Assignment4 Report:

FACE RECOGNITION

CSE 691: Image and video processing

Cheng Wang(cwang76@syr.edu)

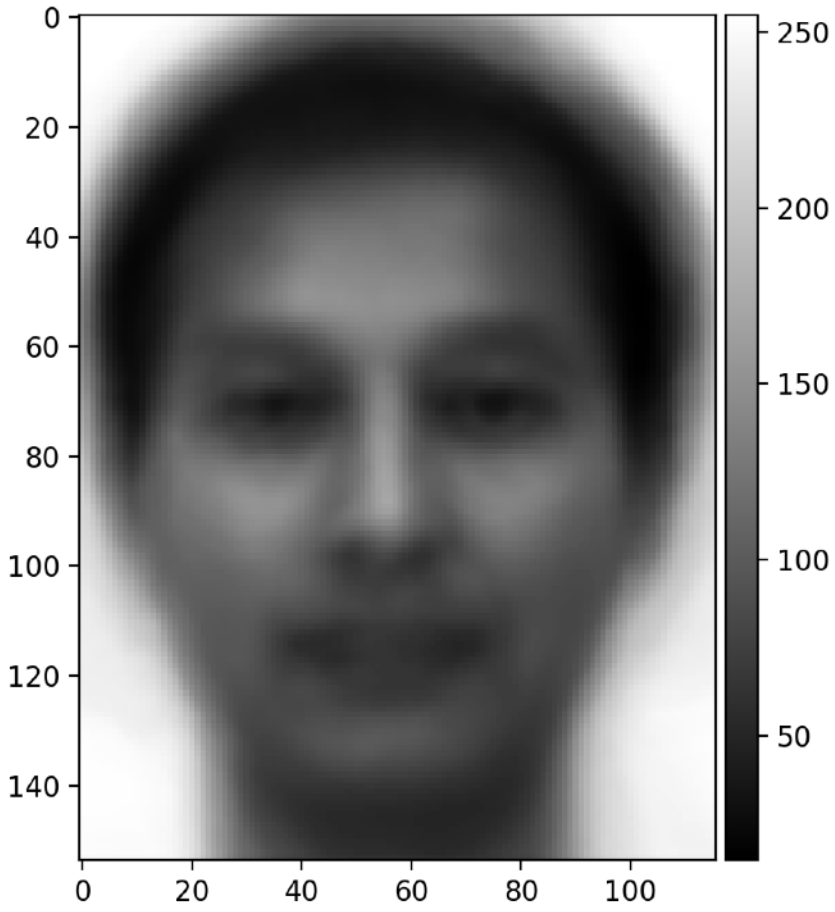
A. Eigenfaces(75%)

Implement a face recognition system using the Eigenfaces described by Turk and Pentland. First, randomly separate the faces in the database into training and test data.

a) [5%] Read images from the training set, and collect them in one matrix (Hint: Each image should be one column of this matrix). Then, find the average face of the training set, and display it (Hint: Help reshape).

I pick 2 images as test images, and other images as training images.

The mean image:



b) Perform PCA in the following ways:

i) [5%] Find the eigenvectors of theXXT, where X is the data set matrix.

ii) [5%] Use SVD (Hint: Please refer to the slides).

iii) [5%] Find the eigenvectors of XTX, then use the method described in the paper titled "Eigenfaces for Recognition" by Turk and Pentland.

Pleases refer to the three method functions in the code.

c) [10%] Compare the principal components from parts b.i, b.ii and b.iii. Measure the number of seconds required for each of the three methods in part (b) above (Hint: help tic, help toc).

Their principal components are the same. The first method takes too much time so that the result can't be measured. I waited for about 30 minutes but it didn't stop so it may due to the efficiency of Python. But when I put a small matrix, the method works just fine.

The third method takes the least time to finish.

