Arda Cankat Bati

A53284500

# EIGENFACE PROJECT

## INTRODUCTION

Before answering the questions, I would like to talk about the structure of my report. To make it less cumbersome for you (and me) I decided to divide it into 4 parts. My code is divided into the same 4 parts too. Please note that I created my own folder hierarchy of the images (all the datasets remain the same, I just created a folder for each one. The code has more detail about this). The parts and their explanations are below:

- **Part 1: Project_Data_Processing.** This part focuses on loading the data, creating the eigenfaces and doing initial analysis. It also finds the eigen values of the matrix L (described in the code), and the singular values of the normalized train dataset, answering part a. It prints the top eigenfaces, finds a good threshold of how many eigenfaces are enough to do good reconstructions. It also loads and shows the non-face image, which is the image of a car. At the end of the code all related plots are provided with appropriate labeling. The pages of this part are labeled Project_Data_Processing on the top.
- **Part 2: Project_B_E:** As the name implies, this part contains the solutions for questions between B and E. I use all the principle components, starting from 1 and up to 190. At each stage I calculate the current MSE. Also, at certain stages I print the face reconstruction with the current set of eigenfaces. All the relevant plots are supplied at the end of the code of **Project_B_E.** The pages of this part are labeled Project_B_E on the top.
- **Part 3: Project_F:** As the name implies, this part answers the question F of the project. It is working with rotated versions of an image from the Train set. The rotations are counter-clockwise and represented in degrees. They are [0, 45, 90, 135, 180, 225, 270, 315] specifically. For each rotation, I do its reconstruction using all the principal components. I report the MSE of the reconstruction for each rotation, and plot them. All the resulting figures and MSE print output is given after the code. The pages of this part are labeled Project_F on the top.
- **Part 4: reconstruct_face:** This part is only composed of the helper function I implemented called **reconstruct_face.** Its task is to create the reconstruction of a given image, with the supplied parameters. Please check the code for further detail, I did as much commenting as I can. I just want to stress out that I implemented this function on my own (as with the rest of the code). This part has no output. The pages of this part are labeled reconstruct_face on the top.

While answering the questions, I will refer to the **figures in under the code**, but will not include them here directly. I am doing this to make the report more readable, I hope it will not cause any problems to the grader. I labeled each plot as much as I can to make it clear. Just to make it clear, each part has its own corresponding figures under the code of that part.

## ANSWERS

**Part A)**

Looking at the eigenvalues of matrix L, and the singular values of the Normalized dataset, I see an obvious pattern. Firstly, after getting the singular and eigen values, I sorted them from maximum to minimum. The pattern I see is that the values are dropping very fast. For example, while the initial singular values have very high values, the later ones quickly drop down. These values show us how significant each principal component will be, in describing the variance inthe data. According to the eigen values, I decided on a threshold for how many eigenfaces (principal components) are enough to represent 90% variance of the data. I found this value to be 62.

Each eigenvalue in PCA, compared to the total sum of the eigen values, shows how much variance the corresponding principal component will give. I have found that the first 62 principal components (after sorting by eigenvalues) correspond to 90% of the total sum of the eigenvectors. In the "Eigenvalues of Matrix L graph", I also showed this cutoff. In the Singular Values of Neutral Normalized Train Set graph, I showed the sorted singular value breakdown of the normalized train dataset of 190 images. I did some example reconstructions with the 62 eigenfaces, and observed that they are pretty close to the original images (subjectively).

**PART B)**

I had a loop in this part, starting from the best eigenface, and keeping on adding more eigenfaces to it, until all 190 eigenfaces are added. I recorded the MSE at each stage. After 25 * i eigenfaces were added, I plotted the current reconstruction image and went on. The reconstructions are above the Neutral Train Image Reconstruction graph, and the graph itself shows how MSE is changing. As expected, it drops as more and more eigenfaces (principal components) are added. This is because each eigenface is orthogonal, and they explain more variance of the Neutral Train dataset. Finally, after all the eigenfaces are added, we see that the resulting MSE is 0 (around 10 ^ -7). This makes sense because, each eigenface has (represents) some part of our image from the train dataset. When we combine all of them, we should be able to get the original image back. However, this won't hold for the next parts, because our images in the next parts are not from the Train Set.

**PART C)**

I had a similar loop for this part as in Part B. However, this time we are using a datapoint not from the Neutral Train Set. The caveat is that the person in the image is actually in the train set, but not smiling. Therefore, in the reconstruction we cannot reach 0 MSE this time, as shown in the graph: Smiling Train Image Reconstruction. As we add more and more eigenfaces, the MSE asymptotes above a certain value. If we look at the reconstructions (above the mentioned MSE graph) we can see that the part of the face above the nose is pretty good. However, as expected, the mouth is very inaccurate, as our principal components do not have the expressive power to represent a smile. (Because they were trained on a Neutral Train Set, which doesn't include smiles). Therefore, the overall result is significantly worse than part B.

**PART D)**

Again, a similar loop to bar B was used. This time the reconstructed image is neutral, but it is from the test dataset. The MSE results are plotted in the graph Neutral Test Image Reconstruction. Some example reconstructions are given above this graph. We can see from the MSE results that the overall

MSE is somewhat better than Part C. Even though the test image is a person's image that wan not in the trainset, the neutral expression is reconstructed by the principal components. An interesting result is that the MSE graph is not monotonically decreasing as before. Looking at the reconstructions, although the noise is high, we as human observers would find them much more accurate than the results of Part C. Although the difference in MSE is slightly better, I think the subjective quality of the reconstruction is much higher than part C.

**PART E)**

The loop is similar to Part B. As expected in this part the reconstruction fails spectacularly (and we have quite funny results). Our principal components are primed very specifically, on face images of a certain orientation and size, also facial expression. The image of the car is very foreign to our eigenfaces subspace. When we try to project it there, we get very inaccurate results as expected. The MSE results can be seen in the Non-Face Image Reconstruction plot, above this plot the reconstructions are plotted. We see the reconstruction is trying to at least imitate the mean image intensity of some parts of the original car image. (Like the shadow under the car). This is why the MSE plot drops until a certain MSE threshold, although it is very different from the original car image (from a human perspective). As the subspaces of car and faces are so different, there is no way we will have an accurate reconstruction in this case.

