David Meyer

Homework 4

**Running Home Work 4**

Requirements:  
 1. Matlab

2. Matlab Libraries

-None

Steps:

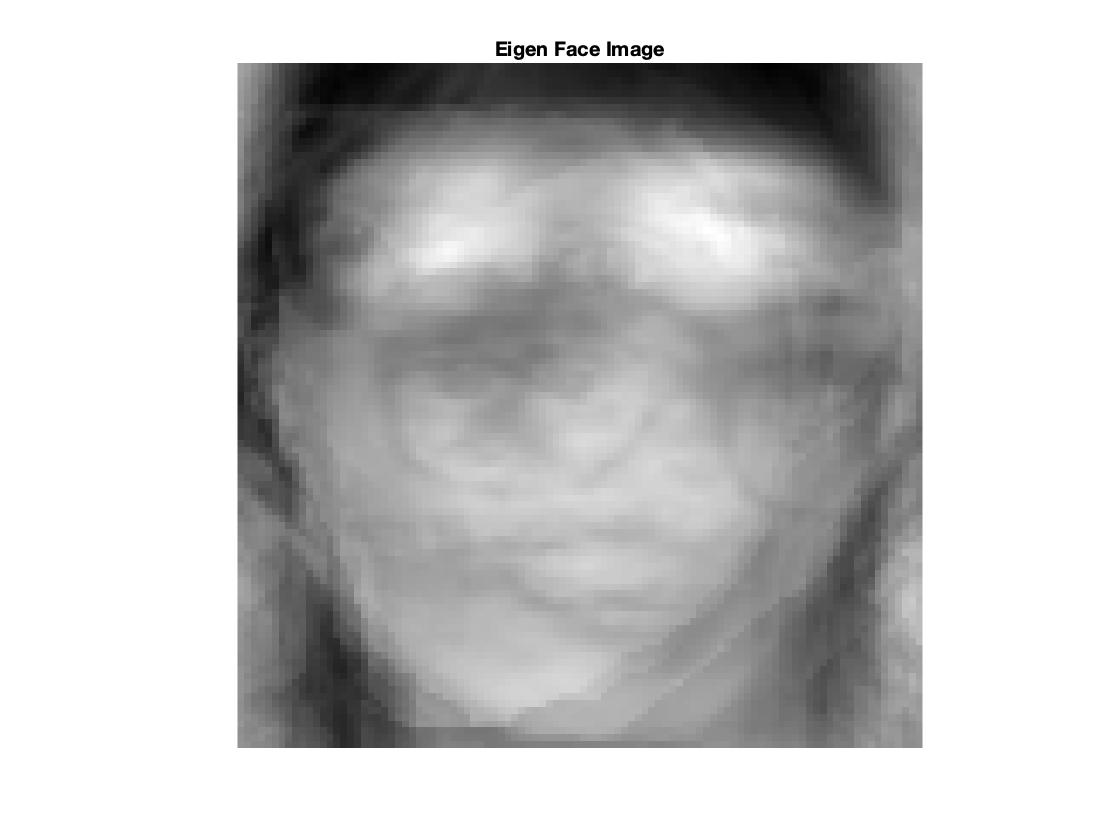
1. Open zip file and choose where to put the .m file
2. Open matlab and make sure that the same directory where you put both the .m file and the pictures is selected in the directory bar.
3. If the code is in the Editor section, the main part we write in within matlab, and the directory is in the correct spot, showing both the “MeyerFaceDetection.m” file then you can run the program by pressing the green arrow.

Pre-program:

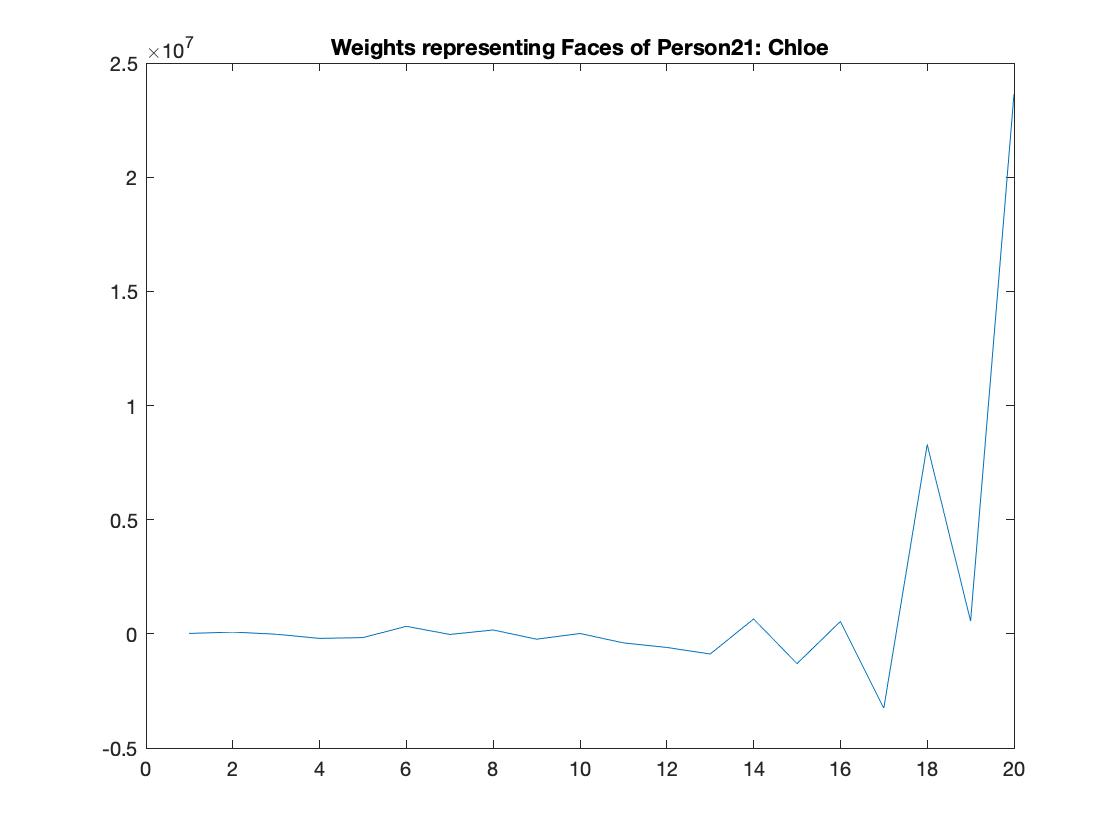
1. Use ipiccy website to upload photos of mine, and then crop out the face and delete everything else.
2. Load edited photos into matlab and change each one to grayscale.
3. Then resize all the images to match the size as the rest of the data set.
4. Manually move these photos into the dataset area to run with the rest of them and change the name to match the same format as the data set (ID45\_###) .

Program :

1. After the preprocessing of the images under the section of “Training” the true program begins. Here we load the dataset to make eigenvectors to “learn” from to compare to photos later. Make a matrix matching the number of people to learn called “x” as well as a giant matrix containing all the vectors of all the photos called “vector of peps”.
2. Loop through the data set and make the matrix’s.
3. Do the same thing to get a vector that represents the mean for all the people.
4. Make a matrix called “a” that contains each vector for each person after subtracting the mean vector.
5. Then use matrix “a” to get the eigenvectors matrix and eigenvector values to use in PCA.
6. Make the eigen faces, “ef”, by multiplying ata calculated in step 4 and p2 from step 5. Teaching is now done. (Custom added person, person 21)



1. To start the face recognition, load in images to match faces.
2. Then get weights, or eigen vector data, by subtracting the mean previously calculated from the new vectors to determine if any pictures match. (Here is the 21st persons weight, the person I added to the data set)



1. Compare the new weight to the old weight by subtracting and find the minimum euclidean distance for each picture to determine if it matches someone from the first data set.
2. First matching determines what it thinks most matches, while the second matching determines the final answer of matching by giving a zero and any other number it thinks might match. In this case, only index(person) 5 and 9 are matched perfectly. Index 4 is matched in the second matching.
3. Finally make cmc curve to determine accuracy.

Output:

1. The 20 eigenfaces of the people respectfully.
2. The 21 weights from the enrolled people.
3. CMC plot

Accuracy:

In the first matching, it’s only a 20% success with persons 5 and 9. In the second matching we get 30% percent with persons 4,5,and 9.

Discussion:

I believe that the recognition is off for a few different reasons. First is that the data set is large for this particular approach, and the algorithm is confusing the faces. The next is that the images that are used in the training section is front facing while the testing section has some where the people are from the person's side, which ends up not producing a euclidean distance that is anywhere near the same person with a different image. Finally, I think the weights need to be dealt with in a way that makes them more distinguishable. As the cmc plot suggests, this code is only 20% accurate with this data set. In this first matching, only person 5 and 9 are matched. However, on the second matching, person four is also matched to the correct person.