PROJECT 1 - ECE 738
ADVANCED DIGITAL SIGNAL PROCESSING

INSTRUCTOR: PROFESSOR KATHLEEN WAGE

BY

SAI KASYAP KAMARAJU

G00845717

Introduction:

Training Set:
The Yale database:
The Yale database consists of 15 individuals each having 11 different poses.
There are 11 different poses in the database which are as follows :
Centerlight, glasses,happy, leftlight,noglasses,normal, rightlight,sad,sleepy,surprised and wink.

So in the training set given there are 11 * 15 = 165 images each image is of size
[195x161 double].

Question 1:
Write a function to select a subset of data and store the images as vectors.
Procedure:
The function inputs are face data structure, a cell of array poses and a faceid of an required individual.
If the desired poses are not given , the function considers all the poses contained in the data set.
Now to access a particular image in the dataset structure we need to use
X=facedata(faceid).(desiredposes{poseid});
Now store the images using reshape in matlab by running it in the 2 loops (one loop is length of 15 individuals and the other is for poses.)
The data is stored as shown below :

```
i = 1;
for faceid = 1:nt
   for poseid = 1:nposes
      X = facedata(faceid).(desiredposes{poseid});
      [nrow,ncol]=size(X);
   Data(:,:, i) = reshape(X,nrow,ncol);
   i = i+1;
   end
end
```

To access a particular individual we need to access the 'i' values.
The data matrix is of the form 195 x161x165. (Where 165 is the number of images in the training set).

2.2 Write a function to compute a low rank representation of image data.
Procedure:
This function takes Image set in the form of 3 –d set (which is converted back to 2 d using reshape) and selection of 'K' best eigen values (dimensionality reduction)
Firstly the mean of the image data set is found out.
To get average mean face, the mean vector is converted to 2 –d form using reshape and the image is plotted.
The Average mean is subtracted from the data set.
Then Covariance matrix of A'*A is calculated instead of A*A'
The Eigen vectors and Eigen values are calculated using svd in matlab.
The Eigen values are squared to get the actual Eigen values.
The Eigen values are sorted in the descending order using sort command in matlab.

Accordingly Eigen vectors are also sorted in descending order depending on their respective eigen values.
Eigen face vectors are product of dataminusmean matrix and EigenVectors .
The Eigen Face vectors are normalized and are truncated by using K greatest Eigen vectors
The 'K' value can be choosen from the variance vs Eigen values graph.
Here K value is taken wherever the graph touches 90 %.

3. Write a function to implement Eigen face detector / classifier.
Procedure:
The input parameters of the function are mean vector, Dataminusmean , eigen face vector and test image.
Images are projected to face space by multiplying eigen face vector and Dataminusmean(ProjImg)(Total W).
The Eigen face vector' and (Input test image – mean vector) are multiplied to get projection of test image.(Wi for each img);
The Euclidean Distance = norm (PrjtestImg-ProjImg);
Now threshold is calculated by trial and error method.

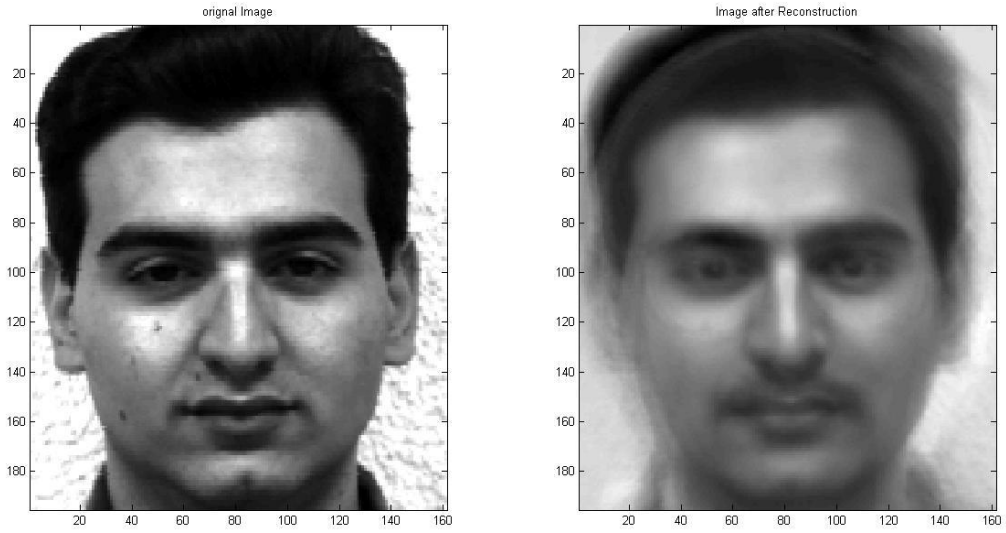
2.4 Testing my function

Part a) you reconstruct signal using low rank representation matrix. How many components does it take to produce an image that looks the person i.e you can recognize it based on simple visual inspection.

