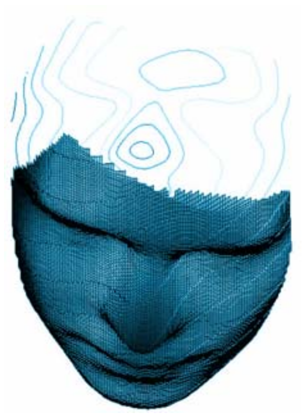
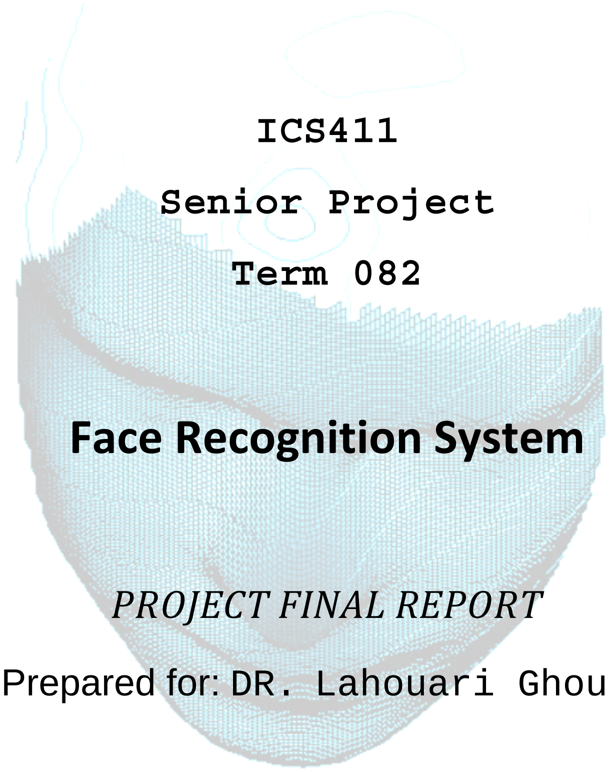
Face Recognition System



KING FAHAD UNIVERSITY OF PETROLEUM AND MINERALS

College of Computer Science and Engineering



ICS411

Senior Project

Term 082

Face Recognition System

PROJECT FINAL REPORT

Prepared for: DR. Lahouari Ghouti

Noor AlSheala	*224236*
Hasan AlOdail	*235287*

Table of Contents

ILLUSTRATIONS

INTRODUCTION

Humans have been using physical characteristics such as face, voice, gait, etc. to recognize each other for thousands of years. With new advances in technology, biometrics has become an emerging technology for recognizing individuals using their biological traits. Now, biometrics is becoming part of day to day life, where in a person is recognized by his/her personal biological characteristics. Examples of different Biometric systems include Fingerprint recognition, Face recognition, Iris recognition, Retina recognition, Hand geometry, Voice recognition, Signature recognition, among others. Face recognition, in particular has received a considerable attention in recent years both from the industry and the research community.

The objective of our project is to create a C# code that can be used to identify people using their face images.

This report gives a brief background about biometrics. A particular attention is given to face recognition. Face recognition refers to an automated or semi-automated process of matching facial images. Many techniques are available to apply face recognition of which is Principle Component Analysis (PCA). PCA is a way of identifying patterns in data and expressing the data in such a way to highlight their similarities and differences. Before applying this method to face recognition, a brief introduction is given for PCA from mathematical point of view. The last part of the report gives an explanation about the implementation of PCA in C#. To understand how we can use our face recognition system, a description of the Graphical User Interface is given.

1. BACKGROUND

1.1 Biometrics

Biometrics is the automated recognition of individuals based on their behavioral or physiological characteristics [4]. The physiological characteristics are related to the shape of the body. The most common example is fingerprint. Other examples include face recognition, hand geometry and iris recognition. The behavioral characteristics are related to the behavior of a person. Signature is one example of these characteristics which is still widely used today. Modern approaches are the study of keystroke dynamics and voice. Figure 1 summarizes the two types of biometrics [7].

Abstract

This report gives a brief background about biometrics. A particular attention is given to face recognition. Face recognition refers to an automated or semi-automated process of matching facial images. Many techniques are available to apply face recognition of which is Principle Component Analysis (PCA). PCA is a way of identifying patterns in data and expressing the data in such a way to highlight their similarities and differences. Before applying this method to face recognition, a brief introduction is given for PCA from mathematical point of view. The last part of the report gives an explanation about the implementation of PCA in C#. To understand how we can use our face recognition system, a description of the Graphical User Interface is given.

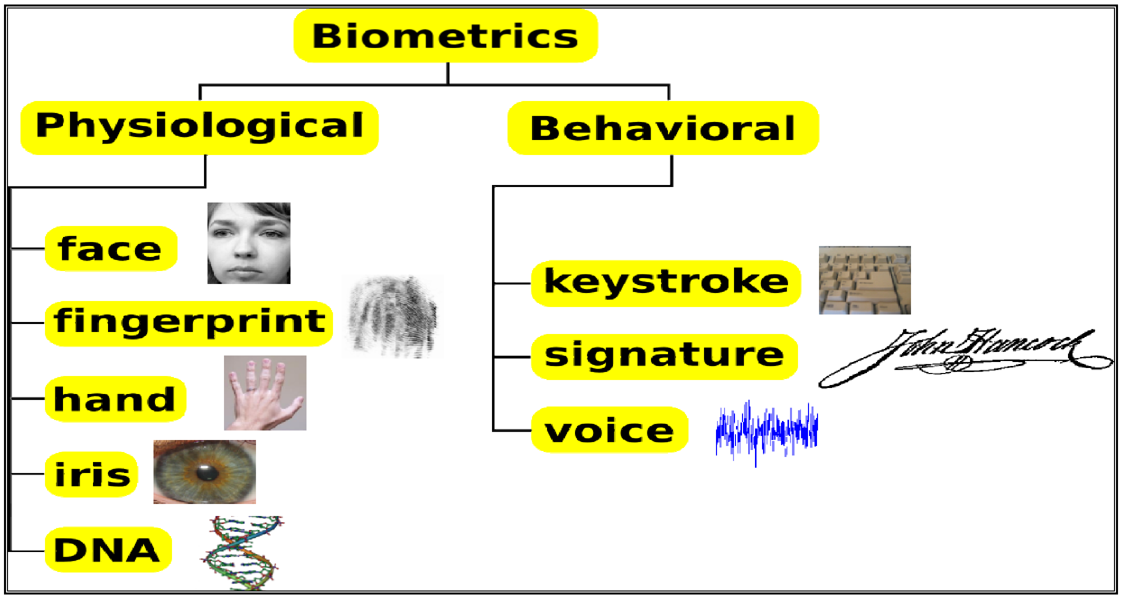


Figure 1: Biometrics Types

1.2 Biometric Characteristics

A number of biometric characteristics exist and are in use in various applications, each biometric has its strengths and weaknesses, and the choice depends on the application. No single biometric is expected to effectively meet the requirements of all the applications. In other words, no biometric is optimal. Any human physiological and/or behavioral characteristic can be used as a biometric characteristic as long as it satisfies the following requirements:

1. **Universality**: each and every person throughout the world should have a biometric characteristic on the basis of which he/she could be represented, for example every person has a thumb impression which is unique with him.

2. **Distinctiveness**: any two persons should be sufficiently different in terms of a biometric characteristic measure. Taking the above example any two persons will have different thumb impression.

3. **Collect-ability**: the biometric characteristic can be measured quantitatively with an ease. For example taking a snap of face is typically easy with a camera, where as it is difficult to take a retina sample of a person, but it is possible.

4. **Permanence**: the characteristic should be sufficiently invariant (with respect to the matching criterion) over a period of time. Taking an example of face recognition as a person grows in age his or her face changes over a time.

5. **Performance**: It refers to the achievable recognition accuracy and speed, the resources required to achieve the desired recognition accuracy and speed, as well as the operational and environmental factors that affect the accuracy and speed.

6. **Acceptability**: It indicates the extent to which people are willing to accept the use of a particular biometric identifier (characteristic) in their daily lives.

7. **Circumvention**: It reflects how easily the system can be fooled using fraudulent methods or could be avoided. For example in voice recognition any person's voice can be recorded and the biometric system can easily be fooled .[1]

The following table shows the most well known biometric characteristics [7]:

Table 1: Well Known Biometrics Characteristics

Biometric characteristic	Description of the features
Fingerprint	Finger lines, pore structure
Signature (dynamic)	Writing with pressure and speed differentials
Facial geometry	Distance of specific facial features (eyes, nose, mouth)
Iris	Iris pattern
Retina	Eye background (pattern of the vein structure)
Hand geometry	Measurement of fingers and palm

Voice	Tone or timbre
DNA	DNA code as the carrier of human hereditary
Keyboard strokes	Rhythm of keyboard strokes (PC or other keyboard)

1.3 Applications of Biometrics

The applications of biometrics can be divided into the following three main groups:

1. **Commercial applications:** such as computer network login, electronic data security, e-commerce, Internet access, ATM, credit card, physical access control, cellular phone, PDA, medical records management, distance learning, etc.

2. **Government applications:** such as national ID card, correctional facility, driver's license, social security, welfare-disbursement, border control, passport control, etc

3. **Forensic applications:** such as corpse identification, criminal investigation, terrorist identification, parenthood determination, missing children, etc can be carried out using biometric systems. [1]

1.4 A Complete Biometric System.

In this part, we will briefly discuses a complete biometric system which might include a hardware part such as camera to scan or picture a user. Due time limitation and proposed scope, the team implements only the software part, where the test face will be enter from a bitmap or PNM pictures.

In Figure 2 below shows a simple block diagram of a biometric system. The first block (Sensor, if any) is the interface between the real world and our system; it has to acquire all the necessary data. The second block performs all necessary pre-processing. In the third block, features needed are extracted and a template is generated. A template is a synthesis of all the characteristics extracted from the source, in the optimal size to allow for adequate identifiably. After creating a template, we have either an enrollment process or a recognition process. If enrollment is being performed, the template is simply stored somewhere within a database. If recognition is being performed, the obtained template is passed to a matcher that compares it with other exiting templates.

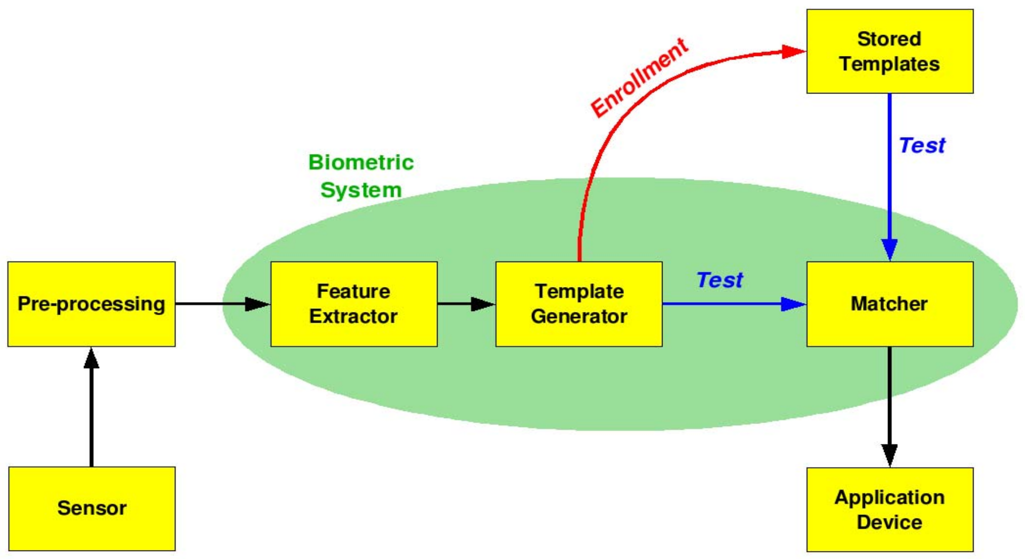


Figure 2: A Complete Biometric System

Due to the statistical nature of biometric samples, there is no exact match possible. For this reason, the decision process will confirm recognition only if the comparison score exceeds an adjustable value. The matching phase is usually done using software. The matching program will analyzes the template with the input. This will then output for any specified purpose. [7]

A complete Biometric system comprises of both hardware and software; the hardware collects the data, and the software interprets the data and evaluates acceptability. For example, you stand in front of a camera so it can capture face or eye features. The system then extracts the appropriate features from the scan and stores the data as a template (feature vector). You then interact with the biometric device again, and the system verifies whether the data corresponds to the stored template or not. If the software fails to get a match, more tries may be needed. Once this procedure is complete, the system is operational (i.e. training complete).