**PART F)**

In this part, the rotation by 0 degrees is just the original Train Neutral image, which is the same as in part b. When we start rotating the image, the actual information in the image mostly remains the same (apart from cropping and resizing). However, as we aren't rotating our eigenfaces as well, they just work assuming the original orientation of the faces from the dataset. This shows how important the orientation of the image is, in such a task as we do. After each rotation, we also are somehow rotating the subspace that the image lives in. However, our eigenfaces subpace is not rotated in anyway. Therefore, the eigenfaces are trying to reconstruct the image, but them and the rotated image live in much different subspaces. Looking at the MSEs, (printed right after the code) we see that they are mostly the same (and very high). The best one is 180 degrees of rotation. However, this is probably due to the 180 rotation not having any black crop marks around it, compared to all the other rotations.

If we rotated our principal components accordingly, we would still get good results as before. Because in that case both the original image and the principal components would be in the same subspaces.

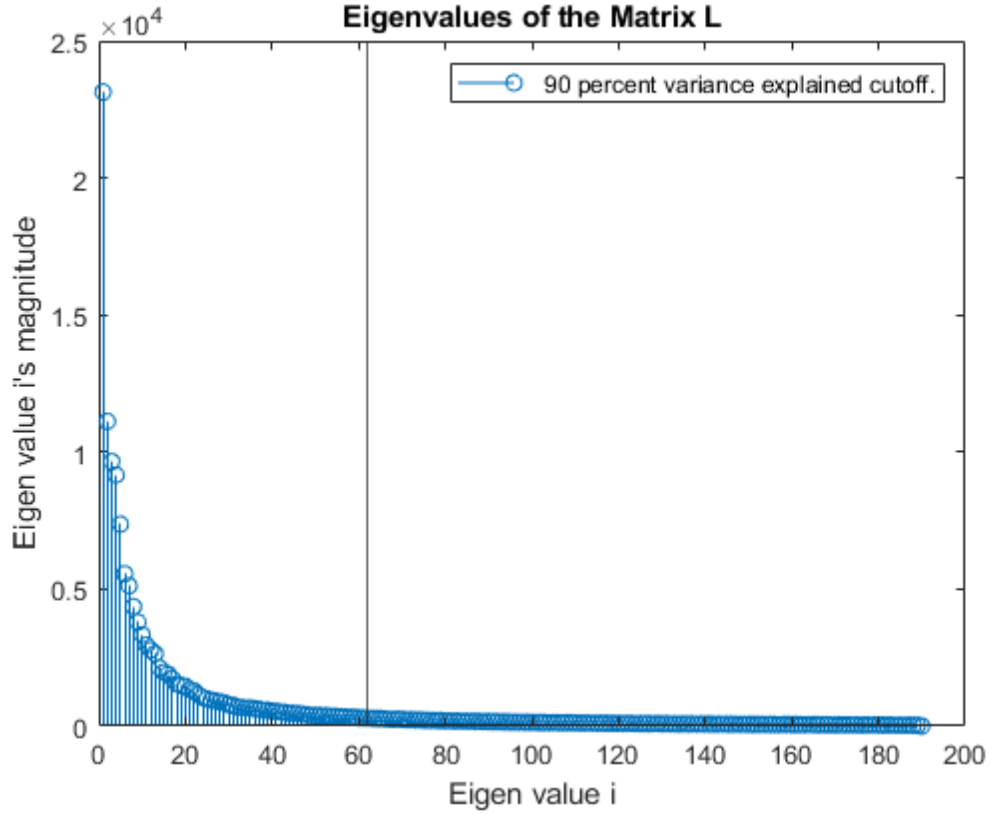
```
% ****************************************
% * (A) DATA PROCESSING & EIGEN FACE CALC. **
% ****************************************

% This script is used to load and process the data. The images are read
% from the corresponding folder, and grouped into train and test sets. The
% data is converted to double type to make the calculations more precise.
% Neutral and smiling faces have separate sets (see below). The mean of
% the datasets are calculated, and stored in dataset_mean. Finally
% eigenfaces are calculated according to the Neutral Training Set.
% This script also includes the solution to part A.

clc
clearvars -except S

% Variables below are created to find and store 4 datasets as follows:
% 1- Train dataset for neutral faces. FIrst 190 datapoints from neutral.
% 2- Test  dataset for neutral faces. Last   10 datapoints from neutral.
% 3- Train dataset for smiling faces. First 190 datapoints from smiling.
% 4- Test  dataset for smiling faces. Last   10 datapoints from smiling.

% *********** IMPORTANT NOTE ******************
% Notice I created my own folder structure to make reading data easier.
Train_Neutral  = 'trainset\neutral\';
Train_Smiling  = 'trainset\smiling\';
Test_Neutral   = 'testset\neutral\';
Test_Smiling   = 'testset\smiling\';
dataset_names = {Train_Neutral, Train_Smiling, Test_Neutral,...
          Test_Smiling};
% *********** IMPORTANT NOTE ******************

% Loop below goes over each dataset and reads the images (in array form)
for i = 1:4
    current = dataset_names{i};
    jpgfiles = dir(fullfile(current,'\*.jpg*'));
    number_of_jpgs = numel(jpgfiles);
    current_dataset = [];
    for j = 1:number_of_jpgs
        im = jpgfiles(j).name;
        im1 = imread(fullfile(current, im));
        im1 = im2double(im1);
        [irow, icol] = size(im1);
%         temp = reshape(im1, irow * icol, 1);
        current_dataset(:,j) = im1(:);
%         figure()
%         imshow(im1);
    end
    datasets{i} = current_dataset;
end

% Dataset names are assigned here. Variable names explain themselves.
Train_Neutral = datasets{1};
Train_Smiling = datasets{2};
Test_Neutral  = datasets{3};
Test_Smiling  = datasets{4};

tra_neu_mean = mean(Train_Neutral, 2);
tra_smi_mean = mean(Train_Smiling, 2);
tes_neu_mean = mean(Test_Smiling,  2);
```