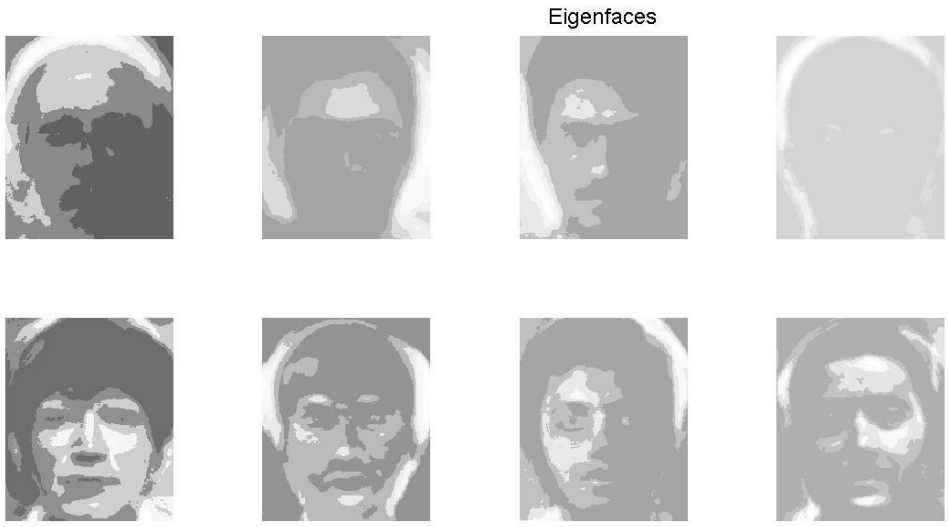
Solution

When K = 8,

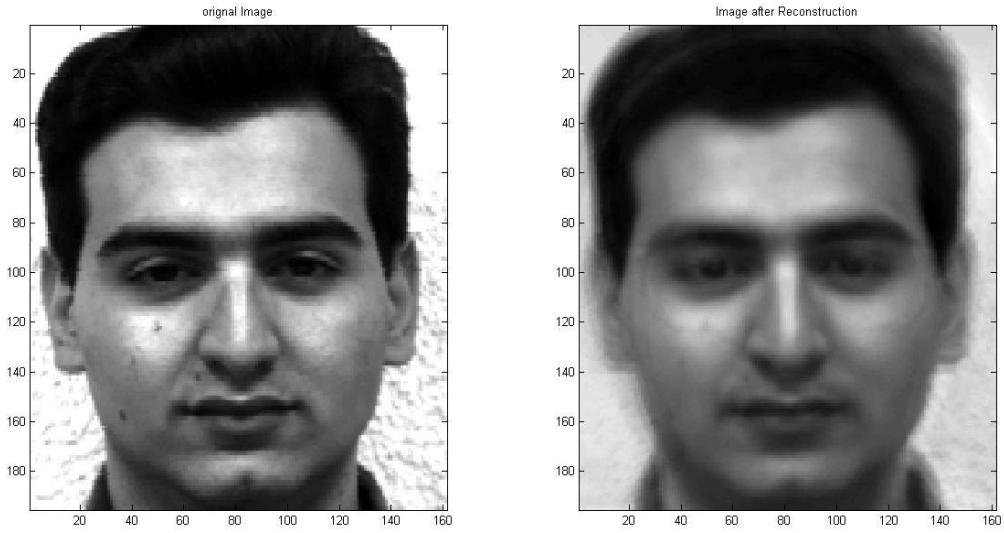
The picture of the 15th individual who poses normal is as shown below :



