Introduction to Computer Vision (ECSE 415)
Assignment 4: Face Detection and Recognition

Due date: 11:59PM, April 14, 2020

This assignment permit you to practice using eigenvectors to perform face detection and identification. You will obtain a set of images containing faces, use principal components analysis (PCA) to represent them, then use that information to automatically recognize those faces in unseen data. You will then compare your implementation's performance to the Viola-Jones face detector (use publicly-available code; cite appropriately).

Please submit your assignment solutions electronically via the myCourses assignment dropbox. The submission should include: a single jupyter notebook. More details on the format of the submission can be found below. Submissions that do not follow the format will be penalized **10%**. Attempt all parts of this assignment. The assignment will be graded out of total of **75 points**. Note that you can use any of the OpenCV and scikit-learn functions shown during tutorial sessions for this assignment, unless stated otherwise.

Students are expected to write their own code. (Academic integrity guidelines can be found here).

Note: Assignments received up to 24 hours late will be penalized by 30%. Assignments received more than 24 hours late will not be graded.

Instruction for the Report

1. Submit one single jupyter notebook for the whole assignment. It should contain everything (code, output, and answers to reasoning questions) [1]
2. Submit your dataset along with your code. Explicitly mention how your code is accessing the dataset in the comments.
3. Write answers section-wise. The numbering of the answers should match the questions exactly.
4. Comment your code appropriately.

[1] If you are facing some memory and RAM issues, you are free to submit multiple notebooks as long as you clearly mention the reason for doing so.

5. The answers to the reasoning questions should be brief but comprehensive. Unnecessarily lengthy answers will be penalized. Use *markdown cell* type in jupyter notebook to write the answers.

6. If external libraries were used in your code please specify its name and version in the README file.

7. Run the whole jupyter notebook one last time using *Cell-> Run All*. This will run all cells in the jupyter notebook and show all outputs. We expect that when we run your jupyter notebook, we should be able to reproduce your results.

1 Data Acquisition

Test image: You will need to acquire or find a group picture containing at least three people, with their faces clearly visible. This image will be used to evaluate the performance of your implementation.
Training images: You will need to acquire or find additional pictures for each individual present in the group picture. The number of pictures required to perform the task varies according to a number of factors. Ten pictures for every person should suffice. These images will be used to build the face space. The images should be cropped to be contain only the person's face, and the cropped images should all be of the same size. The dimension of the images, and the decision as to whether and how to make use of color, are left as design decisions. It would be a good idea to consider smaller greyscale images if you run into computational issues, or in order to reduce the dimensionality of the data. Describe how you selected the images with respect to the conditions of the detection/recognition methods. **(3 points)** Show at least 5 example images for each individual **(5 points)**. You can refer slides of Lecture-17 on Face Recognition for help regarding image selection.

You can download publicly-available images (e.g., celebrities) or acquire your own, but the images must be submitted in .png or .jpeg format. Proper referencing is required.

2 Eigenface Representation

Produce and eigenface representation for your training data through PCA. Please note that you are **not** allowed to use the in-built PCA function in OpenCV/Scikit-Learn. You should implement Snapshot method for PCA (covered in class - Lecture 17) from scratch using numpy. **(15 points)**

Plot the variance against the eigenvector number in descending order (i.e. total variance vs. number of principal components) **(2 points)**. Also plot the total variance against the eigenvector number used for computation. (Refer Slide-57 of Lecture-17) **(2 points)**. Do you need all the vectors to represent the data? Discuss **(3 points)**. Display the first 5 eigenfaces **(5 points)**.

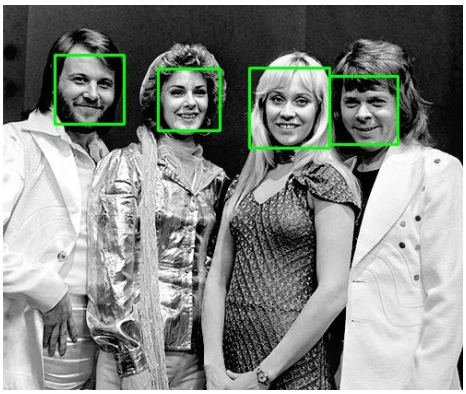

Figure 1: Face Detection: Example Image with bounding boxes around the detected faces.

3 Neighbourhood Clustering:

For every training image, find the nearest neighbour (L2 distance), and check whether both images belong to the same person. What fraction of your data has a neighbour that is of the same person? Do this both in the original high dimensional pixel space and then in the eigenspace, and compare the accuracy values. Would you expect there to be a significant difference? **(10 points)**.

4 Face Detection & Recognition

You will now detect all the faces in the group image. Use a sliding window to detect the faces. Set a threshold on the distance in eigenspace between the window contents and your training data. Try different values of thresholds and use the one which gives you good results. Display your image with bounding boxes around detected faces for your best threshold. (ex. Figure 1) [2] **(15 points)**.

You will now try to recognize each of the people in the group image. For the windows in which a face is correctly detected, project the contents of the window to your eigenspace and find the nearest neighbour from your data. The identity of the detected face will be based on the identity of the closest person in the training set. How well does the method work? How many false positive face detections do you get? For mis-identified faces, report the number of neighbours you need before getting the correct label (correct identification would use 1). In which situations would expect the approach to fail, and what could you do to improve it? **(15 points)**.

[2] You can use any online available code for bounding box genertation. Please cite the source for this in your report.