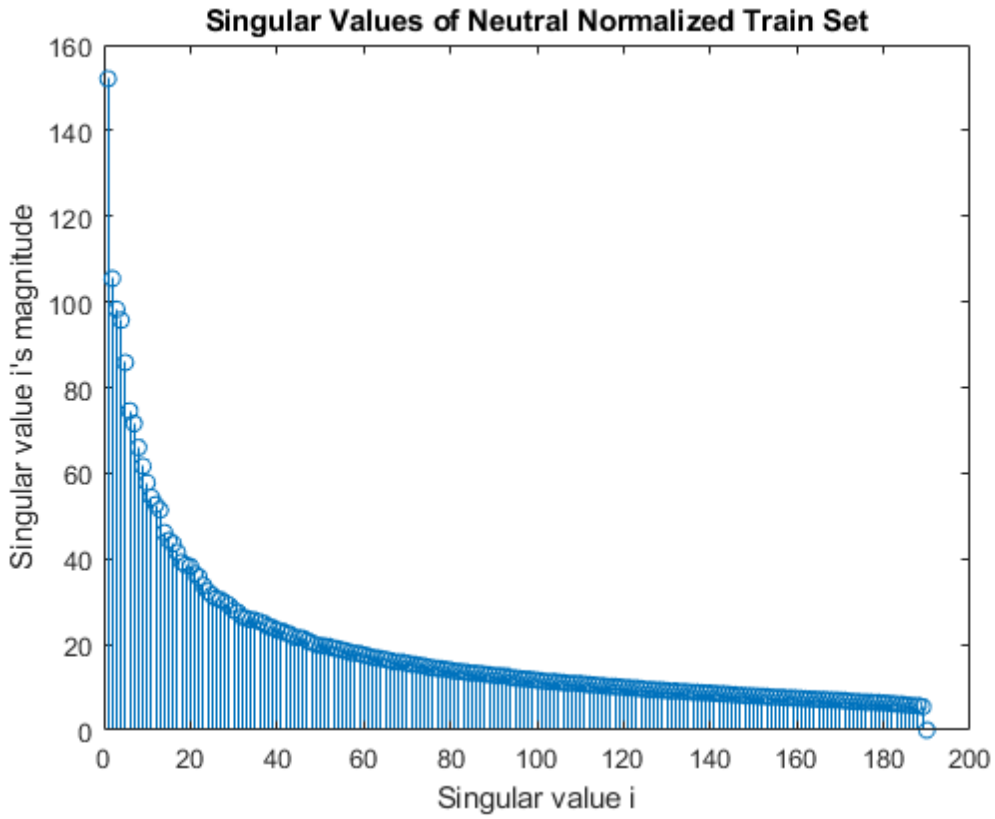
```
tes_smi_mean = mean(Test_Smiling,  2);

feat_len  = size(Train_Neutral, 1);
train_len = size(Train_Neutral, 2);
test_len  = size(Test_Neutral,  2);

% Normalizing the train sets by subtracting mean, storing it
Tra_Neu_Norm = Train_Neutral - repmat(tra_neu_mean, 1, train_len);
Tra_Smi_Norm = Train_Smiling - repmat(tra_smi_mean, 1, train_len);
Tes_Neu_Norm = Test_Neutral  - repmat(tes_neu_mean, 1, test_len);
Tes_Smi_Norm = Test_Smiling  - repmat(tes_smi_mean, 1, test_len);


% Matrix L constructed as outlined in the paper:
% "Eifenfaces for recognition" by Turk and Pentland
% Link: http://www.face-rec.org/algorithms/pca/jcn.pdf
% Getting the eigen values and eigenvectors of the matrix
L = Tra_Neu_Norm' * Tra_Neu_Norm;
[V, D] = eig(L);
D_vector = diag(D);
% Sorting the eigenvalues to get the best eigenvectors
% Best here means the ones that explain maximum variance. (page 5 of  the
% above paper, mentioned just before formula (6)
[values, indices] = sort(D_vector,'descend');
% Below we calculate the cumulative percent variance explained as we keep
% on adding more eigenvalues.
variance_explained = cumsum(values) / sum(values);
variance_explained = variance_explained >= 0.9;
% The cutoff below shows how many eigenfaces we need to get 90% variance
% explained. (In a traditional PCA sense, however here we won't be able to
% explain the full variance because our PCA method is not perfect when we
% use the matrix L).
cutoff = find(variance_explained, 1, 'first');
fprintf('%i eigen faces needed to explain 90 percent variance.\n', cutoff);

% Singular Value Decomposition of the original (not shifted) data matrix
% to plot the Singular Values as asked.
% I usually comment the below line as it takes a long time. I just keep the
% S variable in the workspace.
[~, S, ~] = svd(Tra_Neu_Norm);
S_diag = diag(S);

% Plotting Eigenvalues of matrix L
figure()
stem(1:train_len, values, 'o'); hold on;
xline(cutoff);
legend("90 percent variance explained cutoff.")
title('Eigenvalues of the Matrix L')
ylabel("Eigen value i's magnitude"); xlabel("Eigen value i"); hold off;

% Plotting singular values of the original data matrix (not shifted)
figure()
stem(1:train_len, S_diag, 'o');
title('Singular Values of Neutral Normalized Train Set')
ylabel("Singular value i's magnitude"); xlabel("Singular value i");

% Eigenfaces are calculated from the matrix product of the normalized
% dataset with each eigenvector. (formula (6) from the above paper)
eigenfaces_count = cutoff;
Eigenfaces_PCA = zeros(feat_len, cutoff);
```

```
for count = 1 : eigenfaces_count
    Eigenfaces_PCA(:,count) = Tra_Neu_Norm * V(:, indices(count));
    eigen_face_norm = norm(Eigenfaces_PCA(:,count));
    Eigenfaces_PCA(:,count) = Eigenfaces_PCA(:,count) / eigen_face_norm;
end

% ****************************************
% *********** PLOTTING EIGENFACES ***********
% ****************************************

% Plotting the first 9 Eigenfaces for reference, also as a sanity check.
% From the printed results, they seem as expected.
figure()
image_count = 9;
for count = 1:image_count
    eigenface = Eigenfaces_PCA(:, count);
    % Normalazing the eigenface to print it as grayscale image
    min_val = min(eigenface); max_val = max(eigenface);
    eigenface =((eigenface - min_val).*1)./(max_val - min_val);
    % Reshaping into original image dimensions
    eigenface = reshape(eigenface, irow, icol);
    subplot(3, 3, count);
    imshow(eigenface)
    title(sprintf('Eigenface %d', count));
end
% sgtitle('Q5 a) The 9 highest variance eigenfaces derived by PCA.');

% ****************************************
% ********** PLOTTING RECONSTRUCTIONS *******
% ****************************************

% Plotting some of the reconstructions with all the 190 eigenfaces, as
% reference and also for sanity check
figure()
for face = 1:4
    face_id = face + 6;
    % reconstruct_face is my own function defined in its corresponding .m
    % file. It reconstructs a face from the dataset given the valid
    % parameters as below.
    reco = reconstruct_face(Eigenfaces_PCA, eigenfaces_count,...
                                tra_neu_mean, Tra_Neu_Norm(:, face_id));
    reco = reshape(reco, irow, icol);
    subplot(4, 2, face * 2 - 1);
    % Original Image
    im = reshape(Train_Neutral(:, face_id), irow, icol);
    imshow((im));
    title('Original Image');
    % Reconstruction image
    subplot(4, 2, face * 2);
    imshow(reco);
    title(sprintf('Reconstruction, using %i eigen faces', cutoff));
end
% sgtitle('Q5 a) The 9 highest variance eigenfaces derived by PCA.');

% Loading a car image for future questions. Resizing the image to the same
% dimensions as the face images.
RGB = imread('car.jpg');
I = rgb2gray(RGB);
scaling = irow / size(I, 1);
J = imresize(I, scaling);
cropping = (size(J, 2) - icol) / 2;
```

```
car_image = J(:, cropping : size(J, 2) - cropping - 1);
car_image = im2double(car_image);
figure()
imshow(car_image)
title('Non face image');
car_image = reshape(car_image, irow * icol, 1);

% Resetting the cutoff value, I want to use all 190 eigenfaces for the next
% parts of the project. I am doing this manily to see if I will reach 0 MSE
% while working with train data.
cutoff = 190;
% Eigenfaces are calculated from the matrix product of the normalized
% dataset with each eigenvector. (formula (6) from the above paper)
eigenfaces_count = cutoff;
Eigenfaces_PCA = zeros(feat_len, cutoff);

for count = 1 : eigenfaces_count
    Eigenfaces_PCA(:,count) = Tra_Neu_Norm * V(:, indices(count));
    eigen_face_norm = norm(Eigenfaces_PCA(:,count));
    Eigenfaces_PCA(:,count) = Eigenfaces_PCA(:,count) / eigen_face_norm;
end


% Clearing unnecessary data -except option to keep relevant data
clearvars -except Train_Neutral Train_Smiling Test_Neutral Test_Smiling ...
          irow icol tra_neu_mean tra_smi_mean feat_len train_len ...
          Eigenfaces_PCA Tra_Neu_Norm Tra_Smi_Norm Tes_Neu_Norm ...
          Tes_Smi_Norm tes_neu_mean tes_smi_mean test_len car_image ...
          S cutoff

% Saving relevant data for future questions
save Project_Data
```

```
62 eigen faces needed to explain 90 percent variance.
```


