CSCE 439/839: Robotics: Algorithms and Applications Spring 2020 (COVID-19 Edition)
Homework 3

Started: Thurs, April 16
Due: Tues, May 5 by Midnight

Instructions: This homework is an individual assignment, collaboration is not allowed. If you discuss any problems with others, please note this on the assignment as described in the syllabus. Also note any materials outside of lecture notes, course textbooks, and datasheets that you used. Show your work and describe your reasoning to get partial credit if your solution is incorrect.

You should also make sure that you properly label and **describe** any figures or plots that you include in your writeup. You will not receive full credit if you do not explain these. In addition, you should refer to your code where needed to answer the questions (e.g. say "See file mynode/launch/test.launch for this problem, which ...").

You must turn in a pdf of your assignment on Canvas. Make sure to answer questions in complete sentences and explain answers as needed. **You must also turn in your code for all parts of this problem on Canvas.** Failing to electronically turn in your code will result in a 10 point penalty on this assignment. Points may also be deducted for coding errors, poor style, or poor commenting.

If you use any code from an online or other source you must cite the source in the comments. Otherwise it is considered plagiarism, which we check for!!

Note: This assignment uses OpenCV, which is one of the most popular and powerful image processing libraries. I'm giving you the code you need and just asking you to use it (this is after all a robotics class, even though vision processing is important and often used in robotics). However, feel free to take a look at the source code and to take other image processing or deep learning classes if you want to learn more about image processing.

Name:

Problem 1. *(5 pts)*[1] *(To be completed at end of assignment) Approximately how much time did the total assignment take? Which sub-problem took longest and how much time did it take? Are there any questions that need clarification?*

[1]Each HW counts equally in your overall grade, even if homeworks have different point totals. This one is out of 118 points for 439 and 118+20 points for 839.

Problem 2. *Edge Detection*

a). *(5 pts) Give the basic 3-element vector mask to detect vertical edge.*

b). *(5 pts) Give the output of applying the mask vector part a) (above) to the following matrix. Assume that elements outside of the bounds of this matrix are zero and the resulting matrix should have the same dimensions.*

$$\begin{bmatrix} 50 & 50 & 100 & 200 & 200 \\ 50 & 50 & 100 & 200 & 200 \\ 50 & 50 & 100 & 200 & 200 \\ 50 & 50 & 100 & 100 & 100 \\ 50 & 50 & 50 & 50 & 50 \end{bmatrix}$$

c). *(5 pts) If the edge detection had a significant amount of noise, how could you reduce the noise?*

Problem 3. *Bag Files*

ROS has a tool called `rosbag` *that allows you to record and replay messages. Since you can't run the following code on your actual robot with a camera, I am providing bag files with recorded images for you to use. Read the tutorial on the rosbag command line tutorial for this part at:* `https://wiki.ros.org/rosbag/Commandline`.

NOTE: *For this problem, use the bag file labeled with the letter range that includes the* **FIRST letter of your FIRST name.**

a). *(2 pts) Which bag file are you using and why?*

b). *(5 pts) What topics are contained in this bag file and how many messages of each type are in the bag file? How did you determine this information?*

c). *(5 pts) Start up the basic turtlebot (*`rosrun turtlesim turtlesim_node`*). Playback the bag. How did you playback the bag? What letter does the turtle draw?*

d). *(5 pts) Restart the turtlebot and also startup the teleop (* `rosrun turtlesim turtle_teleop_key`*). Record just the* `/turtle1/pose` *topic in a rosbag (and no others) while you draw (you guessed it) "CS." How did you record just the one topic? Include this bagfile in your submission.*

e). *(5 pts) Passing the flag* `-r` *when playing a bag file allows you to adjust the playback speed. This can be particularly useful when doing vision processing (like in the next sections) if you have a slower virtual machine. How can you play it back at half speed? Give the full command you use.*

Problem 4. *Visual Landmarks*

On the course website there is link to download ROS modules needed to identify barcode-like visual landmarks (called `landmarkSelfSim`*). Download this code and place it in your ros directory. These are "self-similar landmarks"*[2]*, which means they are easily and quickly identified by only looking along one scan line in an image, no matter how far away they are. In addition, they have a binary bar code on the side that uniquely identifies each landmark.*