When a user tries to access the system, the user characteristics are scanned by whatever device being used (In our system, a test face will be fid manually to the system), and the hardware passes the data to the software, which checks the user templates. If there is a match, the user is given access; otherwise, a message reports that the system can't identify the user. [1]

2. FACE RECOGNITION

Face recognition has received considerable interest as a widely accepted biometric, because of the ease of collecting samples of a person, with or without subject's intension [1]. Face recognition refers to an automated or semi-automated process of matching facial images. This type of technology constitutes a wide group of technologies which all work with face but use different scanning techniques. Most common by far is 2D face recognition which is easier and less expensive compared to the other approaches. [2]

2.1 Face Recognition Process

There are four steps in face recognition process:

1. **Acquiring a sample:** In a complete, full implemented biometric system, a sensor takes an observation. The sensor might be a camera and the observation is a snapshot picture. In our system, a sensor will be ignored, and a 2D face picture "observation" will supplied manually.

2. **Extracting Features:** For this step, the relevant data is extracted from the predefined captured sample. This is can be done by the use of software where many algorithms are available. The outcome of this step is a biometric template which is a reduced set of data that represents the unique features of the enrolled user's face.

3. **Comparison Templates:** This depends on the application at hand. For identification purposes, this step will be a comparison between a given picture for the subject and all the biometric templates stored on a database. For verification, the biometric template of the claimed identity will be retrieved (either from a database or a storage medium presented by the subject) and this will be compared to a given picture.

4. **Declaring a Match:** The face recognition system will return a candidate match list of potential matches. In this case, the intervention of a human operator will be required in order to select the best fit from the candidate list. An illustrative analogy is that of a walk-through metal detector, where if a person causes the

detector to beep, a human operator steps in and checks the person manually or with a hand-held detector. [2]

2.2 Face Recognition Techniques

All available face recognition techniques can be classified into four categories based on the way they represent face;

1. Appearance based which uses holistic texture features.
2. Model based which employ shape and texture of the face, along with 3D depth information.
3. Template based face recognition.
4. Techniques using Neural Networks.

Figure 3 summaries these types. [1]

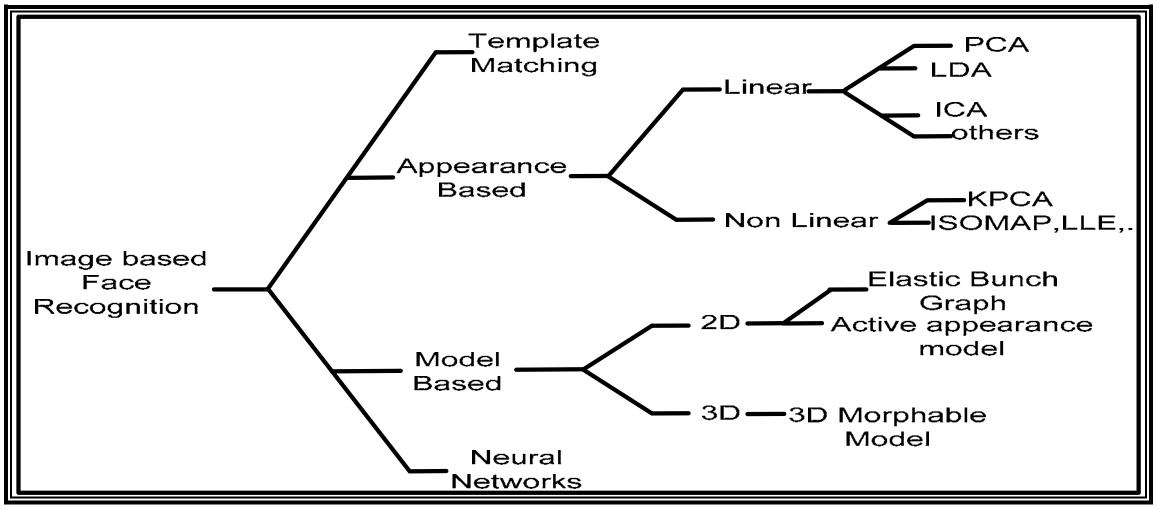


Figure 3: Classification of Face Recognition Methods

3. PRINCIPLE COMPONENT ANALYSIS (PCA)

PCA is a way of identifying patterns in data and expressing the data in such a way to highlight their similarities and differences [6]. The purpose of PCA is to reduce the large dimensionality of the data space (observed variables) to smaller intrinsic dimensionality of feature space (independent variables) which are needed to describe the data economically. [3]

3.1. Mathematical Theory

To perform a PCA on a set of data, the following steps are needed:

Step 1: Get some data.

Step 2: Subtract the mean: take the average across each dimension. This produces a data set whose mean is zero.

Step 3: calculate the covariance matrix.

The aim of finding the covariance is to see if there is any relationship between the data dimensions. Covariance is always measured between two dimensions matrix. If we have a data set with more than one matrix of dimensions 2, there result are more than one covariance measurements that can be calculated. The most practical way to get all the possible covariance values between the different dimensions is to calculate them all and put them in one big 2D matrix. The definition for the covariance matrix for a set of data with n dimensions is:

$$C^{n\times n} = \left(c_{i,j}, c_{i,j} = cov\left(Dim_i, Dim_j\right)\right) \qquad (1)$$

Where; $C^{n\times n}$ is a matrix with n row and n columns and Dim_x is the xth dimension.

This formula tells you that if you have an n-dimensional data set, then the matrix has n rows and n columns (so it is square) and each entry in the matrix is the result of calculating the covariance between two separate dimensions. From the static, we know that the formula for the covariance is given by:

$$cov\,(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)} \qquad (2)$$

This says, "For each data items, multiply the difference between the X value and the mean of X $(X_i - \bar{X})$, by the difference between the Y value and the mean of Y $(Y_i - \bar{Y})$. Add all these up and divide by the total number of data minus one $(n-1)$".

Step 4: Calculate the Eigenvectors and Eigenvalues of the covariance matrix.

Eigenvectors and eigenvalues always come in pairs. The eigenvector can only be found for square matrices and not every square matrix has eigenvectors. For a given a $n \times n$ matrix that does have eigenvectors, there most be n of eigenvectors with their n corresponding eigenvalues. All the eigenvectors of a matrix are perpendicular, no matter how many dimensions you have. This is important because it means that you can express the data in terms of these perpendicular eigenvectors. Finding eigenvectors is unfortunately not an easy task unless you have a rather small matrix, like no bigger than 3×3. After that the usual way to find the eigenvectors is by some complicated iterative method that usually done through computer programs. The process of taking the eigenvectors of the covariance matrix is intended to extract lines that characterize the data set.

Step 5: Choosing components and forming a feature vector.

Once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalues highest to lowest. This gives you the components in order of significance. In fact, it turns out that the eigenvector with the highest eigenvalue is the principle component of the data set. Now, if you like, you can decide to ignore the components of lesser significance. You do loss some information, but if the eigenvalues are small you do not loss much. For example, if you have original n dimensions in your data, and so calculated n eigenvectors, and then you choose only the first k eigenvectors, then the final data set has k dimensions. Now a feature vector is constructed by taking the eigenvectors that you want to keep from the list of eigenvectors and forming a matrix with these eigenvectors in the columns. [6] Basically, the feature vector is the selected k eigenvectors; in our code we name it the Eigenface to be compatible with our understanding.

$$Feature\ Vector = (eig_1\ eig_2 eig_3\ \ldots\ldots\ldots eig_n) \tag{3}$$

3.2. PCA-Based-Face Recognition

3.2.1. Main Idea

The main idea of using PCA for face recognition is to express the large 1-D vector of pixels constructed from 2-D facial image into the compact principle components of feature space. This is can be called eigenspace projection.

Using PCA, we find a subset of principle components in a set of training faces. Then, we project faces into this principle components space and get Eigenface vectors. Comparison is performed by calculating the distance between these vectors. Usually comparison of faces is performed by calculating the Euclidean distance between these feature vectors but other measures are possible. [5]

After we perform a PCA, our original data is expressed in terms of eigenvectors found from the covariance matrix. This is useful because say we want to do facial recognition and so our original images were of people faces. Then, the problem is if a new image is given, whose face from the original set is face's owner? (Note that new image is not one of original images). The way this is done is to measure the difference between the new image and original images, but not along the original axes, along the new axes derived from the PCA analysis. It turns out that these axes work much better for recognition faces, because the PCA analysis has given us the original images in terms of differences and similarities between them. PCA has identified the statistical patterns in the data. [6]

3.2.2. Mathematical Implementations of PCA

The starting point in our analysis is to have a 2D image size (N x M) (face photography). Then, a 1D image size ((N*M) x 1) is formed by scanning all the elements of the 2D image row by row and writing them to the column-vector. Let X-j be N-element one-dimensional image and suppose that we have r images such as $(j = 1,2, \dots . r)$. Then, the mean vector, centered data vectors and the covariance matrix are calculated as follows:

$$m = \frac{1}{r} \sum_{j=1}^{r} Xj \qquad (4)$$

$$dj = Xj - m \qquad (5)$$

$$C = \frac{1}{r} \sum_{j=1}^{r} dj dj^T \qquad (6)$$

In order to perform PCA, it is necessary to find eigenvectors U_k and eigenvalues λ_k of the covariance matrix $C.$ Because of the dimensionality (N^2) of the matrix C is large, even for a small image, and computations of eigenvectors using traditional methods is complicated, dimensionality of matrix C is reduced using the decomposition. The Found eigenvectors $u = (u_1, u_2, u_3, \ldots \ldots, u_N)^T$ are normalized and sorted in decreasing order according to the corresponding eigenvalues.

Then, these vectors are transposed and arranged to form the row-vectors of the transformation matrix T. Now any data X can be projected into the eigenspace using the following formula:

$$Y = T(X - m) \qquad (7)$$

Note that, for projection into eigenspace we can choose not use all found eigenvectors but only a few of them, corresponding to the largest eigenvalues. When the image (human face photograph) is projected into the eigenspace, we get its eigenvector $Z = (z_1, z_2, z_3, \ldots \ldots, z_n)^T$ here n is the number of features chosen. #When we have feature vector Z for each face, identification can be performed. After projecting a new unknown face image into the eigenspace, we get its feature vector Z_{new} and calculate the Euclidean distance between unknown face and the known n faces $\varepsilon_i = \|Z_{new} - Z_i\|$ and say that the face with projection Z_{new} belongs to a person s that have min(ε_i). For rejection of unknown faces a threshold τ is chosen and it is said that the face with projection Z_{new} is unknown if $\varepsilon_s \geq \tau$. [5]

4. PROJECT REQUIREMENTS

4.1 Glossary of the Technical Terms

Term	Definition
PCA	The Principal Component Analysis, it is a techniques used in image recognition and compression. PCA express a large 1-D vector of pixels constructed from 2-D facial image into the compact principal components of the feature space.
Filtering	Basic term in image processing. Process like: resize image, set grayscale, removing the noise that affects the quality of the image … etc.

4.2 User Requirements

1. Functional requirements
    - **a.** The user should be able to enter records.
    - **b.** Each record represents information about a person and contains an image of his/her face.
    - **c.** Records may consists of:
        - **i.** First name.
        - **ii.** Last name.
        - **iii.** Phone.
        - **iv.** Address (city-street address).
        - **v.** Face image(s).
        - **vi.** ID-number.
    - **d.** The system must be able to take face image as an input and search for a matching face in the database, and then show the results.
    - **e.** The results should be viewed by showing the faces matches the input, and display the most possible image first, in numerical order.
    - **f.** The user should be able to chose (click on) the result that he/she wants and view the full record's information of the person.
    - **g.** The system can do noisy filter on images.
2. Non Functional Requirements
    - **a.** Records are maintained in a database.
    - **b.** Every record shall be allocated a unique identifier (id-number).
    - **c.** User should be able to retrieve data by entering id or name.
    - **d.** The system should provide a control for the user over the result. Means, the system mustn't decide what the right match is. Instead, the system should display records above 60% match.
    - **e.** The results may be shown on a box where it has scroll bar activated when the space of the box is not enough.
    - **f.** In a case the user choose to show all result, the system should show N-item in a box, where the user can forward to get next result.
    - **g.** The system should be able to handle images of all formats.
    - **h.** The images should be pre-processed by ignoring the irrelevant parts, means localizing and normalize the face.
    - **i.** The face should be localized by detecting inner and outer boundaries, background must be ignored.
    - **j.** To improve the performance, the database should include a transformed image (template) by PCA algorithm for each record, i.e. the template should be ready when it's needed.
    - **k.** The database should be developed using Microsoft SQL Server.

4.3 System Requirements

4.3.1 Actors with Their Description

Users: Provides the system with an image, expects the system to show all matches and details about them.

