CSc 8830: Computer Vision

Homework Assignment 2

Submission in Classroom:

For Part A,

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 Part B 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

For Part C, 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 video in the Google classroom submission.

Hardware: Unless otherwise specified, use the OAK-D Lite camera provided to you. **Software:** 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/

PART A: Theory

1. 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.

2. 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 corner. Perform the steps of HARRIS CORNER DETECTION manually and note the pixels that correspond to the CORNER.

PART B: MATLAB Prototyping

3. Compare the outcome of problem (1) with MATLAB’s Canny edge detection function.

4. Compare the outcome of problem (2) with MATLAB’s Harris corner detection function.

5. Implement the image stitching application in MATLAB (not necessary to be real-time). Test your application for any FIVE of a set of 3 image-set available in the gsu_building_database. That is, your stitching application should stitch 3 images. You must test the performance of your application for FIVE such sets.

https://drive.google.com/drive/folders/1cgVYdrzn9yUpYYi14mgvNyQUv8Ym5gui?usp=sharing

PART C: Application development

6. Implement an application that will compute and display the INTEGRAL image feed along with the stereo and RGB feed. You **cannot** use a built-in function such as

“output = integral_image(input)”

7. Implement the image stitching, for at least 1 pair of images. This should function in real-time. You **can** use any type of features. You **can** use built-in libraries/tools provided by the DepthAI API.

If available, you can also simply call any built-in function “image_stitch(image1, image1)”. **However,** in that case, you need to show a 180 or 360degree panoramic output.

Questions:

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

1. (25pts) 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's Canny edge detection function.

2. (25pts) 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's Harris corner detection function.

3. (50pts) 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 pixels manually. Compute the sum of squared difference (SSD) value between the SIFT vector for these two pixels. Verify your result using MATLAB -- 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 manually. Verify your result using MATLAB.

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