EE346 - Mobile Robot Navigation and Control

Fall 2023
Laboratory #2 (3%)
Due Date: October 11 (16:00)
Getting Up to Speed with Linux and Python
（Draft）

Objectives

- Learn the basics of the Robot Operating System (ROS) and how to create packages.
- Study how ROS topics work by creating and running simple publishers/subscribers.
- Understand how tf works in ROS.

Procedure

This two-week lab will expose the students to Robot Operating System (ROS). The students will go through sections of existing ROS tutorials to learn how to start ROS and how ROS nodes can communicate with each other through ROS topics. As well, the students will experiment with simple ROS nodes that involve the use of the tf package that handles coordinate frame transformations in ROS.

Part I: ROS Basics

Visit the ROS tutorial for beginners at https://wiki.ros.org/ROS/Tutorials. Under "1.1 Beginner Level", go through sections 1-6. In Section 1, install the ROS distribution "Noetic". Make sure that you can move the simulated TurtleBot with "rostopic" and duplicate the ROS graph at the end of Section 4.1. You do not need to worry about "5. Using rqt_plot".

Show the result to a TA to demonstrate that you have completed this step.

Part II: Simple ROS Publisher and Subscriber

Also, in at https://wiki.ros.org/ROS/Tutorials, study Sections 12 and 13 on "Writing a Simple Publisher and Subscriber (Python)" and "Examining the Simple Publisher and Subscriber", to make sure that you can reproduce the result upon executing:

% rosrun beginner_tutorials talker.py

in one terminal and

% rosrun beginner_tutorials listener.py

in the second terminal.

Then, watch the video in the "Video Demonstration" section. Create the node "turtlebot_saver.py" and make sure that you can duplicate the correct stopping behavior of the simulated turtle robot. Try other constants than "7" to see the change in the robot behavior.

Show the result a TA to demonstrate that you have completed this step.

Part III: TF and Turtle Robot Following

"tf" is a ROS package that lets the user keep track of multiple coordinate frames over time. tf maintains the relationship between coordinate frames in a tree structure, and lets the user transform points, vectors, etc. between any two coordinate frames. Refer to the wiki page at https://wiki.ros.org/tf for a high-level description of what tf can do.

Next, go through the tutorial at https://wiki.ros.org/tf/Tutorials/Introduction%20to%20tf, which contains a concrete example of how tf is used in maintaining coordinates associated with the two simulated turtle robots, one following the other. Go through the tutorial and understand how the following tf tools are used: tf_monitor, view_frames, rqt_tf_tree, tf_echo, and rviz (the ROS robot visualizer) with respect to its support for tf.

Then add another node to this ROS application, based on turtlesim_saver.py defined in the video mentioned in Part II. Modify it so that the robot does not go beyond the box of $x \in [-10, 10]$ and $y \in [10, 10]$ and that the turtle robot moves in the reverse direction whenever it hits a border of the defined square. Include this new node in the launch file turtle_tf_demo.launch.

Upon completing this part, demonstrate your solution and be prepared to interpret the outputs from

% rosrun tf tf_echo turtle1 turtle2

while your application is running.

Marking

If you are able to complete the lab before the end of the second week, you will receive:

Part I: 1% (individual demonstration)
Part II: 1% (individual demonstration)
Part III: 1% (group demonstration)

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).