**ECE3093 Assignment 1 Part A**

* **Task 1: Face Recognition by Eigenfaces**

1. **Modified MATLAB code**

clc;clear;

% prepare the data for face recognition

train\_dir = '/Volumes/PKBACK# 001/ECE3093 Optimisation Estimation and Numerical Methods/Assignments/Assignment 1/eigenfaces/Train'; % change this to wherever the unzipped Face Recognition\Train folder has been saved

img\_dir = dir(fullfile(train\_dir, '\*.jpg'));

img\_num = length(img\_dir);

person\_num = img\_num; % each person has one image, so #persons = #images

traindata = zeros(img\_num, 28\*34); % #images = img\_num, image size = 28\*34.

correct\_rate= zeros(0,952);

% read all images into the matrix traindata:

for i=1:img\_num

img = imread(fullfile(train\_dir, img\_dir(i).name));

img = double(img)/255; % normalize to double in [0,1]

traindata(i,:) = img(:);

end

% train eigenface:

[eigenV,SCORE,LATENT] = princomp(traindata); % alternatively could be done using the svd(traindata) function

% eigenV: eigen vectors.

% SCORE: projection vectors of training data on the eigen vectors.

% LATENT: eigenvalues

%The eigenvalues plot

figure(1)

plot(LATENT(1:50))

xlabel('Index');

ylabel('Eigenvalues');

title('Eigenvalues,First 50');

figure(2)

plot(LATENT)

xlabel('Index');

ylabel('Eigenvalues');

title('Eigenvalues');

%The cumulative sum of eigenvalues and plot

cumLATENT=cumsum(LATENT);

figure(3)

plot(cumLATENT(1:50))

xlabel('Index');

ylabel('Eigenvalues Cumsum');

title('Eigenvalues Cumsum,First 50');

figure(4)

plot(cumLATENT)

xlabel('Index');

ylabel('Eigenvalues Cumsum');

title('Eigenvalues');

for loop = 1:952

pc\_num = loop; % suppose we keep the first 10 Principal Components:

eigenfaces = eigenV(:, 1:pc\_num);

gallery = SCORE(:, 1:pc\_num);

% test:

test\_dir = '/Volumes/PKBACK# 001/ECE3093 Optimisation Estimation and Numerical Methods/Assignments/Assignment 1/eigenfaces/Test'; % change this to wherever the unzipped Face Recognition\Test folder has been saved

img\_dir = dir(fullfile(test\_dir, '\*.jpg'));

img\_num = length(img\_dir);

testdata = zeros(img\_num, 28\*34); % # images = img\_num, image size = 28\*34.

for i=1:img\_num

img = imread(fullfile(test\_dir, img\_dir(i).name));

img = double(img)/255; % normalize to double in [0,1]

testdata(i,:) = img(:);

end

% project onto the eigenfaces:

meanV = mean(traindata);

features = (testdata-repmat(meanV,size(testdata,1),1))\*eigenfaces;

correct = 0; % counter for correct recognitions

for i=1:img\_num

f = features(i,:);

dis = sum((repmat(f, person\_num, 1) - gallery).^2, 2);%Calculate the distance between features and gallery

[B, idx] = sort(dis);

top3rank = idx(1:3); % see whether the correct match is in the top 3 ranked images

if ismember(i, top3rank)

correct = correct+1;

end

end

correct\_rate(loop) = correct/img\_num;

end

figure(5)

plot(correct\_rate)

title('Correct Rate')

xlabel('Principle Component')

ylabel('Correct Rate')

figure(6)

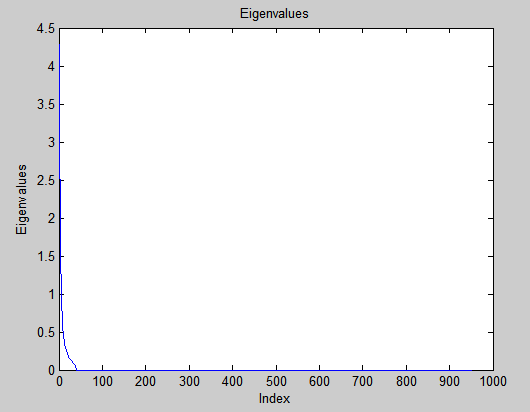
plot(correct\_rate(1:50))

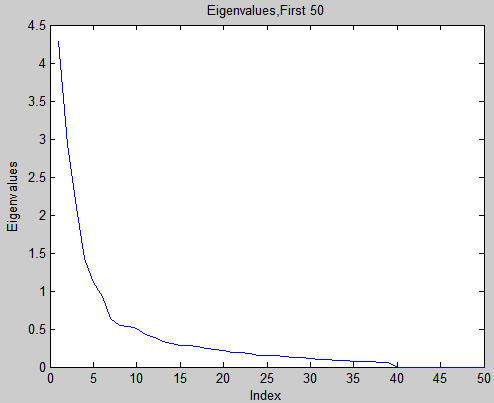
1. **Plot and Results**

The code above plots the following graphs

**- Eigenvalues**

Figure 1 shows the eigenvalues in descending order with a majority of the eigenvalues being zeros. Figure 1 b) indicates that the first 39 elements are non-zero values.

