        -max(MSE_Reconstruction(:,jj))/10 ......
        max(MSE_Reconstruction(:,jj))+........
        (max(MSE_Reconstruction(:,jj))/10)])
    set(gca,'XTick',[1 length(Eigenvalues)/2 length(Eigenvalues)])
    Known_Image=Known_Images(jj).name;
    if jj==1
        title({['The Mean Squared Errors of Reconstructing '....
            'Training'];['Face No.' num2str(jj) .....
            ' for Different Selected Eigenfaces'];.....
            ['(' Known_Image(1:length(Known_Image)-6) .....
            ', 1^{st} Projection)']})
    elseif jj==2
        title({['The Mean Squared Errors of Reconstructing '....
            'Training']; ['Face No.' num2str(jj) .....
            ' for Different Selected Eigenfaces'];.....
            ['(' Known_Image(1:length(Known_Image)-6) .....
            ', 2^{nd} Projection)']})
```

```
    elseif jj==3
        title({['The Mean Squared Errors of Reconstructing '.....
            'Training']; ['Face No.' num2str(jj) .....
            ' for Different Selected Eigenfaces'];.....
            ['(' Known_Image(1:length(Known_Image)-6) .....
            ', 3^{rd} Projection)']})
    elseif jj==4
        title({['The Mean Squared Errors of Reconstructing '.....
            'Training']; ['Face No.' num2str(jj) .....
            ' for Different Selected Eigenfaces'];.....
            ['(' Known_Image(1:length(Known_Image)-6) .....
            ', 1^{st} Projection)']})
    elseif jj==5
        title({['The Mean Squared Errors of Reconstructing '.....
            'Training']; ['Face No.' num2str(jj) ......
            ' for Different Selected Eigenfaces'];....
            ['(' Known_Image(1:length(Known_Image)-6) ....
            ', 2^{nd} Projection)']})
    elseif jj==6
        title({['The Mean Squared Errors of Reconstructing '.....
            'Training']; ['Face No.' num2str(jj) ....
            ' for Different Selected Eigenfaces'];....
            ['(' Known_Image(1:length(Known_Image)-6) ....
            ', 3^{rd} Projection)']})
    elseif jj==7
        title({['The Mean Squared Errors of Reconstructing '....
            'Training']; ['Face No.' num2str(jj) .....
            ' for Different Selected Eigenfaces'];....
            ['(' Known_Image(1:length(Known_Image)-6) .....
            ', 1^{st} Projection)']})
    elseif jj==8
        title({['The Mean Squared Errors of Reconstructing '....
            'Training']; ['Face No.' num2str(jj) ....
            ' for Different Selected Eigenfaces'];.....
            ['(' Known_Image(1:length(Known_Image)-6) .....
            ', 2^{nd} Projection)']})
    elseif jj==9
        title({['The Mean Squared Errors of Reconstructing '....
            'Training']; ['Face No.' num2str(jj) ....
            ' for Different Selected Eigenfaces'];....
            ['(' Known_Image(1:length(Known_Image)-6) .....
            ', 3^{rd} Projection)']})
    end
    xlabel('The Number of Selected Eigenfaces')
    ylabel('The Mean Squared Error')
    Leg=legend([Leg1 Leg2 Leg3],['The Mean Squared Errors of '....
        'Reconstruction for Different Selected Eigenfaces.'],....
        ['The Mean Squared Error of Reconstruction When the '.....
        'First Eigenface Is Selected.'],['The Mean Squared '.....
        'Error of Reconstruction When All Eigenfaces '.....
```

```
2277          'Are Selected.'],'Location','North');
2278      subplot(2,1,2)
2279      MSE_Recons=(MSE_Reconstruction(:,jj)).';
2280      Leg1=plot(1:length(Eigenvalues),MSE_Recons);
2281      axis([0 20 -max(MSE_Recons)/6 ......
2282          max(MSE_Recons)+(max(MSE_Recons)/10)])
2283      Leg2=line([length(Best_Eigenvalues_PCA) .....
2284          length(Best_Eigenvalues_PCA)],[0 .....
2285          MSE_Recons(length(Best_Eigenvalues_PCA))],....
2286          'LineStyle','--','Color','k','LineWidth',3);
2287      line([0 length(Best_Eigenvalues_PCA)],.....
2288          [MSE_Recons(length(Best_Eigenvalues_PCA)) .....
2289          MSE_Recons(length(Best_Eigenvalues_PCA))],......
2290          'LineStyle','--','Color','k','LineWidth',3)
2291      text(0.1,MSE_Recons(length(Best_Eigenvalues_PCA)).....
2292          -(max(MSE_Recons)/88),{'\fontsize{10} \color{black}' .....
2293          num2str(MSE_Recons(length(Best_Eigenvalues_PCA)))})
2294      Leg3=line([length(Best_Eigenvalues_IPCA) .....
2295          length(Best_Eigenvalues_IPCA)],.....
2296          [0 MSE_Recons(length(Best_Eigenvalues_IPCA))],.....
2297          'LineStyle','--','Color','r','LineWidth',4);
2298      line([0 length(Best_Eigenvalues_IPCA)],....
2299          [MSE_Recons(length(Best_Eigenvalues_IPCA)) .....
2300          MSE_Recons(length(Best_Eigenvalues_IPCA))],.....
2301          'LineStyle','--','Color','r','LineWidth',4)
2302      text(0.1,MSE_Recons(length(Best_Eigenvalues_IPCA)).....
2303          +(max(MSE_Recons)/8),{'\fontsize{10} \color{red} \bf' ....
2304          num2str(MSE_Recons(length(Best_Eigenvalues_IPCA)))})
2305      if vv(1)==1 || vv(2)==1
2306          set(gca,'XTick',[0 sort(vv) 20])
2307      else set(gca,'XTick',[0 1 sort(vv) 20])
2308      end
2309      title(['Explaining the Mean Squared Errors for the '.....
2310          'PCA and IPCA Algorithms on the Plot'])
2311      xlabel('The Number of Selected Eigenfaces')
2312      ylabel('The Mean Squared Error')
2313      Leg=legend([Leg1 Leg2 Leg3],['The Mean Squared Errors of '.....
2314          'Reconstruction for Different Selected Eigenfaces.'],....
2315          'The Mean Squared Error for PCA Algorithm.',....
2316          'The Mean Squared Error for IPCA Algorithm.',....
2317          'Location','NorthEast');
2318
2319      disp(['Please, press any keyboard button to explore '....
2320          'the remaining mean'])
2321      disp(['squared errors for other reconstructed training '....
2322          'faces >>>>>>>'])
2323      pause
2324      clc
2325  end
2326  close all
```

```
2327
2328
2329 %%
2330 %%%%%%%%%%%%%%%%%%%% A Face Recognition Process %%%%%%%%%%%%%%%%%%%%
2331
2332
2333 clc
2334 close all
2335
2336
2337 %%%%% When the coordinates number of the projected training faces
2338 %%%%% in the eigenspace increases, the error rate will decrease and
2339 %%%%% vice versa as well as when the number decreases, the
2340 %%%%% processing speed will increase and vice versa. So, the usage
2341 %%%%% of all calculated eigenvectors as eigenfaces will lead to the
2342 %%%%% smallest error rate and biggest processing time then the
2343 %%%%% usage of the calculated eigenfaces by using PCA algorithm
2344 %%%%% finally the usage of the calculated eigenfaces by using IPCA
2345 %%%%% algorithm will lead to the biggest error rate and smallest
2346 %%%%% processing time. It is very important to notice that the
2347 %%%%% calculation of all eigenvectors from the covariance matrix
2348 %%%%% for all training faces is too difficult because the
2349 %%%%% covariance matrix is too big as explained before  so it is
2350 %%%%% impractical to use all eigenvectors as eigenfaces.
2351
2352 % Best_Eigenvectors=Eigenvectors; % Just try it to see how affects
2353                                   % on the results of face
2354                                   % recognition.
2355 Best_Eigenvectors=Best_Eigenvectors_PCA; % Just try it to see how
2356                                         % affects on the results
2357                                         % of face recognition.
2358 % Best_Eigenvectors=Best_Eigenvectors_IPCA; % Just try it to see
2359                                             % how affects on the
2360                                             % results of face
2361                                             % recognition.
2362
2363
2364 % Projecting each training face on the eigenspace. It is just
2365 % expressing each training face in terms of the eigenfaces. This is
2366 % called principal components transform (also called the Hotelling
2367 % or Karhunen–Loéve transform)
2368 All_Known_Transformed_Im=........
2369     zeros(size(Best_Eigenvectors,2),......
2370     Total_No_of_Known_Im); % A 2D matrix where each
2371                            % column represents the
2372                            % coordinates of a
2373                            % projected training
2374                            % face in the eigenspace.
2375 for r=1:Total_No_of_Known_Im
2376     All_Known_Transformed_Im(:,r)=Best_Eigenvectors.'*......
```

```
2377          Known_Im_Subt_Mean(:,r);
2378  end
2379  All_Known_Transformed_Im;
2380
2381
2382  % Projecting each tested image on the eigenspace. It is just
2383  % expressing each tested image in terms of the eigenfaces. This is
2384  % called principal components transform (also called the Hotelling
2385  % or Karhunen–Loéve transform)
2386  All_Tested_Transformed_Im=.....
2387      zeros(size(Best_Eigenvectors,2),....
2388      Total_No_of_Tested_Im); % A 2D matrix where each
2389                              % column represents the
2390                              % coordinates of a
2391                              % projected tested image
2392                              % in the eigenspace.
2393  for r1=1:Total_No_of_Tested_Im
2394      All_Tested_Transformed_Im(:,r1)=Best_Eigenvectors.'*......
2395          Tested_Im_Subt_Mean(:,r1);
2396  end
2397  All_Tested_Transformed_Im;
2398
2399
2400  fid=fopen('Face Recognition Results for Testing.txt',.....
2401      'w'); % A text file for typing
2402            % the results of face
2403            % recognition.
2404  fprintf(fid,['\n\t ***** The Results of Face Recognition '.......
2405      'Obtained from the PCA and IPCA *****\r\n']);
2406  fprintf(fid,['\t\t\t\t***** Code for Testing and Setting up '.....
2407      'Thresholds *****\r\n\n']);
2408  fprintf(fid,['Image No.       The Image Is Originally for   '......
2409      'The Image Is Recognized as   The Status\r\n']);
2410  fprintf(fid,['=============   ==========================   '.....
2411      '==========================   ==========\r\n']);
2412
2413  Distances_Vector=........
2414      zeros(Total_No_of_Tested_Im,.....
2415      Total_No_of_Known_Im); % A 2D matrix where each row
2416                             % represents the distances
2417                             % between the weights of each
2418                             % training face and the
2419                             % weights of one of the
2420                             % tested images.
2421
2422  % The distances between the weights of a training face and the
2423  % weights of each corresponding tested image. Note that, here the
2424  % training face and tested images have the same face and projection
2425  % so these distances must be the smallest. The distances are
2426  % extracted form the Distances_Vector matrix.
```

```
2427 Distances_Vector_P1=zeros(1,Im_P(1));
2428 Distances_Vector_P2=zeros(1,Im_P(2));
2429 Distances_Vector_P3=zeros(1,Im_P(3));
2430 Distances_Vector_P4=zeros(1,Im_P(4));
2431 Distances_Vector_P5=zeros(1,Im_P(5));
2432 Distances_Vector_P6=zeros(1,Im_P(6));
2433 Distances_Vector_P7=zeros(1,Im_P(7));
2434 Distances_Vector_P8=zeros(1,Im_P(8));
2435 Distances_Vector_P9=zeros(1,Im_P(9));
2436
2437 for p=1:Total_No_of_Tested_Im
2438     for q=1:Total_No_of_Known_Im
2439         Distances_Vector(p,q)=.....
2440             norm(All_Tested_Transformed_Im(:,p)-.....
2441             All_Known_Transformed_Im(:,q));
2442     end
2443
2444     if p<=L1
2445         Distances_Vector_P1(1,p)=Distances_Vector(p,1);
2446     elseif p>L1 && p<=L2
2447         Distances_Vector_P2(1,p-L1)=Distances_Vector(p,2);
2448     elseif p>L2 && p<=L3
2449         Distances_Vector_P3(1,p-L2)=Distances_Vector(p,3);
2450     elseif p>L3 && p<=L4
2451         Distances_Vector_P4(1,p-L3)=Distances_Vector(p,4);
2452     elseif p>L4 && p<=L5
2453         Distances_Vector_P5(1,p-L4)=Distances_Vector(p,5);
2454     elseif p>L5 && p<=L6
2455         Distances_Vector_P6(1,p-L5)=Distances_Vector(p,6);
2456     elseif p>L6 && p<=L7
2457         Distances_Vector_P7(1,p-L6)=Distances_Vector(p,7);
2458     elseif p>L7 && p<=L8
2459         Distances_Vector_P8(1,p-L7)=Distances_Vector(p,8);
2460     elseif p>L8 && p<=L9
2461         Distances_Vector_P9(1,p-L8)=Distances_Vector(p,9);
2462     end
2463 end
2464 d1=Distances_Vector_P1;
2465 d2=Distances_Vector_P2;
2466 d3=Distances_Vector_P3;
2467 d4=Distances_Vector_P4;
2468 d5=Distances_Vector_P5;
2469 d6=Distances_Vector_P6;
2470 d7=Distances_Vector_P7;
2471 d8=Distances_Vector_P8;
2472 d9=Distances_Vector_P9;
2473
2474 % The calculation of the mean and standard deviation for each
2475 % vector of the minimum distances and stacking them in a vector for
2476 % the means and another for the standard deviations. This is done
```

```
% for setting up a threshold for face recognition because when a
% distance between a training face and a tested image is the
% smallest with respect to the other training faces, that does not
% mean the tested image is recognized as the training face due to
% the tested image can be different than the training face and has
% the smallest distance in the same time. So, a certain threshold
% must be used to increase the accuracy of recognition.
P_Mean=[mean(d1);mean(d2);mean(d3);mean(d4);mean(d5);mean(d6);.....
    mean(d7);mean(d8);mean(d9)];
P_STD=[std(d1);std(d2);std(d3);std(d4);std(d5);std(d6);std(d7);....
    std(d8);std(d9)];
% save(['Means for Face Recognition When All Eigenvectors '....
%     'Are Used as Eigenfaces'],'P_Mean')
% save(['STDs for Face Recognition When All Eigenvectors '.....
%     'Are Used as Eigenfaces'],'P_STD')
save(['Means for Face Recognition When the Eigenfaces '.....
    'Computed by Using PCA Algorithm Are Used'],'P_Mean')
save(['STDs for Face Recognition When the Eigenfaces '.....
    'Computed by Using PCA Algorithm Are Used'],'P_STD')
% save(['Means for Face Recognition When the Eigenfaces '....
%     'Computed by Using IPCA Algorithm Are Used'],'P_Mean')
% save(['STDs for Face Recognition When the Eigenfaces '.....
%     'Computed by Using IPCA Algorithm Are Used'],'P_STD')

Failures_Vector=zeros(1,......
    Total_No_of_Tested_Im); % A vector for counting the number of
                            % failures in the face recognition
                            % process.

Latex_Matrix=cell(Total_No_of_Tested_Im,4); % This matrix is used
                                            % for creating a table
                                            % in Latex.
for w=1:Total_No_of_Tested_Im
    % The face recognition process.
    for h=1:Total_No_of_Known_Im
        if min(Distances_Vector(w,:))=.......
                Distances_Vector(w,h) && ......
                min(Distances_Vector(w,:))>=......
                (P_Mean(h,1)-P_STD(h,1)) &&.....
                min(Distances_Vector(w,:))<=......
                (P_Mean(h,1)+P_STD(h,1))
            s=transpose(struct2cell(Known_Images));
            c=sortrows(s,1);
            z=c(:,1);
            Recognized_As=char(z(h,1));
            Recognized_Image_Location=.......
                fullfile(Known_Images_Folder,Recognized_As);
            Recognized_image=im2double(rgb2gray(.......
                imread(Recognized_Image_Location)));
            break;
```

```
2527          else
2528              Recognized_As='Unknown Image';
2529              Recognized_image=imread('Unknown_Face.jpg');
2530          end
2531      end
2532
2533      % Defining the face.
2534      if w<=L3
2535          n='Mr. Mansour Alshammari';
2536      elseif L3<w && w<=L6
2537          n='Mr. Methkir Alharthee';
2538      else n='Mr. Mohammed Hanafy';
2539      end
2540
2541      y1=strcmp(Recognized_As(1:length(Recognized_As)-6),....
2542          'Mr. Mansour Alshammari');
2543      y2=strcmp(Recognized_As(1:length(Recognized_As)-6),.....
2544          'Mr. Methkir Alharthee');
2545      y3=strcmp(Recognized_As(1:length(Recognized_As)-6),....
2546          'Mr. Mohammed Hanafy');
2547      f='Success';
2548      if w<=L3 && y1==0;
2549          f='Failure';
2550          Failures_Vector(1,w)=1;
2551      elseif w>L3 && w<=L6 && y2==0;
2552          f='Failure';
2553          Failures_Vector(1,w)=1;
2554      elseif w>L6 && w<=L9 && y3==0;
2555          f='Failure';
2556          Failures_Vector(1,w)=1;
2557      end
2558
2559  %     Tested_Im_Number=[num2str(w) '.jpg'];
2560  %     Tested_Im_Location=.....
2561  %         fullfile(Tested_Images_Folder,Tested_Im_Number);
2562  %     Tested_Im=im2double(rgb2gray(imread(Tested_Im_Location)));
2563  %     subplot(2,1,1)
2564  %     imshow(Tested_Im)
2565  %     title({['Image No.' num2str(w) ' Is Originally for'];n})
2566  %     subplot(2,1,2)
2567  %     imshow(Recognized_image)
2568  %     if y1==1 || y2==1 || y3==1
2569  %         title({'The Image Is Recognized As';.............
2570  %             Recognized_As(1:length(Recognized_As)-6)})
2571  %     else title({'The Image Is Recognized As' .....
2572  %             ['an' blanks(1) Recognized_As]})
2573  %     end
2574
2575      if y1==1 || y2==1 || y3==1
2576          fprintf(fid,['%0.3d\t\t\t    %-22s\t      %-22s\t    '.....
```

```
2577             '%s \r\n\n'],w,n,........
2578             Recognized_As(1:length(Recognized_As)-6),f);
2579         Latex_Matrix(w,1:4)={num2str(w) n ......
2580             Recognized_As(1:length(Recognized_As)-6) f};
2581     else fprintf(fid,.........
2582             '%0.3d\t\t\t    %-22s\t      %-22s\t   %s \r\n\n',....
2583             w,n,Recognized_As,f);
2584         Latex_Matrix(w,1:4)={num2str(w) n Recognized_As f};
2585     end
2586
2587 %     disp(['Please, press Any keyboard button to see another '....
2588 %         'face and its recognition....'])
2589 %     pause
2590 %     clc
2591 end
2592 Total_Number_of_Failures=sum(Failures_Vector); % The total number
2593                                                % of failures in
2594                                                % the face
2595                                                % recognition
2596                                                % process.
2597
2598 fprintf(fid,['=============================================='.....
2599     '=====================================\r\n']);
2600 fprintf(fid,['** The Total Number of Successes: %0.3d out of '.....
2601     '%0.3d (%3.4f%%) \r\n\n'],Total_No_of_Tested_Im-.....
2602     Total_Number_of_Failures,Total_No_of_Tested_Im,....
2603     ((Total_No_of_Tested_Im-Total_Number_of_Failures)/.....
2604     Total_No_of_Tested_Im)*100);
2605 fprintf(fid,['** The Total Number of Failures: %0.3d out of '.....
2606     '%0.3d (%3.4f%%) \r\n\n'],....
2607     Total_Number_of_Failures,Total_No_of_Tested_Im,.......
2608     (Total_Number_of_Failures/Total_No_of_Tested_Im)*100);
2609 fclose(fid);
2610
2611 close all
2612 clc
2613
2614 disp(['Please, see the documented results of face recognition '....
2615     'in the open'])
2616 disp(['text file then press any keyboard button to resume the '....
2617     'code >>>>'])
2618 Text='Face Recognition Results for Testing.txt';
2619 open(Text) % Opening the text file which contains
2620            % the results of face recognition.
2621 pause
2622 clc
2623 open('PCA_IPCA_Testing_and_Setting_up_Thresholds.m')
2624
2625
2626 %%
```

```
%%%%%%%%% The Computation of Face Recognition Performance %%%%%%%%%


clc
close all


Error_Rate_Recognition=.......
    zeros(1,length(Eigenvalues)); % Measuring recognition
                                  % performance. The elements
                                  % of this vector represent
                                  % the error rates in recognition
                                  % when different eigenfaces are
                                  % selected.
Eigenvalues1=fliplr(Eigenvalues);
Eigenvectors1=flipdim(Eigenvectors,2); % This flipping just to make
                                       % the highest correlated
                                       % eigenface is associated
                                       % with the first eigenvalue
                                       % and so on. This is done
                                       % to be compatable with the
                                       % generated document
                                       % "PCA vs. IPCA".
for kk=1:length(Eigenvalues)
    ['Iteration No.: ' num2str(kk) ' Out of ' .......
        num2str(length(Eigenvalues))]
    Selected_Eigenvectors=Eigenvectors1(:,1:kk);


    % Projecting each training face on the eigenspace. It is just
    % expressing each training face in terms of the eigenfaces.
    % This is called principal components transform (also called
    % the Hotelling or Karhunen–Loéve transform)
    All_Known_Transformed_Im=........
        zeros(size(Selected_Eigenvectors,2),......
        Total_No_of_Known_Im); % A 2D matrix where each column
                               % represents the coordinates of
                               % a projected training face in
                               % the eigenspace.
    for r=1:Total_No_of_Known_Im
        All_Known_Transformed_Im(:,r)=Selected_Eigenvectors.'*....
            Known_Im_Subt_Mean(:,r);
    end
    All_Known_Transformed_Im;


    % Projecting each tested image on the eigenspace. It is just
    % expressing each tested image in terms of the eigenfaces. This
    % is called principal components transform (also called the
    % Hotelling or Karhunen–Loéve transform)
```

```
2677      All_Tested_Transformed_Im=.......
2678          zeros(size(Selected_Eigenvectors,2),.....
2679          Total_No_of_Tested_Im); % A 2D matrix where each column
2680                                  % represents the coordinates of
2681                                  % a projected tested image in
2682                                  % the eigenspace.
2683      for r1=1:Total_No_of_Tested_Im
2684          All_Tested_Transformed_Im(:,r1)=.............
2685              Selected_Eigenvectors.'*Tested_Im_Subt_Mean(:,r1);
2686      end
2687      All_Tested_Transformed_Im;
2688
2689
2690      fid=fopen(['.\Face Recognition Performance\Recognition '.....
2691          'Results for Different Eigenfaces\Results for Testing '....
2692          'When q=' num2str(kk) '.txt'],'w'); % A text file for
2693                                              % typing the results
2694                                              % of face recognition.
2695      fprintf(fid,['\n\t ***** The Results of Face Recognition '....
2696          'When q=' num2str(kk) ' Obtained from the *****\r\n']);
2697      fprintf(fid,['\t   ***** PCA and IPCA Code for Testing '....
2698          'and Setting up Thresholds *****\r\n\n']);
2699      fprintf(fid,['Image No.       The Image Is Originally '......
2700          'for   The Image Is Recognized as   The Status\r\n']);
2701      fprintf(fid,['=============   ==========================='.....
2702          '   ========================   ==========\r\n']);
2703
2704      Distances_Vector=........
2705          zeros(Total_No_of_Tested_Im,......
2706          Total_No_of_Known_Im); % A 2D matrix where
2707                                 % each row represents
2708                                 % the distances between
2709                                 % the weights of each
2710                                 % training face and the
2711                                 % weights of one of the
2712                                 % tested images.
2713
2714      % The distances between the weights of a training face and the
2715      % weights of each corresponding tested image. Note that, here
2716      % the training face and tested images have the same face and
2717      % projection so these distances must be the smallest. The
2718      % distances are extracted form the Distances_Vector matrix.
2719      Distances_Vector_P1=zeros(1,Im_P(1));
2720      Distances_Vector_P2=zeros(1,Im_P(2));
2721      Distances_Vector_P3=zeros(1,Im_P(3));
2722      Distances_Vector_P4=zeros(1,Im_P(4));
2723      Distances_Vector_P5=zeros(1,Im_P(5));
2724      Distances_Vector_P6=zeros(1,Im_P(6));
2725      Distances_Vector_P7=zeros(1,Im_P(7));
2726      Distances_Vector_P8=zeros(1,Im_P(8));
```

```
2727      Distances_Vector_P9=zeros(1,Im_P(9));
2728
2729      for p=1:Total_No_of_Tested_Im
2730          for q=1:Total_No_of_Known_Im
2731              Distances_Vector(p,q)=.......
2732                  norm(All_Tested_Transformed_Im(:,p)-.....
2733                  All_Known_Transformed_Im(:,q));
2734          end
2735
2736          if p<=L1
2737              Distances_Vector_P1(1,p)=Distances_Vector(p,1);
2738          elseif p>L1 && p<=L2
2739              Distances_Vector_P2(1,p-L1)=Distances_Vector(p,2);
2740          elseif p>L2 && p<=L3
2741              Distances_Vector_P3(1,p-L2)=Distances_Vector(p,3);
2742          elseif p>L3 && p<=L4
2743              Distances_Vector_P4(1,p-L3)=Distances_Vector(p,4);
2744          elseif p>L4 && p<=L5
2745              Distances_Vector_P5(1,p-L4)=Distances_Vector(p,5);
2746          elseif p>L5 && p<=L6
2747              Distances_Vector_P6(1,p-L5)=Distances_Vector(p,6);
2748          elseif p>L6 && p<=L7
2749              Distances_Vector_P7(1,p-L6)=Distances_Vector(p,7);
2750          elseif p>L7 && p<=L8
2751              Distances_Vector_P8(1,p-L7)=Distances_Vector(p,8);
2752          elseif p>L8 && p<=L9
2753              Distances_Vector_P9(1,p-L8)=Distances_Vector(p,9);
2754          end
2755      end
2756      d1=Distances_Vector_P1;
2757      d2=Distances_Vector_P2;
2758      d3=Distances_Vector_P3;
2759      d4=Distances_Vector_P4;
2760      d5=Distances_Vector_P5;
2761      d6=Distances_Vector_P6;
2762      d7=Distances_Vector_P7;
2763      d8=Distances_Vector_P8;
2764      d9=Distances_Vector_P9;
2765
2766      % The calculation of the mean and standard deviation for each
2767      % vector of the minimum distances and stacking them in a vector
2768      % for the means and another for the standard deviations. This
2769      % is done for setting up a threshold for face recognition
2770      % because when a distance between a training face and a tested
2771      % image is the smallest with respect to the other training
2772      % faces, that does not mean the tested image is recognized as
2773      % the training face due to the tested image can be different
2774      % than the training face and has the smallest distance in the
2775      % same time. So, a certain threshold must be used to increase
2776      % the accuracy of recognition.
```

