CSc 4980/6980: Computer Vision Assignment 3

Submission in Classroom:

Camera images of paper worksheets will NOT be accepted **Python: submit**

jupyter notebook and the .py files associated

MATLAB: submit a MATLAB Live script (.mlx file) and also convert the .mlx file to PDF and append to PDF from Part A.

The MATLAB Live Script document must contain all the solutions, including graphs. The file must be saved as ".mlx" format. See here for live scripts: https://www.mathworks.com/help/matlab/matlab_prog/create-live-script s.html

Manage all your code in a github repo for each assignment. Provide a link to the repo in the documentation workspace (jupyter notebook or mlx file).

Create a working demonstration of your application and record a screen-recording or a properly captured footage of the working system. Upload the video in the Google classroom submission.

Hardware: Unless otherwise specified, CAMERA refers to the OAK-D Lite camera provided to you.

Software: MATLAB: Either of the following will work: Use MATLAB R2018b or later version as installed in your machine (installation instructions already provided) **OR** Use MATLAB Online (https://www.mathworks.com/products/matlab-online.html).

For OAK-D you can implement your solutions in either Python or C/C++: https://docs.luxonis.com/en/latest/

1. Capture a 10 sec video footage using the camera. The footage should be taken with the camera in hand and you need to pan the camera slightly from left-right or right-left during the 10 sec duration. Pick any image frame from the 10 sec video footage. Pick a region of interest corresponding to an object in the image. Crop this region from the image. Then use this cropped region to compare with randomly picked 10 images in the dataset of 10 sec video frames, to see if there is a match for the object in the scenes from the 10 images. For comparison use sum of squared differences (SSD) or normalized correlation.

2. Implement the motion tracking equation from fundamental principles. Select any 2 consecutive frames from the set from problem 1 and compute the motion function estimates. Conduct image registration to realign the frames. Repeat test for all consecutive pairs of frames in the video.

3. For the video (problem 1) you have taken, plot the optical flow vectors on each frame. (i) treating every previous frame as a reference frame (ii) treating every 11th frame as a reference frame (iii) treating every 31st frame as a reference frame

4. Implement a feature based object detection application (from scratch) for detecting an object of your choice. Test it for at least 2 differently looking objects. Validate your results by testing against built-in object detection functions/code in MATLAB/OpenCV.

5. Implement a real-time face tracking application that will detect as many faces there are with a scene, and identify the person's facial region (draw a bounding box) whose is sought for by the user (you must ask for a person in your application and it should show a bounding box over the person of interest). Validate at least 20 times and present the recognition performance metrics (accuracy, precision, recall and Intersection over Union (IoU)).

6. Fix a marker on a wall or a flat vertical surface. From a distance D, keeping the camera stationed static (not handheld and mounted on a tripod or placed on a flat surface), capture an image such that the marker is registered. Then translate the camera by T units along the axis parallel to the ground (horizontal) and then capture another image, with the marker being registered. Compute D using disparity based depth estimation in stereo-vision theory. *(Note: you can pick any value for D and T. Keep in mind that T cannot be large as the marker may get out of view. Of course this depends on D)*

Question 7 is mandatory for all research students working in my lab and those who would like their application for Spring 2023 GRA in my lab to be considered (this is not a guarantee, rather you need to solve Q.7 for your application to be considered as priority). For everyone else Question 7 is optional

7. Implement Lucas-Kanade algorithm for motion tracking when the motion is known to be affine: $u(x,y) = a1*x + b1*y + c1$; $v(x,y) = a2*x + b2*y + c2$ – the numbers are subscripts, not power– You can pick any non-zero values for (a1,b1,c1,a2,b2,c2)