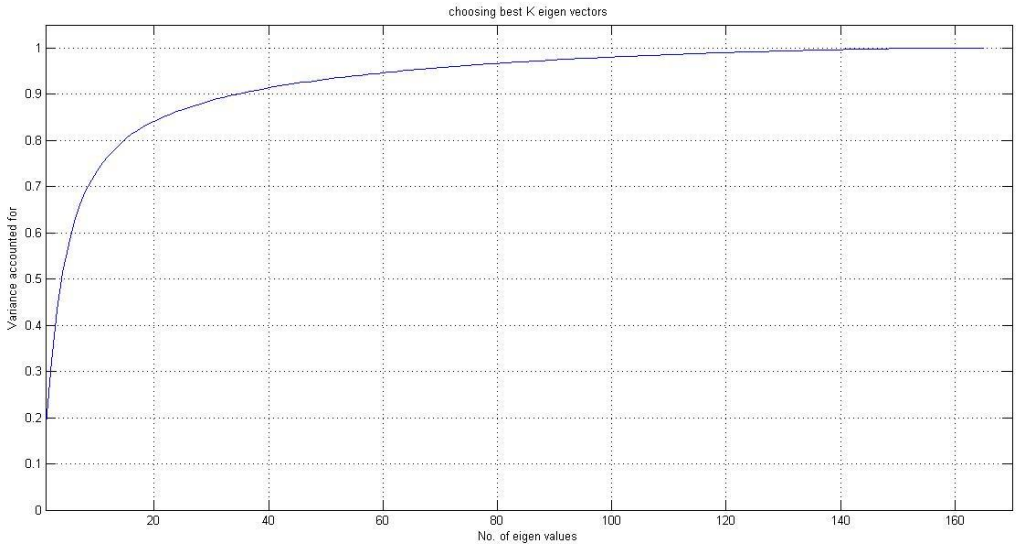
Comment : The reconstructed image looks blur but still it could be recognized.
Their corresponding normalized Eigen faces are as shown:



As We increase K the recognition gets better;
For K =20;



2. How did you select the number of components to include in your low-rank detector? How did you set the thresholds for your detection and classification?
Solution:
The selecting number of components to include in your low –rank detector is based on my eigen values and their variance when sorted in descending order
The plot below shows eigenvalues vs variance



As we increase 'K' the variance increases.
From variance around 0.65 the pictures can be detected i.e for K =8. For better accuracy of face detection of different poses, the variance should be high say around 90 %.
So from K =35 , the eigen face detection gets better.
Choosing 'K' best Eigen vectors done for reducing dimensionality as much as we can.

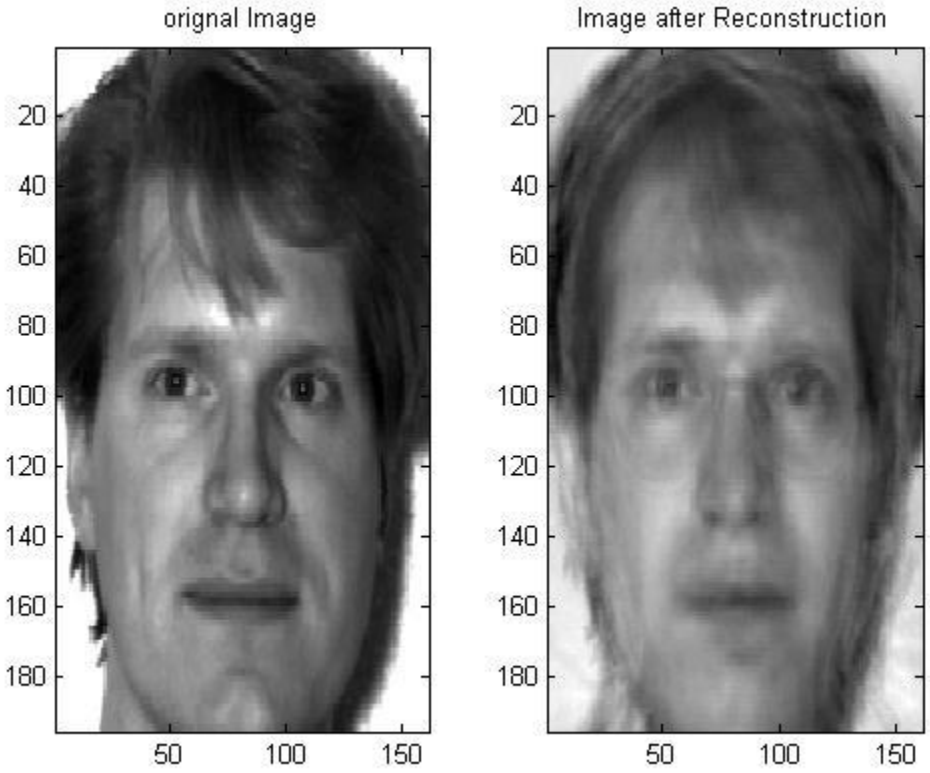
Procedure: for above plot
Sort the Eigen values in descending order and normalize them.
Then use cumsum command for plotting the graph.