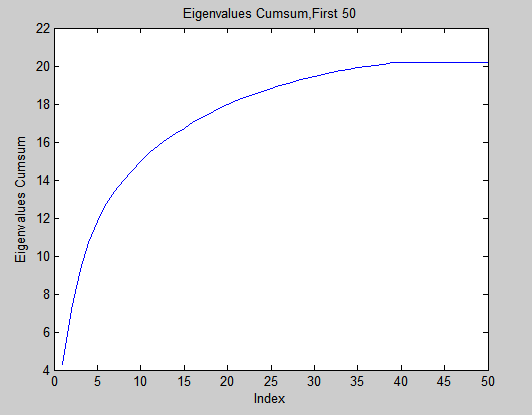
a)

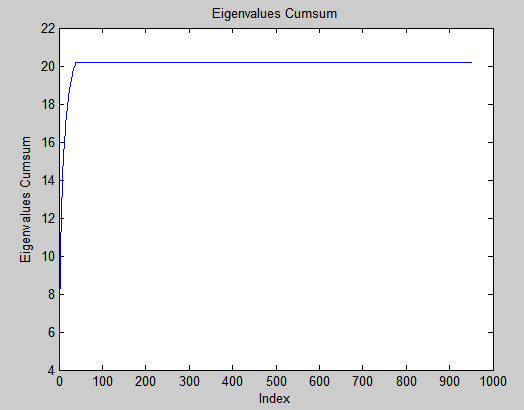
b)

*Figure 1 Plots of a) 952 b) 50 eigenvalues*

**- Eigenvalues Cumulative Sums**

It is suggested that we calculate and plot a cumulative sum of the eigenvalues, which is shown in Figure 2. The curve starts flattening at 40 indicating only the first 39 values contribute to the variation of the data. This corresponds with the conclusion we have drawn above.

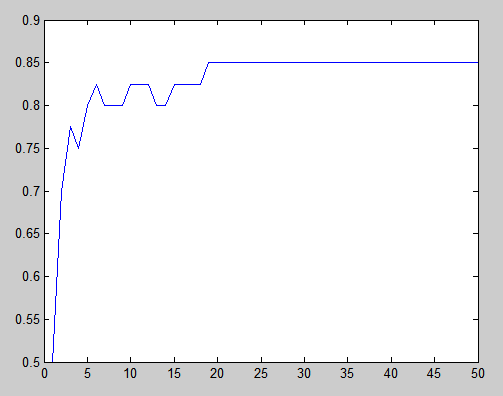
a)

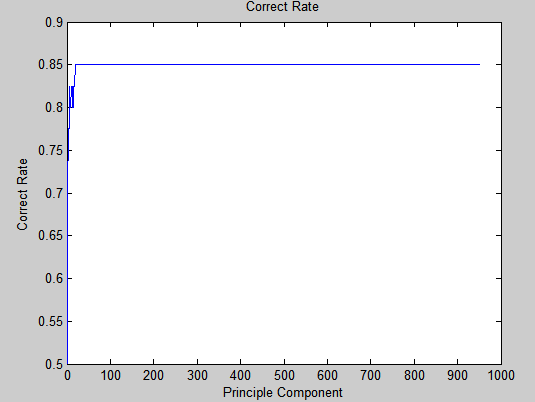
b)

*Figure 2 Plots of a) 952 b) 50 eigenvalues*

**- Correct Rate with varying principle components**

Figure 3 shows the correct rates with different principle components. The maximum correct rate is observed to be 0.85. The curve starts from zero, keeps rising and attains this maximum value when the principle component is 19. It then retains the value until the end.

a)

b)

*Figure 3 Plots of correct rates versus principle components with a) 50 b) 952 principle components*

1. **Discussion and Answers**

**How does the recognition accuracy on the test set vary with the number of principal components selected?**

As shown in Figure 3, the curve has a rising tendency. This indicates that the accuracy is directly correlated with the number of principal components.

