Face Recognition using Localized Facial Features

Fatma Şıker

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
June 2014
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Cem Ergün
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Hasan Demirel

2. Asst. Prof. Dr. Adnan Acan

3. Asst. Prof. Dr. Cem Ergün

ABSTRACT

Face is a complex multi-dimensional structure and needs good computing biometric techniques for recognition. The aim of this study is to understand the role of each localized facial feature component in face recognition system and treat it as a one-dimensional recognition problem. In this context, face recognition is performed by using Principal Component Analysis (PCA) method. Face images are stored in a face database that encodes best variation among face images. Instead of recognizing human characteristic from full face data, identifying the facial feature components seperately might be alternative classification method to get successful recognition performance face is defined by eigenface which are eigenvectors of the set of face components. Each face feature is extracted by using automatic/manual segmentation techniques to have facial features such as left eyes, right eyes, nostrils and mouth. Finally, each segmented facial feature can be one classifier and combination of each may help to form multi-classifier problem to achieve improved recognition results. Proposed face recognizion system, which is using localized facial features along with global face, improves PCA-based face system by average of 4.5 %.

Keywords: Face Recognition, Principal Component Analysis, Segmentation Techniques, Multi-Classifier Problem.

ÖZ

Yüz karmaşık çok boyutlu bir yapı olarak tanınmasından dolayı iyi bir biyometrik hesaplama tekniğine ihtiyacı vardır. Yüz tanıma sisteminde amacı belirlemek için her bir bileşenin rolünü anlamak ve tek boyutlu tanıma sorunu olarak tanımlamak gerekir. Bu şekilde yüz tanıma sistemi Temel Bileşen Analizi (PCA) yöntemi kullanılarak yapılır. Yüz görüntüleri arasındaki en iyi varyasyon kodlar bir yüz veritabanı üzerine tahmin edilerek yüz görüntüleri ile bulunur. Buna bağlı olarak başarılı tanıma test sonuçları elde etmek için alternatif sınıflandırma yönteminin belirlenmesi ve yüz bileşenlerinin tanıma özelliği yerine yüzün bütün verileri ve insanın karakterleri tanımlanır. Yüzün her bir öznitelik parçası olan sol göz, sağ göz, burun delikleri ve ağız gibi yan yüz özellikleri kullanılarak bazı bölütleme teknikleri ile ayıklanması özvektörler olan özyüz ile tanımlanır. Yani, bu özyüz yaklaşım görüntülerinin tanınması için Temel Bileşen Analizi (PCA) yöntemi kullanılarak net bir görünüm elde edilir. En sonda başarılı tanıma sonuçları elde etmek için çoklu sınıflandırıcı kullanılarak her kombinasyon ve sınıflandırma ile yüz özelliği modellenir.

Anahtar Kelimeler: Yüz Tanıma, Temel Bileşenler Analizi, Çoklu Sınıflandırıcı sorunu, Bölütleme teknikleri.

To my beloved father…

ACKNOWLEDGEMENTS

First of all, My sincere thanks goes to Assoc. Prof. Dr. Hasan Demirel for his assistance, direction and guidance.In particular, his recommendations and suggestions have been valuable for the thesis, since he helped me understand the topic about which I had little knowledge about.

I also wish to thank Asst. Prof. Dr. Cem Ergün for his valuable comments and preparation in every step of this thesis.

My sincere thanks go to my parents for their endless love and encouragement. I would like to dedicate this study to them as an indication of their significance in this study as well as in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVATIONS

PCA Principal Component Analysis

ORL Olivetti Research Laboratory Database

C Covariance Matrix

K-NN k - Nearest Neighbour

TS Test Set

Max Maximum Value

A Matrix

Ψ Image Set

Γ_i Image Vector

U_m Eigenfaces

λ_k Eigenvalues

R Classifiers Vector

ω_k Class

Chapter 1

INTRODUCTION

One of the common topics in pattern recognition and image processing is face recognition. It identifies the person in a digital image by analysing and comparing patterns.

Face recognition has become an important part in many applications, such as security models, credit card verification, and criminal identification. For example, the ability to model a particular face and retrieve it from a large number of stored face models makes it possible to improve criminal identification [1]. We know that the human brain encodes and decodes faces.

A face recognition system mostly focuses on identifying an individual from a set of images. The problem can be solved by providing an image of a human face to classifiers which compares the features with stored image models to obtain a match. Modifying face recognition algorithms to work on extracted facial features improves their performance.

Features extracted from a face are processed and matched with similarly processed faces that are present in the database. The main idea of feature extraction is to have features or discriminants that show the face attributes in an effective way.

Usually misclassifications are due to how well these features are extracted. Principal Component Analysis (PCA) is one of the algorithms used to get successful result on frontal face recognition while eigenvectors are defined to be the independent vectors retrived from a subset of the face images [2].

1.1 Aim and Motivation

Face recognition can be used in different application areas to solve issues about image, film processing, human-computer interaction and criminal identification [3]. Typical face recognition system, without dimentionality reduction, is not effective in representing dimensionality of face spaces. The face images provide high dimensional data which span very low dimensional space. If we compare face spaces of lower dimension, the result would be more efficient than the higher dimensional subspaces. Nowadays, one of the most important ideas is to apply a system on a large number of different stored faces with real-time variations [4].

In this thesis, we are using Principal Component Analysis (PCA) algorithm with eigenfaces of an image in face recognition. This algorithm uses eigenfaces for dimensionality reduction to more efficiently perform the recognition of the images. PCA algorithm helps to increase the efficiency of the lower dimensional spaces and can also be used for recognizing the gender of a person by defining their facial expressions.

PCA is basically used for dimensionality reduction. Dimensionality reduction is an important task in machine learning which facilitates classification, compression, and visualization of high-dimensional data. It is important to reduce the number of dimensions under consideration as they increase the computation time slowly and we might also have irrelevant features in the data collected. Dimensionality reduction

helps reduce the curse of dimensionality and cuts the computational cost. These preprocessing steps can also help in reducing the noise, making the data easier to work with as they create uncorrelated features of the data. Reducing the number of dimensions in the dataset helps in deeper understanding of the problem.

Eigenfaces represent an image from use an image to derive similar images from database, but the image is rejected while trying to achieve recognition [2]. Due to the changes in the position of the eye and mouth according to facial size and pose it is not advisable to use normalization and edge improvements in order to keep the position of the eyes and the size of the face fixed [3].

Features cropped from a face are processed and compared with processed faces present in the database. We start with a group of original facial features to calculate vector system for image position. After that we can apply the Eigenfaces to face recognition problem. These projection vectors of facial features in the training set are calculated. The method utilizes projection of the face space by formed eigenfaces. The compression of the Euclidian distance vector is made by eigenfaces of the facial features of images under the face image properties.

Using PCA to each of the facial feature components such as eyes, noses and mouths and obtaining dedicated classifications in the respective domains as well as having a PCA based global classifier is the first step of the proposed scheme. The classification results coming from each classifier are combined through data fusion to improve the overall performance of the PCA based face recognition system. The average improvement reaches to %4.5 with the proposed localized feature based multi-classifier system.

1.2 Outline of Thesis

This thesis is organized as follows, chapter two consist a brief introduction on how to detect similarities between stored face images in the database and extracted features of a face image. By using PCA algorithm system and various feature extraction to shows the performance value of majority, sum and product rule.

In chapter 3, database properties are defined by using facial features extraction and eigenfeatures with grayscale facial images. Training and test processes are implemented using matlab to simulate PCA algorithm. The average images and eigenfaces are illustrated for the reference of the readers. Input images are reconstructed by using the eigenimages obtained through the PCA process. The classification process observes maximum and minimum euclidean distance in determining. Analysis of face component error is generated by showing result of facial features and corresponding values with result of tables.

Finally, chapter 4 focuses on the contributions of future work and on summarizing this thesis.

Chapter 2

FEATURE EXTRACTION AND CLASSIFIERS

2.1 Principal Component Analysis (PCA)

Principal component analysis was developed in 1901 by Karl Pearson [1]. PCA has been used in image recognition and compression to get successful results. PCA is used in linear model systems such as signal processing problems and in classical image processing tecnique. The row image data is decreased to smaller data space from large dimensions with PCA algorithm without loosing discriminative information. The facial parts like eyes and mouth deal with face recognition by including related information that extracted and described face model with concept of evaluated information [4]. This approach will help us to decrease the size of the database to contribute for recognition of test images. Here, the images are stored as feature vectors in the database to find each trained image in the set of eigenfaces. Then, previously stored image features are used to match with all test data to find target image.

In mathematical terms, eigenvectors of the covariance matrix, C, are used as the independent principal components, which can be used to characterize any given vector by the linear integration of the generated eigenvectors. Facial features images can be converted to one dimensional vector and be expressed like linear integration

of the eigenfaces exracted from the covariance matrix of face images. Eigenfaces with higher eigenvalues contribute more to the representation of a typical face image. AC=C, where A is a square matrix, C is the eigenvector of A and C are not a zero vector. In this way, eigenvectors of covariance matrix are calculated along to the direction of components. The data and statistical importance are based on the calculated Eigen values.

2.2 Feature Extraction

In feature extraction, feature vectors are extracted and used for representing face information. Here, Principal Component Analysis is used to extract features for face recognition. The aim is to extract relevant information and encode it as efficiently as possible. In order to identify the subspace of each image space, pixel values are established and spanned by the traning image data. The projection of image is coordinated by classical representation to define principal components. We are projecting face images to achieve more compression, decorrelation and dimensionality reduction by determining the simple way for principal component subspaces [5]. This method uses input and output face images to extract most important features for reducing of eigenvector. In next step the distribution of face images are constructed with covariance matrix by principal component analysis [6].

2.3 Principal Component Analysis (PCA) Algorithms

The mathematical procedure of PCA is as follows:

2.3.1 Face Image Representation

Training set contains M images, with each of size N×N, which are represented by N^2 size vector.

Each face is transformed into single vector such as:

$$\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4, ..., \Gamma_M \quad (2.1)$$

Then, two dimensional vectors are changed to one dimensional column vector where the vector of face is stored in an N×N matrix.

