FACE RECOGNITION USING EIGENFACES AND DATA MINING TECHNIQUES

By

AMIT SIROHI061404
CHANPREET SINGH061414
PARTH BATRA061434
SHASHI SHEKHAR ANAND061451

MAY-2010

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF
COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
JAYPEE UNIVERSITY OF
INFORMATION TECHNOLOGY-WAKNAGHAT

Department of Computer Science and Engineering

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT, SOLAN – 173215

CERTIFICATE

This is to certify that the work entitled, "**Face Recognition Using Eigenfaces and Data Mining Techniques**" submitted by **Amit Sirohi (061404), Chanpreet Singh (061414), Parth Batra (061434)** and **Shashi Shekhar Anand (061451)** in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science and Engineering (B.Tech – CSE), of JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of any other degree or diploma.

[illegible]

Mr. Praveen Kumar Tripathi
(Project Supervisor)
Department of Computer Science and Engineering,
Jaypee University of Information Technology,
Waknaghat, Solan – 173215, India

ACKNOWLEDGMENT

Computer Science is fast developing subject & new dimensions are being added from time to time. During the last decade many new dimensions have been introduced and many old ones have been redefined. In light of new development and recent findings, we devote the task that was asked from us during our IVth year at Jaypee University of Information Technology to Face Recognition Using Eigenfaces and Data Mining Techniques.

We would like to thank Mr. Praveen Kumar Tripathi who helped us during the task given to us. We thank them again for providing us with the necessary facility for conducting this task.

Amit Sirohi	Roll No. 061404	Amit.
Chanpreet Singh	Roll No. 061414	[illegible]
Parth Batra	Roll No. 061434	Parth Batra
Shashi Shekhar Anand	Roll No. 061451	Shashi

B.Tech (Computer Science & Engineering)
Jaypee University of Information Technology

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

I	Face image
N × N	Size of I Training set
M =	| | Number of Eigenfaces
M'	Number of eigenfaces used for face recognition
C	Covariance matrix
X^T	Transposed X (if X is a matrix)
U	Eigenvector (eigenface)
PCA	Principal Component Analysis

ABSTRACT

This project is able to recognize a person's face by comparing facial structure to that of a known person. This is achieved by using forward facing photographs of individuals to render a two-dimensional representation of a human head. The system then projects the image onto a "face space" composed of a complete basis of Eigenfaces. Because of the similarity of face shape and features from person to person, face images fall within a relatively small region of the image space and as such can be reproduced with less than complete knowledge of the image space. When new images are fed into these systems it can identify the person with a high rate of success with the robustness to identify correctly even in the presence of some image distortions.

INTRODUCTION

Do I Know You?

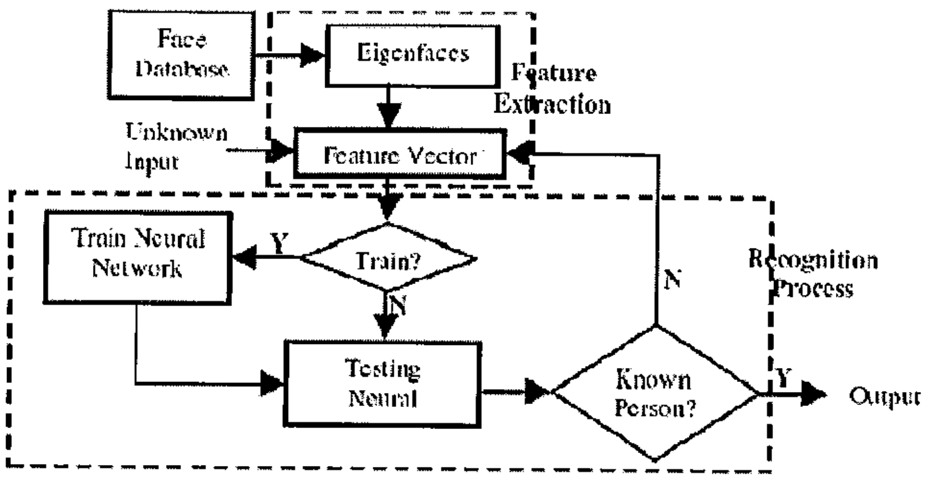
The human capacity to recognize particular individuals solely by observing the human face is quite remarkable. This capacity persists even through the passage of time, changes in appearance and partial occlusion. Because of this remarkable ability to generate near-perfect positive identifications, considerable attention has been paid to methods by which effective face recognition can be replicated on an electronic level. Certainly, if such a complicated process as the identification of a human individual based on a method as non-invasive as face recognition could be electronically achieved then fields such as bank and airport security could be vastly improved, identity theft could be further reduced and private sector security could be enhanced.

Many approaches to the overall face recognition problem (The Recognition Problem) have been devised over the years, but one of the most accurate and fastest ways to identify faces is to use what is called the "Eigenface" technique. The eigenface technique uses a strong combination of linear algebra and statistical analysis to generate a set of basis faces-the Eigenfaces-against which inputs are tested. This project seeks to take in a large set of images of a group of known people and upon inputting an unknown face image, quickly and effectively determine whether or not it matches a known individual.

The following modules will provide a walk through exactly how this goal is achieved. Since this was not the first attempt at automated face recognition it is important to see what other approaches have been tried to appreciate the speed and accuracy of Eigenfaces. This is not a simple and straightforward problem; so many different questions must be considered as one learns about this face recognition approach.

With a basic understanding achieved it is time for the real stuff, the implementation of the procedure. This has been broken down into smaller, more manageable steps. First the set of basis Eigenfaces must be derived from a set of initial images (Obtaining the Eigenface Basis). With this basis known individuals can be processed in order to prepare the system for detection by setting thresholds (Thresholds for Eigenface Recognition) and computing matrices of weights (Face Detection Using Eigenfaces). Finally, with such a system in place, tests of robustness can be performed in order to determine what quality of input images is necessary in order for successful identification to take place (Results of Eigenface Detection Tests). In this way, relevant conclusions (Conclusions for Eigenface Detection) can be drawn about the overall efficacy of the eigenface recognition method.

Figure 1 **Face Detection System**



THE PROBLEM OF FACE RECOGNITION

Face recognition is a very interesting quandary. Ideally a face detection system should be able to take a new face and return a name identifying that person. Mathematically, what possible approach would be robust and fairly computationally economical? If we have a database of people, every face has special features that define that person. Greg may have a wider forehead, while Jeff has a scar on his right eyebrow from rugby match as a young tuck. One technique may be to go through every person in the database and characterize it by these small features. Another possible approach would be to take the face image as a whole identity.

Statistically, faces can also be very similar. Walking through a crowd without glasses, blurry vision can often result in misidentifying someone, thus yielding an awkward encounter. The statistical similarities between faces give way to an identification approach that uses the full face. Using standard image sizes and the same initial conditions, a system can be built that looks at the statistical relationship of individual pixels. One person may have a greater distance between his or her eyes then another, so two regions of pixels will be correlated to one another differently for image sets of these two people.

From a signal processing perspective the face recognition problem essentially boils down to the identification of an individual based on an array of pixel intensities. Using only these input values and whatever information can be gleaned from other images of known individuals the face recognition problem seeks to assign a name to an unknown set of pixel intensities.

Characterizing the dependencies between pixel values becomes a statistical signal processing problem. The eigenface technique finds a way to create ghost-like faces that represent the majority of variance in an image database. Our system takes advantage of these similarities between faces to create a fairly accurate and computationally "cheap" face recognition system.

FACE RECOGNITION BACKGROUND

The intuitive way to do face recognition is to look at the major features of the face and compare them to the same features on other faces. The first attempts to do this began in the 1960's with a semi-automated system. Marks were made on photographs to locate the major features; it used features such as eyes, ears, noses, and mouths. Then distances and ratios were computed from these marks to a common reference point and compared to reference data. In the early 1970's Goldstein, Harmon and Lesk created a system of 21 subjective markers such as hair color and lip thickness. This proved even harder to automate due to the subjective nature of many of the measurements still made completely by hand.

A more automated approach to recognition began with Fisher and Elschlagerb just a few years after the Goldstein paper. This approach measured the features above using templates of features of different pieces of the face and them mapped them all onto a global template. After continued research it was found that these features do not contain enough unique data to represent an adult face.

Another approach is the Connectionist approach, which seeks to classify the human face using a combination of both range of gestures and a set of identifying markers. This is usually implemented using 2-dimensional pattem recognition and neural net principles. Most of the time this approach requires a huge number of training faces to achieve decent accuracy; for that reason it has yet to be implemented on a large scale.

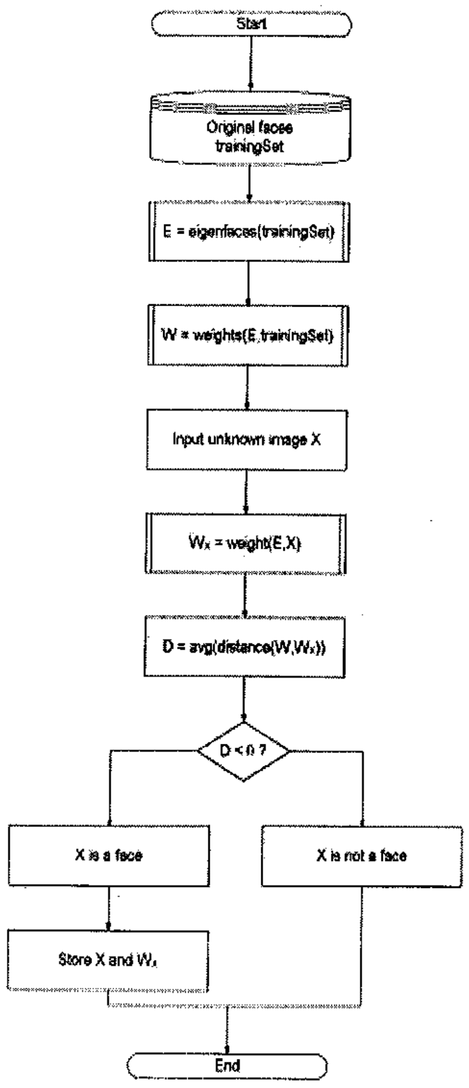
The first fully automated system to be developed utilized very general pattern recognition. It compared faces to a generic face model of expected features and created a series of patters for an image relative to this model. This approach is mainly statistical and relies on histograms and the grayscale value.