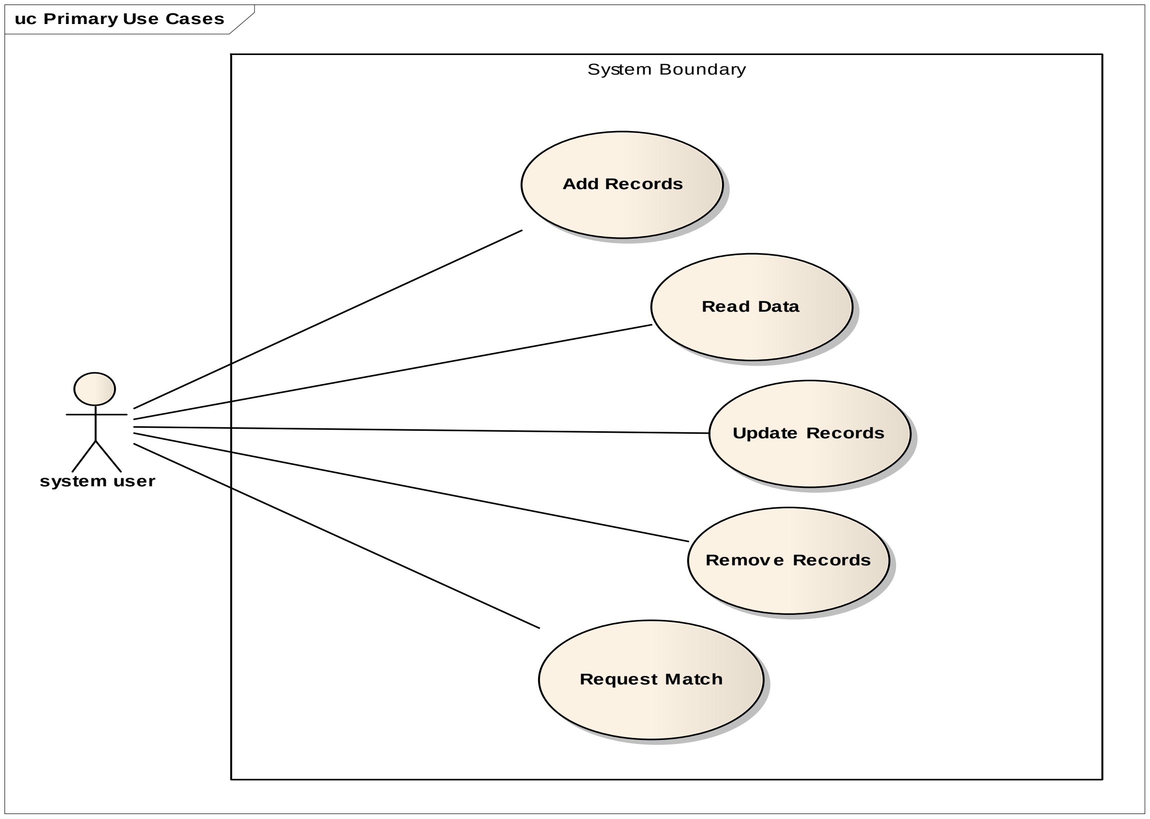
4.3.2 Use Cases with Their Description

Use Case	Description
Enter Image	The user enters an image to the system, and it should be able to handle images of all formats.
Read date	The system reads the data from the SQL server and applies the PCA process to extract the PCA feature.
Request Match	The user requests an image to be matched with an image in the database and retrieve details about it (if possible).
Display all Matches	In a case the user choose to show all results to be displayed, the system should show N-item in a box, where the user can forward to get next result and picks the image of interest.
Add/update Records	Allows the user to add or update (remove) the records in the system, such as name, ID, phone number, etc..

4.3.3 Flow of Events

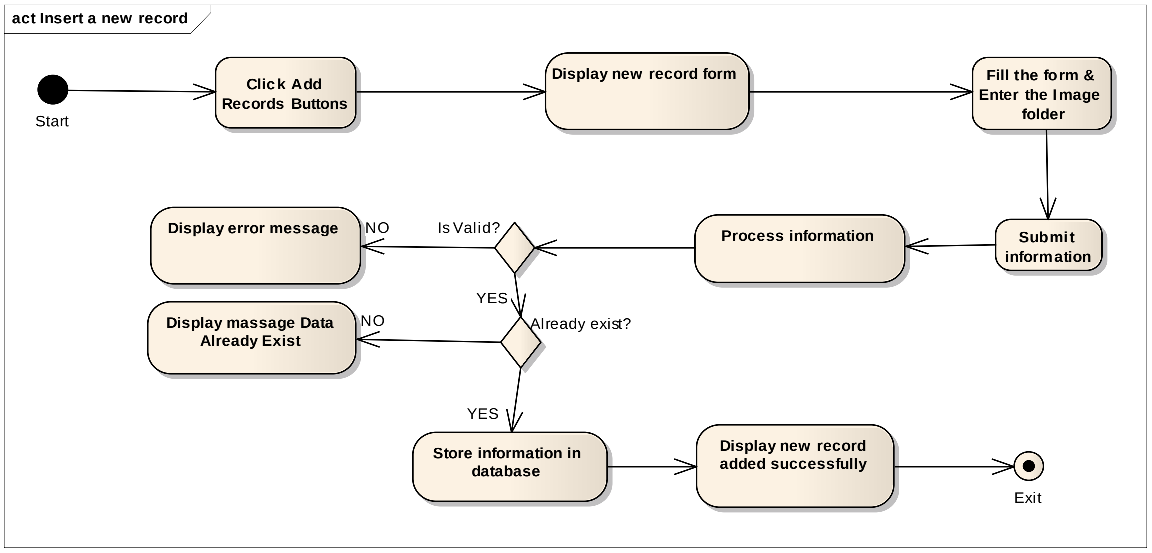
1. Preconditions: The system must be fully functional and connected to the Database.
2. Main flow of events:
   a. The user enters an image.
   b. The user filters the image (Optional).
   c. The user requests a match, from the database.
   d. The system will search the database and return all possible matches.
3. Post conditions: A set of matching images with relevant details are displayed.

4.3.4 Use Case Diagram

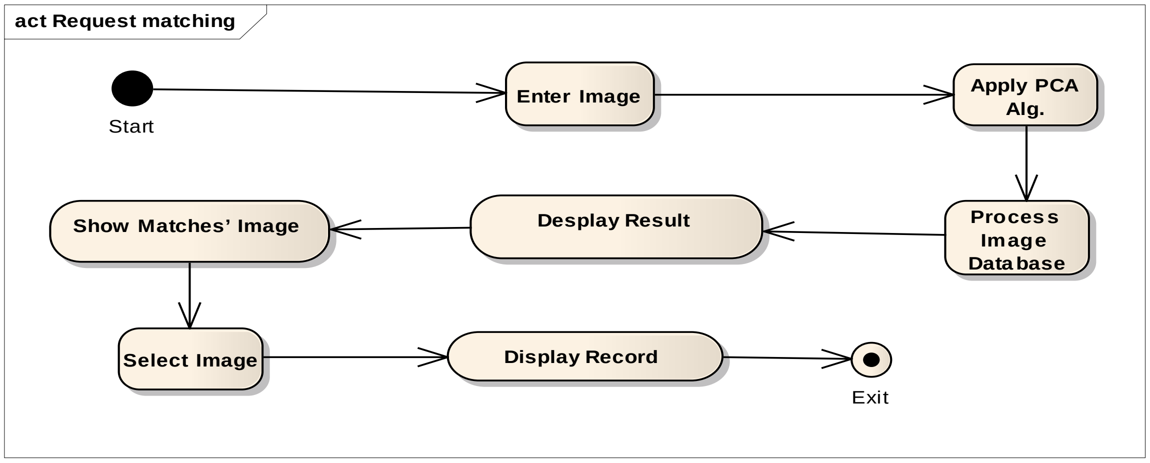


4.3.5 Activity diagrams

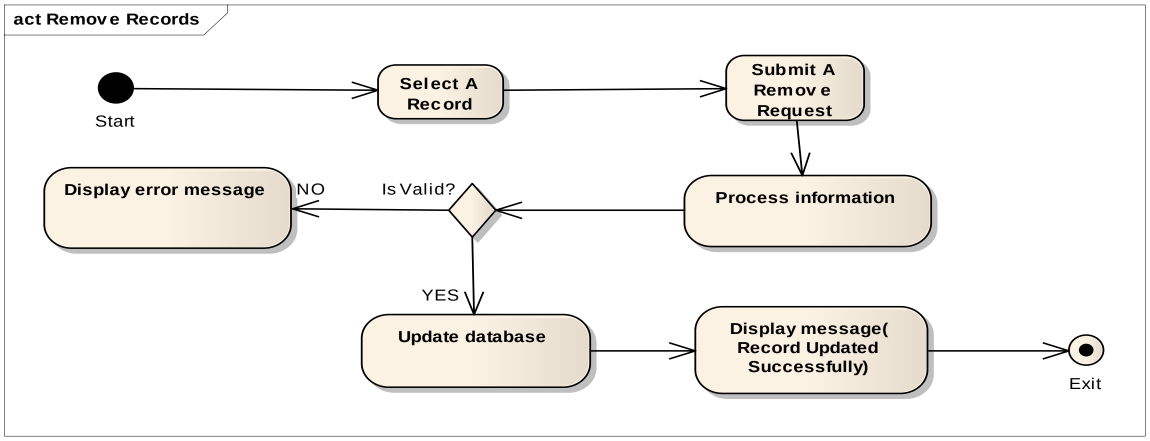
1. Insert a new record


act Insert a new record
Start
Click Add Records Buttons
Display new record form
Fill the form & Enter the Image folder
Submit information
Process information
Is Valid?
NO
Display error message
YES
Already exist?
NO
Display massage Data Already Exist
YES
Store information in database
Display new record added successfully
Exit

