CS 479/679 Pattern Recognition
Spring 2025 – Prof. Bebis
Programming Assignment 3 – Due on 4/30/2025 at 11:59 pm

In this project, you will implement the eigenface approach [2] and perform experiments to evaluate its performance and the effect of several factors on recognition performance.

1. Eigenface implementation

Read carefully and understand the steps of the eigenface approach. Use **jacobi.c** from "Numerical Recipes in C" to compute the eigenvalues/eigenvectors of the covariance matrix. **jacobi.c** is an efficient iterative numerical algorithm that works with symmetric matrices. The eigenvalues (and corresponding eigenvectors) are returned in descending order.

Warning: the [0] location of 1D arrays is NOT used in "Numerical Recipes"; so, if you need to use an array of size N, then you would need to use an array of size N+1 and simply ignore the [0] location of the array. The same holds true for 2D arrays etc.

Eigenvalue/Eigenvector verification test: verify that you are using **jacobi.c** correctly by computing the eigenvalues/eigenvectors of a matrix with known eigenvalues/eigenvectors (e.g., use the example posted on the course's webpage or any other example of your choice). When comparing your results, keep in mind that if **u** is an eigenvector, then **cu** is also an eigenvector where **c** is a constant. (e.g., c =- 1; in this case, the eigenvector direction can change but it is still a valid eigenvector). Also, make sure that the eigenvectors are unit length.

Your program should run in two modes: **training** and **testing**.

Training: In training mode, your program will read in the training face images and compute the average face I_{avg} and the eigenfaces (make sure that you use the A^TA "trick" as discussed in the lecture). It will then project each training face image i onto **all** M eigenfaces and compute the eigen-coefficients $\mathbf{\Omega}_i$, i=1,2,..,M, where M is the number of training face images. Finally, your program will store in a file the **all** M eigen-coefficient vectors $\mathbf{\Omega}_i$, I_{avg}, and **all** M eigenfaces. Typically, this step is performed once unless the training data set has changed.

Covariance matrix verification test: make sure that the covariance matrix C is symmetric. Also, ensure that its eigenvectors u are orthogonal and satisfy the property Cu=λu where λ are the corresponding eigenvalues.

Reconstruction verification test: one way to test that your program computes the projection coefficients **Ω** correctly is as follows: given an image I, (i) compute **Ω** by projecting (I - I_{avg}) onto the eigenfaces, (ii) reconstruct it using **all** M eigenfaces and add back the average face; let's call the reconstructed image $\hat{I}$, (iii) compute e_d=||I - $\hat{I}$ || (i.e., distance from face space (dffs) using Euclidean distance) and divide it by the total number of pixels in I to compute the **average** reconstruction error. If the **average** reconstruction error is NOT very small (i.e., typically less than 1 or 2), then your program has some errors which you would need to fix before performing any of the experiments described below.

Testing: In the testing mode, your program will read in all eigen-coefficient vectors $\mathbf{\Omega}_i$, i=1,2,..,M, I_{avg}, and **all** eigenfaces. Then, it will decide how many eigenfaces to keep for recognition purposes (i.e., determine the number of principal components K<M; this could be done in an interactive mode where the user specifies the amount of the information to be preserved as discussed in the lecture). Use the images in a test set (see below) to evaluate face recognition performance. Given a test image **I**, your program will need to project it onto the K principal components to compute the eigen-coefficients $\mathbf{\Omega}$. To recognize the face in the test image **I**, you will need to find the closest match $\mathbf{\Omega_p}$ to $\mathbf{\Omega}$ (i.e., distance in face space (difs)). Let's refer to the distance of $\mathbf{\Omega}$ from $\mathbf{\Omega_p}$ as $e_r = \min_i ||\Omega_i - \Omega||$, i=1,2,…,M; the distance should be computed using the Mahalanobis distance (see face recognition slides) as it seems to work better in practice than the Euclidean distance. Given the closet match $\mathbf{\Omega_p}$, you will recognize **I** as the person associated with the ID of $\mathbf{\Omega_p}$

Recognition verification test: one way to test that recognition works correctly is by simply using a training image for testing. In this case, you should be able to get an exact match with zero matching error. If NOT, your program is not working correctly.