2.3.2 Mean and Mean Centered Images

Average image set is represented as (Ψ):

$$\Psi=(1/M)\sum_{i=1}^{M}\Gamma_i \quad (2.2)$$

Where M is the number of images and Γ_i is a sample image column vector in the traning set. Matrix A is formed from each Ψ by placing them side by side with the difference images that are defined as $\Phi_i = \Gamma_i - \Psi$. Then, the difference matrix A is defined,

$$A= (\Phi_1, \Phi_2, \Phi_3, \ldots\ldots, \Phi_M) \quad (2.3)$$

Each face Γ_i is subtracted from average image to form the difference image,

$$\Phi_i = \Gamma_i - \Psi \qquad i=1, 2, 3, \ldots, M \quad (2.4)$$

2.3.3 Covariance Matrix

Covariance matrix is obtained by

$$C=(1 / M) \sum_{i=1}^{M} \Phi_n \Phi_n^T = AA^T \quad (2.5)$$

Where V_i and μ_i are eigenvectors and eigenvalues of matrix AA^T then equation 2.5 formed as

$$AA^T A V_i = \mu_i A V_i \quad (2.6)$$

Both sides of eguation 2.6 can be multiplied with A to get

$$AA^{T}A\,V_{i} = \mu_{i}A\,V_{i} \tag{2.7}$$

We can see that AV_{i} are the eigenvectors of $C=AA^{T}$ (2.8)

V_{i} of L is constructed MxN matrix to determine eigenfaces U_{m} to form face images to M training set as formula below and; the vectors U_{k} and scalars λ_{k} are the eigenvectors and eigenvalues, of the covariance matrix U_{n} are orthonormal vectors to describe distribution of data

$$L=AA^{T} \text{ where } L_{mn=}(U_{m}^{T}\Phi_{n}) \tag{2.9}$$

$$\lambda_{k=}1/M\ \sum_{i=1}^{M}(\Phi_{n}U_{k}^{T})^{2} \tag{2.10}$$

$$U_{\iota}=\sum_{k=1}^{M}\Phi_{lk}\Phi_{k} \qquad I=1, 2, \ldots., M \tag{2.11}$$

$$U_{k}U_{l}^{T} = \delta_{lk} = \{\ 1, \quad \text{if} \quad l=k\,; \quad 0, \quad \text{otherwise}\ \} \tag{2.12}$$

2.3.4 Data Fusion / Classifier Combination

Data fusion is the process of integration of multiple data and knowledge representing the same real-world object continually and effectively representation. It is well known that in many cases to combining the output of classifier is produced to improved classification result. Each classifier makes an error on a different area of input space. The subset of the input space for each classifier will symbolize a correct classify by extract from one classifiers to another. That means one classifiers is using more that one times by using necessary information. These identify better way to approach the result of the given problem. If we are combining the outputs of various classifiers with two situation. We can say that all classifiers have same properties or they are founding different result with facial features space. Besides of all that cases we can match the performance of sum and product rules. While we are comparing

sum rule it gives the most usable combination rule to develop an easiest and better result that the other if that classifiers makes an error. However, if we are using these classifiers that mean it introduces posteriori probabilities with k nearest-neighbour (k-NN) classifiers to classify and combine sum and product rule with the case of equality. So this problem is defined by considering p-dimensional value with vector x by connected a class which represented as y $\varepsilon\{ c_1, \ldots\ldots, c_l \}$ also this has a test set classes TS = $\{(x_i , y_i)$, i=1,…., T}.That means the problem of L classes and x was observed with approximation of posteriori probability function with p (c_i/x) which give probabilities of pattern x belonging to a given class c_j.

where $x \varepsilon c_k$ if $p(c_k|x) = \max_j p(c_j|x)$

Posteriori probability is calculated with expected outputs, class i and input x is classifiers p $(c_i| x)$ where combination represent G ( ) function $f_i^{comb}(U_i) = G(u_i)$ by using that arithmetic mean $f_i^{comb}(U_i) = 1/N \sum_{j=1}^{N} U_i(j)$ where $U_i(j)$ denotes with the jth component of vector U_i. And after solving the problem by applying the combination formula with k-NN classifiers the rules are give the result with exact and same performance.

Finally, the flowchart below in figure 2.1 represent k-NN classifiers that contribute input image, database, training and test phase with classifiers module to get result of classifiers value. If we are using different set of condition according to size of input image to change the input and output image to perform and by implementing matlab coding.

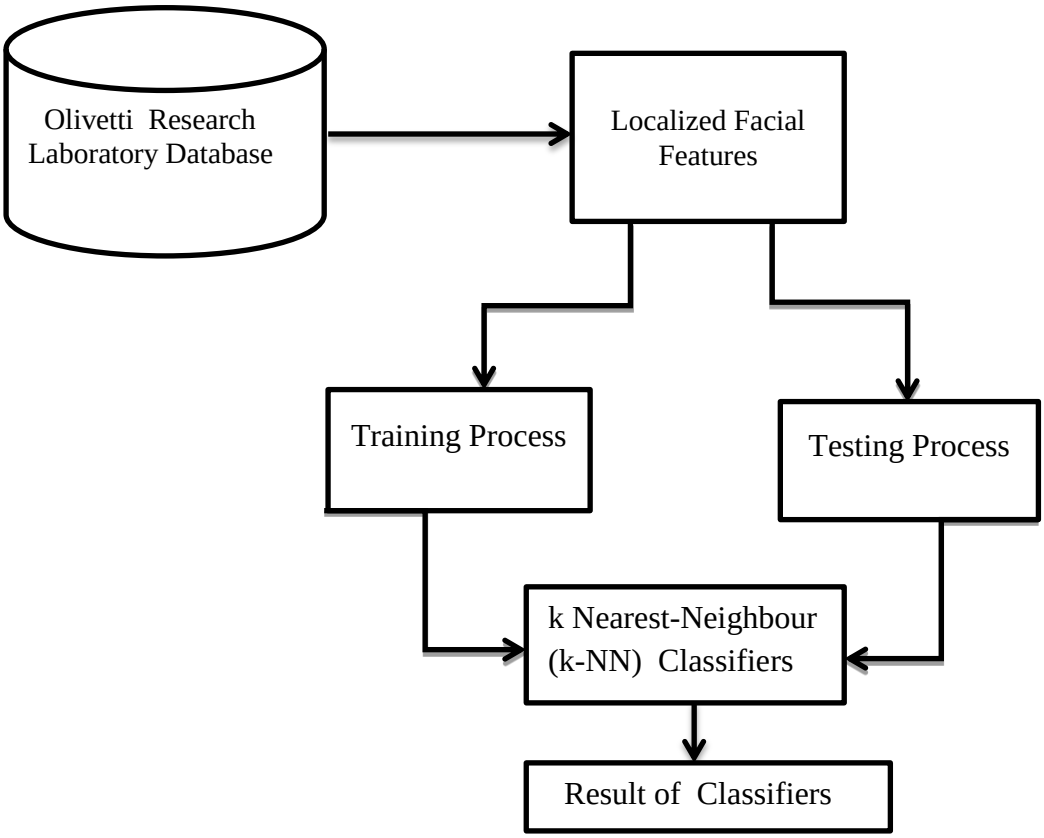

Figure 2.1 Flowchart of k nearest-neighbour (k- NN) classifiers

2.4 Eigenfeatures

The eigenface technique can be extended to include eigenfeatures such as eigeneyes, eigennose, and eigenmouth. That performance value introduces eigenfeatures result to represent the systems are used with localized facial features.

We can match the egienfeatures the same way as the eigenfaces approach. The only difference is the vectors are not coming from faces; instead they are coming from eyes, noses and mouths. Face only global face recognition system has the performance value of 90% of correct classification while database is setting on training and testing facial features result. After using egieneyes, eigennoses,

eigenmouth and combining the results with eigenfaces method increases the performance of face recognition.

2.5 Eigeneyes

PCA is analyzing a set of images of eyes; the data-set cover a large number of eye pairs from a same person, under different position and or in some images where eyes are closed. The Eigen right eye region is based on technique that required value for training and test phase. The performance of system and result is also the eigenfeatures. If we are matching the egien features with classification method from gray-level values and; the result are representing better performance on data-set.

In figure 2.2 images are generated using manual cropping of the right eye. This dataset is used for obtaining encouraging results, where a small number of images are required for training and testing eye recognition system. The recognition performance of the left eye only system is %76.5. The performance of the right eye of the system is %79.5. Figure 2.2 shows manually cropped right eyes and eigeneyes generated from the training set.

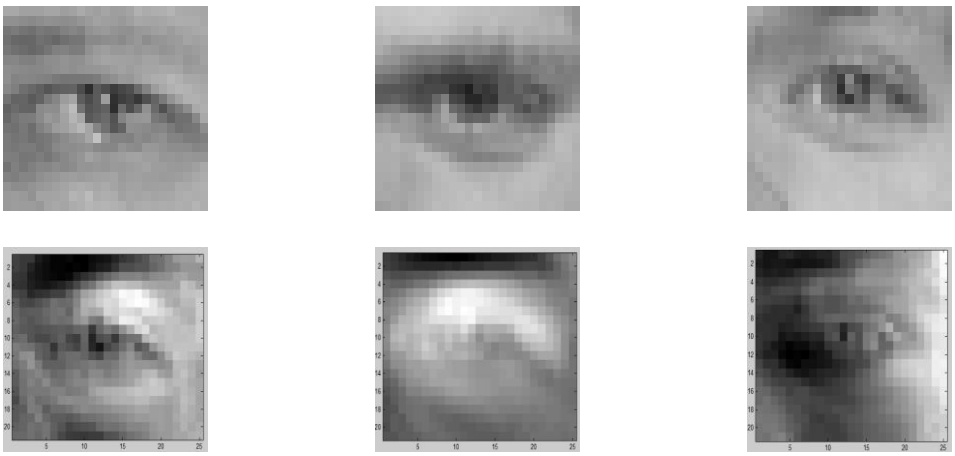

Figure 2.2: First row shows the examples of manually cropped right eyes. Second row shows 3 egieneyes with highest respective eigenvalues.

2.6 Eigennoses

Eigennoses are generated in the same way of eigenfaces, where PCA is used on a set of images of noses and, data-set contains a large number of noses from different persons. Nose only eigennoses method generates %76.5 recognition performance. Figure 2.3 shows manually cropped noses and eigennoses generated from the training set.

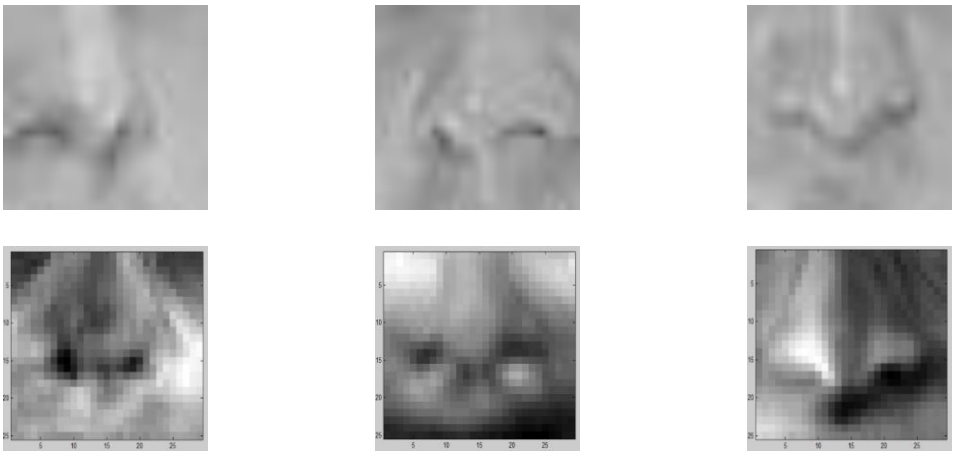

Figure 2.3: First row shows the examples of manually cropped nose. Second row shows 3 egienoses with highest respective eigenvalues

2.7 Eigenmouths

Eigenmouths are generated in the same way of eigenfaces, where PCA is used on a set of images of mouths and, data-set contains a large number of mouths from different persons. Mouth only eigenmouth method generates %74.5 recognition performance. Figure 2.4 shows manually cropped mouth and eigenmouth generated from the training set.

