Autonomous Robots

Lab 3 : Part 3 - Mapping, planning and controlling with the Turtlebot

Shubham Wagh
Hassan Saeed
Nayee Muddin Khan Dousai
Anirudh Puligandla

Dated : May 7, 2017

Escola Politècnica Superior (UdG) - VIBOT

1 Introduction

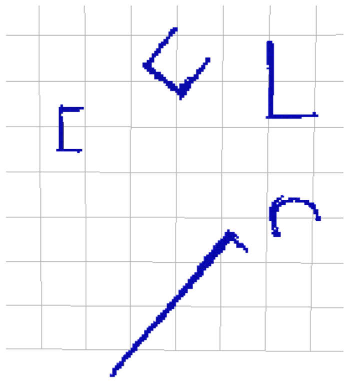
This document deals with the mapping, planning and controlling with the turtlebot. Here we prepare the Turtlebot in a simulated environment to map the environment and plan collision-free paths. We also check the performance of these systems.

2 ROS Dependencies and Package installation

Before continuing with this lab we had to make sure we had required dependencies installed. Therefore *sudo apt-get install ros-kinetic-ompl* command was run in the terminal to install dependencies such as octomaps, octovis etc.
Further on, stated repository was pulled using git pull command, followed by catkin_make.

3 Checking vechile's capability for building a single map

As stated in the lab requirements doc, we used a simplified version of the map.Therefore we created a 2D map rather than utilizing the 3-dimensional (3D) map.To achieve this, Octomap_server was using laser scan of the kinetic camera.
Turtlebot simulator and teleoperation was initiated in one terminal whereas in another terminal mapping, planning and control systems was executed.
Then using the first terminal tutrtlebot was moved around via a keyboard to generate a map inclusive of any nearby obstacles encountered.After achieving this robot navigation built map was saved using octomap_server command.In the diagrams below it can be seen the obstacles encountered in orange after robot navigation around the map.


(a) Octomap showing obstacle visualized using octovis

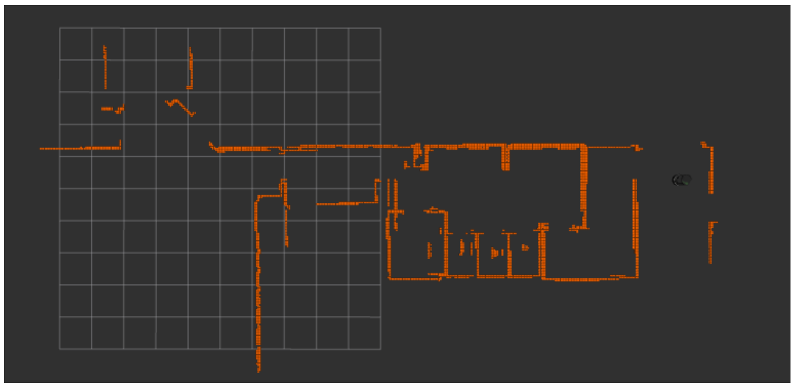
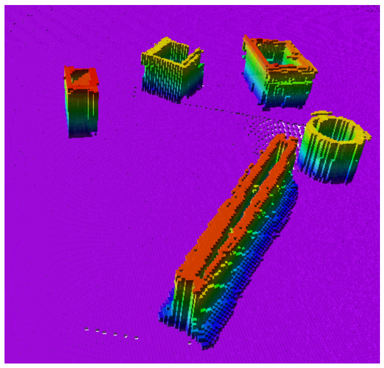

(b) Octomap of willow map in RViz

Figure 1: 2D generated Octomaps

Next step was to verify that turtlebot' generated 2D was in coherent according to the Gazebo simulated environment.Octovis command was used to visualize the the generated 2D map with the previously generated 3D map in simulation.

Following this we had to verify this generated 2D map with the previously generated 3D map.In the diagram it can be seen that the in 2D obstacles are detected and represent by orange color whereas in 3D environment obstacles are detected and represented by different z values encoded with different colors.


