PROBABILISTIC ROBOTICS

Lab report 2: Introduction to TurtleBot

Author: **RAABID HUSSAIN**

Objective:-

The objective of the lab-work was to get familiarized with the Turtlebot 2 and the ROS network configuration. Turtlebot is a multi-purpose robotic platform used in different applications. The objectives can be divided as:-

- configure the turtlebot with our laptops
- tele-operate the turtlebot
- create map of the robotics lab (small area)
- perform autonomous navigation
- run the object follower demo
- write a code to drive the robot to a particular location

Methodology:-

The lab started with configuring the laptop with the turtlebot. The given steps were followed without aany problem however, there were only 5 turtlebots available for the students which were lesser than the students. This ultimately led to them sharing a turtlebot amongst two. This caused the turtlebot to sometimes stop working when accidently two students were connected to the turtlebot network. After configuring the turtlebot, the laptop was connected to the platform using 'ssh' and rostopic list was used to check whether the connection had been established properly or not.

The next thing was to check the turtlebot's status using bringup command. However, due to low bandwidth of the connection, this was constantly throwing an error. So, this command (along with many others) had to be run in the robot's laptop rather than our laptops. The robot was tele-operated next using the inbuilt command.

Using teleoperation, we were next to roam the robot across the lab and create a map using gmapping demo launch file. Due to some obstacle and the lab being too crowded, the entire lab could not be mapped, so only a small portion of the lab was mapped as shown in Figure 1. The areas in purple represent that the robot is too close to the walls, indicating to move away.

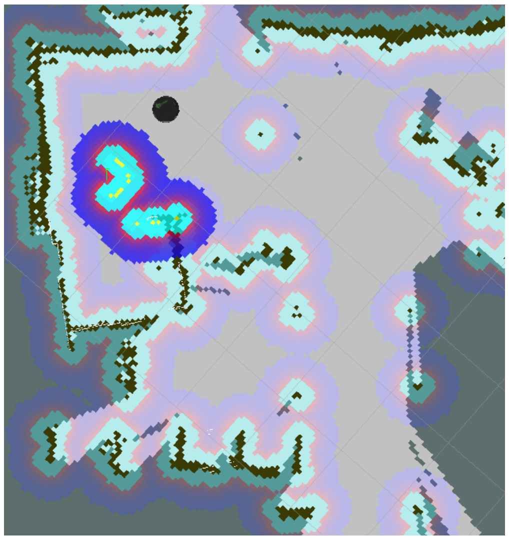
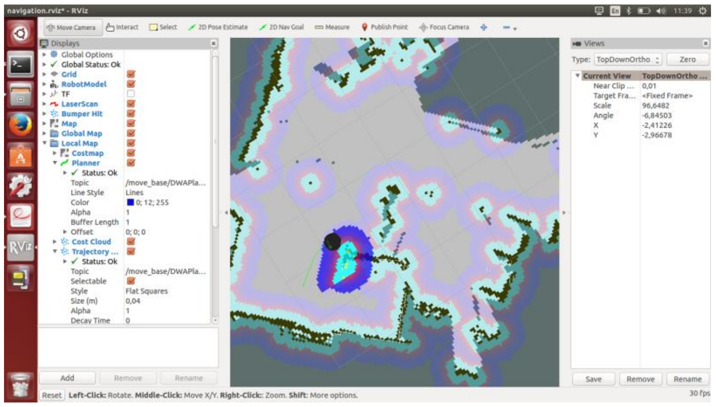


Figure 1: Map of the lab created by turtlebot

After creating and saving the map, autonomous navigation can now be performed. For this tele-operation has to be closed as it would then interfere with the navigation process and the robot ultimately does not move. Using the pose estimation, we were first to initiate the starting position of the robot and then using navigation goal declare the ending position. In the navigation window, the robot can now be seen to be calculating the optimal trajectory for reaching the goal position. Figure 2 represents this stage. The turtlebot then moved to the goal position.



```
Global Options
Global Status: Ok
Grid            ✓
RobotModel      ✓
TF              ☐
LaserScan       ✓
Bumper Hit      ✓
Map             ✓
Global Map      ✓
Local Map       ✓
  Costmap       ✓
  Planner       ✓
    Status: Ok
    Topic          /move_base/DWAPla...
    Line Style     Lines
    Color          0; 12; 255
    Alpha          1
    Buffer Length  1
    Offset         0; 0; 0
  Cost Cloud    ✓
  Trajectory ...✓
    Status: Ok
    Topic          /move_base/DWAPla...
    Selectable     ✓
    Style          Flat Squares
    Size (m)       0,04
    Alpha          1
    Decay Time     0
```

```
Type: TopDownOrtho
Current View   TopDownOrtho ...
  Near Clip ...  0,01
  Target Fra...  <Fixed Frame>
  Scale          96,6482
  Angle          -6,84503
  X              -2,41226
  Y              -2,96678
```