```
2777      P_Mean=[mean(d1);mean(d2);mean(d3);mean(d4);mean(d5);....
2778          mean(d6);mean(d7);mean(d8);mean(d9)];
2779      P_STD=[std(d1);std(d2);std(d3);std(d4);std(d5);std(d6);....
2780          std(d7);std(d8);std(d9)];
2781      save([cd '\Face Recognition Performance\Means and STDs '.....
2782          'for Different Eigenfaces\Means for Face Recognition '.....
2783          'When q=' num2str(kk)],'P_Mean')
2784      save([cd '\Face Recognition Performance\Means and STDs '.....
2785          'for Different Eigenfaces\STDs for Face Recognition '.....
2786          'When q=' num2str(kk)],'P_STD')
2787
2788      Failures_Vector=zeros(1,......
2789          Total_No_of_Tested_Im); % A vector for counting the
2790                                  % number of failures in the
2791                                  % face recognition process.
2792      for w=1:Total_No_of_Tested_Im
2793          % The face recognition process.
2794          for h=1:Total_No_of_Known_Im
2795              if min(Distances_Vector(w,:))==.....
2796                      Distances_Vector(w,h) && .....
2797                      min(Distances_Vector(w,:))>=.....
2798                      (P_Mean(h,1)-P_STD(h,1)) &&.....
2799                      min(Distances_Vector(w,:))<=....
2800                      (P_Mean(h,1)+P_STD(h,1))
2801                  s=transpose(struct2cell(Known_Images));
2802                  c=sortrows(s,1);
2803                  z=c(:,1);
2804                  Recognized_As=char(z(h,1));
2805                  Recognized_Image_Location=....
2806                      fullfile(Known_Images_Folder,Recognized_As);
2807                  Recognized_image=im2double(rgb2gray(......
2808                      imread(Recognized_Image_Location)));
2809                  break;
2810              else
2811                  Recognized_As='Unknown Image';
2812                  Recognized_image=imread('Unknown_Face.jpg');
2813              end
2814          end
2815
2816          % Defining the face.
2817          if w<=L3
2818              n='Mr. Mansour Alshammari';
2819          elseif L3<w && w<=L6
2820              n='Mr. Methkir Alharthee';
2821          else n='Mr. Mohammed Hanafy';
2822          end
2823
2824          y1=strcmp(Recognized_As(1:length(Recognized_As)-6),....
2825              'Mr. Mansour Alshammari');
2826          y2=strcmp(Recognized_As(1:length(Recognized_As)-6),.....
```

```
2827            'Mr. Methkir Alharthee');
2828        y3=strcmp(Recognized_As(1:length(Recognized_As)-6),....
2829            'Mr. Mohammed Hanafy');
2830        f='Success';
2831        if w<=L3 && y1==0;
2832            f='Failure';
2833            Failures_Vector(1,w)=1;
2834        elseif w>L3 && w<=L6 && y2==0;
2835            f='Failure';
2836            Failures_Vector(1,w)=1;
2837        elseif w>L6 && w<=L9 && y3==0;
2838            f='Failure';
2839            Failures_Vector(1,w)=1;
2840        end
2841
2842 %        clc
2843 %        Tested_Im_Number=[num2str(w) '.jpg'];
2844 %        Tested_Im_Location=.....
2845 %            fullfile(Tested_Images_Folder,Tested_Im_Number);
2846 %        Tested_Im=.....
2847 %            im2double(rgb2gray(imread(Tested_Im_Location)));
2848 %        subplot(2,1,1)
2849 %        imshow(Tested_Im)
2850 %        title({['Image No.' num2str(w) ' Is Originally for'];n})
2851 %        subplot(2,1,2)
2852 %        imshow(Recognized_image)
2853 %        if y1==1 || y2==1 || y3==1
2854 %            title({'The Image Is Recognized As';.......
2855 %                Recognized_As(1:length(Recognized_As)-6);.....
2856 %                ['When q=' num2str(kk)]})
2857 %        else title({'The Image Is Recognized As';......
2858 %                ['an' blanks(1) Recognized_As];.....
2859 %                ['When q=' num2str(kk)]})
2860 %        end
2861
2862        if y1==1 || y2==1 || y3==1
2863            fprintf(fid,['%0.3d\t\t\t    %-22s\t      '.....
2864                '%-22s\t   %s \r\n\n'],......
2865                w,n,Recognized_As(1:length(Recognized_As)-6),f);
2866        else fprintf(fid,['%0.3d\t\t\t    %-22s\t      '.....
2867                '%-22s\t   %s \r\n\n'],.....
2868                w,n,Recognized_As,f);
2869        end
2870
2871 %        disp(['Please, press Any keyboard button to see '.....
2872 %            'another face and its recognition when q=' .....
2873 %            num2str(kk) ' .......'])
2874 %        pause
2875 %        clc
2876    end
```

```
2877      Total_Number_of_Failures=......
2878          sum(Failures_Vector); % The total number of failures in
2879                                % the face recognition process.
2880
2881      Error_Rate_Recognition(1,kk)=(Total_Number_of_Failures/......
2882          Total_No_of_Tested_Im)*100;
2883
2884      fprintf(fid,['=========================================='.....
2885          '==========================================\r\n']);
2886      fprintf(fid,['** The Total Number of Successes: %0.3d out '....
2887          'of %0.3d (%3.4f%%) \r\n\n'],.......
2888          Total_No_of_Tested_Im-Total_Number_of_Failures,.....
2889          Total_No_of_Tested_Im,......
2890          ((Total_No_of_Tested_Im-Total_Number_of_Failures)/.....
2891          Total_No_of_Tested_Im)*100);
2892      fprintf(fid,['** The Total Number of Failures: %0.3d out '.....
2893          'of %0.3d (%3.4f%%) \r\n\n'],Total_Number_of_Failures,....
2894          Total_No_of_Tested_Im,(Total_Number_of_Failures/.....
2895          Total_No_of_Tested_Im)*100);
2896      fclose(fid);
2897
2898  %     close all
2899  %     clc
2900  %
2901  %     disp(['Please, see the documented results of face '.......
2902  %         'recognition when q=' num2str(kk) ' in the open'])
2903  %     disp(['text file then press any keyboard button to '.....
2904  %         'resume the code >>>>'])
2905  %     Text=['.\Face Recognition Performance\Testing Face '.....
2906  %         'Recognition Results for Different Eigenfaces\Face ' ....
2907  %         'Recognition Results for Testing When q=' .....
2908  %         num2str(kk) '.txt'];
2909  %     open(Text) % Opening the text file which contains
2910  %                % the results of face recognition.
2911  %     pause
2912  %     clc
2913  %     open('PCA_IPCA_Testing_and_Setting_up_Thresholds.m')
2914  end
2915
2916
2917  % Plotting the error rates of face recognition for different
2918  % selected eigenfaces compared with the error rates for PCA and
2919  % IPCA algorithms.
2920  clc
2921  figure('units','centimeters','position',[0.15 1.2 35.8 16.9])
2922  subplot(2,1,1)
2923  Leg1=plot(1:length(Eigenvalues),Error_Rate_Recognition);
2924  hold on
2925  Leg2=plot(1,Error_Rate_Recognition(1),'kd','LineWidth',1.5,.....
2926      'MarkerEdgeColor','k','MarkerFaceColor','m','MarkerSize',8);
```

```
text(25,Error_Rate_Recognition(1)+2,{........
    '\fontsize{10} \color{black}' Error_Rate_Recognition(1)})
hold on
Leg3=plot(length(Eigenvalues),........
    Error_Rate_Recognition(length(Eigenvalues)),'kd',.....
    'LineWidth',1.5,'MarkerEdgeColor','k','MarkerFaceColor',.....
    'g','MarkerSize',8);
text(length(Eigenvalues)-60,........
    Error_Rate_Recognition(length(Eigenvalues))+8,.....
    {'\fontsize{10} \color{black}' .......
    Error_Rate_Recognition(length(Eigenvalues))})
hold off
axis([0 length(Eigenvalues)+(length(Eigenvalues)/40) ......
    -max(Error_Rate_Recognition)/10 .......
    max(Error_Rate_Recognition)+(max(Error_Rate_Recognition)/10)])
set(gca,'XTick',[1 length(Eigenvalues)/2 length(Eigenvalues)])
title(['The Error Rates of Recognition for Different '......
    'Selected Eigenfaces'])
xlabel('The Number of Selected Eigenfaces')
ylabel('The Error Rate (%)')
legend([Leg1 Leg2 Leg3],['The Error Rates of Recognition for '....
    'Different Selected Eigenfaces.'],...................
    ['The Error Rate of Recognition When the First Eigenface '....
    'Is Selected.'],['The Error Rate of Recognition When All '....
    'Eigenfaces Are Selected.'],'Location','NorthEast');
subplot(2,1,2)
Leg1=plot(1:length(Eigenvalues),Error_Rate_Recognition);
axis([0 20 0 ............
    max(Error_Rate_Recognition)+(max(Error_Rate_Recognition)/10)])
Leg2=line([length(Best_Eigenvalues_PCA) .....
    length(Best_Eigenvalues_PCA)],...
    [0 Error_Rate_Recognition(length(Best_Eigenvalues_PCA))],....
    'LineStyle','--','Color','r','LineWidth',4);
line([0 length(Best_Eigenvalues_PCA)],.....
    [Error_Rate_Recognition(length(Best_Eigenvalues_PCA)) .....
    Error_Rate_Recognition(length(Best_Eigenvalues_PCA))],....
    'LineStyle','--','Color','r','LineWidth',4)
text(length(Best_Eigenvalues_PCA)-0.8,.....
    Error_Rate_Recognition(length(Best_Eigenvalues_PCA))+5.7,.....
    {'\fontsize{10} \color{red} \bf' ......
    num2str(Error_Rate_Recognition(length(Best_Eigenvalues_PCA)))})
Leg3=line([length(Best_Eigenvalues_IPCA) ......
    length(Best_Eigenvalues_IPCA)],......
    [0 Error_Rate_Recognition(length(Best_Eigenvalues_IPCA))],.....
    'LineStyle','--','Color','k','LineWidth',3);
line([0 length(Best_Eigenvalues_IPCA)],.....
    [Error_Rate_Recognition(length(Best_Eigenvalues_IPCA)) .....
    Error_Rate_Recognition(length(Best_Eigenvalues_IPCA))],....
    'LineStyle','--','Color','k','LineWidth',3)
text(0.3,........
```

```
2977      Error_Rate_Recognition(length(Best_Eigenvalues_IPCA))+6,.....
2978      {'\fontsize{10} \color{black}' .....
2979      num2str(.......
2980      Error_Rate_Recognition(length(Best_Eigenvalues_IPCA)))})
2981  vv=[length(Best_Eigenvalues_IPCA) length(Best_Eigenvalues_PCA)];
2982  if vv(1)==1 || vv(2)==1
2983      set(gca,'XTick',[0 sort(vv) 20])
2984  else set(gca,'XTick',[0 1 sort(vv) 20])
2985  end
2986  title(['Explaining the Error Rates for the PCA and IPCA '.......
2987      'Algorithms on the Plot'])
2988  xlabel('The Number of Selected Eigenfaces')
2989  ylabel('The Error Rate (%)')
2990  legend([Leg1 Leg2 Leg3],['The Error Rates of Recognition for '....
2991      'Different Selected Eigenfaces.'],.....
2992      'The Error Rate for PCA Algorithm.',......
2993      'The Error Rate for IPCA Algorithm.','Location','NorthEast');
2994
2995
2996  %%
2997  %%%%%%%%%%%%%%%%%%%%%% A Face Detection Process %%%%%%%%%%%%%%%%%%%%%%
2998
2999
3000  clc
3001  close all
3002
3003
3004  %%%%% When we only select the eigenfaces which contain the most
3005  %%%%% significant patterns from the correlated training faces then
3006  %%%%% the accuracy of face detection and processing speed will
3007  %%%%% increase. That because in face detection, distance
3008  %%%%% calculation will be between a pre—processed unknown image and
3009  %%%%% its reconstruction so if a reconstructed image is not a face
3010  %%%%% then it will have a big distance hence it will not be
3011  %%%%% detected as a face image. As a result of that, the usage of
3012  %%%%% the calculated eigenfaces by using IPCA algorithm will
3013  %%%%% produce the smallest error rate and processing time then the
3014  %%%%% usage of the calculated eigenfaces by using PCA algorithm
3015  %%%%% comes second finally the usage of all calculated eigenvectors
3016  %%%%% as eigenfaces will produce the biggest error rate and
3017  %%%%% processing time. It is very important to notice that the
3018  %%%%% calculation of all eigenvectors from the covariance matrix
3019  %%%%% for all training faces is too difficult because the
3020  %%%%% covariance matrix is too big as explained before so it is
3021  %%%%% impractical to use all eigenvectors as eigenfaces.
3022
3023  % Best_Eigenvectors=Eigenvectors; % Just try it to see how
3024                                    % affects on the results
3025                                    % of face detection.
3026  % Best_Eigenvectors=Best_Eigenvectors_PCA; % Just try it to see how
```

```
3027                                                % affects on the results
3028                                                % of face detection.
3029 Best_Eigenvectors=Best_Eigenvectors_IPCA; % Just try it to see how
3030                                               % affects on the results
3031                                               % of face detection.
3032
3033
3034 % Projecting each training face on the eigenspace. It is just
3035 % expressing each training face in terms of the eigenfaces. This is
3036 % called principal components transform (also called the Hotelling
3037 % or Karhunen–Loéve transform)
3038 All_Known_Transformed_Im=.......
3039     zeros(size(Best_Eigenvectors,2),......
3040     Total_No_of_Known_Im); % A 2D matrix where each column
3041                            % represents the coordinates of
3042                            % a projected training face in
3043                            % the eigenspace.
3044 for r=1:Total_No_of_Known_Im
3045     All_Known_Transformed_Im(:,r)=Best_Eigenvectors.'*.......
3046         Known_Im_Subt_Mean(:,r);
3047 end
3048 All_Known_Transformed_Im;
3049
3050
3051 % Projecting each tested image on the eigenspace. It is just
3052 % expressing each tested image in terms of the eigenfaces. This is
3053 % called principal components transform (also called the Hotelling
3054 % or Karhunen–Loéve transform)
3055 All_Tested_Transformed_Im=.......
3056     zeros(size(Best_Eigenvectors,2),.....
3057     Total_No_of_Tested_Im); % A 2D matrix where each column
3058                             % represents the coordinates of
3059                             % a projected tested image in
3060                             % the eigenspace.
3061 for r1=1:Total_No_of_Tested_Im
3062     All_Tested_Transformed_Im(:,r1)=Best_Eigenvectors.'*......
3063         Tested_Im_Subt_Mean(:,r1);
3064 end
3065 All_Tested_Transformed_Im;
3066
3067
3068 fid=fopen('Face Detection Results for Testing.txt',.......
3069     'w'); % A text file for typing the
3070           % results of face detection.
3071 fprintf(fid,['\n\t ***** The Results of Face Detection '.......
3072     'Obtained from the PCA and IPCA *****\r\n']);
3073 fprintf(fid,['\t\t\t\t***** Code for Testing and Setting '.......
3074     'up Thresholds *****\r\n\n']);
3075 fprintf(fid,['Image No.\t      The Image Originally Is\t  '......
3076     'The Detected Image Is\t    The Status\r\n']);
```

```
3077 fprintf(fid,['=============\t  ======================\t  '.....
3078     '====================\t    ==========\r\n']);
3079
3080 % Note that, the tested images must be unknown whether they are
3081 % face images or not but they are known here to be face images just
3082 % for testing the face detection process and for setting up a
3083 % detection threshold.
3084 Distances_Vector1=zeros(1,......
3085     Total_No_of_Tested_Im); % A vector contains the distances
3086                             % between each pre-processed tested
3087                             % image and its reconstruction.
3088 for i=1:Total_No_of_Tested_Im
3089     g=Best_Eigenvectors*All_Tested_Transformed_Im(:,i);
3090     Reconstructed_Tested_Im=max(Tested_Im_Subt_Mean(:,i))*.......
3091         (double(g)/max(max(g))); % This is for making each
3092                                  % pre-processed tested image
3093                                  % and its reconstruction have
3094                                  % approximately the same
3095                                  % dynamic range.
3096
3097     Distances_Vector1(1,i)=norm(Tested_Im_Subt_Mean(:,i)-.....
3098         Reconstructed_Tested_Im); % Note that, a calculated
3099                                   % distance must be between
3100                                   % a pre-processed tested
3101                                   % image and its reconstruction
3102                                   % but it is not between an
3103                                   % original tested image and
3104                                   % its reconstruction.
3105 end
3106
3107 % The calculation of the mean and standard deviation for the
3108 % calculated distances between each pre-processed tested image and
3109 % its reconstruction. This is done for setting up a threshold for
3110 % face detection.
3111 Mean=mean(Distances_Vector1);
3112 Std=std(Distances_Vector1);
3113 % save(['Computed Mean for Face Detection When All '....
3114 %     'Eigenvectors Are Used as Eigenfaces'],'Mean')
3115 % save(['Computed STD for Face Detection When All '.....
3116 %     'Eigenvectors Are Used as Eigenfaces'],'Std')
3117 % save(['Computed Mean for Face Detection When the Eigenfaces '....
3118 %     'Computed by Using PCA Algorithm Are Used'],'Mean')
3119 % save(['Computed STD for Face Detection When the Eigenfaces '.....
3120 %     'Computed by Using PCA Algorithm Are Used'],'Std')
3121 save(['Computed Mean for Face Detection When the Eigenfaces '....
3122     'Computed by Using IPCA Algorithm Are Used'],'Mean')
3123 save(['Computed STD for Face Detection When the Eigenfaces '.....
3124     'Computed by Using IPCA Algorithm Are Used'],'Std')
3125
3126 Failures_Vector1=zeros(1,.....
```

```
3127      Total_No_of_Tested_Im); % A vector for counting the
3128                              % number of failures in the
3129                              % face detection process.
3130
3131  Latex_Matrix=cell(Total_No_of_Tested_Im,4); % This matrix is used
3132                                              % for creating a table
3133                                              % in Latex.
3134  for w1=1:Total_No_of_Tested_Im
3135      % The face detection process.
3136      if  Distances_Vector1(1,w1)>=(Mean-Std) && ......
3137              Distances_Vector1(1,w1)<=(Mean+Std)
3138          Detected_As='a face';
3139          Detected_Image=imread('A_Face.jpg');
3140      else
3141          Detected_As='not a face';
3142          Detected_Image=imread('Not_a_Face.jpg');
3143      end
3144
3145      b='a face'; % Defining an original tested image.
3146
3147      f1='Success';
3148      e=strcmp(Detected_As,'a face');
3149      if w1<=L9 && e==0
3150          f1='Failure';
3151          Failures_Vector1(1,w1)=1;
3152      end
3153
3154  %     Tested_Im_Number1=[num2str(w1) '.jpg'];
3155  %     Tested_Im_Location1=fullfile(Tested_Images_Folder,.....
3156  %         Tested_Im_Number1);
3157  %     Tested_Im1=im2double(rgb2gray(imread(Tested_Im_Location1)));
3158  %     subplot(2,1,1)
3159  %     imshow(Tested_Im1)
3160  %     title(['Image No.' num2str(w1) ' Originally Is'])
3161  %     subplot(2,1,2)
3162  %     imshow(Detected_Image)
3163  %     title('It Is Detected As')
3164
3165      if e==1
3166          fprintf(fid,['%0.3d\t\t\t\t  %s\t\t\t\t\t\t  %s\t\t\t\t\t'....
3167              '    %s \r\n\n'],w1,b,Detected_As,f1);
3168          Latex_Matrix(w1,1:4)={num2str(w1) b Detected_As f1};
3169      else fprintf(fid,['%0.3d\t\t\t\t  %s\t\t\t\t\t\t  %s\t\t\t\t'.....
3170              '    %s \r\n\n'],w1,b,Detected_As,f1);
3171          Latex_Matrix(w1,1:4)={num2str(w1) b Detected_As f1};
3172      end
3173
3174  %     disp(['Please, press any keyboard button to see another '....
3175  %         'image and its detection....'])
3176  %     pause
```

```
%     clc
end
Total_Number_of_Failures1=......
    sum(Failures_Vector1); % The total number of failures
                           % in the face detection process.

fprintf(fid,['=============================================='....
    '==================================\r\n']);
fprintf(fid,['** The Total Number of Successes: %0.3d out of '....
    '%0.3d (%3.4f%%) \r\n\n'],Total_No_of_Tested_Im-.....
    Total_Number_of_Failures1,Total_No_of_Tested_Im,.....
    ((Total_No_of_Tested_Im-Total_Number_of_Failures1)/.....
    Total_No_of_Tested_Im)*100);
fprintf(fid,['** The Total Number of Failures: %0.3d out '......
    'of %0.3d (%3.4f%%) \r\n\n'],Total_Number_of_Failures1,......
    Total_No_of_Tested_Im,........
    (Total_Number_of_Failures1/Total_No_of_Tested_Im)*100);
fclose(fid);

close all
clc

disp(['Please, see the documented results of face detection '.....
    'in the open'])
disp(['text file then press any keyboard button to resume '.....
    'the code >>>>'])
Text='Face Detection Results for Testing.txt';
open(Text) % Opening the text file which contains
           % the results of face detection.
pause
clc
open('PCA_IPCA_Testing_and_Setting_up_Thresholds.m')


%%
%%%%%%%%%% The Computation of Face Detection Performance %%%%%%%%%%


clc
close all


Error_Rate_Detection=.......
    zeros(1,length(Eigenvalues)); % Measuring detection
                                  % performance. The elements
                                  % of this vector represent
                                  % the error rates in detection
                                  % when different eigenfaces are
                                  % selected.
Eigenvalues1=fliplr(Eigenvalues);
```

```
Eigenvectors1=flipdim(Eigenvectors,2); % This flipping just to make
                                       % the highest correlated
                                       % eigenface is associated
                                       % with the first eigenvalue
                                       % and so on. This is done to
                                       % be compatable with the
                                       % generated document
                                       % "PCA vs. IPCA".
for kk=1:length(Eigenvalues)
    ['Iteration No.: ' num2str(kk) ' Out of ' .....
        num2str(length(Eigenvalues))]
    Selected_Eigenvectors=Eigenvectors1(:,1:kk);


    % Projecting each training face on the eigenspace. It is just
    % expressing each training face in terms of the eigenfaces.
    % This is called principal components transform (also called
    % the Hotelling or Karhunen–Loéve transform)
    All_Known_Transformed_Im=.......
        zeros(size(Selected_Eigenvectors,2),........
        Total_No_of_Known_Im); % A 2D matrix where each column
                               % represents the coordinates of a
                               % projected training face in the
                               % eigenspace.
    for r=1:Total_No_of_Known_Im
        All_Known_Transformed_Im(:,r)=Selected_Eigenvectors.'*....
            Known_Im_Subt_Mean(:,r);
    end
    All_Known_Transformed_Im;


    % Projecting each tested image on the eigenspace. It is just
    % expressing each tested image in terms of the eigenfaces. This
    % is called principal components transform (also called the
    % Hotelling or Karhunen–Loéve transform)
    All_Tested_Transformed_Im=...........
        zeros(size(Selected_Eigenvectors,2),.....
        Total_No_of_Tested_Im); % A 2D matrix where each column
                                % represents the coordinates of a
                                % projected tested image in the
                                % eigenspace.
    for r1=1:Total_No_of_Tested_Im
        All_Tested_Transformed_Im(:,r1)=.....
            Selected_Eigenvectors.'*Tested_Im_Subt_Mean(:,r1);
    end
    All_Tested_Transformed_Im;


    fid=fopen(['.\Face Detection Performance\Detection '.....
        'Results for Different Eigenfaces\Results for Testing '....
```

```
3277          'When q=' num2str(kk) '.txt'],'w'); % A text file for
3278                                              % typing the results of
3279                                              % face Detection.
3280      fprintf(fid,['\n  ***** The Results of Face Detection '.....
3281          'When q=' num2str(kk) ......
3282          ' Obtained from the PCA and *****\r\n']);
3283      fprintf(fid,['\t\t   ***** IPCA Code for Testing and '......
3284          'Setting up Thresholds *****\r\n\n']);
3285      fprintf(fid,['Image No.\t      The Image Originally '.....
3286          'Is\t  The Detected Image Is\t    The Status\r\n']);
3287      fprintf(fid,['=============\t  ======================\t  '....
3288          '====================\t    ==========\r\n']);
3289
3290      % Note that, the tested images must be unknown whether they are
3291      % face images or not but they are known here to be face images
3292      % just for testing the face detection process and for setting
3293      % up a detection threshold.
3294      Distances_Vector1=zeros(1,.......
3295          Total_No_of_Tested_Im); % A vector contains the distances
3296                                  % between each pre-processed tested
3297                                  % image and its reconstruction.
3298      for i=1:Total_No_of_Tested_Im
3299          g=Selected_Eigenvectors*All_Tested_Transformed_Im(:,i);
3300          Reconstructed_Tested_Im=max(Tested_Im_Subt_Mean(:,i))*....
3301              (double(g)/max(max(g))); % This is for making each
3302                                       % pre-processed tested image
3303                                       % and its reconstruction have
3304                                       % approximately the same
3305                                       % dynamic range.
3306
3307          Distances_Vector1(1,i)=norm(Tested_Im_Subt_Mean(:,i)-.....
3308              Reconstructed_Tested_Im); % Note that, a calculated
3309                                        % distance must be between a
3310                                        % pre-processed tested image
3311                                        % and its reconstruction but
3312                                        % it is not between an
3313                                        % original tested image and
3314                                        % its reconstruction.
3315      end
3316
3317      % The calculation of the mean and standard deviation for the
3318      % calculated distances between each pre-processed tested image
3319      % and its reconstruction. This is done for setting up a
3320      % threshold for face detection.
3321      Mean=mean(Distances_Vector1);
3322      Std=std(Distances_Vector1);
3323      save([cd '\Face Detection Performance\Means and STDs for '....
3324          'Different Eigenfaces\Mean for Face Detection When q=' ....
3325          num2str(kk)],'Mean')
3326      save([cd '\Face Detection Performance\Means and STDs for '....
```

