Map My World

Siddharth Kothiyal

Abstract—Mapping of an environment is essential, whether one is trying to perform SLAM in real-time or trying to do just do any form of localization on an already mapped environment. This project uses Real Time Appearance Based Mapping (RTAB-Map), based on GraphSLAM, for creating 2D occupancy grids and 3D octomaps of environments using RGB-D camera and LIDAR senor.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Mapping.

1 Introduction

MAPPING of the environment is key for any autonomous or semi-autonomous robot. Whether the robot is performing Simultaneous Localization and Mapping (SLAM) or performing localization on an already mapped environment, it must be able to keep track of useful features in the environment and later be able to identify them, to know where it is. Mapping the system also makes the robot more versatile and gives it the ability to understand a dynamic environment.

In this project, a robot equipped with a RGB-D camera and a LIDAR sensor is tasked with mapping a simulated environment, and create 2D occupancy grids and 3D octomaps of it. Furthermore, another simulated environment is made and then mapped using the same process, to capture what are some of the key features that need to be kept in mind while mapping an environment.

2 Background

To perform localization, the robot must have a map of the environment, so that it can match the features in its surrounding with the features in the map. Mapping on the other hand is much tougher, as it requires the robot to be aware of its location and orientation to a certain extent, so that the robot can map the features it observes in its surroundings to a location on the map being created.

Proper mapping of the system is required, so that later when the map is being used as a resource, the robot does not misidentify its location and take a faulty action according to it.

Mapping requires both feature understanding of the immediate surrounding of the robot, as well as proper pose estimation of it. Faulty and noisy sensors are a big hurdle in mapping, as mapping of large environments happens in several steps, if a small error creeps up in each step, with each step the errors gets accumulated and map becomes more erroneous. Hence, a good mapping algorithm should have the ability to take corrective measures when errors start getting accumulated.

There are two types of SLAM, Online SLAM and Full SLAM. Online SLAM uses only the present measurements to estimate the robot's pose and map. Whereas, Full SLAM takes the robot's complete trajectory into account while estimating the pose and map.

There are several types of mapping algorithms, like Occupancy Grid Mapping, Grid-based FastSLAM and GraphSLAM.

2.1 Occupancy Grid Mapping

In this approach, the whole environment is divided into an equally spaced grid, where each cell is either occupied or empty. This method assumes the robot pose is known, and gives the other sensors some margin for error, as instead of pin pointing the exact location of the object, the algorithm is tasked with finding if the obstacle is present somewhere in the cell. It also allows the robot to control the resolution of the map, and hence the computational power required in mapping can be controlled.

This algorithm is mainly used for post processing, as several other algorithms do not create maps on which tasks like planning and navigation can be performed.

2.2 Grid based FastSLAM

FastSLAM uses particle filters like the Monte Carlo Localization (MCL) algorithm to approximate the trajectory of the robot, where one particle approximates the robot's at one step. Hence it is a Full SLAM technique. Along with that, it uses Extended Kalman Filter (EKF) to estimate features of the map in the form of a Guassian distribution. FastSLAM suffers in random environments as it assumes the landmarks are known.

Grid-based FastSLAM is using the particle filter approach for trajectory estimation along with Occupancy Grid Mapping, to create a grid with each cell representing if it is occupied or empty.

2.3 GraphSLAM

This is another Full SLAM technique. It attempts to create a graph of the entire trajectory of the robot from its motion, where each node is the pose of the robot at a certain time, and edges between the nodes represents movement from last node to the current. GraphSLAM retains information from locations encountered in the past, and attempts to align the nodes when it encounters the feature again, hence improving the accuracy of the map with each round of mapping.

This project uses RTAB-Map, which is an implementation of GraphSLAM. It takes pictures with camera at every step and attempts to find the match in images using SURF features, to do loop closures by aligning nodes over several runs.

3 Scene and Robot Configuration

RTAB-Mapping was performed on two sets of simulated environments using the SuperBot (the robot created in the Localization project). Gazebo was used for simulation purposes and RViz was used to see realtime creation of the 3D octomap. ROS framework was used for communication between the different nodes, and the SuperBot was teleoperated around the area to perform mapping.

The robot used for mapping was SuperBot. It was equipped with a LIDAR and a RGB camera module. The RGB camera module was replaced with a RGB-D camera module to provide 3D sensing capabilities to the robot. The udacity_bot, used in the localization project was not used, as the positioning of the camera was too close to the ground, and the ground would take up around half of the frame each time. In the SuperBot, the camera is placed at the chest height of the bot, providing it a better field of view. The shoulder and arm part of the SuperBot is the widest part of the robot, and hence the LIDAR sensor was placed there, so that any obstacle at that height can be easily identified.