Eigenvalues of the Matrix L
×10^4
90 percent variance explained cutoff.
Eigen value i's magnitude
Eigen value i
0
0.5
1
1.5
2
2.5
0
20
40
60
80
100
120
140
160
180
200


Singular Values of Neutral Normalized Train Set
Singular value i's magnitude
Singular value i
0
20
40
60
80
100
120
140
160
0
20
40
60
80
100
120
140
160
180
200

Eigenface 1

Eigenface 2

Eigenface 3

Eigenface 4

Eigenface 5

Eigenface 6

Eigenface 7

Eigenface 8

Eigenface 9

Original Image

Reconstruction, using 62 eigen faces

Original Image

Reconstruction, using 62 eigen faces

Original Image

Reconstruction, using 62 eigen faces

Original Image

Reconstruction, using 62 eigen faces

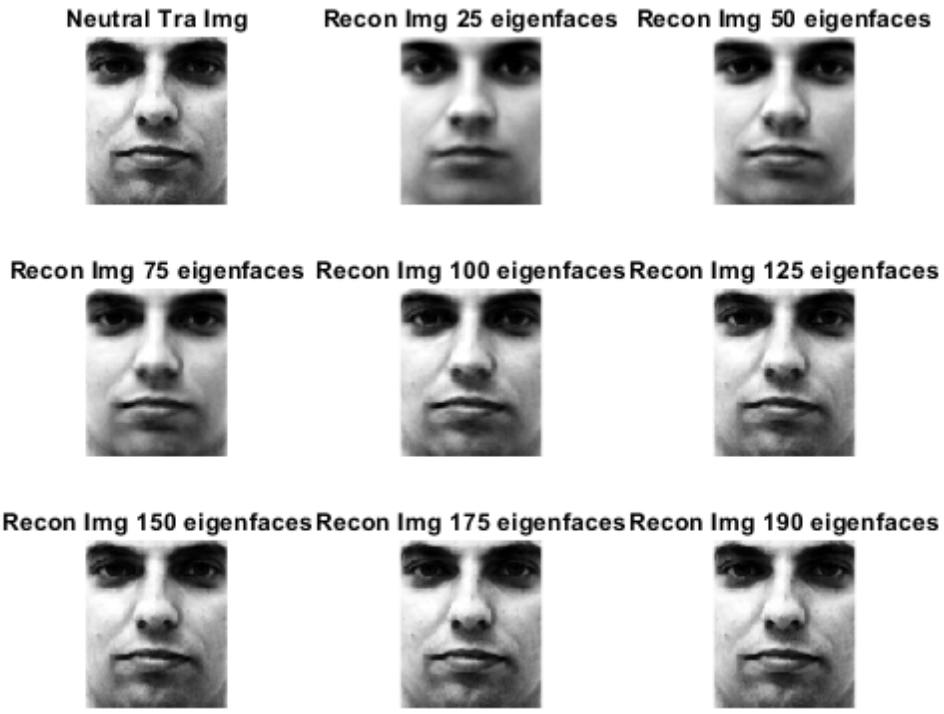
Non face image

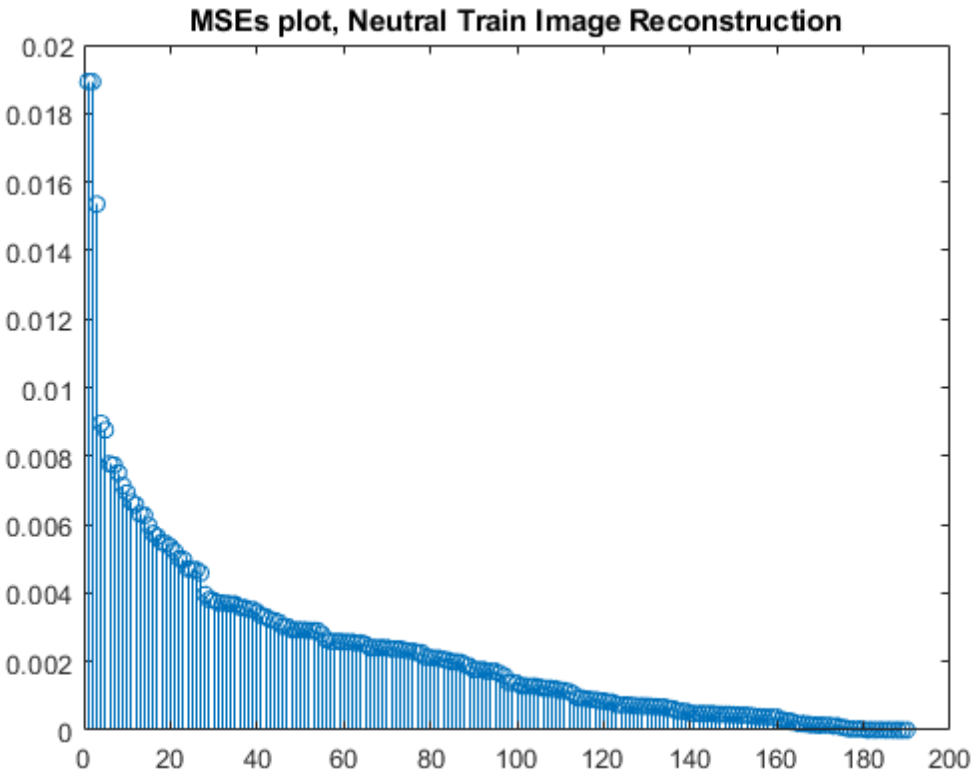
*Published with MATLAB® R2019b*

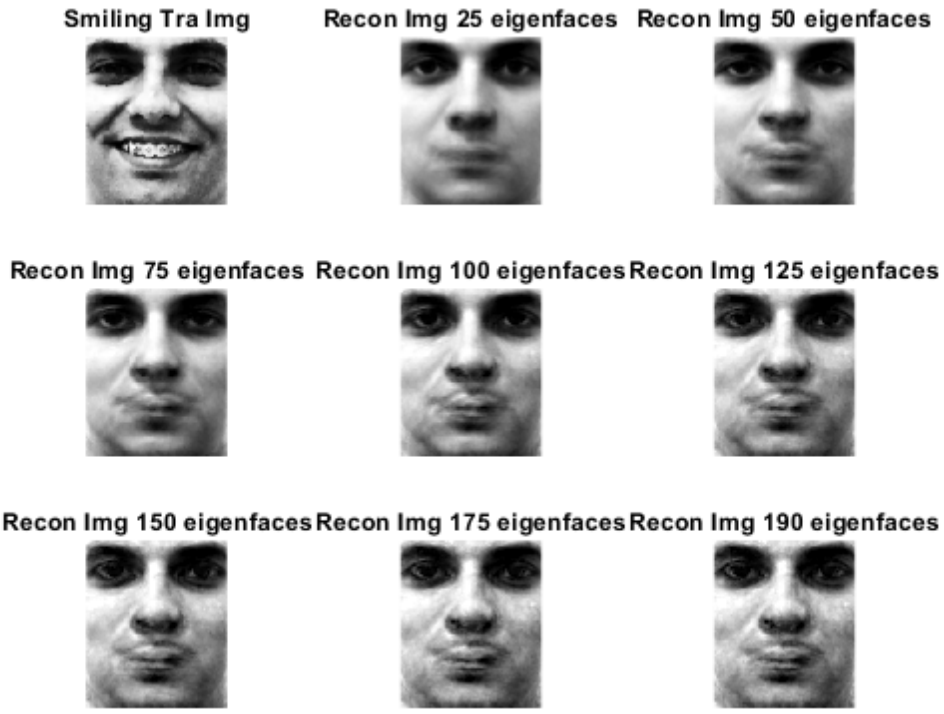
```
% RUN ME!!
% Project_Data_Processing
clc; clear all;
load Project_Data;

% ********************************************************************
% ************** (B) MSE CALC. FOR TRAIN NEUTRAL FACE ***************
% ********************************************************************

face_id = 18;
eigenfaces_count = cutoff;
MSEs = zeros(cutoff, 1);

figure();
plot_count = 1;
subplot(3,3,plot_count);
im = Train_Neutral(:, face_id);
im = reshape(im, irow, icol);
imshow(im);
title(sprintf('Neutral Tra Img'));
MSEs = zeros(cutoff, 1);
for count = 1:eigenfaces_count
    reco = reconstruct_face(Eigenfaces_PCA, count, tra_neu_mean, ...
                            Tra_Neu_Norm(:, face_id));
    cur_image = Train_Neutral(:, face_id);
    MSEs(count) = sum((reco - cur_image).^2, 1) / size(reco, 1);

    if 0 == rem(count, 25) || count == 190
        plot_count = plot_count + 1;
        subplot(3,3,plot_count);
        % Reconstruction image
        reco = reshape(reco, irow, icol);
        imshow(reco);
       title(sprintf('Recon Img %i eigenfaces', count));
    end
end

figure();
stem([1:eigenfaces_count], MSEs, 'o');
title('MSEs plot, Neutral Train Image Reconstruction');

% ********************************************************************
% ************** (C) MSE CALC. FOR TRAIN SMILING FACE ***************
% ********************************************************************

figure();
plot_count = 1;
subplot(3,3,plot_count);
im = Train_Smiling(:, face_id);
