Map My World Robot (RTAB SLAM Project)

Navid Safaeian

Abstract—This project conducts to successfully perform the best solution for SLAM, GraphSLAM based on developing 2D occupancy grids and 3D octomaps of two simulated environments in Gazebo. These methods derived from RTAB-Map's speed, memory management and its software tools that are available for wide range of environments. A custom two-wheel robot equipped with RGB-D and 2D Lidar sensors (configured as a Kinect camera and a Hokuyo lidar, respectively) was simulated to traverse two Gazebo world environments, one provided, and another that was custom built. The robot maps the environment using **rtabmap_ros** package, which is a ROS wrapper (API) for interacting with RTAB-Map (Real-Time-Appearance-Based Mapping) GraphSLAM approach.

1 Introduction

In the previous project, robot localization was performed based on the map already provided which is not always the case in most real-world applications. In most cases, the robot will need to navigate through dynamic environments, therefore an autonomous robot must have the ability to precisely map its environment and localize itself at the same time. Simultaneous Localization and Mapping (SLAM) provides a robot with the capability to dynamically map its environment as it relates to it. When implemented correctly, SLAM solves the localization problem to map the environment, and maps the environment to solve the localization problem. This is all done as an iterative cycle.
RTAB-Map (Real-Time Appearance-Based Mapping), which is implemented in this project, is a RGB-D Graph-Based SLAM approach based on an TAB-MAP (Real-Time Appearance-Based Mapping). The bag-of-words approach is used by loop closure detector to determinate how likely a new image derived from a previous location or a new location. After loop closure detection, a graph optimizer minimizes the errors in the map. RTAB-Map can be used alone with a hand-held Kinect or stereo camera for 6DoF RGB-D mapping, or on a robot equipped with a laser rangefinder for 3DoF mapping.
In this project, two 3D world environments are to be simulated, tested, and mapped using GraphSLAM by using rtabmap_ros package. The environment provided is called kitchen dining (as a benchmark) and another environment which is designed and developed in Gazebo is called my world.

2 Background

Mapping consists of creating a representation of obstacles and features in its environment based on data collected from robot sensors and then accurate robot location and orientation, but simultaneously the robot localization needs accurate mapping. Mapping task is harder with the existing inherent noises in the data obtained from most of the sensors. Many mapping algorithms have been designed to handle the inherent noise and operate either in small indoor or complex outdoor environments. When a robot is constructing or updating a map of its unknown environment while simultaneously keeping track of its location within it, the robot is doing a Simultaneous localization and mapping algorithm or SLAM. There are multiple challenges associate with the SLAM process such as unknown map, noisy collected data from sensors, big size of environment for mapping, perceptual ambiguity, and multiple cycles taken by robot. There are many approaches to perform SLAM such as Extended Kalman Filter SLAM (EKF), Sparse Extended Information Filter (SEIF), Extended Information Form (EIF), but two most commonly used approaches in robotics as follows:

- FastSLAM
- GraphSLAM

2.1 FastSLAM

The FastSLAM algorithm is able to solve the full SLAM problem with known correspondences. FastSLAM is able to estimate the trajectory of the robot which will allow it to map the environment concurrently using a custom particle filter approach along with a low dimensional Extended Kalman Filter. However, each particle in FastSLAM approximates the robots immediate pose, thus this technique also solves the online SLAM problem.
There are different types of FastSLAM, two of them, FastSLAM 1.0 and FastSLAM 2.0, and the other is an extension of the FastSLAM algorithm, Grid-based FastSLAM (Fig. 1).

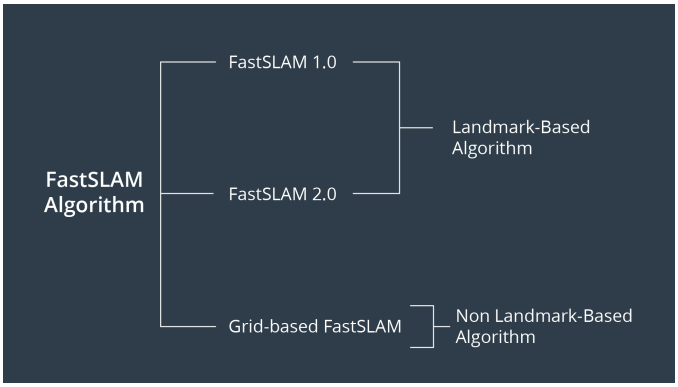

Fig. 1. SLAM types

The biggest disadvantage of FastSLAM is related to an assumption of known landmarks, thus lack of its ability

to model an random environment. This is where Grid-based FastSLAM fills in the gap. This technique keeps the FastSLAMs biggest advantage of using particle filter to solve the localization problem in SLAM, but it extends the algorithm to occupancy grid maps. Grid mapping algorithm can simulate the environment using grid maps with no need of any landmark position which solves the fundamental problem of FastSLAM.

