Introduction to Computer Vision (ECSE 415)
Assignment 2: Image Matching and Face Detection

DEADLINE: March 6, 11:59 PM

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 a total of **100 points**. There are *50 points* for accurate analysis and description, *40 points* for bug-free and clean code, and *10 points* concerning the appropriate structure in writing your report with citations and references. Each assignment will be graded according to defined rubrics that will be visible to students. Check out MyCourses, the "Rubrics" option on the navigation bar. You can use **OpenCV**, **Scikit-Image**, **Numpy**, **matplotlib**, **sklearn** library functions for all parts of the assignment except those stated otherwise. Students are expected to write their own code. (Academic integrity guidelines can be found **here**). Assignments received up to 24 hours late will be penalized by 30%. Assignments received more than 24 hours late will not be graded.

Submission Instructions

1. Submit a single Jupyter notebook consisting of the solution of the entire assignment.
2. Comment your code appropriately.
3. Give references for all codes which are not written by you. (Ex. the code is taken from an online source or from tutorials)
4. Do not forget to run **Markdown** ('Text') cells.
5. Do not submit input/output images. Output images should be displayed in the Jupyter notebook itself.
6. Make sure that the submitted code is running without error. Add a **README** file if required.
7. If external libraries were used in your code please specify their name and version in the **README** file.
8. We are expecting you to make a path variable at the beginning of your codebase. This should point to your working local (or google drive) folder.
   Ex. If you are reading an image in the following format:

   ```
   img = cv2.imread ( '/content/drive/MyDrive/Assignment1/images/shapes.png' )
   ```

   Then you should convert it into the following:

   ```
   path = '/content/drive/MyDrive/Assignment1/images/'
   img = cv2.imread(path + 'shapes.png')
   ```

   Your path variable should be defined at the top of your Jupyter notebook. While grading, we are expecting that we just have to change the path variable once and it will allow us to run your solution smoothly. Specify, your path variable in the **README** file.
9. Answers to reasoning questions should be comprehensive but concise.

1 Image Classification using HoG (10 points)

Supposing that a company hires you to build a logo classification system that can automatically identify logos of different food brands. The company provides you with some images of different logos as training data, and you are asked to test your classification system on test images. You recall this can be solved using Histogram of Gradient (HoG) features.

1.1 Build a classifier with training images

For this task, you are given a set of training images: five 'Kraft' logos and five 'McDonald's' logos. These can be found here: 'Q1/training' folder. Examples are found in Figures 1 and 2.

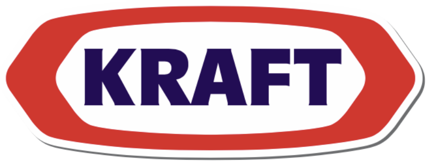

Figure 1: Example of Kraft logo

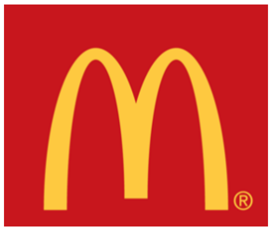

Figure 2: Example of McDonald's logo

1. Resize the training images to 128 x 128.
2. Compute HoG features of size (32, 32, 8) for all training images.
3. Apply blocknorm in 4x4 cell neighborhoods.
4. Fit a nearest neighbor classifier with three neighbors. Use **KNeighborsClassifier** from sklearn library.

1.2 Classify test images

You are given a set of test images: two 'Kraft' logos and two 'McDonald's' logos. These can be found here: Q1/test' folder. Examples can be seen in Figures 3 and 4. You are now asked:

1. Compute HoG features for all test images.
2. Display the HoG features for all test images.
3. Classify the test images using the classifier you built above.
4. Does the classifier work on all test images? Discuss which properties of HoG features help to classify the images. Which computer vision methods will you use for classifying logos under rotation.

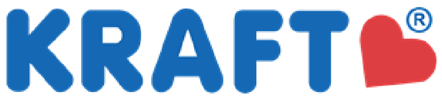

Figure 3: Example of Kraft test logo

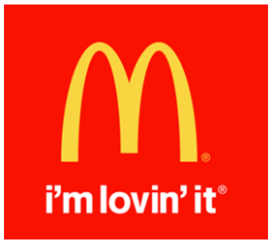

Figure 4: Example of McDonald's test logo

2 Image stitching (10 points)

You are given three different views of the same scene (check 'Q2' folder): image1.jpeg, image2.jpeg and image3.jpeg (see Figure 5).

