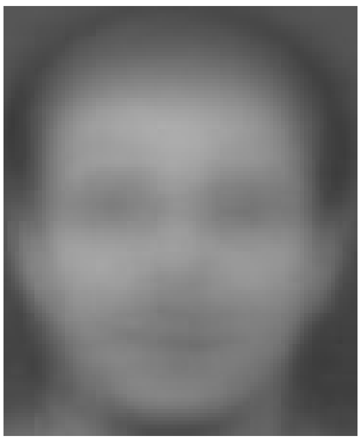
Abelardo Riojas
ISC 4223 Lab 1
Fall 2020 Dr. Meyer-Baese

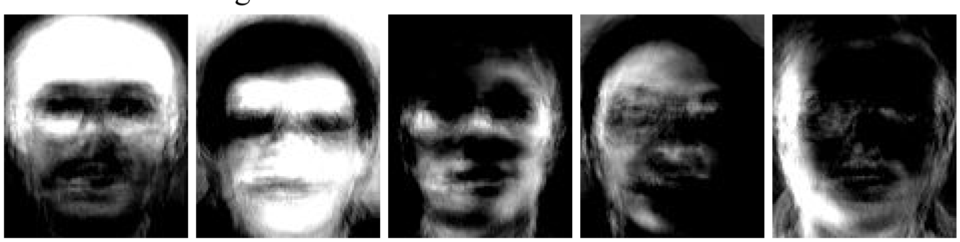
Lab 1 Report

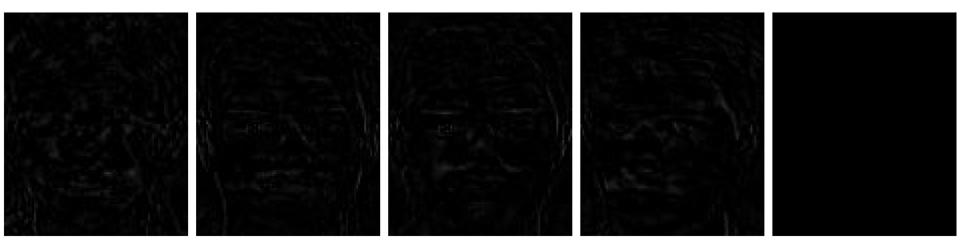
Section 1: Figures (first data set)

1.1: Standardized mean face

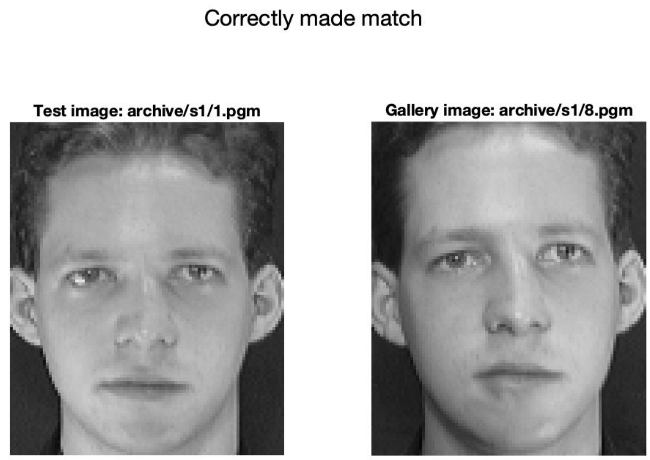


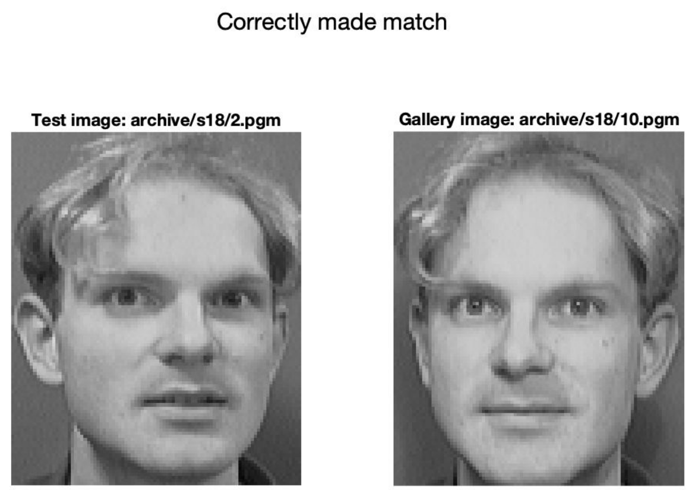
1.2: First five and last five eigenfaces