2.2 GraphSLAM

GraphSLAM is a SLAM algorithm that addresses the Full SLAM Problem. Algorithm uses sparse information matrices produced by generating a factor graph of observation interdependencies such that two observations are related if they contain data about the same landmark. This algorithm recovers the entire path and map, which allows it to consider dependencies between current and previous poses. The advantage of this algorithm is the reduced onboard processing capability and the improved accuracy compared to FastSLAM. Because FastSLAM uses particles to estimate its pose, theres a possibility that a particle does not exist in the most likely position. GraphSLAM has a simple main structure. It extracts from a data set of soft constraints, represented by a graph as shown in Fig. 2.

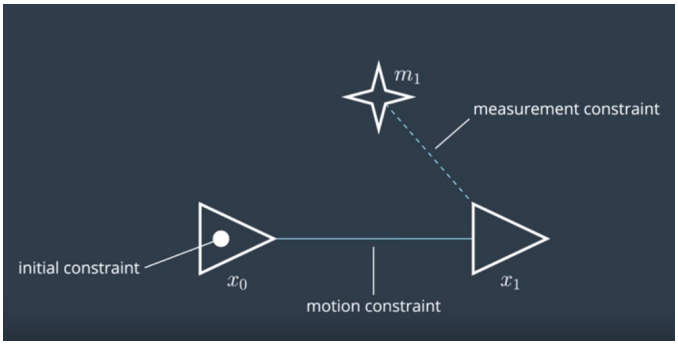

Fig. 2. GraphSLAM Structure

The Real-Time Appearance-Based Mapping (**RTAB-MAP**) algorithm is a GraphSLAM approach and will be used in this project to perform SLAM. This algorithm uses data collected from sensors to localize the robot and map the environment. In RTAB-Map a process called loop closure is used to allow the robot to determine if the location has been observed before. This, over time, increases the number of images for comparison, causing the loop closure process to take longer, proportionally increasing complexity along the way. RTAB-Map is optimized for large-scale and long-term SLAM by using numerous strategies to allow loop closure to be done in real time without dramatically affecting performance.

3 Scenes And Robot Model Configuration

In order to perform SLAM, a custom robot and two different World environment are simulated to perform it in. They will be used to test and simulate a robot performing SLAM and to generate 2D and 3D maps from them.

3.1 Robot Configuration

Simulated robot model was created in Gazebo which initially designed to tackle the localization problem in a previous project. This robot was made of a cylinderical shape, with two cylindrical wheels, and to spherical casters which was equipped with sensors that are required to address SLAM problem. The RGBD camera (kinect sensor) was used that will provide both RGB image and Depth image. Hokuyo lidar was added as a laser scanner that will provide 2D laser scan which allows the robot ROS package slambot to leverage the rtabmap-ros package to perform SLAM (Fig. 3). The mapping is manually implemented by slam-bot package communicates with a teleop package by which the user can control the movement of the mobile robot.

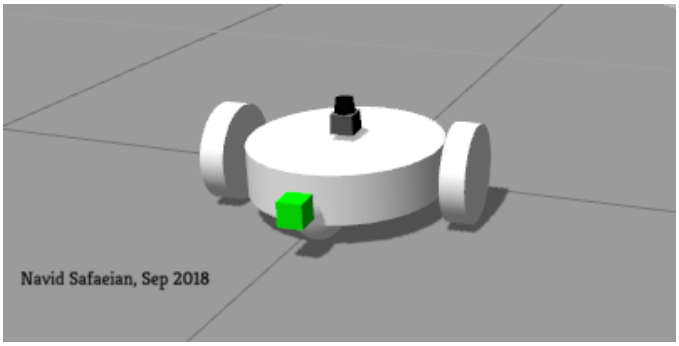

Fig. 3. custom Robot

The transform tree illustrates the backend configuration of the robot including the transform tree links (Fig. 4).