There are two different launch files for the vision code `landmarkSelfSim.launch` *and* `displayDebugImages.launch`*. The first launches the landmark detection system and the second is used to display the output images, which is useful for debugging.*

NOTE: *For this problem, use the bag file labeled with the letter range that includes the* **FIRST letter of your FIRST name.**

a). *(2 pts) Which bag file are you using and why?*

b). *(5 pts) Run the bag file and landmark detection code. How many landmarks are visible and what are their ID numbers?*

c). *(5 pts) What message does the landmark detection code publish about the location of landmarks? How would you find the center of the landmark?*

[2] D. Scharstein and A. Briggs. Real-time recognition of self-similar landmarks. Image and Vision Computing, 19(11):763-772, September 2001.

Problem 5. *Ball Detection*

In the code download for this lab there is code to detect balls (called `ballDetector`*). In the* `launch` *directory, there are two launch files (similar to the landmark detection code). The first,* `ballDetector.launch`*, launches the ball detector code. The second,* `displayDebugImages.launch`*, will display three different images you can use for debugging. If you find that the ball detector slows down your system, you can disable the debug images and information by commenting out the* `#define BALLDETECTOR_DEBUG` *line in the* `ballDetector.cpp` *file.*

The ball detector works by first converting the image to HSV color space. It then performs thresholding to filter out all pixels that are between a low and high HSV threshold. These values can be configured in the launch file (see `ballDetector.launch` *for an example) and can be dynamically changed by running the command* `rosrun ball_detector configGUI.py`*. It then searches for the largest group of connected pixels that have similar height and width (as a crude approximation to find circular groups of pixels). The largest group that meets this criteria is selected as the "ball" in the image and a message is sent out with this ball location.*

The debug images display the HSV image, thresholded image (white pixels indicate those that are between the low and high thresholds), and a marked up image with white pixels indicating the boundaries of connected groups. In addition, in this image the final ball location is marked with a circle. Use these images and the configGUI to pick good HSV low and high thresholds.

NOTE: *For this problem, use the bag file labeled with the letter range that includes the* **LAST letter of your FIRST name.**

a). *(2 pts) Which bag file are you using and why?*

b). *(5 pts) Find HSV thresholds that track the ball well throughout the whole image sequence. What are the values and how did you go about determining these values?*

c). *(5 pts) (Small interlude on pinhole cameras) Draw the pinhole camera model and label the focal distance, image plane, size of an object in the image, the pinhole, distance in the world, and size of an object in the world.*

d). *(5 pts) What is the equation for distance of an object in the world based on the known height of the object and other parameters in the pinhole model? Use the same notation as that in your drawing.*

e). *(15 pts) (Now back to ball detection) Assume a pinhole camera model and that the ball is initially 1 meter from the camera. Create a new ROS node to calculate the distance to the ball throughout the whole*

sequence. **Describe** *how you approached and solved this problem. Include a plot showing the distance over time.*

f). *(5 pts) Implement a filter that averages the prior 15 distance readings to obtain a smoother estimate. How did you handle cases where the ball was missing? Make sure to include a plot showing a comparison of the distances with and without the filter.*

g). *(20 pts)* **839 Only:** *Implement a Kalman Filter to filter the distance of the ball over time by estimating the distance and the change in distance (the z velocity). Describe your implementation and how you handled the cases where the ball was missing. Compare this to the above averaging filter and the raw distance reading. Make sure to include a plot showing the distance and velocity estimates.*

Problem 6. *Robot Visual Tracking*

We will now combine the landmark detection and ball detection. In the bag file there is a recording of a robot moving around with a ball on top. There are two landmarks. The center of landmark 8 defines the origin (0,0) of the operating area. Landmark 16 is placed at (1,0) or 1 meter along the x-axis. For this problem, assume a pinhole model and that the camera is parallel to the ground plane.

NOTE: *For this problem, use the bag file labeled with the letter range that includes the* **LAST letter of your FIRST name.**

a). *(2 pts) Which bag file are you using and why?*

b). *(25 pts) Write a ROS node to calculate the position of the robot throughout the whole sequence. Include a detailed discussion of how you approached this problem and calculated the results. Make sure to address how you handled times when visibility of the landmark is lost. Note: You can ignore the last bit of the bag file after the robot stops moving.*

Do not forget to fill in the amount of time you spent on this assignment in Question 1.