Method 2 takes 59.35528302192688 s

Method 3 takes 0.1790761947631836 s

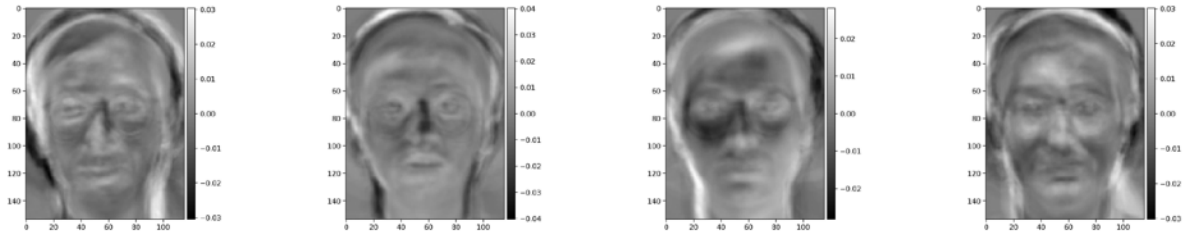
```
[[ 0.08190521  0.09608971  0.09627371 ... -0.28539244 -0.29322357
  -0.29147709]
 [-0.04794852 -0.03715353 -0.06068085 ...  0.11653111  0.12074547
   0.12043748]
 [ 0.06435292  0.07926536  0.04621278 ... -0.10327076 -0.11676466
  -0.12934578]
 ...
 [ 0.02487352 -0.03643909 -0.00856928 ...  0.00828782 -0.00466
  -0.0100309 ]
 [-0.07634673 -0.04584207 -0.05213894 ... -0.03228151 -0.0371455
  -0.04829381]
 [ 0.0359587   0.06072811  0.05201301 ... -0.05154103 -0.06001352
  -0.06675658]]
[[-0.06300123  0.00785745 -0.06770424 ... -0.04617854 -0.05961099
  -0.0381053 ]
 [ 0.05997998  0.06414816  0.0282756  ... -0.05881306 -0.05238831
  -0.07696037]
 [-0.07598533 -0.11587565 -0.07374644 ... -0.16942581 -0.12811159
  -0.06958058]
 ...
 [-0.02721924 -0.03476551 -0.00630717 ...  0.05103863 -0.01132111
  -0.00022377]
 [ 0.08396382  0.08856792  0.08522373 ...  0.07185764  0.08471158
   0.08859421]
 [-0.00814255 -0.0006806  -0.00249494 ... -0.00494626 -0.00475009
   0.00098107]]
```

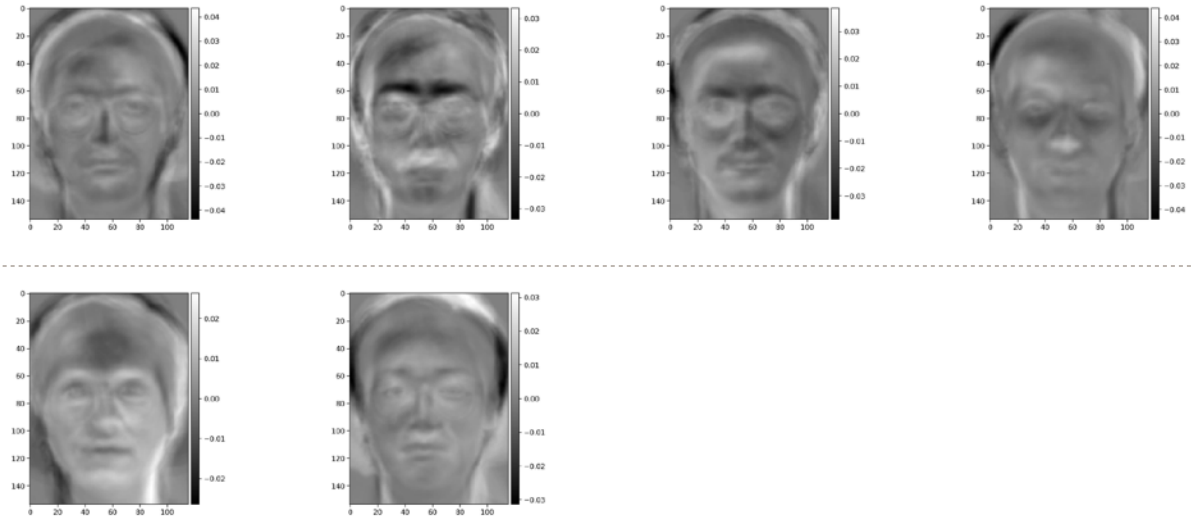
Take part of the result from the second and the third method, the order is different, but the value is the same.

d) [10%] Find the n significant eigenvectors with the largest associated eigenvalues. (Please refer to part 2(a) to experiment with different values of n.)