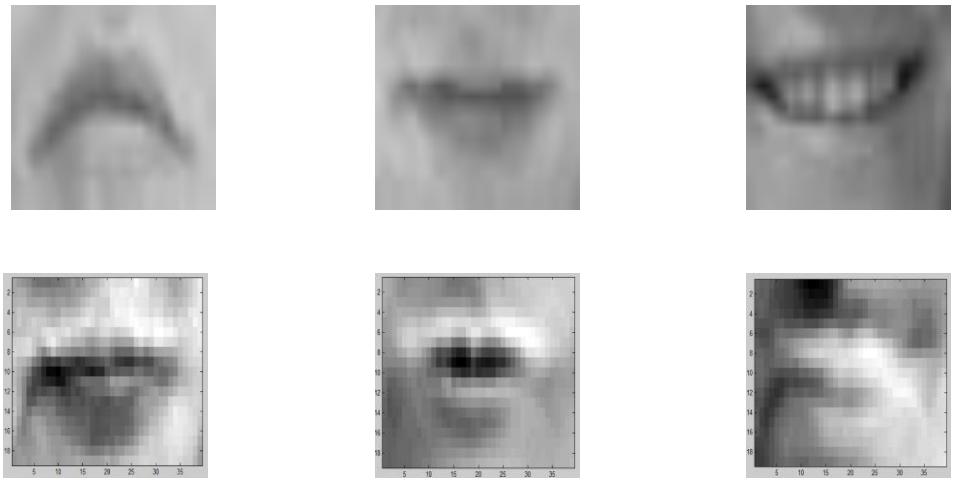

Figure 2.4: First row shows the examples of manually cropped mouth. Second row shows 3 egienmouths with highest respective eigenvalues

2.8 Combination Classifiers / Sum and Product Rule

Classifiers combination is based on context of the fusion data and on distinct representation by using sum, product, and majority rule. The aim of the fusion data is to observe an easiest way to for estimate posteriori class probability.

We can consider Z as assigned with one of the m possible classes $\{ \omega_1, \ldots\ldots, \omega_2 \}$ and R classifiers for distinct measurement vector. This vector are used by i^{th} classifiers with X_i and it is modeled by the probability density function $p(X_i \mid \omega_k)$ and it is a priori probability case to symbolized with $p (\omega_k)$. If $X_i = 1, \ldots, R$ with pattern Z should be assigned to class ω_j, it is assigned to $\theta=\omega_j$, provided posteriori probabilities with formula below:

$$P (\theta = \omega_j \mid X_1, \ldots\ldots, X_R) = \max_k P (\theta = \omega_k \mid X_1, \ldots\ldots, X_R) \quad (2.13)$$

Unconditional measurement joint probability density functions after finding the formula above it represent:

$$P\ (\theta = \omega_k \mid X_1, \ldots\ldots, X_R) = P\ (X_1, \ldots\ldots, X_R \mid \theta = \omega_k)\ P\ (\omega_k) / P(X_1, \ldots\ldots, X_R) \quad (2.14)$$

After the formula above by applying product rule by using X_i= 1, ... , R, $\theta=\omega_j$ to find measurement process model as:

$$P\ (X_1, \ldots\ldots, X_R \mid \theta = \omega_k) = \prod_{i=1}^{R} P\ (X_i \mid \theta = \omega_k) \quad (2.15)$$

And a posteriori probabilities are generated by individual classifiers formula to develop product rule by showing the calculation below:

$$p^{(-(R-1))}(\omega_j) \prod_{i=1}^{R} P(\theta = \omega_{k \mid X_i}) = \max_{k=1}^{m}\ P^{(-(R-1))}(\omega_k)\ \prod_{i=1}^{R} P(\theta = \omega_{k\ \mid X_i})\ P\ (X_i) \quad (2.16)$$

The several rule of fusing the classifiers outputs and the product rule is expanded by calculating summation, reference value P_i, maximum value to obtain sum decision rule:

$$(1 - R)P(\omega_j) + \sum_{i=1}^{R} p(\omega_j \mid X_i)p(X_i)\ / P_i\ =$$
$$\max_{k=1}^{m} \left[(1 - R)P(\omega_k) + \sum_{i=1}^{R} p(\omega_k \mid X_i)p(X_i)\ / P_i \right] \quad (2.17)$$

2.9 Majority Voting Rule

$$\sum_{i=1}^{R} \Delta_{ji} = \max_{k=1}^{m} \sum_{i=1}^{R} \Delta_{ki} \quad (2.18)$$

This formula is finded from sum rule with outputs P ($\theta=\omega_k \mid x_i$), Δ_{ki}=1

If P ($\theta=\omega_k \mid x_i$) = $\max_{k=1}^{m}$ P ($\theta=\omega_l \mid x_i$) otherwise is zero.

Classifiers have a hypothesis by receiving each class to count all votes of ω_k. After doing this counting, each class is selected according to its value to determine the majority rule to estimate error and perform better value for product, sum, majority rule.

2.10 Comparison of Classifier Combination Rules

Classifier has three important rules product, sum and majority voting. These rules are observed and enhanced by priory class probabilities. If we compare product and sum rules the results show that it produces for estimating facial features errors value. After that it will show the sum rule is not effective enough to estimate errors while it makes the decision more realistic. Classifiers combination by means sum rule, product rule; majority voting is the main contribution of this thesis. Figure 2.5 illustrate the main block diagram of the proposed classifier combination set up which is adopted in this thesis.

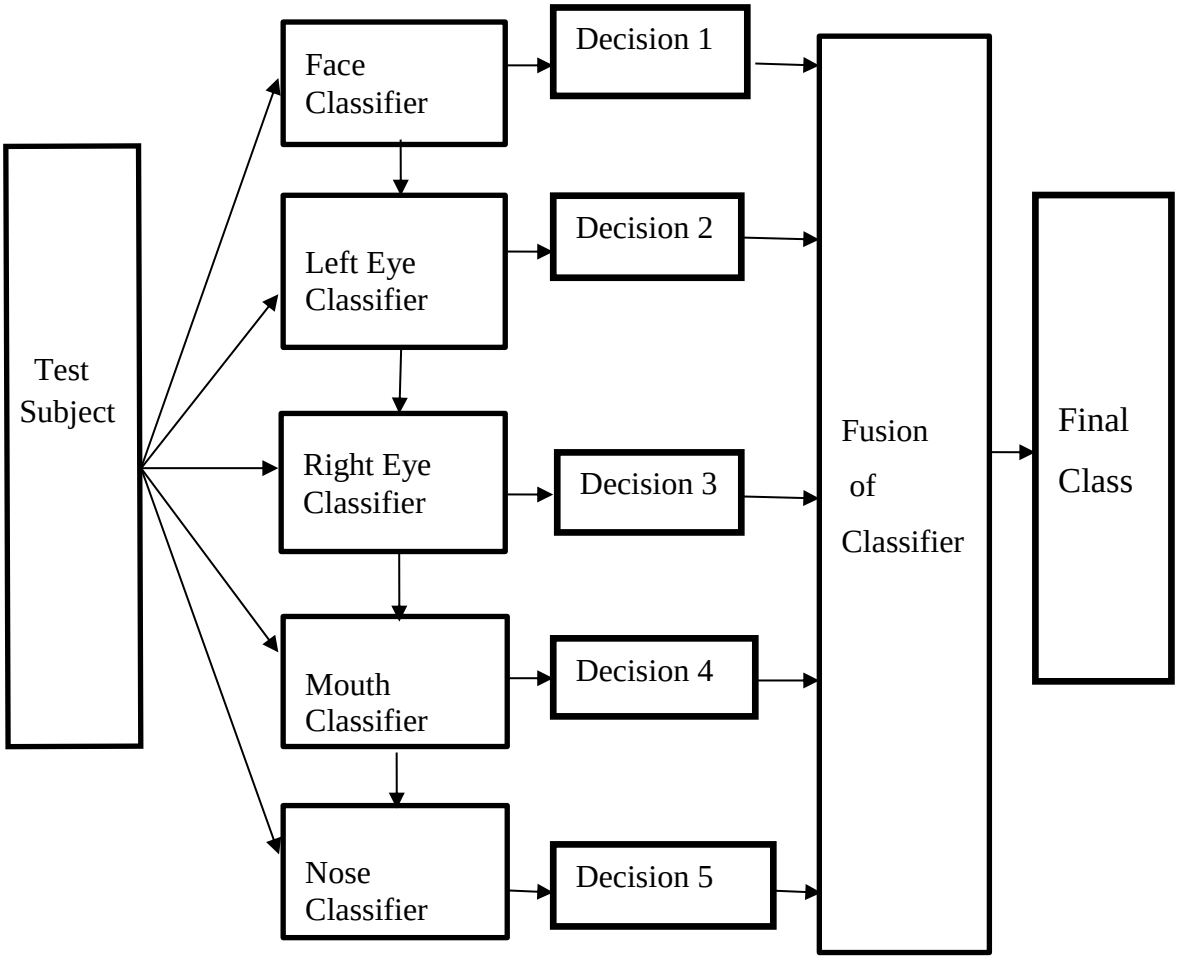

Figure 2.5: Proposed classifier combination set up which is adopted in this thesis.

Chapter 3

IMPLEMENTATION OF PCA-BASED FACE RECOGNITION ALGORITHM

In this chapter, we introduce a system in which face images are stored in a library, feature vectors are extracted using PCA and identifies of test images are determined using these features. In the first section, the PCA approach is introduced. In the second section, feature vector extractions over facial images are introduced.

3.1 Database Properties

This section explains the extraction of important features of facial images in database. The properties of images such as the recording condition, image resolution and total number of images are given in Table 1.1. The facial image database used in this thesis Olivetti Research Laboratory Database (ORL) [8].