im = reshape(im, irow, icol);
imshow(im);
title(sprintf('Smiling Tra Img'));
MSEs = zeros(cutoff, 1);
for count = 1:eigenfaces_count
    reco = reconstruct_face(Eigenfaces_PCA, count, tra_neu_mean, ...
                            Tra_Smi_Norm(:, face_id));
    cur_image = Train_Smiling(:, face_id);
    MSEs(count) = sum((reco - cur_image).^2, 1) / size(reco, 1);
```

```
    if 0 == rem(count, 25) || count == 190
        plot_count = plot_count + 1;
        subplot(3,3,plot_count);
        % Reconstruction image
        reco = reshape(reco, irow, icol);
        imshow(reco);
       title(sprintf('Recon Img %i eigenfaces', count));
    end
end

figure();
stem([1:eigenfaces_count], MSEs, 'o');
yl = ylim; ylim([0, yl(2)]);
title('MSEs plot, Smiling Train Image Reconstruction');
xlabel('number of eigen faces used in reconstruction')
ylabel('MSE')

% ********************************************************************
% ************** (D) MSE CALC. FOR TEST NETURAL FACE ***************
% ********************************************************************

figure();
face_id = 8;
plot_count = 1;
subplot(3,3,plot_count);
im = Test_Neutral(:, face_id);
im = reshape(im, irow, icol);
imshow(im);
title(sprintf('Neutrak Test Img'));
MSEs = zeros(cutoff, 1);
for count = 1:eigenfaces_count
    reco = reconstruct_face(Eigenfaces_PCA, count, tra_neu_mean, ...
                            Tes_Neu_Norm(:, face_id));
    cur_image = Test_Neutral(:, face_id);
    MSEs(count) = sum((reco - cur_image).^2, 1) / size(reco, 1);

    if 0 == rem(count, 25) || count == 190
        plot_count = plot_count + 1;
        subplot(3,3,plot_count);
        % Reconstruction image
        reco = reshape(reco, irow, icol);
        imshow(reco);
       title(sprintf('Recon Img %i eigenfaces', count));
    end
end

figure();
stem([1:eigenfaces_count], MSEs, 'o');
yl = ylim; ylim([0, yl(2)]);
title('MSEs plot, Neutral Test Image Reconstruction');
xlabel('number of eigen faces used in reconstruction')
ylabel('MSE')

% ********************************************************************
% ************** (E) RECONSTRUCTION FOR NON FACE IMAGE ***************
% ********************************************************************

figure();
plot_count = 1;
subplot(3,3,plot_count);
im = car_image;
```

