CIS4930 – Programming Assignment #3; Due Date:04/19/2018

Submit your code via CANVAS. Also submit a TYPED REPORT that contains, concise discussion on your observations from the results and any reasoning necessary to explain your approach. The report should contain, results that are arranged to show the input image and the output images, if and when applicable. Try to minimize your page usage by displaying at least 3 images per page. All your image displays must have a caption.

1. This programming assignment involves the implementation of a simple face recognition algorithm using the well known principal component analysis technique covered in class. You are required to develop code to do the following tasks:

(a) **Building the data matrix:** Vectorize the set of images from the training set (= k), denoted by \( x_i \), and stack them up as columns, \( A_i \), to form a matrix \( A(n,k) \). Each column vector matrix \( A \) is of size equal to the number of pixels in the image (= n).

(b) **Computing the mean image and centralizing the data:** Compute the mean (arithmetic mean) image \( \mu \) and centralize all the images \( A_i \) with respect to this mean \( \mu \), i.e., perform the operation \((A_i - \mu)\) for \( i = 1, ..., k \). Note that all images must be in the same coordinate system prior to computing their mean. If this is not the case, the computed mean will be incorrect. Note also that the mean image, \( \mu \), is a vector.

(c) **Eigen vector computation:** Compute the eigen vectors, \( e_i \), of the covariance matrix \( C = AA^t \). There are at most \( k \) nonzero eigen values of \( C \) and hence at most \( k \) non-zero eigen-vectors. You may use the MATLAB built in function call to compute these eigen vectors.

(d) **Threshold selection and reconstruction:** Now, you are going to select \( m \) eigen-vectors from the above computed set of eigen vectors. Find \( m \) by choosing the set of eigen values that explain most of the variance in the data. For this, you should plot the cumulative sum of eigenvalues (assuming they are in descending order). Divide each value by the total sum of eigenvalues prior to plotting, then your plot will show the fraction of total variance retained vs. number of eigenvalues. This plot will provide you with a good indication of when you will reach the point of diminishing returns i.e., little variance is gained by retaining additional eigenvalues. You will select two additional threshold values around the cutoff point and reconstruct randomly selected training images using all three threshold values. This will allow you to see the difference between the selected thresholds in terms of accuracy. Note that to reconstruct a training image, you will use the following equation:

\[
\hat{x}_i = \mu + \sum_{j=1}^{m} r_{ij} e_j^t.
\]

Here, \( \hat{x}_i \) denotes the reconstructed training image, \( r_{ij} \) denotes the coefficient obtained by projecting \( x_i \) on \( e_j \) described below.

(e) **Coding training data:** Now, project each of the input training images onto the chosen eigen vectors \( e_i \), \( i = 1, ..., m \). Projection is achieved by simply performing \( A_i^t e_j \), for \( i = 1, ..., k \) and \( j = 1, ..., m \). This will give you \( m \) coefficients for each image \( x_i \). Store these coefficients in a vector \( r_i \). This vector is your eigen-code for each image in the training data set.

(f) **Coding the test data probe:** For each test image, \( T_i \), vectorize the image and project the image on to the learned eigen basis \( e_i \), \( i = 1, ..., m \). This will give you \( m \) coefficients representing the test image in the learned eigen basis. Store these coefficients in a vector \( V \).

(g) **Nearest Neighbor Classification:** Now compare \( V \) with each of the code vectors of the training images \( r_i \). Use the standard Euclidean distance between the vectors to achieve this. Assign the test data with the identity of the image from the training data whose code the
test data is closest to. This is equivalent to a nearest-neighbor classifier on the PCA-based representation of the data.

2. **optional:** You may use any other sophisticated classifier on the PCA-based coded image representation. Try using existing MATLAB function call to an SVM-based classifier on the PCA-based representation. You may use a radial basis kernel or some other kernel of your choice.

**What to turn in:** (i) MATLAB programs EigenRecon.m and EigenNeighbor.m along with scripts to run the code on the test data. (ii) A report that shows example outputs from your running the code. This must contain: (a) several example sample test images and the corresponding closest match found from the training set. (b) The threshold selection plot (see above description for threshold selection). (c) A display of your randomly selected image from the training set, it’s reconstruction using the $m$ chosen eigen basis and the difference image showing the difference between these two images. Display all the three images side by side in your report. Repeat this reconstruction process 2 more times for different cutoff values of $m$ and show the reconstruction and difference from the same original image. Comment briefly on the reconstruction accuracy.

**Data sets:** You should use face images from the following database: [http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html](http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html). There are 10 different images for each of the 40 distinct subjects. All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

The files are in PGM format, and can conveniently be viewed on UNIX (TM) systems using the 'xv' program. The size of each image is 92x112 pixels, with 256 grey levels per pixel. The images are organised in 40 directories (one for each subject), which have names of the form sX, where X indicates the subject number (between 1 and 40). In each of these directories, there are ten different images of that subject, which have names of the form Y.pgm, where Y is the image number for that subject (between 1 and 10).


For this assignment, you will randomly choose just 20 subjects from the database of 40 subjects for the training. Since each subject has 10 images taken under different conditions, you will again randomly choose 5 of these 10 images for your training samples. Thus, you will have a total of 100 images in your training 'bag'. For the test set, simply choose the remaining 5 images of each of the 20 chosen subjects. This will give you 100 images for your test 'bag'. Your program should be able to "correctly" label/identify any randomly chosen probe from the test 'bag'.

Further, if you want to reduce the size of the images to reduce processing time, crop the faces from all the training samples you have chosen and obviously, your test set has to be cropped to the same size as well.