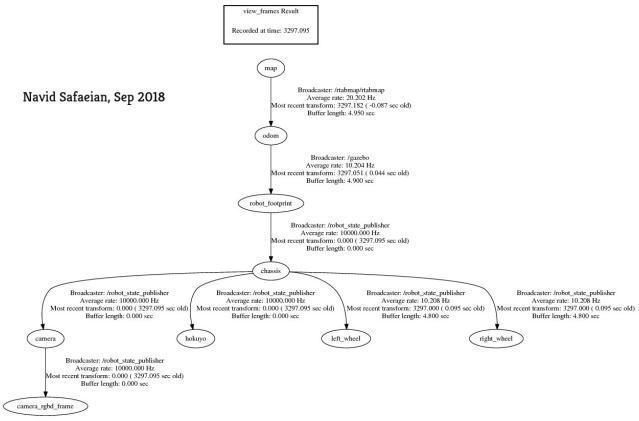

Fig. 4. Robot Transform Tree

3.2 Designed Scenes in Gazebo

The Kitchen and Dining Scene world was provided and will be used as a benchmark for experimentation purposes (Fig. 5). The My World Scene was designed in Gazebo to be used as the custom environment to deploy a robot to implement the SLAM algorithm. This scene was developed with the RTAB-Map Appearance-Based feature in mind. Therefore, this scene is dramatically feature-rich to help the robot easily identify where it is based on appearance, and map the environment more accurately. This scene includes a control

console, a person, a postal mailbox, tables, boxes, totes, cans, and even checkerboard plane that allow the robot to detect more features (Fig. 6).

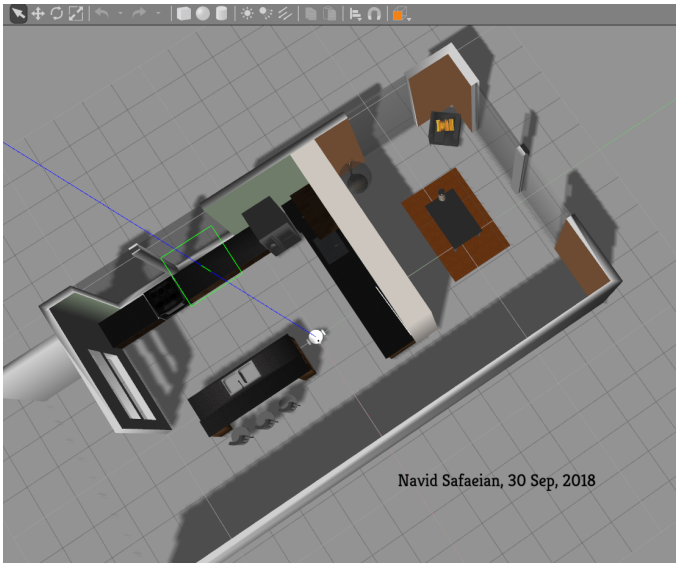

Fig. 5. Kitchen and Dining Scene

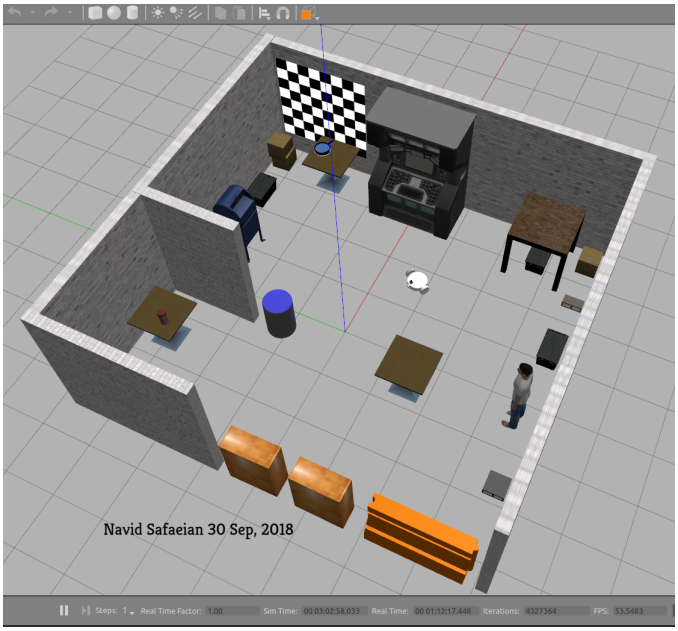

Fig. 6. My World Scene

4 Results

4.1 Kitchen and Dining Scene

The robot started mapping the environment in its immediate vicinity once launching the Kitchen and Dining scene simulation, and launching the mapping node (Fig. 7).
After navigating and loop around the entire world (3 times), the mobile robot was able to generate an accurate map of the world that was provided (Fig. 8).
Several loop closures were observed in the rtabmap database, and a 2D occupancy grid map and a 3D octomap were generated for the provided environment (Fig. 9).