First five: 

Last five: 

1.3: Correct matches with original data set





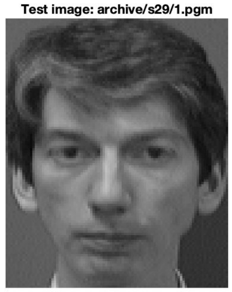
Correctly made match

Test image: archive/s39/2.pgm

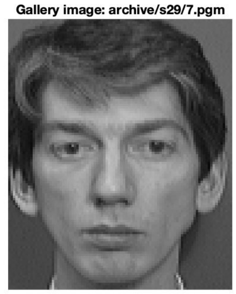
Gallery image: archive/s39/9.pgm

Correctly made match

Test image: archive/s29/1.pgm

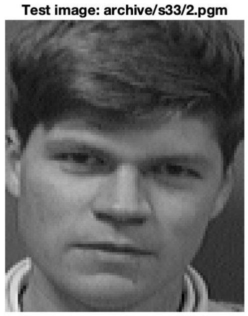


Gallery image: archive/s29/7.pgm

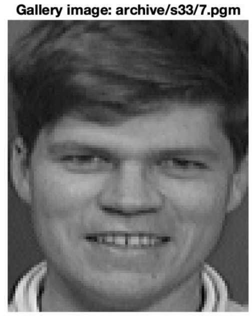


Correctly made match

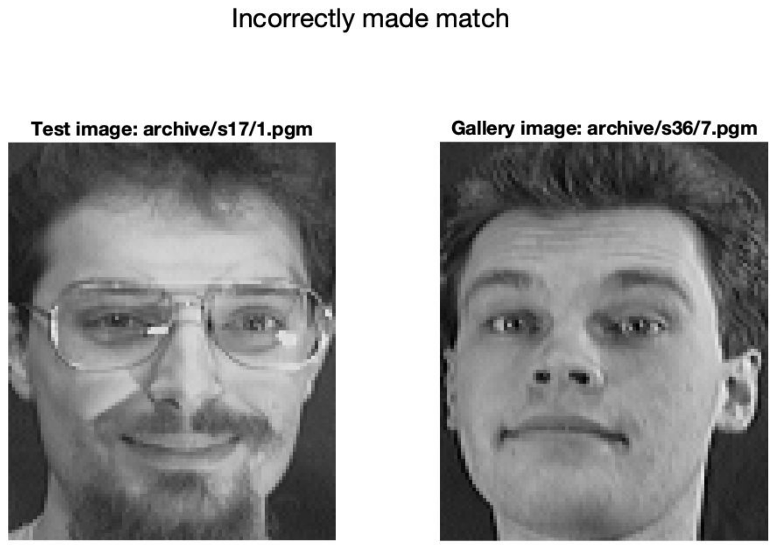
Test image: archive/s33/2.pgm



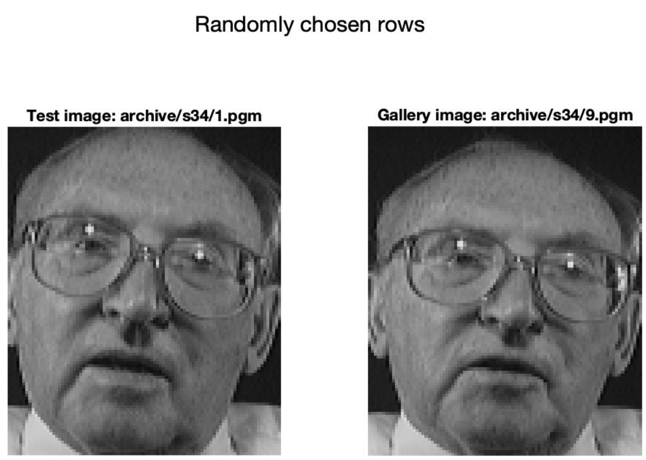
Gallery image: archive/s33/7.pgm

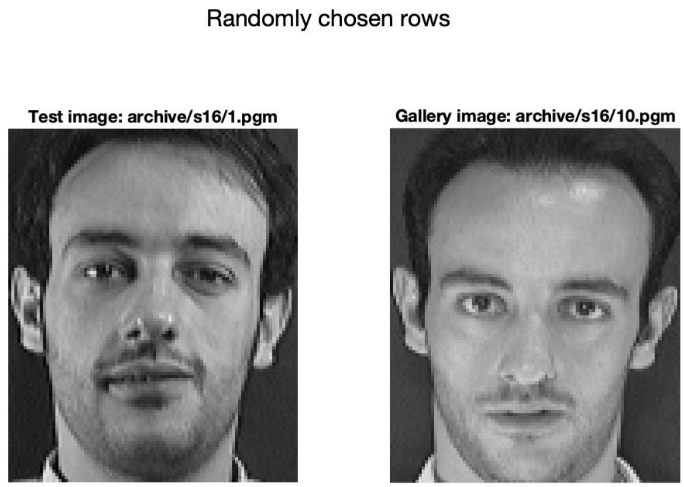


1.4: One (of one) incorrect matches with original data set:



1.5: Randomly chosen rows:

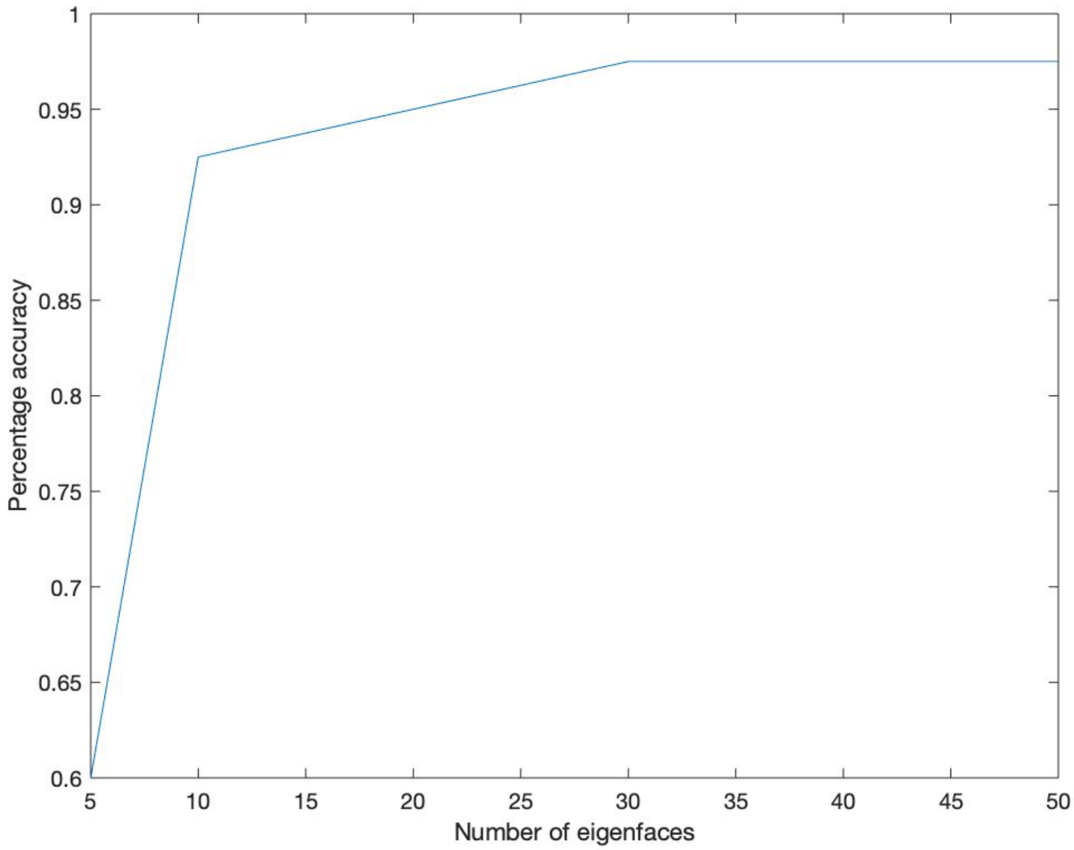




1.6: Table of percentage accuracy with number of eigenfaces in basis

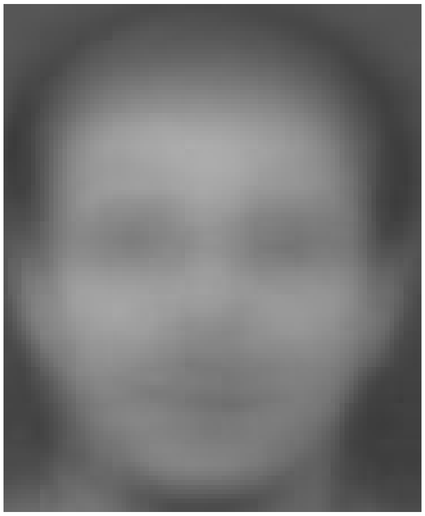
# Of Eigenfaces in Basis	*Percentage Accuracy (%)*
5	60%
10	92.5%
20	95%
30	97.5%
40	97.5%
50	97.5%