Setting Threshold:
The threshold should be greater than minimum error.(Error computed by Euclidean Distance nothing but norm).
If error is greater than threshold the image does not belong to the database.
After storing the values of minimum error for each image at K =38, the maximum in the array of minimum values is said to be 1.5198e+04.
So threshold should be greater than 1.5198e+04.
We need to set threshold since without it, every unknown image would be detected as face image and recognize as some image in the database.
If the resulting reconstruction error is greater than the threshold, then the tested image probably is not a face image.
Procedure:
The values of minimum and maximum error are stored in a array for the images present in data set.
From the minimum value array (m1) the value slightly greater than 1.5198e+04 is choosen as a threshold value.
If > threshold value
Not a face
Else
It's a face which may be or may not be present in data base.

For setting 2nd threshold limit consider an image that has no defects and is outside the training set data . So that min distance can be consider as 2nd threshold limit.

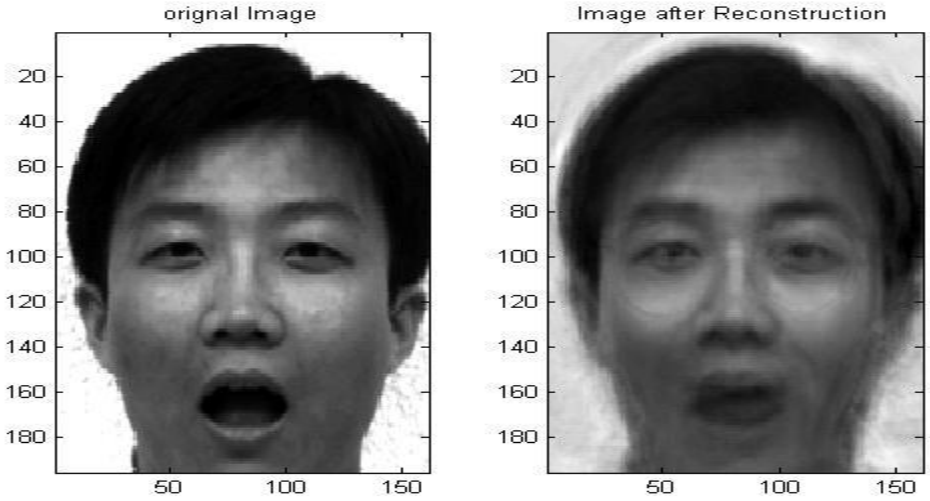
But in the projIsuppmentaldata file there is no such image which has no defect whose min error value >1.5198e+04.
2nd threshold value chosen as 15488.9510553266;

Part c)
How much does your ability to recognize individuals depend on the types of poses and or lighting in the training set.
Solution :
For K = 38 ,
Recognizing individuals on types of poses is poor.
We can identify the individuals but not there poses.
For pose : centerlight



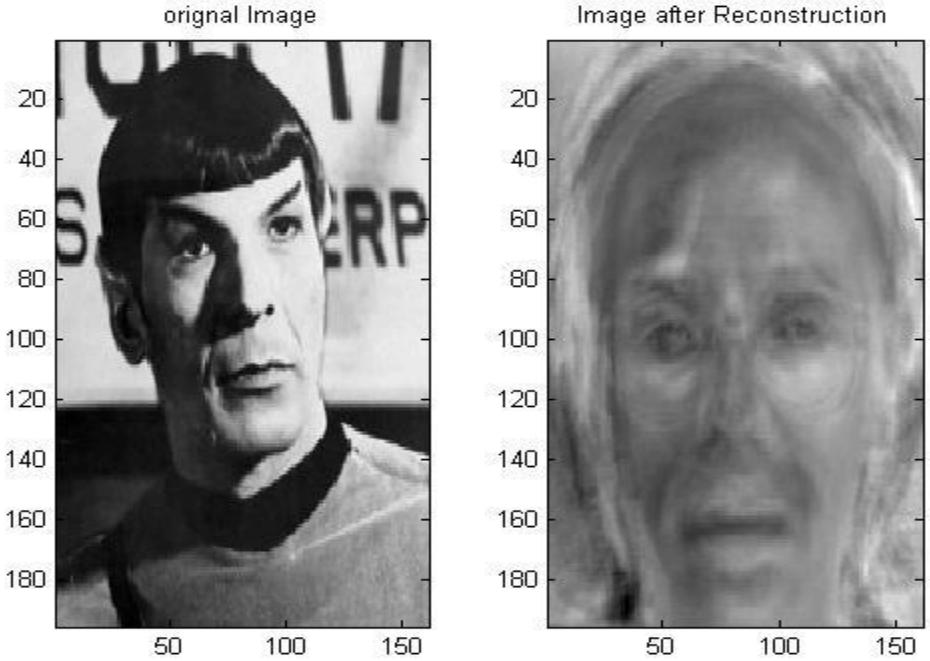
Which category of changes is easier to deal
with?
Identifying individuals is easier to deal with rather their poses.
If we want to deal with a pose surprised that might be okay detection to an extent.