(a) 3D Octomap showing obstacles

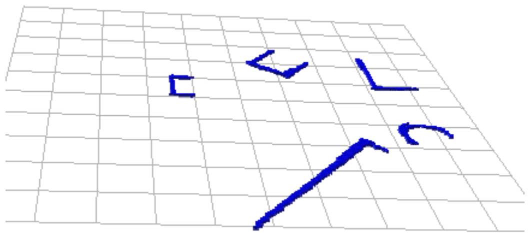
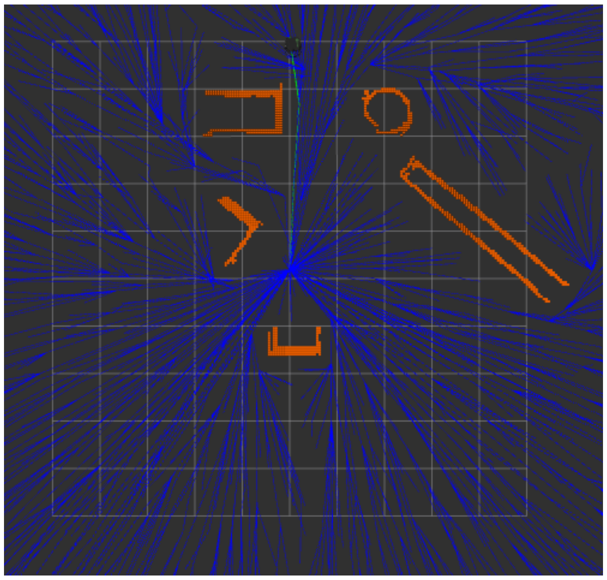

(b) 2D Octomap showing obstacles

Figure 2: Octomaps comparison

4 Planning a collision-free path using Octomap

For this part of the lab we used the Open motion planning library.This library is vastly used for querries related to motion planning.It is composed of several sampling-based algorithms such as RRT, RRT*, PRM and etc. We used the RRT* and PRM sampling based algorithm for collsion-free path planning.Diagram below shows the path generated using the RRT* and PRM.


(a) Path-planning using RRT*

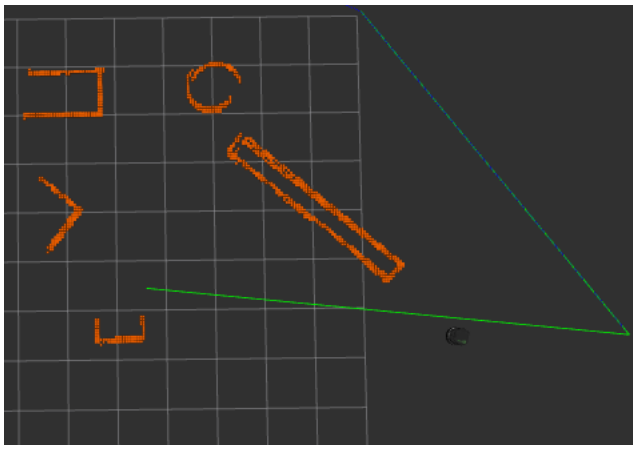

(b) Path-planning using PRM

Figure 3: Path-planning on Octomaps

In another opened terminal **rosservice** call was initialized for turtlebot to find the path to goal.This is depicted in the diagram attached.In the diagram it can be observed using RRT* ap-

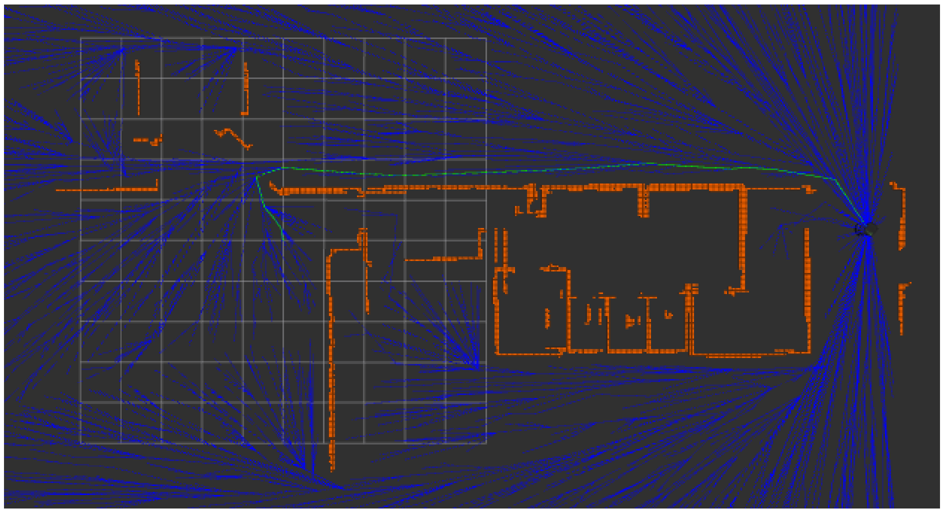


Figure 4: Path-planning on willow map, optimized trajectory generated from start to goal

proach random paths were generated from start to goal.Blue lines represent all these random paths generated from start to goal.Optimal path between start and goal position is shown in green. Then by stating the goal state x and y parameters following rosservice call robot was navigated from it's current position to it's goal position.

