CS 479/679 Pattern Recognition
Spring 2019 – Prof. Bebis
Programming Assignment 3 - Due: 4/23/2019

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" for computing the eigenvalues/eigenvectors of a symmetric matrix (Warning: the [0] location of an array is NOT used in "Numerical Recipes"; start storing your data at location [1]). 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 and eigenfaces. It will then project each training face image onto the eigenspace and compute its representation in that space (i.e., the coefficients of projection Ω_k, $k=1,2,..,M$, where M is the number of training face images). Finally, your program will store into a file the coefficients Ω_k, the average face, and the eigenfaces.

Testing: In testing mode, your program will read in the coefficients Ω_k, the average face, and the eigenfaces. Then, it will decide how many eigenfaces to keep (i.e., this could be done in an interactive mode where the user determines the percentage of the information to be preserved). Use the images in a test set (see below) to evaluate face recognition performance. Given a test image, your program will need to project it onto the eigenspace and compute its projection coefficients Ω. To recognize the face in the test image, you will need to find the closet match Ω_k to Ω (i.e., distance in face space (difs)). Let's call $e_k = ||\Omega_k - \Omega||$ where the distance is computed using the Mahalanobis distance.

Very important: to make sure that your program works correctly, try the following: given an image I, (i) project it onto the eigen-space, (ii) reconstruct it using **all** eigenfaces; let's call the reconstructed image $\hat{I}$, (iii) compute $||I - \hat{I}||$ (i.e., distance from face space (dffs) using Euclidean distance). The difference should be very small; if it is not, then your code is not working correctly. Do not proceed unless you have been able to verify this step.

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 as **fa** (frontal image) or **fb** (alternative frontal image, taken during a different photo session). All faces have been normalized with regards to orientation, position, and size. Also, they have been masked to include only the face region (i.e., 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. So, there will be 1203 images for training and 1196 images for testing (query).

(a.I) Show (as images) the following:
- o   The average face
- o   The eigenfaces corresponding to the 10 largest eigenvalues.
- o   The eigenfaces corresponding to the 10 smallest eigenvalues.

(a.II) Choose the top eigenvectors (eigenfaces) preserving 80% of the information in the data as the basis. Project both training and query images onto this basis after subtracting the average face to obtain the eigen-coefficients. 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 1203 matching distances (i.e., obtained by matching the query with each image in the gallery dataset).Choose the top **N** face gallery images (i.e., **N** is a parameter, see below) having the highest similarity score with the query face. (i.e., **N** smallest matching distances). If the query image is among the **N** 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 **N** from 1 to 50. CMC shows the probability of the query being among the top N faces retrieved from the gallery. The faster the CMC curve approaches the value one, the better the matching algorithm is (see graph below).

Comparative CMC graph (class 2) – Detail
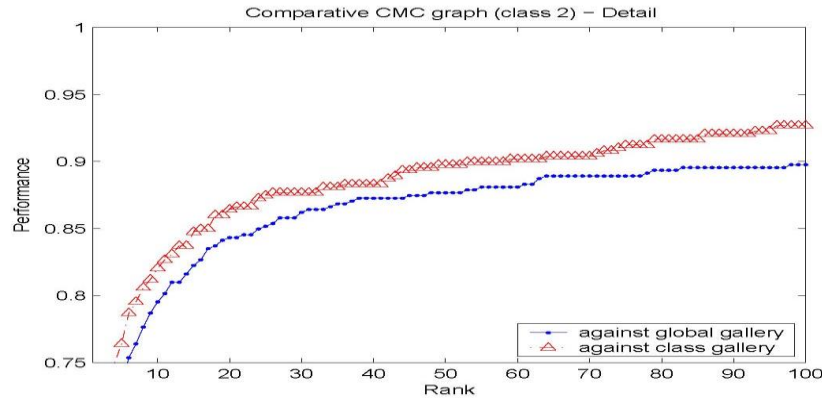
(Part a.III) Assuming N=1, show 3 query images which are **correctly** matched, along with the corresponding best matched training samples.

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

(Part 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 of the first 50 subjects from **fa_H**; let's call the reduced set as **fa2_H**. Perform recognition using **fa2_H** for training (gallery) and **fb_H** for testing (query). In this experiment, use the eigenvectors corresponding to 95% of the information in the data. To reject intruders, you would need to threshold e_k (i.e., accept the match only of $e_k < T$). In this case, the choice of the threshold T is very important. A high threshold value will increase False Positives (FP) while a low threshold value will decrease the number of True Positives (TP). To find out what is a good threshold value, you would need to vary the value of T and compute (FP, TP) for each value. Then, you would need to plot the (FP, TP) values in a graph (i.e., ROC graph; see below).
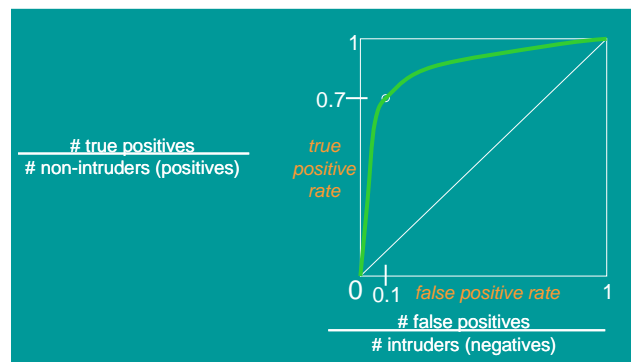
<u>Graduate Students Only – Experiments using low-resolution face images.</u>

(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.