Table 3.1: Olivetti Research Lab (ORL) Database [8]

Number of Subjects	**Number of Samples Per Subject**	**Image Resolution**	**Total number of Images**
40	10	92x100	400

The ORL database was collected between 1992 and 1994 [9], reviewed by Phillips and Newton [10]. Principal component analysis (PCA) is a statistical dimensionality reduction method. Facial features extracted by PCA are commonly referred to as eigenfaces. With PCA, database images must be of the same size and must first be

normalized to line up the eyes and mouth of the subjects within the images. The PCA approach is then applied to reduce the dimensionality of the original data. This reduction in dimensions removes information that is not useful and nearly decomposes the face structure into uncorrelated components that are known as eigenfaces. Each localized facial feature may be represented as a feature vector of the eigenfaces, which are stored and a probe images is compared against with database image by measuring the distance between their respective feature vectors. The PCA approach typically requires the full frontal face to be presented each time. PCA method uses eigenvectors and eigenvalues for representing localized facial features. These eigenvectors can be thought of as a set of features which together characterize the variation between face images. Each image location contributes more or less to each eigenvector, so that we can display the eigenvector as a sort of not proper face which is called as an eigenface. A flowchart showing the implementation of PCA-based face recognition approach is given below:

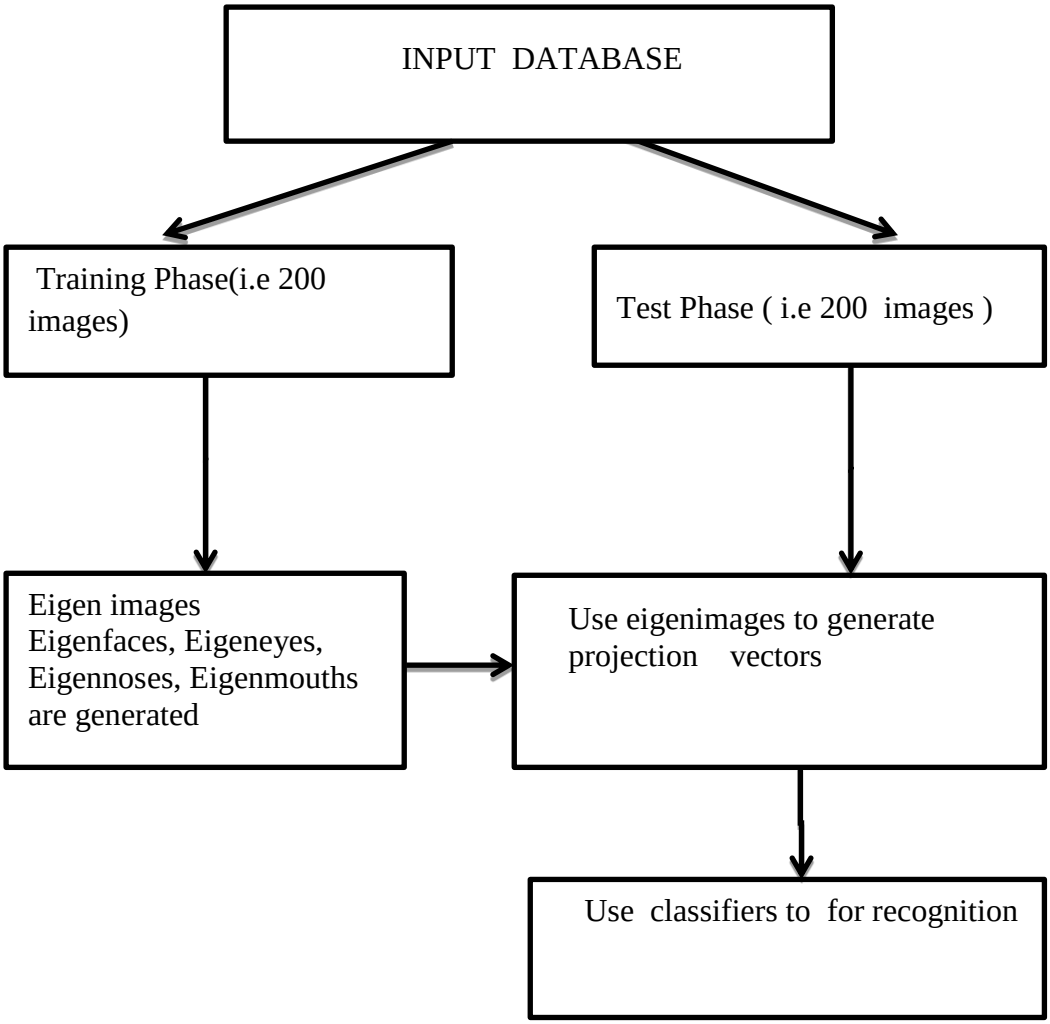

Figure 3.1: General Block diagram of the PCA based recognition system.

3.2 Training Phase

Training sets includes a subset of facial images selected to compute the reference features. A flowchart describing the stops of the training phase of the PCA-based recognition algorithm is given in Figure 3.2.

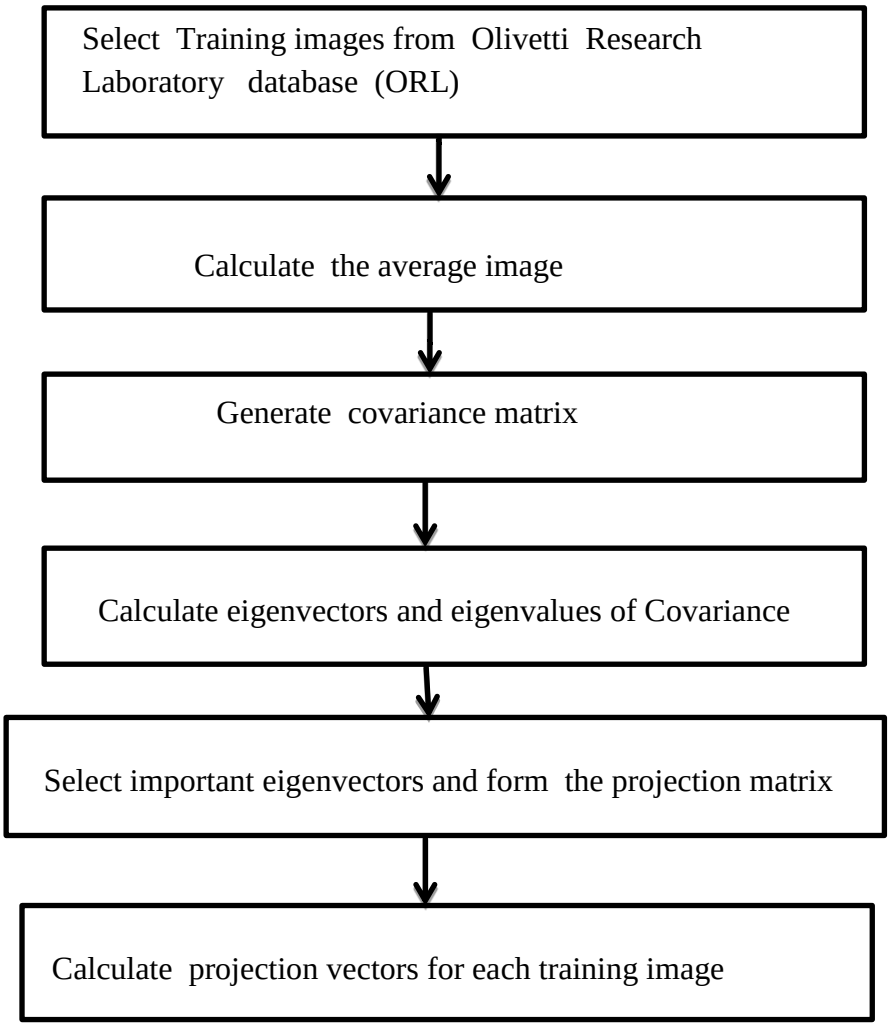

Figure 3.2: Flowchart for training phase of PCA based face recognition

3.3 Testing Phase

The test phase is carried out by following steps as in figure 3.3:

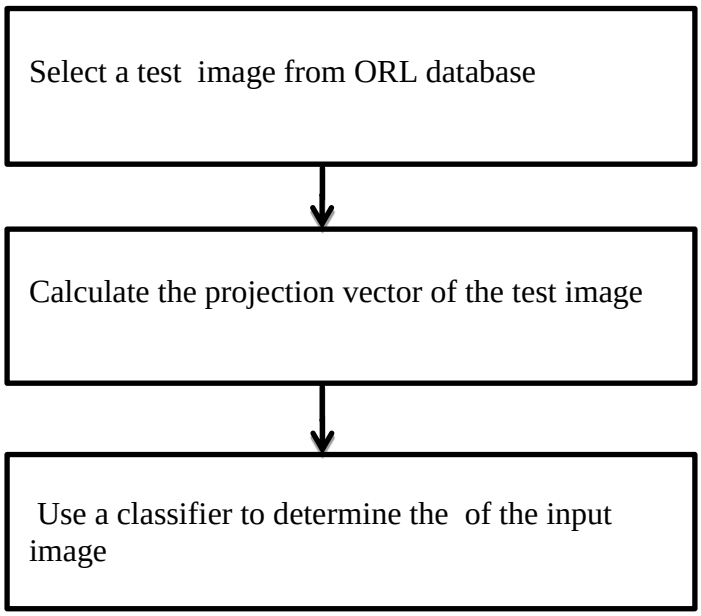

Figure 3.3: Flowchart of test phase of PCA based face recognition

3.4 Experiments

In this thesis, the facial feature extraction algorithm is applied on black and white facial images. The eigenfeatures used in the classifiers correspond to different region of images. For this purpose, left eye, right eye, nose and mouth part of facial images are manually cropped and saved into different libraries. Each part is labeled with identities of the corresponding images. Consequently, PCA features of each local region facial part are computed as explained in the previous section.

3.5 Simulation Results

3.5.1 Average Image

The mean image observed over frontal holistic traning. Also, the mean image observed over frontal holistic training images $\Gamma_1, \Gamma_2, \Gamma_3, \ldots, \Gamma_n$ is shown in Figure 3.4. That is computed as:

$$\Psi = (1/M) \sum_{i=1}^{M} \Gamma_i \quad (2.24)$$

$$\Phi_i = \Gamma_i - \Psi \quad (2.25)$$

Figure 3.4: Average image

3.5.2 Eigenfaces

In figure 3.5 same of the eigenfaces of the training set images, computes using the PCA algorithm is above.

Figure 3.5: Eigenfaces corresponding to tranining set images.

An example of Eucledian Distances from a test image features to all image features in training data set is illustrated in Figure 3.6.

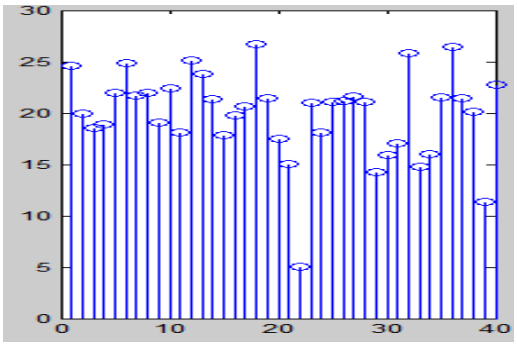

Figure 3.6: Euclidian distance (y-axis) from test image features to train image features each subject ID 22. Subject ID 22 has the smallest distance.

