ECE3093 Semester 1, 2013
Assignment 1, Part A (worth 5%)
Due: Monday 8th April by 10am
Assignment box, ground floor, building 28
(Clayton student only; Malaysian students please check with Dr. Kamrani)

Task 1: Face Recognition by Eigenfaces (2.5 marks)

Download the zip file eigenfaces.zip which contains an m file (called eigenfaces.m) and two directories (called Train and Test) containing images that can be used for training and testing a face recognition system.

Data: There are in total 40 individuals (named by '01', '02', … '40') involved in the data set. Each of them has two face images, one in the "Train" directory, and the other in the "Test" directory. The face images are already normalized and ready to be directly used in the algorithm.

The file eigenfaces.m does the following:

1. reads the training images into a matrix (images as rows) ;
2. performs Principal Component Analysis (PCA) on the training points (using the princomp function), to get the principal components (called "eigenfaces" if you transform the vector back to images because they look like faces);
3. *defines the number of principal components (k) to be considered;*
4. projects each image in the training set onto the principal components, and stores the projection vectors, one for each image, as the feature vector for that image. Until now, you have a gallery of "known" persons. Each person has a unique feature vector.
5. Displays the mean face and the eigenfaces using the montage function;
6. Evaluates a set of test images by first converting images into a matrix;
7. Projects the test images (rows of the matrix) onto the k principal components to get the feature vector for each test image (after subtracting off the mean);
8. Calculates the distance between the feature vector of each test image and the feature vectors of the "known" people
9. Sorts the distances in ascending order, and selects the top three distances as the three highest ranked candidates for the face recognition task (i.e. the top three people most likely to be the same as the person in a test image)
10. Calculates the rate of accuracy (where recognition is deemed accurate if the test image corresponds to one of the top three "known" people).

Your task: Modify the eigenfaces.m code to work out the best number of eigenfaces (principal components, k) to balance the trade-off between accuracy and dimensionality. How does the recognition accuracy on the test set vary with the number of principal components selected? What is the best number of principal components to retain to maximize the accuracy on the test set? (Hint: calculate the cumsum and plot to select eigenvalues, and add a loop and plot the test set accuracy for varying k)

Reference

M. A. Turk and A. Pentland, "Eigenface for recognition," Journal of Cognitive Neuroscience, vol. 3, pp. 71–86, 1991.

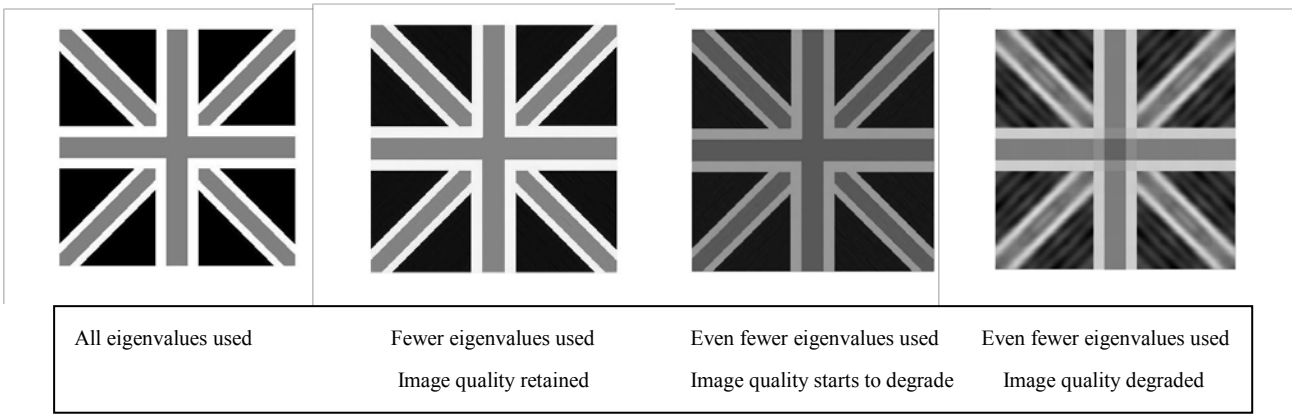
Task 2: Image Compression (2.5 marks)

Your task: Download "unionjack.png" and write a MATLAB m-file to do image compression by diagonalising this *real symmetric square matrix*.

Determine the smallest number of eigenvalues needed to reconstruct the image (approximate the matrix) before the image quality degrades perceptibly (to your eye).

Submit one page showing the original image i), and a sequence of 3 compressed images approximations showing ii) no visible degradation, iii) the start of some degradation, iv) poor degradation.
Include a plot of the eigenvalues to determine the point at which the approximation is likely to be perceptibly degraded, and make sure you select this as one of the images you reconstruct to confirm degradation.

The example below illustrates what is expected to be presented, but you need to indicate under each image the number of eigenvalues/eigenvectors used, show the plot of the eigenvalues (or cumsum), and submit your m-file code.



All eigenvalues used	Fewer eigenvalues used Image quality retained	Even fewer eigenvalues used Image quality starts to degrade	Even fewer eigenvalues used Image quality degraded

Hint: You will probably want to use MATLAB functions like *imread* (to read in the image), *double* and *uint16* (to convert between the uint16 image matrix and double precision), *eig* (to find the eigenvalues and eigenvectors of the image matrix), *sort* (to make sure the eigenvalues are sorted by magnitude), and *imshow* to reveal the reconstructed images.

END OF ASSIGNMENT 1, PART A