EE346 - Mobile Robot Navigation and Control
Fall 2024
Laboratory #4 (4%)
Due: November 13, 2024
Rqt and Homing

Objectives

- Learn how to use rqt as a tool to visual robot sensory data
- Observe the behavior of the IMU on OpenCR of TurtleBot3
- Practice how to create a ROS package.
- Implement a robot trajectory controller in the homing application.

Procedure

In this lab, in Part I, you will learn how to use the rqt_multiplot and rqt as a tool for visualizing real-time data and debugging ROS applications in general. In Part II, you will create your own package where the robot looks for a pole-like object in the environment and approaches the object until it is within 15cm of the object.

Part I: Rqt, Rqt_multiplot and other rqt plugins

Rqt – built from with qt – is a software framework of ROS that implements various graphic user interface (GUI) tools in the form of plugins. For plotting a ROS graph, for example, you have used the rqt plugin: rqt_graph. Another useful rqt plugin to learn in this lab is rqt_multiplot, an advanced version of rqt_plot that allows you to visualize streamed data that are published on ROS topics. For further information, this page provides a complete list of rqt plugins.

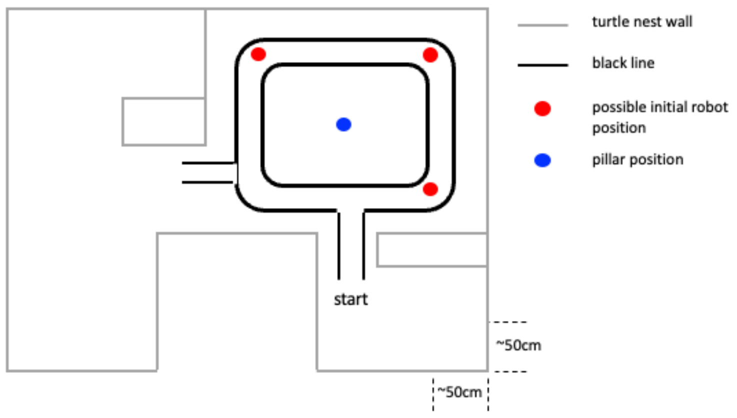
In the first part of the lab, you will use rqt_multipot to visual the IMU data measured by ICM-20648 on OpenCR of TurtleBot3. Once you are familiar with the operation of rqt, you can use it in later lab assignments for debugging your software implementations. Follow the steps below.

1. Watch this 12-min tutorial on rqt_multiplot to gain an understanding of rqt_multiplot.
2. To install rqt_multiplot on your computer, run:
   `% sudo apt install ros-noetic-rqt-multiplot`
3. Start your TurtleBot3 as in Part I of Lab 3 so that you can move your robot with RC-100.
4. Start rqt_multiplot: `% rosrun rqt_multiplot rqt_multiplot`
5. Create three curves, one for each of the axes of the IMU that can have non-zero measurements.
6. As an option, create a curve for the acceleration vertical to the ground, and move the robot manually to observe the acceleration measurement.
7. Move the robot around with RC-100 and observe how the IMU curves change over time.
8. Start rqt by running: `% rqt`
9. Run rqt_graph plugin by selecting from the “plugins” pull down menu: `Introspection` and then `Node Graph`.
10. Browse messages published on topics by using the “Topic Monitor” plugin under the Topics group and select the three axes you plot in Step 4 through the /imu topic.

Once the above is working, demo your solution to a TA and be prepared to answer questions.

Part II: **Homing**

In this part of the lab, you will create a package that looks for a pole-like object in the robot environment and approaches the pole once it is found. The location of the pole is near the center of an open environment, and you can assume that only one such object exists. You can use the trajectory control algorithm that we have studied in the class described in Siegwart's book or the controller provided in an example (see below). Your robot should stop near the pole within 15cm without hitting the pole for the homing task to be considered a success. Tune the gains of the trajectory controller so that you robot can approach the goal position *as quickly as possible*. Refer to the C++ example at this GitHub site to experiment with a simple pole/pillar detection method (study the "`LaserCallback`" method).



To test your solution, the pillar is placed in the center of the robot environment marked by the blue circle. Your robot is placed at one of the three locations marked by the red circle. As a first step, your robot should detect the direction of the pillar and turn so that its heading is oriented toward the pillar. Once successful, in the second step, your robot should detect and move toward the pillar until it is within 15 cm of the pillar, all along a single smooth trajectory.

Once you have completed the first or second step of this part, demonstrate it to a TA. Record the time it takes for your robot to complete this task. Submit the package by uploading it to as a GitHub package and inform the TA of your GitHub link.

Marking

If you are able to complete the lab before the end of the second week, you will receive 2% for Part I and 2% for Part II. If you are not able to complete any parts of the demo within the lecture session, you will get a 20% penalty of the part weight, and an additional 20% for each day of delayed demo (checking by a TA).