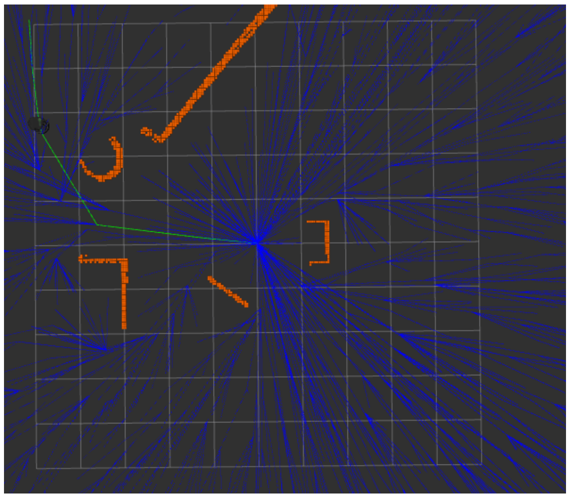


Figure 5: Turtlebot following the optimized trajectory. start: (0.0, 0.0), goal: (-5.0, -5.0)

5 Analysis

1. **Planner and map consistency** - When the turtlebot follows the path in the map, the map incorporates the new obstacles in the map which were not mapped during mapping stage. But sometimes after incorporating the new obstacles, turtlebot gets stuck at the obstacle or wall and this can be seen in gazebo whereas in RViz, we see that it passes through the newly mapped obstacles and follow the path till the goal point. This may be due to less amount of explored areas, as robot tries to localize itself while building an octomap. It becomes

more aware about the environment if more information is mapped by the robot. Considering if less information is mapped in RViz and if the optimized path generated is through the unexplored path, then the robot might not be able to reach the goal point and may get stuck at the obstacle.

2. **Repeatability** - Also the map can be used for a second start-to-goal query after reaching the first one, with the existing map. Hence repeatability is quite possible.

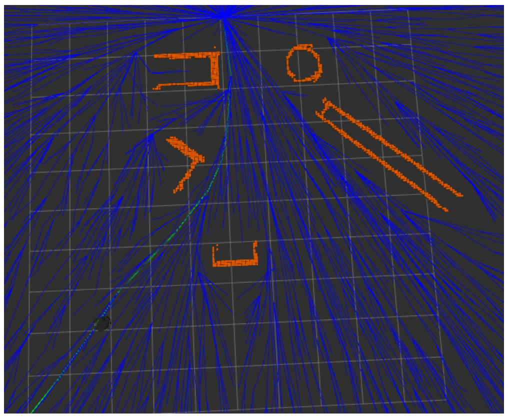
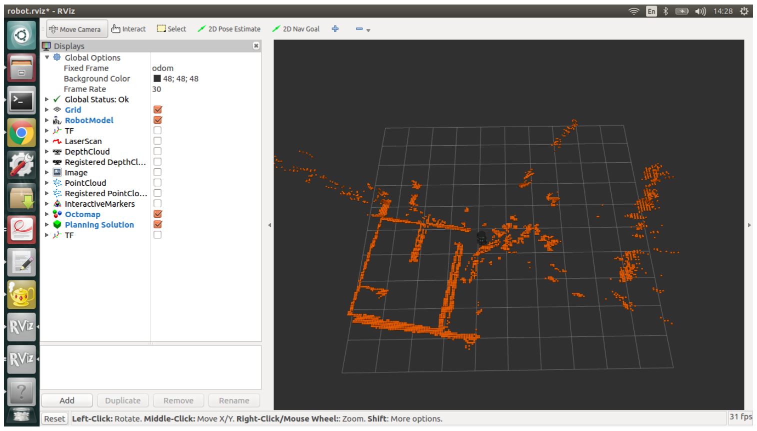


Figure 6: Repeatability in action

3. **Mapping with real Turtlebot** - The above described procedures are also applied to real turtlebot. We were able to map the environment with the real turtlebot and the 2D octomap was saved for its planning in simulator. Also it was planned in real time environment



```
▼ Global Options
    Fixed Frame          odom
    Background Color     ■ 48; 48; 48
    Frame Rate           30
▸ ✓ Global Status: Ok
▸ Grid                   ☑
▸ RobotModel             ☑
▸ TF                     ☐
▸ LaserScan              ☐
▸ DepthCloud             ☐
▸ Registered DepthCl...  ☐
▸ Image                  ☐
▸ PointCloud             ☐
▸ Registered PointClo... ☐
▸ InteractiveMarkers     ☐
▸ Octomap                ☑
▸ Planning Solution      ☑
▸ TF                     ☐
```

Figure 7: 2D Octomap using real turtlebot

6 Conclusion

This lab was overall a good exercise for proper understanding of mapping, planning and control both in virtual and real time environment.This lab gave us the vital and basic understanding for the coming Autonomous Robotics project.