For pose: surprised



Is it easier to identify individuals with different expressions than those in training set?
No , it is difficult.

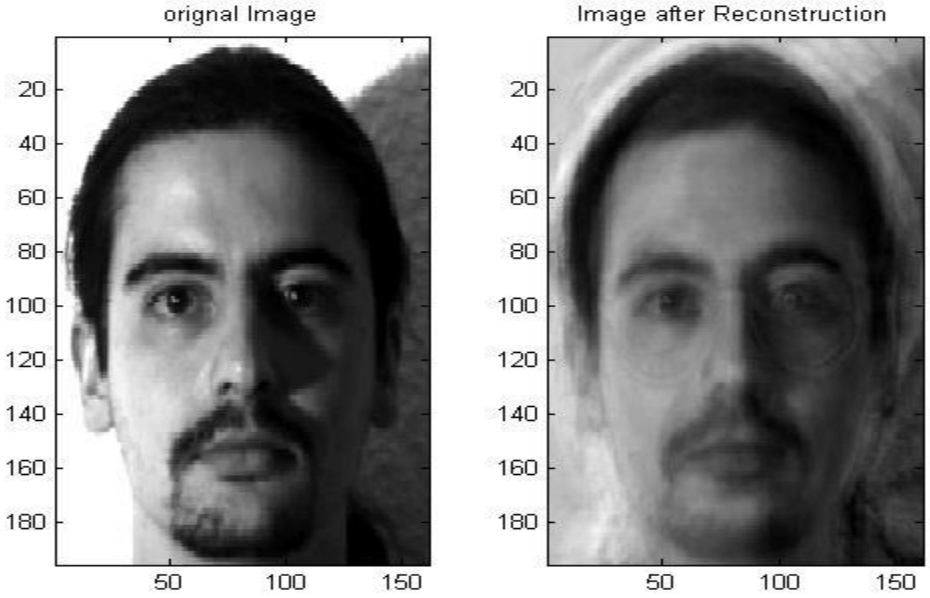
Consider 13th image in the proj1supplementaldata



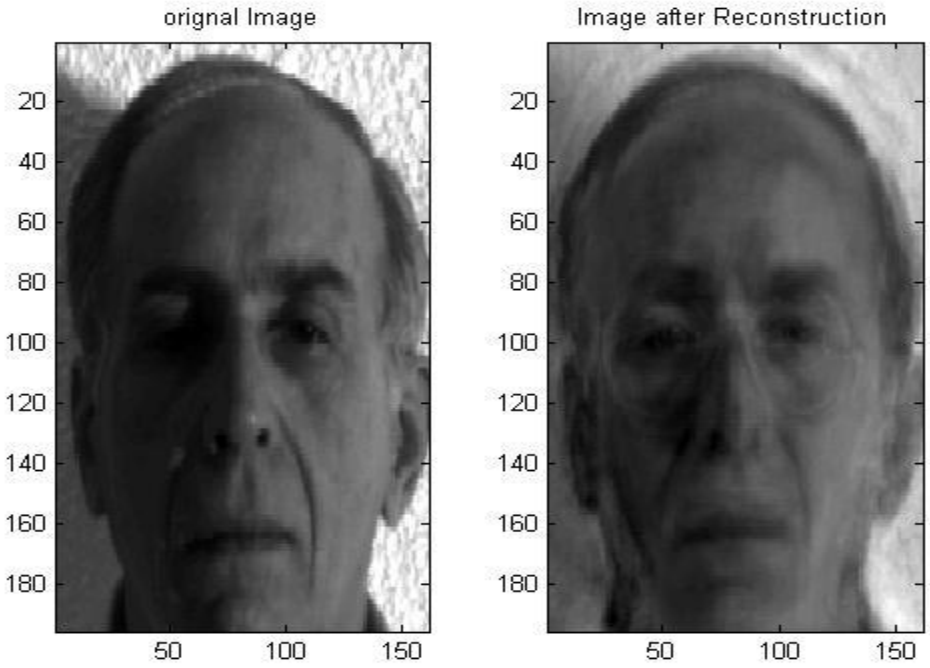
The individuals other than the training set are identified as ghosts (according to eigenface detection).
Identify individuals when the lighting has changed from training set ?