3.6 Performances of Face, Left Eye, Right Eye, Nose and Mouth Classifiers

There are different performance results for the face, left eye, right eye, and nose and mouth classifiers. In the figure 3.1, the respective performance of each classifier is generated for the ORL dataset, where five samples for each class are used for traning and the remaining five samples for each class are used for test set.

The reported performance (%) in figure 3.1 and throughout the thesis is the recognition rate which is defined as follows:

$$\text{Recognition Rate (\%)} = \frac{\text{Number of corrrect classifications}}{\text{Total number of test images}} \times 100$$

(2.26)

Table 3.2: Performance of face, Left Eye, Right Eye, Nose and Mouth

Localized Facial Features Classifier	**Performance (%)**
Face Classifiers	90.0
Left Eye Classifier	76.5
Right Eye Classifier	79.5
Nose Classifier	76.5
Mouth Classifier	74.5

3.7 Result of Face, Left Eye, Right Eye, Nose and Mouth

For each classifier in Table 3.2 the classification results of five different test samples are given for the first five Person IDs. As you can see, the first row shows that the majority of the reported classes are for Person ID 1, which is the correct class. The rest of the rows can also be studied to determine the correct classes.

Table 3.3: Classifier output for Face, Left Eye, Right Eye, Nose and Mouth Classifiers.

Distinct Classifiers	Class ID	Classifier Output for each sample (True/ False)
Face Classifier	1	**1 1 1** 32 **1**
	2	**2 2 2 2 2**
	3	3 23 8 40 21
	4	4 3 **5 5** 33
	5	**5 5** 33 23 40
Left Eye Classifier	1	**1 1** 40 **1** 25
	2	**2 2 2 2 2**
	3	3 **23** 3 **23 23**
	4	2 4 **5 5** 31
	5	**5 5** 31 **5** 15
Right Eye Classifier	1	**1 1 1 1 1**
	2	**2 2 2 2 2**
	3	38 35 **12 12** 18
	4	**4 4 5 5** 31
	5	**5 5** 31 5 29
Nose Classifier	1	**1 1 1 1 1**
	2	19 **2** 19 **2 2**
	3	13 9 7 **13 13**
	4	**4 4 5 5** 31
	5	**5 5** 31 **5** 33
Mouth Classifier	1	**1 1 1 1 1**
	2	**2 2 2 2 2**
	3	**3 3 3** 23 **3**
	4	**4 4 5 5** 31
	5	**5 5** 31 **5** 40

3.8 Performance Analysis of Majority Voting, Sum and Product Methods

Performance of each data fusion technique is given in Table 3.3 as you can observe majority voting give the best performance by the %94.5.

Table 3.4: Classification performance of different fusion techniques.

Data Fusion Technique	Performance Rate (%)
Majority Voting	94.50
Sum	89.50
Product	90.00

3.9 Sum and Product Data Fusion Methods

There are five classifiers that can be subject to data fusion approaches such as Sum Rule and Product Rule. These classifiers are: face classifier, left eye classifier, right eye classifier, nose classifier and mouth classifier. Table 3.4 shows the known IDs (first column of the table) of the test samples in the test set for the first five subjects. Note that for each subject there are five samples in the training set. Second and third columns show the classified ID and its respective probability for the face classifier. The columns for 3 to 10 correspond the classified IDs and the respective probabilities of each feature classifier. Columns 11 and 12 are showing the IDs recognized after the data fusion techniques sum rule and product rule respectively.

Table 3.5: Classification Performance of individual classifiers and data fusion techniques of sum and product rules.

Known ID	Classifier Type											
	Face		Left Eye		Right Eye		Nose		Mouth		Sum	Pro.
Column Number	1	2	3	4	5	6	7	8	9	10	11	12
	ID	Prob	ID	Prob	ID	Prob	ID	Prob	ID	Prob	ID	ID
1	1	0,67	1	0,88	1	0,91	1	0,90	1	0,88	1	1
	1	0,67	1	0,88	1	0,91	1	0,89	1	0,89	1	1
	1	0,60	1	0,88	1	0,90	1	0,89	1	0,89	1	1
	1	0,59	1	0,89	1	0,90	1	0,90	1	0,91	1	1
	1	0,67	1	0,91	1	0,89	1	0,90	1	0,87	1	1
2	2	0,59	1	0,88	1	0,86	2	0,89	2	0,87	2	2
	2	0,64	2	0,89	2	0,89	2	0,84	2	0,84	2	2
	2	0,68	2	0,86	2	0,88	2	0,88	2	0,89	2	2
	2	0,78	2	0,89	2	0,89	2	0,88	2	0,87	2	2
	2	0,61	2	0,88	2	0,88	2	0,86	2	0,89	2	2
3	3	0,66	2	0,85	2	0,87	38	0,81	23	0,82	23	3
	3	0,49	3	0,83	23	0,82	3	0,82	9	0,80	3	3
	3	0,49	23	0,83	38	0,84	25	0,81	9	0,80	3	3
	3	0,46	3	0,84	23	0,82	4	0,85	4	0,83	4	3
	3	0,60	3	0,86	4	0,85	4	0,85	3	0,84	3	3
4	4	0,67	3	0,86	4	0,85	4	0,83	4	0,85	4	4
	4	0,51	4	0,85	4	0,84	4	0,92	4	0,93	4	4
	4	0,74	4	0,92	4	0,91	4	0,94	4	0,91	4	4
	4	0,76	4	0,88	4	0,89	4	0,89	4	0,92	4	4
	4	0,68	4	0,86	4	0,87	4	0,89	4	0,89	4	4
5	5	0,70	4	0,92	4	0,88	5	0,91	5	0,94	5	5
	5	0,81	5	0,93	5	0,94	5	0,93	5	0,92	5	5
	5	0,81	5	0,92	5	0,94	5	0,85	5	0,86	5	5
	5	0,57	5	0,86	5	0,87	5	0,88	5	0,89	5	5
	5	0,60	5	0,88	5	0,88	5	0,86	5	0,85	40	40

3.10 Classifier output for Face+Left Eye, Face+Right Eye Classifiers

Table 3.5 first column show the known IDs for the first two subjects. Columns 1-2 and 3-4 shows face+left eye and face+right eye results.

Table 3.6: Classification Performance of individual classifiers and data fusion techniques of sum and product rules.

Known IDs	Classifier Type			
	Face+Left Eye		Face+Right Eye	
	IDs	Recognition Ratio	IDs	Recognition Ratio
1	1	0.91	1	0.91
	1	0.89	1	0.89
	1	0.89	1	0.89
	1	0.89	1	0.89
	1	0.85	1	0.85
2	2	0.88	1	0.88
	2	0.87	2	0.87
	2	0.88	2	0.88
	2	0.87	2	0.87
	2	0.86	2	0.86
3	3	0.82	2	0.82
	3	0.83	3	083
	3	0.81	23	0.81
	3	0.84	3	0.84
	3	0.84	3	0.84
4	4	0.84	3	0.84
	4	0.91	4	0.91
	4	0.88	4	0.88
	4	0.86	4	0.86
	4	0.88	4	0.88
5	5	0.94	4	0.94
	5	0.94	5	0.94
	5	0.87	5	0.87
	5	0.88	5	0.88
	5	0.88	5	0.88

3.11 Classifier output for Face+Nose, Face+Mouth Classifiers

Table 3.6 first column show the known IDs for the first two subjects. Columns 1-2 and 3-4 shows face+nose eye and face+mouth eye results.

Table 3.7: Classification Performance of individual classifiers and data fusion techniques of sum and product rules.

Known IDs	Classifier Type			
	Face+Nose		Face+Mouth	
	IDs	Recognition Ratio	IDs	Recognition Ratio
1	1	0.89	1	0.88
	1	0.88	1	0.88
	1	0.89	1	0.89
	1	0.89	1	0.90
	1	0.89	1	0.87
2	2	0.89	1	0.87
	2	0.84	2	0.84
	2	0.87	2	0.88
	2	0.88	2	0.86
	2	0.86	2	0.88
3	3	0.81	2	0.82
	3	0.81	3	0.80
	3	0.81	23	0.79
	3	0.84	3	0.83
	3	0.84	3	0.84
4	4	0.82	3	0.84
	4	0.92	4	0.92
	4	0.93	4	0.90
	4	0.88	4	0.91
	4	0.89	4	0.89
5	5	0.91	4	0.94
	5	0.93	5	0.91
	5	0.85	5	0.86
	5	0.87	5	0.88
	5	0.85	5	0.85

3.12 Result of Majority Voting Rule

Data fusion techniques are very useful in the present of multiple classifiers. As we have five classifiers (Face / Left eye / Rigth eye / Nose and Mouth classifiers) using data fusion techniques such as Sum Rule, Product Rule, and Majority Voting are expected to improve the classification performance.

Performance of each data fusion technique is given in Table 3.7. As you can observe majority voting give the best performance by the %95.0.

Table 3.8: Performance of Majority Voting Rule

Face(%) Classifier	**Face+ Left eye (%) Classifier**	**Face+Right eye (%) Classifier**	**Face+ Nose(%) Classifier**	**Face+Mouth(%) Classifier**	**Majority Voting(%) Classifier**
90.0	93.5	91.5	93.5	90.0	95.0

3.13 Performance result of Sum and Product

Performance of each data fusion technique is given in Table 3.8 As you can observe Sum and product rule give the best performance by the %94.5 and %90.0 respectively.

Table 3.9: Performance of Sum and Product Rule

Face(%) Classifier	**Face+ Left eye(%) Classifier**	**Face+ Right eye(%) Classifier**	**Face+ Nose(%) Classifier**	**Face+ Mouth(%) Classifier**	**Sum(%) Classifier**	**Product(%) Classifier**
90.0	93.5	91.5	93.5	90.0	94.5	90.0

Chapter 4

CONCLUSION

The contribution of feature classifiers such as eye classifier, nose classifier, and mouth classifier to the face only classifier system has been studied and analized. In this context PCA based eigeneyes, eigennose, eigenmouth, based classifier have been fused by the three classifier combination rules.

In this thesis, face images are defined by Olivetti Research Laboratory database (ORL) and facial features in the form of eigenfaces are extracted using PCA algorithm.

PCA is a general algorithm to calculate dimension for interchanging correlated vectors. The main advantage of PCA is providing a lower dimensional representation of very high dimensional data.

The principal components of training images are computed by PCA and localized facial fetures is succeeding by using facial images on lower-dimensional space represented by eigenfaces vectors. Euclidean distance between projection vectors of training set images and test image is compared to identify class of test image.