Pleases refer to the find_n_eigenvector function in the code.

Here are several Eigen faces.



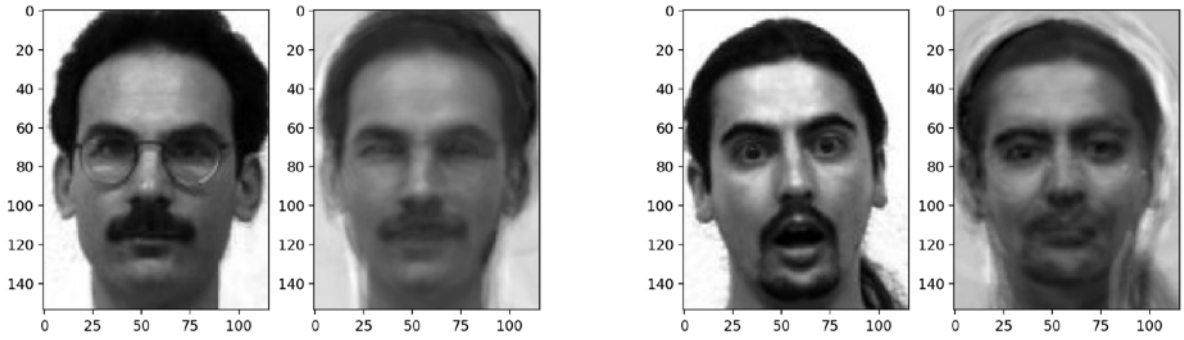


e) [15%] Read images from your test set. For each image, subtract the average image obtained in part 1(a), and project it onto the basis spanned by the top n eigenfaces. (This will give you the weights.)

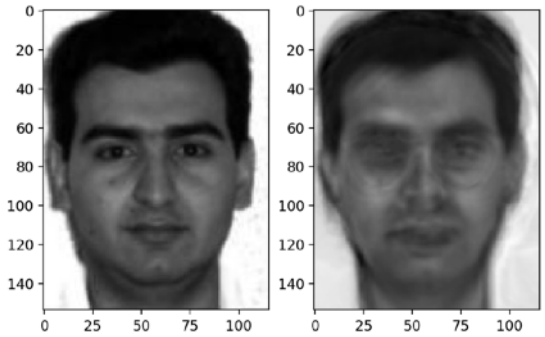
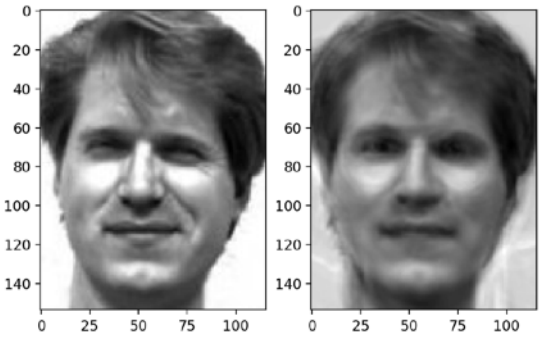
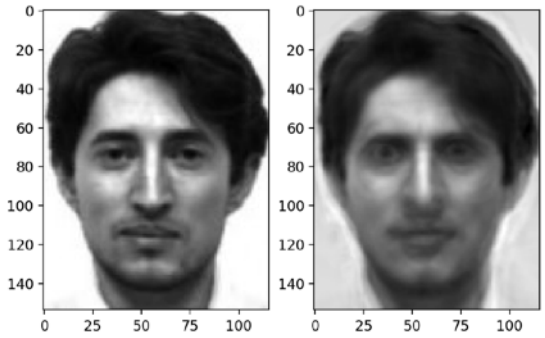
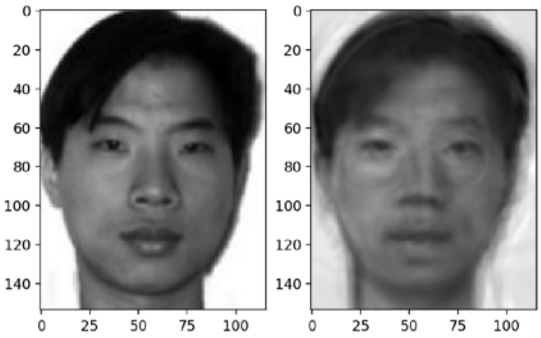
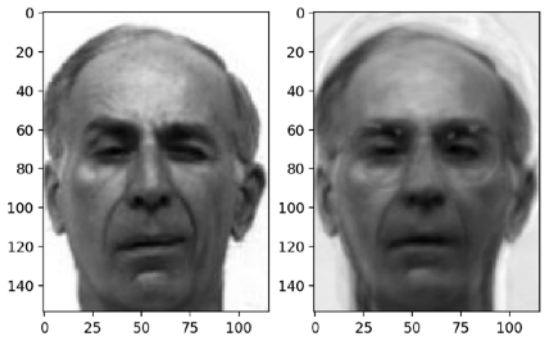
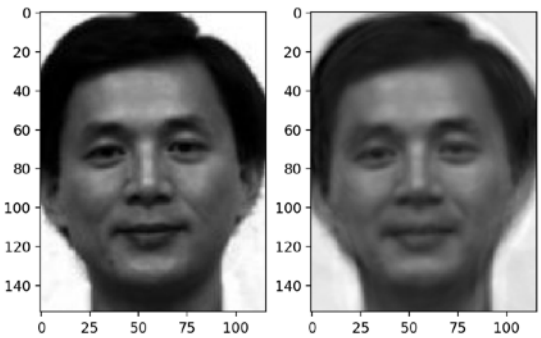
Pleases refer to the reconstruction function in the code.

