CSE 468/568 Programming Assignment 3 Laser-Based Perception and Navigation with Obstacle Avoidance

The objective of this assignment is to perform perception using a laser range finder, and use the perceived information to navigate to a given destination while avoiding obstacles.

Use the lab3 package provided here and the husky_playpen same as lab2

Perception Using Laser Range Finder (50 points)

For this section, you will implement the RANSAC algorithm to determine the walls "visible" to the robot from the data obtained from the laser range finder. Your program should take the laser scans as inputs, and output a set of lines seen by the robot identifying the obstacles in view.

The RANSAC algorithm is as described in class. From the set of points the laser range finder gives, pick two at random and draw a line. Find out the distance of each of the other points from this line, and bin them as *inliers* and *outliers* based on if the distance is lesser or greater than a threshold distance `(min_dist)`. Repeat this for k iterations. After k iterations, pick the line that has the most number of inliers. Drop the inlier points of that line, and repeat the algorithm to the remaining set of points until you have lower than a threshold number of points `(min_points)`. You'll need to experiment with these parameters to find values for the number of iterations, the threshold distance for inliers, and the threshold for the number of points below which you cannot detect lines.

Read through the `rviz` tutorials. You have to read through the user guide, and built-in data types. Then read through the first two tutorials that explain how you can use markers.

As a demonstration of your implementation of the RANSAC algorithm, publish the detected lines as lines that can be visualized in `rviz` . `rviz` should visualize the detected lines in the robot's local frame. A simple way to verify that your published lines are correct is to enable both the published lines, as well as the laser scan in `rviz` . If they overlap, then you are detecting the lines correctly.

Please note that `rviz` visualization takes some time for students and it has challenges. You should get started as soon as you can. Some of `rviz` also depends on the graphics hardware and processor speed you have. You can get around these limitations by

- Not using hardware graphics rendering. You can do this by

```
$ export LIBGL_ALWAYS_SOFTWARE=1
```

in your `.bashrc` or on the command line where you are running the `rviz` node

- reduce the frame rate so it renders at a reasonable speed. You can do this as soon as the `rviz` window shows up by changing the frame rate parameter in the column on the left. e.g. keep it at 5 frames per second instead of the default 30 frames per second.

Test to see if you can launch everything by running the command:
`$ roslaunch lab3 perception.launch`

Note : You need to save your `rviz` configuration and include it in the perception.launch file

Bug2 algorithm (50 points)

For this portion, you will implement the `bug2` algorithm we recently learnt about. Your robot starts at a location defined in the world file. It should plan its path to a `goal` location. The robot will have to navigate itself avoiding the various obstacles in its way. Implementing bug2 can be based on the pseudocode that is on the slides from class, or the original described bug2 algorithm (either is fine, as long as the behavior is correct). Your robot will be in one of two states: *GOAL_SEEK* and *WALL_FOLLOW*. The key to this will be the *WALL_FOLLOW* where you will have to use the lines detected in the previous section to drive in parallel to it. Please use launch file `bug2.launch` that will launch the world and execute your controller. You should use "rosparams" to obtain the `goal` point. In your node, get rosparams and save them in your variables. To ensure those rosparams are populated, set a value for them in the launch file or in a dedicated params.yaml file. When grading your assignment, we'll modify rosparam values in your launch file and test your code, pick reasonable names. Test to see if you can launch everything by running the command:
`$ roslaunch lab3 bug2.launch`

Note : It is not mandatory to use RANSAC for Bug2 although you may choose to do so.

Submission Instructions

You will submit `lab3.zip`, a compressed archive file containing the lab3 package (folder). Please take care to follow the instructions carefully so we can script our tests. Problems in running will result in loss of points.

Please use UB Learns for submission.
The assignment is due Friday, October 21st before midnight.