There are different performance result of facial features manually cropped as left eye, right eye, nose and mouth. This data fusion techniques such as Sum Rule,

Product Rule, and Majority Voting are expected to improve the classification performance such as %90 performance of face while we are matching according to the other component of face, %76 performance of left eye, %79 performance of right eyes, %76 performance of nose and %74 performance of result are founded by runing matlab coding according to the performance of the ORL database.

4.1 Future Works

Use of other dimensionality algorithms in comparison to PCA, such as linear discriminant analysis, locally linear embedding, and independent component analysis, is planned for future studies.

REFERENCES

[1] Yusuf, A., (2009). “Face Recognition” http://www.mcs.cankaya.edu.tr/proje/ 2009/yaz/yusuf_atilgan/rapor.pdf.

[2] Rajkiran, G., Vijayan, K. (2002). “An improved face recognition technique based on modular PCA approach”VLSI Systems Laboratory, Department of Electrical and Computer Engineering, Old Dominion University, 231 Kaufman Hall, Norfolk, VA 23529-0246, USA, Pattern Recognition Letters 25 , pp.429–436.

[3] “Face Recognition”, (2013). “http://www. Facial Recognition”.

[4] Vinay.H., Ashwını.M. “Face Recognition Using Eigenface Approach” Malardalen University, Vasteras, Sweden.http://ircse09_submission_17.pdf.

[5] Taranpreet, S.,Face Recognition Based On Pca Algorithm”.Special Issue of International Journal of Computer Science & Informatics **(**IJCSI), ISSN **(**PRINT**)** : 2231–5292, Vol.- II, Issue-1, 2.

[6] Mazen,E.,(2008) “PCA And LDA Based Face Recognition Using Region Division With Majority Voting”, Department of Computer Engineering.

[7] Josef,K.,Mohamad,H.,Robert,D.,Jiri,M.,(1998).“On Combining Classifiers”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol: 20, No.3, pp: 226– 238.

[8] Ralph, G.,(2005) “Face Databases”, The Robotics Inistitute, Carnegie Mellon University. Handbook of Face Recognition. Springer-Verlag.

[9] F. Samaria and A. Harter. Parameterisation of a stochastic model for human face identification.In 2nd IEEE Workshop on Applications of Computer Vision, Sarasota, FL, 1994.

[10] P. J. Phillips and E. M. Newton. Meta-analysis of face recognition algorithms. In 5th IEEE Conf. on Automatic Face and Gesture Recognition, Washington, DC, 2002.

[11] Gökberk, B., “Face Recognition using Principal Component Analysis” http://www.Project_spec.pdf.

APPENDIX

face_Recog

```
Clear   all    close all    clc
Cd   'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400'
N=40;    %number of   images
NT=1;    %number of row images
k=1;
S=[];    % img matrix
for i=1:N
for j=1:10
if(j<=NT)
idx=(i-1)*10+j;
str=strcat(int2str (idx ),'.pgm');     % concatenates two strings that form the name of the image
X=double(imread(str))/255.0;  img=X;
[irow icol]=size(img);    % get the number of rows (N1) and columns (N2)
temp=reshape(img', irow*icol ,1);    % creates a (N1*N2)x1 vector
S=[S temp];
k=k+1;
end
end
end
% mean image
m=mean(S,2);  % obtains the mean of each row instead of each column
tmimg=(m); % converts to unsigned 8-bit integer. Values range from 0 to 255
img=reshape(tmimg,icol,irow); % takes the N1*N2x1 vector and creates a N1xN2 matrix
img=img';
figure;
imshow(uint8(255.0*img));
title( 'Mean Image  ','fontsize',16)
for i=1:N*NT
dbx(:,i)=double(S(:,i)-m);
end
%Covariance matrix C=A'A, L=AA'
A=dbx';
L=A*A';
[vv dd]=eig(L);
v=[];
d=[];
for i=1:size(vv,2)
if(dd(i,i)>1e-4)
v=[v vv(:,i)];
d=[d dd(i,i)];
end  end


%sort, will return an ascending sequence
[B index]=sort(d);
ind=zeros(size(index));
```

```
dtemp=zeros(size(index));
vtemp=zeros(size(v));
len=length(index);
for i=1:len
dtemp(i)=B(len+1-i);
ind(i)=len+1-index(i);
vtemp(:,ind(i))=v(:,i);
end
d=dtemp;    v=vtemp;
%Normalization of eigenvectors
for i=1:size(v,2) %access each column
kk=v(:,i);
temp=sqrt(sum(kk.^2));
v(:,i)=v(:,i)./temp;
end

%Eigenvectors of C matrix
u=[];
for i=1:size(v,2)
temp=sqrt(d(i));
u=[u (dbx*v(:,i))./temp];
end

%Normalization of eigenvectors
for i=1:size(u,2)
kk=u(:,i);
temp=sqrt(sum(kk.^2));
u(:,i)=u(:,i)./temp;
end

% show eigenfaces
figure(4);
for i=1:10
img=reshape(u(:,i),icol,irow);
img=img';
img=histeq(img,255);
subplot(2,5,i) ,imshow(img );
if i==3
title(' Eigenfaces','fontsize',18)
end end

% Find the weight of each face in the training set
omega = [];
for h=1:size(dbx,2)
WW=[];
for i=1:size(u,2)
t = u(:,i)';

WeightOfImage = dot(t,dbx(:,h)');
WW = [WW; WeightOfImage];  end
omega = [omega WW];   end
```

```
%%%%%%%%%%%%%%%% test phase %%%%%%%

cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400'
k=1;
T=[];    % img matrix

for i=1:N
for j=1:10
if(j>NT)
idx=(i-1)*10+j ;

str=strcat(int2str (idx ),'.pgm');     % concatenates two strings that form the name of the image
X=double(imread(str))/255.0;  img=X;
[irow icol]=size(img);    % get the number of rows (N1) and columns (N2)
temp=reshape(img',irow*icol,1);    % creates a (N1*N2)x1 vector
T=[T temp];

End end end
temp=double(T(:, 196)); %InImage;
me=mean(temp);
NormImage = temp;
Difference = temp-m;
figure(3) ;subplot(1,2,1), imshow((reshape(temp,icol,irow))');
p = [];
aa=size(u,2);
for i = 1:aa
pare = dot(NormImage,u(:,i));
p = [p; pare]; end

ReshapedImage = m + u(:,1:aa)*p; %m is the mean image, u is the eigenvector
ReshapedImage = reshape(ReshapedImage,icol,irow);
ReshapedImage = ReshapedImage';
%show the reconstructed image.

figure(3),subplot(1,2,2)
imagesc(ReshapedImage); colormap('gray');
title('Reconstructed image','fontsize',18)
InImWeight=[];
for i=1:size(u,2)
t = u(:,i)';
WeightOfInputImage = dot(t,Difference');
InImWeight=[InImWeight; WeightOfInputImage]; end
ll = 1:N*NT-1;
figure subplot(1,2,1)

stem(ll,InImWeight)
title('Weight of Input Face','fontsize',14)
% Find Euclidean distance
e=[];
```

```
for i=1:size(omega,2)
q=omega(:,i);
DiffWeight=InImWeight-q;
mag=norm(DiffWeight);
e=[e mag]; end
kk=1:size(e,2);
subplot(1,2,2)
stem(kk,e)
title('Eucledian distance of input image','fontsize',14)
[MaximumValue maxidx]=max(e)  % maximum eucledian distance
[MinimumValue minidx]=min(e)    % minimum eucledian distance
Result : MaximumValue = 26.7495

maxidx =18    MinimumValue =5.0569     minidx =22
```

---

face_Recog_tek

```
clear all    close all    clc
%cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400'
N= 40;    %number of  images
NT= 5;    %number of training set
k=1;
S=[];    % img matrix
for i=1:N
for j=1:10
if(j<=NT)
idx=(i-1)*10+j ;
str=strcat(int2str (idx ),'.pgm'); % concatenates two strings that form the name of the
image
X=double(imread(str))/255.0;  img=X;
[irow icol]=size(img); % get the number of rows (N1) and columns (N2)
temp=reshape(img', irow*icol ,1);    % creates a (N1*N2)x1 vector
S=[S temp];
k=k+1;
end end end
% mean image
m=mean(S,2);  % obtains the mean of each row instead of each column
tmimg=(m); % converts to unsigned 8-bit integer. Values range from 0 to 255

img=reshape(tmimg,icol,irow); % takes the N1*N2x1 vector and creates a N1xN2
matrix
img=img';
for i=1:N*NT
dbx(:,i)=double(S(:,i)-m);  end

%Covariance matrix C=A'A, L=AA'
A=dbx';
```

```
L=A*A';
[vv dd]=eig(L);
% Sort and eliminate those whose eigenvalue is zero
v=[];   d=[];

for i=1:size(vv,2)
if(dd(i,i)>1e-4)
v=[v vv(:,i)];
d=[d dd(i,i)]; end end
%sort, will return an ascending sequence
[B index]=sort(d);
ind=zeros(size(index));
dtemp=zeros(size(index));
vtemp=zeros(size(v));
len=length(index);
for i=1:len
dtemp(i)=B(len+1-i);
ind(i)=len+1-index(i);
vtemp(:,ind(i))=v(:,i); end
d=dtemp;    v=vtemp;

%Eigenvectors of C matrix
u=[];
for i=1:size(v,2)
temp=sqrt(d(i));
u=[u (dbx*v(:,i))./temp]; end
% Find the weight of each face in the training set
omega = [];
for h=1:size(dbx,2)
WW=[];
for i=1:size(u,2)
t = u(:,i)';
WeightOfImage = dot(t,dbx(:,h)');
WW = [WW; WeightOfImage]; end
omega = [omega WW]; end

%%%%%%%%%%%%%%% test phase %%%%%%

cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400'
k=0; count=0 ;   kk=0;
T=[]; % img matrix
for i=1:N
for j=1:10
if(j>NT)
idx=(i-1)*10+j ; count=count+1;
str=strcat(int2str (idx ),'.pgm'); % concatenates two strings that form the name of the
image
X=double(imread(str))/255.0;  img=X;
[irow icol]=size(img); % get the number of rows (N1) and columns (N2)
temp=reshape(img',irow*icol,1); % creates a (N1*N2)x1 vector
```

```
temp=double(temp); %InImage;
me=mean(temp);
NormImage = temp;
Difference = temp-m;
InImWeight=[];
for ii=1:size(u,2)
t = u(:,ii)';
WeightOfInputImage=dot(t,Difference');
InImWeight=[InImWeight; WeightOfInputImage]; end
% Find Euclidean distance
e=[];
for iii=1:size(omega,2)
q=omega(:,iii);
DiffWeight=InImWeight-q;
mag=norm(DiffWeight);
e=[e mag]; end
[MinimumValue minidx]=min(e);    % minimum eucledian distance
%minidx
outidx=floor((minidx-1)/(NT))+1;
RESULT(count,1)=i;
RESULT(count,2)=outidx;
if(i==outidx)
k=k+1;
else
kk=kk+1; end end end  end
performance=k/count*100
```