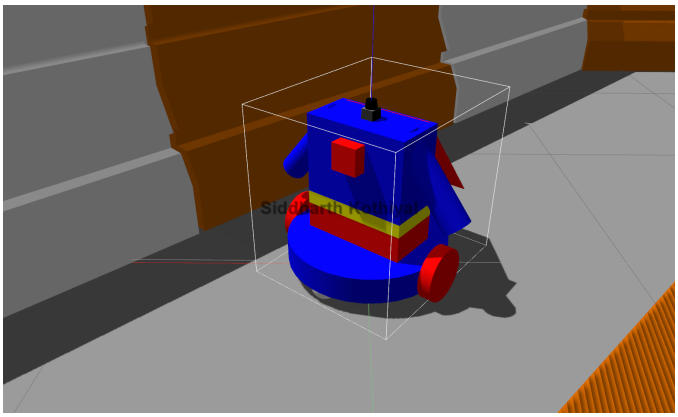

Fig. 1. SuperBot

The first simulated environment which was provided, was the kitchen and dining scene available in the Gazebo simulator.

The second simulated world was created after several iterations, by trial and error technique. First an open world was tried with several objects placed all around the map, though this helped with the feature identification part of RTAB-Map, as there were no erroneous feature matching being done with the background wall, but the overall octomap was not looking very clean and the occupancy map was a little muddled. Adding walls to this environment was helpful, but then the pattern in the walls would be misidentified by RTAB and it would match it the same pattern in the wall on the other sides, hence messing up the mapping. The open map was then replaced with the cafe scene, with several artifacts placed all around to increase the number of identifiable features for RTAB. The floor of the

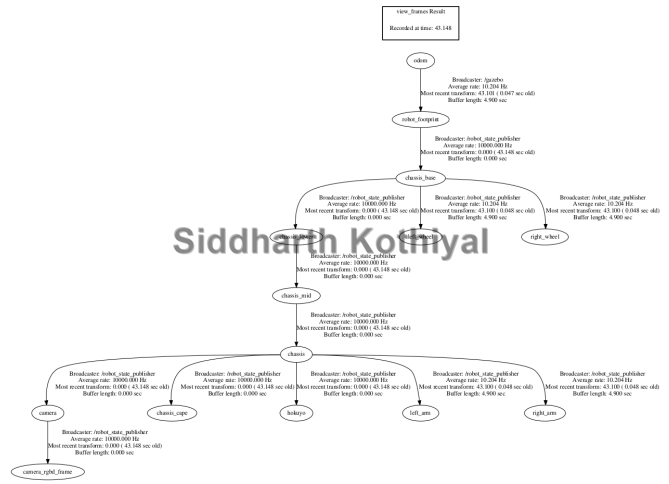

Fig. 2. SuperBot tf view_frames

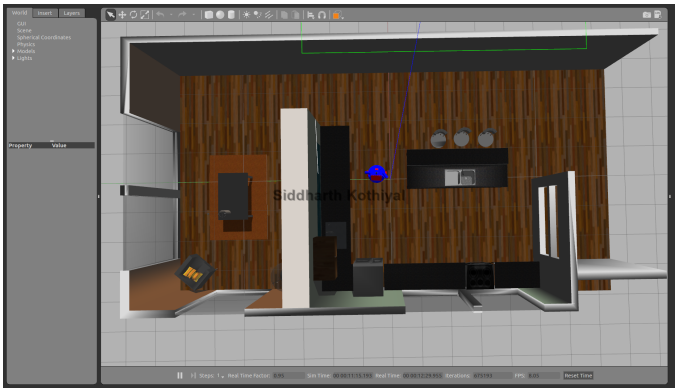

Fig. 3. Kitchen and Dining

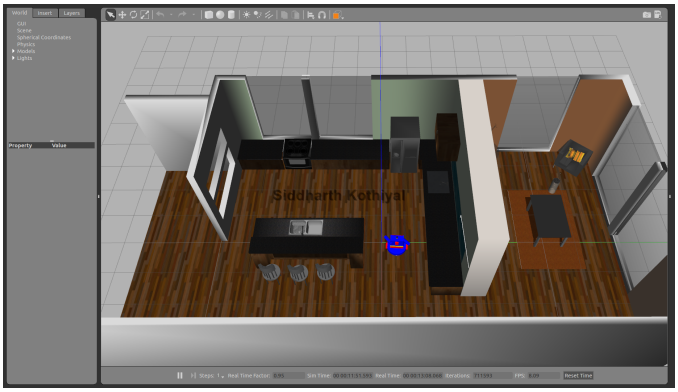

Fig. 4. Kitchen and Dining

cafe was then modified, as the zigzag pattern of a part of the floor was responsible for excessive number of features being identified and hence taking away focus from the features present in the kitchen area. The kitchen area of the Cafe was not mapped in the end as SuperBot was too wide to pass through the door.