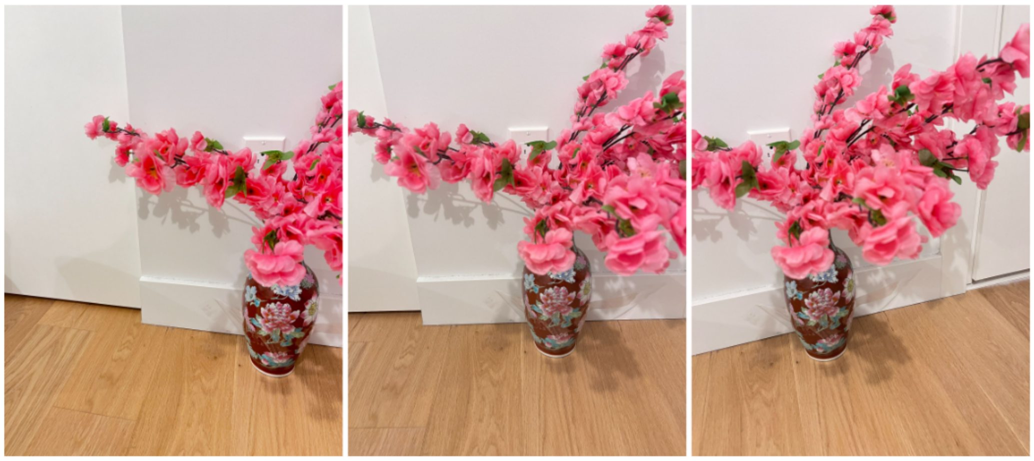

Figure 5: Flower taken from different views

1. Compute the SIFT keypoints and corresponding descriptors for image1 and image 2.
2. Find matching keypoints in image 1 and image 2, and display the 20 best pairs.
3. Using RANSAC method, find the homography that best describes matches the keypoints, and apply the resulting transformation to image 1. Image 2 should not be transformed.
4. Stitch the transformed image 1 and the original image 2 together using pyramid image blending. Let us call it image 12, and display it.
5. Compute the SIFT keypoints and corresponding descriptors for images 12 and 3.
6. Find matching keypoints in image 12 and image 3, and display the 20 best pairs.
7. Find the homography using the RANSAC method, and apply the resulting transformation to image 3. Image 12 should not be transformed.
8. Stitch the transformed image 3 and image 12 together using linear image blending. Display the resulting image.
9. Discuss which method you prefer for stitching - pyramid blending or linear blending, and why?

3 Face detection (10 points)

You are given a set of celebrity faces (check 'Q3/celeb_faces' folder). Load all images as training dataset and convert to grayscale image data.

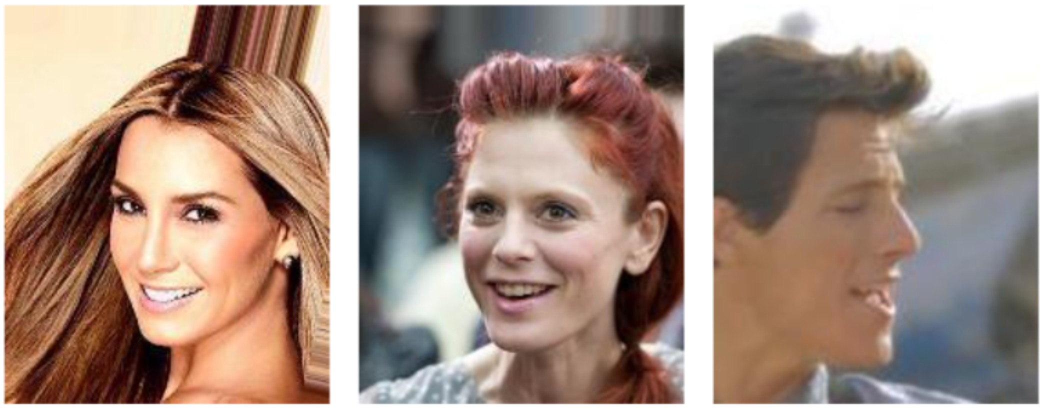

Figure 6: Example of faces

3.1 Eigenface representation

1. Display the mean face of the training dataset.
2. Find corresponding eigenvectors and eigenvalues, sort in descending order.
3. Print out the first 4 eigenvalues.
4. Let k be the first k eigenvectors. Try k = 20, 10, 5, and report the reconstruction error in each case.

Note : you are NOT allowed to use built-in PCA function from online packages, implement your own instead.

3.2 Face detection

Using a sliding window, detect face(s) for any test image of your choice. The test image should have at least one face it in. It can also have several faces:

1. Set k = 10, and display the test image with bounding boxes around all detected face(s) for the best threshold value.
2. Describe how well the method works.