**What is the best number of principal components to retain to maximize the accuracy on the test set?**

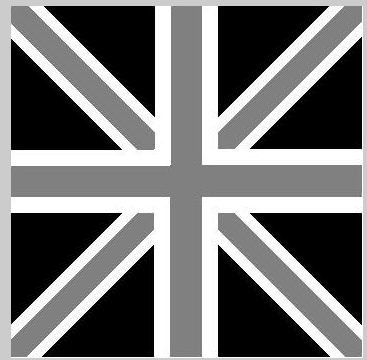
The best number of principal components to maintain the highest correct rate is 19. It is shown in Figure 3 and 4 that the smallest number to retain the maximum accuracy 0.85 is 19. In other word any principal components that is greater than 19 produce the same result. Therefore, it is redundant and inefficient to use those greater values.



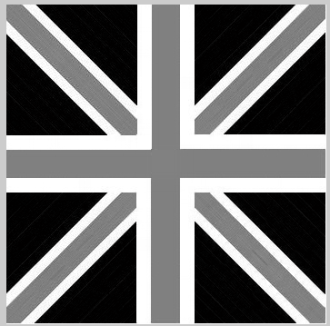
Figure 4 Correct rate values with varying number of principal components

* **Task 2: Image Compression**

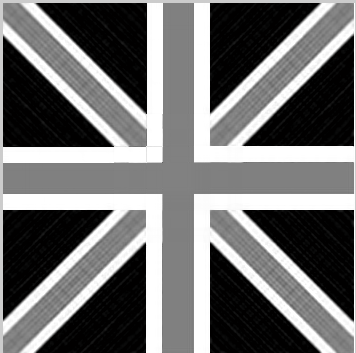
1. **Images**
2. Original Image with all 352 eigenvectors



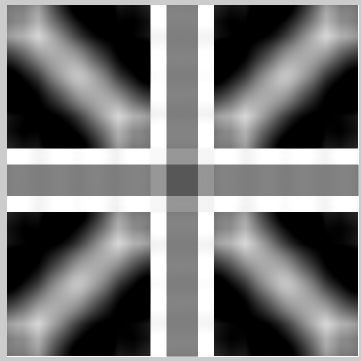
1. No visible degradation, 120 eigenvectors



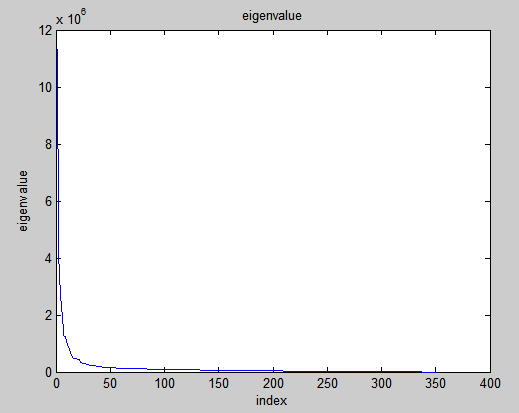
1. Onset of degradation, 35 eigenvectors

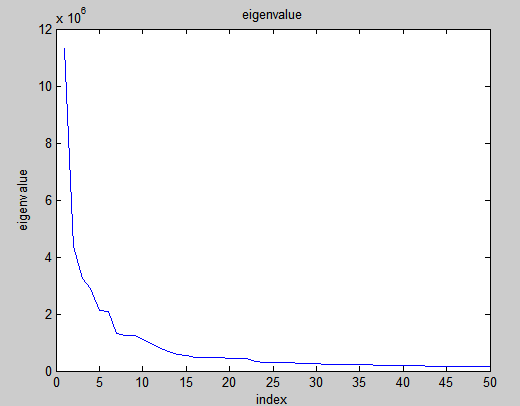


1. Heavy degradation, 5 eigenvalues



1. **Discussion**

a)

b)

*Figure 5 Plot of eigenvalues with a) 352 b) 50 values*

Figure 5 shows the eigenvalues in descending orders. It can be observed the largest 35 eigenvalues are more significant comparing to the rest. This indicates that the 35 eigenvalues are able to retain the quality of the image without distinguishable degradation, which is in accord with the images shown above

1. **M-file Code**

clc;clear;

origin = imread('unionjack.png');

origin\_d= double(origin);

%eigen decomposition

[evec eval] = eig(origin\_d);

%sort the absolute values

sorted=sort(abs(eval),'descend');

%excerpt the largest values

[sortedeigen idx]=sort(sorted(1,:),'descend');

plot(sortedeigen(1:50))

title('eigenvalue');

xlabel('index');

ylabel('eigenvalue');

keepV= 35; %the values that we keep

nidx=idx(keepV:end);%corresponding index of colums

for i=1:max(size(nidx))

eval(:,nidx(i))=zeros(max(size(eval)),1);%make the insignificant eigenvalues zero

end

figure(2)

imshow(uint16((evec\*eval\*transpose(evec)))) %reconstruction