CSc 8830: Computer Vision

Assignment 2

Submission in Classroom: Manage all your code in a github repo for each assignment. Provide a link to the repo in the PDF document for Part A. Create a working demonstration of your application and record a screen-recording or a properly captured footage of the working system. Upload the PDF document and video in the Google classroom submission. (copying the script in the document is not required; GitHub repo must be accessible)

For parts that require or ask for "solve by hand" or "show by example" methods:

convert your problem solving by hand into a digital format (typed or scanned only. You can use camera scanner apps) and embedded/appended into the final PDF documentation. Camera images of paper worksheets will NOT be accepted

For programming, you can choose to program in either MATLAB or Python

If 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-scripts.html

If Python, you can submit it as a Jupyter notebook or the .py file. You should include clear commenting and ReadMe

Capture a 10 sec video footage using a camera of your choice. 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. For all the images, operate at grayscale unless otherwise specified:

1. Pick any image frame from the 10 sec video footage. Pick a region of interest in the image making sure there is an EDGE in that region. Pick a 5 x 5 image patch in that region that constitutes the edge. Perform the steps of CANNY EDGE DETECTION manually and note the pixels that correspond to the EDGE. Compare the outcome with MATLAB or OpenCV or DepthAI's Canny edge detection function.

2. Pick any image frame from the 10 sec video footage. Pick a region of interest in the image making sure there is a CORNER in that region. Pick a 5 x 5 image patch in that region that constitutes the edge. Perform the steps of HARRIS CORNER DETECTION manually and note the pixels that correspond to the CORNER. Compare the outcome with MATLAB or OpenCV or DepthAI's Harris corner detection function.

3. Consider an image pair from your footage where the images are separated by at least 2 seconds. Also ensure there is at least some overlap of scenes in the two images.

a. Pick a pixel (super-pixel patch as discussed in class) on image 1 and a corresponding pixel ((super-pixel patch as discussed in class)) on image 2 (the pixel on image 2 that corresponds to the same object area on image 1). Compute the SIFT feature for each of these 2 patches. Compute the sum of squared difference (SSD) value between the SIFT vector for these two pixels. Use MATLAB or Python or C++ implementation -- The MATLAB code for SIFT feature extraction and matching can be downloaded from here: https://www.cs.ubc.ca/~lowe/keypoints/ (Please first read the ReadMe document in the folder to find instructions to execute the code).
b. Compute the Homography matrix between these two images using MATLAB or Python or C++ implementation. Compute its inverse.

You can make assumptions as necessary, however, justify them in your answers/description.

4. Implement an application that will compute and display the INTEGRAL image feed along with the RGB feed. You **cannot** use a built-in function such as "output = integral_image(input)"

5. Implement the image stitching for a 360 degree panoramic output. This should function in real-time. You **can** use any type of features. You **can** use built-in libraries/tools provided by OpenCV or DepthAI API. **You cannot** use any built-in function that does output = image_stitch(image1, image2). You are supposed to implement the image_stitch() function

6. Integrate the applications developed for problems 4 and 5 with the web application developed in Assignment 1 problem 4*

**(Write an application – must run as a Web application on a browser and be OS agnostic – that implements the solution for assignment1 problem (3) [An application that can compute real-world dimensions of an object in view]. Make justifiable assumptions (e.g. points of interest on the object can be found by clicking on the view or touching on the screen).*