Map My World Robot

Saminda Abeyruwan

Abstract—Simultaneous Localization and Mapping (SLAM), a robot acquires a map of the environment and simultaneously localize itself relative to the map. Each robot platform contains a set of constraints on sensors, locomotion, and compute power. SLAM approaches that address these constraints provide real-time benefits to practitioners. Real-Time Appearance-Based Mapping (RTAB-Map) is a an open source loop closure detection with memory management library that addresses these limitations for long-term and large scale environment mapping. This project evaluates RTAB-Map 2D and 3D mapping and localization modalities in two Gazebo environments using a RGB-D camera mounted on a four-wheeled robot.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, SLAM, RTAB-Map.

1 Introduction

One of the fundamental problems inherent to an autonomous robots is to identify an unknown environment and localize itself relative to it, which has been known as the Simultaneous Localization and Mapping (SLAM) problem [1]. Unlike the localization problem, the robot must infer the map and its location from the measurements $z_{1:t}$ and controls $u_{1:t}$. In addition, SLAM must address constraints impose by sensors, locomotion, compute power, cost, accuracy, integration, and real-time requirements of the applications.

Real-Time Appearance-Based Mapping (RTAB-Map) [2] is an open source library that implements loop closure detection with a memory management approach that addresses the prior constraints and able to use long-term and large-scale environment mapping. RTAB-Map provides online processing, robust under low-drift odometry, robust localization, practical map generation and exploitation, and multi-session mapping.

This project leverages RTAB-Map implementation available in the Robotic Operating Systems (ROS) [3] to map two Gazebo environments. A four-wheeled robot (developed in the localization project) mounted with a RGB-D camera has been used to map the environments, and localize itself relative to the the databases constructed from the RTAB-Map packages.

2 Background

There are two versions of the SLAM problem. The first is known as *online SLAM*, which estimates the pose, x_t at time t and map m, given the measurements $z_{1:t}$ and controls $u_{1:t}$., $p(x_t, m|z_{1:t}, u_{1:t})$. The graphical model of the online SLAM problem is shown in Fig. 1. This method evaluates variables at time t, it is incremental, and discard past measurement and controls once they have been processed. The second SLAM problem is known as *full SLAM*, which estimates posterior over the entire path $x_{1:t}$ and the map m, $p(x_{1:t}, m|z_{1:t}, u_{1:t})$. The online SLAM problem is the result of integrating all the variables past time t of the full SLAM problem, i.e., $p(x_t, m|z_{1:t}, u_{1:t}) = \int\int\dots\int p(x_{1:t}, m|z_{1:t}, u_{1:t})dx_1dx_2\dots dx_{t-1}$. The graphical

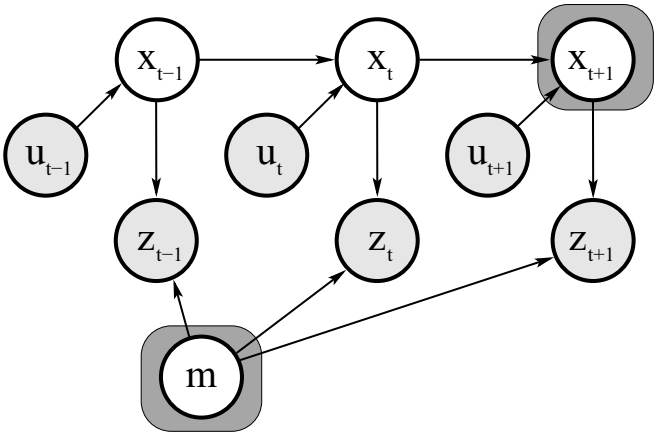

Fig. 1. Graphical model of the online SLAM problem. The orignal figure is available in [1] (Figure 10.1).

model of the full SLAM problem is shown in Fig. 2. There are several popular SLAM algorithms:

1) SLAM with extended Kalman filter: that applies the EKF to online SLAM using maximum likelihood data association. The map is feature based and the landmarks are usually limited, and requires significant efforts to detect features. With known correspondence, the updates are time quadratic in the number of landmarks in the map.
2) The GraphSLAM algorithm: which addresses the full SLAM problem. It calculates posterior over full robot path and map, thus, it is a batch algorithm. GraphSLAM constructs a graph of soft constraints, i.e., measurement are mapped into edges that represent nonlinear constraints between poses and sensed features, while motion commands are mapped into soft constraints between consecutive poses. The graph is sparse, and maximizing the sum of the constraints via maximum likelihood estimates the robot path and the map.
3) The sparse extended information filter (SEIF): which is similar to GraphSLAM, however, it integrates out

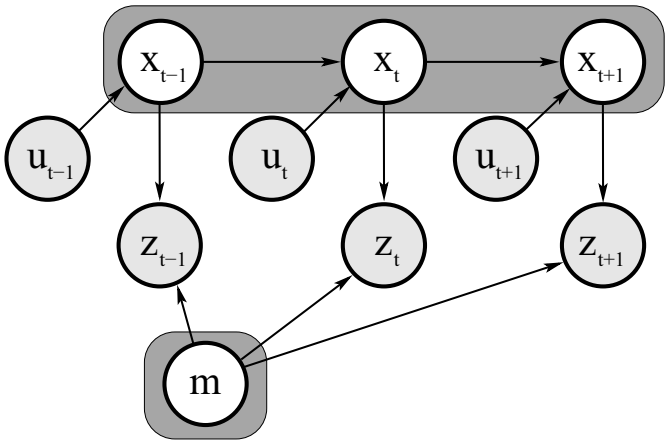

Fig. 2. Graphical model of the full SLAM problem. The orignal figure is available in [1] (Figure 10.2).

past poses, which resulted in an online SLAM. This is an efficient with information representation and integrating out past poses. To achieve online SLAM, SEIF makes number of approximations, which results in less accurate that GraphSLAM or the EKF.

4) The FastLAM algorithm: is a particle filter approach to SLAM problem. FastSLAM maintains a set of particles, which contains a sampled robot path. It also contains a map, where each feature of the map is represented by it own local Gaussian. This representation requires space linear in the size of the map, and linear in the number of particles. The update in the algorithm follows the conventional particle filter, and updates are preformed online, thus, it solves the online SLAM problem (for further discussion, please refer to chapters 11, 12, and 13 in the reference [1].

This project uses RTAB-Map, an open source loop closure detection with memory management, is a graph-based SLAM approach that has been integrated to ROS as the *rtabmap_ros*. Fig. 4 shows the block diagram of rtabmap ROS node. A RGB-D camera, odometry, and laser scan has been used as the external inputs to RTAB-Map. The ROS package, *depthimage_to_laserscan* has been used to obtain the laser scan from the RGB-D camera.

RTAB-MAP, structure of the map is a graph with linked nodes. After sensor synchronization, the short-term memory (STM) creates a node memorizing the odometry pose, and additional sensor data, e.g., visual words for loop closure and proximity detection, and local occupancy grid for global map assembly. Nodes are created at a fixed rate (how much data created from nodes overlap each other). Links are added in the STM between consecutive nodes with odometry transformation. Loop closure and proximity links are added through loop closure detection or proximity detection. All the links are used as constraints for graph optimization. When there is a new loop closure or proximity link added to the graph, graph optimization propagates the computed error to the whole graph, to decrease odometry drift. With the graph optimized, OctoMap, and 2D occupancy grid outputs are assembled and published to external packages. RTAB-Map memory management runs on top of the graph, which limits the size of the graph such that long-term online SLAM is achieved in large environment. This is implemented via a working memory (WM) and a long-term memory (LTM). More information about the operations, please refer to [4].