```
3327          'Different Eigenfaces\STD for Face Detection When q=' .....
3328          num2str(kk)],'Std')
3329
3330      Failures_Vector1=zeros(1,.....
3331          Total_No_of_Tested_Im); % A vector for counting the
3332                                  % number of failures in the
3333                                  % face detection process.
3334      for w1=1:Total_No_of_Tested_Im
3335          % The face detection process.
3336          if  Distances_Vector1(1,w1)>=(Mean-Std) && .....
3337                  Distances_Vector1(1,w1)<=(Mean+Std)
3338              Detected_As='a face';
3339              Detected_Image=imread('A_Face.jpg');
3340          else
3341              Detected_As='not a face';
3342              Detected_Image=imread('Not_a_Face.jpg');
3343          end
3344
3345          b='a face'; % Defining an original tested image.
3346
3347          f1='Success';
3348          e=strcmp(Detected_As,'a face');
3349          if w1<=L9 && e==0
3350              f1='Failure';
3351              Failures_Vector1(1,w1)=1;
3352          end
3353
3354  %         clc
3355  %         Tested_Im_Number1=[num2str(w1) '.jpg'];
3356  %         Tested_Im_Location1=fullfile(Tested_Images_Folder,....
3357  %             Tested_Im_Number1);
3358  %         Tested_Im1=im2double(......
3359  %             rgb2gray(imread(Tested_Im_Location1)));
3360  %         subplot(2,1,1)
3361  %         imshow(Tested_Im1)
3362  %         title(['Image No.' num2str(w1) ' Originally Is'])
3363  %         subplot(2,1,2)
3364  %         imshow(Detected_Image)
3365  %         title({'It Is Detected As';['(When q=' num2str(kk) ')']})
3366
3367          if e==1
3368              fprintf(fid,['%0.3d\t\t\t\t  %s\t\t\t\t\t  '......
3369                  '%s\t\t\t\t    %s \r\n\n'],w1,b,Detected_As,f1);
3370          else fprintf(fid,['%0.3d\t\t\t\t  %s\t\t\t\t\t  '.....
3371                  '%s\t\t\t    %s \r\n\n'],w1,b,Detected_As,f1);
3372          end
3373
3374  %         disp(['Please, press any keyboard button to see '......
3375  %             'another image and its detection when q=' .....
3376  %             num2str(kk) ' .......'])
```

```
3377 %         pause
3378 %         clc
3379     end
3380     Total_Number_of_Failures1=.......
3381         sum(Failures_Vector1); % The total number of failures
3382                                % in the face detection process.
3383
3384     Error_Rate_Detection(1,kk)=.........
3385         (Total_Number_of_Failures1/Total_No_of_Tested_Im)*100;
3386
3387     fprintf(fid,['=========================================='.....
3388         '=====================================\r\n']);
3389     fprintf(fid,['** The Total Number of Successes: %0.3d '....
3390         'out of %0.3d (%3.4f%%) \r\n\n'],...
3391         Total_No_of_Tested_Im-Total_Number_of_Failures1,....
3392         Total_No_of_Tested_Im,.......
3393         ((Total_No_of_Tested_Im-Total_Number_of_Failures1)/.....
3394         Total_No_of_Tested_Im)*100);
3395     fprintf(fid,['** The Total Number of Failures: %0.3d '....
3396         'out of %0.3d (%3.4f%%) \r\n\n'],....
3397         Total_Number_of_Failures1,Total_No_of_Tested_Im,.....
3398         (Total_Number_of_Failures1/Total_No_of_Tested_Im)*100);
3399     fclose(fid);
3400
3401 %     close all
3402 %     clc
3403 %
3404 %     disp(['Please, see the documented results of face '.....
3405 %         'detection when q=' num2str(kk) ' in the open'])
3406 %     disp(['text file then press any keyboard button to '.....
3407 %         'resume the code >>>>'])
3408 %     Text=['.\Face Detection Performance\Testing Face '.....
3409 %         'Detection Results for Different Eigenfaces\Face '......
3410 %         'Detection Results for Testing When q=' ......
3411 %         num2str(kk) '.txt'];
3412 %     open(Text) % Opening the text file which contains
3413 %                % the results of face detection.
3414 %     pause
3415 %     clc
3416 %     open('PCA_IPCA_Testing_and_Setting_up_Thresholds.m')
3417 end
3418
3419
3420 % Plotting the error rates of face detection for different selected
3421 % eigenfaces compared with the error rates for the PCA and IPCA
3422 % algorithms.
3423 clc
3424 figure('units','centimeters','position',[0.15 1.2 35.8 16.9])
3425 subplot(2,1,1)
3426 Leg1=plot(1:length(Eigenvalues),Error_Rate_Detection);
```

```
hold on
Leg2=plot(1,Error_Rate_Detection(1),'kd','LineWidth',1.5,.....
    'MarkerEdgeColor','k','MarkerFaceColor','m','MarkerSize',8);
text(24.8,Error_Rate_Detection(1)+1.8,......
    {'\fontsize{10} \color{black}' Error_Rate_Detection(1)})
hold on
Leg3=plot(length(Eigenvalues),.......
    Error_Rate_Detection(length(Eigenvalues)),'kd',....
    'LineWidth',1.5,'MarkerEdgeColor','k',....
    'MarkerFaceColor','g','MarkerSize',8);
text(length(Eigenvalues)-55,......
    Error_Rate_Detection(length(Eigenvalues))-1.8,.....
    {'\fontsize{10} \color{black}' .....
    Error_Rate_Detection(length(Eigenvalues))})
hold off
axis([0 length(Eigenvalues)+(length(Eigenvalues)/40) ......
    -max(Error_Rate_Detection)/10 ..............
    max(Error_Rate_Detection)+(max(Error_Rate_Detection)/10)])
set(gca,'XTick',[1 length(Eigenvalues)/2 length(Eigenvalues)])
title(['The Error Rates of Detection for Different '.....
    'Selected Eigenfaces'])
xlabel('The Number of Selected Eigenfaces')
ylabel('The Error Rate (%)')
legend([Leg1 Leg2 Leg3],['The Error Rates of Detection for '.....
    'Different Selected Eigenfaces.'],['The Error Rate of '....
    'Detection When the First Eigenface Is Selected.'],....
    ['The Error Rate of Detection When All Eigenfaces Are '.....
    'Selected.'],'Location','NorthWest');
subplot(2,1,2)
Leg1=plot(1:length(Eigenvalues),Error_Rate_Detection);
axis([0 20 0 .......
    max(Error_Rate_Detection)+(max(Error_Rate_Detection)/10)])
Leg2=line([length(Best_Eigenvalues_PCA) ......
    length(Best_Eigenvalues_PCA)],.....
    [0 Error_Rate_Detection(length(Best_Eigenvalues_PCA))],....
    'LineStyle','--','Color','k','LineWidth',3);
line([0 length(Best_Eigenvalues_PCA)],........
    [Error_Rate_Detection(length(Best_Eigenvalues_PCA)) ......
    Error_Rate_Detection(length(Best_Eigenvalues_PCA))],....
    'LineStyle','--','Color','k','LineWidth',3)
text(length(Best_Eigenvalues_PCA)-0.9,.......
    Error_Rate_Detection(length(Best_Eigenvalues_PCA))+3.6,.....
    {'\fontsize{10} \color{black}' .........
    num2str(Error_Rate_Detection(length(Best_Eigenvalues_PCA)))})
Leg3=line([length(Best_Eigenvalues_IPCA) ......
    length(Best_Eigenvalues_IPCA)],......
    [0 Error_Rate_Detection(length(Best_Eigenvalues_IPCA))],.....
    'LineStyle','--','Color','r','LineWidth',4);
line([0 length(Best_Eigenvalues_IPCA)],.......
    [Error_Rate_Detection(length(Best_Eigenvalues_IPCA)) .....
```

```
3477      Error_Rate_Detection(length(Best_Eigenvalues_IPCA))],....
3478      'LineStyle','--','Color','r','LineWidth',4)
3479  text(0.3,......
3480      Error_Rate_Detection(length(Best_Eigenvalues_IPCA))+3.6,.....
3481      {'\fontsize{10} \color{red} \bf' .....
3482      num2str(Error_Rate_Detection(length(Best_Eigenvalues_IPCA)))})
3483  vv=[length(Best_Eigenvalues_IPCA) length(Best_Eigenvalues_PCA)];
3484  if vv(1)==1 || vv(2)==1
3485      set(gca,'XTick',[0 sort(vv) 20])
3486  else set(gca,'XTick',[0 1 sort(vv) 20])
3487  end
3488  title(['Explaining the Error Rates for the PCA and IPCA '.....
3489      'Algorithms on the Plot'])
3490  xlabel('The Number of Selected Eigenfaces')
3491  ylabel('The Error Rate (%)')
3492  legend([Leg1 Leg2 Leg3],['The Error Rates of Detection for '.....
3493      'Different Selected Eigenfaces.'],......
3494      'The Error Rate for PCA Algorithm.',.....
3495      'The Error Rate for IPCA Algorithm.','Location','NorthEast');
```

Appendix B

Databases for the Digital and Optical Models

All images in Figure B.1, Figure B.2, Figure B.3, Figure B.4, Figure B.5, Figure B.6, Figure B.7, Figure B.8, and Figure B.9 form the database of the tested images. The database of the objects consists of images that have vertical faces to people's shoulders shown in Figure B.1, Figure B.4, and Figure B.7.

Figure B.1: Images for Mr. Mansour Alshammari, 1^{st} projection.

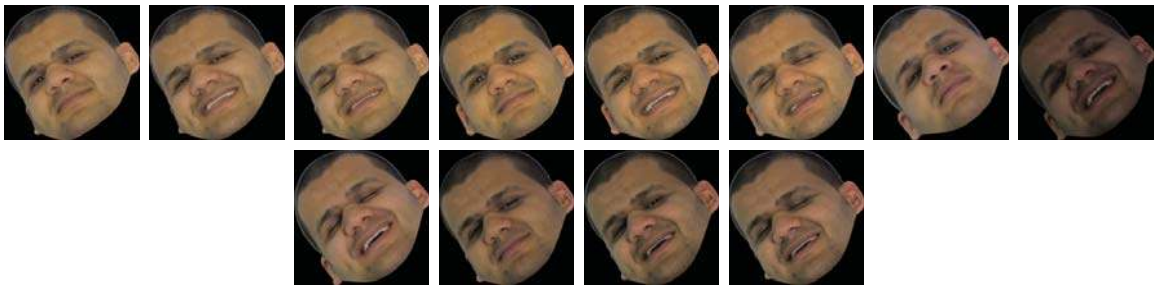

Figure B.2: Images for Mr. Mansour Alshammari, 2^{nd} projection.

Figure B.3: Images for Mr. Mansour Alshammari, 3^{rd} projection.

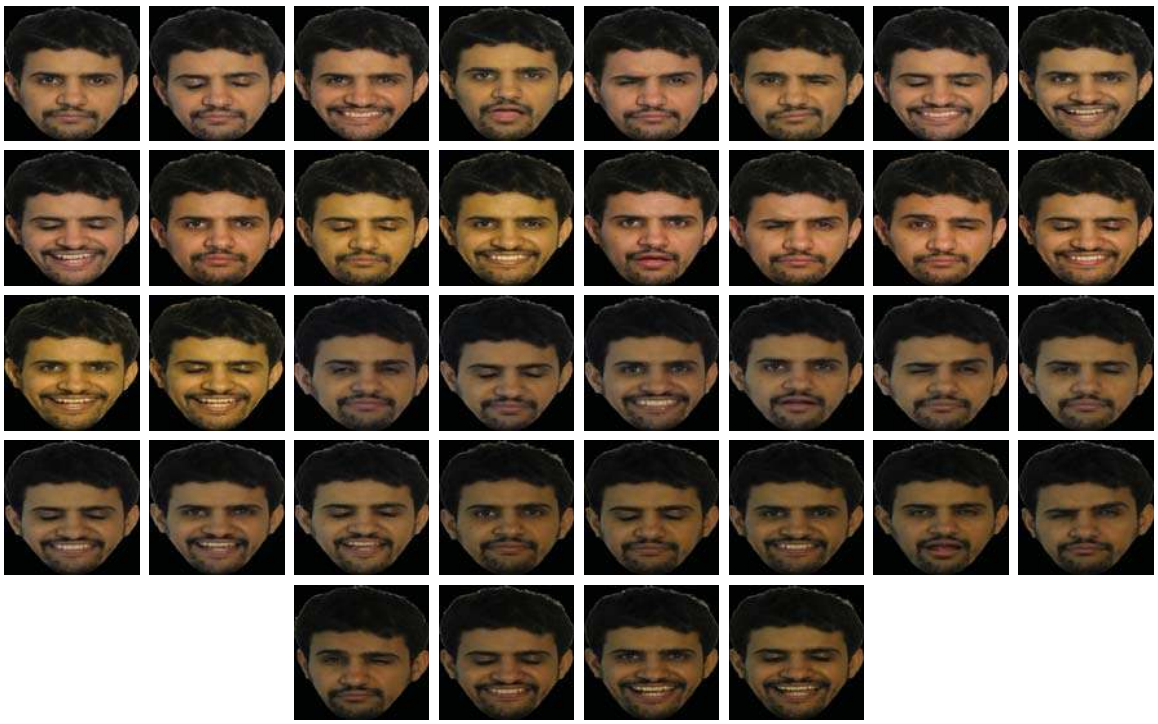

Figure B.4: Images for Mr. Methkir Alharthee, 1^{st} projection.

Figure B.5: Images for Mr. Methkir Alharthee, 2^{nd} projection.

Figure B.6: Images for Mr. Methkir Alharthee, 3^{rd} projection.

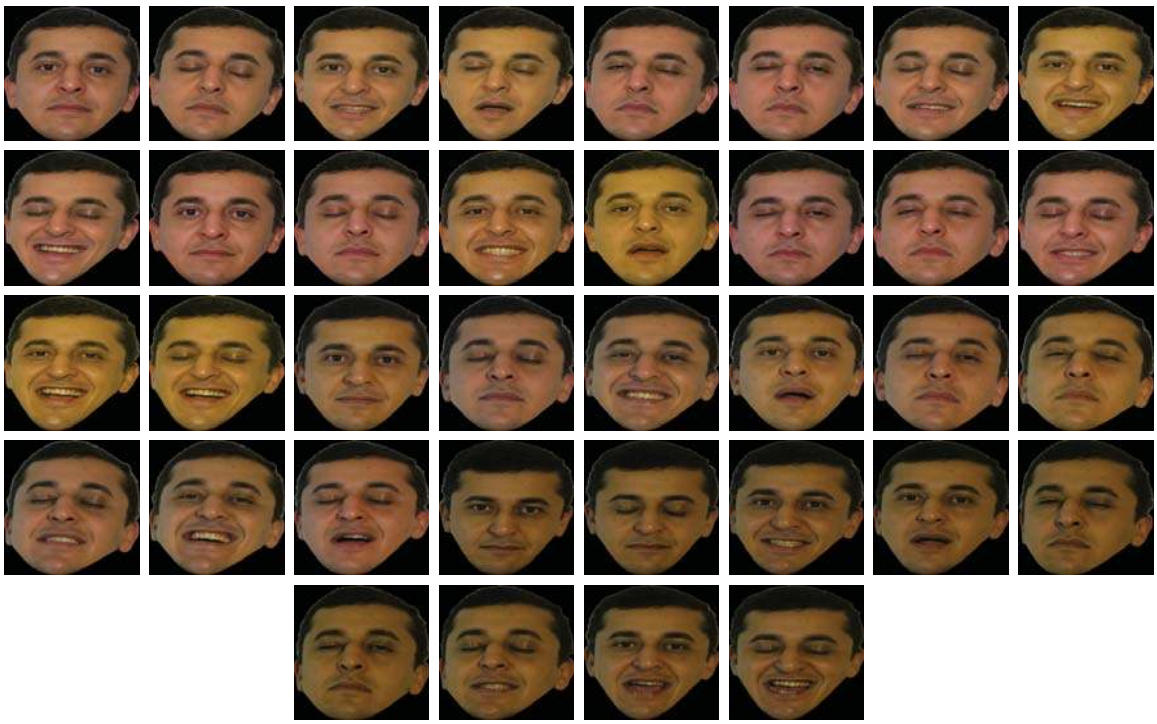

Figure B.7: Images for Mr. Mohammed Hanafy, 1^{st} projection.

Figure B.8: Images for Mr. Mohammed Hanafy, 2^{nd} projection.

Figure B.9: Images for Mr. Mohammed Hanafy, 3^{rd} projection.

Appendix C

Results of the Digital Recognition

Table C.1: The recognition of all 180 tested images by using the PCA algorithm.

Tested Image No.	Input Face	Recognized Output Face	Status
1	Mr. Mansour Alshammari	Unknown Image	Failure
2	Mr. Mansour Alshammari	Unknown Image	Failure
3	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
4	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
5	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
6	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
7	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
8	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
9	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
10	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
11	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
12	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
13	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
14	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
15	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
16	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
17	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
18	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
19	Mr. Mansour Alshammari	Unknown Image	Failure
20	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
21	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success

Continued on the next page ...

Tested Image No.	Input Face	Recognized Output Face	Status
22	Mr. Mansour Alshammari	Unknown Image	Failure
23	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
24	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
25	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
26	Mr. Mansour Alshammari	Unknown Image	Failure
27	Mr. Mansour Alshammari	Unknown Image	Failure
28	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
29	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
30	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
31	Mr. Mansour Alshammari	Unknown Image	Failure
32	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
33	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
34	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
35	Mr. Mansour Alshammari	Unknown Image	Failure
36	Mr. Mansour Alshammari	Unknown Image	Failure
37	Mr. Mansour Alshammari	Unknown Image	Failure
38	Mr. Mansour Alshammari	Unknown Image	Failure
39	Mr. Mansour Alshammari	Unknown Image	Failure
40	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
41	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
42	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
43	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
44	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
45	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
46	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
47	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
48	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
49	Mr. Mansour Alshammari	Unknown Image	Failure
50	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
51	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
52	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
53	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
54	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
55	Mr. Mansour Alshammari	Unknown Image	Failure
56	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
57	Mr. Mansour Alshammari	Unknown Image	Failure
58	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
59	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success

Continued on the next page ...

Tested Image No.	Input Face	Recognized Output Face	Status
60	Mr. Mansour Alshammari	Mr. Mansour Alshammari	Success
61	Mr. Methkir Alharthee	Unknown Image	Failure
62	Mr. Methkir Alharthee	Unknown Image	Failure
63	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
64	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
65	Mr. Methkir Alharthee	Unknown Image	Failure
66	Mr. Methkir Alharthee	Unknown Image	Failure
67	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
68	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
69	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
70	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
71	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
72	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
73	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
74	Mr. Methkir Alharthee	Unknown Image	Failure
75	Mr. Methkir Alharthee	Unknown Image	Failure
76	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
77	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
78	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
79	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
80	Mr. Methkir Alharthee	Unknown Image	Failure
81	Mr. Methkir Alharthee	Unknown Image	Failure
82	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
83	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
84	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
85	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
86	Mr. Methkir Alharthee	Unknown Image	Failure
87	Mr. Methkir Alharthee	Unknown Image	Failure
88	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
89	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
90	Mr. Methkir Alharthee	Unknown Image	Failure
91	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
92	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
93	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
94	Mr. Methkir Alharthee	Unknown Image	Failure
95	Mr. Methkir Alharthee	Unknown Image	Failure
96	Mr. Methkir Alharthee	Unknown Image	Failure
97	Mr. Methkir Alharthee	Unknown Image	Failure

Continued on the next page ...

Tested Image No.	Input Face	Recognized Output Face	Status
98	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
99	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
100	Mr. Methkir Alharthee	Unknown Image	Failure
101	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
102	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
103	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
104	Mr. Methkir Alharthee	Unknown Image	Failure
105	Mr. Methkir Alharthee	Unknown Image	Failure
106	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
107	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
108	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
109	Mr. Methkir Alharthee	Unknown Image	Failure
110	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
111	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
112	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
113	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
114	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
115	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
116	Mr. Methkir Alharthee	Unknown Image	Failure
117	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
118	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
119	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
120	Mr. Methkir Alharthee	Mr. Methkir Alharthee	Success
121	Mr. Mohammed Hanafy	Unknown Image	Failure
122	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
123	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
124	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
125	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
126	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
127	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
128	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
129	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
130	Mr. Mohammed Hanafy	Unknown Image	Failure
131	Mr. Mohammed Hanafy	Unknown Image	Failure
132	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
133	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
134	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
135	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success

Continued on the next page ...

Tested Image No.	Input Face	Recognized Output Face	Status
136	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
137	Mr. Mohammed Hanafy	Unknown Image	Failure
138	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
139	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
140	Mr. Mohammed Hanafy	Unknown Image	Failure
141	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
142	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
143	Mr. Mohammed Hanafy	Unknown Image	Failure
144	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
145	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
146	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
147	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
148	Mr. Mohammed Hanafy	Unknown Image	Failure
149	Mr. Mohammed Hanafy	Unknown Image	Failure
150	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
151	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
152	Mr. Mohammed Hanafy	Unknown Image	Failure
153	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
154	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
155	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
156	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
157	Mr. Mohammed Hanafy	Unknown Image	Failure
158	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
159	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
160	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
161	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
162	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
163	Mr. Mohammed Hanafy	Mr. Mansour Alshammari	Failure
164	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
165	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
166	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
167	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
168	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
169	Mr. Mohammed Hanafy	Unknown Image	Failure
170	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
171	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
172	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
173	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success

Continued on the next page ...

Tested Image No.	Input Face	Recognized Output Face	Status
174	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
175	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
176	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
177	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
178	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
179	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
180	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success

The table end.

Appendix D

Results of the Digital Detection

Table D.1: The detection of all 180 tested images by using the PCA algorithm.

Tested Image No.	Input Image	Detected Output Image	Status
1	a face	not a face	Failure
2	a face	not a face	Failure
3	a face	a face	Success
4	a face	a face	Success
5	a face	a face	Success
6	a face	a face	Success
7	a face	a face	Success
8	a face	a face	Success
9	a face	a face	Success
10	a face	a face	Success
11	a face	a face	Success
12	a face	a face	Success
13	a face	a face	Success
14	a face	a face	Success
15	a face	a face	Success
16	a face	a face	Success
17	a face	a face	Success
18	a face	a face	Success
19	a face	a face	Success
20	a face	a face	Success
21	a face	a face	Success

Continued on the next page ...

Tested Image No.	Input Image	Detected Output Image	Status
22	a face	a face	Success
23	a face	a face	Success
24	a face	a face	Success
25	a face	a face	Success
26	a face	a face	Success
27	a face	a face	Success
28	a face	a face	Success
29	a face	a face	Success
30	a face	a face	Success
31	a face	a face	Success
32	a face	a face	Success
33	a face	a face	Success
34	a face	not a face	Failure
35	a face	a face	Success
36	a face	a face	Success
37	a face	not a face	Failure
38	a face	not a face	Failure
39	a face	not a face	Failure
40	a face	a face	Success
41	a face	a face	Success
42	a face	a face	Success
43	a face	a face	Success
44	a face	a face	Success
45	a face	a face	Success
46	a face	a face	Success
47	a face	a face	Success
48	a face	a face	Success
49	a face	not a face	Failure
50	a face	not a face	Failure
51	a face	not a face	Failure
52	a face	a face	Success
53	a face	a face	Success
54	a face	a face	Success
55	a face	a face	Success
56	a face	a face	Success
57	a face	a face	Success
58	a face	a face	Success
59	a face	a face	Success

Continued on the next page ...

Tested Image No.	Input Image	Detected Output Image	Status
60	a face	a face	Success
61	a face	not a face	Failure
62	a face	a face	Success
63	a face	a face	Success
64	a face	a face	Success
65	a face	not a face	Failure
66	a face	a face	Success
67	a face	a face	Success
68	a face	a face	Success
69	a face	a face	Success
70	a face	a face	Success
71	a face	a face	Success
72	a face	a face	Success
73	a face	a face	Success
74	a face	a face	Success
75	a face	a face	Success
76	a face	a face	Success
77	a face	a face	Success
78	a face	a face	Success
79	a face	a face	Success
80	a face	a face	Success
81	a face	a face	Success
82	a face	a face	Success
83	a face	a face	Success
84	a face	a face	Success
85	a face	a face	Success
86	a face	not a face	Failure
87	a face	a face	Success
88	a face	a face	Success
89	a face	a face	Success
90	a face	a face	Success
91	a face	not a face	Failure
92	a face	a face	Success
93	a face	not a face	Failure
94	a face	a face	Success
95	a face	not a face	Failure
96	a face	not a face	Failure
97	a face	not a face	Failure

Continued on the next page ...

Tested Image No.	Input Image	Detected Output Image	Status
98	a face	a face	Success
99	a face	a face	Success
100	a face	a face	Success
101	a face	a face	Success
102	a face	a face	Success
103	a face	not a face	Failure
104	a face	a face	Success
105	a face	a face	Success
106	a face	a face	Success
107	a face	a face	Success
108	a face	not a face	Failure
109	a face	not a face	Failure
110	a face	a face	Success
111	a face	a face	Success
112	a face	a face	Success
113	a face	a face	Success
114	a face	a face	Success
115	a face	not a face	Failure
116	a face	not a face	Failure
117	a face	a face	Success
118	a face	not a face	Failure
119	a face	a face	Success
120	a face	not a face	Failure
121	a face	not a face	Failure
122	a face	a face	Success
123	a face	a face	Success
124	a face	a face	Success
125	a face	a face	Success
126	a face	a face	Success
127	a face	a face	Success
128	a face	a face	Success
129	a face	a face	Success
130	a face	a face	Success
131	a face	a face	Success
132	a face	a face	Success
133	a face	a face	Success
134	a face	a face	Success
135	a face	a face	Success

Continued on the next page ...

Tested Image No.	Input Image	Detected Output Image	Status
136	a face	a face	Success
137	a face	a face	Success
138	a face	a face	Success
139	a face	a face	Success
140	a face	a face	Success
141	a face	a face	Success
142	a face	a face	Success
143	a face	a face	Success
144	a face	a face	Success
145	a face	a face	Success
146	a face	a face	Success
147	a face	a face	Success
148	a face	not a face	Failure
149	a face	a face	Success
150	a face	not a face	Failure
151	a face	a face	Success
152	a face	not a face	Failure
153	a face	a face	Success
154	a face	not a face	Failure
155	a face	a face	Success
156	a face	not a face	Failure
157	a face	not a face	Failure
158	a face	a face	Success
159	a face	a face	Success
160	a face	a face	Success
161	a face	a face	Success
162	a face	a face	Success
163	a face	a face	Success
164	a face	a face	Success
165	a face	a face	Success
166	a face	a face	Success
167	a face	a face	Success
168	a face	a face	Success
169	a face	not a face	Failure
170	a face	not a face	Failure
171	a face	a face	Success
172	a face	a face	Success
173	a face	a face	Success

Continued on the next page ...

Tested Image No.	Input Image	Detected Output Image	Status
174	a face	a face	Success
175	a face	a face	Success
176	a face	a face	Success
177	a face	a face	Success
178	a face	a face	Success
179	a face	a face	Success
180	a face	a face	Success

The table end.

Appendix E

Derivation of $U_2(x_2, y_2)$ for the JTC

In the joint transform correlator (JTC), the derivation of the complex amplitude distribution $U_2(x_2, y_2)$ of the Fourier transformed field in the back focal plane P_2 can be found as follows,

$$\begin{aligned} U_2(x_2, y_2) =& \frac{1}{j\lambda f} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} U_1(x_1, y_1) \exp^{-j\frac{2\pi}{\lambda f}(x_1 x_2 + y_1 y_2)} dx_1 dy_1 \\ =& \frac{A}{j\lambda f} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h\left(x_1, y_1 - \frac{Y}{2}\right) \exp^{-j\frac{2\pi}{\lambda f}(x_1 x_2 + y_1 y_2)} dx_1 dy_1 + \\ &+ \frac{A}{j\lambda f} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g\left(x_1, y_1 + \frac{Y}{2}\right) \exp^{-j\frac{2\pi}{\lambda f}(x_1 x_2 + y_1 y_2)} dx_1 dy_1 \end{aligned}$$

Changing variables: $y_1 - \frac{Y}{2} \to a, dy_1 \to da$ &
$y_1 + \frac{Y}{2} \to b, dy_1 \to db$

$$
\begin{aligned}
U_2(x_2,y_2) =& \frac{A}{j\lambda f}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x_1,a)\exp^{-j\frac{2\pi}{\lambda f}\left(x_1x_2+\left(a+\frac{Y}{2}\right)y_2\right)}\,dx_1da+ \\
&+\frac{A}{j\lambda f}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x_1,b)\exp^{-j\frac{2\pi}{\lambda f}\left(x_1x_2+\left(b-\frac{Y}{2}\right)y_2\right)}\,dx_1db \\
=& \frac{A}{j\lambda f}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x_1,a)\exp^{-j\frac{2\pi}{\lambda f}(x_1x_2+ay_2)-j\frac{2\pi}{\lambda f}\frac{Y}{2}y_2}\,dx_1da+ \\
&+\frac{A}{j\lambda f}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x_1,b)\exp^{-j\frac{2\pi}{\lambda f}(x_1x_2+by_2)+j\frac{2\pi}{\lambda f}\frac{Y}{2}y_2}\,dx_1db \\
=& \frac{A}{j\lambda f}\exp^{-j\frac{\pi Y}{\lambda f}y_2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x_1,a)\exp^{-j\frac{2\pi}{\lambda f}(x_1x_2+ay_2)}\,dx_1da+ \\
&+\frac{A}{j\lambda f}\exp^{j\frac{\pi Y}{\lambda f}y_2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x_1,b)\exp^{-j\frac{2\pi}{\lambda f}(x_1x_2+by_2)}\,dx_1db
\end{aligned}
$$

Therefore,

$$
U_2(x_2,y_2) = \frac{A}{j\lambda f}\exp^{-j\frac{\pi Y}{\lambda f}y_2} H\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right) + \frac{A}{j\lambda f}\exp^{j\frac{\pi Y}{\lambda f}y_2} G\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right)
$$

Appendix F

A Code for the Optical Model

This appendix presents the joint transform correlator (JTC) code for object detection and face recognition.