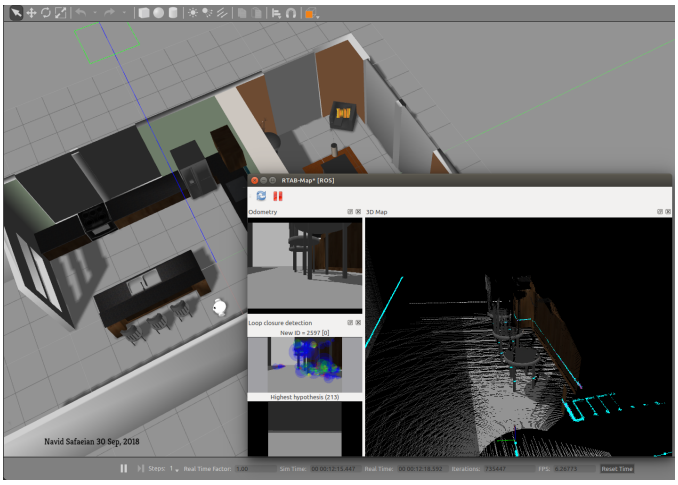

Fig. 7. Kitchen World: SLAM Initialized

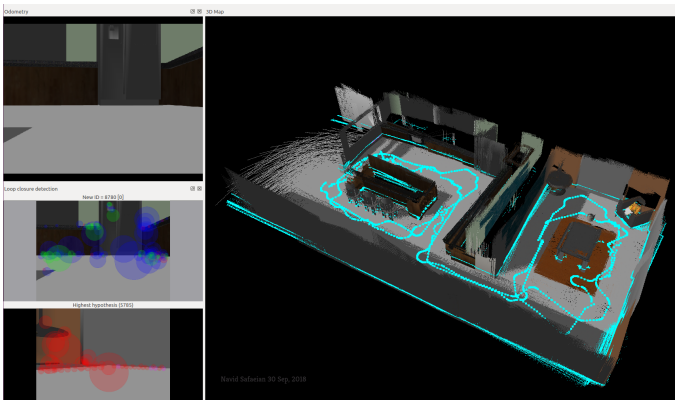

Fig. 8. Kitchen World: Mapped

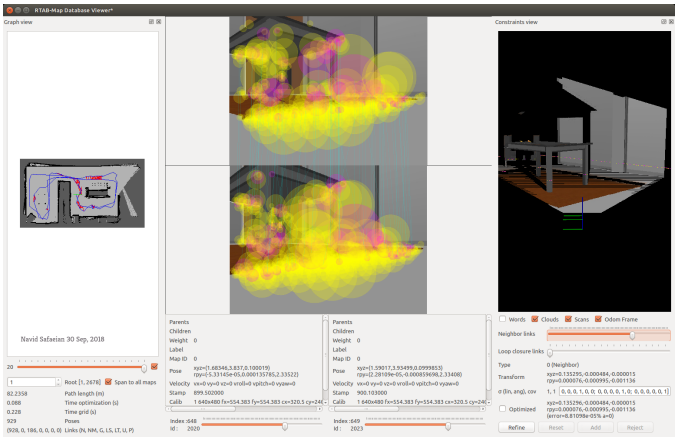

Fig. 9. Kitchen World: Rtabmap Database

4.2 My World Scene

Once the My World scene simulation was launched along with the mapping node the slam-bot started mapping the immediate environment (Fig. 10).
After navigating and looping throughout the entire environment (several loops), the mobile robot was able to generate an accurate map of the world (Fig. 11).

In Fig. 12 from the rtabmap database, several loop closures were observed, and a 2D occupancy grid map and a 3D octomap were generated for the custom developed environment.