Yes , it is slightly easier to identify individuals when the lighting has changed from training set.
But the reconstructed images are more blur than compared to other expressions.

For pose : leftlight

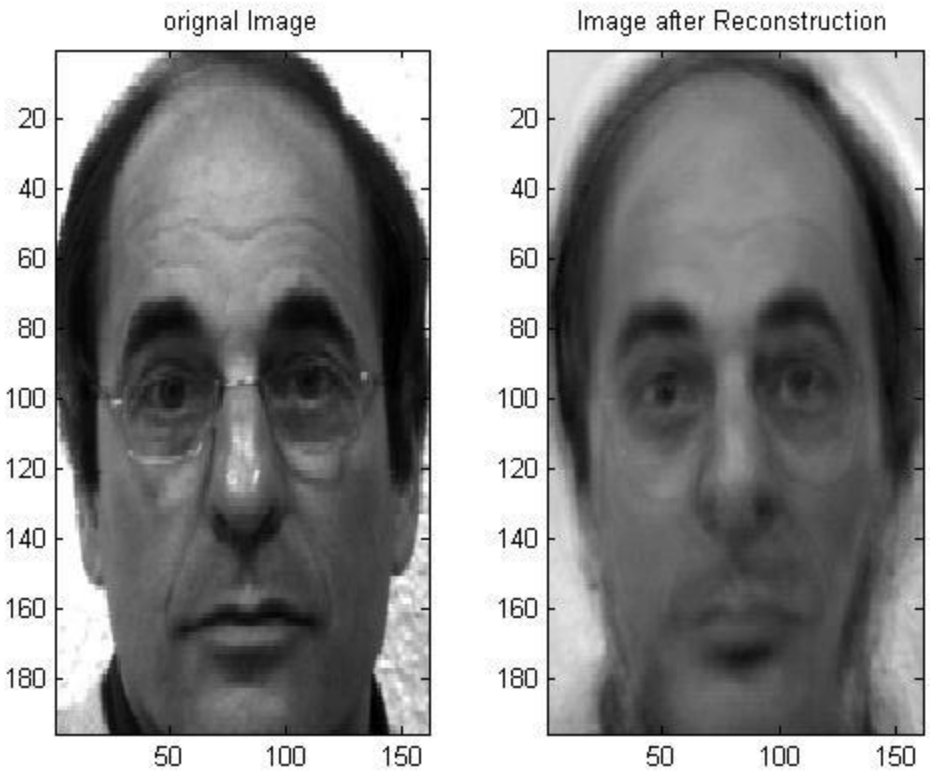


For pose: rightlight



Comment : It is easier to identify individuals rather than their expressions in the dataset.

Do your results change when leave out pictures with glasses ?
For pose : glasses



Comment : With glasses again the reconstruction image becomes more blur and it is slightly difficult to identify even the individual.

4. Try your software on the images in projIsupplemental.mat.
Can your code detect images contain the data set

Solution :

No it cannot detect the images contain in the data set except the 13th image which shows it is present in training data set .
Except one image i.e 13th recognition code shows whether they are present in the data base or not.

The 13th image is detected due to low Minimum error which is less than the threshold.
Code :

```
for i = 1:numIMG
[m1(i) ,m2(i) r(i) ] = eigenfacedetector(I(:,:,i), Me, D, u);
end

for j = 1: 13
   [m3(j) ,m4(j) r2(j) ] = eigenfacedetector(supplementaldata(:,:,j), Me, D, u);
end
```

```
threshold1 = 1.5199e+04;
threshold2 = 15488.9510553266;

%Reconginition Procedure
for i = 1:13


   if m3(i)<threshold1
   fprintf('the image is present in database');
   ig = i % to know which images are wrong detections
   Reconstruct(u,supplementaldata(:,:,i),K,Me )
   end
   if threshold2<m3(i)<threshold1
     imageno= i
      fprintf('the image is not  present in database ');
   end
   if m3(i)> threshold2
      imgno = i
       fprintf('the image is not  present in database and not a face image');
       end
end
```

Conclusion :

The eigen face detection is successfully implemented. The Eigen face detection is useful in identifying the individuals correctly but not their poses though it reduces the dimensionality.