1.7: Plot of the face # vs percentage



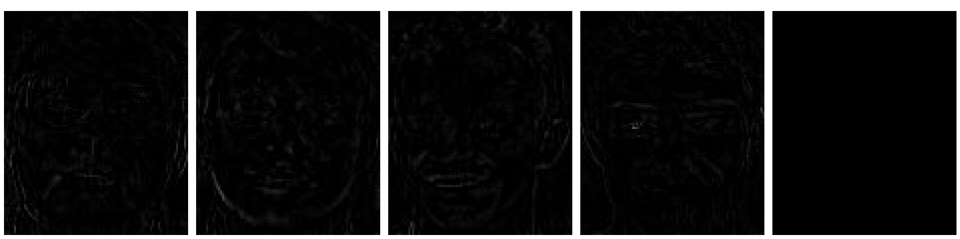
Section 2: Figures (second data set)

2.1: Standardized mean face



2.2: First five and last five eigenfaces

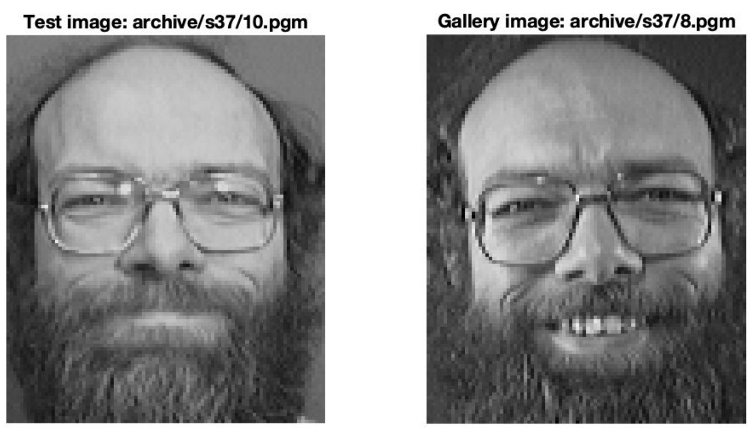
First five: 

Last five: 