```
1
2  % Testing the Joint Transform Correlator (JTC) for Object Detection
3  % and Recognition. Note That, Objects Here Are Known for Us in
4  % Order to Use Them for Testing Object Detection and Recognition
5  % Processes as Well as Setting up Recognition and Detection
6  % Thresholds.
7
8
9  clc
10 clear all
11 close all
12
13
14 Impulses_Folder=[cd .....
15     '\The Black Background Impulses']; % The folder of the black
16                                        % background impulses.
17 % Impulses_Folder=[cd ......
18 %     '\The White Background Impulses']; % The folder of the white
19                                        % background impulses.
20 if isdir(Impulses_Folder)==0
21     Error_Message=sprintf(['Error: The following folder does '....
22         'not exist\n%s'],Impulses_Folder);
23     warndlg(Error_Message);
24     break
25 end
26
27 Impulses=dir(fullfile(Impulses_Folder,'*.jpg')); % Listing the
28                                                  % folder of the
29                                                  % black or white
```

```
                                                    % background
                                                    % impulses.

Total_No_of_Impulses=length(Impulses); % The total number
                                       % of the impulses.


% Imhist for setting up a threshold to work on just the pixels of a
% face and throwing the background pixels. Imhist calculates the
% number of pixels in an impulse that have the same intensity
% levels. So, if an impulse has a unified background then the
% biggest histogram of the intensity levels will be for the
% background pixels because the total number of pixels that have
% the same intensity levels are the background pixels of the
% impulse. Note that, the histogram of a digital image is defined
% as the discrete function, h(rk)=nk, where rk is the kth intensity
% level and nk is the number of pixels in the image whose intensity
% level is rk.
hist_Impulses=zeros(Total_No_of_Impulses,256);
for A=1:Total_No_of_Impulses

    One_Impulse=Impulses(A).name;
    Impulse_Location=fullfile(Impulses_Folder,One_Impulse);
    Impulse=double(rgb2gray(.....
        imread(Impulse_Location))); % The impulse response.
    hist_Impulses(A,:)=......
        imhist(uint8(Impulse)); % Note that, each impulse must be
                                % scaled between 0 to 255 before
                                % using imhist. For doing that,
                                % uint8 can be used for converting
                                % the impulse class form double to
                                % uint8.

%     plot(hist_Impulses(A,:))
%     if A==1
%         title(['The Histogram of The First Impulse for '.....
%             'Mr. Mansour Alshammari'])
%     elseif A==2
%         title(['The Histogram of The Second Impulse for '.....
%             'Mr. Methkir Alharthee'])
%     elseif A==3
%         title(['The Histogram of The Third Impulse for '.....
%             'Mr. Mohammed Hanafy'])
%     end
%     xlabel('Intensity Level r_{k}')
%     ylabel({'The Number of Pixels in the Impulse Whose '.....
%         'Intensity Level Is r_{k} Where h(r_{k})=n_{k}'})
%     axis tight
%
%     disp(['Please, press any keyboard button to explore '......
```

```
%         'the remaining histograms >>>>>>>'])
%     pause
%     clc
end

Mean_hist_Impulses=sum(hist_Impulses,1)/......
    Total_No_of_Impulses; % The average histogram
                          % for all impulses.
% plot(Mean_hist_Impulses)
% title('The Mean Histogram of All Impulses')
% xlabel('Intensity Level r_{k}')
% ylabel({'The Mean Number of Pixels from All Impulses Whose';.....
%     'Intensity Level Is r_{k}'})
% axis tight
% pause

% Setting up a threshold in order to work on just the pixels of the
% faces and blocking the pixels of the backgrounds.
Impulses_Threshold=8; % The picked threshold is based on the
                      % average histogram for all impulses when the
                      % impulses have black backgrounds. Note that,
                      % all intesity levels below the threshold
                      % represent the impulses backgrounds because
                      % these levels have the biggest histogram.
% Impulses_Threshold=180; % The Picked threshold is based on the
%                         % average histogram for all impulses when
%                         % the impulses have white backgrounds.
%                         % Note that, all intesity levels above
%                         % the threshold represent the impulses
%                         % backgrounds because these levels have
%                         % the biggest histogram.


Objects_Folder=[cd ......
    '\The Black Background Tested Objects']; % The folder of the
                                             % black background
                                             % tested objects.
% Objects_Folder=[cd .........
%     '\The White Background Tested Objects']; % The folder of the
                                             % white background
                                             % tested objects.
if isdir(Objects_Folder)==0
    Error_Message=sprintf(['Error: The following folder does '....
        'not exist\n%s'],Objects_Folder);
    warndlg(Error_Message);
    break
end

Objects=dir(fullfile(Objects_Folder,'*.jpg')); % Listing the folder
                                               % of the black or
```

```
130                                                 % white background
131                                                 % objects.
132
133 Total_No_of_Objects=length(Objects); % The total number
134                                     % of the objects.
135
136 Objs=[36 36 36]; % Each element in this vector represents
137                 % the total number of the objects that
138                 % are taken for each impulse.
139
140 L1=Objs(1); % L1=60 is the total number of the
141            % objects for Mr. Mansour Alshammari.
142 L2=L1+Objs(2); % L2=120 is the total number of the
143               % objects for Mr. Methkir Alharthee.
144 L3=L2+Objs(3); % L3=180 is the total number of the
145               % objects for Mr. Mohammed Hanafy.
146
147
148 % Imhist for setting up a threshold to work on just the pixels of a
149 % face and throwing the background pixels. Imhist calculates the
150 % number of pixels in an object that have the same intensity
151 % levels. So, if an object has a unified background then the
152 % biggest histogram of the intensity levels will be for the
153 % background pixels because the total number of pixels that have
154 % the same intensity levels are the background pixels of the
155 % object. Note that, the histogram of a digital image is defined as
156 % the discrete function, h(rk)=nk, where rk is the kth intensity
157 % level and nk is the number of pixels in the image whose intensity
158 % level is rk.
159 hist_Objects=zeros(Total_No_of_Objects,256);
160 for A1=1:Total_No_of_Objects
161
162     Object_Number=[num2str(A1) '.jpg'];
163     Object_Location=fullfile(Objects_Folder,Object_Number);
164     Object=double(rgb2gray(imread(Object_Location))); % The object.
165     hist_Objects(A1,:)=imhist(uint8(Object)); % Note that, each
166                                               % object must be
167                                               % scaled between 0 to
168                                               % 255 before using
169                                               % imhist. For doing
170                                               % that, uint8 can be
171                                               % used for converting
172                                               % the object class
173                                               % form double to
174                                               % uint8.
175
176 %     plot(hist_Objects(A1,:))
177 %     if A1<=L1
178 %         title(['The Histogram of Object No.' num2str(A1) ....
179 %             ' for Mr. Mansour Alshammari'])
```

```
%     elseif A1>L1 && A1<=L2
%         title(['The Histogram of Object No.' num2str(A1) .....
%             ' for Mr. Methkir Alharthee'])
%     elseif A1>L2 && A1<=L3
%         title(['The Histogram of Object No.' num2str(A1) ....
%             ' for Mr. Mohammed Hanafy'])
%     end
%     xlabel('Intensity Level r_{k}')
%     ylabel({'The Number of Pixels in the Object Whose '....
%         'Intensity Level Is r_{k} Where h(r_{k})=n_{k}'})
%     axis tight
%
%     disp(['Please, press any keyboard button to explore '......
%         'the remaining histograms >>>>>>>'])
%     pause
%     clc
end

Mean_hist_Objects=sum(hist_Objects,1)/...........
    Total_No_of_Objects; % The average histogram
                         % for all objects.
% plot(Mean_hist_Objects)
% title('The Mean Histogram of All Objects')
% xlabel('Intensity Level r_{k}')
% ylabel({'The Mean Number of Pixels from All Objects Whose';....
%     'Intensity Level Is r_{k}'})
% axis tight
% pause

% Setting up a threshold in order to work on just the pixels of the
% faces and blocking the pixels of the backgrounds.
Objects_Threshold=8; % The picked threshold is based on the average
                     % histogram for all objects when the objects
                     % have black backgrounds. Note that, all
                     % intesity levels below the threshold
                     % represent the objects backgrounds because
                     % these levels have the biggest histogram.
% Objects_Threshold=180; % The Picked threshold is based on the
%                        % average histogram for all objects when
%                        % the objects have white backgrounds. Note
%                        % that, all intesity levels above the
%                        % threshold represent the objects
%                        % backgrounds because these levels have
%                        % the biggest histogram.


Max_Desired_Cross_Fields=zeros(Total_No_of_Objects,.....
    Total_No_of_Impulses); % Each element of each row in this
                           % matrix represents the maximum value
                           % of the desired crosscorrelated field
```

```
                            % between an object and one of the
                            % impulses.


for n=1:Total_No_of_Objects

    % Normalizing all the objects for removing lightening effects
    % on them then increasing the resolution of object detection
    % and recognition. Note that, the normalization will be done
    % just for the faces pixels for keeping variations among the
    % objects and impulses just in the faces without the
    % backgrounds effects.
    Object_Number=[num2str(n) '.jpg'];
    Object_Location=fullfile(Objects_Folder,Object_Number);
    Object1=........
        double(rgb2gray(imread(Object_Location))); % The object.

    T=Object1>Objects_Threshold; % The pixels bigger than the
                                 % threshold are of interest
                                 % because they represent the
                                 % pixels of a face.
%     T=Object1<Objects_Threshold; % The pixels smaller than the
%                                  % threshold are of interest
%                                  % because they represent the
%                                  % pixels of a face.
    Object=zeros(size(T,1),size(T,2)); % The normalized object.
    for R2=1:size(T,1)
        for C2=1:size(T,2)
            if T(R2,C2)==1
                Object(R2,C2)=ceil(255*(Object1(R2,C2)/......
                    max(max(Object1)))); % The normalization of the
                                         % object. This is done to
                                         % increase the dynamic
                                         % range of the object for
                                         % visualization by scaling
                                         % the intensities from 0
                                         % to 255.
            end
        end
    end
    [r c]=size(Object);


%     figure
%     subplot(2,1,1)
%     imshow(Object1)
%     if n<=L1
%         title({['This Is To Show How Good the Objects '.....
%             'Threshold Is,'];blanks(1);['Object No.' ....
%             num2str(n) ' for Mr. Mansour Alshammari']})
```

```
%     elseif n>L1 && n<=L2
%         title({['This Is To Show How Good the Objects '....
%             'Threshold Is,'];blanks(1);['Object No.' .....
%             num2str(n) ' for Mr. Methkir Alharthee']})
%     elseif n>L2 && n<=L3
%         title({['This Is To Show How Good the Objects '.....
%             'Threshold Is,'];blanks(1);['Object No.' ....
%             num2str(n) ' for Mr. Mohammed Hanafy']})
%     end
%     subplot(2,1,2)
%     imshow(Object)
%     if n<=L1
%         title(['Normalized Object No.' num2str(n) ....
%             ' for Mr. Mansour Alshammari'])
%     elseif n>L1 && n<=L2
%         title(['Normalized Object No.' num2str(n) .....
%             ' for Mr. Methkir Alharthee'])
%     elseif n>L2 && n<=L3
%         title(['Normalized Object No.' num2str(n) ....
%             ' for Mr. Mohammed Hanafy'])
%     end
%
%     figure
%     subplot(2,1,1)
%     imshow(uint8(Object1))
%     if n<=L1
%         title(['Object No.' num2str(n) .....
%             ' for Mr. Mansour Alshammari'])
%     elseif n>L1 && n<=L2
%         title(['Object No.' num2str(n) .....
%             ' for Mr. Methkir Alharthee'])
%     elseif n>L2 && n<=L3
%         title(['Object No.' num2str(n) ....
%             ' for Mr. Mohammed Hanafy'])
%     end
%     subplot(2,1,2)
%     imshow(uint8(Object))
%     if n<=L1
%         title(['Normalized Object No.' num2str(n) .....
%             ' for Mr. Mansour Alshammari'])
%     elseif n>L1 && n<=L2
%         title(['Normalized Object No.' num2str(n) ....
%             ' for Mr. Methkir Alharthee'])
%     elseif n>L2 && n<=L3
%         title(['Normalized Object No.' num2str(n) .....
%             ' for Mr. Mohammed Hanafy'])
%     end

    for m=1:Total_No_of_Impulses

```

```
330         % Normalizing all the impulses for removing lightening
331         % effects on them then increasing the resolution of object
332         % detection and recognition. Note that, the normalization
333         % will be done just for the faces pixels for keeping
334         % variations among the objects and impulses just in the
335         % faces without the backgrounds effects.
336         One_Impulse=Impulses(m).name;
337         Impulse_Location=fullfile(Impulses_Folder,One_Impulse);
338         Impulse1=double(rgb2gray(......
339             imread(Impulse_Location))); % The impulse response.
340
341         T1=Impulse1>Impulses_Threshold; % The pixels bigger than
342                                        % the threshold are of
343                                        % interest because they
344                                        % represent the pixels of
345                                        % a face.
346 %         T1=Impulse1<Impulses_Threshold; % The pixels smaller than
347 %                                        % the threshold are of
348 %                                        % interest because they
349 %                                        % represent the pixels
350 %                                        % of a face.
351         Impulse=zeros(size(T1,1),size(T1,2)); % The normalized
352                                               % impulse response.
353         for R1=1:size(T1,1)
354             for C1=1:size(T1,2)
355                 if T1(R1,C1)==1
356                     Impulse(R1,C1)=ceil(255*(Impulse1(R1,C1)/.....
357                         max(max(Impulse1)))); % The normalization
358                                               % of the impulse
359                                               % response. This is
360                                               % done to increase
361                                               % the dynamic range
362                                               % of the impulse for
363                                               % visualization by
364                                               % scaling the
365                                               % intensities from
366                                               % 0 to 255.
367                 end
368             end
369         end
370         [p q]=size(Impulse);
371
372
373         % Equalizing the width of the impulse response with the
374         % width of the object in order to collimate them on the
375         % input transparencies.
376         if q>c
377             if mod(q-c,2)==0
378                 Object=[zeros(r,(q-c)/2) Object zeros(r,(q-c)/2)];
379             elseif mod(q-c,2)==1
```

```
380                 Object=[zeros(r,floor((q-c)/2)) Object ....
381                     zeros(r,floor((q-c)/2)+1)];
382             end
383         elseif q<c
384             if mod(c-q,2)==0
385                 Impulse=[zeros(p,(c-q)/2) Impulse .....
386                     zeros(p,(c-q)/2)];
387             elseif mod(c-q,2)==1
388                 Impulse=[zeros(p,floor((c-q)/2)) Impulse .....
389                     zeros(p,floor((c-q)/2)+1)];
390             end
391         end
392
393         % Collimating the impulse response and the object on the
394         % input transparencies. Note that, the separation between
395         % the impulse response and object must be bigger than
396         % max{Wh,Wg}+Wg/2+Wh/2 in order to make the output
397         % crosscorrelations of the impulse response and object are
398         % completely separated without any overlapping.
399         Wh=size(Impulse,1); % The width of the impulse response in
400                             % the direction of the y1-coordinate.
401         Wg=size(Object,1); % The width of the object in the
402                            % direction of the y1-coordinate.
403         Y=max(Wh,Wg)+((Wh+Wg)/2); % The separation between the
404                                   % centers of the impulse response
405                                   % and object.
406         dis=10; % This distance in order to make the separation
407                 % between the centers of the impulse response and
408                 % object bigger than max{Wh,Wg}+Wg/2+Wh/2.
409         Impulse_Object=[.......
410             zeros(ceil(max(Wh,Wg)/2+Wh/4+3/4*Wg)+ceil(dis/2),.....
411             size(Impulse,2));Impulse;.....
412             zeros(max(Wh,Wg)+dis,size(Impulse,2));Object;.....
413             zeros(ceil(max(Wh,Wg)/2+3/4*Wh+Wg/4)+ceil(dis/2),.....
414             size(Object,2))];
415
416         % Equalizing the dimensions of the input plane P1 by
417         % padding it with zeros.
418         [R C]=size(Impulse_Object);
419         if R>C
420             if mod(R-C,2)==0
421                 Impulse_Object=[.......
422                     zeros(size(Impulse_Object,1),(R-C)/2) .....
423                     Impulse_Object ......
424                     zeros(size(Impulse_Object,1),(R-C)/2)];
425             elseif mod(R-C,2)==1
426                 Impulse_Object=[......
427                     zeros(size(Impulse_Object,1),.....
428                     floor((R-C)/2)) Impulse_Object .......
429                     zeros(size(Impulse_Object,1),......
```

```
                floor((R-C)/2)+1)];
        end
    elseif R<C
        if mod(C-R,2)==0
            Impulse_Object=[......
                zeros((C-R)/2,size(Impulse_Object,2));....
                Impulse_Object;....
                zeros((C-R)/2,size(Impulse_Object,2))];
        elseif mod(C-R,2)==1
            Impulse_Object=[zeros(floor((C-R)/2),......
                size(Impulse_Object,2));Impulse_Object;......
                zeros(floor((C-R)/2)+1,......
                size(Impulse_Object,2))];
        end
    end


    U1=Impulse_Object; % The transmitted field from
                       % the input plane P1.
    [M N]=size(U1);

    L=10; % The physical side length of the
          % array which holds the input
          % plane P1 in meters (m).
    dx1_Input=L/N; % The sample spacing in the input plane
                   % array in the direction of the spatial
                   % space coordinate x1 in meters (m).
    dy1_Input=L/M; % The sample spacing in the input plane
                   % array in the direction of the spatial
                   % space coordinate y1 in meters (m).
    x1_Axis_Input=-floor(N/2)*dx1_Input:dx1_Input:......
        ceil(N/2)*dx1_Input-dx1_Input; % Sampling the input
                                       % plane P1 in the
                                       % direction of the
                                       % spatial space
                                       % coordinate x1.
    y1_Axis_Input=-floor(M/2)*dy1_Input:dy1_Input:.....
        ceil(M/2)*dy1_Input-dy1_Input; % Sampling the input
                                       % plane P1 in the
                                       % direction of the
                                       % spatial space
                                       % coordinate y1.


    U2=fftshift(fft2(fftshift(U1))); % The Fourier transform of
                                     % the transmitted field in
                                     % the back focal plane of
                                     % the lens L2.
    I=(abs(U2)).^2; % The intensity of the Fourier transformed
                    % field in the plane P2.
```

```
480
481         lambda=550e-9; % The wavelength in meters (m).
482         f=0.055; % The focal length in meters (m).
483         dx2=(lambda*f)/(N*dx1_Input); % The sample spacing in the
484                                       % plane P2 in the direction
485                                       % of the spatial space
486                                       % coordinate x2 in meters
487                                       % (m).
488         dy2=(lambda*f)/(M*dy1_Input); % The sample spacing in the
489                                       % plane P2 in the direction
490                                       % of the spatial space
491                                       % coordinate y2 in meters
492                                       % (m).
493         x2_Axis=-floor(N/2)*dx2:dx2:........
494             ceil(N/2)*dx2-dx2; % Sampling the plane P2 in the
495                                % direction of the spatial space
496                                % coordinate x2.
497         y2_Axis=-floor(M/2)*dy2:dy2:.....
498             ceil(M/2)*dy2-dy2; % Sampling the plane P2 in the
499                                % direction of the spatial space
500                                % coordinate y2.
501
502
503         U3=ifftshift(ifft2(ifftshift(I))); % The crosscorrelated
504                                            % field in the back
505                                            % focal plane of the
506                                            % lens L4.
507
508         dx3=(lambda*f)/(N*dx2); % The sample spacing in the plane
509                                 % P3 in the direction of the
510                                 % spatial space coordinate x3 in
511                                 % meters (m).
512         dy3=(lambda*f)/(M*dy2); % The sample spacing in the plane
513                                 % P3 in the direction of the
514                                 % spatial space coordinate y3 in
515                                 % meters (m).
516         x3_Axis=-floor(N/2)*dx3:dx3:.....
517             ceil(N/2)*dx3-dx3; % Sampling the plane P3 in the
518                                % direction of the spatial space
519                                % coordinate x3.
520         y3_Axis=-floor(M/2)*dy3:dy3:......
521             ceil(M/2)*dy3-dy3; % Sampling the plane P3 in the
522                                % direction of the spatial space
523                                % coordinate y3.
524
525
526         % Synthesizing a desired filtering mask then filtering the
527         % crosscorrelated field in the plane P3.
528         Cen=floor(M/2)+1; % The center of the filtering mask.
529         Cen1=Cen-(Y+dis); % The center of the desired
```

```
                        % crosscorrelated field.
        Wh1=q; % The width of the impulse response in
               % the direction of the x1-coordinate.
        Wg1=c; % The width of the object in the
               % direction of the x1-coordinate.

        Mask=zeros(M,N);
        for P=Cen1-floor((Wh+Wg)/2):Cen1+ceil((Wh+Wg)/2)
            for Q=Cen-floor((Wh1+Wg1)/2):Cen+ceil((Wh1+Wg1)/2)
                Mask(P,Q)=1;
            end
        end

        Cross_Field=Mask.*U3; % The filtered crosscorrelated
                              % field in the plane P3.