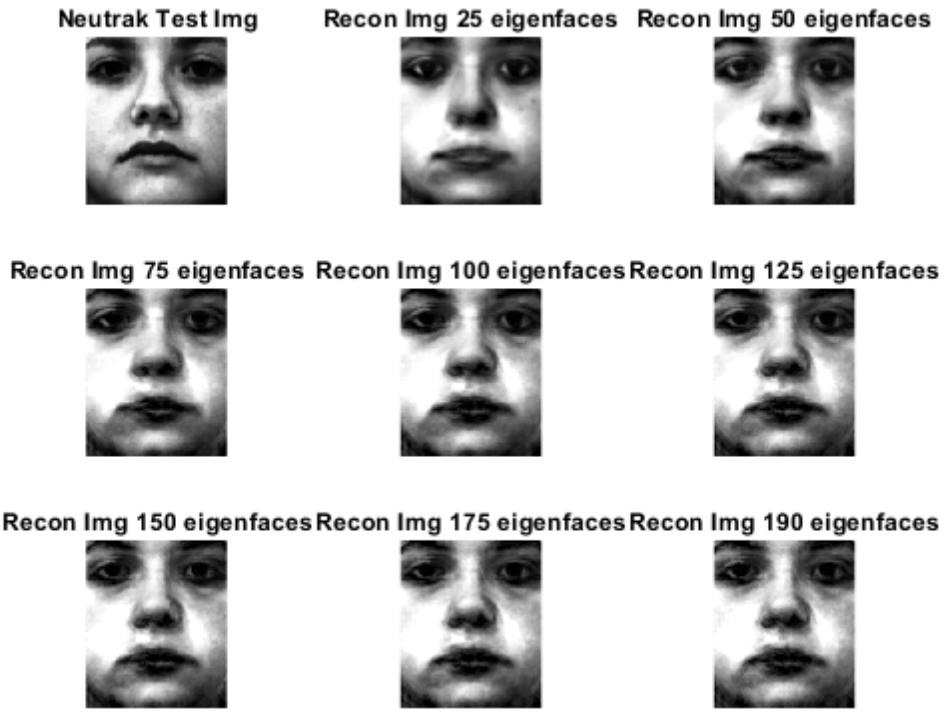
```
im = reshape(im, irow, icol);
imshow(im);
title(sprintf('Non Face Image'));
MSEs = zeros(cutoff, 1);
for count = 1:eigenfaces_count-1
    reco = reconstruct_face(Eigenfaces_PCA, count, tra_neu_mean, ...
                            car_image);
    cur_image = car_image;
    MSEs(count) = sum((reco - cur_image).^2, 1) / size(reco, 1);

    if 0 == rem(count, 25)
        plot_count = plot_count + 1;
        subplot(3,3,plot_count);
        % Reconstruction image
        reco = reshape(reco, irow, icol);
        imshow(reco);
       title(sprintf('Recon Img %i eigenfaces', count));
    end
end

figure()
stem([1:eigenfaces_count], MSEs, 'o')
yl = ylim; ylim([0, yl(2)]);
title('MSEs plot, Non Face Image Reconstruction')
xlabel('number of eigen faces used in reconstruction')
ylabel('MSE')
```

