Introduction to Computer Vision (ECSE 415)
Assignment 3

Due: November 5^{th}, 11:59PM

Please submit your assignment solutions electronically via the mycourses assignment dropbox. Students are expected to write their own code. (Academic integrity guidelines can be found at `https://www.mcgill.ca/students/srr/academicrights/integrity`). Assignments received up to 24 hours late will be penalized by 30%. Assignments received more than 24 hours late will not be graded.

There are five questions in this assignment. The questions permit you to explore segmentation and motion algorithms. Attempt all parts of this assignment. The assignment will be graded out of total of **85 points**. The first part of the first question requires you to implement several algorithms from scratch. You can use library-implemented functions for the rest of the assignment.

1 Submission Instructions

1. Title two jupyter notebooks as (i) segmentation (ii) motion_algorithm.
2. Comment your code appropriately.
3. Do not forget to run Markdown cells.
4. Do not submit input/output images. Output images should be displayed in the jupyter notebook itself. Assume input images are kept in a same directory as the codes.
5. Make sure that the submitted code is running without error. Add a README file if required.
6. If external libraries were used in your code please specify its name and version in the README file.
7. Answers to reasoning questions should be comprehensive but concise.
8. Submissions that do not follow the format will be penalized 10%.

2 Segmentation

Use image 'flower.jpg' for this question (Figure 1).

Figure 1: Image to be used for segmentation questions.

2.1 K-means clustering and Expectation Maximization

In this question, you will implement K-means and EM from scratch. The algorithms should be implemented using only the numpy library. You can use opencv and matplotlib libraries only to read and display images but not for clustering.

1. Implement K-means clustering from scratch. (10 points)
    - Apply your implementation of K-means to the provided image with K=2 and K=3. Use R, G, B color channels of the image as three features. Display (a) the segmentation evolving during the first 5 iterations and (b) the final segmentation. (6 points)
    - Convert the given image into gray-scale. Apply K-means to the provided image with K=2 and K=3. For every pixel, use gray-scale intensity as feature. Display (a) the segmentation evolving during the first 5 iterations and (b) the final segmentation. (6 points)
    - Compare final segmentation maps of color and gray-scale images. Which feature results into better segmentation? (2 points)
2. Implement EM from scratch. (15 points)
    - Apply your implementation of EM to the provided image with 2 and 3 Gaussian components. Use R, G, B color channels of the image as three features. Display (a) the segmentation evolving during the first 5 iterations and (b) the final segmentation. (6 points)

- Convert the given image into gray-scale. Apply EM to the provided image with 2 and 3 Gaussian components. For every pixel, use gray-scale intensity as feature. Display (a) the segmentation evolving during the first 5 iterations and (b) the final segmentation. (6 points)
- Compare final segmentation maps of color and gray-scale images. Which feature results into better segmentation? (2 points)

3. Generate datapoints by copying following code in your jupyter notebook. Cluster these datapoints into two clusters using your implementation of K-means and GMM. Which algorithm performs better and why? (5 points)

```
import numpy as np
"""
Following code generates datapoints for question 1.1 part
3.
"""
data = np.float32(
            np.vstack((
                np.random.normal(
                     loc=np.array([5,5]),
                     scale=np.array([3,2]),
                     size=(1000,2)),
                np.random.normal(
                    loc=np.array([-5,-5]),
                    scale=np.array([5,1]),
                    size=(1000,2)))))

```

Listing 1: Copy this code in your jupyter notebook to generate datapoints for question 1.1 part 3.

2.2 Normalized graph-cut and Mean-Shift segmentation

You can use functions from OpenCV, skimage and matplotlib libraries for this question.

- Segment the given image using normalized graph-cuts. Vary the following parameters (try several values of each parameter): compactness and n_segments (*slic* function), thresh (*cut_normalized* function). Display segmentation results for several parameters and state their effect on the output. (8 points)
- Segment the given image using mean-shift. Vary the following parameters (try several values of each parameter): ratio, kernel_size, max_dist. Display segmentation results for several parameters and state their effect on the output. (8 points)

3 Motion Algorithm

Use the given two video frames 'frame1.png' and 'frame2.png' shown in Figure 2 for this question. You can use functions from OpenCV and matplotlib for this

question.

(a) (b)

Figure 2: Video frames to be used for testing motion algorithm (a) frame1.png (b) frame2.png

3.1 Multi-resolution Lucas-Kanade optic flow detection

- Extract good points to track from 'frame1.png' using Harris corner detection algorithm. Use openCV function `goodFeaturesToTrack` and set parameter value `maxCorners`=500. Search optimal values for the rest of the parameters. Let us call the detected set of points: p_1. (2 points)
- Compute the optical flow between 'frame1.png' and 'frame2.png' at the points in p_1. Use the openCV function `calcOpticalFlowPyrLK`. Search for optimal values for the parameters. (2 points)
  Note that the function `calcOpticalFlowPyrLK` returns `nextPts` (a set of shifted positions of each point in p_1 which we refer as p_2), `status` (whether the search of shifted position is successful or not) and `err` (error measure between p_1 and p_2). Please read the manual for more details [1].)
- Display the optical flow image. (1 point)
- Vary the maximum pyramid level from 0 to 10 in the function `calcOpticalFlowPyrLK`. For each setting, compute the mean of the error at those points whose correspondence search is successful i.e. returned status is 1. Plot the mean (on y-axis) vs. pyramid level (on x-axis). Discuss the trends you observe in the plot. (4 points)
- Display the optical flow for each setting of maximum pyramid level. Comment on the quality of the results. (2 points)

[1] `https://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html#calcopticalflowpyrlk`