Kirby and Sirovich pioneered the eigenface approach in 1988 at Brown University. Since then, many people have built and expanded on the basic ideas described in their original paper. We received the idea for our approach from a paper by Turk and Pentland based on similar research conducted at MIT.

PROCESS DESCRIPTION

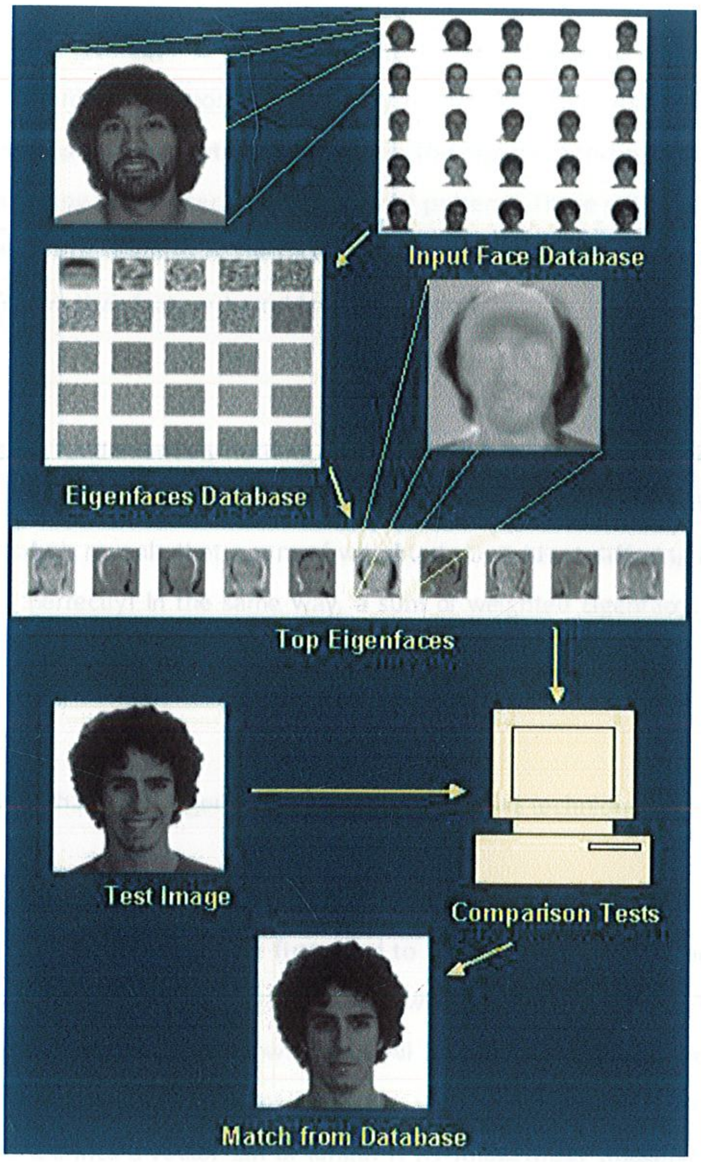
The eigenface face recognition system can be divided into two main segments: creation of the eigenface basis and recognition, or detection, of a new face.

Figure 2**Flow Diagram for Face Recognition System**



Start
Original faces trainingSet
E = eigenfaces(trainingSet)
W = weights(E,trainingSet)
Input unknown image X
W_x = weight(E,X)
D = avg(distance(W,W_x))
D < θ ?
X is a face
X is not a face
Store X and W_x
End

Figure 3 **Summary of Overall Face Recognition Process**



EIGENFACE BASIS

The eigenface technique is a powerful yet simple solution to the face recognition dilemma. In fact, it is really the most intuitive way to classify a face. As we have shown, old techniques focused on particular features of the face. The eigenface technique uses much more information by classifying faces based on general facial patterns. These patterns include, but are not limited to, the specific features of the face. By using more information, eigenface analysis is naturally more effective than feature-based face recognition.

Eigenfaces are fundamentally nothing more than basis vectors for real faces. This can be related directly to one of the most fundamental concepts in electrical engineering: Fourier analysis. Fourier analysis reveals that a sum of weighted sinusoids at differing frequencies can recompose a signal perfectly! In the same way, a sum of weighted Eigenfaces can seamlessly reconstruct a specific person's face.

Determining what these Eigenfaces are is the crux of this technique.

Before finding the Eigenfaces, we first need to collect a set of face images. These face images become our database of known faces. We will later determine whether or not an unknown face matches any of these known faces. All face images must be the same size (in pixels), and for our purposes, they must be grayscale, with values ranging from 0 to 255. Each face image is converted into a vector □n of length N (N=imagewidth*imageheight). The most useful face sets have multiple images per person. This sharply increases accuracy, due to the increased information available on each known individual. We will call our collection of faces _face space._ This space is of dimension N.

Next we need to calculate the average face in face space. Here M is the number of faces in our set:

1.

$$\Psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma_n$$

We then compute each face's difference from the average:

2.

$$\Phi_i = \Gamma_i - \Psi$$

We use these differences to compute a covariance matrix (C) for our dataset. The covariance between two sets of data reveals how much the sets correlate.

3.

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = \frac{1}{M}\sum_{n=1}^{M}\begin{pmatrix} \mathrm{var}(p_1) & \dots & \mathrm{cov}(p_1,p_N) \\ \vdots & \ddots & \vdots \\ \mathrm{cov}(p_N,p_1) & \cdots & \mathrm{var}(p_N) \end{pmatrix}_n = AA^T$$

The eigenfaces that we are looking for are simply the eigenvectors of C. However, since C is of dimension N (the number of pixels in our images), solving for the eigenfaces gets complex very quickly. Eigenface face recognition would not be possible if we had to do this.

SIMPLIFYING THE INITIAL EIGENFACE BASIS

Based on a statistical technique known as Principal Component Analysis (PCA), we can reduce the number of eigenvectors for our covariance matrix from N (the number of pixels in our image) to M (the number of images in our dataset).

PCA (PRINCIPAL COMPONENT ANALYSIS)

The Principal Component Analysis (PCA) is one of the most successful techniques that have been used in image recognition and compression. PCA is a statistical method under the broad title of *factor analysis*. The purpose of PCA is to reduce the large dimensionality of the data space (observed variables) to the smaller intrinsic dimensionality of feature space (independent variables), which are needed to describe the data economically. This is the case when there is a strong correlation between observed variables.

The jobs which PCA can do are prediction, redundancy removal, feature extraction, data compression, etc. Because PCA is a classical technique which can do something in the linear domain, applications having linear models are suitable, such as signal processing, image processing, system and control theory, communications, etc.

Face recognition has many applicable areas. Moreover, it can be categorized into face identification, face classification, or sex determination. The most useful applications contain crowd surveillance, video content indexing, personal identification (ex. driver's license), mug shots matching, entrance security, etc. 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 principal components of the feature space. This can be called eigenspace projection. Eigenspace is calculated by identifying the eigenvectors of the covariance matrix derived from a set of facial images(vectors).

In general, PCA is used to describe a large dimensional space with a relative small set of vectors. It is a popular technique for finding patterns in data of high dimension, and is used commonly in both face recognition and image compression. PCA is applicable to face recognition because face images usually are very similar to each other (relative to images of non-faces) and clearly share the same general pattern and structure.

PCA tells us that since we have only M images, we have only M non-trivial eigenvectors. We can solve for these eigenvectors by taking the eigenvectors of a new M x M matrix:

4.

$$L = A^T A$$

Because of the following math trick:

5.

$$A^T A v_i = \mu_i v_i$$
$$A A^T A v_i = \mu_i A v_i$$

Where vi is an eigenvector of L. From this simple proof we can see that Avi is an eigenvector of C. The M eigenvectors of L are finally used to form the M eigenvectors ul of C that form our eigenface basis:

6.

$$u_l = \sum_{k=1}^{M} v_{lk} \Phi_k$$

It turns out that only M-k eigenfaces are actually needed to produce a complete basis for the face space, where k is the number of unique individuals in the set of known faces.

In the end, we get a decent reconstruction of the image using only a few eigenfaces (M'), where M' usually ranges anywhere from .1M to .2M. These correspond to the vectors with the highest eigenvalues and represent the most variance within face space.

These eigenfaces provide a small yet powerful basis for face space. Using only a weighted sum of these eigenfaces, it is possible to reconstruct each face in the dataset. Yet the main application of eigenfaces, face recognition, takes this one step further.

First take all the mean subtracted images in the database and project them onto the face space. This is essentially the dot product of each face image with one of the eigenfaces. Combining vectors as matrices, one can get a weight matrix (M*N, N is total number of images in the database)

7.

$$\omega_k = \mu_k \left(\Gamma_{new} - \Psi\right)$$

$$\Omega^T = [\omega_1 \omega_2 ... \omega_{M'}]$$

$$WeightMatrix = \begin{pmatrix} \omega_{11} & \cdots & \omega_{1n} \\ \vdots & \ddots & \vdots \\ \omega_{m'1} & \cdots & \omega_{m'n} \end{pmatrix}$$

An incoming image can similarly be projected onto the face space. This will yield a vector in M dimensional space. M again is the number of used Eigenfaces. Logically, faces of the same person will map fairly closely to one another in this face space. Recognition is simply a problem of finding the closest database image, or mathematically finding the minimum Euclidean distance between a test point and a database point.

