Introduction

In this assignment, we will use the eigenfaces technique discussed in class to write a Matlab program for face recognition and also for face detection.

Please download the homework5.zip file from the course piazza page as it contains images and code needed for the assignment.

Requirements

You should perform this assignment in Matlab and is due on December 15th, 2014 by 5pm. Copying others' code and changing all the variable names is not permitted! You are not allowed to use solutions from similar assignments in courses from other institutions, or those found elsewhere on the web. If you access such solutions YOU MUST refer to them in your submission write-up but you may still be penalized.

Your solutions should be uploaded to the CSE server using the CSE_SUBMIT command. Your submitted zipped file for this assignment should be named **PersonNumber_hw5.zip**. Failure to follow this naming convention will result in your file not being being picked up by the grading script. Your zipped file should contain: (i) a PDF file named LastnameFirstname_hw5.pdf with your report, showing output plots and images for different parts of the assignment and explanatory text, where appropriate; (ii) the source code used to generate the images (with code comments), along with a running script (named runfile_hw5.m). We should be able to cut and paste from your runfile to execute the code for each part of the assignment in turn.

Problem 1. Offline Processing: Computing the Eigenfaces[1]

Using the starter code and data provided in the assignment folder, start by writing a Matlab routine:

```
function [mean_face, eigenfaces, singularvalues, celebrity_PCA_coefs] =
ComputeEigenfaces(celebrity_directory, binary_mask)
```

where `celebrity_directory` is a directory containing a small database of celebrity photographs. Each of these images is a $330 \times 280 \times 3$ color image, and the scale, rotation, and offset of each image has been normalized so that the eyes of each celebrity are in the same location.

Use `dir([celebrity_directory, '*.jpg'])` to get a list of files in the celebrity directory.

`binary_mask` is the boolean $330 \times 280 \times 3$ array that is found the assignment folder. This boolean mask zeros-out parts of the images that typically lie outside of faces, so that the

[1] We adapted this assignment from `http://www.ittc.ku.edu/~potetz/EECS_741_Fall11/Homeworks/hw3/Hmwk03.pdf`

image background does not affect the results.

`ComputeEigenfaces` should first compute the mean celebrity face, and then "centers" the celebrity images by subtracting out this mean face. `mean_face` should be a $330 \times 280 \times 3$ array. `ComputeEigenfaces` should apply the binary mask to each image, to zero-out the background.

Next, compute the `eigenfaces`. To do this, you will first want to construct an array that contains each celebrity image. It is convenient here to keep only those pixels that are not masked out. You can do this by:

```
unmasked_pixels = find(binary_mask);
im_vector = im(unmasked_pixels);
```

Later, when you need obtain the masked image from `im_vector`, use:
Once you have an array that contains each image vector, compute the eigenfaces using the Matlab function `svd`. Note that you have far more unmasked pixels (51951) than celebrity images (163), so most of the 51951 eigenfaces will be degenerate. There's no need to compute them all. Look over the Matlab help page for the `svd` command

Use `[U, S, V] = svd(X, 0);` to compute the eigenvectors and singular values.

To identify similar faces, there is no need to use all 163 nondegenerate eigenfaces. In this assignment, use only 10. Select the 10 eigenfaces with the highest singular values, and save them in the matrix `eigenfaces`. `eigenfaces` should be a $330 \times 280 \times 3 \times 10$ matrix. Also save the singular vectors in the 10×1 array `singularvalues`.

Look up the Matlab function `diag` for getting the diagonal of a matrix.

Finally, you will need to have the PCA coefficients for each celebrity. In the 163×10 matrix `celebrity_PCA_coefs`, store these coefficients so that

```
offcenter = reshape(eigenfaces, [330*280*3,10]) * celebrity_PCA_coefs(i,:)';
recon = mean_face + reshape(offcenter, [330, 280, 3]);
```

will approximate the original celebrity face. Don't expect good reconstructions here. Remember it is characterizing each face by only 10 numbers (you may want to vary this slightly to improve reconstructions, but not significantly as this will become more computationally intensive. `ComputeEigenfaces` should take just a few seconds to run.

Problem 2. Online Processing: Finding the Nearest Match

Now write the function:

```
function closest_celebrities = ComputeLookAlike(face_im, celebrity_directory,...
binary_mask, mean_face, eigenfaces, singularvalues, celebrity_PCA_coefs)
```

that accepts a face image `face_im` along with the previously computed variables. Expect `face_im` to be a $330 \times 280 \times 3$ array of doubles with values between 0 and 1. The other input variables are as described above.

The function should find the two celebrity faces that most closely match the input image (as measured by the Euclidean distance).

`ComputeLookAlike` should then do two things: First, your function should open a figure window, and using the Matlab routine subplot, display from left to right the input image, the nearest celebrity match, and the second nearest celebrity match. Second, `ComputeLookAlike` should return a 1×2 vector containing the indices (into the array `celebrity_PCA_coefs`) of the two nearest celebrity matches (in order). Also indicate the value of the measured distance.

Test your function on the photographs in the `TestImages` folder. If you would like to test your program on other photographs of people, you can use the script `FaceAligner` in the folder to register any photograph of a face so that the eyes line up with the celebrity photographs. When you run `FaceAligner`, just click on the left eye, then the right eye in the image. Then pass it through `ComputeLookAlike` to get the nearest match.

Problem 3. Report

You report should: (i) provide a description of your overall approach; (ii) discuss how well this approach works for face recognition (iii) describe how you can extend it to face detection from the results of your test images; and (iv) show each of the test images along with the 2 nearest matches and the value of the distances from the nearest match.

Extra credits will be given for any novelty in approaches to processing the data.