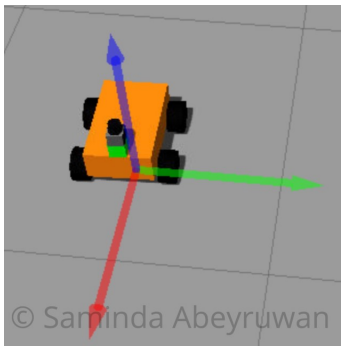

Fig. 3. Visual appearance of the mobile robot.

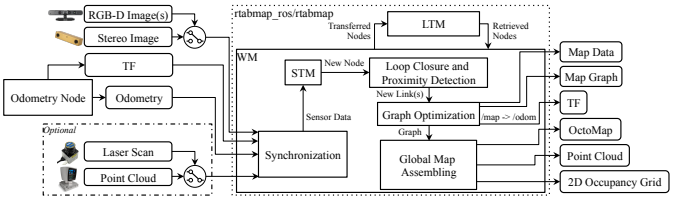

Fig. 4. Block diagram of *rtabmap_ros* node. The orignal figure is available in [4] (Figure 1).

3 Scene and Robot Configuration

The mobile robot is developed using Xacro[1] and Gazebo[2] definitions. The robot is constructed with four-wheels, a Hokuyo laser scanner[3], and a RGB-D camera attached the chassis. The chassis is a box 0.2, 0.3, 0.1, while the wheels are cylinders with radius 0.05 and length 0.05. The moment of inertia for each object shape has been calculated explicitly. A Skid Steering Drive has been attached to front/back left/right wheels and configured via Gazebo. Fig. 3 shows the structure of the mobile robot, and the visual appearance is given in Fig. 5. An addition frame, *camera_rbgd_frame*, has been added to for the depth image frame. Even though the robot is equipped with Hokuyo laser scanner, the ROS package *depthimage_to_laserscan* has been used to convert the depth image to laser scan, and the project uses this topic instead (Fig. 6).

There are two worlds for mapping and localization: 1) Kitchen and Dinning world (KAD): which is the baseline environment provided to the project (Fig. 7), and 2) Apartment world (APT): which has been developed using Gazebo's building editor and objects (Fig. 8).

1. http://wiki.ros.org/xacro
2. http://gazebosim.org/tutorials/?tut=ros_urdf
3. http://gazebosim.org/tutorials?tut=ros_gzplugins

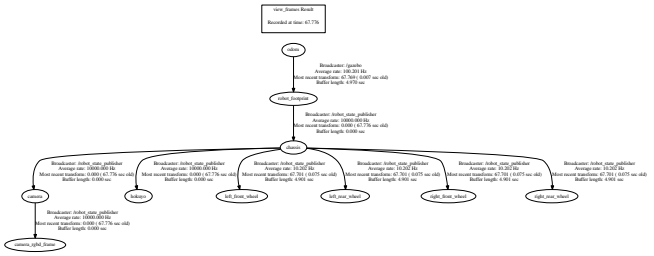

Fig. 5. Structure of mobile robot.

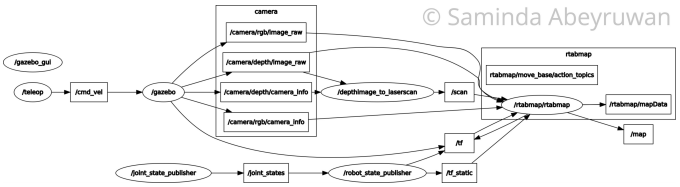

Fig. 6. ROS graph for environment mapping.