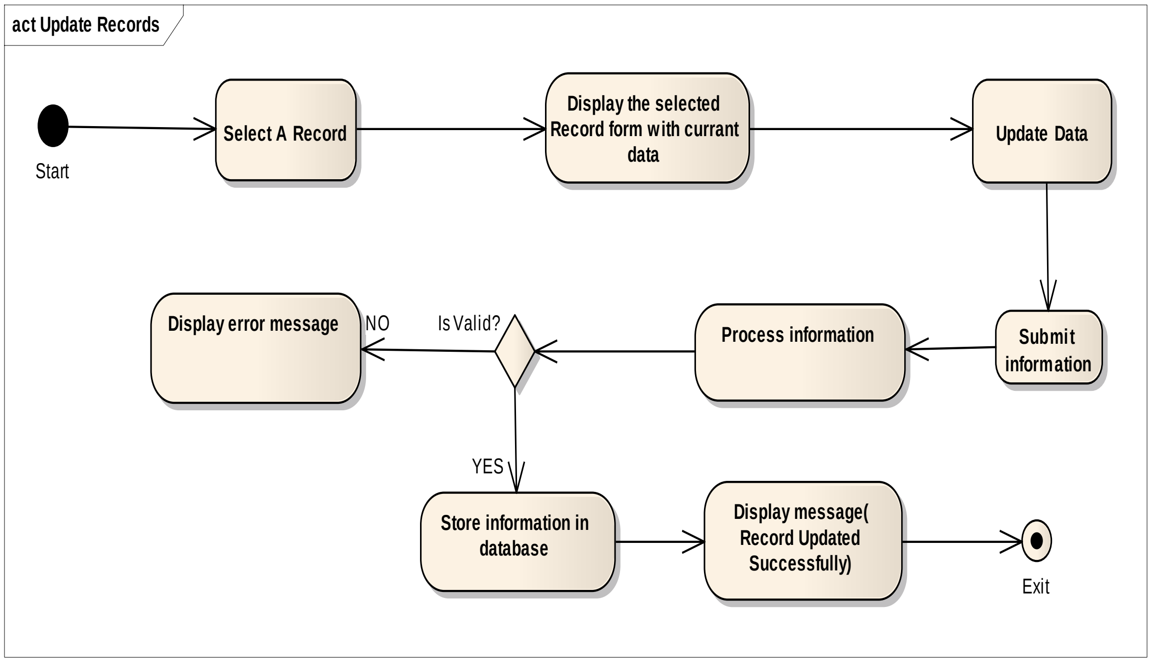
2. Request matching



3. Remove Records



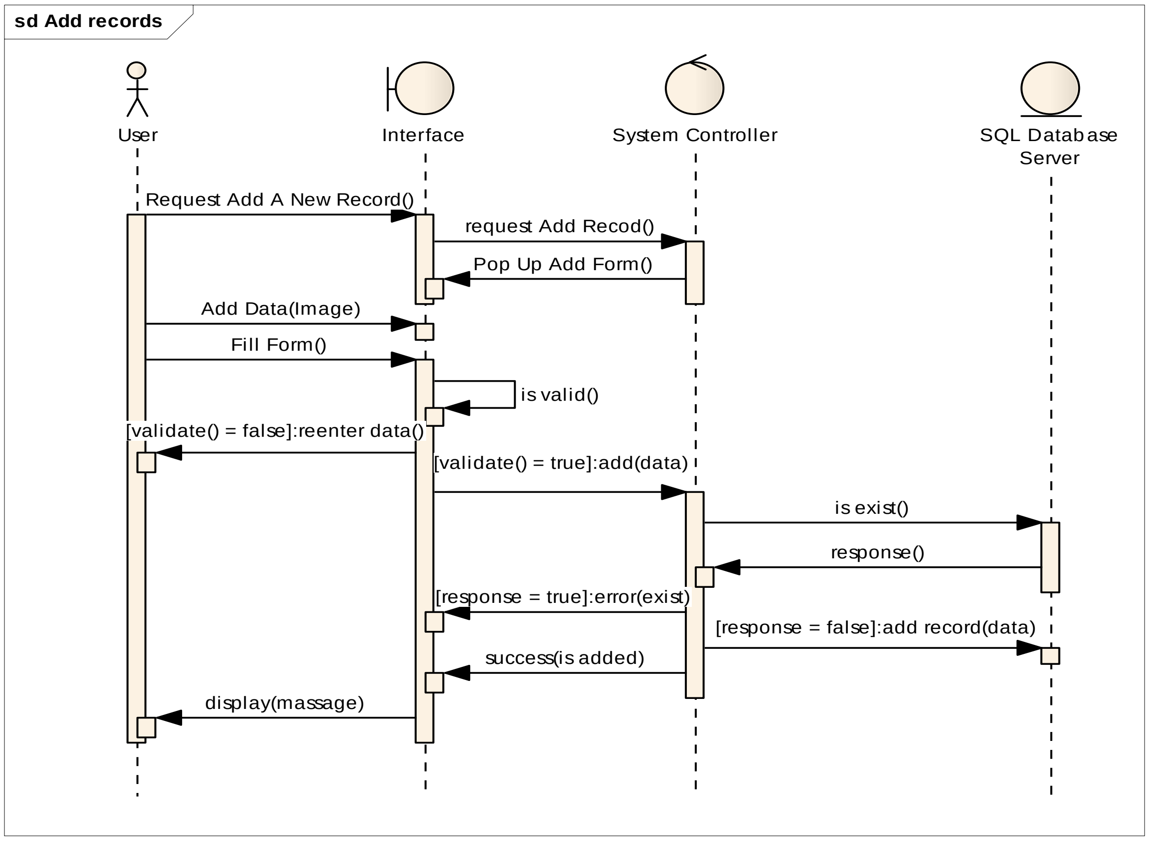
4. Update records



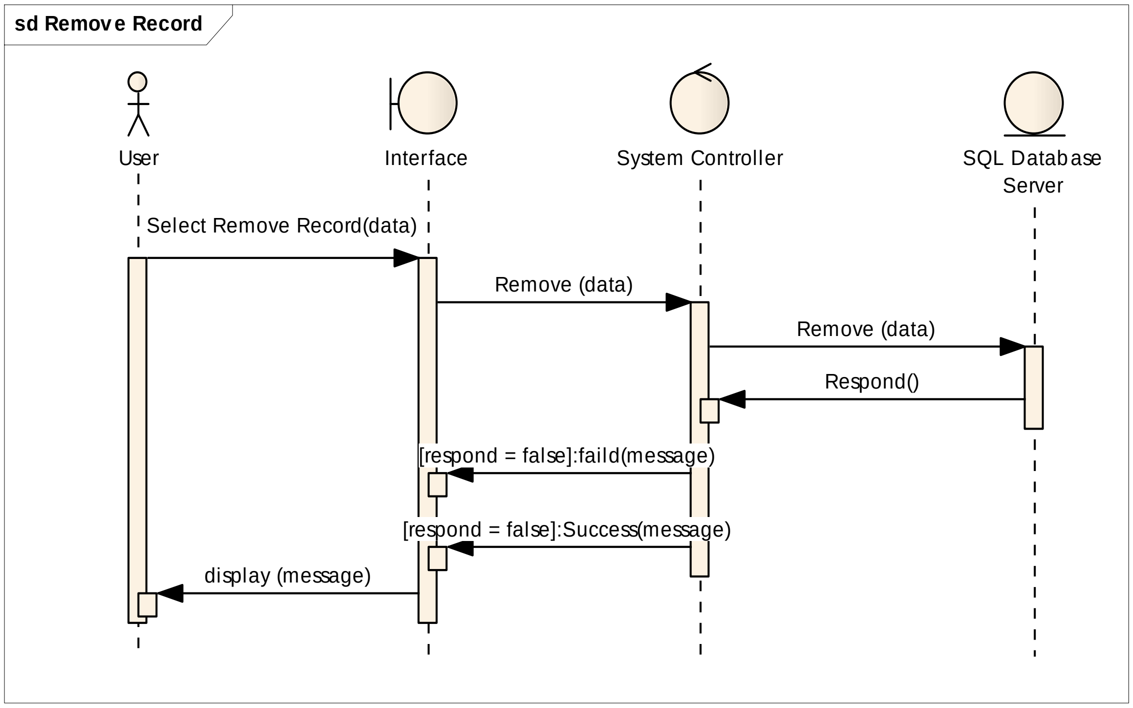
4.3.6 Sequence Diagrams

1. Add records:

This diagram shows the sequence of adding records to the database.

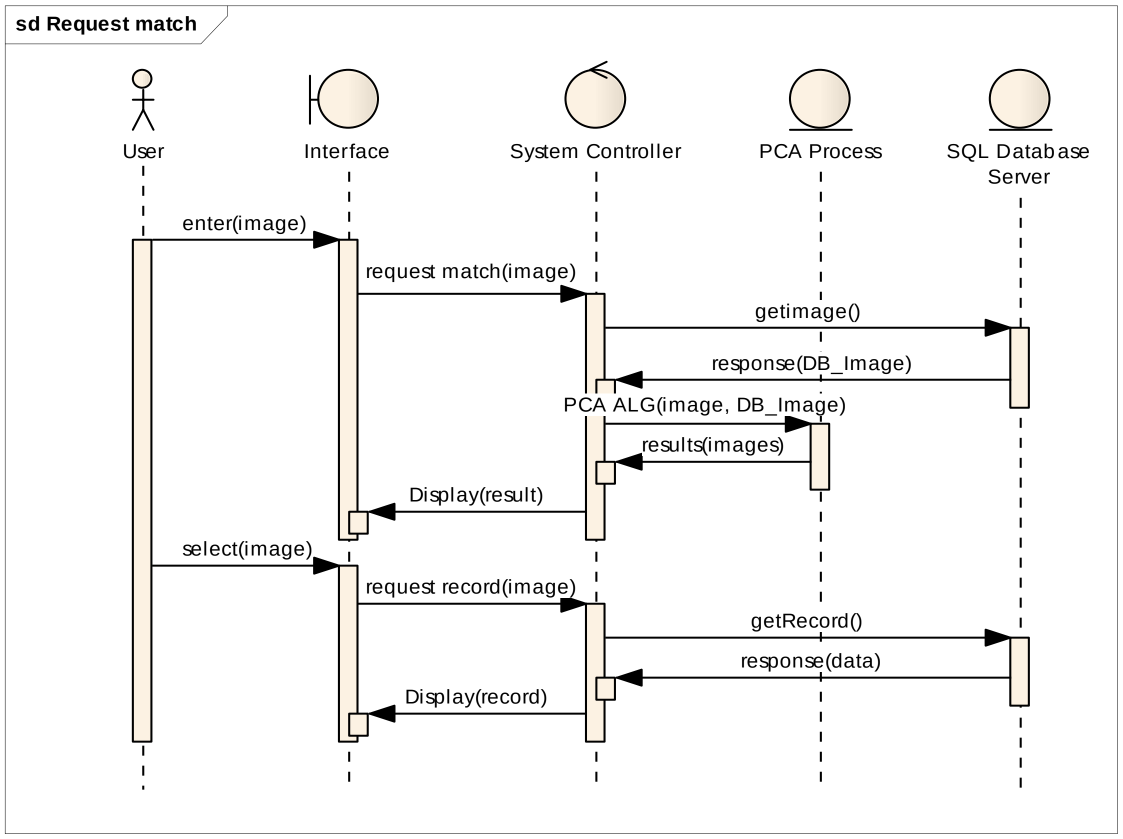


2. Remove records:

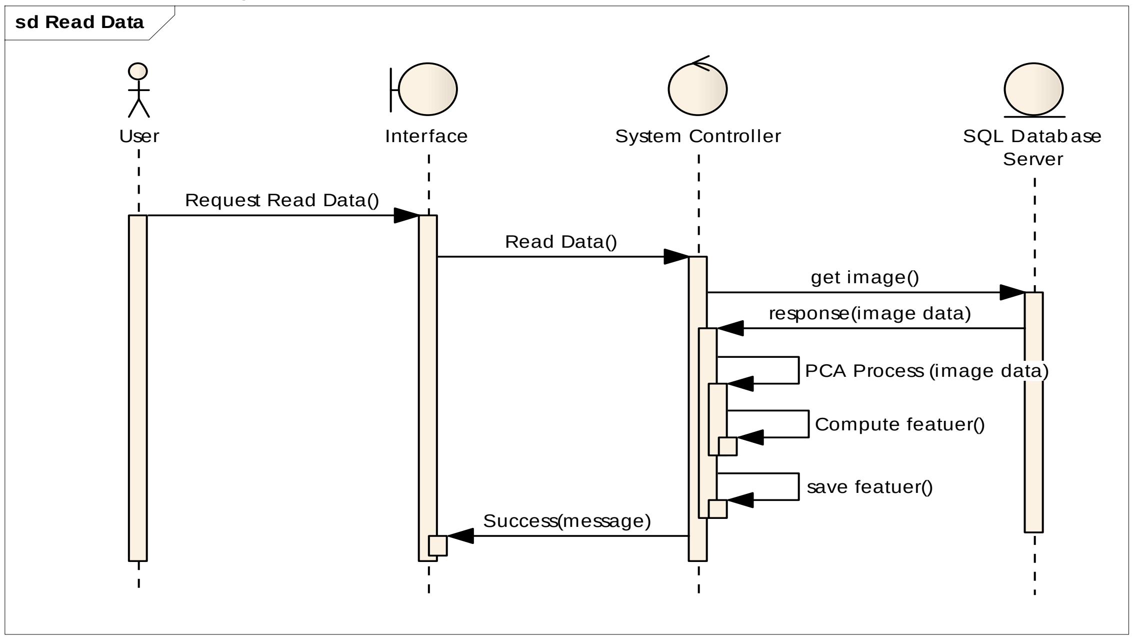


3. Request match:

This diagram shows the sequence of requesting a matching to an image.