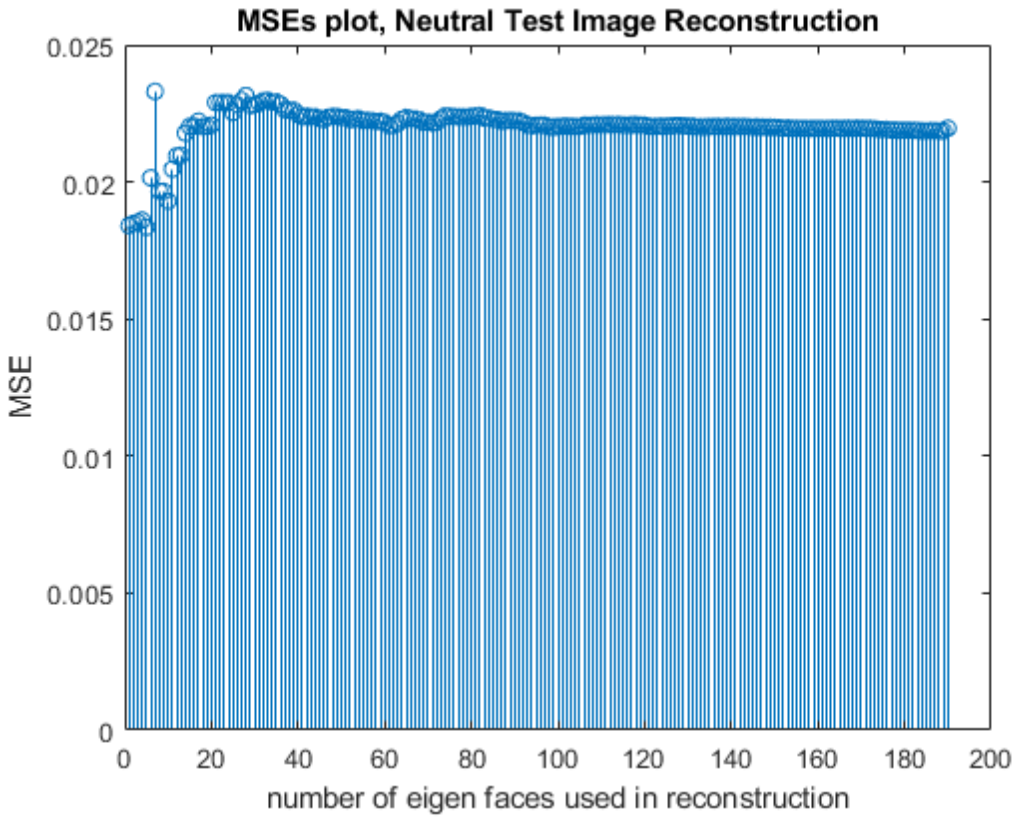

Neutral Tra Img
Recon Img 25 eigenfaces
Recon Img 50 eigenfaces
Recon Img 75 eigenfaces
Recon Img 100 eigenfaces
Recon Img 125 eigenfaces
Recon Img 150 eigenfaces
Recon Img 175 eigenfaces
Recon Img 190 eigenfaces

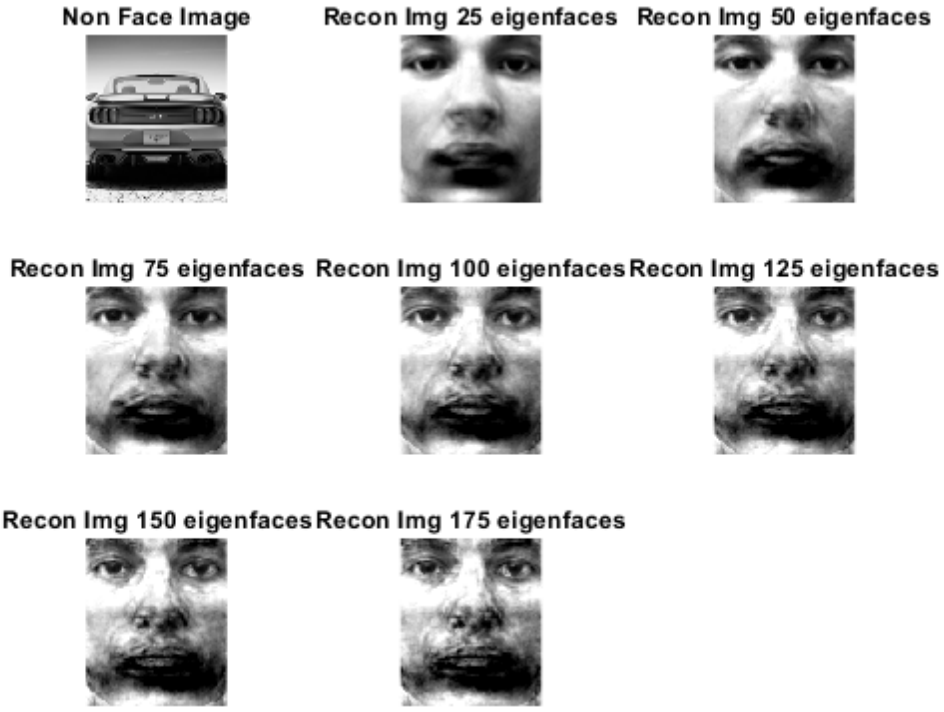

MSEs plot, Neutral Train Image Reconstruction
0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
20
40
60
80
100
120
140
160
180
200