2. Datasets

To test eigenface recognition, you will use images from the FERET face database [1]. FERET contains a large number of images acquired during different photo sessions and has a good variety of gender, ethnicity, and age groups. The lighting conditions, face orientation, and time of capture vary. In this project, you will concentrate on frontal face poses named **fa** (frontal image) or **fb** (alternative frontal image, taken during a different photo session). All faces have been normalized with regard to orientation, position, and size. Also, they have been masked to include only the face region (i.e., the upper body and background were cropped out).

The first subset **(fa)** contains 1204 images from 867 subjects while the second subset **(fb)** contains 1196 images from the 866 subjects (i.e., there is one subject in **fa** who is not in **fb**). You have been provided with two different sizes for each image: low resolution (16 x 20) and high resolution (48 x 60). All datasets can be downloaded from the course's webpage:

FA_L (fa, low resolution), FA_H (fa, high resolution)
FB_L (fb, low resolution), FB_H (fb, high resolution)

The file naming convention for the FERET database is as follows:

nnnnn_yymmdd_xx_q.pgm

where **nnnnn** is a five digit integer that uniquely identifies the subject, **yymmdd** indicates the year, month, and date when the photo was taken, xx is a lowercase character string (i.e., either fa or fb), and **q** is a flag (e.g., indicating whether the subject wears glasses - not always present).

3. Experiments

(a) Use **fa_H** for training (i.e., to compute the eigenfaces and build the gallery set) and **fb_H** for testing (query). So, there will be 1204 images for training and 1196 images for testing.

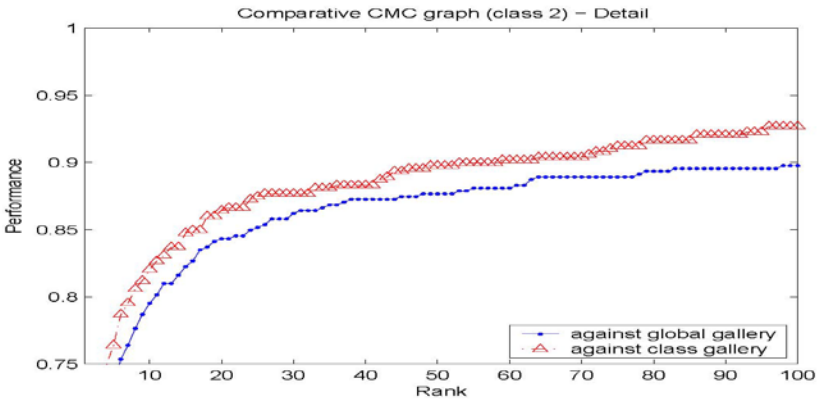
(a.I) Show (as an image) the following:

- The average face
- The eigenfaces corresponding to the 10 largest eigenvalues.
- The eigenfaces corresponding to the 10 smallest eigenvalues.

(a.II) In this experiment, consider the top eigenvectors (eigenfaces) preserving 80% of the information in the data. Project the query images onto this set of eigenvectors after subtracting the average face (from the training set). Then, compute the Mahalanobis distance between the eigen-coefficient vectors for each pair of training and query images as the matching distance.

Please note that for each query image, there will be 1204 matching distances (i.e., obtained by matching the query with each image in the gallery dataset). Choose the top **r** face gallery images (i.e., **r** is a parameter, see below) having the highest similarity score with the query face. (i.e., **r** smallest matching distances). If the query image is among the **r** most similar faces retrieved, then it is considered as a correct match, otherwise; it is considered as an incorrect match.

Count the number of correct matches and divide it by the total number of images in the test set (e.g., 1196) to report the identification accuracy. Draw the Cumulative Match Characteristic (CMC) curve [1] by varying **r** from 1 to 50. CMC shows the probability of the query being among the top **r** faces retrieved from the gallery. The faster the CMC curve approaches the value one, the better the matching algorithm is (see graph below).



