COMPUTER VISION

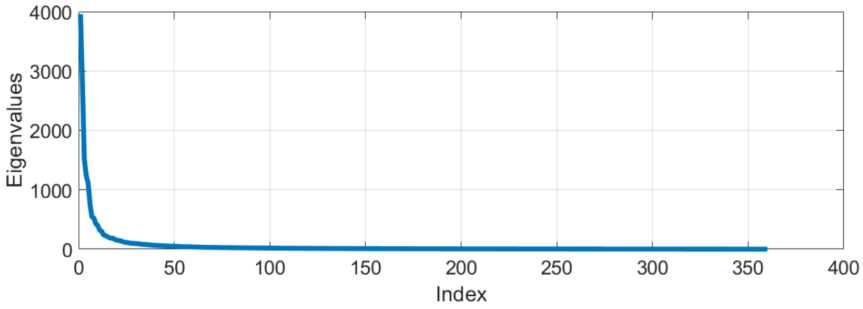
LAB#9

GOKTUG KORKULU
27026

INTRODUCTION

- In this section of the lab, I will write a program to detect whether a given image is a face or not using the eigenfaces algorithm. There is an an image dataset consisting of 10 images (56×46) taken from different views of 40 people. I will use 9 images per person to build my model and the rest to test its performance. Note that, in the given dataset the images are already flattened and concatenated therefore it has 2576 (56×46) columns and 400 rows. A sample from the dataset is shown in the figure below:

-I will first need to process my image dataset to compute the eigenfaces and then choose only the most effective K of them corresponding to the largest eigenvalues. To choose K properly, I will look at the at the decay of the eigenvalues after plotting them as shown below.

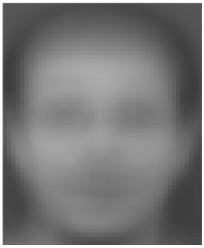


-After selecting my K eigenvectors, I will show the first 12 computed eigenfaces as follows:

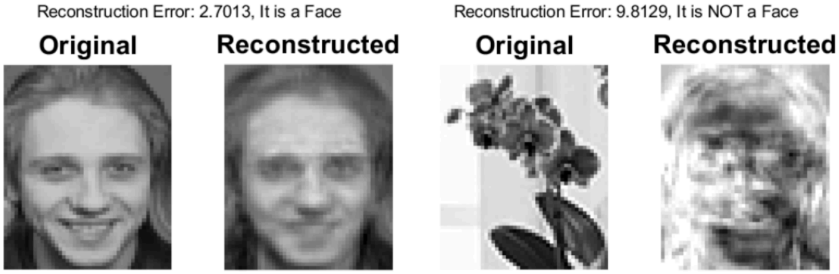
-Then given a new image, I will decide if it is a face or not by making use of the equation below and specifying a proper threshold value.

$$\|x-(x+a_1v_1+a_2v_2+\cdots+a_kv_k)\|<\text{Threshold}$$

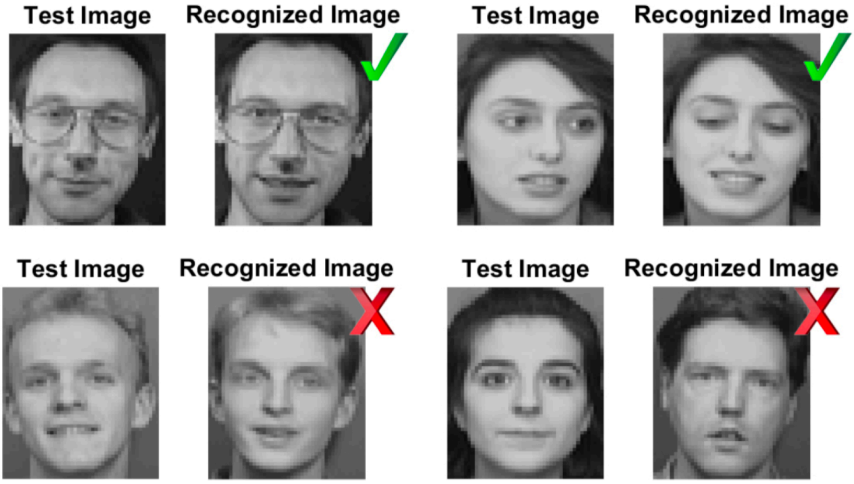
where, x is a new image in the vector form (row.col × 1), x is the vector form of the mean image, v_i is the computed eigenvectors and $a_i = (x - x)^T v_i$. My calculated x will look as follows:



- Finally, my resulting images will look as follows:



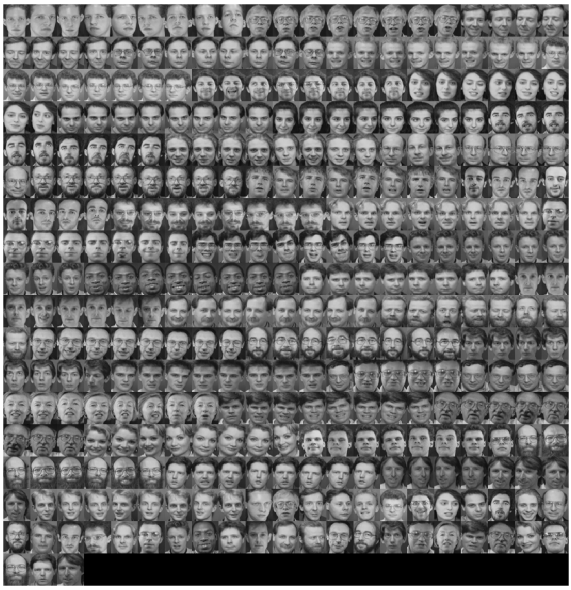
- Face Recognition: Once I will be able to detect faces, I will build a database of descriptors composed of the calculated ai for each face image in my training set. Then given a new face, will calculate its descriptor and compare it with the ones in my dataset using Euclidean distance metric to recognize it. My results will look as follows:



METHOD

```
%% Step 1: load the faces.mat matrix and
% display the faces for visualization purposes
clear all; close all; clc;
face = load('faces.mat').faces;
plot_face = reshape(face(:, 1:40), [56,46,1,40]);
montage(plot_face);
face = double(face);
test_face = [];
train_face = [];
k=1;
for i= 1:400
    if mod(i,10)==0
        test_face = [test_face, face(:,i)];
    else
        train_face = [train_face, face(:,i)];
    end
end
plot_face = reshape(uint8(train_face), [56,46,1,360]);
montage(plot_face);
```

Step 1: I have loaded the data matrix called faces.mat which keeps the information of 400 amount 56x46 sized images of 40 people in total and displayed the resulting images as below.



```
%% Step 2: mean face of all
mean_face = mean(test_face,2);
figure;
imshow(uint8(reshape(mean_face,56, 46)));
```

Step 2: I have calculated the mean face.

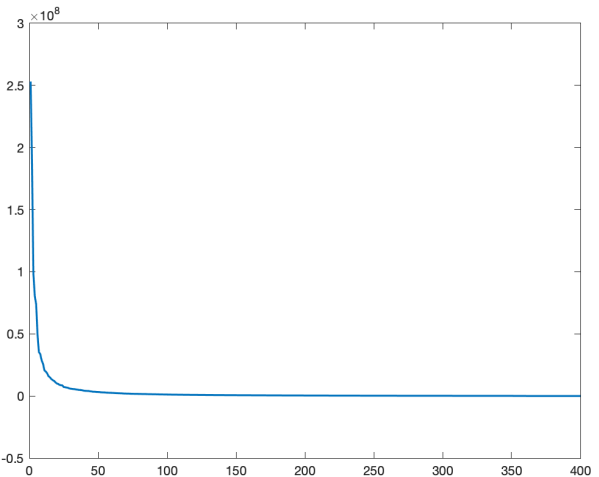
```
%% Step 3: Subtract mean from the dataset train_face and Compute A
= S2*S2'
Horse = train_face - mean_face;
A = Horse*Horse';
```

Step 3: I have calculated the variable called ‘Horse’ by subtracting mean face from the train face dataset. Then computed A = Horse*Horse’

```
%% Step 4: compute and plot eigenvalues.
[V,D] = eigs(A, 400);

plot(diag(D), 'LineWidth',1.5);
```

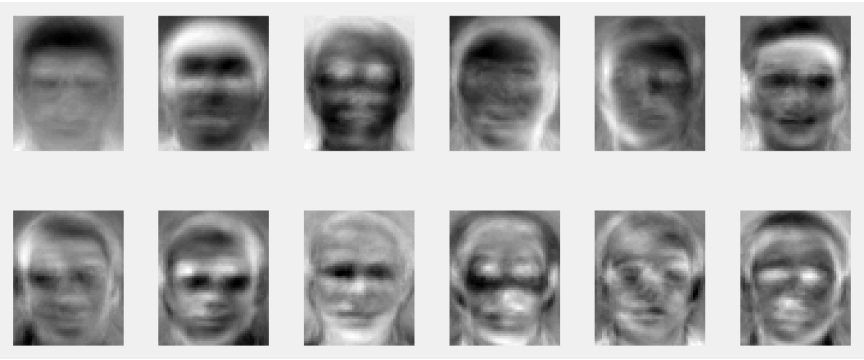
Step 4: I have calculated the eigenvalues of A. Then plotted so that we can see the significantly valuable values from the graph.



```
%% Step 5:
%I selected k=50 so that;
V = V(:,1:50);

figure
for i=1:12
    subplot(2,6,i);
    imshow(reshape(V(:,i),56,46),[]);
    colormap(gray);
end
```