        % For simplicity, instead of processing the entire image of
        % the filtered crosscorrelated field, we select only the
        % crosscorrelated field of interest.
        P=Cen1-floor((Wh+Wg)/2):Cen1+ceil((Wh+Wg)/2);
        Q=Cen-floor((Wh1+Wg1)/2):Cen+ceil((Wh1+Wg1)/2);
        Desired_Cross_Field=U3(P,Q);


        Max_Desired_Cross_Fields(n,m)=......
            max(max(Desired_Cross_Field));
        n


%         figure
%         subplot(2,1,1)
%         imshow(Impulse1)
%         if m==1
%             title({['This Is To Show How Good the Impulses '.....
%                 'Threshold Is,']; blanks(1);.....
%                 ['Impulse Response No.' num2str(m) .....
%                 ' for Mr. Mansour Alshammari']})
%         elseif m==2
%             title({['This Is To Show How Good the Impulses '.....
%                 'Threshold Is,'];blanks(1);.....
%                 ['Impulse Response No.' num2str(m) .....
%                 ' for Mr. Methkir Alharthee']})
%         elseif m==3
%             title({['This Is To Show How Good the Impulses '....
%                 'Threshold Is,'];blanks(1);....
%                 ['Impulse Response No.' num2str(m) ......
%                 ' for Mr. Mohammed Hanafy']})
%         end
%         subplot(2,1,2)
```

```
580 %           imshow(Impulse)
581 %           if m==1
582 %               title({['Normalized Impulse Response No.' ....
583 %                   num2str(m) ' for Mr. Mansour Alshammari']})
584 %           elseif m==2
585 %               title({['Normalized Impulse Response No.' .....
586 %                   num2str(m) ' for Mr. Methkir Alharthee']})
587 %           elseif m==3
588 %               title({['Normalized Impulse Response No.' .....
589 %                   num2str(m) ' for Mr. Mohammed Hanafy']})
590 %           end
591 %
592 %           figure
593 %           subplot(2,1,1)
594 %           imshow(uint8(Impulse1))
595 %           if m==1
596 %               title(['Impulse Response No.' num2str(m) .....
597 %                   ' for Mr. Mansour Alshammari'])
598 %           elseif m==2
599 %               title(['Impulse Response No.' num2str(m) ....
600 %                   ' for Mr. Methkir Alharthee'])
601 %           elseif m==3
602 %               title(['Impulse Response No.' num2str(m) .....
603 %                   ' for Mr. Mohammed Hanafy'])
604 %           end
605 %           subplot(2,1,2)
606 %           imshow(uint8(Impulse))
607 %           if m==1
608 %               title(['Normalized Impulse Response No.' .....
609 %                   num2str(m) ' for Mr. Mansour Alshammari'])
610 %           elseif m==2
611 %               title(['Normalized Impulse Response No.' .....
612 %                   num2str(m) ' for Mr. Methkir Alharthee'])
613 %           elseif m==3
614 %               title(['Normalized Impulse Response No.' ....
615 %                   num2str(m) ' for Mr. Mohammed Hanafy'])
616 %           end
617 %
618 %           figure('units','centimeters','position',[7 1.2 25 16.9])
619 %           imagesc(x1_Axis_Input,y1_Axis_Input,U1)
620 %           colorbar
621 %           if n<=L1
622 %               if m==1
623 %                   title({['The Transmitted Field from the '....
624 %                       'Input Plane P_1'];['(Impulse No.1 Is '....
625 %                       'for Mr. Mansour Alshammari as Well '.....
626 %                       'as Object No.' num2str(n) .....
627 %                       ' Is for Him)']})
628 %               elseif m==2
629 %                   title({['The Transmitted Field from the '.....
```

```
630 %                     'Input Plane P_1'];['(Impulse No.2 Is '....
631 %                     'for Mr. Methkir Alharthee and Object '....
632 %                     'No.' num2str(n) .....
633 %                     ' Is for Mr. Mansour Alshammari)'']})
634 %                 else
635 %                     title({['The Transmitted Field from the '.....
636 %                     'Input Plane P_1'];['(Impulse No.3 Is '....
637 %                     'for Mr. Mohammed Hanafy and Object '....
638 %                     'No.' num2str(n) .......
639 %                     ' Is for Mr. Mansour Alshammari)'']})
640 %                 end
641 %             elseif n>L1 && n<=L2
642 %                 if m==1
643 %                     title({['The Transmitted Field from the '....
644 %                     'Input Plane P_1'];['(Impulse No.1 Is '....
645 %                     'for Mr. Mansour Alshammari and Object '....
646 %                     'No.' num2str(n) ......
647 %                     ' Is for Mr. Methkir Alharthee)'']})
648 %                 elseif m==2
649 %                     title({['The Transmitted Field from the '....
650 %                     'Input Plane P_1'];['(Impulse No.2 Is '....
651 %                     'for Mr. Methkir Alharthee as Well as '....
652 %                     'Object No.' num2str(n) ' Is for Him)'']})
653 %                 else
654 %                     title({['The Transmitted Field from the '....
655 %                     'Input Plane P_1'];['(Impulse No.3 Is '....
656 %                     'for Mr. Mohammed Hanafy and Object No.'....
657 %                     num2str(n) ........
658 %                     ' Is for Mr. Methkir Alharthee)'']})
659 %                 end
660 %             elseif n>L2 && n<=L3
661 %                 if m==1
662 %                     title({['The Transmitted Field from the '....
663 %                     'Input Plane P_1'];['(Impulse No.1 Is '....
664 %                     'for Mr. Mansour Alshammari and Object '....
665 %                     'No.' num2str(n) .......
666 %                     ' Is for Mr. Mohammed Hanafy)'']})
667 %                 elseif m==2
668 %                     title({['The Transmitted Field from the '....
669 %                     'Input Plane P_1'];['(Impulse No.2 Is '....
670 %                     'for Mr. Methkir Alharthee and '.....
671 %                     'Object No.' num2str(n) ......
672 %                     ' Is for Mr. Mohammed Hanafy)'']})
673 %                 else
674 %                     title({['The Transmitted Field from the '....
675 %                     'Input Plane P_1'];['(Impulse No.3 Is '....
676 %                     'for Mr. Mohammed Hanafy as Well as '.....
677 %                     'Object No.' num2str(n) ' Is for Him)'']})
678 %                 end
679 %             end
```

```
680 %           colormap('gray')
681 %           xlabel('x_1 (m)')
682 %           ylabel('y_1 (m)')
683 %
684 %           figure('units','centimeters','position',[7 1.2 25 16.9])
685 %           imagesc(x2_Axis,y2_Axis,255*(I/max(max(I))))
686 %           colorbar
687 %           if n<=L1
688 %               if m==1
689 %                   title({['The Incident Intensity on the '.....
690 %                       'Plane P_2'];['(Impulse No.1 Is for '....
691 %                       'Mr. Mansour Alshammari as Well as '....
692 %                       'Object No.' num2str(n) ' Is for Him)']})
693 %               elseif m==2
694 %                   title({['The Incident Intensity on the '.....
695 %                       'Plane P_2'];['(Impulse No.2 Is for '.....
696 %                       'Mr. Methkir Alharthee and Object No.'....
697 %                       num2str(n) ..........
698 %                       ' Is for Mr. Mansour Alshammari)']})
699 %               else
700 %                   title({['The Incident Intensity on the '....
701 %                       'Plane P_2'];['(Impulse No.3 Is for '....
702 %                       'Mr. Mohammed Hanafy and Object No.' ....
703 %                       num2str(n) ......
704 %                       ' Is for Mr. Mansour Alshammari)']})
705 %               end
706 %           elseif n>L1 && n<=L2
707 %               if m==1
708 %                   title({['The Incident Intensity on the '.....
709 %                       'Plane P_2'];['(Impulse No.1 Is for '.....
710 %                       'Mr. Mansour Alshammari and Object No.'....
711 %                       num2str(n) .......
712 %                       ' Is for Mr. Methkir Alharthee)']})
713 %               elseif m==2
714 %                   title({['The Incident Intensity on the '.....
715 %                       'Plane P_2'];['(Impulse No.2 Is for '.....
716 %                       'Mr. Methkir Alharthee as Well as '.....
717 %                       'Object No.' num2str(n) ' Is for Him)']})
718 %               else
719 %                   title({['The Incident Intensity on the '....
720 %                       'Plane P_2'];['(Impulse No.3 Is for '....
721 %                       'Mr. Mohammed Hanafy and '......
722 %                       'Object No.' num2str(n) ......
723 %                       ' Is for Mr. Methkir Alharthee)']})
724 %               end
725 %           elseif n>L2 && n<=L3
726 %               if m==1
727 %                   title({['The Incident Intensity on the '....
728 %                       'Plane P_2'];['(Impulse No.1 Is for '.....
729 %                       'Mr. Mansour Alshammari and '......
```

```
730 %                     'Object No.' num2str(n) ......
731 %                     ' Is for Mr. Mohammed Hanafy)']})
732 %                 elseif m==2
733 %                     title({['The Incident Intensity on the '....
734 %                     'Plane P_2'];['(Impulse No.2 Is for '....
735 %                     'Mr. Methkir Alharthee and '.....
736 %                     'Object No.' num2str(n) .....
737 %                     ' Is for Mr. Mohammed Hanafy)']})
738 %                 else
739 %                     title({['The Incident Intensity on the '.....
740 %                     'Plane P_2'];['(Impulse No.3 Is for '.....
741 %                     'Mr. Mohammed Hanafy as Well as '....
742 %                     'Object No.' num2str(n) ' Is for Him)']})
743 %                 end
744 %             end
745 %             colormap('gray')
746 %             xlabel('x_2 (m)')
747 %             ylabel('y_2 (m)')
748 %
749 %             figure('units','centimeters','position',[7 1.2 25 16.9])
750 %             imagesc(x3_Axis,y3_Axis,U3)
751 %             colorbar
752 %             if n<=L1
753 %                 if m==1
754 %                     title({['The Crosscorrelated Field in '.....
755 %                     'the Plane P_3'];['(Impulse No.1 Is '.....
756 %                     'for Mr. Mansour Alshammari as Well as '....
757 %                     'Object No.' num2str(n) ' Is for Him)']})
758 %                 elseif m==2
759 %                     title({['The Crosscorrelated Field in '....
760 %                     'the Plane P_3'];['(Impulse No.2 Is '.....
761 %                     'for Mr. Methkir Alharthee and '.....
762 %                     'Object No.' num2str(n) ......
763 %                     ' Is for Mr. Mansour Alshammari)']})
764 %                 else
765 %                     title({['The Crosscorrelated Field in '....
766 %                     'the Plane P_3'];['(Impulse No.3 Is '.....
767 %                     'for Mr. Mohammed Hanafy and '.....
768 %                     'Object No.' num2str(n) ......
769 %                     ' Is for Mr. Mansour Alshammari)']})
770 %                 end
771 %             elseif n>L1 && n<=L2
772 %                 if m==1
773 %                     title({['The Crosscorrelated Field in '....
774 %                     'the Plane P_3'];['(Impulse No.1 Is '....
775 %                     'for Mr. Mansour Alshammari and '.....
776 %                     'Object No.' num2str(n) .....
777 %                     ' Is for Mr. Methkir Alharthee)']})
778 %                 elseif m==2
779 %                     title({['The Crosscorrelated Field in '....
```

```
780 %                     'the Plane P_3'];['(Impulse No.2 Is '.....
781 %                     'for Mr. Methkir Alharthee as Well as '....
782 %                     'Object No.' num2str(n) ' Is for Him)']})
783 %             else
784 %                 title({['The Crosscorrelated Field in '.....
785 %                     'the Plane P_3'];['(Impulse No.3 Is '....
786 %                     'for Mr. Mohammed Hanafy and '.....
787 %                     'Object No.' num2str(n) .....
788 %                     ' Is for Mr. Methkir Alharthee)']})
789 %             end
790 %         elseif n>L2 && n<=L3
791 %             if m==1
792 %                 title({['The Crosscorrelated Field in '.....
793 %                     'the Plane P_3'];['(Impulse No.1 Is '....
794 %                     'for Mr. Mansour Alshammari and '.....
795 %                     'Object No.' num2str(n) .....
796 %                     ' Is for Mr. Mohammed Hanafy)']})
797 %             elseif m==2
798 %                 title({['The Crosscorrelated Field in '.....
799 %                     'the Plane P_3'];['(Impulse No.2 Is '.....
800 %                     'for Mr. Methkir Alharthee and '.....
801 %                     'Object No.' num2str(n) .....
802 %                     ' Is for Mr. Mohammed Hanafy)']})
803 %             else
804 %                 title({['The Crosscorrelated Field in '....
805 %                     'the Plane P_3'];['(Impulse No.3 Is '.....
806 %                     'for Mr. Mohammed Hanafy as Well as '.....
807 %                     'Object No.' num2str(n) ' Is for Him)']})
808 %             end
809 %         end
810 %         colormap('gray')
811 %         xlabel('x_3 (m)')
812 %         ylabel('y_3 (m)')
813 %
814 %         figure('units','centimeters','position',[7 1.2 25 16.9])
815 %         imagesc(x3_Axis,y3_Axis,Mask)
816 %         colorbar
817 %         if n<=L1
818 %             if m==1
819 %                 title({'The Adaptive Filtering Mask';....
820 %                     ['(Impulse No.1 Is for Mr. Mansour '.....
821 %                     'Alshammari as Well as Object No.' ....
822 %                     num2str(n) ' Is for Him)']})
823 %             elseif m==2
824 %                 title({'The Adaptive Filtering Mask';.....
825 %                     ['(Impulse No.2 Is for Mr. Methkir '.....
826 %                     'Alharthee and Object No.' num2str(n) ....
827 %                     ' Is for Mr. Mansour Alshammari)']})
828 %             else
829 %                 title({'The Adaptive Filtering Mask';.....
```

```
830 %                     ['(Impulse No.3 Is for Mr. Mohammed '....
831 %                     'Hanafy and Object No.' num2str(n) .....
832 %                     ' Is for Mr. Mansour Alshammari)']})
833 %             end
834 %         elseif n>L1 && n<=L2
835 %             if m==1
836 %                 title({'The Adaptive Filtering Mask';.....
837 %                     ['(Impulse No.1 Is for Mr. Mansour '.....
838 %                     'Alshammari and Object No.' num2str(n) ....
839 %                     ' Is for Mr. Methkir Alharthee)']})
840 %             elseif m==2
841 %                 title({'The Adaptive Filtering Mask';.....
842 %                     ['(Impulse No.2 Is for Mr. Methkir '.....
843 %                     'Alharthee as Well as Object No.' .....
844 %                     num2str(n) ' Is for Him)']})
845 %             else
846 %                 title({'The Adaptive Filtering Mask';.....
847 %                     ['(Impulse No.3 Is for Mr. Mohammed '.....
848 %                     'Hanafy and Object No.' num2str(n) .....
849 %                     ' Is for Mr. Methkir Alharthee)']})
850 %             end
851 %         elseif n>L2 && n<=L3
852 %             if m==1
853 %                 title({'The Adaptive Filtering Mask';.....
854 %                     ['(Impulse No.1 Is for Mr. Mansour '.....
855 %                     'Alshammari and Object No.' num2str(n) ....
856 %                     ' Is for Mr. Mohammed Hanafy)']})
857 %             elseif m==2
858 %                 title({'The Adaptive Filtering Mask';.....
859 %                     ['(Impulse No.2 Is for Mr. Methkir '.....
860 %                     'Alharthee and Object No.' num2str(n) ....
861 %                     ' Is for Mr. Mohammed Hanafy)']})
862 %             else
863 %                 title({'The Adaptive Filtering Mask';.....
864 %                     ['(Impulse No.3 Is for Mr. Mohammed '.....
865 %                     'Hanafy as Well as Object No.' .....
866 %                     num2str(n) ' Is for Him)']})
867 %             end
868 %         end
869 %         colormap('gray')
870 %         xlabel('x_3 (m)')
871 %         ylabel('y_3 (m)')
872 %
873 %         figure('units','centimeters','position',[7 1.2 25 16.9])
874 %         imagesc(x3_Axis,y3_Axis,Cross_Field)
875 %         colorbar
876 %         if n<=L1
877 %             if m==1
878 %                 title({['The Filtered Crosscorrelated '....
879 %                     'Field in the Plane P_3'];['(Impulse '....
```

```
880 %                     'No.1 Is for Mr. Mansour Alshammari '....
881 %                     'as Well as Object No.' num2str(n) ....
882 %                     ' Is for Him)']})
883 %                 elseif m==2
884 %                     title({['The Filtered Crosscorrelated '....
885 %                     'Field in the Plane P_3'];['(Impulse '....
886 %                     'No.2 Is for Mr. Methkir Alharthee '.....
887 %                     'and Object No.' num2str(n) ......
888 %                     ' Is for Mr. Mansour Alshammari)']})
889 %                 else
890 %                     title({['The Filtered Crosscorrelated '....
891 %                     'Field in the Plane P_3'];['(Impulse '....
892 %                     'No.3 Is for Mr. Mohammed Hanafy and '....
893 %                     'Object No.' num2str(n) ......
894 %                     ' Is for Mr. Mansour Alshammari)']})
895 %                 end
896 %             elseif n>L1 && n<=L2
897 %                 if m==1
898 %                     title({['The Filtered Crosscorrelated '.....
899 %                     'Field in the Plane P_3'];['(Impulse '....
900 %                     'No.1 Is for Mr. Mansour Alshammari '.....
901 %                     'and Object No.' num2str(n) ......
902 %                     ' Is for Mr. Methkir Alharthee)']})
903 %                 elseif m==2
904 %                     title({['The Filtered Crosscorrelated '.....
905 %                     'Field in the Plane P_3'];['(Impulse '....
906 %                     'No.2 Is for Mr. Methkir Alharthee '....
907 %                     'as Well as Object No.' num2str(n) .....
908 %                     ' Is for Him)']})
909 %                 else
910 %                     title({['The Filtered Crosscorrelated '....
911 %                     'Field in the Plane P_3'];['(Impulse '....
912 %                     'No.3 Is for Mr. Mohammed Hanafy and '....
913 %                     'Object No.' num2str(n) ......
914 %                     ' Is for Mr. Methkir Alharthee)']})
915 %                 end
916 %             elseif n>L2 && n<=L3
917 %                 if m==1
918 %                     title({['The Filtered Crosscorrelated '....
919 %                     'Field in the Plane P_3'];['(Impulse '....
920 %                     'No.1 Is for Mr. Mansour Alshammari '....
921 %                     'and Object No.' num2str(n) ......
922 %                     ' Is for Mr. Mohammed Hanafy)']})
923 %                 elseif m==2
924 %                     title({['The Filtered Crosscorrelated '....
925 %                     'Field in the Plane P_3'];['(Impulse '....
926 %                     'No.2 Is for Mr. Methkir Alharthee '.....
927 %                     'and Object No.' num2str(n) .....
928 %                     ' Is for Mr. Mohammed Hanafy)']})
929 %                 else
```

```
930 %                 title({['The Filtered Crosscorrelated '.....
931 %                     'Field in the Plane P_3'];['(Impulse '....
932 %                     'No.3 Is for Mr. Mohammed Hanafy as '....
933 %                     'Well as Object No.' num2str(n) ......
934 %                     ' Is for Him)']})
935 %             end
936 %         end
937 %         colormap('gray')
938 %         xlabel('x_3 (m)')
939 %         ylabel('y_3 (m)')
940 %
941 %         figure('units','centimeters','position',[7 1.2 25 16.9])
942 %         imagesc(x3_Axis,y3_Axis,Desired_Cross_Field)
943 %         colorbar
944 %         if n<=L1
945 %             if m==1
946 %                 title({['The Crosscorrelated Field of '.....
947 %                     'Interest in the Plane P_3'];.....
948 %                     ['(Impulse No.1 Is for Mr. Mansour '....
949 %                     'Alshammari as Well as Object No.' ....
950 %                     num2str(n) ' Is for Him)']})
951 %             elseif m==2
952 %                 title({['The Crosscorrelated Field of '.....
953 %                     'Interest in the Plane P_3'];['(Impulse'....
954 %                     ' No.2 Is for Mr. Methkir Alharthee '.....
955 %                     'and Object No.' num2str(n) ......
956 %                     ' Is for Mr. Mansour Alshammari)']})
957 %             else
958 %                 title({['The Crosscorrelated Field of '.....
959 %                     'Interest in the Plane P_3'];['(Impulse'....
960 %                     ' No.3 Is for Mr. Mohammed Hanafy and '....
961 %                     'Object No.' num2str(n) ......
962 %                     ' Is for Mr. Mansour Alshammari)']})
963 %             end
964 %         elseif n>L1 && n<=L2
965 %             if m==1
966 %                 title({['The Crosscorrelated Field of '......
967 %                     'Interest in the Plane P_3'];['(Impulse'....
968 %                     ' No.1 Is for Mr. Mansour Alshammari '....
969 %                     'and Object No.' num2str(n) ......
970 %                     ' Is for Mr. Methkir Alharthee)']})
971 %             elseif m==2
972 %                 title({['The Crosscorrelated Field of '.....
973 %                     'Interest in the Plane P_3'];['(Impulse'....
974 %                     ' No.2 Is for Mr. Methkir Alharthee '.....
975 %                     'as Well as Object No.' num2str(n) .....
976 %                     ' Is for Him)']})
977 %             else
978 %                 title({['The Crosscorrelated Field of '.....
979 %                     'Interest in the Plane P_3'];['(Impulse'....
```

```
980 %                       ' No.3 Is for Mr. Mohammed Hanafy and '....
981 %                       'Object No.' num2str(n) .....
982 %                       ' Is for Mr. Methkir Alharthee)'']})
983 %               end
984 %           elseif n>L2 && n<=L3
985 %               if m==1
986 %                   title({['The Crosscorrelated Field of '.....
987 %                       'Interest in the Plane P_3'];['(Impulse'....
988 %                       ' No.1 Is for Mr. Mansour Alshammari '.....
989 %                       'and Object No.' num2str(n) ......
990 %                       ' Is for Mr. Mohammed Hanafy)'']})
991 %               elseif m==2
992 %                   title({['The Crosscorrelated Field of '.....
993 %                       'Interest in the Plane P_3'];['(Impulse'....
994 %                       ' No.2 Is for Mr. Methkir Alharthee '....
995 %                       'and Object No.' num2str(n) ......
996 %                       ' Is for Mr. Mohammed Hanafy)'']})
997 %               else
998 %                   title({['The Crosscorrelated Field of '.....
999 %                       'Interest in the Plane P_3'];['(Impulse'....
1000 %                      ' No.3 Is for Mr. Mohammed Hanafy '.....
1001 %                      'as Well as Object No.' ......
1002 %                      num2str(n) ' Is for Him)'']})
1003 %               end
1004 %           end
1005 %           colormap('gray')
1006 %           xlabel('x_3 (m)')
1007 %           ylabel('y_3 (m)')
1008 %
1009 %
1010 %
1011 %           %%%% Note that, "imshow" is better to be used instead
1012 %           %%%% of "imagesc" during designing the adaptive mask
1013 %           %%%% because "imshow" dispalays the cross-correlations
1014 %           %%%% in the plane P3 clearer than "imagesc"!!
1015 %
1016 %
1017 %
1018 %           disp(['Please, press any keyboard button to explore '....
1019 %               'the remaining crosscorrelated fields >>>>>>>'])
1020 %           pause
1021 %           clc
1022 %           close all
1023       end
1024   end
1025
1026
1027   % Testing the process of object recognition.
1028   % Note that, the objects here are known for us in order to use them
1029   % for testing the object recognition process as well as setting up
```

```
% a recognition threshold.

% The elements in each of the following vectors represent the
% maximum values of the desired crosscorrelated fields between the
% objects and their corresponding impulse. Note that, each impulse
% and its corresponding objects have a same face so the vectors
% will include the biggest crosscorrelations.
V1=(Max_Desired_Cross_Fields(1:L1,1)).';
V2=(Max_Desired_Cross_Fields(L1+1:L2,2)).';
V3=(Max_Desired_Cross_Fields(L2+1:L3,3)).';

% The calculation of the mean and the standard deviation for each
% vector of the biggest crosscorrelations and stacking them in a
% vector for the means and another for the standard deviations.
% This is done for setting up a threshold for object recognition.
V_Mean=[mean(V1);mean(V2);mean(V3)];
V_STD=[std(V1);std(V2);std(V3)];
save('Computed Means for Object Recognition','V_Mean')
save('Computed STDs for Object Recognition','V_STD')

fid=fopen('Object Recognition Results for Testing.txt',....
    'w'); % A text file for typing
          % the object recognition
          % results for testing.
fprintf(fid,['\n ***** The Object Recognition Results '.....
    'for Testing Obtained from the Code of the *****\r\n']);
fprintf(fid,['   **** Joint Transform Correlator (JTC) for '.....
    'Testing and Setting up Thresholds ****\r\n\n']);
fprintf(fid,['The Object No.   The Object Is Originally for'......
    '   The Object Is Recognized as   The Status\r\n']);
fprintf(fid,['==============   ============================'.....
    '   ==========\r\n']);

Failures_Vector=zeros(1,Total_No_of_Objects); % A vector for
                                              % counting the number
                                              % of failures in the
                                              % object recognition
                                              % process.