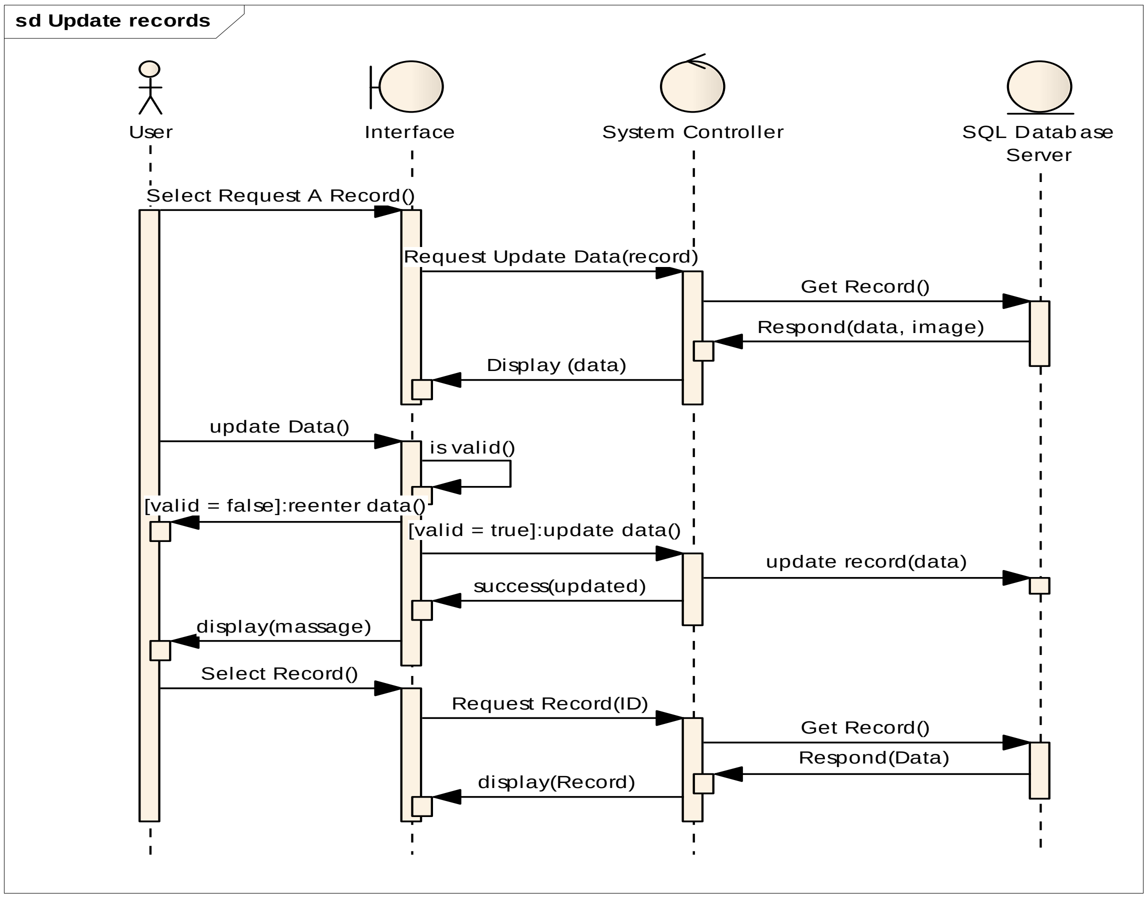


4. Read Image Data



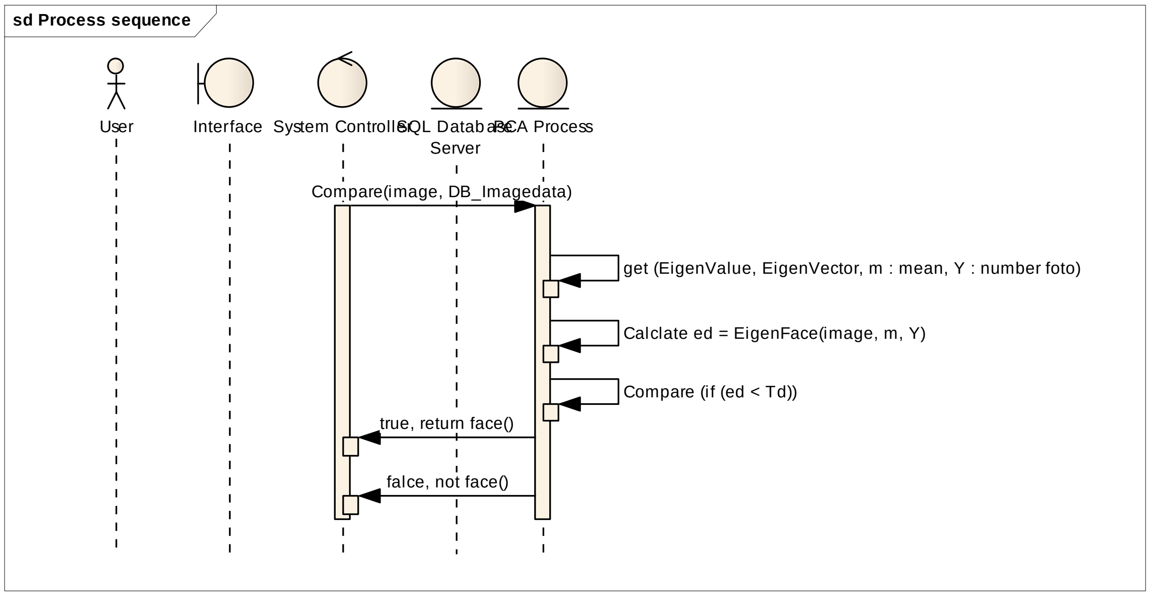
5. Update records:

This diagram shows the process to update the records in the database.



6. Process sequence:

This diagram shows the sequence of the main algorithm and processes to match an image.



5. SYSTEM ARCHITECTURE

5.1 System Process

1. Acquiring a sample
2. In this step, the system fed with a 2D image by the system user.
3. Extracting Features
4. For this step, the relevant data is extracted from the sample. This is can be done by using software where many algorithms are available, e.g. matlab library. The outcome of this step is a biometric template which is a reduced set of data that represents the unique features of the enrolled user's face.
5. Comparison Templates
6. This depends on the application at hand. For identification purposes, this step will be a comparison between the biometric template captured from the subject at that moment and all the biometric templates stored on a database. For verification, the biometric template of the claimed identity will be retrieved (either from a database or a storage medium presented by the subject) and this will be compared to the biometric data captured at that moment.
7. Declaring a Match
8. The face recognition system will either return a match or a candidate list of potential matches. In the second case, the intervention of a human operator will be required in order to select the best fit from the candidate list. An illustrative analogy is that of a walk-through metal detector, where if a person causes the detector to beep, a human operator steps in and checks the person manually or with a hand-held detector.

5.2 Design Rationale

1. It runs smoothly on machine.
2. Easy for management.
3. Easy to learn.

5.3 Architectural Description

1. Application Interface:
2. It's on top of the System and provides a graphical front end to the system.
3. Matcher:
4. It provides the algorithms and image processing needed to test if a match exists. In addition to the classes and functions needed to communicate with the Database.
5. Template Generator:
6. It holds the PCA algorithm and all libraries need to generate the Eigenface from the face images. Egienfaces are a large set of digitized images of human faces, taken under the same lighting conditions. Those images are normalized to line up the eyes and mouths. They are then all resample at the same pixel resolution. Egienfaces can be extracted out of the image data by mean of mathematical tool called principal component analysis.
7. Template Database:

8. It contains of two tables, one for the Eigenface template + face images and the second for personal information related to his/her face image.

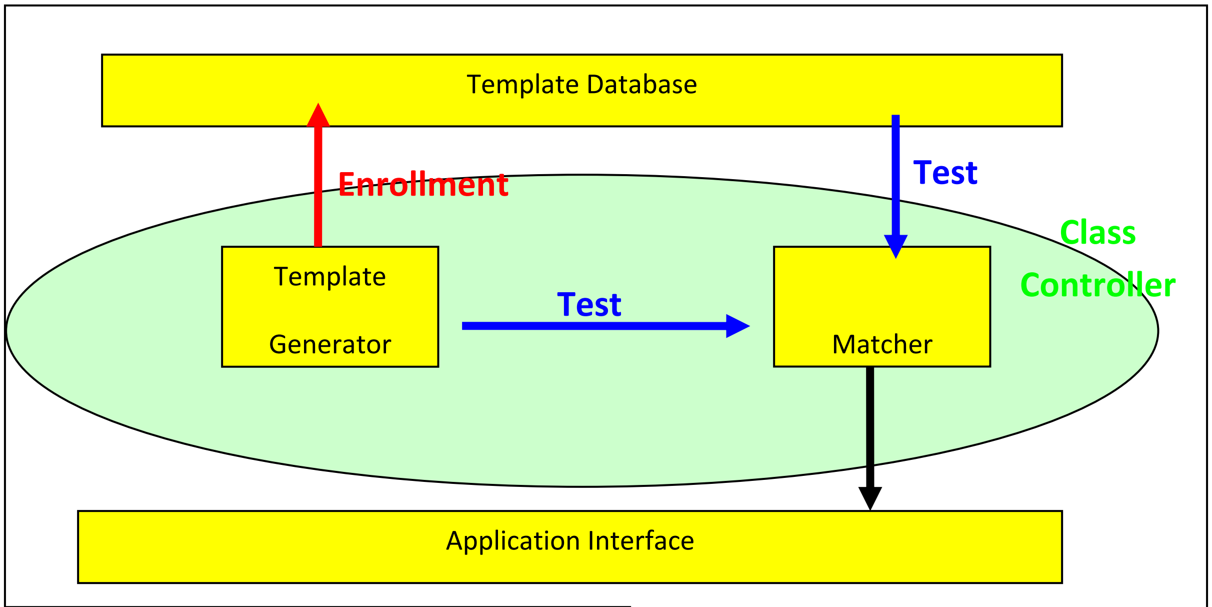
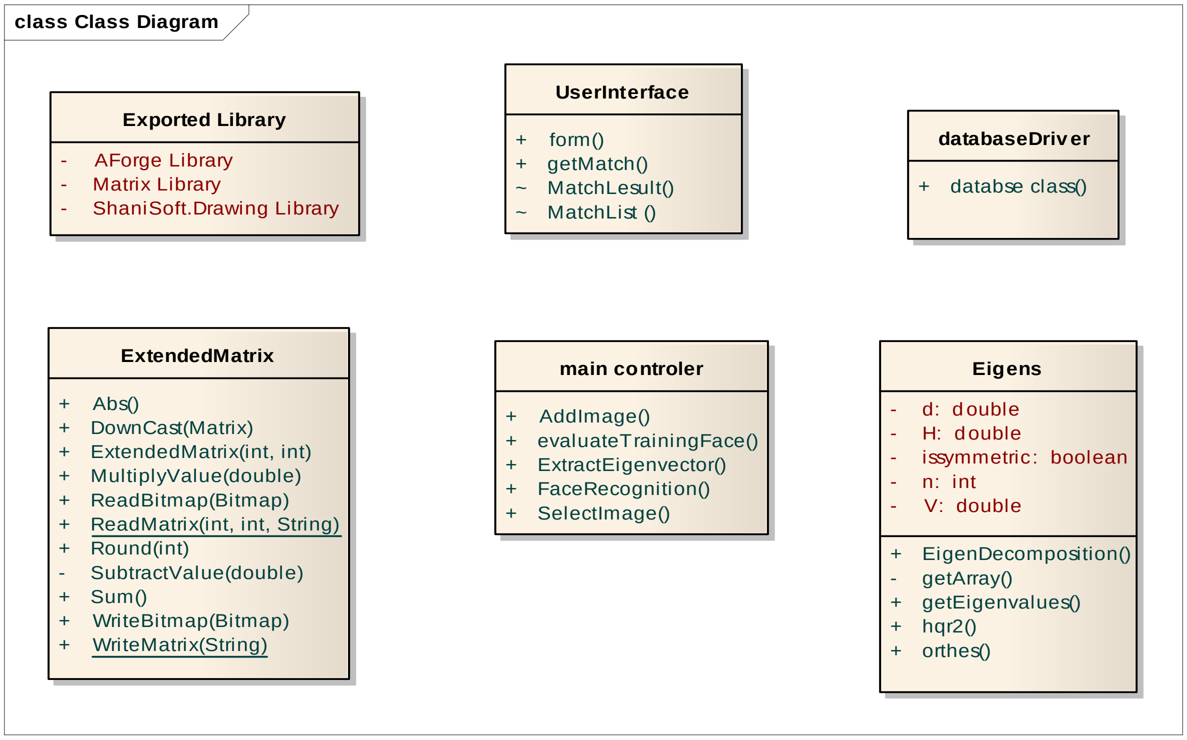