APPENDIX

MATLAB CODE

2.1

```
function [ Data,data] = storedata(idface ,desiredposes,facedata)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
%Data for all images in training set
%data for set of images/image for a particular individual.

nt = length(facedata);

if nargin<2
   desiredposes=fieldnames(facedata);
end
nposes=length(desiredposes);
i = 1;
for faceid = 1:nt
   for poseid = 1:nposes
      X = facedata(faceid).(desiredposes{poseid});
      [nrow,ncol]=size(X);
   Data(:,:, i) = reshape(X,nrow,ncol);
   i = i+1;
   end
end
%to plot a particular face of an individual
data = Data(:,:,idface);

end
```

2.2

```
function [mV,V,u] = lowrankrep(K,I)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
%Input
% Select K best eigen vectors depending on variance
%
% I is a image matrix which is in 2-d vector after using store data
%ouput
%mV---> mean vector
%V---> Eigen vectors
%u1 --> set of eigen vectors defining face class

[m,n,numIMG] = size(I);

A = reshape(I,[m*n numIMG]);

% Removing Mean prior to computing eigen decomposition of the sample
% covariance  matri

[nrow,ncol] = size(A);
%mean vector mV
mV =mean(A,2)*ones(1,ncol);
D = A-mV;

% Sample covariance matrix
C = D'*D;
% calculating svd
[U,S,V] = svd(C,0);
% U - eigen vector of A'*A;
%S - eigen values
[S,ind] = sort(diag(abs(S)),'descend');   % sort eigenvalues in descending order
  V      = V (:,ind);         % arrange eigenvectors in same order

 for i=1:size(V,2) %access each column
kk=V(:,i);
temp=sqrt(sum(kk.^2));
V(:,i)=V(:,i)./temp;
 end

   u = D*V;
%    normalising
%    u = u/norm(u);

%ncol  - number of images
%D - difference of Image and mean

for i = 1 : ncol
```

```
    kk=u(:,i);
temp=sqrt(sum(kk.^2));
u(:,i)=u(:,i)./temp;
end
% U is considered for projection not V

End
```