(a.III) Assuming **r**=1, show 3 query images that are **correctly** matched, along with the corresponding best matched training samples.

(a.IV) Assuming **r**=1, show 3 query images that are **incorrectly** matched, along with the corresponding mismatched training samples.

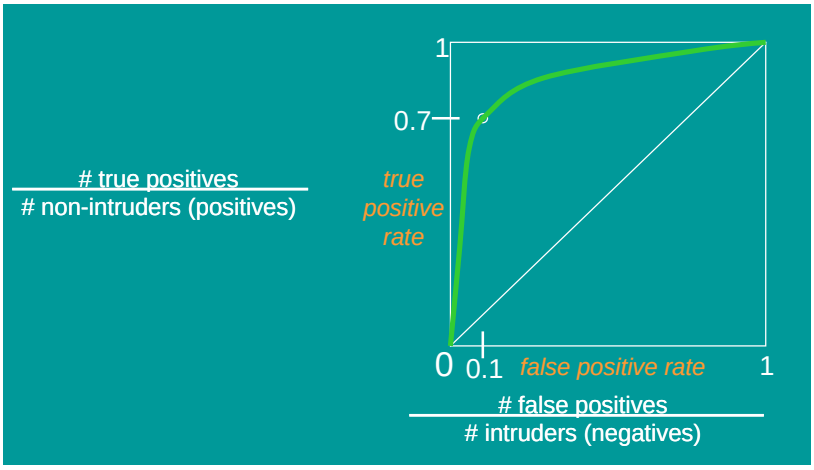
(a.V) Repeat (a.II – a.IV) by keeping the top eigenvectors corresponding to 90% and 95% of the information in the data. Plot the CMC curves on the same graph for comparison purposes. If there are significant differences in terms of identification accuracy in (a.II) and (a.V), try to explain why. If there are no significant differences, explain why too.

(b) In this experiment, you will test the performance of the eigenface approach on faces not in the gallery set (i.e., **intruders**). For this, remove **all** the images corresponding to the first 50 subjects in **fa_H** (please note that a given subject might have more than one image in **fa_H**); let's call the reduced set **fa2_H**. Perform recognition using **fa2_H** for training (gallery) and **fb_H** for testing (query). Since the training set has changed, you would need to compute a new eigenspace for this experiment (i.e., compute the new covariance matrix and its eigenvalues/eigenvectors). Use the eigenvectors corresponding to 95% of the information in the data (i.e., do not experiment with different percentages as in (a)).

Note: The last ID to be removed for the 50 subjects should be 140; the first ID after that should be 146. A total of 85 images should be removed.

To reject intruders, you would need to threshold e_r (i.e., accept a match only if $e_r < T_r$). Therefore, the choice of the threshold T_r would be very important. A high threshold value would increase False Positives (FP) while a low threshold value would decrease the number of True Positives (TP). Vary the value of T_r and compute (FP, TP) for each value. Then, plot the (FP, TP) values in a graph (i.e., ROC graph; see below).

Please note that the FPs should be computed using the images corresponding to the intruders (i.e., you know their IDs) while the TPs should be computed using the images corresponding to the non-intruders. The maximum possible T_r value can be found by computing the maximum distance between the test and training images (the threshold values have been normalized in [0, 1] in the graph below).



(Extra Credit 30%)

(c) Repeat experiment (a) using **fa_L** for training (gallery) and **fb_L** for testing.

(d) Remove all the images of the first 50 subjects from **fa_L**; let's call the reduced set as **fa2_L**. Repeat experiment (b) using **fa2_L** for training (gallery) and **fb_L** for testing.

(e) What is the effect of using low-resolution images? Are there any significant differences in identification performance? Explain.

References

[1] Phillips, J. and Moon, H. and Risvi, S. and Rauss, J., "The FERET Evaluation Methodology for Face Recognition Algorithms", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, no. 10, pp. 1090-1104, 2000.

[2] M. Turk and A. Pentland, Face Recognition Using Eigenfaces, *Computer Vision and Pattern Recognition Conference*, 1991.