CS/ECE/ME/BME/AE 7785
Lab 4

Due: November 9th, 2018. 3pm

1 Overview

The objective of this lab is to get you familiar with the mapping, localization and path planning capabilities of the ROS navigation stack. The final goal is to have a simulated or real robot autonomously navigate to a series of global waypoints.

We strongly encourage you to use all available resources to complete this assignment. This includes looking at the sample code provided with the robot, borrowing pieces of code from online tutorials, and talking to classmates. You may discuss solutions and problem solve with others in the class, but this remains an individual assignment and each person must submit their own solution. Multiple people should not jointly write the same program and each submit a copy, this will not be considered a valid submission.

2 Lab Instructions

2.1 Create a map through teleoperation

Generate a map of your environment using the instructions found at:

http://emanual.robotis.com/docs/en/platform/turtlebot3/slam/

In later stages of the lab, you will have your robot navigate to a predefined goal. To ensure that everyone uses (approximately) the same global coordinates, start your robot with its wheels on the blue tape marks on the floor, facing in the direction of the arrow. Then drive your robot around to complete the map.

2.2 Localization, Path Planning, and Navigation

Use the map youve generated to have the robot navigate to a point you specify in the rviz GUI. Instructions found at:

http://emanual.robotis.com/docs/en/platform/turtlebot3/navigation/

Note that this will use a particle filter for localization and A* for path planning.

2.3 Gazebo Simulation

Check that both of the above capabilities work in simulation. Start by launching Gazebo using the instructions linked in the lab assignment. Note that sometimes Gazebo crashes on startup, you may need to start it more than once.

Once Gazebo is running, run subsection 2.1 and subsection 2.2 above in simulation instead of on the robot. To achieve this, perform the same steps as in subsection 2.1 and subsection 2.2 above, but with Gazebo running instead of bringing up the real Turtlebot. Gazebo will subscribe and publish the same messages as the real robot.

2.4 Drive to global waypoints

Read in `global_waypoints.txt` and navigate the robot to each waypoint in turn. Specifically, the file will contain 3 waypoints not known to you until grading time. An example file is provided with this assignment, but feel free to try other waypoints. Your code should have the robot drive to the first waypoint, stop for 2 seconds, then go on to the second waypoint and so on. The waypoint file lists seven values per line, corresponding to the 3D position and the orientation quaternion for the waypoint, in that order.

3 Parameters that should be tuned

When using the navigation stack, there are a few parameters you can adjust to get more consistent performance. All of the parameters are found in the directory `turtlebot3/turtlebot3_navigation/param` directory, and they're loaded by the navigation launch file. All you have to do is change the parameters in the various .yaml files (you can open these with a text editor), no compiling needed.

Recommended parameters to adjust:

- **dwa_local_planner_params.yaml**
  - **xy_goal_tolerance, yaw_goal_tolerance:** Increasing the goal tolerance will allow the robot to stop when it's "close enough" to a goal point. This can prevent the robot spinning around in circles for a while when it gets close to a goal to get to the exact correct point, provided your solution can handle reaching a larger area as a goal rather than a very specific point/angle.
  - **path_distance_bias, goal_distance_bias:** Increasing these will cause the robot to more strongly follow the planned path or move towards the goal. Think of it like the switching controller we discussed in the lecture.
  - **max_vel_x, min_vel_x, max_rot_vel, min_rot_vel:** These all set bounds on how fast the robot will move. Fast moving robots may

shake and mess up localization, so reducing maximum velocities may improve performance.

- **costmap_common_params_burger.yaml**
  - **inflation_radius:** This will determine how large to make obstacles. Increasing this will prevent the robot from driving close to walls.
  - **cost_scaling_factor:** This will determine how much the robot should be repelled by obstacles. Increasing this will make the robot prefer not to drive close to walls, but will allow it if necessary.

4 Grading

Show an instructor your robot driving around in Gazebo	40%
Navigate to 3 waypoints specified at grading time	60%

5 Submission

You have two required submissions for this lab.

1 Your ROS package in a single zip file called `LastName_FirstName.zip` uploaded on Canvas under Assignments–Lab5.

2 A live demonstration of your code to one of the instructors. This can be done anytime before the due date at the top of this lab. Class will meet in the lab room on the due date to allow everyone to demo their navigation. If you demo before the due date you do not need to come attend class that day.