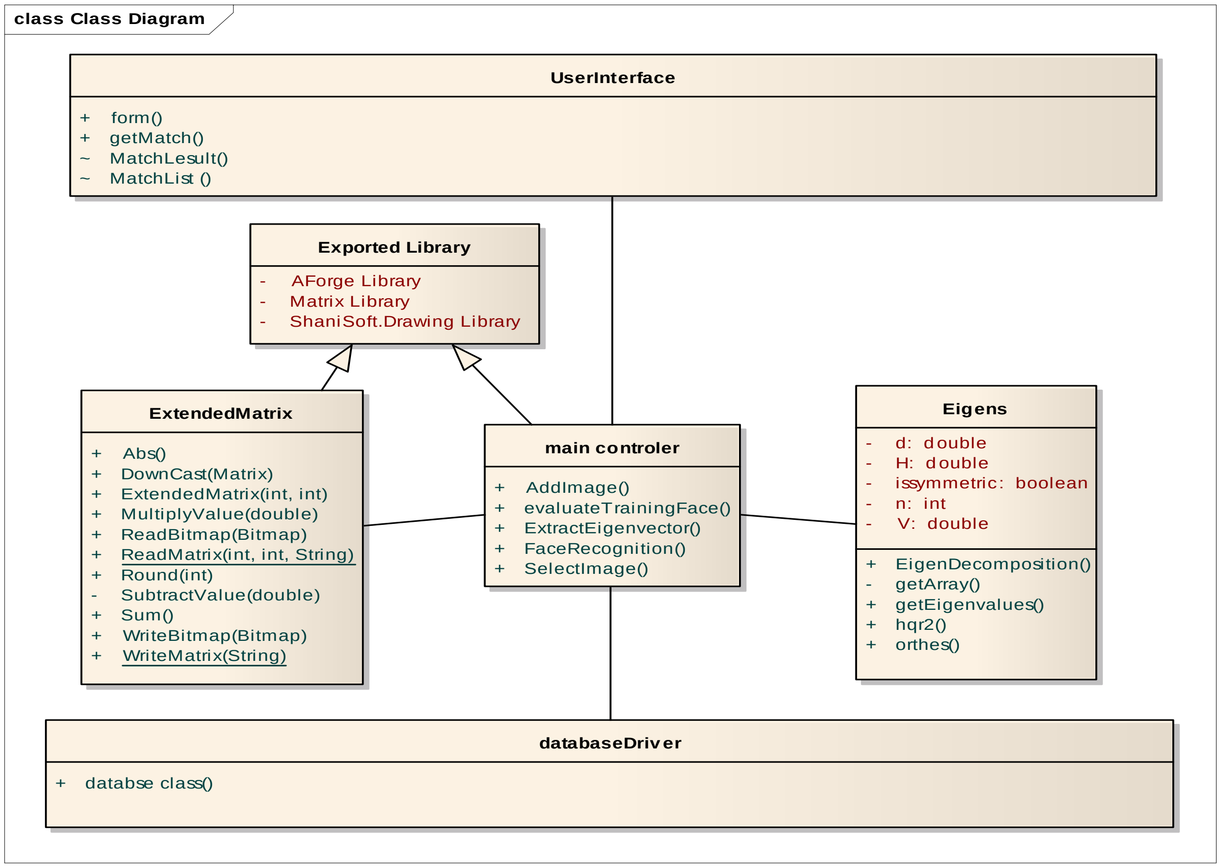
Figure 4: **Deployment Architecture**

5.4 Component Decomposition Description

1. Classes description



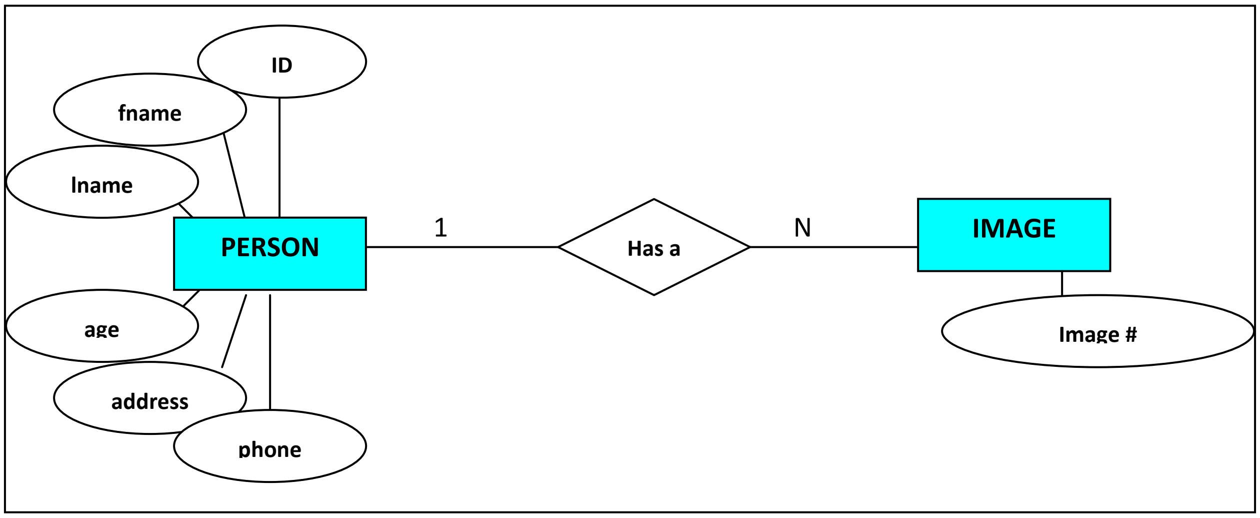
2. Relation between classes (Overview).



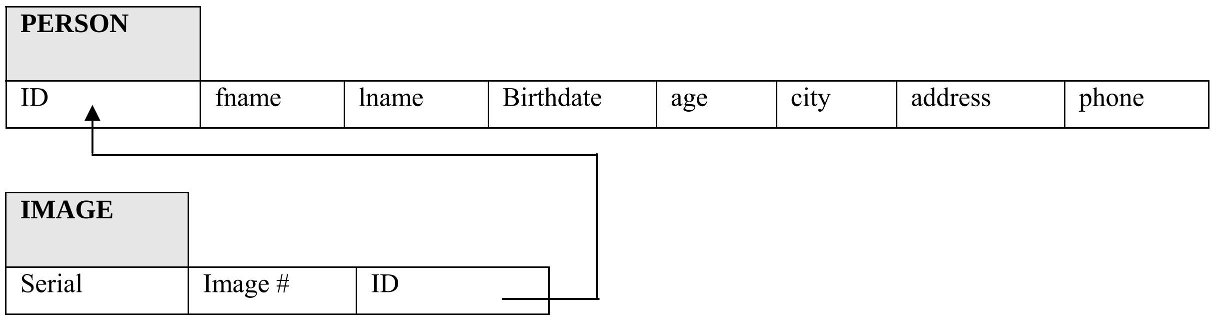
6. DATABASE DESIGN

In the fallowing it shows the database description and the relational schema:

6.1 Database Description



6.2 Relational Schema



6.3 Data Dictionary

Entity name	**PERSON**
Services	It holds all attribute about the owner of the image. A PERSON might have much face's image in the database.
Relations	IMAGE.

Entity name	**IMAGE**
Services	It holds the image number in the real database; each image must be associated to only one PERSON.
Relations	PERSON.

6.4 Create Table

PERSON

```
create table PERSON
(ID                 CHAR     UNIQUE,
fname               VARCHAR           NOT
NULL,
lname               VARCHAR           NOT
NULL,
birthDate  datetime   ,
streetAddress       VARCHAR           ,
city                VARCHAR           ,
phone               CHAR     NOT NULL,
PRIMARY KEY (ID) );
);
```

IMAGE

```
CREATE TABLE IMAGE
(N        INT       NOT NULL,
Image     image     NOT NULL,
id                  CHAR(6)
          NOT NULL,
PRIMARY KEY (N,ID),
FOREIGN KEY (id) REFERENCES PERSON
ON DELETE CASCADE
);
```

7. PCA IMPLEMENTATION USEING C#

To simplify our work, we will implement the system to be a static form, i.e. we will have a predefined database and predefined matrix dimensions where X=2500 and Y=number of images. We have 40 folders corresponding to 40 persons and each folder contains 10 images. We will read 7-image from each PERSON, whish result Y=280.

7.1. Used Libraries and Utility

Microsoft SQL Server2005 was used in our system to test and save the face image database and PERSON information.

1. Visual Studio2008 and C# were used to develop & test all the classes, libraries and the interface.
2. Libraries:
    a. CSML.Matrix; originally developed by hanzzoid under open license. However, the library doesn't contain all needed function. For this reason, we create ExtendedMatrix class which extends the CSML.Matrix. <http://www.codeproject.com/KB/cs/CSML.aspx/>
    b. Aforge image shop from <http://www.aforgenet.com/>
    c. Eigens class. It's used to extract the Eigenvectors and the Eigenvalues using the nonsymmetrical Hessenberg form. A full explanations is available on <http://books.google.com/books?hl=en&id=1aAOdzK3FegC&dq=numerical+recipes+in+C%2B%2B&printsec=frontcover&source=web&ots=3hOhC7Jpkk&sig=ISAcP04R4zFGNz5bVCV-xZN0Mgk&sa=X&oi=book_result&resnum=3&ct=result#PPA2,M1>
    d. ShaniSoft.Drawing. its open source library provided by codeproject.com. It is developed by Zeeshan Ejaz Bhatti under open license. It's used to read PGM image files. <http://www.codeproject.com/KB/graphics/Managed_PGM.aspx?display=Print>
3. Image Database was provided by aleix for research purposes under restricted approval for the team only. <http://cobweb.ecn.purdue.edu/~aleix/aleix_face_DB.html>.

7.2. Obstacle and Difficulty.

7.2.1 Difficulty in Matrix Operations

The team has faced so many obstacles during developing and researches. The major impediment was to develop an optimal Matrix library that can hand and process the image Array of byte. However, the hardest part is to develop an iterative function to decompose the Eigenvectors from a given matrix. . Finding eigenvectors is unfortunately not an easy task unless you have a rather small matrix, like no bigger than 3×3. It's also, getting harder and harder because

C# isn't a suitable platform to develop mathematical functions such as Eigen decomposition. This problem was overcome, by using a numerical method called Hessenberg form.

SIDE NOTE: Our implementation is based on nonsymmetric Hessenberg form. Basically, finding Eigenvalues and eigenvectors of a real matrix is depending on if it's symmetric matrix or non-symmetric matrix:

```
If A is symmetric, then A = V*D*V' where the eigenvalue matrix D is
diagonal and the eigenvector matrix V is orthogonal. I.e. A =
V.times(D.times(V.transpose())) and V.times(V.transpose()) equals
the identity matrix.
```

```
If A is not symmetric, then the eigenvalue matrix D is block
diagonal with the real eigenvalues in 1-by-1 blocks and any complex
eigenvalues, lambda + i*mu, in 2-by-2 blocks, [lambda, mu; -mu,
lambda].  The columns of V represent the eigenvectors in the sense
that A*V = V*D, i.e. A.times(V) equals V.times(D).  The matrix V
may be badly conditioned, or even singular, so the validity of the
equation A = V*D*inverse(V) depends upon V.cond().
```

7.2.2 Difficulty to Define a Proper Threshold

For rejection of unknown faces a threshold τ is chosen and it is said that the face with projection z_{new} is unknown if $\varepsilon_s \geq \tau$. One way to calculate the threshold is: divide the database into half for training set and the remaining for the testing set. Assume; we have a database of have 40 folders corresponding to 40 persons and each folder contains 10 images. We take 5 images for training and 5 for testing from each folder. This is in order to investigate the accuracy of our code. It's noticed that if we increase the number of training, the accuracy increases. Also, it's noticed the relation is positive correlation between the number of training images taken for a person and the accuracy. As we increase the training images for a person, the accuracy increases.

7.3. PCA Implementation.

At the beginning, we have out initial database that contains all the images of the people that we want to identify. The images database will be read from the SQL server by using **`readImage(),`** this function selects all records in table IMAGE, and associate each image to the person ID. Then, it creates 1-D matrix from each image by using the **`matrix.ReadVMatrixFromBitmap(Image[i]).`** Finally, it inserts each created matrix side by side in the main matrix. Eventually, the returned result is 2-dimentional matrix each column is 1-dimentional vector image. This 2-

dimentional matrix is the main matrix; it's also called **`TRmatrix`** (training matrix) which contains 400 columns corresponding to 400 images (40 persons and 10 images each).

Now, our data set is ready and we can now find the features using the function **`PCAFeatuers(ExtendedMatrix TRmatrix, int k)`**. This function takes two arguments, the TRmatrix which is 2- dimensional matrix as explained above. And the second is **`k,`** it's an integer that holds how many eigenvector we which to keep. This function will compute the features for the given matrix. At first it will take TRmatrix to compute **`eigenFaceMatrix`** (Eigenface matrix, it also known as Eigenvectors) and then, it will compute **`transformedMatrix`** (Transformed matrix)

```
        private ExtendedMatrix[] PCAFeatuers(ExtendedMatrix TRmatrix, int k)
        {
            ExtendedMatrix mean = TRmatrix.MeanVector();
            ExtendedMatrix adjTRmatrix = TRmatrix.SubtractVector(mean);
            ExtendedMatrix transposeAdj = 
ExtendedMatrix.DownCast(adjTRmatrix.Transpose());
            msg = "On Progress: Extracting Mean Vector, Adjusted Matrix & \nThe 
Coveriance Matrix!!";
            ExtendedMatrix covariance = (transposeAdj * adjTRmatrix);

            msg = "On Progress: Extracting the Eigenvectors & the 
Eigenvalues!!";
            Eigens e = new Eigens(covariance);
            //ExtendedMatrix eigenvalues = e.getEigenvalues;
            ExtendedMatrix eigenvectors = e.getEigenvector.Abs();
            ExtendedMatrix eigenFaceMatrix = 
eigenvectors.SubSample(eigenvectors.RowCount, k);

            msg = "On Progress: Calculating the Transformed Matrix!!";
            ExtendedMatrix transformedMatrix = adjTRmatrix * eigenFaceMatrix;

            ExtendedMatrix[] featuer = new ExtendedMatrix[3];
            featuer[0] = eigenFaceMatrix;
            featuer[1] = transformedMatrix;

            eigenFaceMatrix.WriteMatrixToFile(rootDirectory + "\\feature.dat");
            transformedMatrix.WriteMatrixToFile(rootDirectory + "\\tfm.dat");

            return featuer;       }
```

Figure 4: the Code for PCAFeatuers

First, the function will calculate **`eigenFaceMatrix`**. To do that, it takes the mean from each row in the given training matrix using **`TRmatrix.MeanVector();`** the result is one-dimensional vector. This 1-D vector is subtracted from each column (image) in the **`TRmatrix`** in order to get the **`adjTRmatrix`** (adjusted training matrix). Then, the **`covariance`** is calculated by multiplying the adjusted training matrix with its transpose. By using the command **`Eigens(covariance).getEigenvector.Abs(),`** the normalized **`eigenvectors`** is found from the **`covariance`** matrix. The result is given back in a decreasing order from highest corresponding eigenvalue to the lowest one. By now, the whole eigenvectors is found; we will keep only the highest k-eigenvectors corresponding to the k largest eigenvalues. This is done by using `eigenvectors.SubSample(eigenvectors.RowCount, k)`. The result is `eigenFaceMatrix`. Finally, by multiplying the `adjTRmatrix` with the found `eigenFaceMatrix`, the transformation matrix `transformedMatrix` has been computed.