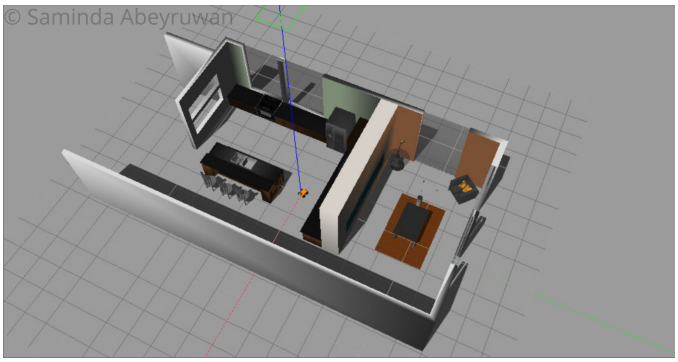

Fig. 7. Kitchen and dinning world.

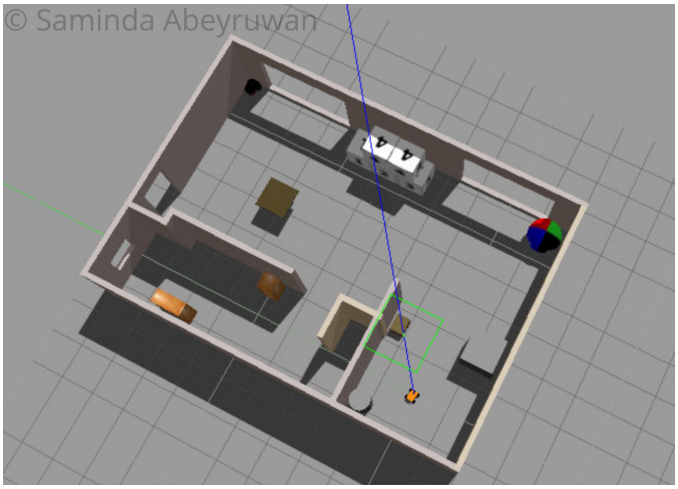

Fig. 8. Apartment world.

4 Results

The results are shown herewith based on the images captures from Rviz and *rtabmap-databaseViewer*. The robot is placed on the map (0, 0, 0) coordinates and teleop module has been used to move the robot. Fig. 9 show the partial 3D map, and Fig. 10 shows the 2D view for the KAD world. Similarly, 3D and 2D maps for APT world is shown in Fig. 12 and Fig. 13 respectively. The robot has completed as least two paths in both world to obtain the final results. Fig. 11 shows a snapshot of the localization for the KAD world.

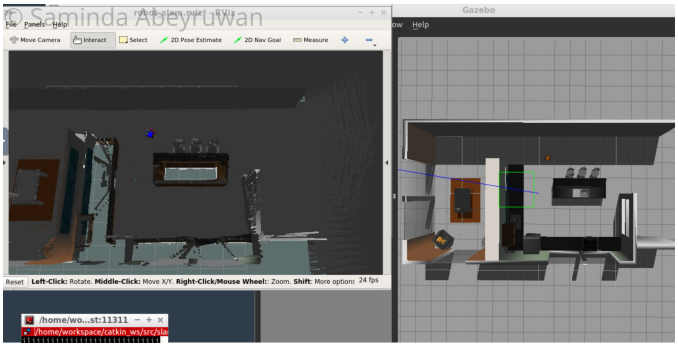

Fig. 9. Kitchen and dinning world: partial 3D map.

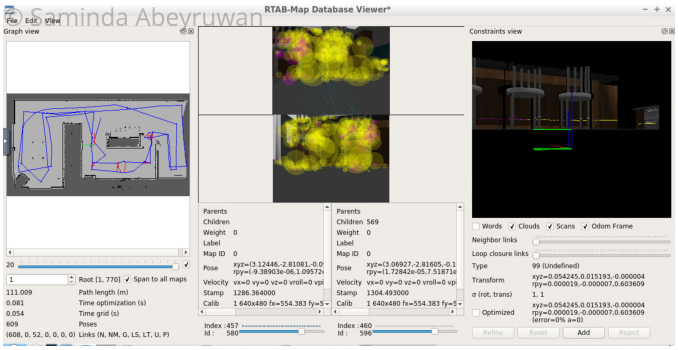

Fig. 10. Kitchen and dinning world: 2D map from RViz.