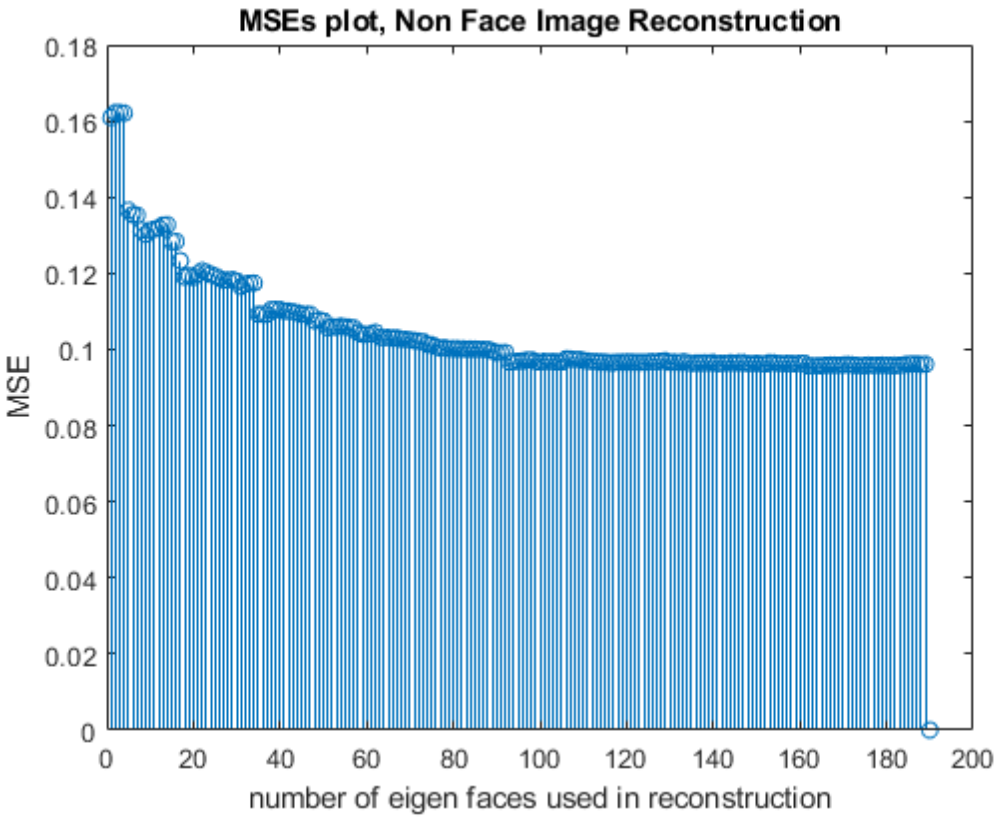

Smiling Tra Img
Recon Img 25 eigenfaces
Recon Img 50 eigenfaces
Recon Img 75 eigenfaces
Recon Img 100 eigenfaces
Recon Img 125 eigenfaces
Recon Img 150 eigenfaces
Recon Img 175 eigenfaces
Recon Img 190 eigenfaces


MSEs plot, Smiling Train Image Reconstruction
MSE
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
20
40
60
80
100
120
140
160
180
200
number of eigen faces used in reconstruction


Neutrak Test Img
Recon Img 25 eigenfaces
Recon Img 50 eigenfaces
Recon Img 75 eigenfaces
Recon Img 100 eigenfaces
Recon Img 125 eigenfaces
Recon Img 150 eigenfaces
Recon Img 175 eigenfaces
Recon Img 190 eigenfaces


MSEs plot, Neutral Test Image Reconstruction
MSE
number of eigen faces used in reconstruction
0
0.005
0.01
0.015
0.02
0.025
0
20
40
60
80
100
120
140
160
180
200




*Published with MATLAB® R2019b*

```
% RUN ME!!
% Project_Data_Processing
clc; clear all;
load Project_Data;

face_id = 18;
face = reshape(Train_Neutral(:, face_id), irow, icol);
degrees = linspace(0, 360, 9);
degrees = degrees(1:8);
eigenfaces_count = cutoff;

% All the rotation operations are counter-clockwise

index = 0;
for j = 1:4
    figure()
    for i = 1:2
        index = index + 1;
        degree = degrees(index);
        J = imrotate(face,degree,'bilinear','crop');

        cur_image = J(:) - tra_neu_mean;
        reco = reconstruct_face(Eigenfaces_PCA, eigenfaces_count, tra_neu_mean, cur_image);
        MSE = sum((reco - J(:)).^2, 1) / size(reco, 1);
        fprintf("MSE for %i degrees", degree);
        MSE

        subplot(2, 2, i * 2 - 1);
        im = reshape(J, irow, icol);
        imshow((im));
        title(sprintf('Rotated Img, %i degrees', degree));
        % Reconstruction image
        subplot(2, 2, i * 2);
        reco = reshape(reco, irow, icol);
        imshow(reco);
        title('Reconstruction Image');
    end
end
```

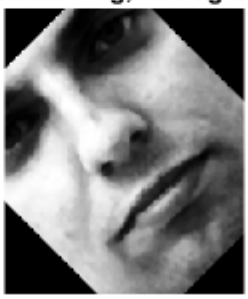
```
MSE for 0 degrees
MSE =

   7.9501e-07

MSE for 45 degrees
MSE =

    0.0445

MSE for 90 degrees
MSE =

    0.0428

MSE for 135 degrees
MSE =
```

```
        0.0448

    MSE for 180 degrees
    MSE =

        0.0254

    MSE for 225 degrees
    MSE =

        0.0481

    MSE for 270 degrees
    MSE =

        0.0428

    MSE for 315 degrees
    MSE =

        0.0496
```

Rotated Img, 0 degrees

Reconstruction Image

Rotated Img, 45 degrees



Reconstruction Image

Rotated Img, 90 degrees

Reconstruction Image

Rotated Img, 135 degrees

Reconstruction Image

Rotated Img, 180 degrees

Reconstruction Image

Rotated Img, 225 degrees

Reconstruction Image

Rotated Img, 270 degrees

Reconstruction Image

Rotated Img, 315 degrees

Reconstruction Image

*Published with MATLAB® R2019b*

```
function reco = reconstruct_face(Eigenfaces_PCA, eigenfaces_count,...
                                 tra_neu_mean, image)
    feat_len = size(Eigenfaces_PCA, 1);
    train_len = size(Eigenfaces_PCA, 2);
    weights = zeros(eigenfaces_count, 1);
    reco = zeros(feat_len, 1);
    % In the below loop we are finding the weights corresponding to each
    % eigen face. This is outlined in section 2.2 of the main paper
    for i = 1:eigenfaces_count
        weights(i) = Eigenfaces_PCA(:, i)' * image;
        % After gettting each weight the reconstruction is formed by a
        % weighted sum of the eigenfaces. This is Outlined in section 2.3
        % of the main paper.
        eigen_sum = weights(i) * Eigenfaces_PCA(:, i);
        reco = reco + eigen_sum;
    end
    % Finally, the reconstruction is still in shifted. We should shift it
    % by the train mean so that it is correct. As outlined in section 2.3
    % and 2.4 o the main paper.
    reco = reco + tra_neu_mean;
end
```

```
Not enough input arguments.

Error in reconstruct_face (line 3)
    feat_len = size(Eigenfaces_PCA, 1);
```

---

*Published with MATLAB® R2019b*