f) [10%] Reconstruct the test images by using the weights from part 1(e) and the n eigenfaces. Show the original and reconstructed images for 10 of your test images (Hint: Do not forget to add the average image at the end).

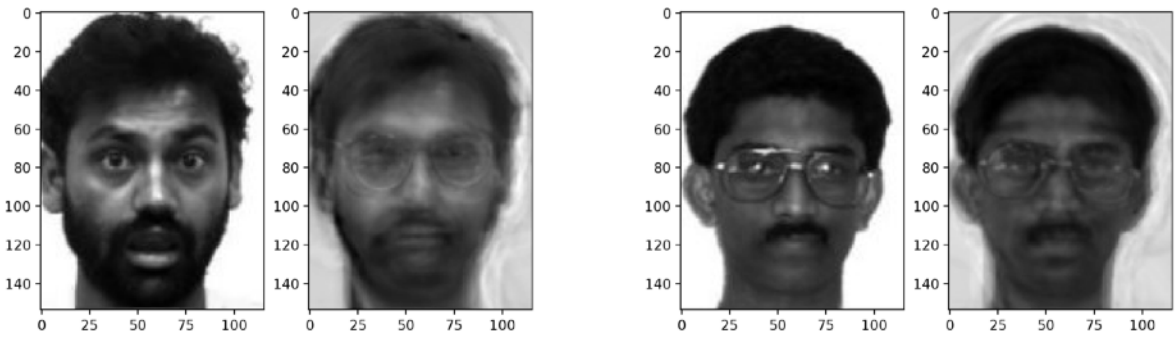
Reconstruct Image(with 50 eigenfaces)



Reconstruct Image(with 50 eigenfaces)