A face can reconstructed by using its feature, WTvector and previous Eigenfaces, $Um¢$ as :

8.

$$\Gamma' = \Psi + \Phi_f$$

$$\text{where } \Phi_f = \sum_{i=1}^{m'} w_i U_i$$

Averaging Technique

Within a given database, all weight vectors of a like person are averaged together. This creates a "face class" where an even smaller weight matrix represents the general faces of the entire system. When a new image comes in, its weight vector is created by projecting it onto the face space. The face is then matched to the face class that minimizes the euclidean distance. A 'hit' is counted if the image matches correctly its own face class. A 'miss' occurs if the minimum distance matches to a face class of another person. For example, the ATT database has four hundred images total, composed of forty people with ten images each.The averaging technique thus yields a weight matrix with forty vectors (forty distinct face classes).

Removal Technique

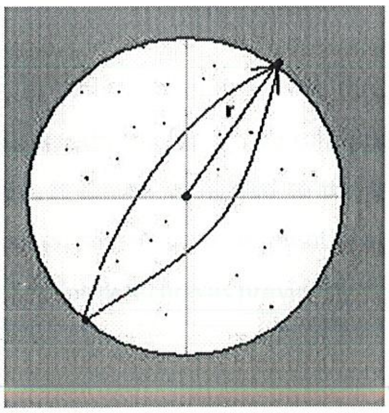
This procedure varies only slightly from the averaging technique in one key way. The weight matrix represents the image projection vectors for images of the entire database. For empirical results, an image is removed from the system, and then projected onto the face space. The resulting weight vector is then compared to the weight vector of all images. The image is then matched to the face image that minimizes the Euclidean distance. A 'hit' is counted if the tested image matches closest to another image of the same person. A 'miss' occurs when the image matches to any image of a different person. The main difference from the average technique is the number of possible images that the test face can match to that will still result in a hit. For the ATT database, a weight matrix with four hundred vectors is used, but a new image could potentially 'hit' to ten distinct faces.

THRESHOLDS FOR EIGENFACE RECOGNITION

When a new image comes into the system, there are three special cases for recognition.

- Image is a known face in the database
- Image is a face, but of an unknown person
- Image is not a face at all. May be a coke can, a door, or an animal.

For a real system, where the pictures are of standard format like a driver's license photo, the first two cases are useful. In general, the case where one tries to identify a random picture, such a slice of pizza, with a set of faces images is pretty unrealistic. Nonetheless, one can still define these threshold values to characterize the images. Looking back at the weight matrix of values using M Eigenfaces, let's define the face space as an M dimensional sphere encompassing all weight vectors in the entire database. A fairly approximate radius of this face space will then be half the diameter of this sphere, or mathematically, half the distance between the furthest points in the sphere.



$$\theta_{threshold} = \frac{1}{2}\max\left(\sqrt{||\Omega - \Omega_k||^2}\right)$$

To judge whether a new image falls within this radius, let's calculate the reconstruction error between the image and its reconstruction using M Eigenfaces. If the image projects fairly well onto the face space (image follows a face distribution), then the error will be small. However a non face image will almost always lie outside the radius of the face space.

$$\Phi_{recon} = \sum_{i=1}^{M} \omega_i \mu_i$$

$$\epsilon^2 = ||\Phi_{image} - \Phi_{recon}||^2$$

$$\epsilon > \theta_{threshold}$$

If the resulting reconstruction error is greater than the threshold, then the tested image probably is not a face image. Similar thresholds can be calculated for images of like faces. If a image passes the initial face test, it can be compared to the threshold values of faces in the database. A similar match process can be used as mentioned earlier. Also the removal or averaging technique can be applied for detection as previously described.

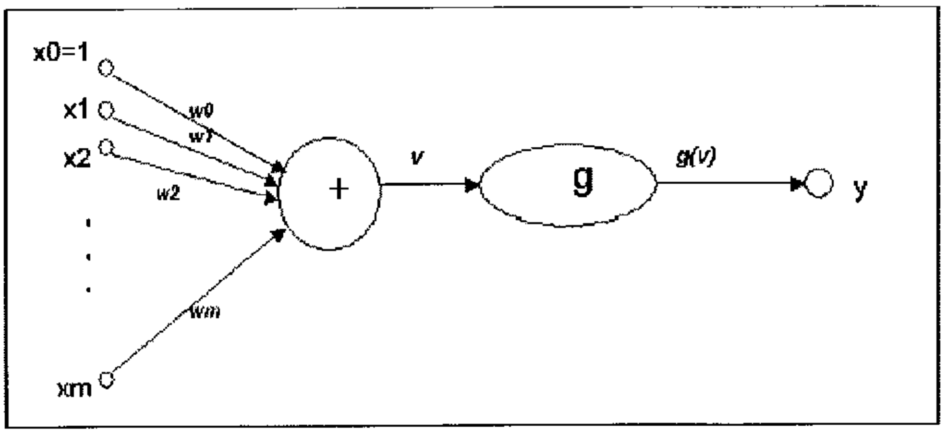
NEURAL NETWORK

Artificial neural networks are relatively crude electronic networks of "neurons" based on the neural structure of the brain. They process records one at a time, and "learn" by comparing their classification of the record (which, at the outset, is largely arbitrary) with the known actual classification of the record. The errors from the initial classification of the first record is fed back into the network, and used to modify the networks algorithm the second time around, and so on for many iterations.

Roughly speaking, a neuron in an artificial neural network is

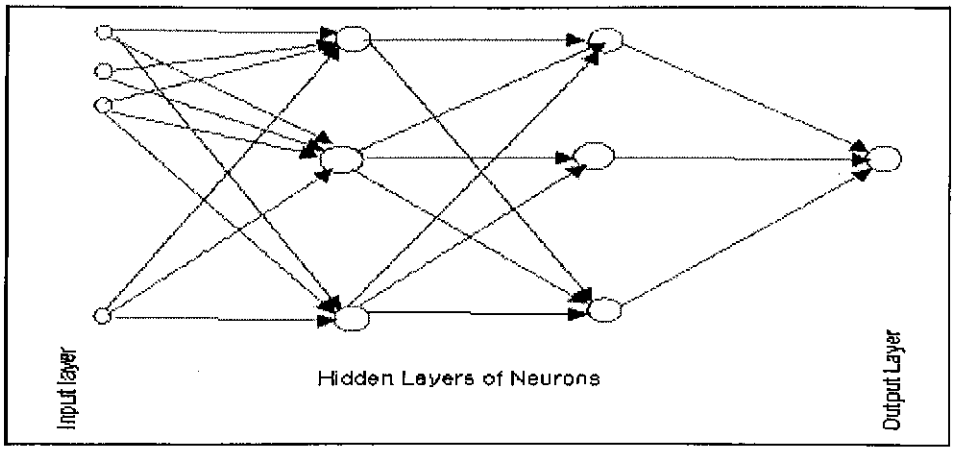
1. A set of input values (xi) and associated weights (wi)
2. A function (g) that sums the weights and maps the results to an output (y).

Figure 4**Artificial Neuron Network**



Neurons are organized into layers.

Figure 5**Layers of Neuron Network**



The input layer is composed not of full neurons, but rather consists simply of the values in a data record, that constitutes inputs to the next layer of neurons. The next layer is called a hidden layer; there may be several hidden layers. The final layer is the output layer, where there is one node for each class. A single sweep forward through the network results in the assignment of a value to each output node, and the record is assigned to whichever class's node had the highest value.

Training an Artificial Neural Network

In the training phase, the correct class for each record is known (this is termed supervised training), and the output nodes can therefore be assigned "correct" values -- "1" for the node corresponding to the correct class, and "0" for the others. (In practice it has been found better to use values of 0.9 and 0.1, respectively.) It is thus possible to compare the network's calculated values for the output nodes to these "correct" values, and calculate an error term for each node (the "Delta" rule). These error terms are then used to adjust the weights in the hidden layers so that, hopefully, the next time around the output values will be closer to the "correct" values.

The Iterative Learning Process

A key feature of neural networks is an iterative learning process in which data cases (rows) are presented to the network one at a time, and the weights associated with the input values are adjusted each time. After all cases are presented, the process often starts over again. During this learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of input samples. Neural network learning is also referred to as "connectionist learning," due to connections between the units. Advantages of neural networks include their high tolerance to noisy data, as well as their ability to classify patterns on which they have not been trained.Once a network has been structured for a particular application, that network is ready to be trained. To start this process, the initial weights (described in the next section) are chosen randomly. Then the training, or learning, begins.

The network processes the records in the training data one at a time, using the weights and functions in the hidden layers, then compares the resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights for application to the next record to be processed. This process occurs over and over as the weights are continually tweaked. During the training of a network the same set of data is processed many times as the connection weights are continually refined.

Feed forward, Back-Propagation

The typical back-propagation network has an input layer, an output layer, and at least one hidden layer. There is no theoretical limit on the number of hidden layers but typically there are just one or two. Some work has been done which indicates that a maximum of five layers (one input layer, three hidden layers and an output layer) are required to solve problems of any complexity. Each layer is fully connected to the succeeding layer.