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click: Zoom. Shift: More options. 30 fps

Figure 2: Autonomous navigation in progress by the turtlebot

The next thing was to run the turtlebot follower application. After running that I stood in front of the robot and nothing happened. However when I used the good old computer solving technique of restarting, it started working and automatically detected me and started to follow me. However, if you move too fast then it loses you and stops. Also, it can be a challenging task to try to stop it as it keeps on following you so you have to place another object in front of it to stop it from following you. Next we were to create a panorama of the environment. However the time of the lab finished and I had to stop here, however I managed to take a snapshot of the output as shown in Figure 3.

In order to run the remaining part, we used gazebo to simulate everything. In order to test the setup, I re-ran the autonomous navigation in the simulator and it worked fine. A snapshot of it is shown in Figure 4.

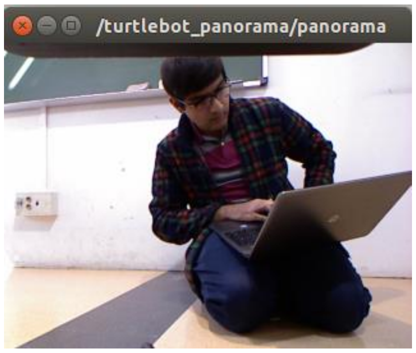


Figure 3: Panorama view from turtlebot

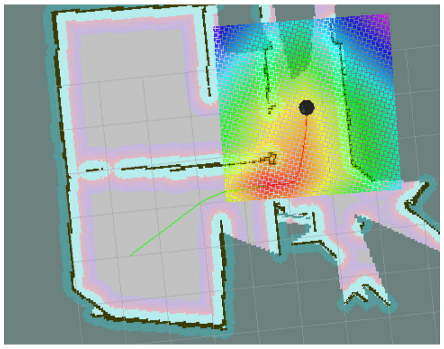


Figure 4: Autonomous nvigation in simulation mode

Next, we had to edit the driver.py file to make the robot go to a particular location. This was very similar to the task in lab0. The current position and orientation of the robot was found using /odom topic. This odometry message contains 7 parameters that can be used to find the position and orientation of the robot. For this tf.transformation library was used. After getting the current position, it was merely a copy paste of the previous lab assignment. I used the same system as the previous lab: rotating first to face the target and then simply going straight to the goal, upon reaching the robot turns to match the angle criteria. The code has taken care of going to multiple goal locations which are read from a separate file as input.

However, my turtle used to move in spiral manner and the position parameters of the turtle would totally change (not match with the coordinates in the gazebo rviz window) whenever the input arguments were more than 4. This is a strange problem that occurred as when I checked the input to the robot, it was saying zero. After running thorough tests using rqt_plot, it was found that if the linear velocity is given more than a certain value, the robot starts to rotate as well and the coordinate system corrupts. For this reason, the linear velocity has been fixed to 0.15 units. The rotational velocity has also been kept very low that is why sometimes the turtle appears stationary but it actually is rotating (be patient when running the code).

Also, as there is no check for checking obstacles in the way, so if there are obstacles in the way, the robot hits them and runs in a strange trajectory. Another problem encountered with gazebo is that if we try to check the position and orientation of the robot, we see a slight change taking place in the robot's orientation. So the simulator is not perfect. To compensate this situation, I increased the threshold for reaching goal. Also it is apparent that the final orientation of the robot would depend on how the student has calculated the angle for the robot.

<u>Conclusion:-</u>

In this lab-work, the basic knowledge of turtlebot was gained and a small program successfully implemented that used the gazebo simulator in ROS to move the turtlebot to the desired location. An autonomous navigation application was also tested along with the follower application of the turtlebot. The lab was really help in understanding the key concepts.