Instructions:

- Please upload all the code and the report on GitHub/GitLab.
- Clearly explain your findings and methodology.
- Do not put the images in the GitHub/GitLab, Any and all results should be included directly into the report, along with explanations and supporting code snippets.

Part1:
Face Recognition using K Nearest Neighbours and PCA

The CMU PIE Dataset

The CMU Pose, Illumination, and Expression (PIE) database (http://ieeexplore.ieee.org/abstract/document/1004130/) consists of 41,368 images of 68 subjects. Each person is under 13 different poses, 43 different illumination conditions, and with 4 different expressions.

For this project, we use a simplified version of the database which only contains 10 subjects spanning five near-frontal poses, and there are 170 images for each individual. In addition, all the images have been resized to 32x32 pixels. The dataset is provided in the form of a csv file with 1700 rows and 1024 columns. Each row is an instance and each column a feature. The first 170 instances belong to the first subject, the next 170 to the second subject and so on. Following illustrate various instances of the first subject.

Requirements

1. Pre-process the dataset by normalizing each face image vector to unit length (i.e., dividing each vector by its magnitude). Next, for each of the 10 subjects, randomly select 150 images for training and use the remaining 20 for testing. Create such random splits a total of 5 times. You must carry out each of the following experiments 5 times and report average accuracy and standard deviation over the 5 random splits, as well as computation times.

2. Implement a k Nearest Neighbours (k-NN) classifier using the training set and evaluate its performance on the test set. You may not use built-in / library functions to implement the classifier.
    - You should experiment with different distance measures (e.g., Euclidean, Mahalanobis, cosine similarity), and different values of k.
    - You should also present results where you show the effect of involving more classes vs. fewer classes in your experiment(s).
    - You should also present results when fewer training images are used (for instead 100 training images and 70 test images per category).

3. Now make use of the dimensionality reduction technique PCA in conjunction with the k-NN implemented above. Note that PCA is an unsupervised technique, hence should be trained based on all training images. Does this techniques help yield uncorrelated covariance matrices

(you can visualize the covariance matrices before and after dimensionality reduction as images to see this)? Does it improve computation times and classification performance? Try varying the number of principal components used and identify the best number.

Part 2:
Interactive Foreground Segmentation

In this part of the task, you will implement a basic version of the interactive image cut-out / segmentation approach called Lazy Snapping [1]. You are given several test images, along with corresponding auxiliary images depicting the foreground and background "seed" pixels marked with red and blue brush-strokes respectively. Your program should exploit these partial human annotations in order to compute a precise figure-ground segmentation.

You should first write a function that performs *k*-means clustering on color pixels. The input arguments are the desired number of clusters *k* and the data points to cluster. It outputs a cluster index for each input data point, as well as the *k* cluster centroids. You should then extract the seed pixels for each class (i.e., foreground and background) and use your *k*-means function to obtain *N* clusters for each class. A good choice for *N* is 64, but you can experiment with smaller or bigger values.

Next, compute the likelihood of a given pixel p to belong to each of the *N* clusters of a given class (either foreground or background) using an exponential function of the negative of the Euclidean distance between the respective cluster center C_k and the given pixel I_p in the RGB color space. The overall likelihood $p(p)$ of the pixel to belong to this class is a weighted sum of all these cluster likelihoods, where the weight of each cluster w_k is the proportion of the seed pixels in that cluster among the total seed pixels for that class (note this is similar to a multivariate Gaussian Mixture Model with the covariance matrix assumed to be the identity matrix).

$$p(p) = \sum_k w_k e^{(-\|I_p - C_k\|^2)}$$

You might see improved results if you ditch the square operation in equation above. The reason may have to do with the fact that squaring essentially reduces small values even further, mitigating (adversely) the effects of the negative exponential. Finally, a given pixel is simply assigned to the class to which it is more likely to belong. That is, if $p_{fg}(p) > p_{bg}(p)$, pixel p is assigned to the foreground class, i.e., $x_p = 1$, and vice versa.

Include your results for all test images in your report, and explain what you get. For test images with two stroke images, you should report results for both cases. Also compare results for different values of *N*, i.e., the number of clusters evolved in the foreground and background classes.

Compare the results obtained by your implementation of the K-Means algorithm vs. built-in / library functions.

Reference

[1] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. In ACM SIGGRAPH 2004 Papers, pages 303–308, 2004.