Latex_Matrix=cell(Total_No_of_Objects,4); % This matrix is used
                                          % for creating a table
                                          % in Latex.
for w=1:Total_No_of_Objects
    % The object recognition process.
    for ii=1:Total_No_of_Impulses
        if  max(Max_Desired_Cross_Fields(w,:))==.....
                Max_Desired_Cross_Fields(w,ii) && .....
                max(Max_Desired_Cross_Fields(w,:))>=....
                (V_Mean(ii,1)-V_STD(ii,1)) && .....
                max(Max_Desired_Cross_Fields(w,:))<=....
```

```
                (V_Mean(ii,1)+V_STD(ii,1))
            S=transpose(struct2cell(Impulses));
            d=sortrows(S,1);
            z=d(:,1);
            Recognized_As=char(z(ii,1));
            Recognized_Object_Location=......
                fullfile(Impulses_Folder,Recognized_As);
            Recognized_Object=im2double(rgb2gray(.....
                imread(Recognized_Object_Location)));
            break
        else
            Recognized_As='Unknown Object';
            Recognized_Object=imread('Unknown_Object.jpg');
        end
    end

    % Defining the object.
    if w<=L1
        name='Mr. Mansour Alshammari';
    elseif L1<w && w<=L2
        name='Mr. Methkir Alharthee';
    else name='Mr. Mohammed Hanafy';
    end

    Str1=strcmp(Recognized_As(1:length(Recognized_As)-6),....
        'Mr. Mansour Alshammari');
    Str2=strcmp(Recognized_As(1:length(Recognized_As)-6),.....
        'Mr. Methkir Alharthee');
    Str3=strcmp(Recognized_As(1:length(Recognized_As)-6),....
        'Mr. Mohammed Hanafy');
    F='Success';
    if w<=L1 && Str1==0;
        F='Failure';
        Failures_Vector(1,w)=1;
    elseif w>L1 && w<=L2 && Str2==0;
        F='Failure';
        Failures_Vector(1,w)=1;
    elseif w>L2 && w<=L3 && Str3==0;
        F='Failure';
        Failures_Vector(1,w)=1;
    end

%     Object_Number=[num2str(w) '.jpg'];
%     Object_Location=fullfile(Objects_Folder,Object_Number);
%     Object=im2double(rgb2gray(imread(Object_Location)));
%     subplot(2,1,1)
%     imshow(Object)
%     title({['Object No.' num2str(w) ' Is Originally for'];name})
%     subplot(2,1,2)
%     imshow(Recognized_Object)
```

```
%     if Str1==1 || Str2==1 || Str3==1
%         title({'The Object Is Recognized As';.............
%             Recognized_As(1:length(Recognized_As)-6)})
%     else title({'The Object Is Recognized As' ......
%             ['an' blanks(1) Recognized_As]})
%     end

    if Str1==1 || Str2==1 || Str3==1
        fprintf(fid,['%0.3d\t\t\t     %-23s\t    '.......
            '%-24s\t  %s \r\n\n'],w,name,......
            Recognized_As(1:length(Recognized_As)-6),F);
        Latex_Matrix(w,1:4)={num2str(w) name .....
            Recognized_As(1:length(Recognized_As)-6) F};
    else fprintf(fid,['%0.3d\t\t\t     %-23s\t    %-24s\t'.....
            '  %s \r\n\n'],w,name,Recognized_As,F);
        Latex_Matrix(w,1:4)={num2str(w) name Recognized_As F};
    end

%     disp(['Please, press Enter button to see another '......
%         'object and its recognition....'])
%     pause
%     clc
end

Total_Number_of_Failures=sum(Failures_Vector); % The total number
                                               % of failures in the
                                               % object recognition
                                               % process.

fprintf(fid,['=============================================='....
    '=========================================\r\n']);
fprintf(fid,['** The Total Number of Successes: %0.3d out '......
    'of %0.3d (%3.4f%%) \r\n\n'],Total_No_of_Objects-......
    Total_Number_of_Failures,Total_No_of_Objects,......
    ((Total_No_of_Objects-Total_Number_of_Failures)/.....
    Total_No_of_Objects)*100);
fprintf(fid,['** The Total Number of Failures: %0.3d out'.....
    ' of %0.3d (%3.4f%%) \r\n\n'],....
    Total_Number_of_Failures,Total_No_of_Objects,.....
    (Total_Number_of_Failures/Total_No_of_Objects)*100);
fclose(fid);

close all
clc

disp(['Please, see the documented results of object '......
    'recognition in the open'])
disp(['text file then press any keyboard button to '.....
    'resume the code >>>>'])
Text='Object Recognition Results for Testing.txt';
```

```
open(Text) % Opening the text file which contains
           % the results of object recognition.
pause
clc
open('The_JTC_Testing_and_Setting_up_Thresholds.m')


% Testing the process of object detection.
% Note that, the objects here are known for us in order to use them
% for testing the object detection process as well as setting up a
% detection threshold.

% The calculation of the mean and standard deviation for the
% crosscorrelations between the objects and each impulse response.
% This is done for setting up a threshold for object detection.
Vectorization=reshape(Max_Desired_Cross_Fields,1,.....
    size(Max_Desired_Cross_Fields,1)*.....
    size(Max_Desired_Cross_Fields,2));
Mean=mean(Vectorization);
STD=std(Vectorization);
save('Computed Mean for Object Detection','Mean')
save('Computed STD for Object Detection','STD')

fid=fopen('Object Detection Results for Testing.txt',.....
    'w'); % A text file for typing
          % the object detection
          % results for testing.
fprintf(fid,['\n ***** The Object Detection Results '.......
    'for Testing Obtained from the Code of the *****\r\n']);
fprintf(fid,['  **** Joint Transform Correlator (JTC) for '.....
    'Testing and Setting up Thresholds ****\r\n\n']);
fprintf(fid,['The Object No.     The Object Originally '.......
    'Is     The Detected Object Is     The Status\r\n']);
fprintf(fid,['==============     ========================'.....
    '     ======================     ==========\r\n']);

Failures_Vector1=zeros(1,.......
    Total_No_of_Objects); % A vector for counting the number
                          % of failures in the object detection
                          % process.

Latex_Matrix=cell(Total_No_of_Objects,4); % This matrix is used
                                          % for creating a table
                                          % in Latex.
for w=1:Total_No_of_Objects
    % The object detection process.
    if  max(Max_Desired_Cross_Fields(w,:))>=(Mean-STD) && .....
            max(Max_Desired_Cross_Fields(w,:))<=(Mean+STD)
        Detected_As='a face';
        Detected_Object=imread('A_Face.jpg');
```

```
    else
        Detected_As='not a face';
        Detected_Object=imread('Not_a_Face.jpg');
    end

    b='a face'; % The object originally is a face.

    Str=strcmp(Detected_As,'a face');
    F='Success';
    if w<=L3 && Str==0;
        F='Failure';
        Failures_Vector1(1,w)=1;
    end

%     Object_Number=[num2str(w) '.jpg'];
%     Object_Location=fullfile(Objects_Folder,Object_Number);
%     Object=im2double(rgb2gray(imread(Object_Location)));
%     subplot(2,1,1)
%     imshow(Object)
%     title(['Object No.' num2str(w) ' Originally Is'])
%     subplot(2,1,2)
%     imshow(Detected_Object)
%     title('It Is Detected As')

    if Str==1
        fprintf(fid,['%0.3d\t\t\t       %-23s\t    %-22s\t'.....
            '   %s \r\n\n'],w,b,Detected_As,F);
        Latex_Matrix(w,1:4)={num2str(w) b Detected_As F};
    else fprintf(fid,['%0.3d\t\t\t       %-23s\t    %-22s\t'....
            '   %s \r\n\n'],w,b,Detected_As,F);
        Latex_Matrix(w,1:4)={num2str(w) b Detected_As F};
    end

%     disp(['Please, press Enter button to see another '......
%         'object and its detection....'])
%     pause
%     clc
end

Total_Number_of_Failures1=sum(Failures_Vector1); % The total number
                                                 % of failures in
                                                 % the object
                                                 % detection
                                                 % process.

fprintf(fid,['=============================================='....
    '=====================================\r\n']);
fprintf(fid,['** The Total Number of Successes: %0.3d out '.....
    'of %0.3d (%3.4f%%) \r\n\n'],Total_No_of_Objects-.....
    Total_Number_of_Failures1,Total_No_of_Objects,.....
```

