# Probabilistic Robotics
# Lab 1 Report: Introduction to Turtlebot

**Mohammad Rami Koujan**
M.Sc. VIBOT
University of Girona

March 8, 2016

## 1 Introduction

During this lab session, I started to work on the turtlebot by firstly setting the network configuration required to connect properly to this robot, using some of the provided packages, e.g. Navigation, mapping, and follower, and finally, implementing a driver node to move the turtlebot to one or set of goals.

## 2 Testing the turtlebot

The procedure started by choosing a turtlebot among the available ones. The next step is to connect to this robot's wireless network after setting the ROS_MASTER_URI and ROS_HOSTNAME parameters. To obtain access to the robot laptop terminal, an ssh connection should be established, which allows controlling the robot remotely. At this stage, an essential thing to be done is to bring up the turtlebot by launching the minimal file.

Among the tested packages were the teleoperation, navigation and mapping. In this part, I activated the gmapping and started to control the robot by sending velocity commands using the keyboard_teleop in order to build a map. Figure 1 shows the built map inside the lab, which was captured from turtlebot_rviz_launchers program.

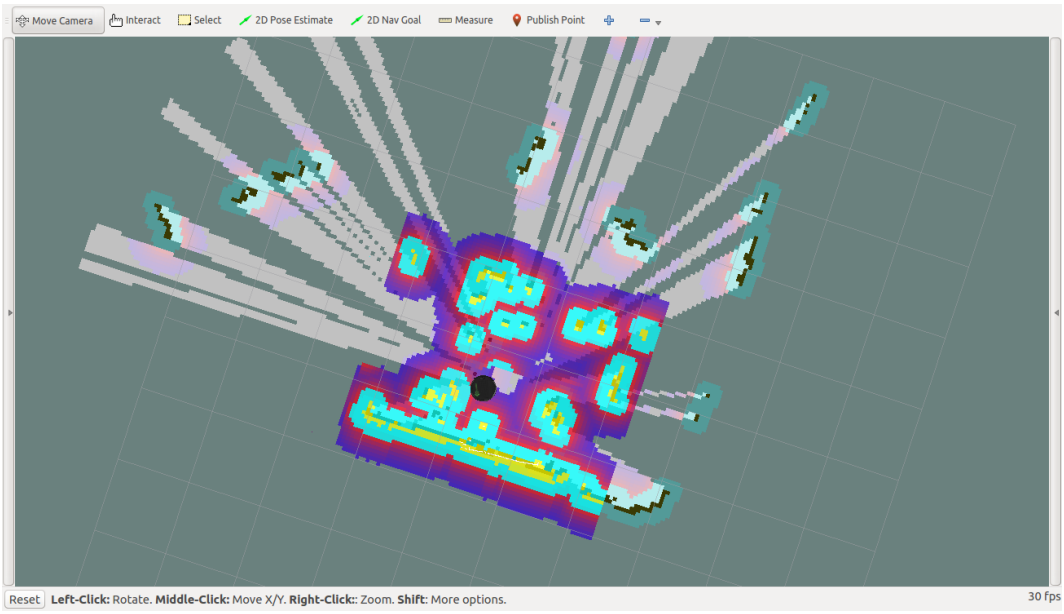



Figure 1: Gmap navigation

To let the turtlebot navigate autonomously on this map, a map saving step on the map server was performed. Then, the amcl_demo.launch was used to trigger this feature. What I noticed is that to navigate more accurately the map should be built more precisely by moving in as many directions as possible, which obviously helps the robot in deciding next control command to achieve. Figure 2 presents the autonomous navigation carried out inside the lab.

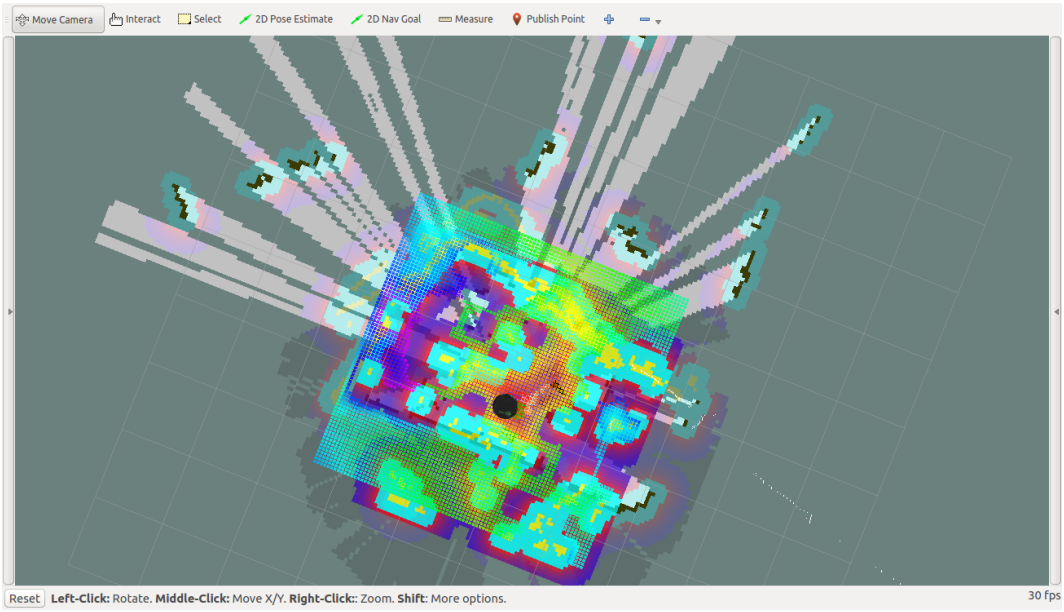



Figure 2: Autonomous navigation

Another available package with the turtlebot is the follower. To start this package the follower.launch file must be run. After that the idea is to stand in front of the robot and move. The turtlebot, in its turn, follows the person on condition that the person is not moving quickly. All of the previously mentioned packages, in fact, were also tested by me in the gazebo simulator, except the panorama because it is not possible to do so in the stated simulator.