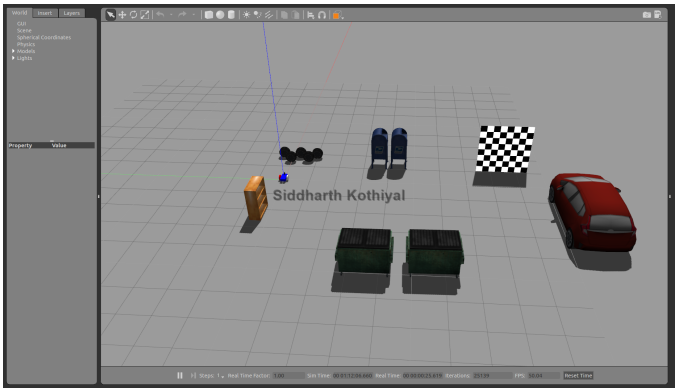

Fig. 5. Open world trial

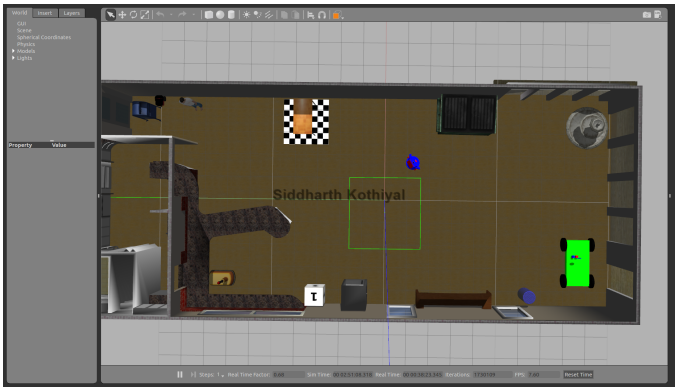

Fig. 6. Created Simulation Environment

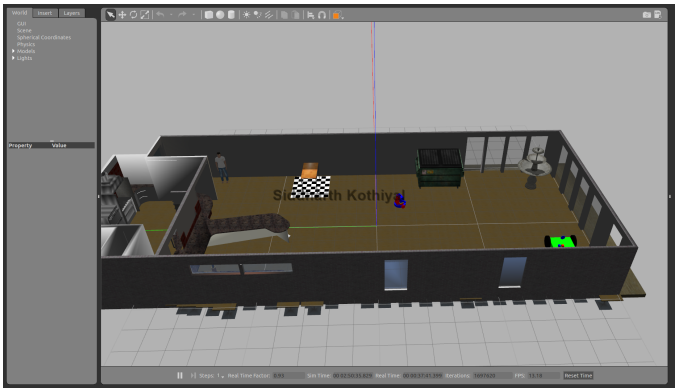

Fig. 7. Created Simulation Environment

3.1 Results

The kitchen and dining environment was looped around once, at a very slow speed, with the robot frequently being made to rotate on its spot, so that it is able to identify

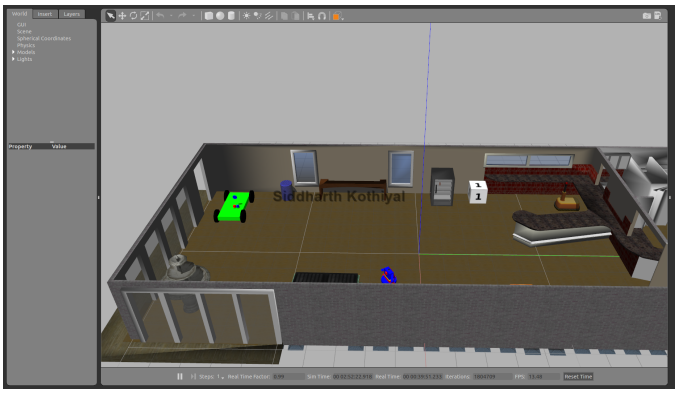

Fig. 8. Created Simulation Environment

features again. Since it was a small environment and not very complex, the mapping was pretty accurate even in one run around the whole environment. (The mapping of kitchen and dining scene was done with two runs around the whole environment, but no significant difference in the mapping was observed other than the significant increase in loop closures.)

