MINGI JEONG
F00422M
Mingi.Jeong.GR@dartmouth.edu

1 Description

1.1 Introduction

This program was made to implement the codes of Rosbot's control and communication. By considering pose on odom frame, this ROS program subscribes to 'PoseStamped' message which contains position and orientation after the program runs serialbridge.sh. When the program is executed on a command shell, the user is advised to input distance to be travelled on task 1. While the robot is moving, you can check how far the robot has moved on the command shell by printing function. Basically, I made rosbot class on task 1, task 2 so that the class can contain modularity of robot operation including subscriber and publisher.

1.2 How my program work

- To run and configure the package, please see the attached readme file. Make sure to execute ssh, launch, rviz, and the python code accordingly.
- For task 1, you can run "roslaunch cs169_prog_assign_1 task1.launch" and input the distance. There is a time delay for distance traveller.
- For task 2, you can run "roslaunch cs169_prog_assign_1 task2.launch." and it will go 1 meter. It also records the designated topics by me. There is time delay for distance traveller and logging.
- For task 3, you can run "roslaunch cs169_prog_assign_1 task3.launch" and make sure you run rviz on your laptop's command line by setting ROS_MASTER_URI correctly inside .bachrc file. For controlling robot, I made it through teleop keyboard node. Therefore, while you are using the robot by the keyboard, you will see how rviz changes.
- On each launch file, even if the professor did not ask, I contained sensors or other nodes together so that I can check all things at the same time.

1.3 Design Decision

Basically, each launch file contains everything. Task 1's distance input screen is as shown in Fig. 1 For task 2, I tested the robot's movement in front of lab 201 in Sudikoff building of Dartmouth. Rosbag launch file was separately made for executing logging in a modular way. The recorded topics are found from rostopic echo and image raw topics were filtered out. The result is shown as Fig. 2. For task 3, I wrapped a marker publisher helper function inside subscriber call back function so that it can be initiated whenever the message is subscribed.