2.3 Correct matches with second data set

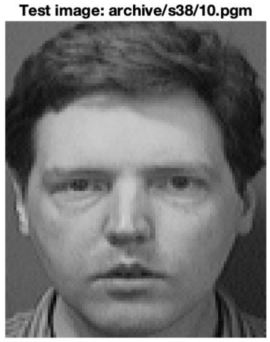
Correctly made match

Correctly made match

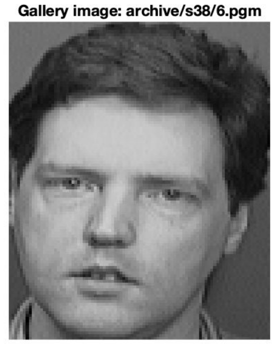


Correctly made match

Test image: archive/s38/10.pgm



Gallery image: archive/s38/6.pgm



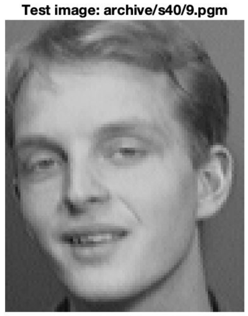
Correctly made match

Test image: archive/s39/9.pgm

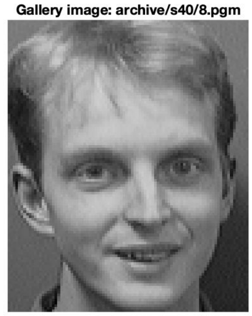
Gallery image: archive/s39/6.pgm

Correctly made match

Test image: archive/s40/9.pgm

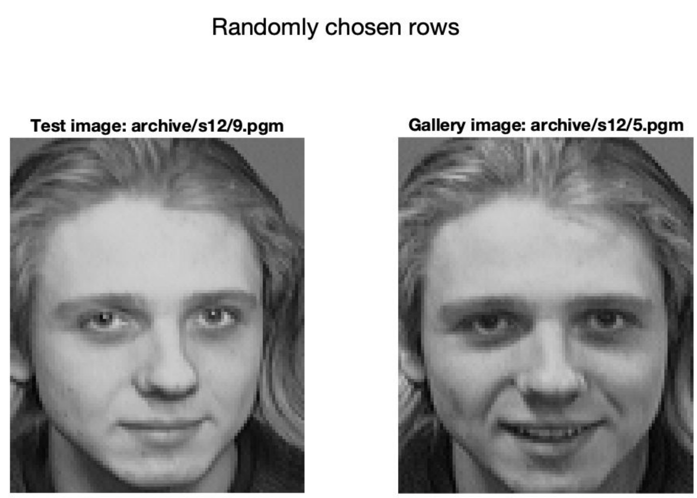


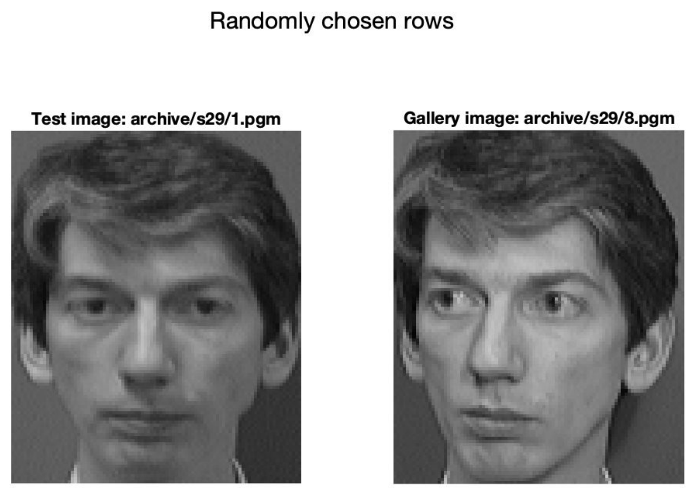
Gallery image: archive/s40/8.pgm



2.4: One (of one) incorrect matches with second data set

2.5: Randomly chosen rows

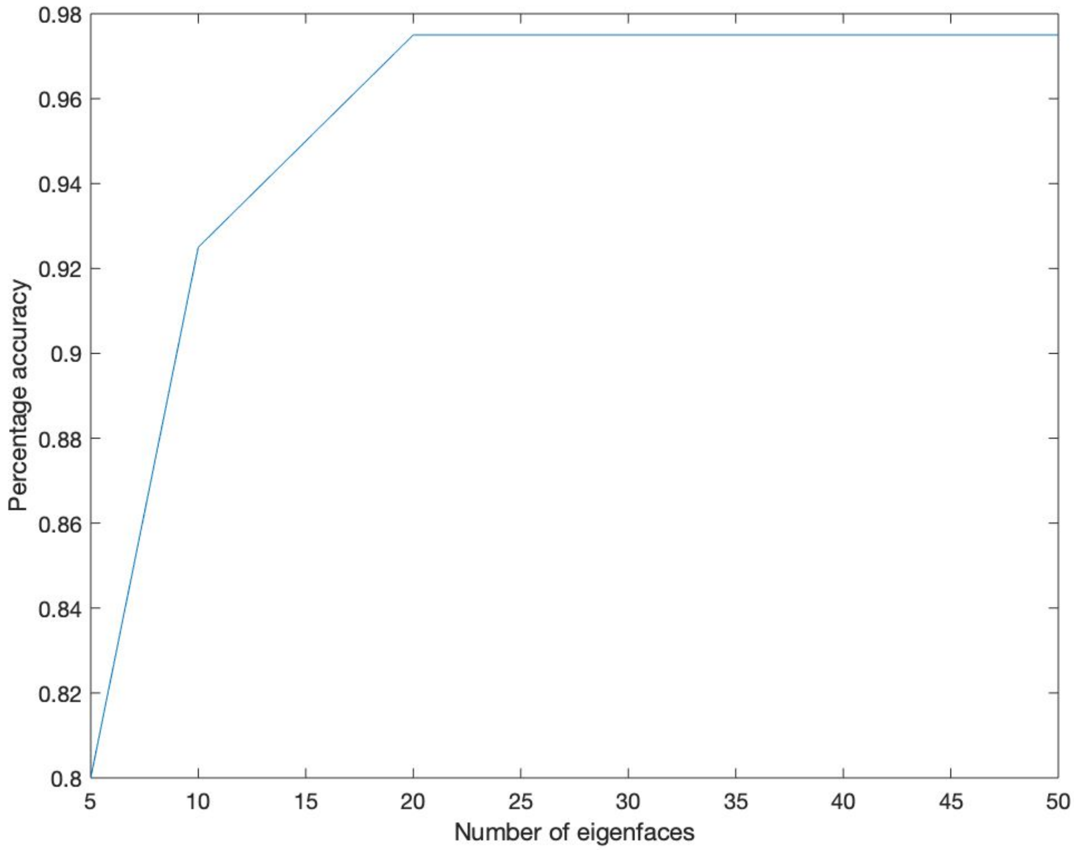




2.6: Table of percentage accuracy with number of eigenfaces in basis

# Of Eigenfaces in Basis	*Percentage Accuracy (%)*
5	80%
10	92.5%
20	97.5%
30	97.5%
40	97.5%
50	97.5%

2.7: Plot of face # vs percentage accuracy



Section 3: An explanation of the PCA method, an explanation of how my code implements it and my observations from Steps 14 and 15.

PCA Explanation:

To best explain PCA, it's best to look at it in steps. Step one is, as you probably guessed, get some data, arrange it into a matrix of vectors, each vector corresponding to some instance of what you're measuring. Step two is to subtract the mean of each vector from each vector. This standardizes your vectors (mean is now zero) and makes calculating the covariance matrix easy. Speaking of the covariance matrix! It's step 4: calculating the covariance matrix. My code calculates the covariance matrix in a different way than the normal method (as instructed) and computes a quasi-covariance matrix that is then manipulated to produce the same basis matrix as if you had done it the normal way. Step four is to get the eigenvectors and corresponding eigenvalues of the covariance matrix (those describe the strongest relationships between the data vectors). The fifth step is where the term "principal components" gets applied. This step requires you to choose the eigenvectors with the highest eigenvalues (however many you choose depends on the problem) to form the basis set of your projection matrix. Choosing the highest eigenvalues gives you a basis set which best describes the relationships of the data. Step six is the final step: deriving the new data set. Here you take the transpose of the eigenbasis for your projection and multiply it by your mean-adjusted vector to produce a projection of the data on the eigenbasis. Now the data is in terms of the most significant eigenvectors and describes how above or below each vector is from the overall trend.