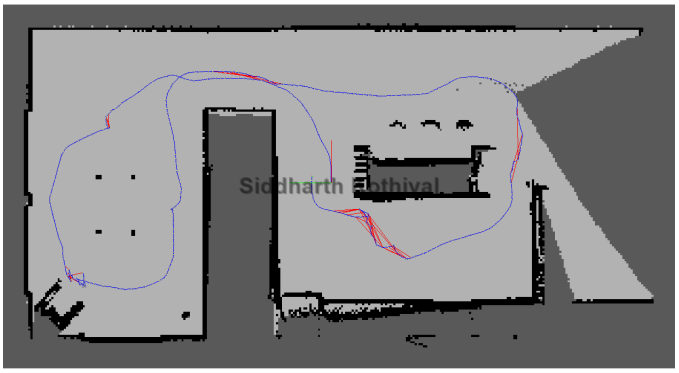

Fig. 9. Map created by RTAB

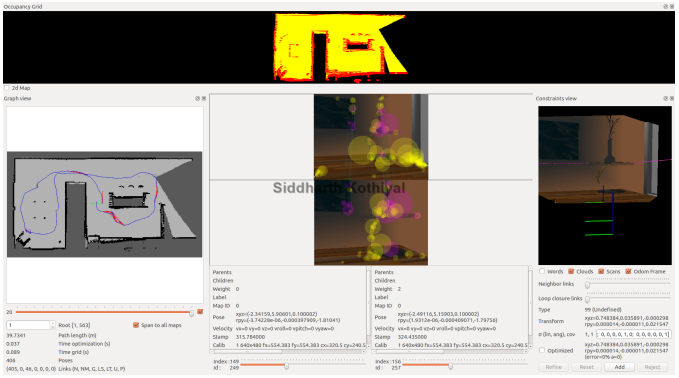

Fig. 10. RTAB viewer kitchen, loop closure detected, with occupancy map

In case of the Cafe environment, the environment was looped over twice, again at the slow speed. The map became significantly more accurate after the second loop, and the generated was adjusted when loop closures were detected.

