ECE4580 Homework #2

Due: Jan. 26, 2017

On t-square you should be able to find the necessary Matlab files containing all that you will need to implement the homework.

Note: There is no need to turn in any code. It is sufficient to include your lines of code in the homework document, or the values requested.

Problem 1. (25 pts) Implement the projection of a point in the world to a point on the image plane, then to a point read in by a sensor. Use the viewer-centered projection equations given by,

$$r^1 = f\frac{x}{z} \qquad \text{and} \qquad r^2 = f\frac{y}{z},$$

and the evenly-gridded sensor array equations,

$$R^1 = \lceil r^1/dr^1 \rceil + \frac{W}{2} \qquad \text{and} \qquad R^2 = \lceil r^2/dr^2 \rceil + \frac{H}{2},$$

where $\lceil \cdot \rceil$ is the ceiling function. The sensor properties are those from a Panasonic WV-BP104 CCTV Camera, which are as follows: the focal length parameter is 4 millimeters, the sensor dimensions are 4.8×3.6 millimeters, and the sensor resolution is 800×600 (all given as width $\times$ height). These should be implemented in Matlab by a function `camera.m` that takes in (x, y, z) world coordinate in meters (in 3x1 column vector form) and returns (R^1, R^2) coordinate in pixels (in 2x1 column vector form).

(a) What is the field of view of the camera (horizontal and vertical) in degrees?

(b) The camera projection equations plot generalized lines to to generalized lines (where a point is a *degenerate line*, e.g., a line that goes nowhere). After transformation under the camera projection equations, plot the following lines and tell me what you see:

1. the line going from $(-36.6, -25.7, 66.0)$ to $(23.1, 0.1, 77.0)$, and
2. the line going from $(-45.0, 49.7, 150.0)$ to $(-81.0, 89.5, 270.0)$.

Either plot it with the axes set to $[0,\ 800,\ 0,\ 600]$ using the axis command, or actually generate an image (this is a bit more challenging) with zeros where there is nothing and ones where the line is.

(c) Likewise, plot the following:

1. a square whose bottom left corner is located at $(-30.0, -15, 100.0)$ of width 7.5, and another one located at $(0.0, 27.6, 160.0)$ of width 12. The other corners of the squares should have the same z-distance as the bottom left corner (so they are standing squares, so to speak).

Comment on their relative sizes? Make sure to set axes to be equal (e.g., 'axis equal') for all parts of the problem.

Make sure that your homework submission includes the plots in it. Please include your code in the submission document.

Problem 2. (15 pts) On t-square, there will be a Matlab file, called `dpic`, containing a depth image taken using Microsoft's Kinect camera. If you've played on the X-Box with the Kinect, you know that the game console can depict a little image of you based on the camera information. In the Matlab file there is a depth image with a person in it. Try to find two threshold values, upper and lower, such that when you threshold the images, what is left is a binary blob that captures the person. Now, there might be other things that pass the threshold too. Use Matlab's `bwselect` to pick out only the blob the corresponds to the person.

Turn in, the depth range thresholds, the pixel coordinate used for `bwselect`, and the final extracted blob of the person.

Problem 3. (10 pts) One thing that you may or may not have noticed about the edge score threshold problem is that trying to get some edges to pop up may have introduced some classified edge areas that you did not want, or that maybe trying to exclude areas you didn't want also removed edges that you did want. These are examples of classification errors.

In computer vision, there are lots of algorithms that try to decide between two options. Usually these are called (two-class) classification algorithms. There are actually four types of answers that can result from a classification algorithm. They must be computed and provided if you want to demonstrate the value of an algorithm.

Explain these four types. Select two of these types and explain how they relate to the edge classification problem? It would be nice if you could go back to your original image and annotate the regions that correspond to these two types. When reporting classification errors of an algorithm, a confusion matrix is used. Explain the confusion matrix in terms of the different answer types.

Hint: Keywords include "true positive" and "false negative". What are the others?

Problem 4. (15 pts) In computer vision, binary decisions can sometimes be frustrating due to their, well, binary nature. It is or it isn't, yet sometimes you'd like it to be the other. Well, the clever people in image processing have come up with this notion of performing a hysteresis type of computation.

Hysteresis is hard to explain, but a simplification is to say that the direction matters when considering the decision. If a value is going upwards, then the threshold should be a bit higher. When it is going downwards, then the threshold should be lower. By creating two thresholds, that are conditionally dependent, it is possible to capture regions that might fail the first threshold test.

In a hysteresis approach to thresholding, there is an upper and a lower threshold. First, the algorithm binarizes based on the upper threshold to arrive at candidate seed points. The second step seeks out neighbors of the candidate seed points that pass the lower threshold test. In this way, some of the intially rejected regions can be subsequently accepted.

The algorithm goes as:

1. First apply a high threshold to get candidate regions. (the easiest might be to use a threshold from your homework, or maybe a threshold from the solutions).
2. Apply a lower threshold to get a bigger set of regions, let's call them the forgiving regions.
3. Using the points in the candidate regions and accept their (connected) neighbors if they are part of the forgiving regiions.
4. The final answer is the set of candidate regions and the accepted neighbors.

A Matlab code stub, `edgefind.m`, already exists sort of outlining the above algorithm. The code stub gives hints as to what Matlab functions to use in order to have simpler, more efficient code. Your job is to follow through on the documented hints to finish the hysteresis algorithm as applied to edge detection. (Just as a note, the `edgefind` function generates the same score as the `edges1` score, so that's the threshold that can be recycled.)

Using the grayscale image from the last edge assignment, finish up with the hysteresis-based edge finding function and give it a run:

a) Flesh out the code associated to the code stub.

b) Pick a pair of hysteresis threshold values for the edge finding code stub. Turn in the threshold values plus the edge image, and explain your selection.

c) Explain what you found easy or challenging about the upper and lower threshold selection. Do you like the results better than when using a single threshold?

Note: This won't be discussed, but is used a lot in computer vision/image processing. Wikipedia has an OK discussion of hysteresis in engineering to explain it. A similar idea can apply to the binary decision process of determining edges. However rather than performing temporal hysteresis this problem is asking for spatial hysteresis. For this, you will need to find connected components (Matlab has a function for this as hinted in the code stub).