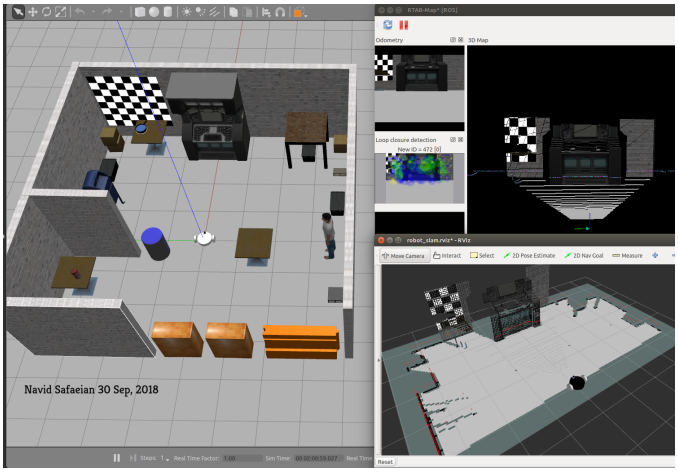

Fig. 10. My World: SLAM Initialized

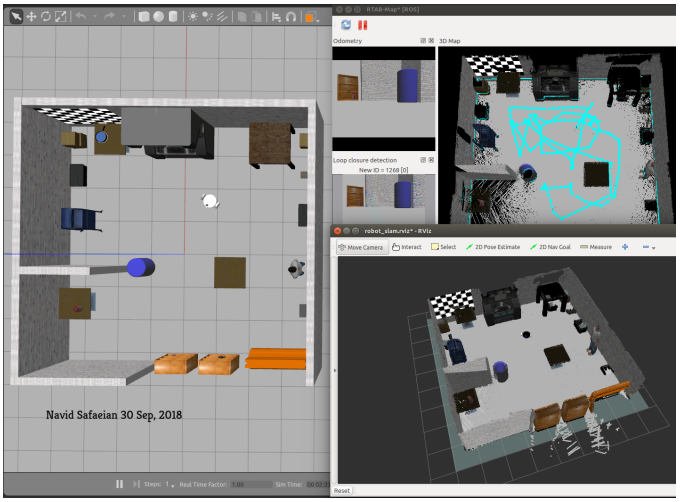

Fig. 11. My World: Mapped

5 Discussion

Overall mapping the environments for both was successful. By configuring all parameters properly, the kitchen world mapping was straight forward and it generated good results for both 2D occupancy map and 3D map. However, the slow mapping was found to be the best method because increasing the speed took more passes to create a recognizable 3D map. Since it was a manual navigation, traversing through the two environment was rather tedious. My world that was created in the first attempt was too large and less features than in the kitchen world. Robot was not able to address the environment mapping and to generate any acceptable 3D map or 2D map because of the lack of features which are required to get loop closure detection. Therefore, in the second attempt, the scene of my world was designed with a smaller size and more features such as adding a cylinder, boxes, checkerboard plane on the wall, and person.
Although the Kitchen and Dining scene seemed to have less noticeable features that the My World scene, it was less complicated and faster to map that the latter. In order to obtain better results in mapping the My World scene, looping around the objects that were placed to serve as both obstacles and features had to be done cautiously. In the case of too close distance to the objects, the robot was not able to accurately localize itself, throwing off the mapping process. Inversely, in a far distance, the mapping of the entire My World scene was not accurate in the feature details. This slight mapping difficulty may comes from the existing open space in my world scene while the laser range-finder had a lower range than necessary for the designed environment. However, the robot was able to successfully map both environments accurately and addressing the SLAM problem.
In conclusion, RTAB-Map is a great algorithm to do SLAM and obtain 2D/3D maps in multiple scenarios, although comparing it to other algorithms such as FastSLAM, this method (RTAB-Map) is more heavy computational and it will require GPU acceleration to tackle complicate scenarios with better results.

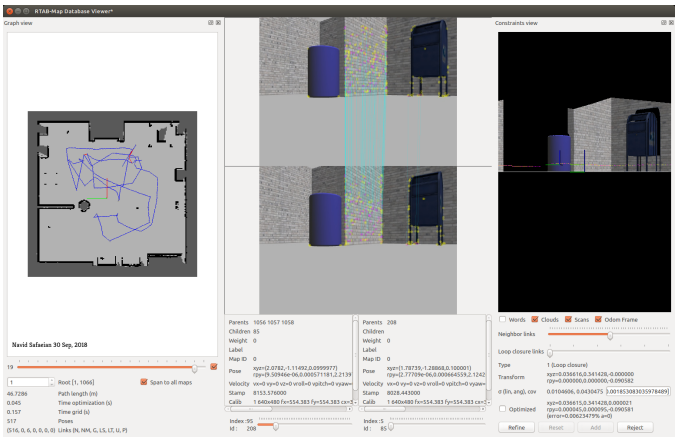

Fig. 12. My World: Rtabmap Database

6 Future Work

RTAB-Map algorithm has a great capability that can be implemented in more complicated scenarios and much wider scopes than what was tested in this project.
In first possible future work, the robot can be developed with a 4-wheeled robot equipped with the Jetson TX2 and a kinect RGBD camera sensor to map an entire real-life apartment or industrial scene to vacuum or somehow load the objects. The second, RTAB-Map algorithm can be used in flying robots to augment the odometry/IMU sensors and improve its accuracy.