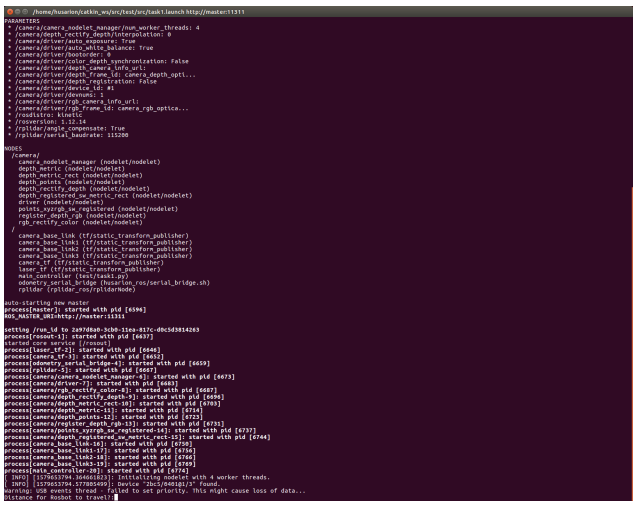


Figure 1: Task 1 execution screen

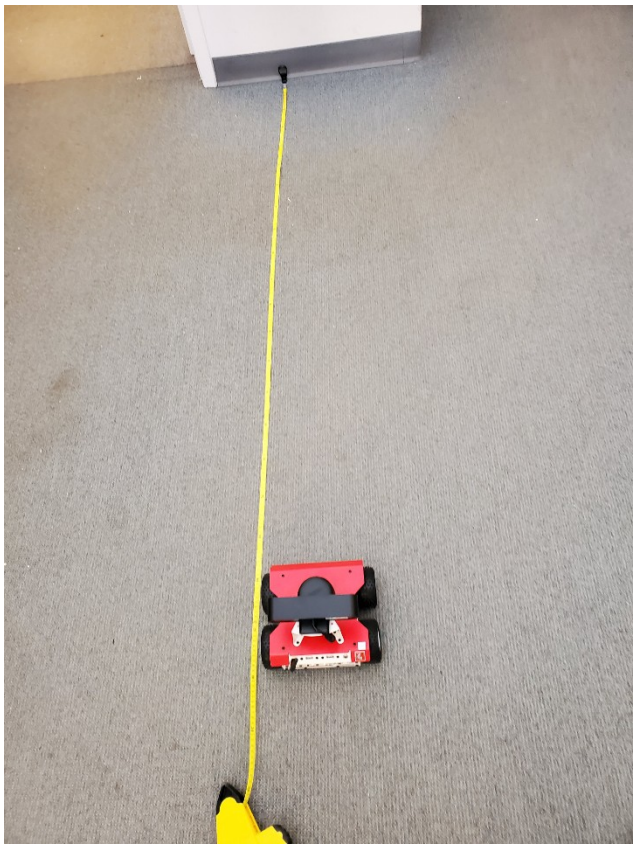


Figure 2: Task 2's test at Sudikoff building

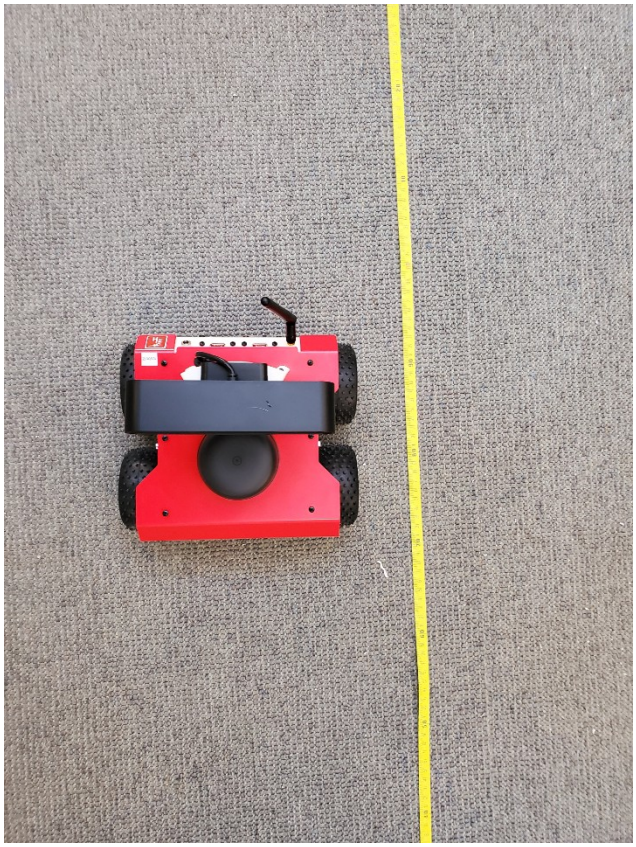


Figure 3: Task 2's test result

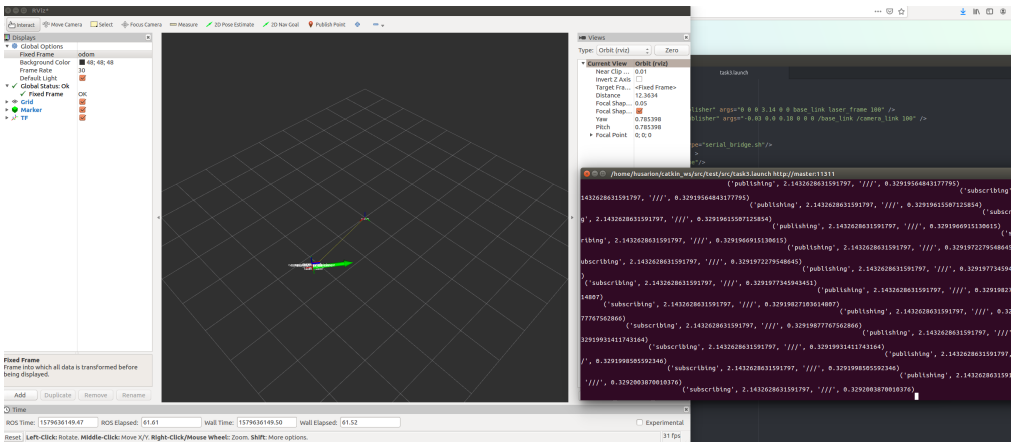
2 Evaluation

2.1 Performance

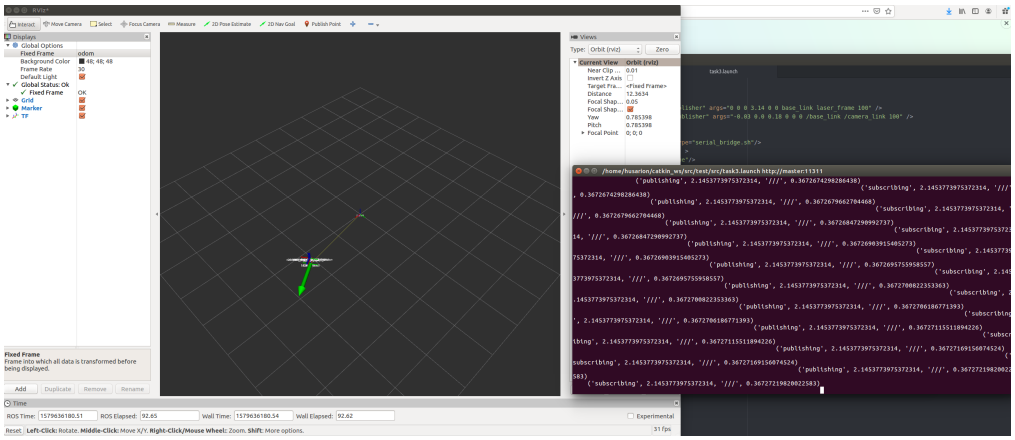
- For task 2, while the logging launch is executed, I kept encountering errors regarding transformation of images. However, you can ignore this as the rosbag file is correctly recorded by dropping unnecessary topics. Other than that, it works fine.
- As for 1 meter travelling, it is a bit off by making the robot reach about 1.15 meter as you can see from the attached csv file. I guess one reason