RESULT : performance = 90

---

leye_Recog_tek

```
clear all    close all    clc
cd 'C:\Users\TOSHIBA\Desktop\l_eye'
N= 40;    %number of  images
NT=5;    %number of training set
k=1;    %number of  true result  performance of  images
S=[];    % img matrix
for i=1:N
for j=1:10
if(j<=NT)
idx=(i-1)*10+j ;
str=strcat('l_eye',int2str (idx ),'.bmp'); % concatenates two strings that form the name
of the image
X=double(imread(str))/255.0;  img=X;
[irow icol]=size(img); % get the number of rows (N1) and columns (N2)
temp=reshape(img', irow*icol ,1);    % creates a (N1*N2)x1 vector
S=[S temp];
k=k+1; end end  end
% mean image
```

```
m=mean(S,2);  % obtains the mean of each row instead of each column
tmimg=(m); % converts to unsigned 8-bit integer. Values range from 0 to 255
img=reshape(tmimg,icol,irow); % takes the N1*N2x1 vector and creates a N1xN2 matrix
img=img';
for i=1:N*NT
dbx(:,i)=double(S(:,i)-m); end
%Covariance matrix C=A'A, L=AA'
A=dbx';
L=A*A';
[vv dd]=eig(L);
% Sort and eliminate those whose eigenvalue is zero
v=[]; d=[];
for i=1:size(vv,2)
if(dd(i,i)>1e-4)
v=[v vv(:,i)];
d=[d dd(i,i)]; end  end

%sort, will return an ascending sequence
[B index]=sort(d);
ind=zeros(size(index));
dtemp=zeros(size(index));
vtemp=zeros(size(v));
len=length(index);
for i=1:len
dtemp(i)=B(len+1-i);
ind(i)=len+1-index(i);
vtemp(:,ind(i))=v(:,i); end
d=dtemp;
v=vtemp;
%Eigenvectors of C matrix
u=[];
for i=1:size(v,2)
temp=sqrt(d(i));
u=[u (dbx*v(:,i))./temp];  end

% Find the weight of each face in the training set
omega = [];
for h=1:size(dbx,2)
WW=[];
for i=1:size(u,2)
t = u(:,i)';
WeightOfImage = dot(t,dbx(:,h)');
WW = [WW; WeightOfImage]; end
omega = [omega WW]; end
%%%%%%%%%%%%%%%% test phase %%%%%%

cd 'C:\Users\TOSHIBA\Desktop\l_eye'
k=0; count=0 ; kk=0; %number of false results of images
T=[]; % img matrix
```

```
for i=1:N
for j=1:10
if(j>NT)
idx=(i-1)*10+j ; count=count+1;
str=strcat('l_eye',int2str (idx ),'.bmp'); % concatenates two strings that form the name
of the image
X=double(imread(str))/255.0;  img=X;
[irow icol]=size(img); % get the number of rows (N1) and columns (N2)
temp=reshape(img',irow*icol,1); % creates a (N1*N2)x1 vector
temp=double(temp); %InImage;
me=mean(temp);
NormImage = temp;
Difference = temp-m;
InImWeight=[];
for ii=1:size(u,2)
t = u(:,ii)';
WeightOfInputImage=dot(t,Difference');
InImWeight=[InImWeight; WeightOfInputImage]; end
% Find Euclidean distance
e=[];
for iii=1:size(omega,2)
q=omega(:,iii);
DiffWeight=InImWeight-q;
mag=norm(DiffWeight);
e=[e mag]; end
[MinimumValue minidx]=min(e);    % minimum eucledian distance
%minidx
outidx=floor((minidx-1)/(NT))+1;
RESULT(count,1)=i;
RESULT(count,2)=outidx;
if(i==outidx)
k=k+1; else
%minidx
kk=kk+1; end  end end  end
performance=k/count*100
```

RESULT : performance =79   **// NT:5

Reye_Recog_son

```
clear all    close all    clc
cd 'C:\Users\TOSHIBA\Desktop\r_eye'
N= 40;    %number of  images
NT= 1;    %number of training set
k=1; S=[];    % img matrix
for i=1:N
for j=1:10
if(j<=NT)
```

```
idx=(i-1)*10+j ;
str=strcat('r_eye',int2str (idx ),'.bmp'); % concatenates two strings that form the name of the image
X=double(imread(str))/255.0;  img=X;
[irow icol]=size(img); % get the number of rows (N1) and columns (N2)
temp=reshape(img', irow*icol ,1);    % creates a (N1*N2)x1 vector
S=[S temp];
k=k+1; end end  end
% mean image
m=mean(S,2);  % obtains the mean of each row instead of each column
tmimg=(m); % converts to unsigned 8-bit integer. Values range from 0 to 255
img=reshape(tmimg,icol,irow); % takes the N1*N2x1 vector and creates a N1xN2 matrix
img=img';
for i=1:N*NT
dbx(:,i)=double(S(:,i)-m); end
%Covariance matrix C=A'A, L=AA'
A=dbx';
L=A*A';
[vv dd]=eig(L);

% Sort and eliminate those whose eigenvalue is zero
v=[];   d=[];
for i=1:size(vv,2)
if(dd(i,i)>1e-4)
v=[v vv(:,i)];
d=[d dd(i,i)]; end end
%sort, will return an ascending sequence
[B index]=sort(d);
ind=zeros(size(index));
dtemp=zeros(size(index));
vtemp=zeros(size(v));
len=length(index);
for i=1:len
dtemp(i)=B(len+1-i);
ind(i)=len+1-index(i);
vtemp(:,ind(i))=v(:,i); end
d=dtemp;
v=vtemp;
%Eigenvectors of C matrix
u=[];
for i=1:size(v,2)
temp=sqrt(d(i));
u=[u (dbx*v(:,i))./temp]; end

% Find the weight of each face in the training set
omega = [];
for h=1:size(dbx,2)
WW=[];
for i=1:size(u,2)
```

```
t = u(:,i)';
WeightOfImage = dot(t,dbx(:,h)');
WW = [WW; WeightOfImage]; end
omega = [omega WW]; end

%%%%%%%%%%%%%%% test phase %%%%%%

cd 'C:\Users\TOSHIBA\Desktop\r_eye'
k=0; count=0 ;kk=0;
T=[]; % img matrix
for i=1:N
for j=1:10
if(j>NT)
idx=(i-1)*10+j ; count=count+1;
str=strcat('r_eye',int2str (idx ),'.bmp'); % concatenates two strings that form the name
of the image
X=double(imread(str))/255.0;  img=X;
[irow icol]=size(img); % get the number of rows (N1) and columns (N2)
temp=reshape(img',irow*icol,1); % creates a (N1*N2)x1 vector
temp=double(temp); %InImage;
me=mean(temp);
NormImage = temp;
Difference = temp-m;
InImWeight=[];
for ii=1:size(u,2)
t = u(:,ii)';
WeightOfInputImage=dot(t,Difference');
InImWeight=[InImWeight; WeightOfInputImage]; end

% Find Euclidean distance
e=[];
for iii=1:size(omega,2)
q=omega(:,iii);
DiffWeight=InImWeight-q;
mag=norm(DiffWeight);
e=[e mag]; end
[MinimumValue minidx]=min(e);    % minimum eucledian distance
%i
%minidx
outidx=floor((minidx-1)/(NT))+1;
if(i==outidx)
k=k+1; else
minidx
kk=kk+1; end  end end end
performance=k/count*100

RESULT :  performance =47.5000        **//  NT=1
```

majority

```
face_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result1.mat' RESULT
Reye_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result2.mat' RESULT
leye_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result3.mat' RESULT
Nnose_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result4.mat' RESULT
mouths_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result5.mat' RESULT
clear all
load result1.mat
MRESULT=RESULT;
load result2.mat
MRESULT(:,3)=RESULT(:,2);
load result3.mat
MRESULT(:,4)=RESULT(:,2);
load result4.mat
MRESULT(:,5)=RESULT(:,2);
load result5.mat
MRESULT(:,6)=RESULT(:,2);
s=size(MRESULT,1);
k=0;

for i=1:s
a=zeros(1,40);
a(MRESULT(i,2))=a(MRESULT(i,2))+1;
a(MRESULT(i,3))=a(MRESULT(i,3))+1;
a(MRESULT(i,4))=a(MRESULT(i,4))+1;
a(MRESULT(i,5))=a(MRESULT(i,5))+1;
a(MRESULT(i,6))=a(MRESULT(i,6))+1;
 [maxmv idxmv ]=max(a);
 MRESULT(i,7)=idxmv ;
 if(MRESULT(i,1)==MRESULT(i,7))
 k=k+1;    end     end
 performanceMV=k/s*100

>> majority
performance =  90
performance = 76.5000
performance = 79
performance =76.5000
performance = 74.5000
performanceMV =  94.5000
```

sumproduct

```
face_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result1.mat' RESULT
Reye_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result2.mat' RESULT
leye_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result3.mat' RESULT
Nnose_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result4.mat' RESULT
mouths_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result5.mat' RESULT
clear all
load result1.mat
MRESULT=RESULT;
load result2.mat
MRESULT(:,4)=RESULT(:,2);
MRESULT(:,5)=RESULT(:,3);
load result3.mat
MRESULT(:,6)=RESULT(:,2);
MRESULT(:,7)=RESULT(:,3);

load result4.mat
MRESULT(:,8)=RESULT(:,2);
MRESULT(:,9)=RESULT(:,3);
load result5.mat
MRESULT(:,10)=RESULT(:,2);
MRESULT(:,11)=RESULT(:,3);
s=size(MRESULT,1); k=0;

for i=1:s
a=zeros(1,40);
a(MRESULT(i,2))=a(MRESULT(i,2))+MRESULT(i,3);
a(MRESULT(i,4))=a(MRESULT(i,4))+MRESULT(i,5);
a(MRESULT(i,6))=a(MRESULT(i,6))+MRESULT(i,7);
a(MRESULT(i,8))=a(MRESULT(i,8))+MRESULT(i,9);
a(MRESULT(i,10))=a(MRESULT(i,10))+MRESULT(i,11);
[maxsum idxsum ]=max(a);
MRESULT(i,12)=idxsum;
if(MRESULT(i,1)==MRESULT(i,12))
k=k+1; end end
performanceSUM=k/s*100
s=size(MRESULT,1);
```

```
k=0; for i=1:s
a=ones(1,40);
a(MRESULT(i,2))=a(MRESULT(i,2))*MRESULT(i,3);
a(MRESULT(i,4))=a(MRESULT(i,4))*MRESULT(i,5);
a(MRESULT(i,6))=a(MRESULT(i,6))*MRESULT(i,7);
a(MRESULT(i,8))=a(MRESULT(i,8))*MRESULT(i,9);
a(MRESULT(i,10))=a(MRESULT(i,10))*MRESULT(i,11);
[maxproduct idxproduct ]=max(abs(a-1));
MRESULT(i,13)=idxproduct;
if(MRESULT(i,1)==MRESULT(i,13))
k=k+1; end end

performancePRODUCT=k/s*100
MRESULT:
>> sumproduct
performance = 90
performance =76.5000
performance = 79
performance =76.5000
performance = 74.5000
performanceSUM =92.5000
performancePRODUCT = 93.5000
```

