Lab1 Probabilistic Robotics : Introduction to Turtlebot

Shubham WAGH, Savinien BONHEUR, Nayee Muddin Khan DOUSAI
and Thomas DREVET

March 9th, 2017

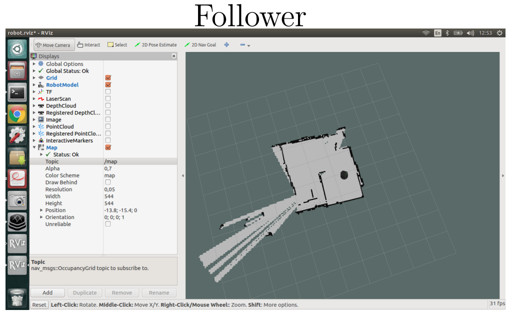
1 Introduction

The aim of this lab of Probabilistic Robotic is to start to be familiar with the Turtlebot environment (virtual and real world). It is very important to have knowledge about this topic, because it will be present in all our future lab works and during the final project. In this lab, we have a first part where we had to 'play' with the functions that already exist for the Turtlebot (mapping, follower, panorama...) and in the second part, we had to create a small code in order to make the robot to navigate to a location defined by coordinates. We won't talk a lot about the first part because it's not very interesting, so we will more focus on the navigation code.

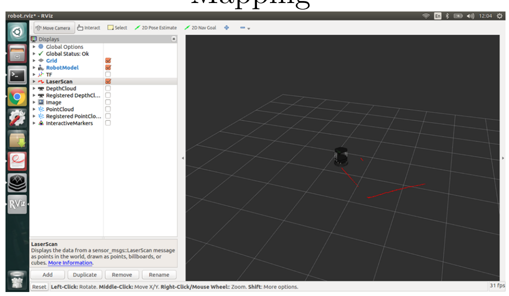
2 Playing with the Turtlebot

As we said in the introduction, the first part of this lab was to 'play' with the integrated functions from Turtlebot, in order to have an idea about what we can easily do using it. As all the other group, we had a lot of troubles to connect our laptops to the robot because of problems linked to our machines to the robot 'wifi' and problems due to the ssh (the system that permits to take the control of the laptop present in the Turtlebot). During the around 3 hours of lab, we were able to test different functions: we succeed to use the keyboard movement control, use the Kinect to map a small area and use the follower function of the robot (we add some screenshots of what we obtained after we used those functions at the end of this part). From the guidelines, we just have some trouble to use the autonomous navigation function using rviz because our first map was not really precise and then, we couldn't use the 'panorama' functions (functions that permits to create a panorama of what the robot sees, by rotating the robot and joining the image captured by the Kinect) because the library was not installed either in our laptops or in the Turtlebot. We will try to do those two missing parts, if we have some time to continue this lab, in the next session on March 16th.

Follower



Mapping



Detection

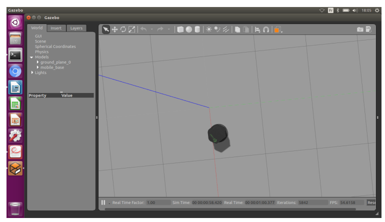
Figure 1: Playing with the Robot

3 Navigation Code

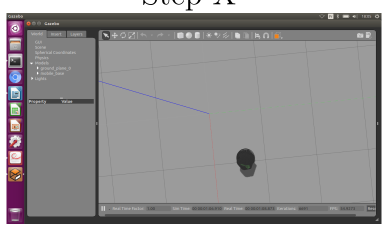
Now, we will discuss about the homework we had to do for this lab: the coordinates navigation code. The aim of this code is to send the Turtlebot to the location we want by giving it the coordinates of this location. The work we had to do here was to add the missing lines of code in the python file (driver.py), which contains the actions to do. Along the missing lines of code, there are the publisher, the subscriber and the velocity message, that are the three main part of that file (create the link between the control and the action part and the message that contains the different velocities (linear and angular). We had also to modify two functions from this file to in order to read correctly the position of the robot and compute the velocity that we have to send to the Turtlebot. To know the position of the robot, we use the pose messages that we convert to have usable ones (x, y and theta). After that, to have the velocity to send, we compute each time we re-compute the angle to the goal and from it, get the new angle and adapt the speed from where we are and where we go.

When we had changed this python file, we test it using the Gazebo Turtlebot which is a virtual environment for testing codes made for Turtlebot (we installed this tool and tested it during the pre-lab).

To run our code, we have to run the Gazebo application first using the same command line as in the prelab (**roslaunch turtlebot_gazebo turtlebot_world.launch**) and then launch the launch file that contains the python file for this lab, followed by the coordinates (**roslaunch lab1_turtlebot driver.launch x:=1 y:=1 theta:=0**, for instance). Here is some screenshots from the simulation and the code running, and also the rqt_graph that shows the communication (publishers, subscribers and messages) happening during the process.



Step X



Step X+n

Figure 2: Simulation

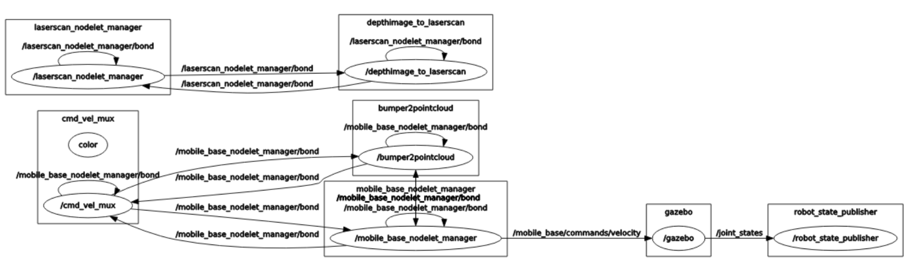


Figure 3: rqt_graph

Optional Part

We also did the optional work, that was to be able to give as input a text file containing the coordinates to reach by the robot. To do it, we modified the load_goal function from driver.py, to be able to read a text file, if a parameter is given as input for the 'fname' variable (this variable should contains the path to get this file). We will use the function loadtxt to extract the data from the file and we will use the same strategy as the 'normal' function to store the destination. We have also to modify the launch file to add another input parameter for the launch file.
To run this new launch file, we have to do as the first one, by running Gazebo first and then, launch the modify launcher by replacing the 3 parameters by fname:="pathofthetxtfile", as shown after, for instance.

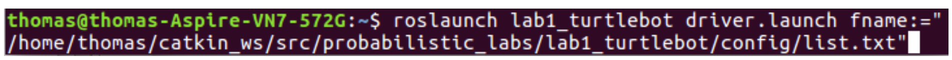


Figure 4: Example of command for the optional work

4 Conclusion

This lab permitted us to be more familiar with the Turtlebot environment by using the Gazebo simulation and the real Turtlebot, even if due to bugs with the robot connection. We did the work with the navigation code that "drive" the robot to the location we want, by simply giving the coordinates of the location or via a text file.