SIDE NOTE: Features Computation Sequence:

```
TRmatrix[2500, 280]
mean [2500, 1]        = TRmatrix.MeanVector ();
adjTRmatrix[2500, 280]= TRmatrix.SubtractVector(mean);
covariance [280, 280] = transposeAdj[280, 2500] * adjTRmatrix[2500,
280];
eigenvectors[280, 280]= Eigens(covariance).getEigenvector.Abs();
eigenFaceMatrix[280, 64]   = eigenvectors.SubSample(280, 64);
transformedMatrix[2500, 64]  =
      adjTRmatrix[2500, 280]  * eigenFaceMatrix[280, 64];
```

For seek fast performance, the system will save the features in the local directory. Next time when we need the Eigenface or the transformed matrix, they will be read from the local directory if they're available.

After we perform a PCA, our original data is expressed in terms of eigenvectors found from the covariance matrix. This is useful because say we want to do facial recognition and so our original images were of people faces. Then, the problem is given a new image, whose face from the original set is it? (Note that new image is not one of original images). The way this is done is to measure the difference between the new image and original images, but not along the original axes, along the new axes derived from the PCA analysis. It turns out that these axes work much better for recognition faces, because the PCA analysis has given us the original images in terms of differences and similarities between them. PCA has identified the statistical patterns in the data.

Next step is to perform the PCA algorithm to test and identify a given image. The idea is, if the image is in dataset, it's possible to return back the eigenvector for the given image matrix. It's done by multiplying the transposed `transformed matrix` for the dataset by the image matrix. The result is 1-dimentional vector Eigenface for this given image; we call it (`EigenImage`). To identify the image, the minimum Euclidean distance is computed between the found `Eigenimage` and each column in the Eigenface of the whole training set, and returns back the image index, which is assumed to be the image index in the training matrix.

First step in finding match process is to prepare the image. Assume an image was given for a person. No matter what's its light condition and size, the system will read the given image, resize its dimensions to be compatible with the database images, i.e. the image will be resized to image (50, 50) using function **`ResizeNearestNeighbor(50, 50).Apply(image)`** which is given by **`AForge.Imaging.Filters`** library. Also, **`Grayscale(0.2125, 0.7154, 0.0721).Apply(image);`** is applied to bring back the image to lightless, gray color image. Those two functions are important to element any undesirable differences in the image status, for example, light, background color and image sizes.

```
            ExtendedMatrix imEigenFace = transformedMatrix * Im;

            ExtendedMatrix tmp =
                ExtendedMatrix.DownCast(imEigenFace -
        eigenFaceMatrix.Column(1)).Abs();
            ArrayList normList = new ArrayList();
            normList.Add(new Association(tmp.Norm() + "", 1));
            for (int index = 2; index <= eigenFaceMatrix.ColumnCount; index++)
            {
                tmp =
                ExtendedMatrix.DownCast(imEigenFace -
        eigenFaceMatrix.Column(index)).Abs();
                normList.Add(new Association(tmp.Norm() + "", index));
            }
            // sort
            normList.Sort();

            ArrayList imList = new ArrayList();
            for (int i = 0; i < 40; i++)
            {

                imList.Add((Association)matrixlist[(int)((Association)normList[i]).Val
                ue]);
            }
```

Figure 5: the Code for `faceDetection`

After, the image has been prepared. The second step is to declare a match. The image will be transformed into 1-dimentional vector using **`Imatrix.ReadVMatrixFromBitmap(image).`** Then, it will be multiplied by the **`transformedMatrix.Transpose(),`** the result is 1-dimentional image **`Eigenimage`**. Then, the Euclidean distance will be computed between each column in **`eigenFaceMatrix`** and the 1-D image **`Eigenimage`**. The result will be saved in Array list and sorted from the minimum distance to the highest based on the key of the Association class. Finally, we will select the first 40-minimum distance, get the index value, get the matrix from the matrix-image list and return the result to the interface class to display the result.

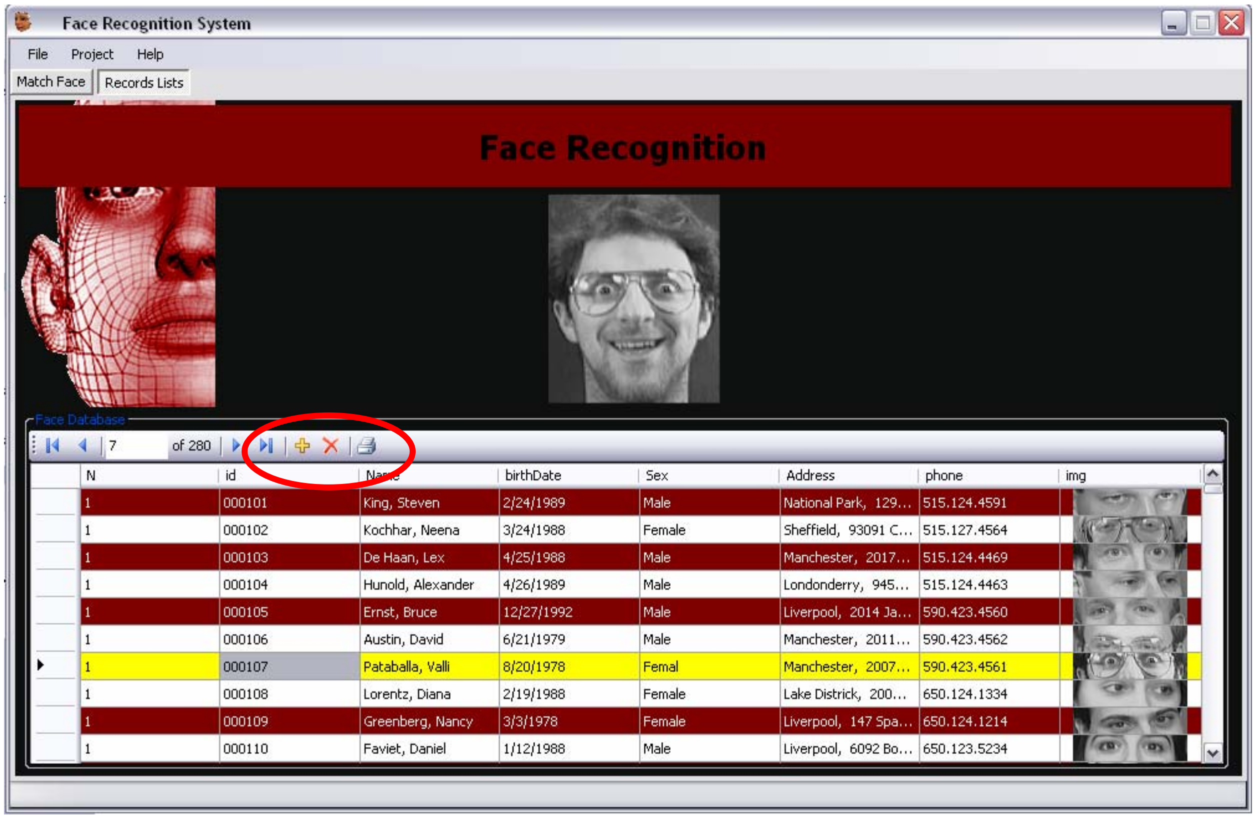
SIDE NOTE: Image Detection Sequence:

```
imageMatrix[2500, 1]
mean [2500, 1]              = TRmatrix.MeanVector ();
adjImgMatrix[2500, 1]       = imageMatrix.SubtractVector(mean);
imFeatuer[64, 1]            = transformedMatrix.Transpose()[64, 2500] *
adjImgMatrix[2500, 1];
Euclidian dis =
||imFeatuer. Transpose()[ [1, 64] - eigenFaceMatrix[280, 64].[i] ||
```

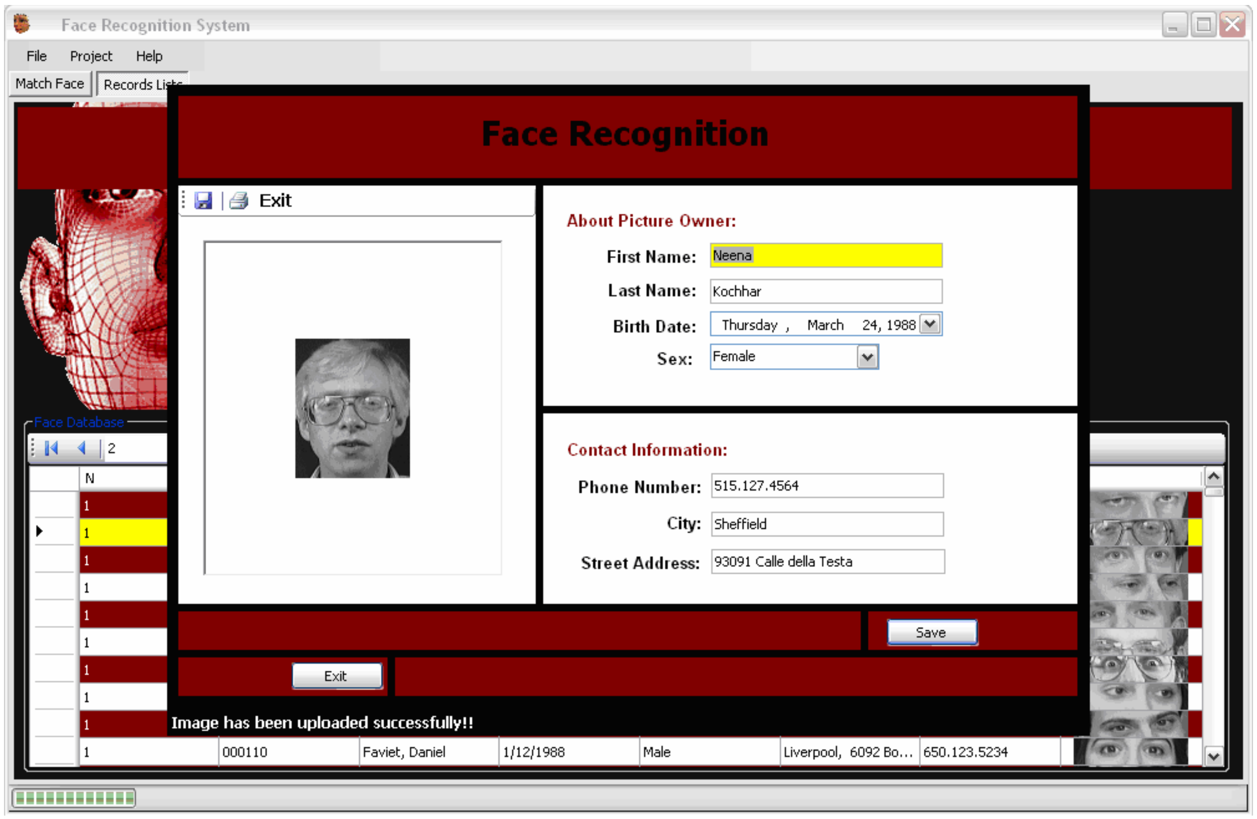
7.4. Graphical User Interface

1. Main Page, Database view

The fallowing figure shows the system main page, it's simply present an over view to face database. In addition to the basic function (Traverse, Add, Remove, Print), the user can double click on the any record, to retrieve back the selected data in popup form as shown in next figure, where the user can modify the record, change the image, and print the record.