Step 5: After selecting my K eigenvectors above, I showed the first 12 computed eigenfaces as follows:



```
%% Step 6: Face detection
test1 = test_face(:,30); %any number between 1-40

a_weights = (test1-mean_face)'*V;
test1_recovered = (mean_face + V*a_weights');

err = immse(test1_recovered, test1);
Threshold = 250; %sth more than 197
if err < Threshold
    disp('It is a face')
end

figure;
subplot(1,2,1),imshow(uint8(reshape(test1_recovered, [56, 46])));
subplot(1,2,2),imshow(uint8(reshape(test1, [56, 46])));
```

Step 6: By utilizing the equation $\|x-(x+a_1v_1+a_2v_2+\cdot+a_kv_k)\|<$Threshold, I have decided whether arbitrarily taken image is a face or not. Here are different examples.

Test face [30]

Test face [20]

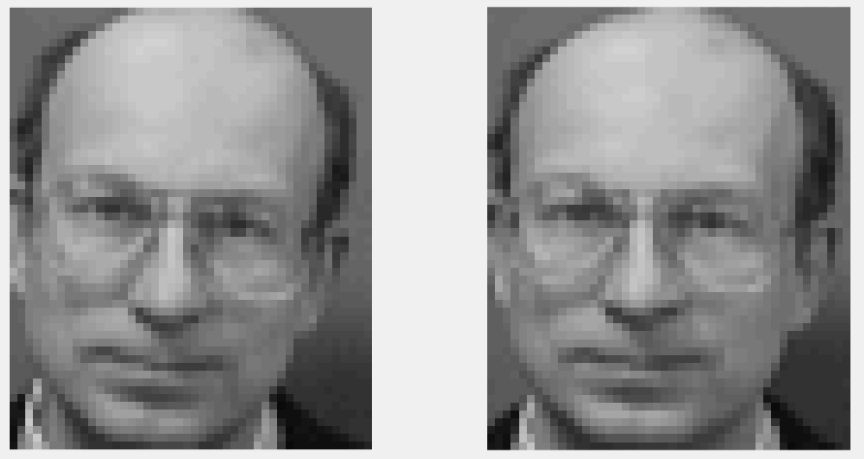
```
%% Step 7: Face Recognition
test2 = test_face(:,6);
test2_weights = (test2 - mean_face)'*V;
test2_recovered = (mean_face + V*test2_weights');
err=[];

weights = (train_face - mean_face)'*V;

for i = 1:length(weights)
    err(i) = norm(test2_weights - weights(i,:));
end
err_min = min(err);
ind = find(err_min == err);

figure;
subplot(1,2,2), imshow(uint8(reshape(train_face(:,ind), 56, 46)));
subplot(1,2,1), imshow(uint8(reshape(test2, 56, 46)));
```

Step 6: Face Recognition: Once I am able to detect faces, I built a database of descriptors composed of the calculated ai for each face image in my training set before. Then given a new arbitrary face, calculated its descriptor and compare it with the ones in my dataset using Euclidean distance metric to recognize it.



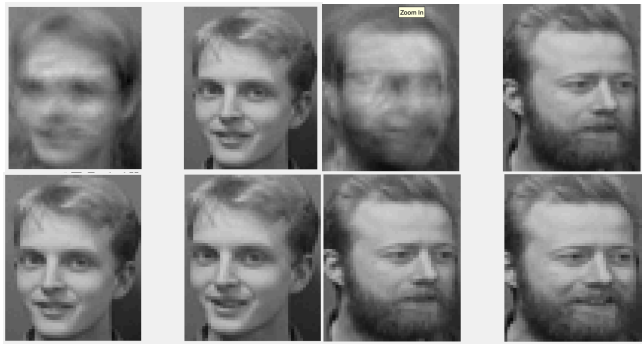
Test face [11]

CONCLUSION

- By general overlook, I have written a program to detect whether a given image is a face or not using the eigenfaces algorithm.
- There was an an image dataset consisting of 10 images (56×46) taken from different views of 40 people and I have used 9 images per person to build my model and the rest to test its performance.
- I first processed my image dataset to compute the eigenfaces and then choose only the most effective K of them corresponding to the largest eigenvalues. To choose K properly, I have looked at the at the decay of the eigenvalues after plotting them, i.e. I chose K = 50 in my case.
- Then given a new image, I decided if it is a face or not by making use of the equation above and specifying a proper threshold value.
- Finally, I have worked on Face Recognition. Once I was able to detect faces, I built a database of descriptors composed of the calculated ai for each face image in my training set. Then given a new face, have calculated its descriptor and compared it with the ones in my dataset using Euclidean distance metric to recognize it.

- In step 6 and 7, I have tried different images in order to check whether my algorithm works properly or not. Here are some different images chosen and their results.

- As seen, my algorithm and codes work well for different images.



Test_Face[40] Test_Face[23]