```
1280      ((Total_No_of_Objects-Total_Number_of_Failures1)/.....
1281      Total_No_of_Objects)*100);
1282  fprintf(fid,['** The Total Number of Failures: %0.3d '.....
1283      'out of %0.3d (%3.4f%%) \r\n\n'],Total_Number_of_Failures1,....
1284      Total_No_of_Objects,......
1285      (Total_Number_of_Failures1/Total_No_of_Objects)*100);
1286  fclose(fid);
1287
1288  close all
1289  clc
1290
1291  disp(['Please, see the documented results of object '.....
1292      'detection in the open'])
1293  disp(['text file then press any keyboard button to '.....
1294      'resume the code >>>>'])
1295  Text='Object Detection Results for Testing.txt';
1296  open(Text) % Opening the text file which contains
1297             % the results of object detection.
1298  pause
1299  clc
1300  open('The_JTC_Testing_and_Setting_up_Thresholds.m')
```

Appendix G

Results of the Optical Recognition

Table G.1: The recognition of all 108 objects by using the joint transform correlator (JTC).

Object No.	Input Face	Recognized Output Face	Status
1	Mr. Mansour Alshammari	Unknown Object	Failure
2	Mr. Mansour Alshammari	Unknown Object	Failure
3	Mr. Mansour Alshammari	Unknown Object	Failure
4	Mr. Mansour Alshammari	Unknown Object	Failure
5	Mr. Mansour Alshammari	Unknown Object	Failure
6	Mr. Mansour Alshammari	Unknown Object	Failure
7	Mr. Mansour Alshammari	Unknown Object	Failure
8	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
9	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
10	Mr. Mansour Alshammari	Unknown Object	Failure
11	Mr. Mansour Alshammari	Unknown Object	Failure
12	Mr. Mansour Alshammari	Unknown Object	Failure
13	Mr. Mansour Alshammari	Unknown Object	Failure
14	Mr. Mansour Alshammari	Unknown Object	Failure
15	Mr. Mansour Alshammari	Unknown Object	Failure
16	Mr. Mansour Alshammari	Unknown Object	Failure
17	Mr. Mansour Alshammari	Unknown Object	Failure
18	Mr. Mansour Alshammari	Unknown Object	Failure
19	Mr. Mansour Alshammari	Unknown Object	Failure
20	Mr. Mansour Alshammari	Unknown Object	Failure
21	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
22	Mr. Mansour Alshammari	Unknown Object	Failure

Continued on the next page ...

Object No.	Input Face	Recognized Output Face	Status
23	Mr. Mansour Alshammari	Unknown Object	Failure
24	Mr. Mansour Alshammari	Unknown Object	Failure
25	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
26	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
27	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
28	Mr. Mansour Alshammari	Unknown Object	Failure
29	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
30	Mr. Mansour Alshammari	Unknown Object	Failure
31	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
32	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
33	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
34	Mr. Mansour Alshammari	Unknown Object	Failure
35	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
36	Mr. Mansour Alshammari	Mr. Mohammed Hanafy	Failure
37	Mr. Methkir Alharthee	Unknown Object	Failure
38	Mr. Methkir Alharthee	Unknown Object	Failure
39	Mr. Methkir Alharthee	Unknown Object	Failure
40	Mr. Methkir Alharthee	Unknown Object	Failure
41	Mr. Methkir Alharthee	Unknown Object	Failure
42	Mr. Methkir Alharthee	Unknown Object	Failure
43	Mr. Methkir Alharthee	Unknown Object	Failure
44	Mr. Methkir Alharthee	Unknown Object	Failure
45	Mr. Methkir Alharthee	Unknown Object	Failure
46	Mr. Methkir Alharthee	Unknown Object	Failure
47	Mr. Methkir Alharthee	Unknown Object	Failure
48	Mr. Methkir Alharthee	Unknown Object	Failure
49	Mr. Methkir Alharthee	Unknown Object	Failure
50	Mr. Methkir Alharthee	Unknown Object	Failure
51	Mr. Methkir Alharthee	Unknown Object	Failure
52	Mr. Methkir Alharthee	Unknown Object	Failure
53	Mr. Methkir Alharthee	Unknown Object	Failure
54	Mr. Methkir Alharthee	Unknown Object	Failure
55	Mr. Methkir Alharthee	Unknown Object	Failure
56	Mr. Methkir Alharthee	Unknown Object	Failure
57	Mr. Methkir Alharthee	Unknown Object	Failure
58	Mr. Methkir Alharthee	Unknown Object	Failure
59	Mr. Methkir Alharthee	Unknown Object	Failure
60	Mr. Methkir Alharthee	Unknown Object	Failure
61	Mr. Methkir Alharthee	Unknown Object	Failure

Continued on the next page ...

Object No.	Input Face	Recognized Output Face	Status
62	Mr. Methkir Alharthee	Unknown Object	Failure
63	Mr. Methkir Alharthee	Unknown Object	Failure
64	Mr. Methkir Alharthee	Unknown Object	Failure
65	Mr. Methkir Alharthee	Unknown Object	Failure
66	Mr. Methkir Alharthee	Unknown Object	Failure
67	Mr. Methkir Alharthee	Unknown Object	Failure
68	Mr. Methkir Alharthee	Unknown Object	Failure
69	Mr. Methkir Alharthee	Unknown Object	Failure
70	Mr. Methkir Alharthee	Unknown Object	Failure
71	Mr. Methkir Alharthee	Unknown Object	Failure
72	Mr. Methkir Alharthee	Unknown Object	Failure
73	Mr. Mohammed Hanafy	Unknown Object	Failure
74	Mr. Mohammed Hanafy	Unknown Object	Failure
75	Mr. Mohammed Hanafy	Unknown Object	Failure
76	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
77	Mr. Mohammed Hanafy	Unknown Object	Failure
78	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
79	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
80	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
81	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
82	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
83	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
84	Mr. Mohammed Hanafy	Unknown Object	Failure
85	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
86	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
87	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
88	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
89	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
90	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
91	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
92	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
93	Mr. Mohammed Hanafy	Unknown Object	Failure
94	Mr. Mohammed Hanafy	Unknown Object	Failure
95	Mr. Mohammed Hanafy	Unknown Object	Failure
96	Mr. Mohammed Hanafy	Unknown Object	Failure
97	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
98	Mr. Mohammed Hanafy	Unknown Object	Failure
99	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
100	Mr. Mohammed Hanafy	Unknown Object	Failure

Continued on the next page ...

Object No.	Input Face	Recognized Output Face	Status
101	Mr. Mohammed Hanafy	Unknown Object	Failure
102	Mr. Mohammed Hanafy	Unknown Object	Failure
103	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
104	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
105	Mr. Mohammed Hanafy	Mr. Mohammed Hanafy	Success
106	Mr. Mohammed Hanafy	Unknown Object	Failure
107	Mr. Mohammed Hanafy	Unknown Object	Failure
108	Mr. Mohammed Hanafy	Unknown Object	Failure

The table end.

Appendix H

Results of the Optical Detection

Table H.1: The detection of all 108 objects by using the joint transform correlator (JTC).

Object No.	Input Object	Detected Output Object	Status
1	a face	not a face	Failure
2	a face	not a face	Failure
3	a face	not a face	Failure
4	a face	not a face	Failure
5	a face	not a face	Failure
6	a face	not a face	Failure
7	a face	not a face	Failure
8	a face	not a face	Failure
9	a face	a face	Success
10	a face	not a face	Failure
11	a face	not a face	Failure
12	a face	not a face	Failure
13	a face	not a face	Failure
14	a face	not a face	Failure
15	a face	not a face	Failure
16	a face	not a face	Failure
17	a face	not a face	Failure
18	a face	not a face	Failure
19	a face	not a face	Failure
20	a face	not a face	Failure
21	a face	a face	Success
22	a face	not a face	Failure

Continued on the next page ...

Object No.	Input Object	Detected Output Object	Status
23	a face	not a face	Failure
24	a face	not a face	Failure
25	a face	not a face	Failure
26	a face	a face	Success
27	a face	a face	Success
28	a face	not a face	Failure
29	a face	not a face	Failure
30	a face	a face	Success
31	a face	not a face	Failure
32	a face	not a face	Failure
33	a face	a face	Success
34	a face	a face	Success
35	a face	not a face	Failure
36	a face	a face	Success
37	a face	a face	Success
38	a face	a face	Success
39	a face	a face	Success
40	a face	a face	Success
41	a face	a face	Success
42	a face	a face	Success
43	a face	a face	Success
44	a face	a face	Success
45	a face	a face	Success
46	a face	a face	Success
47	a face	a face	Success
48	a face	a face	Success
49	a face	a face	Success
50	a face	a face	Success
51	a face	a face	Success
52	a face	a face	Success
53	a face	a face	Success
54	a face	a face	Success
55	a face	a face	Success
56	a face	a face	Success
57	a face	not a face	Failure
58	a face	a face	Success
59	a face	a face	Success
60	a face	a face	Success
61	a face	a face	Success

Continued on the next page ...

Object No.	Input Object	Detected Output Object	Status
62	a face	a face	Success
63	a face	a face	Success
64	a face	a face	Success
65	a face	a face	Success
66	a face	a face	Success
67	a face	a face	Success
68	a face	a face	Success
69	a face	a face	Success
70	a face	not a face	Failure
71	a face	not a face	Failure
72	a face	not a face	Failure
73	a face	a face	Success
74	a face	not a face	Failure
75	a face	not a face	Failure
76	a face	a face	Success
77	a face	not a face	Failure
78	a face	not a face	Failure
79	a face	not a face	Failure
80	a face	not a face	Failure
81	a face	not a face	Failure
82	a face	a face	Success
83	a face	not a face	Failure
84	a face	a face	Success
85	a face	not a face	Failure
86	a face	a face	Success
87	a face	not a face	Failure
88	a face	a face	Success
89	a face	a face	Success
90	a face	a face	Success
91	a face	a face	Success
92	a face	not a face	Failure
93	a face	a face	Success
94	a face	not a face	Failure
95	a face	not a face	Failure
96	a face	not a face	Failure
97	a face	not a face	Failure
98	a face	a face	Success
99	a face	not a face	Failure
100	a face	a face	Success

Continued on the next page ...

Object No.	Input Object	Detected Output Object	Status
101	a face	a face	Success
102	a face	a face	Success
103	a face	not a face	Failure
104	a face	a face	Success
105	a face	not a face	Failure
106	a face	a face	Success
107	a face	a face	Success
108	a face	a face	Success

The table end.

Appendix I

MSEs Plots of Reconstructing Some Training Faces

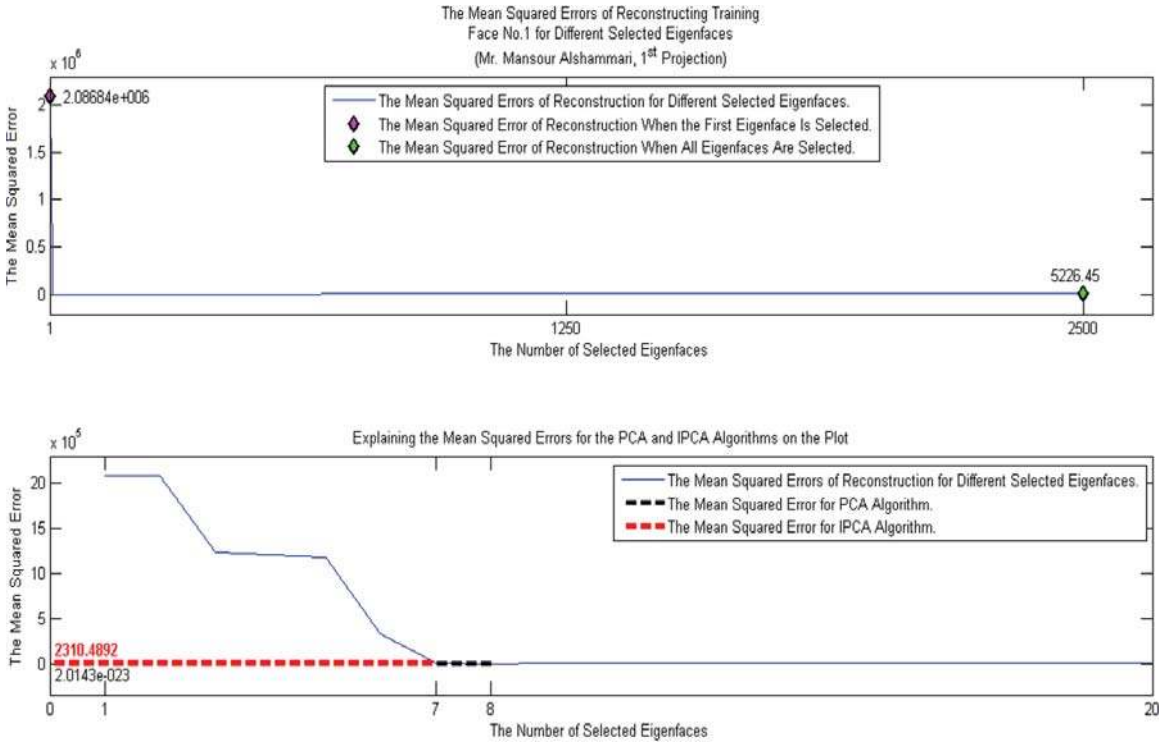

Figure I.1: The plot of the mean squared errors (MSEs) of reconstructing training face number one for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

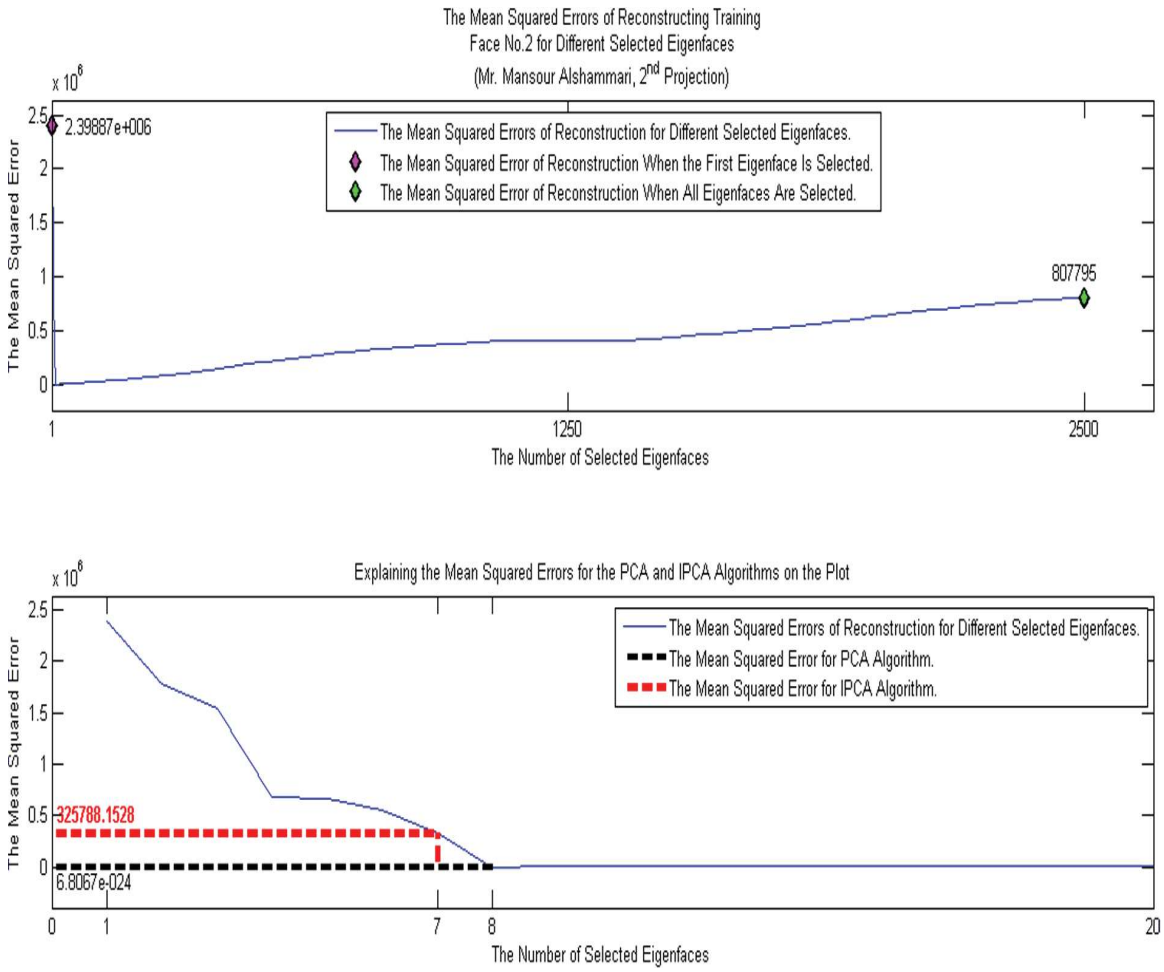

Figure I.2: The plot of the mean squared errors (MSEs) of reconstructing training face number two for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

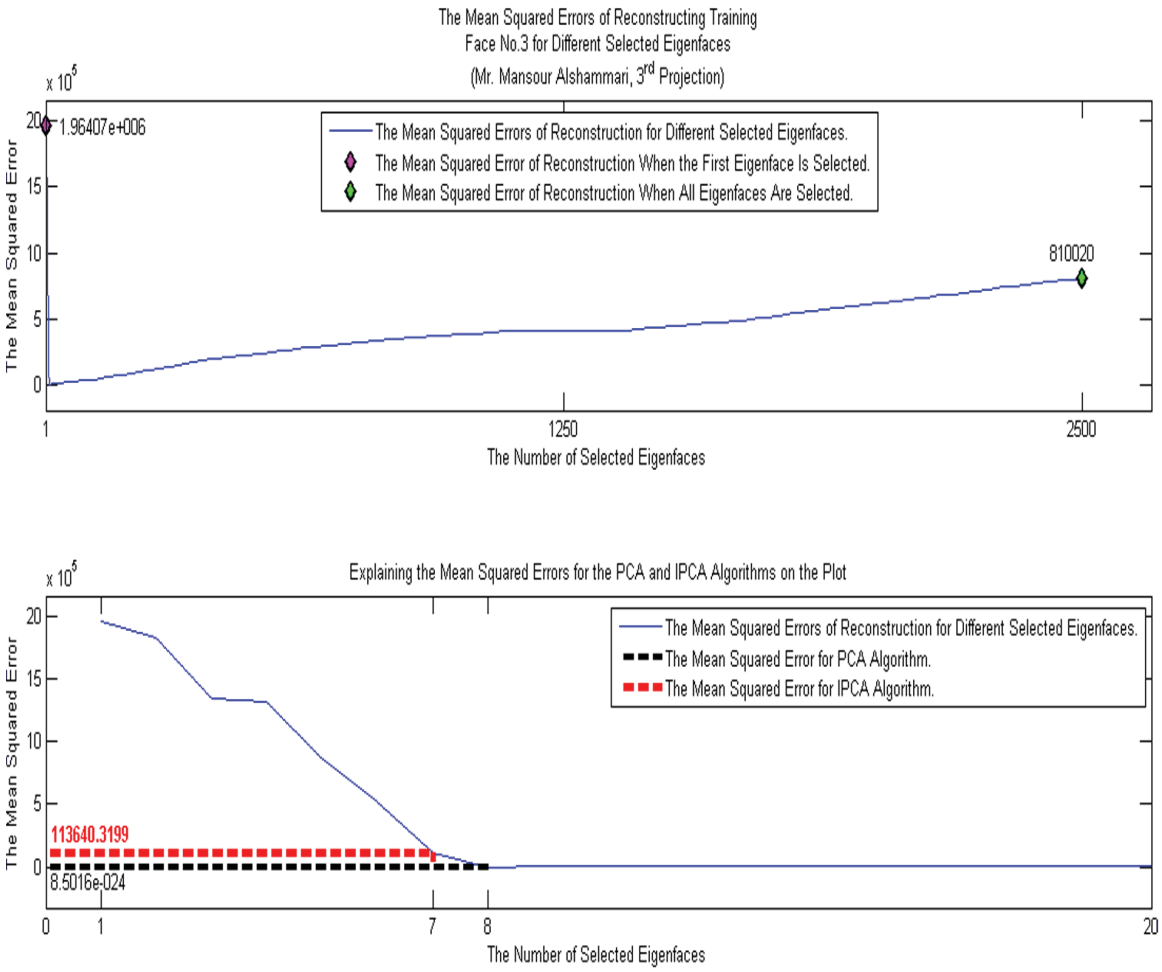

Figure I.3: The plot of the mean squared errors (MSEs) of reconstructing training face number three for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

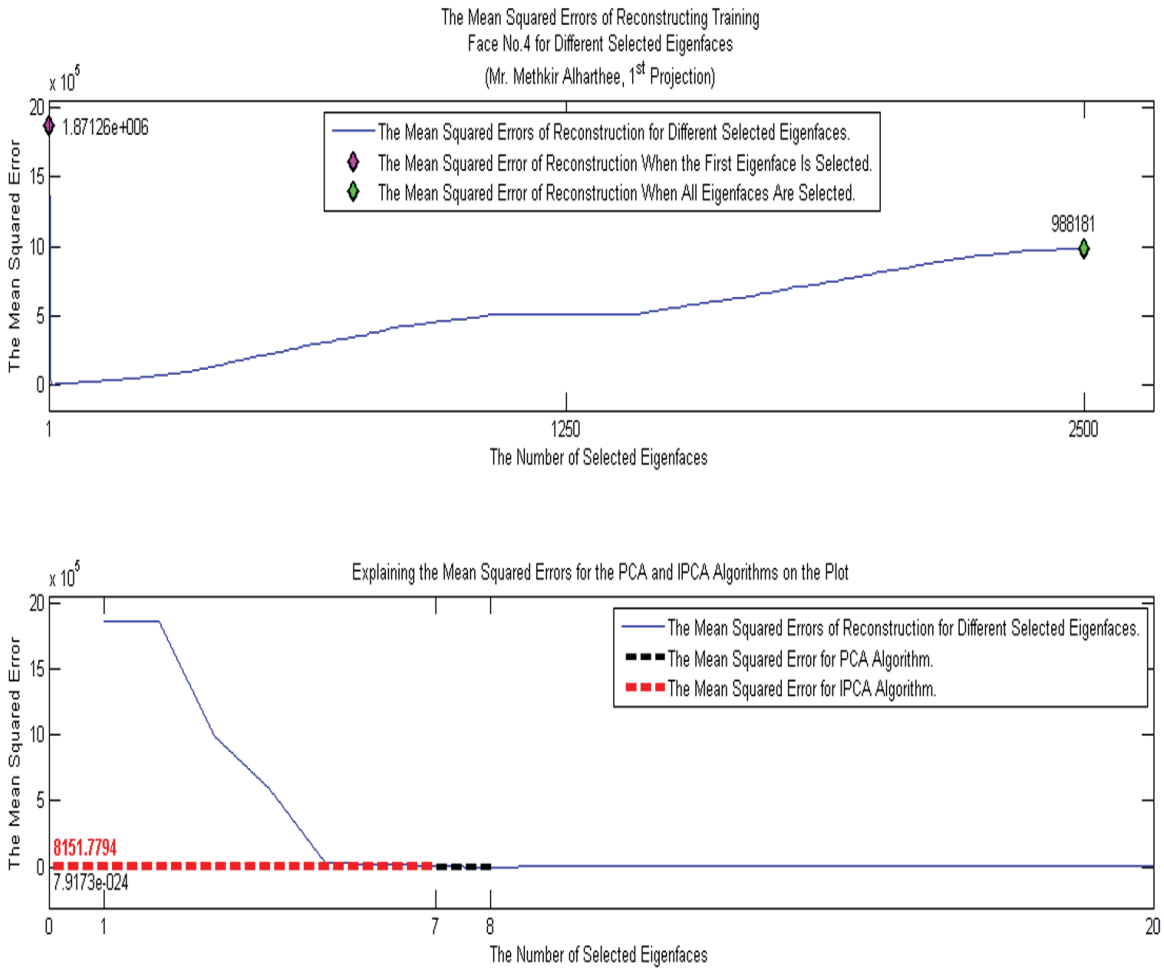

Figure I.4: The plot of the mean squared errors (MSEs) of reconstructing training face number four for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

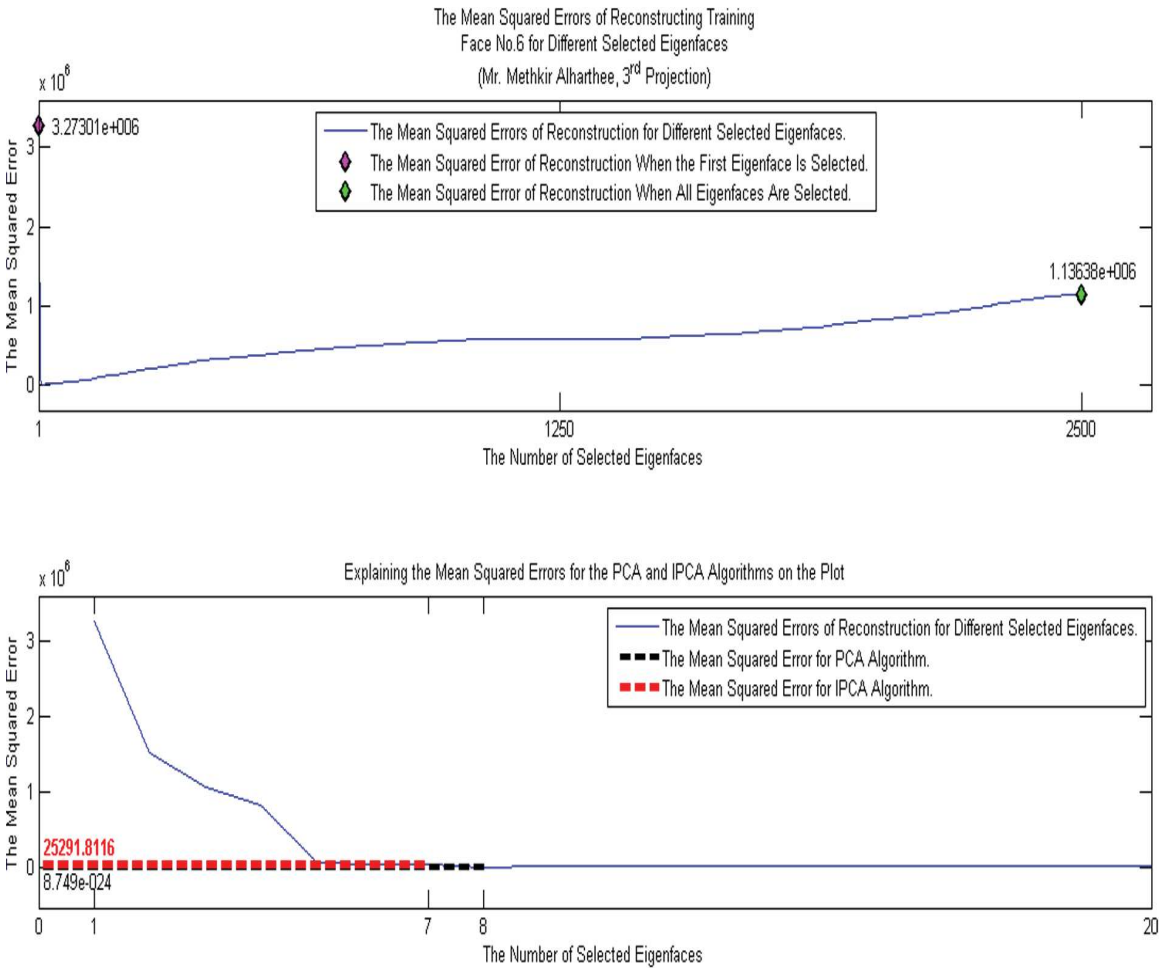

Figure I.5: The plot of the mean squared errors (MSEs) of reconstructing training face number six for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

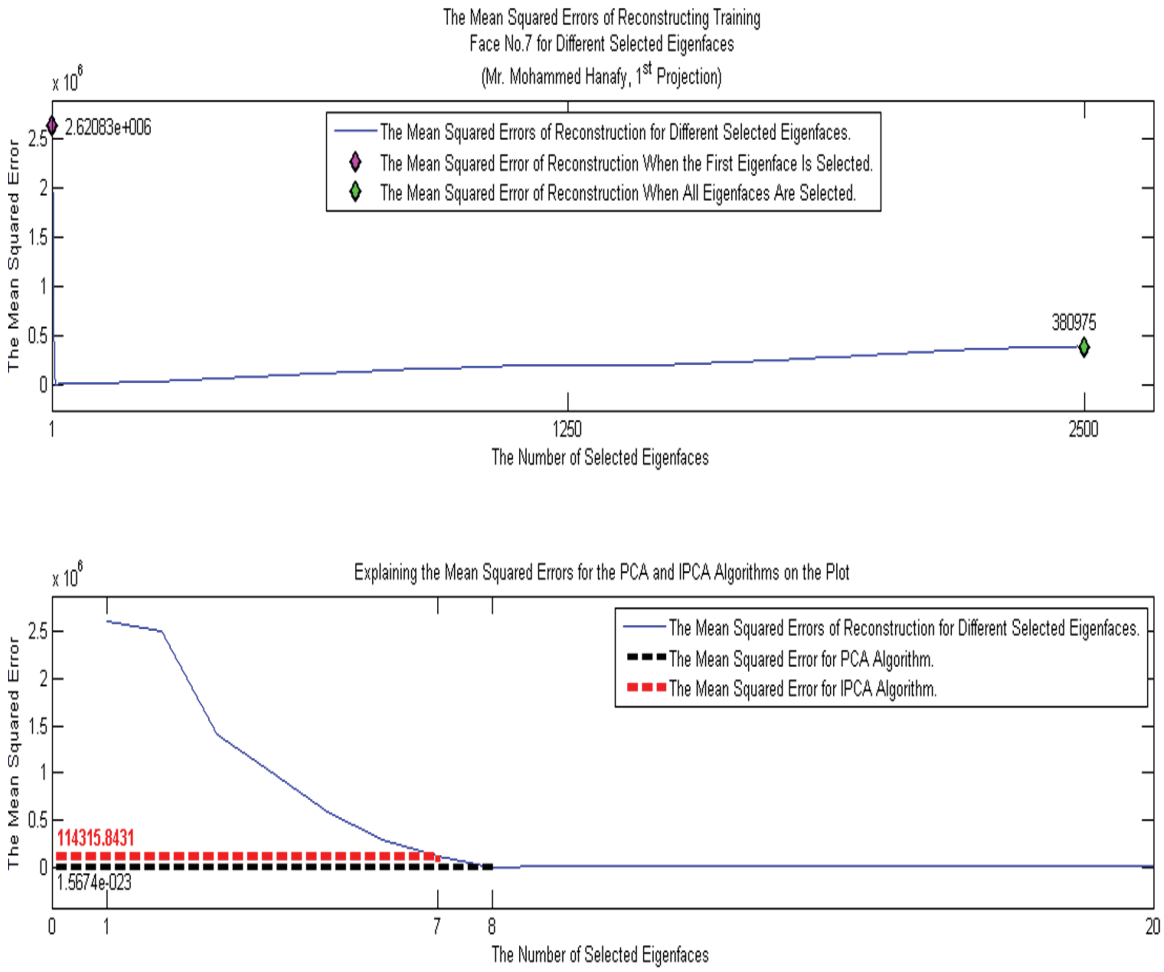

Figure I.6: The plot of the mean squared errors (MSEs) of reconstructing training face number seven for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

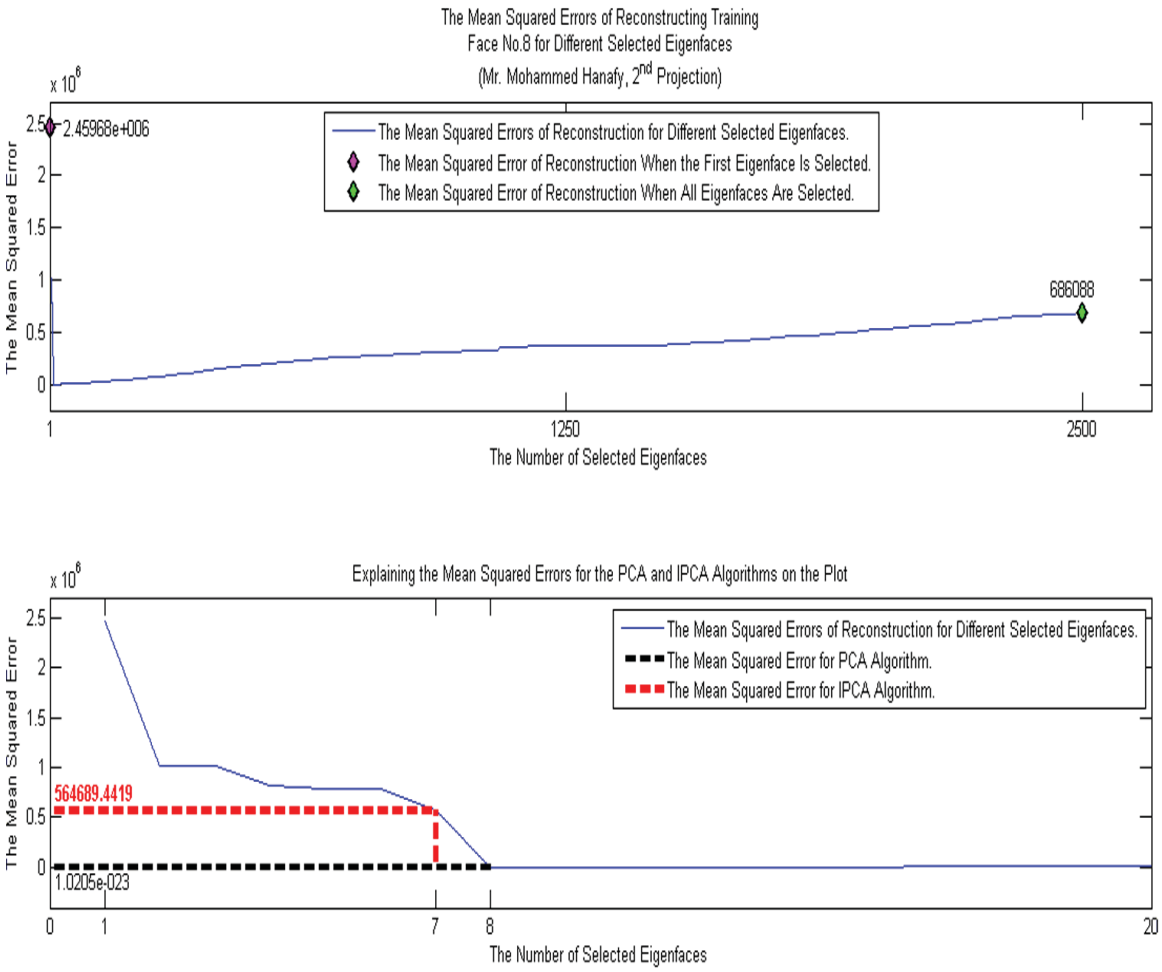

Figure I.7: The plot of the mean squared errors (MSEs) of reconstructing training face number eight for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

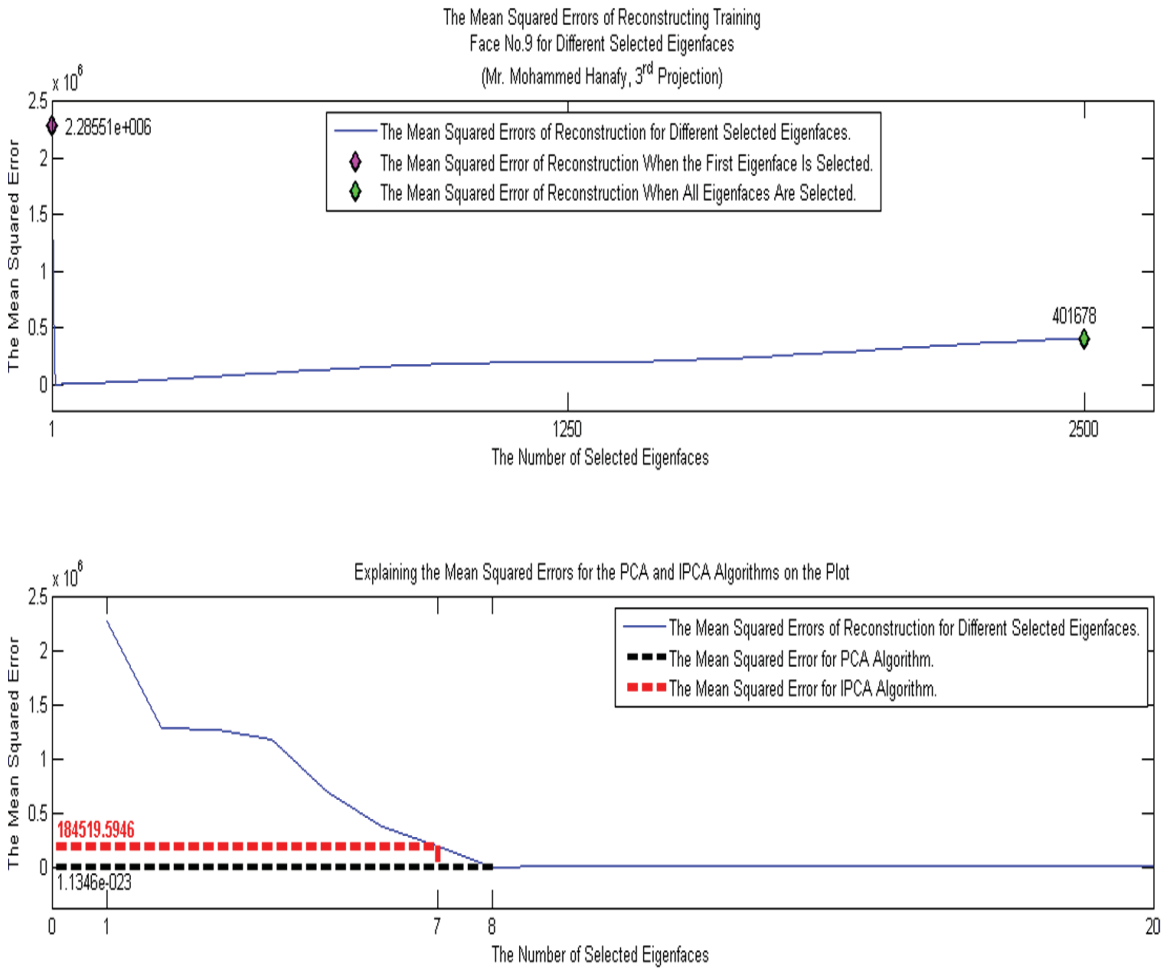

Figure I.8: The plot of the mean squared errors (MSEs) of reconstructing training face number nine for different selected eigenfaces compared with resulted mean squared errors when the PCA and IPCA algorithms are used.

Appendix J

A Code for Optimizing the Optical Model Database

This code is for optimizing the impulses database of the joint transform correlator (JTC) for object detection and face recognition.

```
1
2  % Optimizing the Impulses' Database of the Joint Transform
3  % Correlator (JTC) for Object Detection and Face Recognition.
4
5
6  clc
7  clear all
8  close all
9
10
11 Total_No_of_Impulses=3; % The total number of the impulses.
12
13
14 Objects_Folder=[cd ........
15     '\The Black Background Tested Objects']; % The folder of the
16                                              % black background
17                                              % tested objects.
18 % Objects_Folder=[cd .........
19 %     '\The White Background Tested Objects']; % The folder of the
20                                              % white background
21                                              % tested objects.
22 if isdir(Objects_Folder)==0
23     Error_Message=sprintf(.....
24         'Error: The following folder does not exist\n%s',.....
25         Objects_Folder);
26     warndlg(Error_Message);
```

```
    break
end

Objects=.......
    dir(fullfile(Objects_Folder,'*.jpg')); % Listing the folder of
                                           % the black or white
                                           % background objects.

Total_No_of_Objects=length(Objects); % The total number
                                     % of the objects.

Objs=[36 36 36]; % Each element in this vector represents
                 % the total number of the objects that
                 % are taken for each impulse.

L1=Objs(1); % L1=60 is the total number of the
            % objects for Mr. Mansour Alshammari.
L2=L1+Objs(2); % L2=120 is the total number of the
               % objects for Mr. Methkir Alharthee.
L3=L2+Objs(3); % L3=180 is the total number of the
               % objects for Mr. Mohammed Hanafy.


% Imhist for setting up a threshold to work on just the pixels of
% a face and throwing the background pixels. Imhist calculates the
% number of pixels in an object that have the same intensity
% levels. So, if an object has a unified background then the
% biggest histogram of the intensity levels will be for the
% backgroubd pixels because the total number of pixels that have
% the same intensity levels are the background pixels of the
% object. Note that, the histogram of a digital image is defined as
% the discrete function, h(rk)=nk, where rk is the kth intensity
% level and nk is the number of pixels in the image whose intensity
% level is rk.
hist_Objects=zeros(Total_No_of_Objects,256);
for A1=1:Total_No_of_Objects

    Object_Number=[num2str(A1) '.jpg'];
    Object_Location=fullfile(Objects_Folder,Object_Number);
    Object=double(rgb2gray(imread(Object_Location))); % The object.
    hist_Objects(A1,:)=.....
        imhist(uint8(Object)); % Note that, each object must be
                               % scaled between 0 to 255 before
                               % using imhist. For doing that,
                               % uint8 can be used for converting
                               % the object class form double to
                               % uint8.

%     plot(hist_Objects(A1,:))
%     if A1<=L1
```

```
77 %         title(['The Histogram of Object No.' num2str(A1) ....
78 %             ' for Mr. Mansour Alshammari'])
79 %     elseif A1>L1 && A1<=L2
80 %         title(['The Histogram of Object No.' num2str(A1) .....
81 %             ' for Mr. Methkir Alharthee'])
82 %     elseif A1>L2 && A1<=L3
83 %         title(['The Histogram of Object No.' num2str(A1) .....
84 %             ' for Mr. Mohammed Hanafy'])
85 %     end
86 %     xlabel('Intensity Level r_{k}')
87 %     ylabel({'The Number of Pixels in the Object Whose '....
88 %         'Intensity Level Is r_{k} Where h(r_{k})=n_{k}'})
89 %     axis tight
90 %
91 %     disp(['Please, press any keyboard button to explore '....
92 %         'the remaining histograms >>>>>>>'])
93 %     pause
94 %     clc
95 end
96
97 Mean_hist_Objects=sum(hist_Objects,1)/......
98     Total_No_of_Objects; % The average histogram
99                          % for all objects.
100 % plot(Mean_hist_Objects)
101 % title('The Mean Histogram of All Objects')
102 % xlabel('Intensity Level r_{k}')
103 % ylabel({'The Mean Number of Pixels from All Objects Whose';....
104 %     'Intensity Level Is r_{k}'})
105 % axis tight
106 % pause
107
108 % Setting up a threshold in order to work on just the pixels of the
109 % faces and blocking the pixels of the backgrounds.
110 Objects_Threshold=8; % The picked threshold is based on the average
111                      % histogram for all objects when the objects
112                      % have black backgrounds. Note that, all
113                      % intesity levels below the threshold
114                      % represent the objects backgrounds because
115                      % these levels have the biggest histogram.
116 % Objects_Threshold=180; % The Picked threshold is based on the
117 %                        % average histogram for all objects when
118 %                        % the objects have white backgrounds. Note
119 %                        % that, all intesity levels above the
120 %                        % threshold represent the objects
121 %                        % backgrounds because these levels have
122 %                        % the biggest histogram.
123
124
125 Max_Desired_Cross_Fields=zeros(Total_No_of_Objects,.....
126     Total_No_of_Impulses); % Each element of each row in this
```

```
127                             % matrix represents the maximum value
128                             % of the desired crosscorrelated field
129                             % between an object and one of the
130                             % impulses.
131
132
133 Recognition_Combinations=zeros(L1*(L2-L1)*(L3-L2),.....
134     Total_No_of_Impulses+1); % The first three columns of this
135                              % matrix contain different
136                              % combinations of impulses and the
137                              % third column represents the error
138                              % rates of recognition associated with
139                              % those combinations.
140
141
142 Detection_Combinations=zeros(L1*(L2-L1)*(L3-L2),......
143     Total_No_of_Impulses+1); % The first three columns of this
144                              % matrix contain different
145                              % combinations of impulses and the
146                              % third column represents the error
147                              % rates of detection associated with
148                              % those combinations.
149
150
151 i=0;
152
153
154 for Im1=1:L1
155     for Im2=L1+1:L2
156         for Im3=L2+1:L3
157
158             Impulses=[Im1 Im2 Im3]; % The selected impulses.
159
160             for n=1:Total_No_of_Objects
161
162                 % Normalizing all the objects for removing
163                 % lightening effects on them then increasing the
164                 % resolution of object detection and recognition.
165                 % Note that, the normalization will be done just
166                 % for the faces pixels for keeping variations among
167                 % the objects and impulses just in the faces
168                 % without the backgrounds effects.
169                 Object_Number=[num2str(n) '.jpg'];
170                 Object_Location=.....
171                     fullfile(Objects_Folder,Object_Number);
172                 Object1=double(rgb2gray(.....
173                     imread(Object_Location))); % The object.
174
175                 T=Object1>Objects_Threshold; % The pixels bigger
176                                              % than the threshold
```