will be publishing messages based on rospy rate. Somehow, frictional force on the floor might be a small factor to this. However, it is necessary to keep watch on how it would travel according to different rate and termination of node when it reaches the goal.
- For task 3, you can easily find out the robot's pose including position and orientation once you also visualize TF. It is possible that the robot's position might be off the grid of RVIZ, but you can check with TF easily. I tested task 3 as the robot moves.

2.2 Result

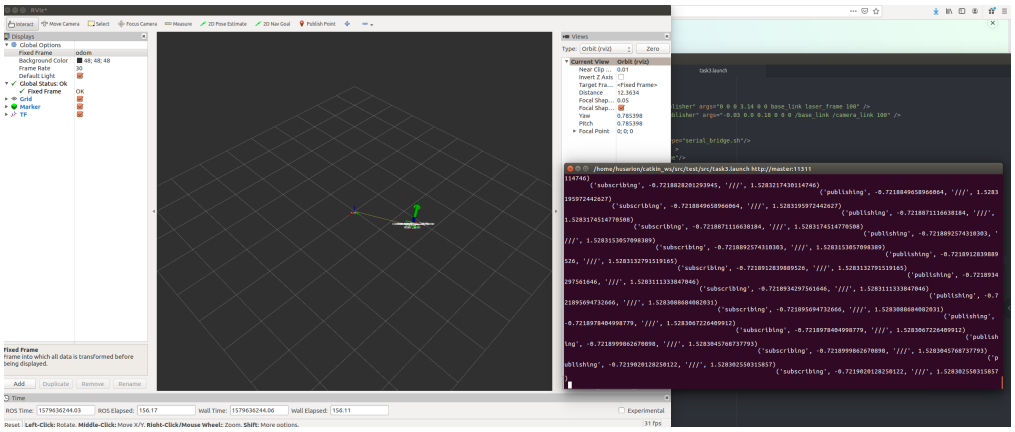
- Results are given as shown in and Fig. 3 and Fig. 4.
- You can check all the code in my git repository.



(a)



(b)



(c)

Figure 4: Visualization of robot's position on RVIZ