As noted above, the training process normally uses some variant of the Delta Rule, which starts with the calculated difference between the actual outputs and the desired outputs. Using this error, connection weights are increased in proportion to the error times a scaling factor for global accuracy. Doing this for an individual node means that the inputs, the output, and the desired output all have to be present at the same processing element. The complex part of this learning mechanism is for the system to determine which input contributed the most to an incorrect output and how does that element get changed to correct the error. An inactive node would not contribute to the error and would have no need to change its weights. To solve this problem, training inputs are applied to the input layer of the network, and desired outputs are compared at the output layer. During the learning process, a forward sweep is made through the network, and the output of each element is computed layer by layer. The difference between the output of the final layer and the desired output is back-propagated to the previous layer(s), usually modified by the derivative of the transfer function, and the connection weights are normally adjusted using the Delta Rule. This process proceeds for the previous layer(s) until the input layer is reached.

PROGRAM CODE:

1. **Mainfile.m**

```
clc

disp('            Welcome to Face Recognition System    ');

disp('**********************************************************
********');

disp('            Press various keys for the required operation
');

disp('**********************************************************
********');

disp('            Calculate Eigenfaces............[1]');

disp('            Input Image for Recognition......[2]');

disp('            To quit without exiting matlab...[press any key
except above]');

disp('**********************************************************
********');

disp('**********************************************************
********');
```

```
disp('        You need to calculate Eigenfaces before recognition
');
disp('       Above statement is applicable only for the first
image        ');
disp('If you have already calculated the eigenfaces then just
kickstart with Recognition');
y=input('Press appropriate keys for required operation........',
's');


switch(y)
    case '1'
        imageno=input('Enter the no of data images with which you
want to continue');
        [T, m1, Eigenfaces, ProjectedImages,
imageno]=Eigenface_calculation(imageno);
        save Eigenface.mat;
        mainfile;
    case '2'
        load Eigenface.mat;
        outputimage=Recognition(T, m1, Eigenfaces,
ProjectedImages, imageno);
```

```
        disp('Press any key to continue..............');

        pause;

        mainfile;




    end

    case 'q'

        disp('Are you sure you want to quit');

        f=input('Press Y for yes or any other key for No','s');

        switch(f)

            case 'y'

                quit;

            case 'Y'

                quit;

            otherwise

               mainfile;

        end

    case 'Q'

        disp('Are you sure you want to quit');
    f=input('Press Y for yes or any other key for No','s');

    switch(f)

        case 'y'

            quit;

        case 'Y'

            quit;

        otherwise

           mainfile;

    end

     otherwise

        disp('Invalid Entry');

        disp('Press any key to continue........');

        pause;

       mainfile;

end
```

2. **Eigenface_calculation.m**

```
function [T,m1, Eigenfaces, ProjectedImages,
imageno]=Eigenface_calculation(imageno);
% we need to make it a function to increase modularity of our
program
T=imageno;
n=1;
aftermean=[];
I=[];
%figure(1);
T=imageno;
for i=1:T  %used to display in single window all images selected
    imagee=strcat(int2str(i),'.jpg');% use strcat function to
call T no of images


    eval('imagg=imread(imagee);');
    subplot(ceil(sqrt(T)),ceil(sqrt(T)),i)% plot them as matrix
    imagg=rgb2gray(imagg);
    imagg=imresize(imagg,[200,180],'bilinear');
    [m n]=size(imagg);

    %imshow(imagg)

    temp=reshape(imagg',m*n,1);%to get elements along rows we
take imagg'

    I=[I temp];

end




m1=mean(I,2);

 ima=reshape(m1',n,m);% to display the eigenfaces now we need to
again take images in 200x180 format

     ima=ima';

     %figure,imshow(ima);



     for i=1:T

    temp=double(I(:,i));

    I1(:,i)=(temp-m1);% normalizing the images by substracting
each column with the mean vector

end
```

```
for i=1:T

    subplot(ceil(sqrt(T)),ceil(sqrt(T)),i);

    imagg1=reshape(I1(:,i),n,m);%to again get original size so
that we can get the value of m and n

    imagg1=imagg1';

    [m n]=size(imagg1);

    %imshow(imagg1);% displays the mean images

end




a1=[];

for i=1:T% to change the format of values to double

    te=double(I1(:,i));

    a1=[a1,te];

end

a=a1';

covv=a*a';
```

```
[eigenvec eigenvalue]=eig(covv);          %calculating eigen values

d=eig(covv);

sorteigen=[];

eigval=[];


for i=1:size(eigenvec,2);  %takes no of col of eigenvec

    if(d(i)>(0.5e+008))% we can take any value to suit our
algorithm

        % this values generally are taken by trial and error

        sorteigen=[sorteigen, eigenvec(:,i)];

        eigval=[eigval, eigenvalue(i,i)];

    end;

end;



Eigenfaces=[];

Eigenfaces=a1*sorteigen;% got matrix of principal Eigenfaces


for i=1:size(sorteigen,2)   %sorting eigenfaces
```

```
    k=sorteigen(:,i);

    tem=sqrt(sum(k.^2));

    sorteigen(:,i)=sorteigen(:,i)./tem;

end

Eigenfaces=a1*sorteigen;




for i=1:size(Eigenfaces,2)    %disabled

    ima=reshape(Eigenfaces(:,i)',n,m);% to display the eigenfaces
now we need to again take images in 200x180 format

    ima=ima';

      subplot(ceil(sqrt(T)),ceil(sqrt(T)),i)

     %imshow(ima);

end



ProjectedImages=[];

for i = 1 : T

    temp = Eigenfaces'*a1(:,i); % Projection of centered images
into facespace

    ProjectedImages = [ProjectedImages temp];
```

```
end

end
```

3. Recognition.m

```
function [outputimage]=Recognition(T,m1, Eigenfaces,
ProjectedImages, imageno)

MeanInputImage=[];

[fname pname]=uigetfile('*.jpg','Select the input image for
recognition');

InputImage=imread(fname);

InputImage=rgb2gray(InputImage);

InputImage=imresize(InputImage,[200 180],'bilinear');

%resizing of input image. This is a part of preprocessing
techniques of images




[m n]=size(InputImage);

imshow(InputImage);

Imagevector=reshape(InputImage',m*n,1);

%to get elements along rows as we take InputImage'
```

```
MeanInputImage=double(Imagevector)-m1;

ProjectInputImage=Eigenfaces'*MeanInputImage;

% here we get the weights of the input image with respect to
our eigenfaces

% next we need to euclidean distance of our input image and
compare it

% with our face space and check whether it matches the
answer...we need

% to take the threshold value by trial and error methods

Euclideandistance=[];


for i=1:T

    temp=ProjectedImages(:,i)-ProjectInputImage;

    Euclideandistance=[Euclideandistance temp];

end


% the above statements will get you a matrix of Euclidean
distance and you

% need to normalize it and then find the minimum Euclidean
distance
```

```
tem=[];

for i=1:size(Euclideandistance,2)

    k=Euclideandistance(:,i);

    tem(i)=sqrt(sum(k.^2));

end



% We now set some threshold values to know whether the image
is face or not

% and if it is a face then if it is known face or not

% The threshold values taken are done by trial and error
methods

[MinEuclid, index]=min(tem);

if(MinEuclid<0.8e008)    %values found by trial

if(MinEuclid<0.35e008)

    outputimage=(strcat(int2str(index),'.jpg'));

    figure,imshow(outputimage);

    switch index % we are entering the name of the persons in
the code itself

        % There is no provision of entering the name in real
time
```