```
177                                             % are of interest
178                                             % because they
179                                             % represent the pixels
180                                             % of a face.
181 %                 T=Object1<Objects_Threshold; % The pixels smaller
182 %                                               % than the threshold
183 %                                               % are of interest
184 %                                               % because they
185 %                                               % represent the
186 %                                               % pixels of a face.
187                 Object=zeros(size(T,1),size(T,2)); % The normalized
188                                                    % object.
189                 for R2=1:size(T,1)
190                     for C2=1:size(T,2)
191                         if T(R2,C2)==1
192                             Object(R2,C2)=.....
193                                 ceil(255*(Object1(R2,C2)/max(....
194                                 max(Object1)))); % The
195                                                  % normalization
196                                                  % of the object.
197                                                  % This is done to
198                                                  % increase the
199                                                  % dynamic range of
200                                                  % the object for
201                                                  % visualization by
202                                                  % scaling the
203                                                  % intensities from
204                                                  % 0 to 255.
205                         end
206                     end
207                 end
208                 [r c]=size(Object);
209
210                 for m=1:Total_No_of_Impulses
211
212                     % Normalizing all the impulses for removing
213                     % lightening effects on them then increasing
214                     % the resolution of object detection and
215                     % recognition. Note that, the normalization
216                     % will be done just for the faces pixels for
217                     % keeping variations among the objects and
218                     % impulses just in the faces without the
219                     % backgrounds effects.
220                     Impulse_Number=[num2str(Impulses(m)) '.jpg'];
221                     Impulse_Location=......
222                         fullfile(Objects_Folder,Impulse_Number);
223                     Impulse1=double(rgb2gray(.....
224                         imread(Impulse_Location))); % The impulse
225                                                     % response.
226
```

```
227                     T1=Impulse1>......
228                         Objects_Threshold; % The pixels bigger than
229                                            % the threshold are of
230                                            % interest because they
231                                            % represent the pixels
232                                            % of a face.
233 %                     T1=Impulse1<.....
234 %                         Objects_Threshold; % The pixels smaller
235 %                                            % than the threshold
236 %                                            % are of interest
237 %                                            % because they
238 %                                            % represent the
239 %                                            % pixels of a face.
240                     Impulse=zeros(size(T1,1),....
241                         size(T1,2)); % The normalized
242                                      % impulse response.
243                     for R1=1:size(T1,1)
244                         for C1=1:size(T1,2)
245                             if T1(R1,C1)==1
246                                 Impulse(R1,C1)=.....
247                                     ceil(255*(Impulse1(R1,C1)/....
248                                     max(max(Impulse1))))....
249                                     ; % The normalization of the
250                                       % impulse response. This is
251                                       % done to increase the
252                                       % dynamic range of the
253                                       % impulse for visualization
254                                       % by scaling the intensities
255                                       % from 0 to 255.
256                             end
257                         end
258                     end
259                     [p q]=size(Impulse);
260
261
262                     % Equalizing the width of the impulse response
263                     % with the width of the object in order to
264                     % collimate them on the input transparencies.
265                     if q>c
266                         if mod(q-c,2)==0
267                             Object=[zeros(r,(q-c)/2) Object .....
268                                 zeros(r,(q-c)/2)];
269                         elseif mod(q-c,2)==1
270                             Object=[zeros(r,floor((q-c)/2)) .....
271                                 Object zeros(r,floor((q-c)/2)+1)];
272                         end
273                     elseif q<c
274                         if mod(c-q,2)==0
275                             Impulse=[zeros(p,(c-q)/2) Impulse ....
276                                 zeros(p,(c-q)/2)];
```

```
277                         elseif mod(c-q,2)==1
278                             Impulse=[zeros(p,floor((c-q)/2)) .....
279                                 Impulse zeros(p,floor((c-q)/2)+1)];
280                         end
281                     end
282
283                     % Collimating the impulse response and the
284                     % object on the input transparencies. Note
285                     % that, the separation between the impulse
286                     % response and object must be bigger than
287                     % max{Wh,Wg}+Wg/2+Wh/2 in order to make the
288                     % output crosscorrelations of the impulse
289                     % response and object are completely separated
290                     % without any overlapping.
291                     Wh=size(Impulse,1); % The width of the impulse
292                                         % response in the direction
293                                         % of the y1-coordinate.
294                     Wg=size(Object,1); % The width of the object in
295                                        % the direction of the
296                                        % y1-coordinate.
297                     Y=max(Wh,Wg)+((Wh+Wg)/2); % The separation
298                                               % between the centers
299                                               % of the impulse
300                                               % response and
301                                               % object.
302                     dis=10; % This distance in order to make the
303                             % separation between the centers of the
304                             % impulse response and object bigger
305                             % than max{Wh,Wg}+Wg/2+Wh/2.
306                     Impulse_Object=[.............
307                         zeros(ceil(max(Wh,Wg)/2+Wh/4+3/4*Wg)+.....
308                         dis,size(Impulse,2));Impulse;....
309                         zeros(max(Wh,Wg)+dis,size(Impulse,2));....
310                         Object;zeros(ceil(max(Wh,Wg)/2+.....
311                         3/4*Wh+Wg/4)+dis,size(Object,2))];
312
313                     % Equalizing the dimensions of the input plane
314                     % P1 by padding it with zeros.
315                     [R C]=size(Impulse_Object);
316                     if R>C
317                         if mod(R-C,2)==0
318                             Impulse_Object=[.....
319                                 zeros(size(Impulse_Object,1),....
320                                 (R-C)/2) Impulse_Object ....
321                                 zeros(size(Impulse_Object,1),....
322                                 (R-C)/2)];
323                         elseif mod(R-C,2)==1
324                             Impulse_Object=[....
325                                 zeros(size(Impulse_Object,1),....
326                                 floor((R-C)/2)) Impulse_Object ....
```

```
                zeros(size(Impulse_Object,1),....
                floor((R-C)/2)+1)];
        end
    elseif R<C
        if mod(C-R,2)==0
            Impulse_Object=[zeros((C-R)/2,.....
                size(Impulse_Object,2));....
                Impulse_Object;zeros((C-R)/2,....
                size(Impulse_Object,2))];
        elseif mod(C-R,2)==1
            Impulse_Object=[.....
                zeros(floor((C-R)/2),....
                size(Impulse_Object,2));....
                Impulse_Object;.....
                zeros(floor((C-R)/2)+1,.....
                size(Impulse_Object,2))];
        end
    end


    U1=Impulse_Object; % The transmitted field from
                       % the input plane P1.
    [M N]=size(U1);

    L=10; % The physical side length of the array
          % which holds the input plane P1 in
          % meters (m).
    dx1_Input=L/N; % The sample spacing in the
                   % input plane array in the
                   % direction of the spatial space
                   % coordinate x1 in meters (m).
    dy1_Input=L/M; % The sample spacing in the
                   % input plane array in the
                   % direction of the spatial space
                   % coordinate y1 in meters (m).
    x1_Axis_Input=-floor(N/2)*dx1_Input:....
        dx1_Input:ceil(N/2)*dx1_Input-.....
        dx1_Input; % Sampling the input plane P1 in
                   % the direction of the spatial
                   % space coordinate x1.
    y1_Axis_Input=-floor(M/2)*dy1_Input:....
        dy1_Input:ceil(M/2)*dy1_Input-....
        dy1_Input; % Sampling the input plane P1 in
                   % the direction of the spatial
                   % space coordinate y1.


    U2=fftshift(fft2(fftshift(....
        U1))); % The Fourier transform of the
               % transmitted field in the back
```

```
377                                % focal plane of the lens L2.
378                 I=(abs(U2)).^2; % The intensity of the Fourier
379                                 % transformed field in the
380                                 % plane P2.
381
382                 lambda=550e-9; % The wavelength in meters (m).
383                 f=0.055; % The focal length in meters (m).
384                 dx2=(lambda*f)/....
385                     (N*dx1_Input); % The sample spacing in the
386                                    % plane P2 in the direction
387                                    % of the spatial space
388                                    % coordinate x2 in meters
389                                    % (m).
390                 dy2=(lambda*f)/.....
391                     (M*dy1_Input); % The sample spacing in the
392                                    % plane P2 in the direction
393                                    % of the spatial space
394                                    % coordinate y2 in meters
395                                    % (m).
396                 x2_Axis=-floor(N/2)*dx2:dx2:ceil(N/2)*dx2-....
397                     dx2; % Sampling the plane P2 in the
398                          % direction of the spatial space
399                          % coordinate x2.
400                 y2_Axis=-floor(M/2)*dy2:dy2:ceil(M/2)*dy2-....
401                     dy2; % Sampling the plane P2 in the
402                          % direction of the spatial space
403                          % coordinate y2.
404
405
406                 U3=ifftshift(ifft2(ifftshift(.....
407                     I))); % The crosscorrelated field in the
408                           % back focal plane of the lens L4.
409
410                 dx3=(lambda*f)/(N*dx2); % The sample spacing in
411                                         % the plane P3 in the
412                                         % direction of the
413                                         % spatial space
414                                         % coordinate x3 in
415                                         % meters (m).
416                 dy3=(lambda*f)/(M*dy2); % The sample spacing in
417                                         % the plane P3 in the
418                                         % direction of the
419                                         % spatial space
420                                         % coordinate y3 in
421                                         % meters (m).
422                 x3_Axis=-floor(N/2)*dx3:dx3:ceil(N/2)*dx3-.....
423                     dx3; % Sampling the plane P3 in the
424                          % direction of the spatial space
425                          % coordinate x3.
426                 y3_Axis=-floor(M/2)*dy3:dy3:ceil(M/2)*dy3-....
```

```
                        dy3; % Sampling the plane P3 in the
                             % direction of the spatial space
                             % coordinate y3.


                    % Synthesizing a desired filtering mask then
                    % filtering the crosscorrelated field in the
                    % plane P3.
                    Cen=floor(M/2)+1; % The center of the filtering
                                      % mask.
                    Cen1=Cen-(Y+dis); % The center of the desired
                                      % crosscorrelated field.
                    Wh1=q; % The width of the impulse response in
                           % the direction of the x1-coordinate.
                    Wg1=c; % The width of the object in the
                           % direction of the x1-coordinate.

                    Mask=zeros(M,N);
                    for P=Cen1-floor((Wh+Wg)/2):Cen1+.....
                            ceil((Wh+Wg)/2)
                        for Q=Cen-floor((Wh1+Wg1)/2):Cen+.....
                                ceil((Wh1+Wg1)/2)
                            Mask(P,Q)=1;
                        end
                    end

                    Cross_Field=Mask.*U3; % The filtered
                                          % crosscorrelated field
                                          % in the plane P3.


                    % For simplicity, instead of processing the
                    % entire image of the filtered crosscorrelated
                    % field, we select only the crosscorrelated
                    % field of interest.
                    P=Cen1-floor((Wh+Wg)/2):Cen1+ceil((Wh+Wg)/2);
                    Q=Cen-floor((Wh1+Wg1)/2):Cen+ceil((Wh1+Wg1)/2);
                    Desired_Cross_Field=U3(P,Q);


                    Max_Desired_Cross_Fields(n,m)=.....
                        max(max(Desired_Cross_Field));

                end
            end


            % Testing the process of object recognition.
            % Note that, the objects here are known for us in
            % order to use them for testing the object recognition
```

```
477             % process as well as setting up a recognition
478             % threshold.
479
480             % The elements in each of the following vectors
481             % represent the maximum values of the desired
482             % crosscorrelated fields between the objects and their
483             % corresponding impulse. Note that, each impulse and
484             % its corresponding objects have a same face so the
485             % vectors will include the biggest crosscorrelations.
486             V1=(Max_Desired_Cross_Fields(1:L1,1)).';
487             V2=(Max_Desired_Cross_Fields(L1+1:L2,2)).';
488             V3=(Max_Desired_Cross_Fields(L2+1:L3,3)).';
489
490             % The calculation of the mean and the standard
491             % deviation for each vector of the biggest
492             % crosscorrelations and stacking them in a vector for
493             % the means and another for the standard deviations.
494             % This is done for setting up a threshold for object
495             % recognition.
496             V_Mean=[mean(V1);mean(V2);mean(V3)];
497             V_STD=[std(V1);std(V2);std(V3)];
498
499             Failures_Vector=zeros(1,.....
500                 Total_No_of_Objects); % A vector for counting the
501                                       % number of failures in the
502                                       % object recognition process.
503
504             for w=1:Total_No_of_Objects
505                 % The object recognition process.
506                 for ii=1:Total_No_of_Impulses
507                     if  max(Max_Desired_Cross_Fields(w,:))==....
508                             Max_Desired_Cross_Fields(w,ii) && ....
509                             max(Max_Desired_Cross_Fields(w,:))>=...
510                             (V_Mean(ii,1)-V_STD(ii,1)) && ....
511                             max(Max_Desired_Cross_Fields(w,:))<=...
512                             (V_Mean(ii,1)+V_STD(ii,1))
513                         z={'Mr. Mansour Alshammari';....
514                             'Mr. Methkir Alharthee';.....
515                             'Mr. Mohammed Hanafy'};
516                         Recognized_As=char(z(ii,1));
517                         break
518                     else
519                         Recognized_As='Unknown Object';
520                     end
521                 end
522
523                 % Defining the object.
524                 if w<=L1
525                     name='Mr. Mansour Alshammari';
526                 elseif L1<w && w<=L2
```

```
527                     name='Mr. Methkir Alharthee';
528                 else name='Mr. Mohammed Hanafy';
529                 end
530
531                 Str1=strcmp(Recognized_As,....
532                     'Mr. Mansour Alshammari');
533                 Str2=strcmp(Recognized_As,'Mr. Methkir Alharthee');
534                 Str3=strcmp(Recognized_As,'Mr. Mohammed Hanafy');
535                 F='Success';
536                 if w<=L1 && Str1==0;
537                     F='Failure';
538                     Failures_Vector(1,w)=1;
539                 elseif w>L1 && w<=L2 && Str2==0;
540                     F='Failure';
541                     Failures_Vector(1,w)=1;
542                 elseif w>L2 && w<=L3 && Str3==0;
543                     F='Failure';
544                     Failures_Vector(1,w)=1;
545                 end
546             end
547
548             Total_Number_of_Failures=.....
549                 sum(Failures_Vector); % The total number of
550                                       % failures in the object
551                                       % recognition process.
552
553             Recognition_Error_Rate=(Total_Number_of_Failures/.....
554                 Total_No_of_Objects)*100; % The error rate
555                                           % of recognition.
556
557
558             % Testing the process of object detection.
559             % Note that, the objects here are known for us in order
560             % to use them for testing the object detection process
561             % as well as setting up a detection threshold.
562
563             % The calculation of the mean and standard deviation
564             % for the crosscorrelations between the objects and
565             % each impulse response. This is done for setting up a
566             % threshold for object detection.
567             Vectorization=reshape(Max_Desired_Cross_Fields,1,.....
568                 size(Max_Desired_Cross_Fields,1)*.....
569                 size(Max_Desired_Cross_Fields,2));
570             Mean=mean(Vectorization);
571             STD=std(Vectorization);
572
573             Failures_Vector1=zeros(1,.....
574                 Total_No_of_Objects); % A vector for counting the
575                                       % number of failures in the
576                                       % object detection process.
```

```
577
578             for w=1:Total_No_of_Objects
579                 % The object detection process.
580                 if  max(Max_Desired_Cross_Fields(w,:))>=.....
581                         (Mean-STD) && .....
582                        max(Max_Desired_Cross_Fields(w,:))<=....
583                         (Mean+STD)
584                     Detected_As='a face';
585                 else
586                     Detected_As='not a face';
587                 end
588
589                 b='a face'; % The object originally is a face.
590
591                 Str=strcmp(Detected_As,'a face');
592                 F='Success';
593                 if w<=L3 && Str==0;
594                     F='Failure';
595                     Failures_Vector1(1,w)=1;
596                 end
597             end
598
599             Total_Number_of_Failures1=....
600                 sum(Failures_Vector1); % The total number of
601                                        % failures in the object
602                                        % detection process.
603
604             Detection_Error_Rate=(Total_Number_of_Failures1/.....
605                 Total_No_of_Objects)*100; % The error rate
606                                           % of detection.
607
608
609             Recognition_Combinations((Im3-L2)+i,:)=.....
610                 [Impulses Recognition_Error_Rate];
611
612
613             Detection_Combinations((Im3-L2)+i,:)=.....
614                 [Impulses Detection_Error_Rate];
615
616         end
617         i=i+(Im3-L2);
618     end
619     Im1
620 end
621
622
623 save('Impulses Combinations for Recognition',....
624     'Recognition_Combinations')
625 save('Impulses Combinations for Detection',....
626     'Detection_Combinations')
```

```
627
628
629 % Finding the optimal combination of impulses which obtain the
630 % lowest error rate of recognition.
631 Optimal_Combination_Recognition=0; % The optimal combination of
632                                    % impulses for recognition.
633 Optimal_Error_Rate_Recognition=0; % The lowest error rate of
634                                   % recognition that is produced
635                                   % when the optimal combination
636                                   % of impulses for recognition
637                                   % is used.
638 for j=1:size(Recognition_Combinations,1)
639     if Recognition_Combinations(j,Total_No_of_Impulses+1)==.....
640             min(Recognition_Combinations(:,Total_No_of_Impulses+1))
641         Optimal_Combination_Recognition=.......
642             Recognition_Combinations(j,1:Total_No_of_Impulses);
643         Optimal_Error_Rate_Recognition=.....
644             Recognition_Combinations(j,Total_No_of_Impulses+1);
645     end
646 end
647
648
649 % Finding the optimal combination of impulses which obtain the
650 % lowest error rate of detection.
651 Optimal_Combination_Detection=0; % The optimal combination of
652                                  % impulses for detection.
653 Optimal_Error_Rate_Detection=0; % The lowest error rate of
654                                 % detection that is produced when
655                                 % the optimal combination of
656                                 % impulses for detection is used.
657 for j=1:size(Detection_Combinations,1)
658     if Detection_Combinations(j,Total_No_of_Impulses+1)==.....
659             min(Detection_Combinations(:,Total_No_of_Impulses+1))
660         Optimal_Combination_Detection=.....
661             Detection_Combinations(j,1:Total_No_of_Impulses);
662         Optimal_Error_Rate_Detection=.....
663             Detection_Combinations(j,Total_No_of_Impulses+1);
664     end
665 end
```

Appendix K

Authorizations for Using People Photos

Figure K.1, Figure K.2 and Figure K.3 show respectively the authorizations for using the photos of Mr. Mansour Thuwaini Al-Shammari, Mr. Mathkar Alawi Alharthi and Mr. Mohamed Elsayed Hanafy.

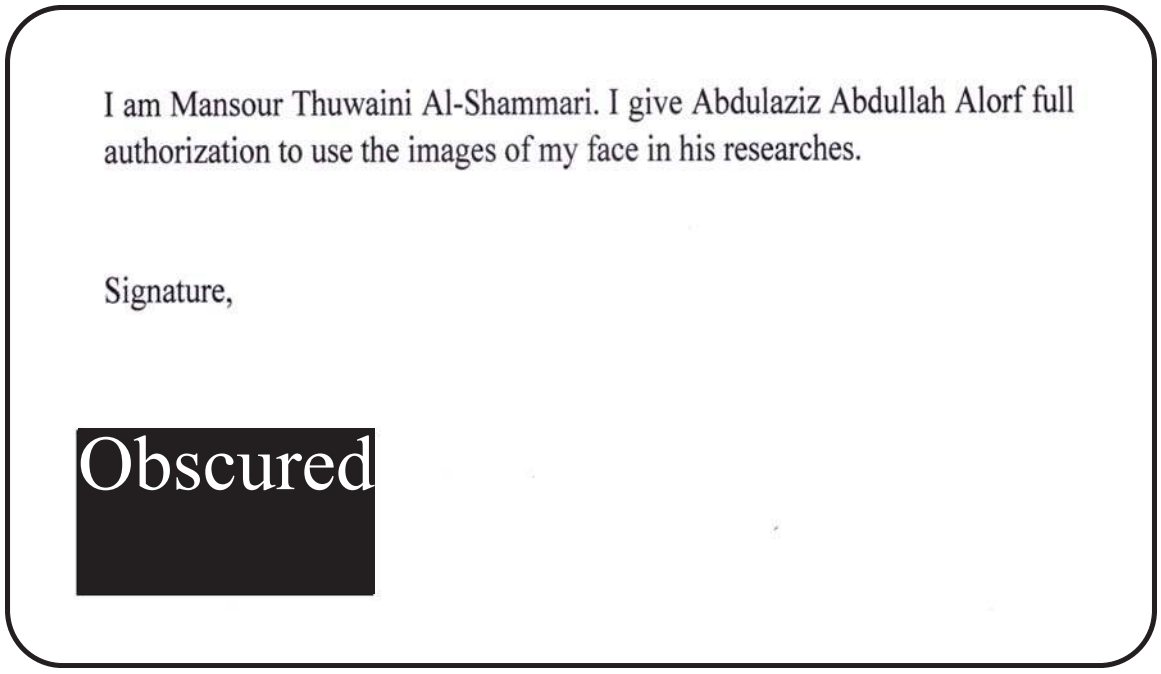

Figure K.1: The authorization for using the photos of Mr. Mansour Thuwaini Al-Shammari.

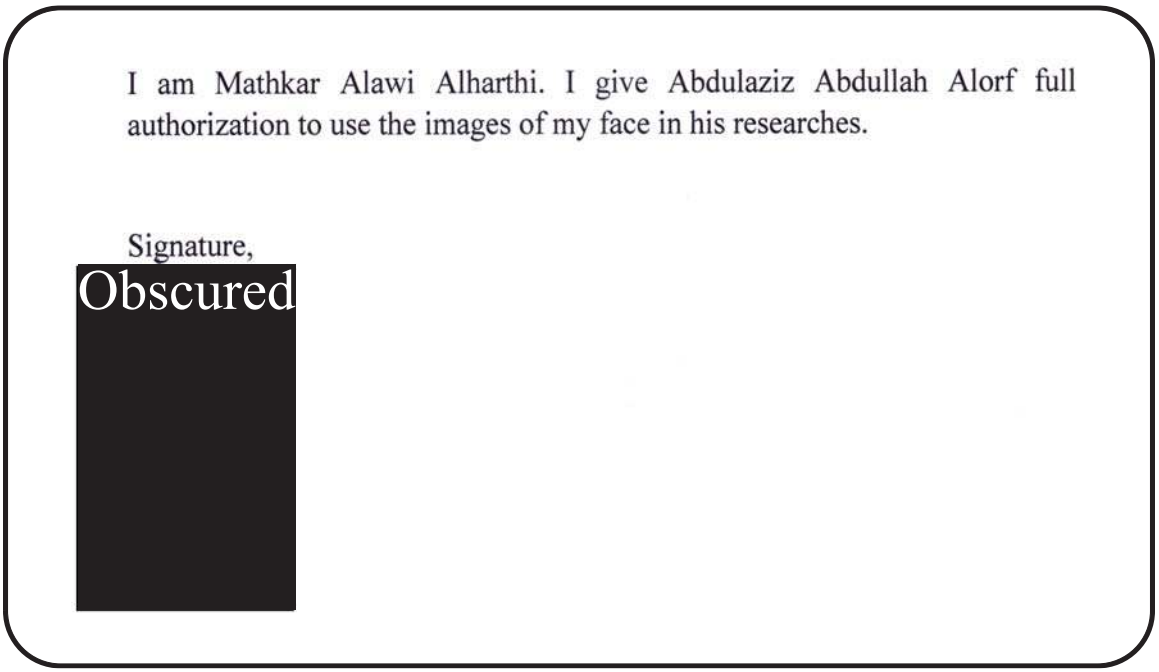

Figure K.2: The authorization for using the photos of Mr. Mathkar Alawi Alharthi.

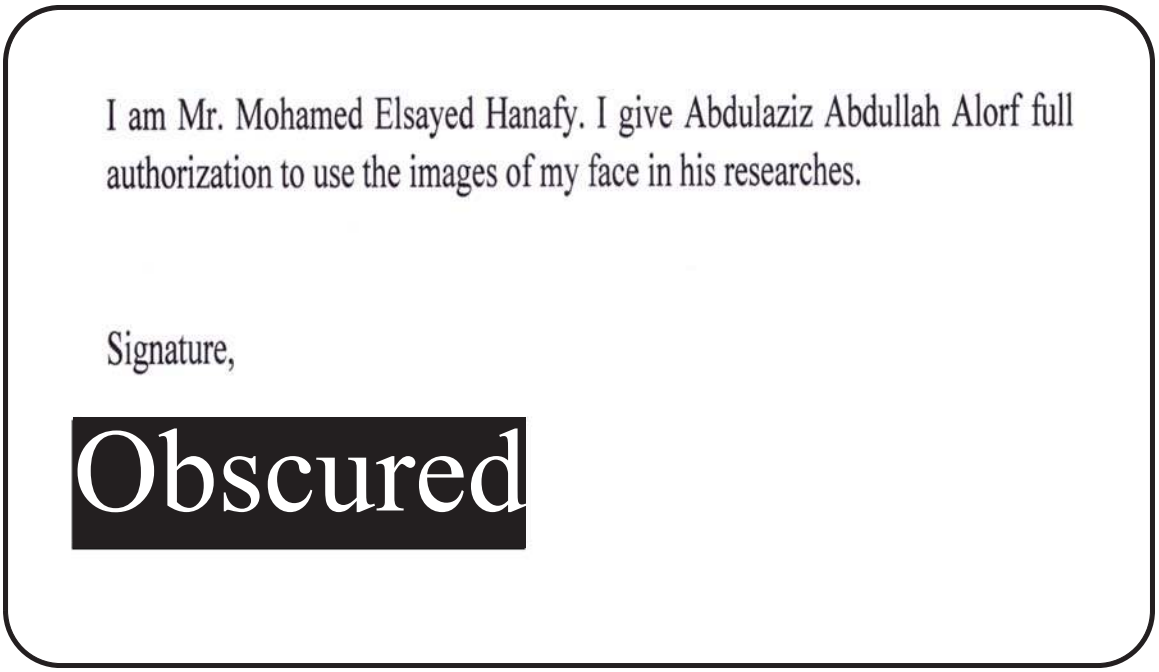

Figure K.3: The authorization for using the photos of Mr. Mohamed Elsayed Hanafy.