---

majorityFaceXXX

```
path(path, 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400')
face_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result1.mat' RESULT
faceLE
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result2.mat' RESULT
faceRE
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result3.mat' RESULT
faceNose
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result4.mat' RESULT
faceMouth
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result5.mat' RESULT
clear all
load result1.mat
MRESULT=RESULT;
load result2.mat
MRESULT(:,3)=RESULT(:,2);
load result3.mat
MRESULT(:,4)=RESULT(:,2);
load result4.mat
```

```
MRESULT(:,5)=RESULT(:,2);
load result5.mat
MRESULT(:,6)=RESULT(:,2);
s=size(MRESULT,1);
k=0;
for i=1:s
a=zeros(1,40);
a(MRESULT(i,2))=a(MRESULT(i,2))+1;
a(MRESULT(i,3))=a(MRESULT(i,3))+1;
a(MRESULT(i,4))=a(MRESULT(i,4))+1;
a(MRESULT(i,5))=a(MRESULT(i,5))+1;
a(MRESULT(i,6))=a(MRESULT(i,6))+1;
[maxmv idxmv ]=max(a);
MRESULT(i,7)=idxmv ;
if(MRESULT(i,1)==MRESULT(i,7))
k=k+1;
end end
performanceMV=k/s*100
```

RESULT:

```
>> majorityFaceXXX
performance = 90
performance =93.5000
performance = 91.5000
performance = 93.5000
performance = 90
performanceMV = 95
```

sumproductXXX

```
face_Recog_tek
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result1.mat' RESULT faceLE

cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result2.mat' RESULT
faceRE
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result3.mat' RESULT
faceNose
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result4.mat' RESULT
faceMouth
cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400\majority_voting'
save 'result5.mat' RESULT
clear all
load result1.mat
MRESULT=RESULT;
load result2.mat
```

```
MRESULT(:,4)=RESULT(:,2);
MRESULT(:,5)=RESULT(:,3);
load result3.mat
MRESULT(:,6)=RESULT(:,2);
MRESULT(:,7)=RESULT(:,3);
load result4.mat
MRESULT(:,8)=RESULT(:,2);
MRESULT(:,9)=RESULT(:,3);
load result5.mat
MRESULT(:,10)=RESULT(:,2);
MRESULT(:,11)=RESULT(:,3);
s=size(MRESULT,1);
k=0;
for i=1:s
a=zeros(1,40);
a(MRESULT(i,2))=a(MRESULT(i,2))+MRESULT(i,3);
a(MRESULT(i,4))=a(MRESULT(i,4))+MRESULT(i,5);
a(MRESULT(i,6))=a(MRESULT(i,6))+MRESULT(i,7);
a(MRESULT(i,8))=a(MRESULT(i,8))+MRESULT(i,9);
a(MRESULT(i,10))=a(MRESULT(i,10))+MRESULT(i,11);
[maxsum idxsum ]=max(a);
MRESULT(i,12)=idxsum;
if(MRESULT(i,1)==MRESULT(i,12))
k=k+1;    end   end
performanceSUM=k/s*100
s=size(MRESULT,1);
k=0;
for i=1:s
a=ones(1,40);
a(MRESULT(i,2))=a(MRESULT(i,2))*MRESULT(i,3);
a(MRESULT(i,4))=a(MRESULT(i,4))*MRESULT(i,5);
a(MRESULT(i,6))=a(MRESULT(i,6))*MRESULT(i,7);
a(MRESULT(i,8))=a(MRESULT(i,8))*MRESULT(i,9);
a(MRESULT(i,10))=a(MRESULT(i,10))*MRESULT(i,11);
[maxproduct idxproduct ]=max(abs(a-1));
MRESULT(i,13)=idxproduct;
if(MRESULT(i,1)==MRESULT(i,13))
k=k+1; end end
performancePRODUCT=k/s*100
```

RESULT :

```
>> sumproductXXX
performance = 90
performance = 93.5000
performance = 91.5000
performance = 93.5000
performance = 90
performanceSUM = 94.5000
performancePRODUCT =  90
```

faceMouth

```
clear all    close all    clc
path(path, 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400')
path(path, 'C:\Users\TOSHIBA\Desktop\mouths')
N= 40;    %number of  images
NT=5;    %number of training set
k=1;
S=[]; S2=[];    % img matrix
for i=1:N
for j=1:10
if(j<=NT)
idx=(i-1)*10+j ;
str=strcat(int2str (idx ),'.pgm'); % concatenates two strings that form the name of the
image
str2=strcat('mouth',int2str (idx ),'.bmp');
X=double(imread(str))/255.0;  img=X;


X2=double(imread(str2))/255.0;  img2=X2;
[irow icol]=size(img); % get the number of rows (N1) and columns (N2)
[irow2 icol2]=size(img2);
temp=reshape(img', irow*icol ,1);    % creates a (N1*N2)x1 vector
temp2=reshape(img2', irow2*icol2 ,1);
S=[S temp];
S2=[S2 temp2];
k=k+1; end end end

% mean image
m=mean(S,2);  % obtains the mean of each row instead of each column
m2=mean(S2,2);
tmimg=(m); % converts to unsigned 8-bit integer. Values range from 0 to 255
tmimg2=(m2);
img=reshape(tmimg,icol,irow); % takes the N1*N2x1 vector and creates a N1xN2
matrix
img2=reshape(tmimg2,icol2,irow2);
img=img';
img2=img2';
 for i=1:N*NT
dbx(:,i)=double(S(:,i)-m);
dbx2(:,i)=double(S2(:,i)-m2); end
 %Covariance matrix C=A'A, L=AA'
A=dbx';  A2=dbx2';
L=A*A';  L2=A2*A2';
 [vv dd]=eig(L);  [vv2 dd2]=eig(L2);
% Sort and eliminate those whose eigenvalue is zero
v=[];  v2=[];
d=[];  d2=[];
for i=1:size(vv,2)
```

```
if(dd(i,i)>1e-4)
v=[v vv(:,i)];
d=[d dd(i,i)];   end
if(dd2(i,i)>1e-4)
v2=[v2 vv2(:,i)];
d2=[d2 dd2(i,i)];    end   end
 %sort, will return an ascending sequence
[B index]=sort(d);  [B2 index2]=sort(d2);
ind=zeros(size(index));  ind2=zeros(size(index2));
dtemp=zeros(size(index));   dtemp2=zeros(size(index2));
vtemp=zeros(size(v));     vtemp2=zeros(size(v2));
len=length(index);  len2=length(index2);
for i=1:len
dtemp(i)=B(len+1-i);
ind(i)=len+1-index(i);
vtemp(:,ind(i))=v(:,i); end
for i=1:len2
dtemp2(i)=B2(len2+1-i);
ind2(i)=len2+1-index2(i);

vtemp2(:,ind2(i))=v2(:,i); end
d=dtemp;  d2=dtemp2;
 v=vtemp; v2=vtemp2;

%Eigenvectors of C matrix
u=[];  u2=[];
for i=1:size(v,2)
temp=sqrt(d(i));
u=[u (dbx*v(:,i))./temp]; end
for i=1:size(v2,2)
temp2=sqrt(d2(i));
u2=[u2 (dbx2*v2(:,i))./temp2]; end

% Find the weight  of each face in the training set
omega = [];  omega2 = [];
for h=1:size(dbx,2)
WW=[];
for i=1:size(u,2)
t = u(:,i)';
WeightOfImage = dot(t,dbx(:,h)');
WW = [WW; WeightOfImage]; end
omega = [omega WW]; end
for h=1:size(dbx2,2)
WW2=[];
for i=1:size(u2,2)
t2 = u2(:,i)';

WeightOfImage2 = dot(t2,dbx2(:,h)');
WW2 = [WW2; WeightOfImage2]; end
omega2 = [omega2 WW2]; end
```

```
%%%%%%%%%%%%%%% test phase %%%%%%

cd 'C:\Users\TOSHIBA\Desktop\att_faces\orl_faces\f1-400'
k=0; count=0 ;   kk=0;
for i=1:N
for j=1:10
if(j>NT)
idx=(i-1)*10+j ; count=count+1;
str=strcat(int2str (idx ),'.pgm'); % concatenates two strings that form the name of the
image
str2=strcat('mouth',int2str (idx ),'.bmp');
X=double(imread(str))/255.0;  img=X;
X2=double(imread(str2))/255.0;  img2=X2;
[irow icol]=size(img); % get the number of rows (N1) and columns (N2)
[irow2 icol2]=size(img2);
temp=reshape(img',irow*icol,1); % creates a (N1*N2)x1 vector
temp2=reshape(img2',irow2*icol2,1);
temp=double(temp); %InImage;
temp2=double(temp2);
me=mean(temp);  me2=mean(temp2);
NormImage = temp;  NormImage2 = temp2;
Difference = temp-m;      Difference2 = temp2-m2;
InImWeight=[];InImWeight2=[];
for ii=1:size(u,2)
t = u(:,ii)';
WeightOfInputImage=dot(t,Difference');
InImWeight=[InImWeight; WeightOfInputImage]; end
for ii=1:size(u2,2)
t2 = u2(:,ii)';
WeightOfInputImage2=dot(t2,Difference2');
InImWeight2=[InImWeight2; WeightOfInputImage2]; end
% Find Euclidean distance
e=[];  e2=[];
for iii=1:size(omega,2)
q=omega(:,iii);  q2=omega2(:,iii);
DiffWeight=InImWeight-q;   DiffWeight2=InImWeight2-q2;
DiffWeight=DiffWeight/4.46;

DiffWeight=[DiffWeight' DiffWeight2'];
mag=norm(DiffWeight);
e=[e mag]; end

mag2=norm(InImWeight-0);
[MinimumValue minidx]=min(e);    % minimum eucledian distance

%minidx
outidx=floor((minidx-1)/(NT))+1;
RESULT(count,1)=i;
RESULT(count,2)=outidx;
```

```
RESULT(count,3)=1-MinimumValue/(2*mag2);
if(i==outidx)
k=k+1;
else
kk=kk+1;
end end end end
performance=k/count*100
RESULT :  performance = 90
```