CSc 8830: Computer Vision

Assignment 1

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 documentation to execute the script.

Preparatory exercises (need not submit this)**:**

(1) Go over the camera calibration toolbox demonstration: https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html. Use this exercise to calibrate any camera of your choice.

(2) Execute the Measuring Planar Objects with a Single Camera example: https://www.mathworks.com/help/vision/examples/measuring-planar-objects-with-a-calibrated-camera.html.

(3) Familiarize yourself with the Depth AI SDK. Run the tutorials/examples: https://docs.luxonis.com/projects/sdk/en/latest/

(4) With the OAK-D camera, set up your application to show a RGB stream from the mono camera and a depth map stream from the stereo camera simultaneously. Make a note of what is the maximum frame rate and resolution achievable?

1. Report the calibration matrix for the camera chosen and verify (using an example) the same.

2. Point the camera to a chessboard pattern or any known set of reference points that lie on the same plane. Capture a series of 10 images by changing the orientation of the camera in each iteration. Select any 1 image, and using the image formation pipeline equation, set up the linear equations in matrix form and solve for intrinsic and extrinsic parameters (extrinsic for that particular orientation). You will need to make measurements of the actual 3D world points, and mark pixel coordinates. Once you compute the Rotation matrix, you also need to compute the angles of rotation along each axis. Choose your order of rotation based on your experimentation setup.

3. Write a script to find the real world dimensions (e.g. diameter of a ball, side length of a cube) of an object using perspective projection equations. Validate using an experiment where you image an object using your camera from a specific distance (choose any distance but ensure you are able to measure it accurately) between the object and camera.

Note: In MATLAB, you can use MATLAB's *ginput()* function to record the pixel coordinates of the object's end points on the image.

Example usage:

```
I = imread('rice.png'); % Read the image
imshow(I); % Display the image
[x y] = ginput(2); % reads two points. x is a 2x1 column vector with x
                   coordinates and y is a 2x1 column vector with y
                   coordinates.
```

4. Write an application – must run as a Web application on a browser and be OS agnostic – that implements the solution for 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).