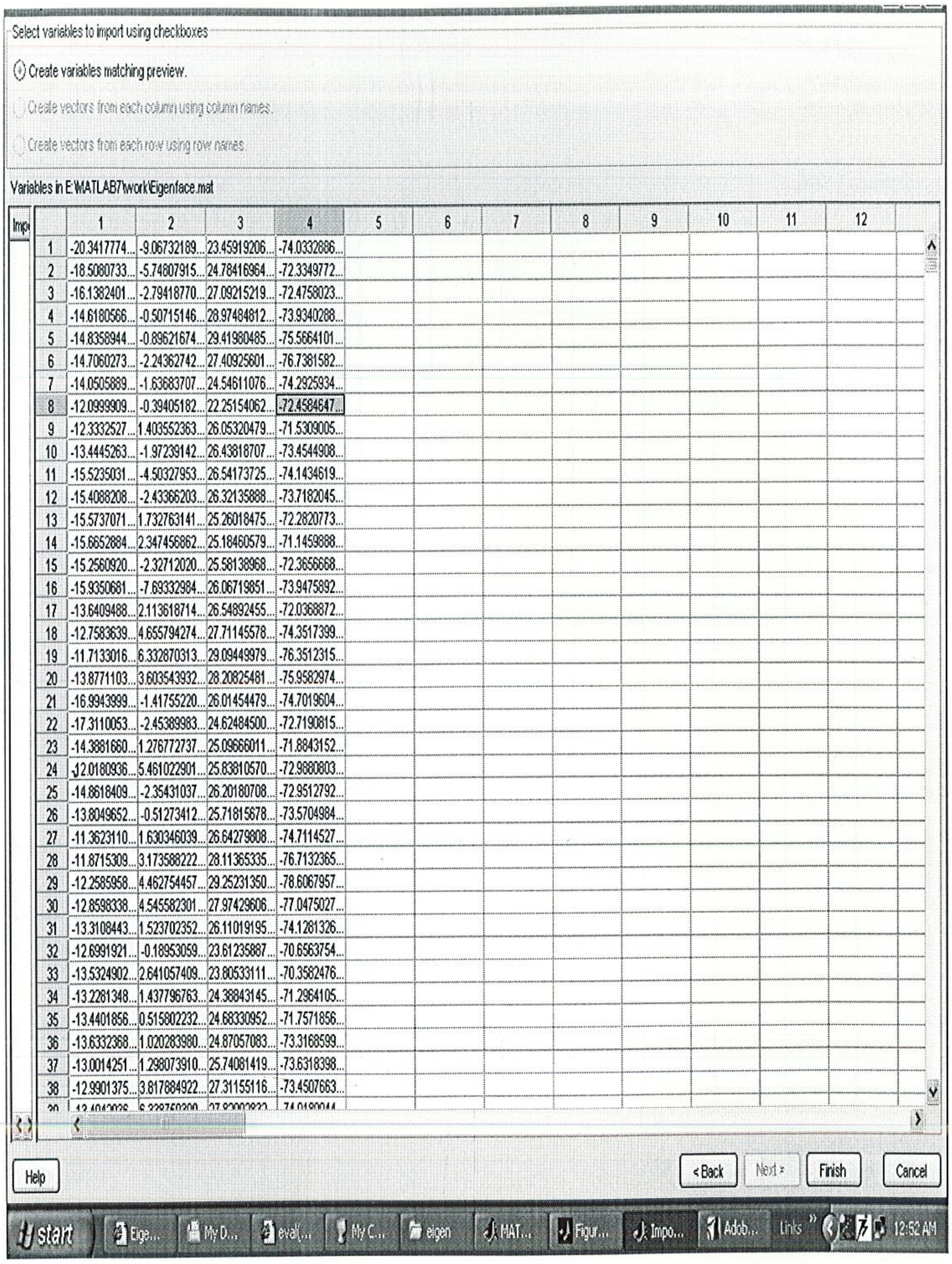
```
case 1

    disp('Jonathan Swift');

    disp('Age=22');

case 2

    disp('Eliyahu Goldratt');

    disp('Age=25');

case 3

    disp('Anpage');

    disp('Age=35');

case 4

    disp('Rizwana');

    disp('Age=30');

case 5

    disp('Rihana');

    disp('Age=48');

case 6

    disp('Seema');

    disp('Age=19');

case 7
```

```
    disp('Kasana');

    disp('Age=27');

case 8

    disp('Hanifa');

    disp('Age=33');

case 9

    disp('Alefiya');

    disp('Age=22');

case 10

    disp('Mamta');

    disp('Age=50');

case 11

    disp('Mayawati');

    disp('Age=39');

case 12

    disp('Elizabeth');

    disp('Age=87');

case 13

    disp('Cecelia Ahern');
```

```
        disp('Age=78');
    case 14
        disp('Shaista Khatun');
        disp('Age=56');
    case 15
        disp('Rahisa Khatun');
        disp('Age=45');
    case 16
        disp('Ruksana');
        disp('Age=64');
    case 17
        disp('Parizad Zorabian');
        disp('Age=38');
    case 18
        disp('Heena kundanani');
        disp('Age=20');
    case 19
disp('Setu Savani');
           disp('Age=21');
```

```
            case 20

                disp('Mohd Zubair Saifi');

                disp('Age=20');

            otherwise

               disp('Image in database but name unknown')

        end


    else

        disp('No matches found');

        disp('You are not allowed to enter this system');

        outputimage=0;

    end

    else

        disp('Image is not even a face');

        outputimage=0;

    end

    end
```

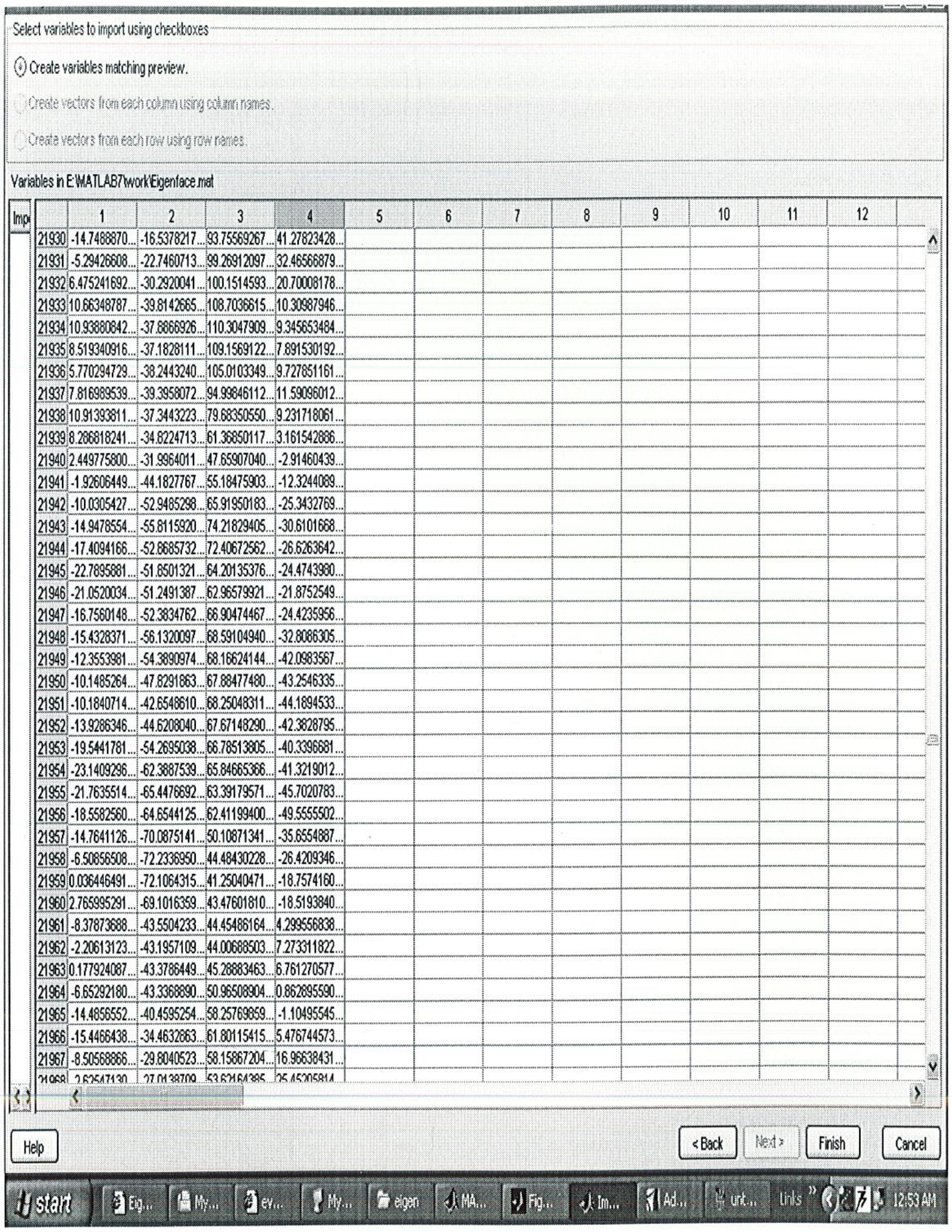
Figure 6


Select variables to import using checkboxes
[✓ Create variables matching preview.]
Create vectors from each column using column names.
Create vectors from each row using row names.

Variables in E:\MATLAB7\work\Eigenface.mat

	1	2	3	4	5	6	7	8	9	10	11	12
1	-20.3417774...	-9.06732189...	23.45919206...	-74.0332886...								
2	-18.5080733...	-5.74807915...	24.78416964...	-72.3349772...								
3	-16.1382401...	-2.79418770...	27.09215219...	-72.4758023...								
4	-14.6180566...	-0.50715146...	28.97484812...	-73.9340288...								
5	-14.8358944...	-0.89621674...	29.41980485...	-75.5664101...								
6	-14.7060273...	-2.24362742...	27.40925601...	-76.7381582...								
7	-14.0505889...	-1.63683707...	24.54611076...	-74.2925934...								
8	-12.0999909...	-0.39405182...	22.25154062...	-72.4584647...								
9	-12.3332527...	1.403552363...	26.05320479...	-71.5309005...								
10	-13.4445263...	-1.97239142...	26.43818707...	-73.4544908...								
11	-15.5235031...	-4.50327953...	26.54173725...	-74.1434619...								
12	-15.4088208...	-2.43366203...	26.32135888...	-73.7182045...								
13	-15.5737071...	1.732763141...	25.26018475...	-72.2820773...								
14	-15.6652884...	2.347456862...	25.18460579...	-71.1459888...								
15	-15.2560920...	-2.32712020...	25.58138968...	-72.3656668...								
16	-15.9350681...	-7.69332984...	26.06719851...	-73.9475892...								
17	-13.6409488...	2.113618714...	26.54892455...	-72.0368872...								
18	-12.7583639...	4.655794274...	27.71145578...	-74.3517399...								
19	-11.7133016...	6.332870313...	29.09449979...	-76.3512315...								
20	-13.8771103...	3.603543932...	28.20825481...	-75.9582974...								
21	-16.9943999...	-1.41755220...	26.01454479...	-74.7019604...								
22	-17.3110053...	-2.45389983...	24.62484500...	-72.7190815...								
23	-14.3881660...	1.276772737...	25.09666011...	-71.8843152...								
24	-12.0180936...	5.461022901...	25.83810570...	-72.9880803...								
25	-14.8618409...	-2.35431037...	26.20180708...	-72.9512792...								
26	-13.8049652...	-0.51273412...	25.71815678...	-73.5704984...								
27	-11.3623110...	1.630346039...	26.64279808...	-74.7114527...								
28	-11.8715309...	3.173588222...	28.11365335...	-76.7132365...								
29	-12.2585958...	4.462754457...	29.25231350...	-78.6067957...								
30	-12.8598338...	4.545582301...	27.97429606...	-77.0475027...								
31	-13.3108443...	1.523702352...	26.11019195...	-74.1281326...								
32	-12.6991921...	-0.18953059...	23.61235887...	-70.6563754...								
33	-13.5324902...	2.641057409...	23.80533111...	-70.3582476...								
34	-13.2281348...	1.437796763...	24.38843145...	-71.2964105...								
35	-13.4401856...	0.515802232...	24.68330952...	-71.7571656...								
36	-13.6332368...	1.020283980...	24.87057083...	-73.3168599...								
37	-13.0014251...	1.298073910...	25.74081419...	-73.6318398...								
38	-12.9901375...	3.817884922...	27.31155116...	-73.4507663...								