Explanation of how my code applies PCA:

My code takes images of grayscale faces and makes each individual subject it's own vector by flattening their 112 by 92 pixel images into a 10304 by 1 vector. These 'subject vectors' are separated into 3 sets, training, gallery, and testing. The three sets are mean-adjusted, and then a quasi-covariance matrix is made by taking the transpose of the training data set and multiplying it by the data set itself. The eigenvectors of that matrix are then found, and are sorted by eigenvalue, highest to lowest. The training set is then projected onto the eigenvectors by multiplying the set by the matrix of sorted eigenvectors. A basis is made by taking the first fifty vectors (or principal components) of the new matrix. The testing and gallery sets are then projected onto the basis by taking each basis vector's transpose, multiplying it by each image in the testing and gallery sets, multiplying it by each basis vector, and adding them up. Weights are then taken for comparison, and distances are measured between each of the testing images and their projected gallery data. The image with the smallest distances to its testing image is interpreted to be the 'match' that the algorithm makes.

Observations from steps 14 and 15:

For the first data set, we see that the optimal number of principal components is somewhere between 20 and 30. The maximum percentage accuracy is 97.5% (I checked even with 160 PCs). When the dataset is changed in step 15, the accuracy of the 5 PC model goes up by 20% and the optimal number of PCs stays somewhere between 20 and 30.

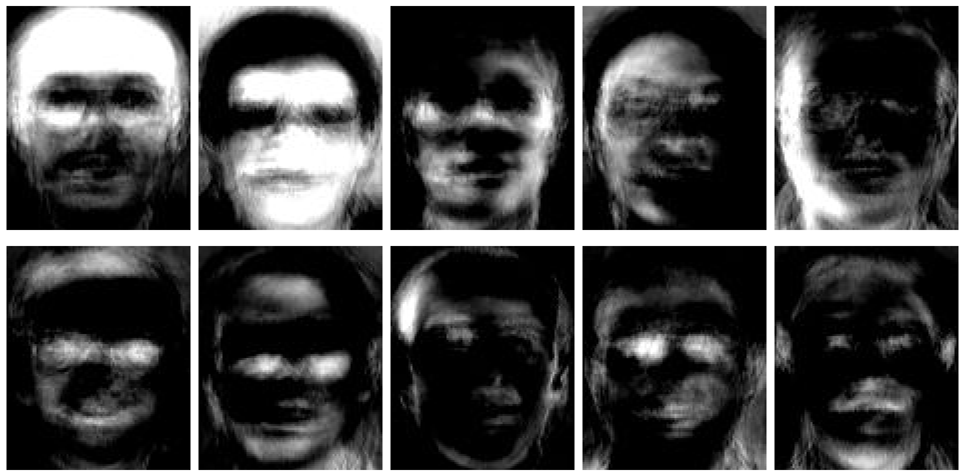
Section 4: Answers to the questions posed in Parts A, B and C in the Project Section of the original Lab Document.

A. Implement PCA-based face recognition. For each user, use images No. 3 through 6 to construct the training dataset, images No. 7 to 10 to construct the gallery set, and images No. 1 and 2 for testing. So, there would be 400 images for training and 100 images for testing.