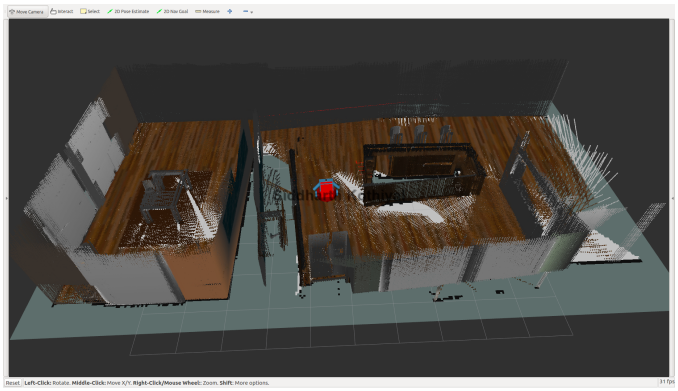

Fig. 11. 3D map created on Rviz

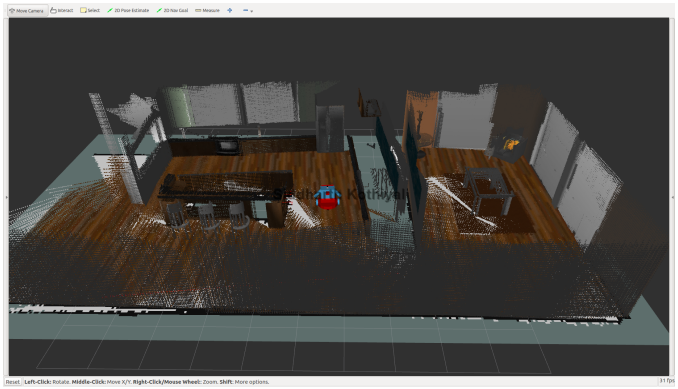

Fig. 12. 3D map created on Rviz

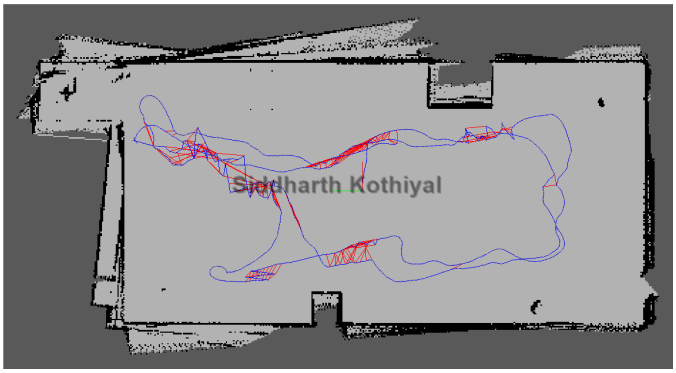

Fig. 13. Map created by RTAB

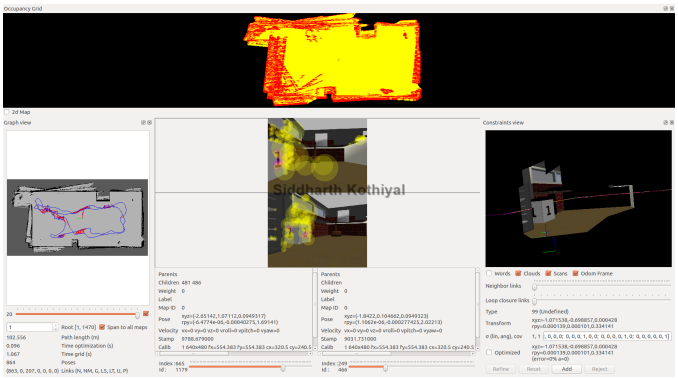

Fig. 14. RTAB viewer cafe, loop closure detected, with occupancy map

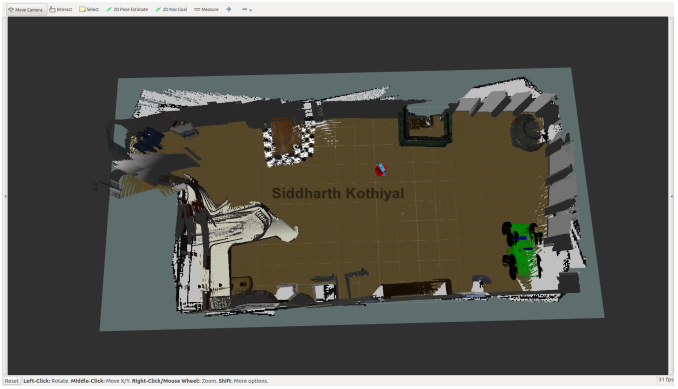

Fig. 15. 3D map created on Rviz

Fig. 16. 3D map created on Rviz

4 Discussion

Speed and FPS at which the environment was being simulated played a significant role in the mapping process. If the speed was too fast, the robot would move ahead too fast and miss some features, if the FPS was too high, the robot would take the teleop command more frequently and end up moving too far ahead.

The mapping of kitchen and dining scene seemed more accurate. This fact can be attributed to the small size of the environment, as well as the environment full of a variety of objects with non-similar features, i.e, there were very few instances, where RTAB-Map was showing highest hypothesis from a different area in the environment. Also there were no Hessian features (used by SURF, which was used in this mapping scenario) detected on the wall or the floor.

During the mapping of the self created world, a lot of issues were faced. When similar colored objects with even slightly similar shapes were placed around the map, there were a lot of incorrect loop closures and that would completely mess up the generated map. If the walls had patterns, RTAB would erroneously match features from two different areas and muddle the generated map. A similar behavior was observed when floor had design with sharp corners (as SURF checks for Hessian features), it would detect a lot of feature points and would perform loop closures, just on the basis of the features on the floor. The generated map was very uneven, when objects were placed without any boundary walls around. Finally, the cafe scene was selected for mapping, it was made sure that

similar colored objects are either in completely different surroundings or not present at all, so that the accidental loop closures are avoided. The floor of cafe was replaced with a different material, so that incorrect loop closures do not occur. Post the environment was finalized, after two runs around the environment, the mapping seemed pretty accurate and localization was also accurate.

5 Conclusion / Future work

RTAB-Map works very well when mapping a static environment, but since it uses features of objects for loop closures, in a dynamic environment, loop closures will end a being a bane.

Even though RTAB-Map uses 3D sensors, it processes the point-cloud to decrease the computation required, hence loosing out on a lot of key features, reducing point cloud works well only in controlled environments.

Mapping is a very useful tool in Robotics. Mapping can be done in warehouses where robots are tasked with managing all the inventory. This will allow the robot to easily locate and find a path to the location its supposed to reach, to pickup and drop-off items. Furthermore, in case of surveillance drones, outside of a house or a large area can be mapped, so that the drone can plan a path on which it will patrol; mapping the environment will also allow it to find any irregularities in the area, which it can investigate further and even detect possible threats.