Figure 7


Select variables to import using checkboxes

[✓ Create variables matching preview.]

Create vectors from each column using column names.

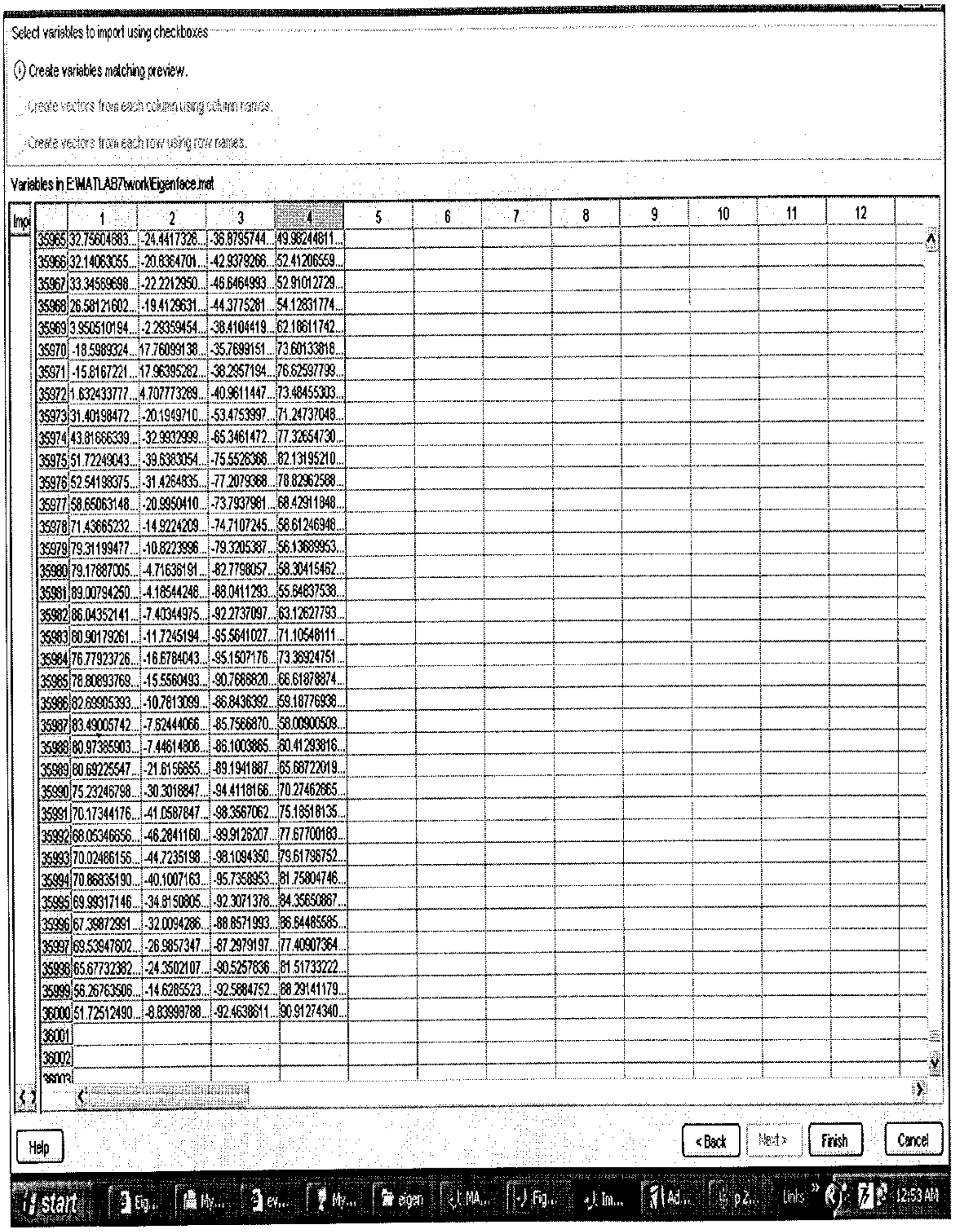
Create vectors from each row using row names.

Variables in E:\MATLAB7\work\Eigenface.mat

Imp		1	2	3	4	5	6	7	8	9	10	11	12
	21930	-14.7488870...	-16.5378217...	93.75569267...	41.27823428...								
	21931	-5.29426608...	-22.7460713...	99.26912097...	32.46566879...								
	21932	6.475241692...	-30.2920041...	100.1514593...	20.70008178...								
	21933	10.66348787...	-39.8142665...	108.7036615...	10.30987946...								
	21934	10.93880842...	-37.8866926...	110.3047909...	9.345653484...								
	21935	8.519340916...	-37.1828111...	109.1569122...	7.891530192...								
	21936	5.770294729...	-38.2443240...	105.0103349...	9.727851161...								
	21937	7.816989539...	-39.3958072...	94.99846112...	11.59096012...								
	21938	10.91393811...	-37.3443223...	79.68350550...	9.231718061...								
	21939	8.286818241...	-34.8224713...	61.36850117...	3.161542886...								
	21940	2.449775800...	-31.9964011...	47.65907040...	-2.91460439...								
	21941	-1.92606449...	-44.1827767...	55.18475903...	-12.3244089...								
	21942	-10.0305427...	-52.9485298...	65.91950183...	-25.3432769...								
	21943	-14.9478554...	-55.8115920...	74.21829405...	-30.6101668...								
	21944	-17.4094166...	-52.8685732...	72.40672562...	-26.6263642...								
	21945	-22.7895881...	-51.8501321...	64.20135376...	-24.4743980...								
	21946	-21.0520034...	-51.2491387...	62.96579921...	-21.8752549...								
	21947	-16.7560148...	-52.3834762...	66.90474467...	-24.4235956...								
	21948	-15.4328371...	-56.1320097...	68.59104940...	-32.8086305...								
	21949	-12.3553981...	-54.3890974...	68.16624144...	-42.0983567...								
	21950	-10.1485264...	-47.8291863...	67.88477480...	-43.2546335...								
	21951	-10.1840714...	-42.6548610...	68.25048311...	-44.1894533...								
	21952	-13.9286346...	-44.6208040...	67.67148290...	-42.3828795...								
	21953	-19.5441781...	-54.2695038...	66.78513805...	-40.3396681...								
	21954	-23.1409296...	-62.3887539...	65.84665366...	-41.3219012...								
	21955	-21.7635514...	-65.4476692...	63.39179571...	-45.7020783...								
	21956	-18.5582560...	-64.6544125...	62.41199400...	-49.5555502...								
	21957	-14.7641126...	-70.0875141...	50.10871341...	-35.6554887...								
	21958	-6.50856508...	-72.2336950...	44.48430228...	-26.4209346...								
	21959	0.036446491...	-72.1064315...	41.25040471...	-18.7574160...								
	21960	2.765995291...	-69.1016359...	43.47601810...	-18.5193840...								
	21961	-8.37873688...	-43.5504233...	44.45486164...	4.299556838...								
	21962	-2.20613123...	-43.1957109...	44.00688503...	7.273311822...								
	21963	0.177924087...	-43.3786449...	45.28883463...	6.761270577...								
	21964	-6.65292180...	-43.3368890...	50.96508904...	0.862895590...								
	21965	-14.4856552...	-40.4595254...	58.25769859...	-1.10495545...								
	21966	-15.4466438...	-34.4632863...	61.80115415...	5.476744573...								
	21967	-8.50568866...	-29.8040523...	58.15867204...	16.96638431...								
	21968	2.62547130	27.0138709	53.62164385	25.45205814								

44

Figure 8



Select variables to import using checkboxes

(•) Create variables matching preview.

( ) Create vectors from each column using column names.

( ) Create vectors from each row using row names.

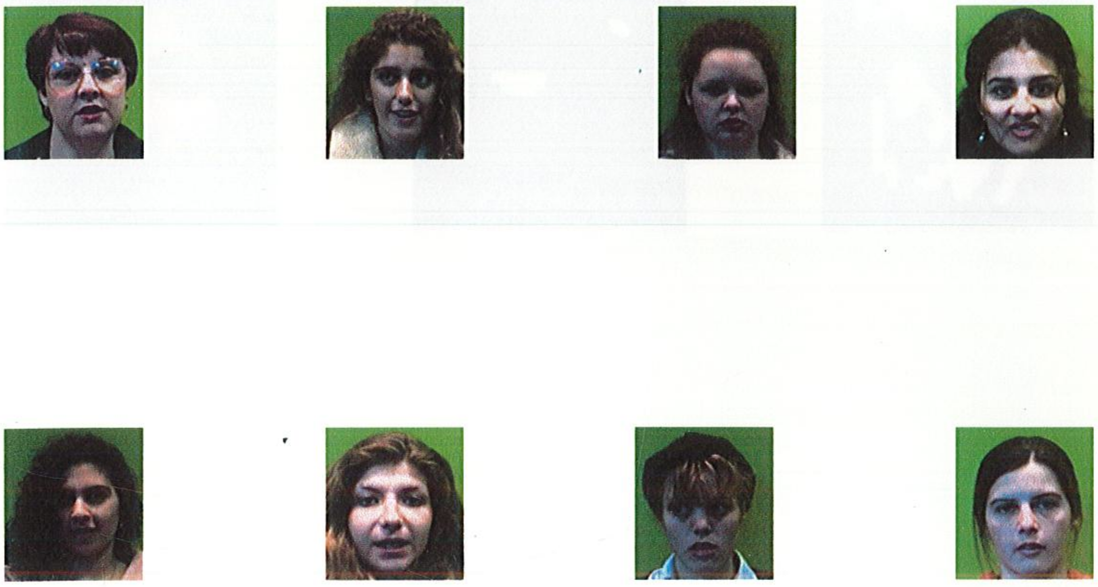
Variables in E:\MATLAB7\work\Eigenface.mat