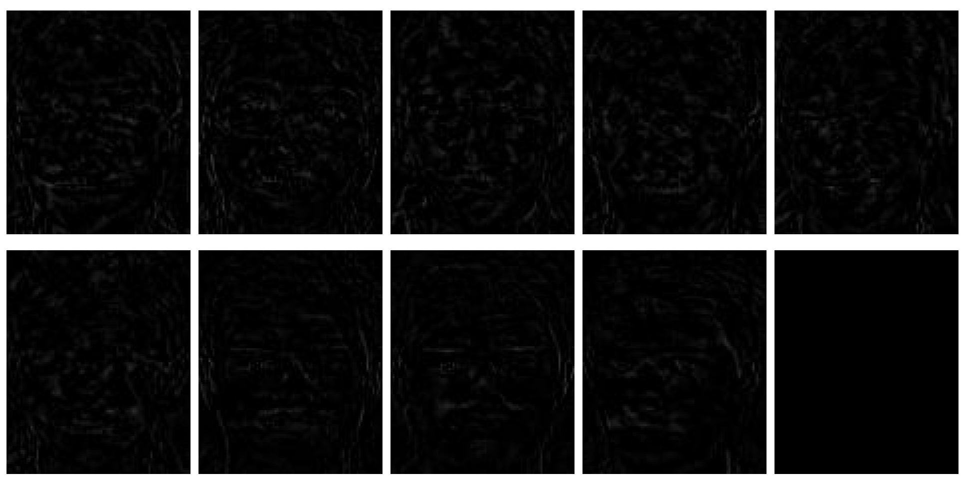
   Yep, that's kinda what the entire lab was.

   i. Compute the PCA space using the training dataset. Show the average face, the 10 eigenfaces corresponding to the 10 largest eigenvalues, and the 10 eigenfaces corresponding to the 10 smallest eigenvalues.

(See figure 1.1)
First 10:



Last 10:



ii. Choose the top 50 eigenvectors (eigenfaces) as the basis. Project both the gallery and test images onto the basis after subtracting the average to obtain the PCA representation of each image. Compute the Mahalanobis distance between the coefficient vectors for each pair of gallery and query images as the matching distance; you will obtain a matching distance matrix (400 training images, 100 query images). For each test image, there will be 400 matching distances obtained by matching the test with each image in the gallery dataset. Choose the smallest matching distance; if the associated subject is the same as that of the query image, then it is considered as a correct match, otherwise an incorrect match. For the test database, count the number of correct matches and divide it by the total number (100) to report the identification accuracy.

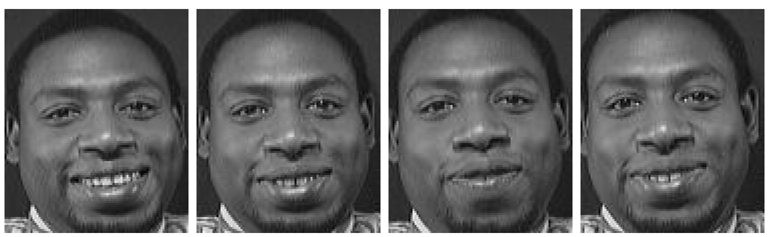
Yep, that's what I did in the code

iii. Select two subjects randomly. Show their gallery images and test images, separately.

Subject 4 (I didn't use matlab to pick my subject, I just asked my roommate)
Gallery:

Testing:

Subject 22 (Asked my other roommate for a number from one to forty)
Gallery:



Testing:

iv. Show 5 test images which are correctly matched, along with the corresponding best matched gallery images.

See figure 1.3.

v. Show 5 test images which are incorrectly matched, along with the corresponding mismatched gallery images. (If the number of incorrectly matched test images is less than 5, report all you get.)

See figure 1.4

B. Choose different numbers of eigenvectors as the basis, e.g., 5, 10, 20, 30, 40, 50. Conduct the experiment for each basis with different numbers of eigenvectors, and compute the identification accuracy using the same procedure as in a.II). Plot the curve of identification accuracy vs. number of eigenvectors.

   See figure 1.7.

C. For each user, use images No. 1 through 4 to construct the training dataset, images No. 5 to 8 to construct the gallery set, and images No. 9 and 10 for testing.

   See figures 2.3, 2.4, and 2.7.