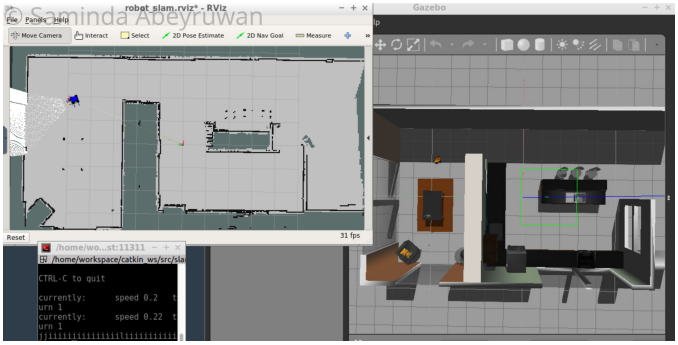

Fig. 11. Kitchen and dinning world: robot is configured to localize using the database.

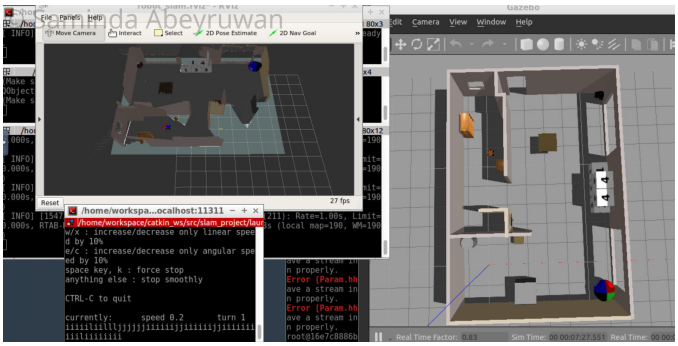

Fig. 12. Apartment world: partial 3D map.

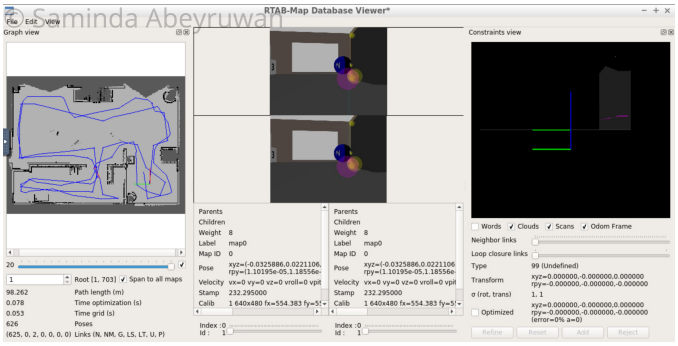

Fig. 13. Apartment world: 2D map from RViz.

5 Discussion

The rtabmap ROS package has mapped both worlds and the robot has been localized on the map. The teleop module has been used for manual navigation. The mapping module has been configured to use depth image based scan. As a future work, Hokuyo scan can be used instead. It has been observed that ratabmap requires a feature rich environment for loop closures. If the feature were not distinct, or the robot moves faster, it has been observed that when a loop closure is found, the graph optimization distorts the mapped environment significantly. The APT world contains more open space than the KAD world, which may have made the mapping complicated at times.

6 Conclusion / Future work

The project evaluates a solution to the online SLAM problem using RTAB-Map. It has been shown that the robots were successfully mapped and localized on the worlds. All tasks have been experimented in simulation. It is also interesting to integrate the packages on real hardware and tune for the real world constrains. This project uses RGB-D camera, though, RTAB-Map can be configured to use point cloud devices or stereo images as external inputs.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[2] M. Labbé and F. Michaud, "Long-term online multi-session graph-based splam with memory management," *Auton. Robots*, vol. 42, pp. 1133–1150, Aug. 2018.

[3] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[4] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*.