	1	2	3	4	5	6	7	8	9	10	11	12
35965	32.75604683...	-24.4417328...	-36.8795744...	49.98244811...								
35966	32.14063055...	-20.8364701...	-42.9379266...	52.41206559...								
35967	33.34569698...	-22.2212950...	-46.6464993...	52.91012729...								
35968	26.58121602...	-19.4129631...	-44.3775281...	54.12831774...								
35969	3.950510194...	-2.29359454...	-38.4104419...	62.18611742...								
35970	-18.5989324...	17.76099138...	-35.7699151...	73.60133818...								
35971	-15.8167221...	17.96395282...	-38.2957194...	76.62597799...								
35972	1.632433777...	4.707773289...	-40.9611447...	73.48455303...								
35973	31.40198472...	-20.1949710...	-53.4753997...	71.24737048...								
35974	43.81666339...	-32.9932999...	-65.3461472...	77.32654730...								
35975	51.72249043...	-39.6383054...	-75.5526366...	82.13195210...								
35976	52.54198375...	-31.4264835...	-77.2079368...	78.82962588...								
35977	58.65063148...	-20.9950410...	-73.7937981...	68.42911848...								
35978	71.43665232...	-14.9224209...	-74.7107245...	58.61246948...								
35979	79.31199477...	-10.8223996...	-79.3205387...	56.13689953...								
35980	79.17887005...	-4.71636191...	-82.7798057...	58.30415462...								
35981	89.00794250...	-4.18544248...	-88.0411293...	55.64837538...								
35982	86.04352141...	-7.40344975...	-92.2737097...	63.12627793...								
35983	80.90179261...	-11.7245194...	-95.5641027...	71.10548111...								
35984	76.77923726...	-16.6784043...	-95.1507176...	73.36924751...								
35985	78.80893769...	-15.5560493...	-90.7666820...	66.61878874...								
35986	82.69905393...	-10.7813099...	-86.8436392...	59.18776938...								
35987	83.49005742...	-7.62444066...	-85.7566870...	58.00900509...								
35988	80.97385903...	-7.44614808...	-86.1003865...	60.41293816...								
35989	80.69225547...	-21.6156655...	-89.1941887...	65.68722019...								
35990	75.23246798...	-30.3018847...	-94.4118166...	70.27462865...								
35991	70.17344176...	-41.0587847...	-98.3567062...	75.18518135...								
35992	68.05346656...	-46.2841160...	-99.9126207...	77.67700183...								
35993	70.02486156...	-44.7235198...	-98.1094350...	79.61796752...								
35994	70.86835190...	-40.1007163...	-95.7358953...	81.75804746...								
35995	69.99317146...	-34.8150805...	-92.3071378...	84.35650867...								
35996	67.39872991...	-32.0094286...	-88.8571993...	86.64485585...								
35997	69.53947602...	-26.9857347...	-87.2979197...	77.40907364...								
35998	65.67732382...	-24.3502107...	-90.5257836...	81.51733222...								
35999	56.26763506...	-14.6285523...	-92.5884752...	88.29141179...								
36000	51.72512490...	-8.83998788...	-92.4638611...	90.91274340...								
36001												
36002												

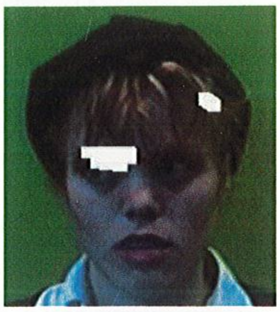
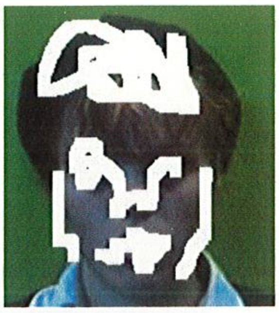
RESULT AND CONCLUSION

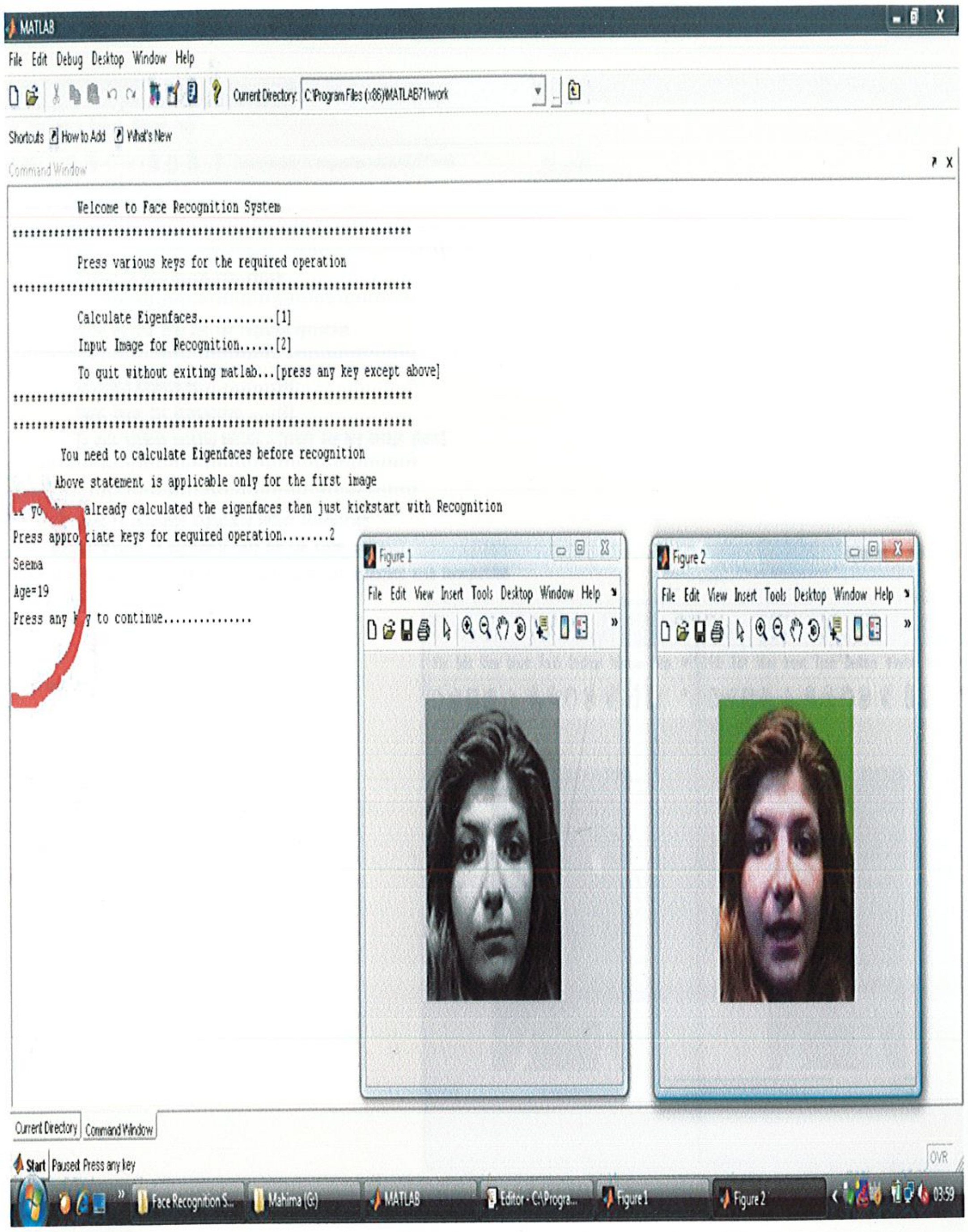
We have made use of Eigenfaces to represent the features vectors for human faces. The features are extracted from the original image to represent unique identity used as inputs to the neural network to measure similarity in classification and recognition.