2.3

```
function [m1,m2,Recogn_index] = eigenfacedetector(TestImage, mV, D, u)
%UNTITLED8 Summary of this function goes here
%   Detailed explanation goes here
% TestImage -> image which has to be tested
% Me --> Mean vector
%u ---> Eigen FAce vectors
% DataminusMean --> D

% Images are projected to face space by multiplying V and egn face vector

L = size(u,2);
PrjImg = [];
for i = 1 : L
   PrjImg(:,i) = u'*D(:,i);

end

temp = TestImage(:,:,1);

[irow icol] = size(temp);
InImage = reshape(temp,irow*icol,1);
Diff = InImage-mV; % Centered test image
ProjectedTestImage = u'*Diff; % Test image feature vector

% Calculating Euclidean distances


for i = 1 : L
   q = PrjImg(:,i);
    Euc_dist(:,i) =norm( ProjectedTestImage - q ) ;
end

[m1, Recogn_index] = min(Euc_dist);
```

```
m2 = max(Euc_dist);


%
% OutputName = strcat(int2str(Recogn_index),'.jpg');


end



% Reconstruction of images

function [ output_args ] = Reconstruct(u,L1,K,Me )
%UNTITLED9 Summary of this function goes here
%   Detailed explanation goes here
% u -->Eigen face vector
% L1 --> Input Image
% K ---> Choosing K best Eigen Values
%
% Me --> Mean vector
%



[irow,icol,b] = size(L1);

B = reshape(L1,irow*icol,b);
figure ;
subplot(1,2,1)
imagesc(L1(:,:,1));  colormap('gray');
%[mV2,V2,u2 ] = lowrankrep(10,L1);
  %m is the mean image, u is the eigenvector
  title('orignal Image')
  projection = u(:,1:K)'*(B-Me);
ReshapedImage = u(:,1:K)*projection+Me;

%ReshapedImage = u*V(:,1:K);

ReshapedImage = reshape(ReshapedImage,irow,icol);

%show the reconstructed image.
subplot(1,2,2)
imagesc(ReshapedImage(:,:,1)); colormap('gray');

title('Image after Reconstruction');
```

```
end
```

```
%  Matlab script where I plotted graphs.


clc;
% load('projldata.m');
desiredposes = fieldnames(facedata);

[I,L] = storedata((1:15),desiredposes,facedata);
% I is matrix is of the form 195 x 161 x 165

[m,n,numIMG] = size(I);

A = reshape(I,[m*n numIMG]);

% Removing Mean prior to computing eigen decomposition of the sample
% covariance  matri

[nrow,ncol] = size(A);

D = A-(mean(A,2)*ones(1,ncol));
Me = mean(A,2);
% Finding mean image
Mimg =  reshape(Me,m,n);

% Sample covariance matrix
C = D'*D;
% calculating svd
[U,S,V] = svd(C,0);
% U - eigen vector of A'*A;
%S - eigen values


[S,ind] = sort(diag(abs(S)),'descend');   % sort eigenvalues in descending order
   Vs     = V (:,ind);         % arrange eigenvectors in same order

   %normalising eigen vectors
for i=1:size(V,2) %access each column
kk=Vs(:,i);
temp=sqrt(sum(kk.^2));
Vs(:,i)=Vs(:,i)./temp;
```

```
end


  u = D*V;

  % normalising

% calculation of wieght vectors


for i = 1 :size(u,2)
   kk=u(:,i);
temp=sqrt(sum(kk.^2));
u(:,i)=u(:,i)./temp;

end




% selecting best K Eigen vectors
% K = 40;
% Dimensionality reduction.
% fprintf('Creating lower dimensional subspace\n')
% u1 = u(:, 1:K);
% D1 = D(:, 1:K);

% Caculating Eigen Vectors

%finding min error through eculdian distance


% display the eigenvalues for C
for j = length(S);
   S1(j) = S(j)^2;
end

normalised_evalues = S / sum(S);
figure(1)
 plot(cumsum(normalised_evalues));
title('choosing best K eigen vectors')
xlabel('No. of eigen values'), ylabel('Variance accounted for');
xlim([1 170]), ylim([0 1.05]), grid on;


%
%plot eigen vectors
num_eigenfaces = 15;
```

```
%display the eigenvectors
figure;
figure(2);
K = 38;
u = u(:,1:K);
D1 = D(:, 1:K);
Vs1 = Vs(:,1:K);
%Caculating Wieght vector

w = u'*D1;

for i=1:size(u,2)
img=reshape(u(:,i),m,n);
img=histeq(img,255);
 subplot(2, ceil(K/2), i);
imshow(img)
drawnow;

if i==3
title('Eigenfaces','fontsize',18)
end
end

%for a single image taken or tested

%Reconstruction for a test image
[I1,L1] = storedata(15,{'normal'},facedata);

[irow,icol,b] = size(L1);

B = reshape(L1,irow*icol,b);
figure (3)
subplot(1,2,1)
imagesc(L1(:,:,1));   colormap('gray');
[mV2,V2,u2 ] = lowrankrep(10,L1);
title('orignal Image')
 % m is the mean image, u is the eigenvector

  projection = u(:,1:K)'*(B-Me);
ReshapedImage = u(:,1:K)*projection+Me;


ReshapedImage = reshape(ReshapedImage,irow,icol);
subplot(1,2,2)
%show the reconstructed image.
imagesc(ReshapedImage(:,:,1)); colormap('gray');

title('Image after Reconstruction');
```

```
%Storing all the distances of Training data set


for i = 1:numIMG
[m1(i) ,m2(i) r(i) ] = eigenfacedetector(I(:,:,i), Me, D, u);
End




%Recognition matlab code



clc;

for i = 1 :13
   imagesc(supplementaldata(:,:,i));
truesize;
      colormap(gray);


      pause(0.2)                % short pause between plots
end



for i = 1:numIMG
[m1(i) ,m2(i) r(i) ] = eigenfacedetector(I(:,:,i), Me, D, u);
end

for j = 1: 13
   [m3(j) ,m4(j) r2(j) ] = eigenfacedetector(supplementaldata(:,:,j), Me, D, u);
end


threshold1 = 1.5199e+04;
threshold2 = 15488.9510553266;

%Reconginition Procedure
for i = 1:13
```

```
   if m3(i)<threshold1
   fprintf('the image is present in database');
   ig = i % to know which images are wrong detections
   Reconstruct(u,supplementaldata(:,:,i),K,Me )
   end
   if threshold2<m3(i)<threshold1
     imageno= i
      fprintf('the image is not  present in database ');
   end
   if m3(i)> threshold2
      imgno = i
       fprintf('the image is not  present in database and not a face image');
       end
end
```