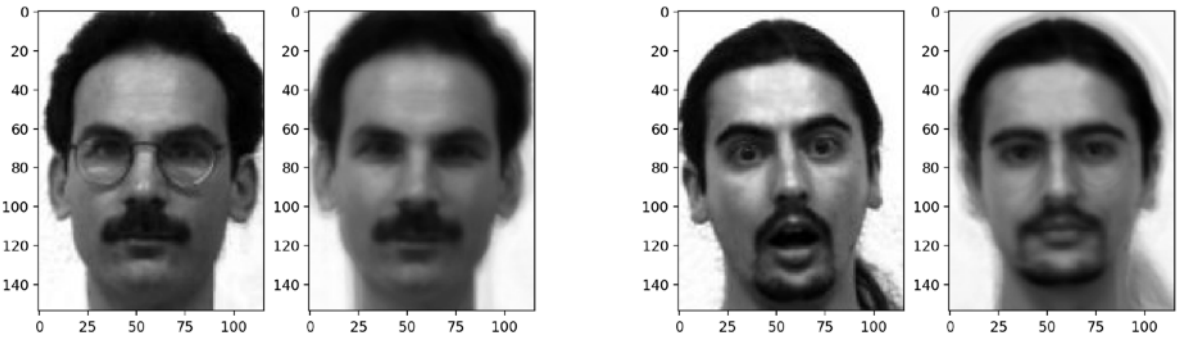







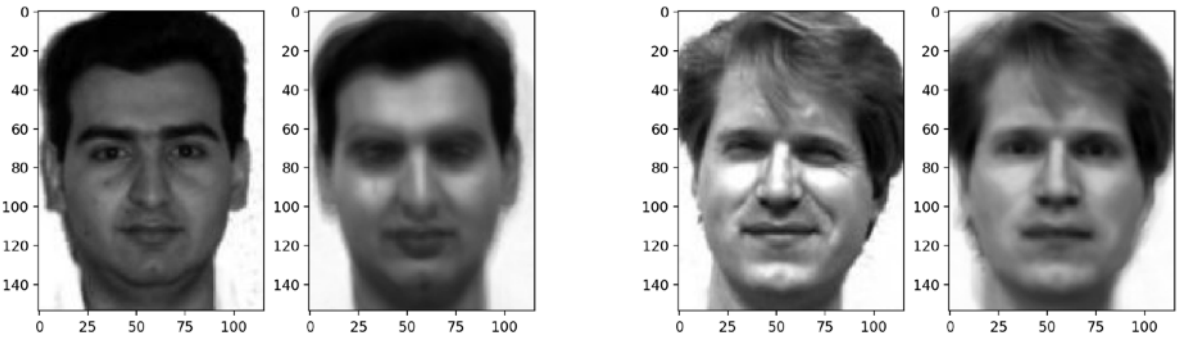
Reconstruct Image(with 50 eigenfaces)