N	id	Name	birthDate	Sex	Address	phone	img
1	000101	King, Steven	2/24/1989	Male	National Park, 129…	515.124.4591	
1	000102	Kochhar, Neena	3/24/1988	Female	Sheffield, 93091 C…	515.127.4564	
1	000103	De Haan, Lex	4/25/1988	Male	Manchester, 2017…	515.124.4469	
1	000104	Hunold, Alexander	4/26/1989	Male	Londonderry, 945…	515.124.4463	
1	000105	Ernst, Bruce	12/27/1992	Male	Liverpool, 2014 Ja…	590.423.4560	
1	000106	Austin, David	6/21/1979	Male	Manchester, 2011…	590.423.4562	
1	000107	Pataballa, Valli	8/20/1978	Femal	Manchester, 2007…	590.423.4561	
1	000108	Lorentz, Diana	2/19/1988	Female	Lake Districk, 200…	650.124.1334	
1	000109	Greenberg, Nancy	3/3/1978	Female	Liverpool, 147 Spa…	650.124.1214	
1	000110	Faviet, Daniel	1/12/1988	Male	Liverpool, 6092 Bo…	650.123.5234	



About Picture Owner:

First Name: Neena

Last Name: Kochhar

Birth Date: Thursday , March 24, 1988

Sex: Female

Contact Information:

Phone Number: 515.127.4564

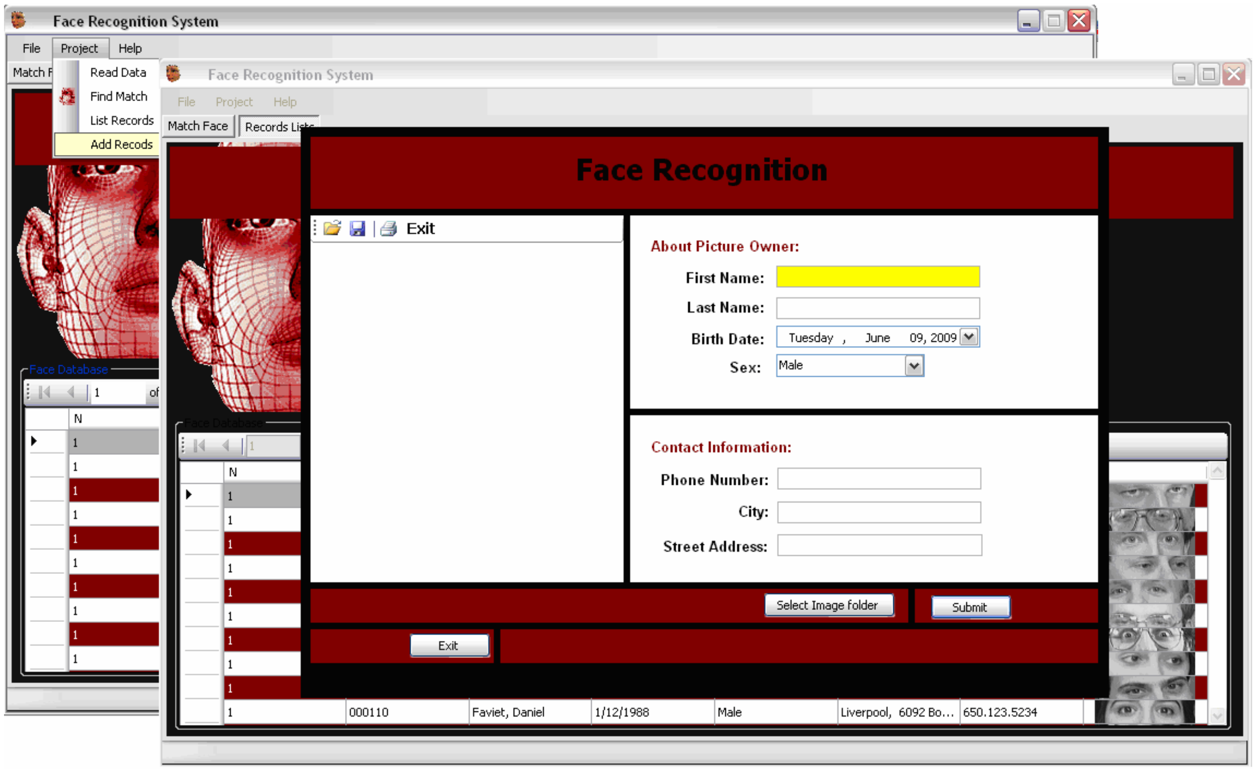
City: Sheffield

Street Address: 93091 Calle della Testa

Image has been uploaded successfully!!

1	000110	Faviet, Daniel	1/12/1988	Male	Liverpool, 6092 Bo…	650.123.5234

2. Add New Data Record



About Picture Owner:

First Name:

Last Name:

Birth Date: Tuesday , June 09, 2009

Sex: Male

Contact Information:

Phone Number:

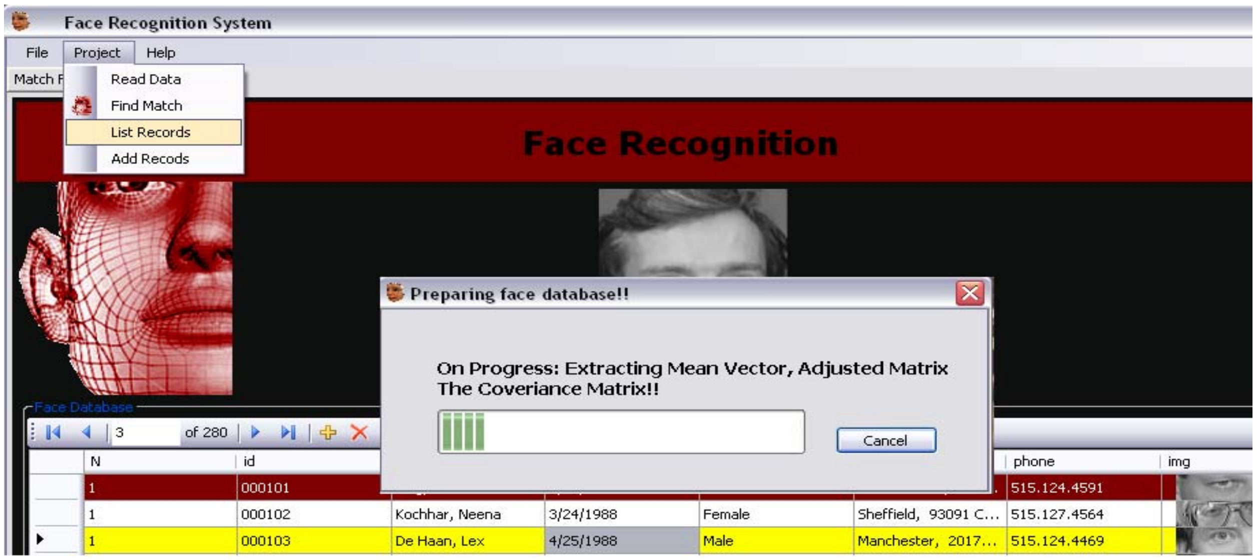
City:

Street Address:

1	000110	Faviet, Daniel	1/12/1988	Male	Liverpool, 6092 Bo…	650.123.5234

3. Read Data.

The system need to read the face database to create and save the Eigenface & the eigenvector. This process required expert supervision; it must be done manually, whenever there is update in the database content. Otherwise, the system will keep reading the old saved features. a more developed version where the data will be read automatically is possible, but it's involve more overhead, and time consuming.

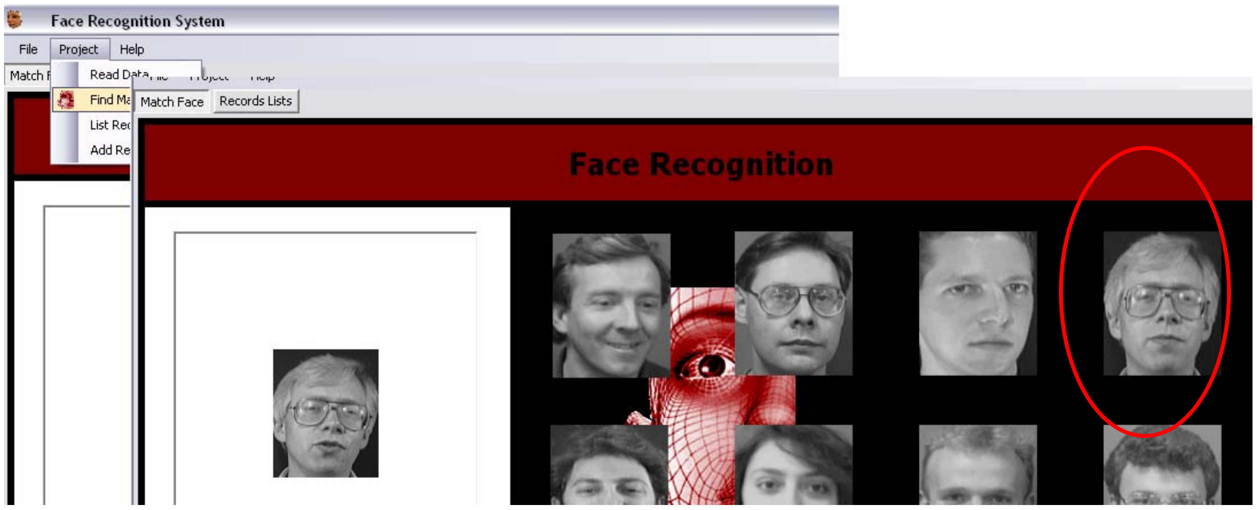


On Progress: Extracting Mean Vector, Adjusted Matrix
The Coveriance Matrix!!

N	id					phone	img
1	000101					515.124.4591	
1	000102	Kochhar, Neena	3/24/1988	Female	Sheffield, 93091 C...	515.127.4564	
1	000103	De Haan, Lex	4/25/1988	Male	Manchester, 2017...	515.124.4469	

4. Find Match

To find a match, a user is required to drag and drop the image in drag-drop box or the user can just add the image normally, from the File drop menu.



8. SYSTEM EVALUATION

8.1 Tests and Results

In order to investigate the accuracy of our code, a set of accuracy test was performed. It's noticed that if we increase the number of training, the accuracy increases. Also, it's noticed the relation is positive correlation between the number of training images taken for a person and the accuracy. As we increase the training images for a person, the accuracy increases.

8.2 Observation

During our test, we remark some undesirable conditions that must be take care of to get the best result. We notice that, uncontrolled environmental condition; to ensure the best result, the image must strip out of its background, or all training set must have a unique background color. Also, must have the same dimensions size and the same color. We enforce out training set to be 50x50 in, and to have a gray scale color, all this will be take care of it automatically. Another notice is uncooperative subject condition; we notice the subject must look forward to get the best result. In addition, the result will be effected if the subject wearing glasses, having facial hair, or having an unusual expression.

8.3 RECOMMENDATIONS

At the end of this project we learn the basic idea of face recognition system. We discover some the points that affect the operation of the system. So, we recommend the following in order to get an efficient system.

1. Use a high quality image with no background.
2. Take a sufficient number of training images for a person. The specific number depends on the total number of the people in the database. As people increases, we have to take more training pictures.
3. Provide a good location to take pictures and utilize the face detection.
4. Investigate parameters that affect the performance of the code like the distance measures.

CONCLUSION

By the end of this project we were able to design a simple face recognition system using the C# windows application. The first stage in the project was the modeling. We created a C# code to implement the PCA. The main problems we faced in this stage were related to programming issues like extract Eigenvectors from matrix using C# codes and how to read/write image binary data into SQL server.

The second stage was testing the performance of the code. At the beginning we worked on small number of training images and one test image. Then, we applied the code to the whole database of 40 people and conduct many tests. The result of those test show up a proportional relationship between the training images set and the accuracy. This is because as we increase the number of training images, we get more information (features). Also, researcher finds that there is an effect of the type of the distance measures used on the accuracy. It was found that different distance measures give different accuracies. We tried three different distance measures which are Euclidean Distance, Squared Euclidean Distance and Modified Manhattan Distance. The last one was the best where it gave us an accuracy of around 88%.

The final stage was to covert the code to a Graphical User interface so that the system can be used friendly. The most important component of this interface was to provide the system with a person image and identifying him using the stored data.

We faced several problems in this step that affect the result. These include the image size and lightning of the room. A solution to this problem was to focus a white light on a person before we take his picture. Other problem was the picture's background. When a picture is taken for a person, unwanted disturbances from the surrounding that result in a low accuracy. A solution was to use face detection.

Work Cited

1. Aleemuddin, Mohammed and Mohamed Deriche. "*Pose-invariant Face Recognition Using Subspace Techniques*".
(library.kfupm.edu.sa/lib-downloads/A1A4672.pdf)

2. "*An Introduction to Biometrics - Face Recognition*".
(http://www.tiresias.org/guidelines/biometrics_face.htm)

3. Kim, Kyungnam." *Face Recognition using Principle Component Analysis*".
(www.umiacs.umd.edu/~knkim/KG_VISA/PCA/**Face**Recog_PCA_Kim.pdf)

4. Manfred U. A. Bromba. "*Biometrics: Frequently Asked Questions*"
(http://www.bromba.com/faq/biofaqe.htm)

5. Perlibakas, Vytautas. "*Distance measures for PCA-based face recognition*"
(portal.acm.org/citation.cfm?id=987382)

6. Smith, Lindsay." *A tutorial on Principal Components Analysis*".
(www.cs.otago.ac.nz/cosc453/student_tutorials/**principal_components**.pdf)

7. Wikipedia, the free encyclopedia. " *Biometrics*"
(http://en.wikipedia.org/wiki/Biometrics)