IMAGE DATABASE

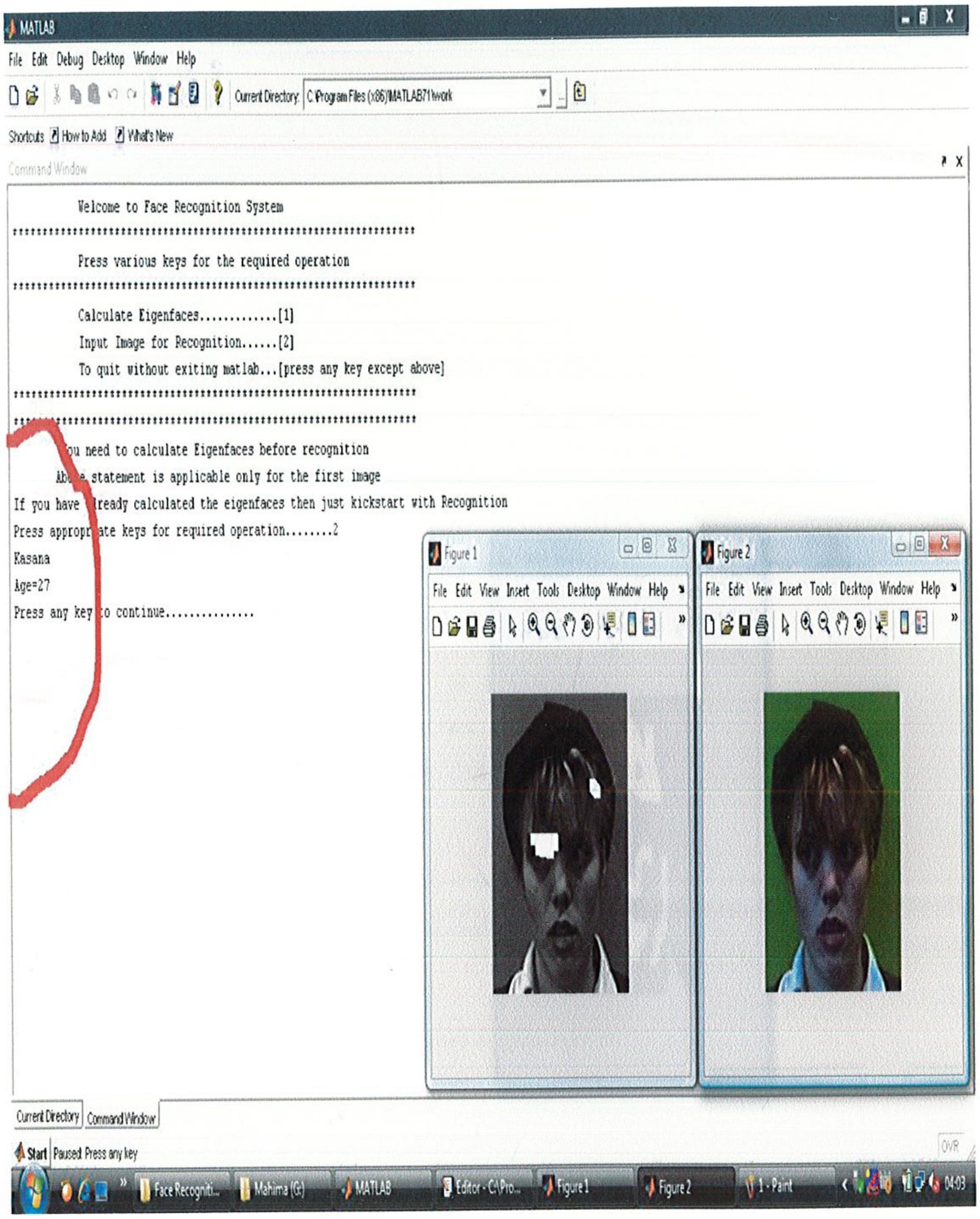


TEST IMAGES

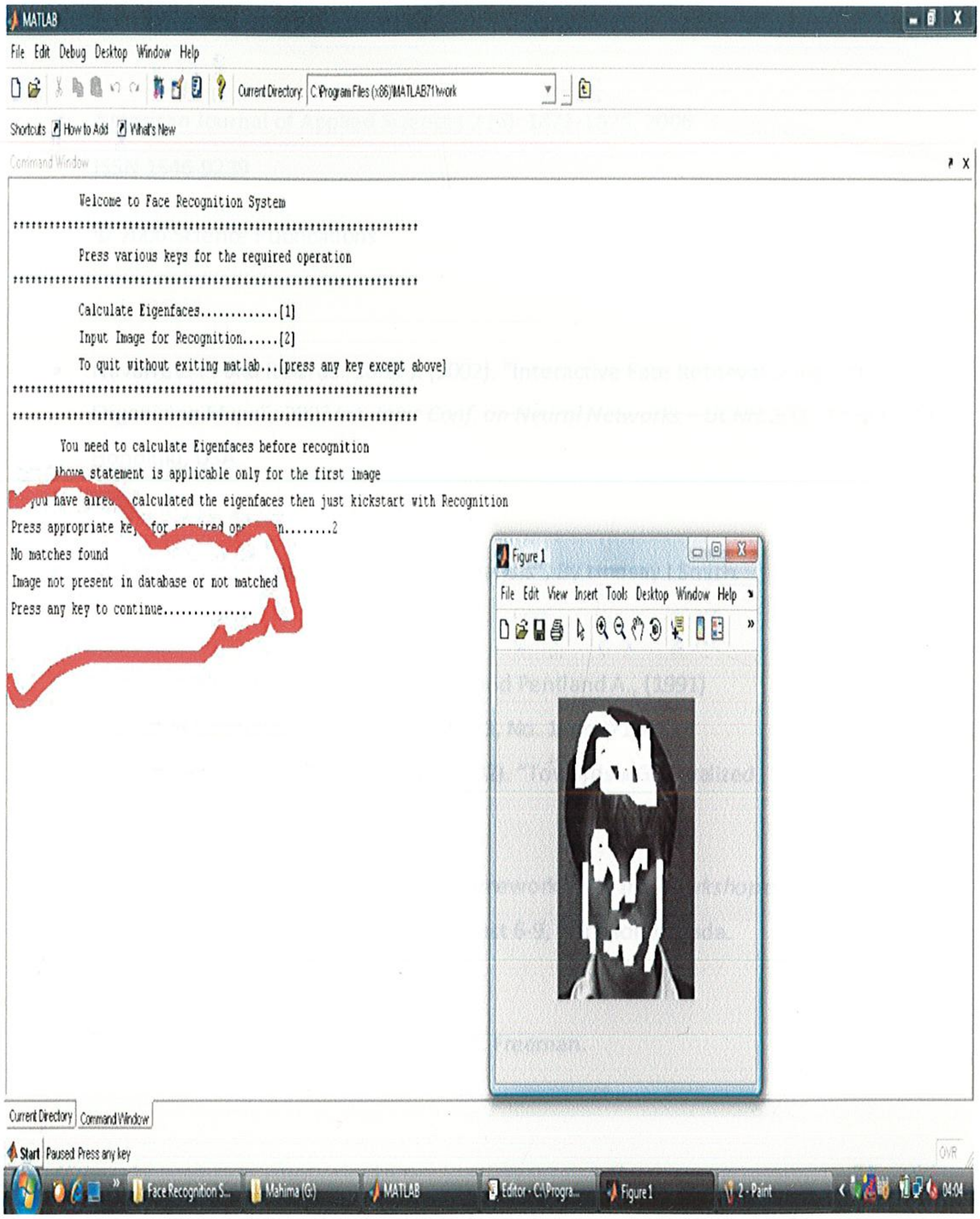




MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\Program Files (x86)\MATLAB71\work
Shortcuts How to Add What's New
Command Window
Figure 1
Figure 2
File Edit View Insert Tools Desktop Window Help
Current Directory
Command Window
Start Paused: Press any key
OVR
Face Recognition S...
Mahima (G:)
MATLAB
Editor - C:\Progra...
Figure 1
Figure 2
03:59

```
          Welcome to Face Recognition System
*************************************************************

          Press various keys for the required operation
*************************************************************

          Calculate Eigenfaces.............[1]
          Input Image for Recognition......[2]
          To quit without exiting matlab...[press any key except above]
*************************************************************
*************************************************************

       You need to calculate Eigenfaces before recognition
      Above statement is applicable only for the first image
If you have already calculated the eigenfaces then just kickstart with Recognition
Press appropriate keys for required operation........2
Seema
Age=19
Press any key to continue...............
```


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\Program Files (x86)\MATLAB71\work
Shortcuts How to Add What's New
Command Window
Figure 1
Figure 2
File Edit View Insert Tools Desktop Window Help
Current Directory
Command Window
Start Paused: Press any key
OVR
Face Recogniti...
Mahima (G:)
MATLAB
Editor - C:\Pro...
Figure 1
Figure 2
1 - Paint
04:03

```
        Welcome to Face Recognition System
*************************************************************
        Press various keys for the required operation
*************************************************************
        Calculate Eigenfaces............[1]
        Input Image for Recognition......[2]
        To quit without exiting matlab...[press any key except above]
*************************************************************
*************************************************************
        You need to calculate Eigenfaces before recognition
      Above statement is applicable only for the first image
If you have already calculated the eigenfaces then just kickstart with Recognition
Press appropriate keys for required operation........2
Kasana
Age=27
Press any key to continue..............
```


MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\Program Files (x86)\MATLAB71\work
Shortcuts How to Add What's New
Command Window
Figure 1
File Edit View Insert Tools Desktop Window Help
Current Directory Command Window
Start Paused: Press any key
Face Recognition S... Mahima (G:) MATLAB Editor - C:\Progra... Figure 1 2 - Paint
04:04

```
        Welcome to Face Recognition System
**************************************************************
        Press various keys for the required operation
**************************************************************
        Calculate Eigenfaces............[1]
        Input Image for Recognition......[2]
        To quit without exiting matlab...[press any key except above]
**************************************************************
**************************************************************
      You need to calculate Eigenfaces before recognition
      Above statement is applicable only for the first image
[illegible] you have alrea[illegible] calculated the eigenfaces then just kickstart with Recognition
Press appropriate ke[illegible] for re[illegible]ired op[illegible]n........2
No matches found
Image not present in database or not matched
Press any key to continue...............
```

BIBLIOGRAPHY

- American Journal of Applied Sciences 2 (6): 1872-1875, 2006
  ISSN 1546-9239

  © 2006 Science Publications

- Navarrete P. and Ruiz-del-Solar J. (2002), "Interactive Face Retrieval using Self-Organizing Maps", *2002 Int. Joint Conf. on Neural Networks – IJCNN 2002*, May 12-17, Honolulu, USA.

- "A tutorial on Principal Components Analysis", By Lindsay I Smith.

- "Eigenfaces for Recognition", Turk, M. and Pentland A., (1991)
  Journal of Cognitive Neuroscience, Vol. 3, No. 1, pp. 71-86.
- Ruiz-del-Solar, J., and Navarrete, P. (2002). "Towards a Generalized

- Eigenspace-based Face Recognition Framework", *4th Int. Workshop on Statistical Techniques in Pattern Recognition*, August 6-9, Windsor, Canada.

- Simulating Neural Networks by James A. Freeman.