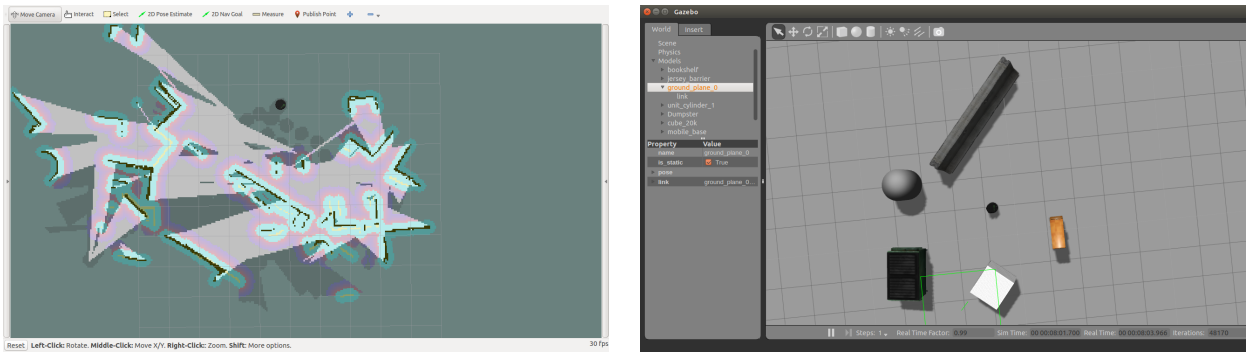
# 3 Turtlebot driver

The last task is to guide the turtlebot to a preset goal(x,y,$\theta$). To accomplish this task, first the topics and type of messages should be determined. The turtlebot, indeed, publishes odometer messages on the 'odom' topic. Therefore, it is necessary for the implemented node to subscribe to this topic. The odometer messages include the pose and orientation of the robot. However, since the orientation is measured in quaternion angles, a conversion function, tf.transformation, from quaternion to euler angles should be used. This function takes orientation parameters (x,y,z,w) in quaternion unit as input and gives Roll, Pitch and Yaw as output. Yaw is used as $\theta$ and x and y positions are extracted from the odom.pose messages. In order to send the velocity commands to the turtlebot, twist messages should be published on the '/mobile_base/commands/velocity' topic, to which the turtlebot is a subscriber.

Subsequently, in the read_position function, the position received in the message (msg) is copied to the internal class variables self.position_x, self.position_y and self.position_theta with taking into account the previously mentioned conversion step from quaternion to euler angles. In contrast, the compute_velocity function is used to compute the velocity message to be published. This was done by setting two important parameters. The first one is the angular velocity parameter z which is set to the difference between the angle calculated based on the current and goal positions and the present orientation of the turtlebot. The second one is the linear velocity parameter x which is set to a fixed speed, chosen experimentally to be 0.3, if the angular velocity parameter z, set previously, is greater than a fixed value, chosen experimentally to be $\frac{\pi}{12}$. Otherwise, x is set to the euclidean distance between the current and desired positions.

One important thing to mention is, if the linear speed variable x is only set based on the euclidean distance then for a little far distances from the goal the linear velocity will be high and along with continuous rotation and linear speed the turtlebot will enter an infinite loop of rotation around itself. Thus, the advantage of this method is the quite good balance between speed and robustness. This means we can adjust the $\frac{\pi}{12}$ limit to a higher value but at the cost of more drifting, less robustness. Another feasible implementation is to always set the linear velocity parameter x to a factor multiplied by the mentioned euclidean distance. Nevertheless, this method provides less balanced approach between the speed and robustness.

Furthermore, to move the turtlebot to a series of goals, as required in the optional part of this assignment, the load_goals function will load the 'file' parameter from the server and then fetch the corresponding text file and store the goals associated with this file inside the x,y,$\theta$ arrays. Thereafter, the next_goal function will increment the index of the goal, self.active_goal, and check whether or not the robot has reached the final goal by checking the active_goal parameter if it is greater than the size of the goals array or not. Figure three shows two photos for driving the robot to a set of points and the map built during this movement.



(a) rviz launcher map

(b) gazebo simulator map

Figure 3: Test images

# 4 Conclusion

In this lab session, as an introduction to the turtlebot and to ros launch system, a number of turtlebot packages were tested on a real robot and on gazebo simulator. Moreover, a driver node were implemented in python programming language. This node is responsible for driving the robot to a preset goal or to a set of goals taken from text file. It is noteworthy that configuring the network setting in an improper way will prevent the robot from receiving any commands. Therefore, attention should be made while doing this step. Additionally, some of the packages,mapping and navigation, demand high bandwidth due to the type of data being sent. Hence, it is recommended to run these packages directly on the turtlebots pc, not via ssh.

# References


[1] http://wiki.ros.org/.

[2] https://docs.python.org/2/.