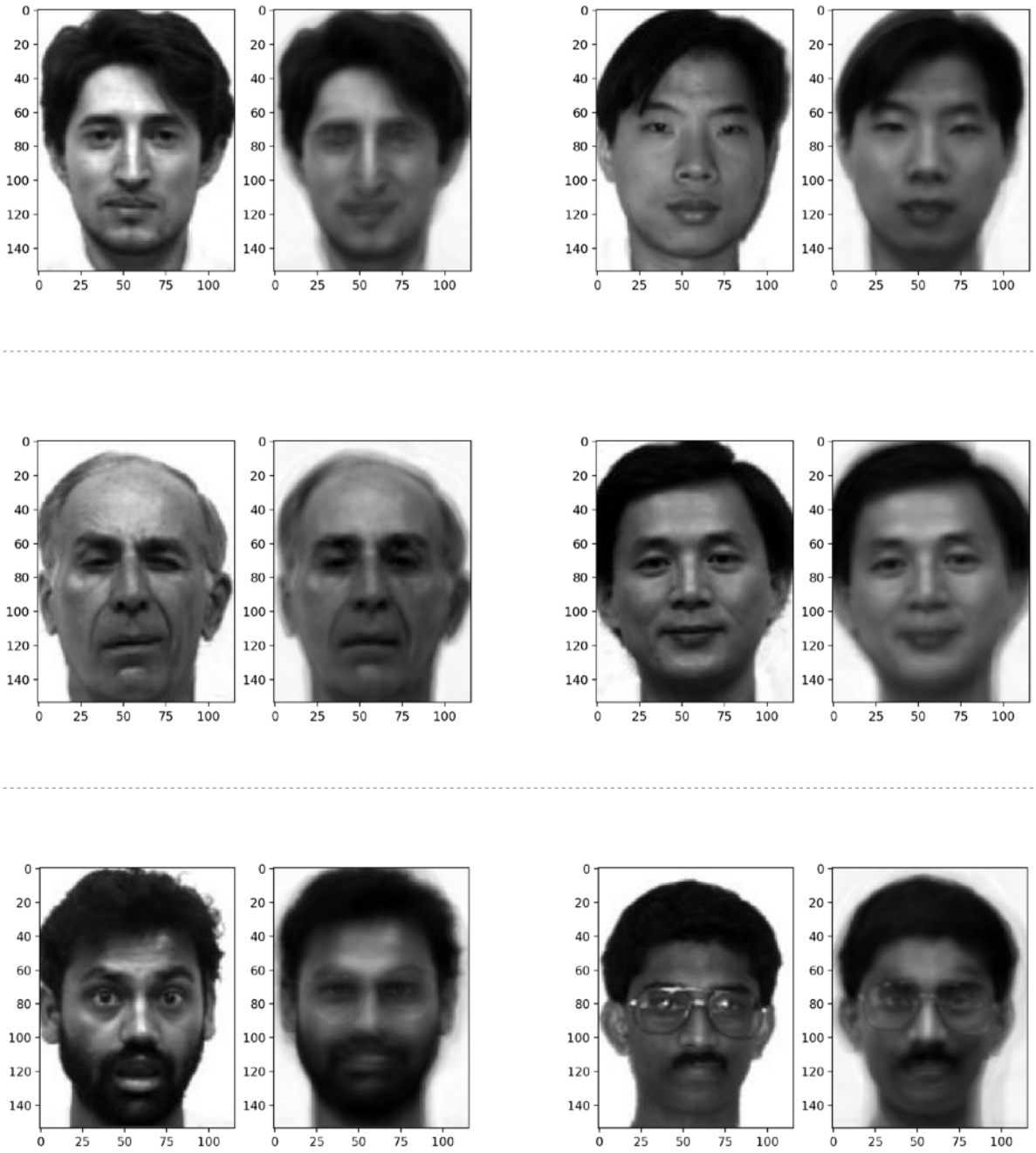


g) [10%] Pick 20 images from your test set, and find the closest image in the training data set to each of these 20 images. Show 10 of the matches as examples (Hint: you will use the weight vectors).

Match Images, 10 as examples





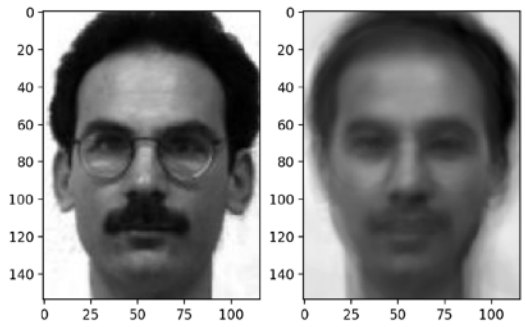


B. Questions, evaluation and comments (25%)

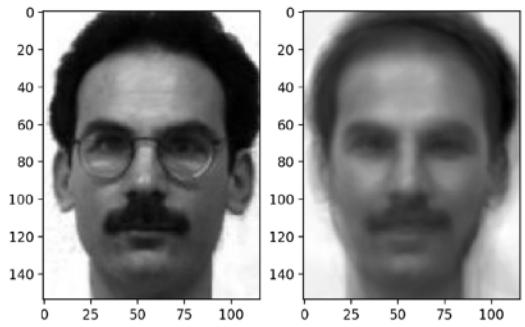
a) [5%] How many eigenfaces are necessary to obtain recognizable reconstructions? i.e. how big does n have to be before the reconstructed images look recognizable?

From 12, the image reconstructed is recognizable.

11 eigenfaces



12 eigenfaces



b) [5%] Calculate the recognition rate from part 1(g) above.

With 15 eigenfaces, the recognition rate can be 100%.

But When I change the train set and test it with other images. The recognition rate drops by four people. Different train sets may lead to different recognition rate.

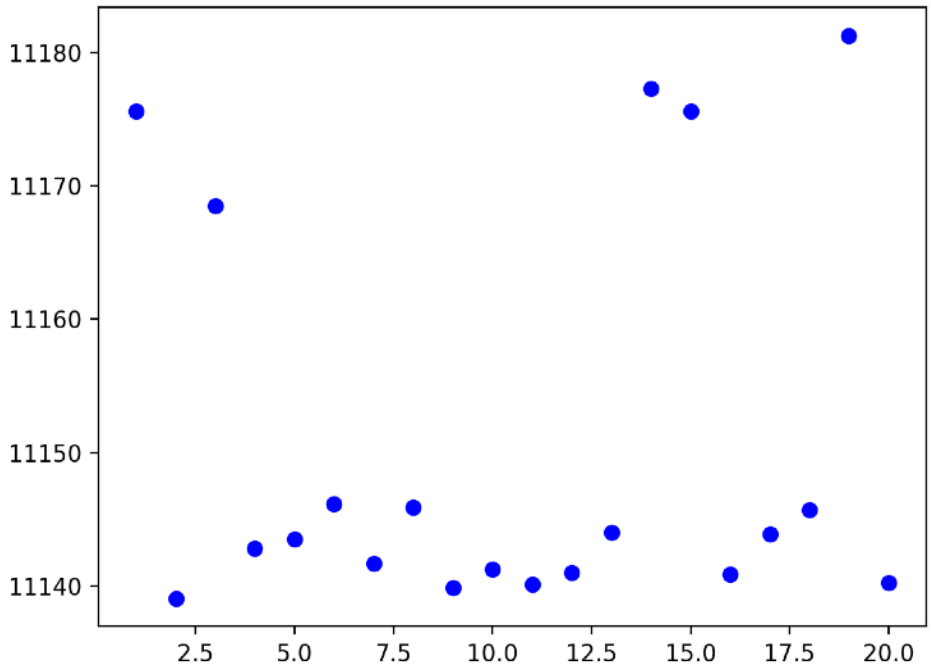
c) [5%] What happens when you do not include the images with glasses in your experiments? How does the recognition rate change? Why?

The recognition rate is 100%. Because the eigenfaces itself won't change too much. The factor of glasses takes a minor effect.

d) [5%] Project the provided non-face images onto the space spanned by the n eigenfaces. Find the difference between each image and its reconstructed version. Show

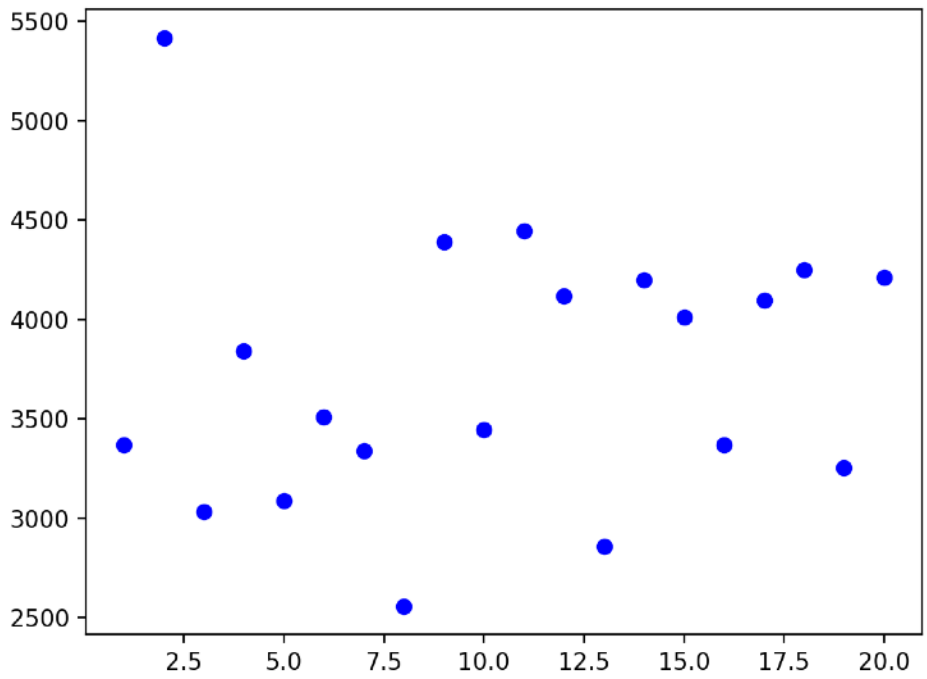
the original images, reconstructed images and the difference images. Find the Frobenius norm of each difference image, and plot these values. (You will need to resize these images to match their sizes to the database images.)

Frobenius Norm value of non-face image(20 images):



e) [5%] Repeat part 2(d) for 10 of the face images from the test set. (Note: You already have the reconstructed versions of the test images from part 1(f) above. Thus, you will only need to find the difference images, find the Frobenius norm of each difference image, and plot these values for the face images from your test set.) Comment on the results of parts 2(d) and 2(e).

Frobenius Norm value of face image(20 images):



Comments:

We can see from the two output images that the images reconstructed from non-face images are mainly a unknown image. The difference between original image and the reconstructed image is very huge so the shown image is primarily decided by the mean image. The norm values plotted show the difference around 11140-11180 which is huge number. But for the face images, the difference is very small which means the distance between the original and the reconstructed image is very similar. So this face recognition program works to classify the test faces and the non-faces can be eliminated due to the large distance between it and the face class.