RTAB SLAM Project

Muthanna A. Attyah

Abstract—RTAB-Map is considered to be the best solution for SLAM to develop robots that can map environments in 3D. These considerations come from RTAB-Map's speed, memory management, and its software tools that are available for wide range of environments. In this project **rtabmap_ros** package, which is a ROS wrapper (API) for interacting with RTAB-Map will be configured in ROS on a simulated robot with RGBD camera and laser scanner. The algorithm will be used to SLAM in two different simulated worlds to generate a 2-D and 3-D occupancy maps for both worlds.

Index Terms—Robot, IEEEtran, Udacity, Localization, SLAM, RTAB Map, Octomap, ROS

1 Introduction

RTAB-Map (Real-Time Appearance-Based Mapping) is a RGB-D Graph-Based SLAM approach based on an incremental appearance-based loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location. When a loop closure hypothesis is accepted, a new constraint is added to the maps graph, then a graph optimizer minimizes the errors in the map. A memory management approach is used to limit the number of locations used for loop closure detection and graph optimization, so that real-time constraints on large-scale environments are always respected. 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 range-finder for 3DoF mapping. RTAB-Map tools are available for multiple environments including ROS, Ubuntu, MacOS, Windows, Android, Raspberry Pi, and Docker. [1]

2 Background

Mapping is one of the most important capabilities for a mobile robot to enable it to achieve its tasks. It consists of creating a representation of obstacles in its environment based on data obtained from robot sensors. Mapping task is harder when considering the inherent noise in the data obtained from most of the sensors. There are many mapping algorithms that are designed to handle the inherent noise and operate either in small indoor structures or complex outdoor environments.

Algorithms that are designed for mapping indoor spaces usually generates two dimensional floor plan-like maps, which can fairly represent walls and other features. This type of representation becomes very poor when we try to model more complex environments, which usually have much richer features to be represented (e.g. buildings, trees, and cars on outdoors). A natural extension for 2D maps is to use three-dimensional representations. [2]

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**. SLAM process has multiple challenges including unknown map, big size of environment, huge hypothesis space, noise from sensors, perceptual ambiguity, and number of cycles taken by robot. Following are some of the popular approximate solution methods that will do SLAM and handle some of the challenges.

- **EKF SLAM** algorithms are feature based, and use the maximum likelihood algorithm for data association. For the past decade, the EKF SLAM has been the de facto method for SLAM, until the introduction of FastSLAM. Associated with the EKF is the Gaussian noise assumption, which significantly impairs EKF SLAM's ability to deal with uncertainty. With greater amount of uncertainty in the posterior, the linearizion in the EKF fails. [3]

- **GraphSLAM** algorithm uses sparse information matrices produced by generating a factor graph of observation interdependencies (two observations are related if they contain data about the same landmark) [3]

- **Occupancy Grid** to represent a map of the environment as an evenly spaced field of binary random variables each representing the presence of an obstacle at that location in the environment. Occupancy grid algorithms compute approximate posterior estimates for these random variables. The most common type of occupancy grid maps are 2d maps that describe a slice of the 3d world.

- **FastSLAM 1 and 2** estimates a posterior over the trajectory using a particle filter approach. This will give an advantage to SLAM to solve the problem of mapping with known poses. It uses a low dimensional Extended Kalman Filter to solve independent features of the map which are modeled with local Gaussian.

- **ORB-SLAM 1 and 2** a real-time SLAM library for Monocular, Stereo and RGB-D cameras that computes the camera trajectory and a sparse 3D

reconstruction (in the stereo and RGB-D case with true scale). It is able to detect loops and re-localize the camera in real time. [4]

- **CoSLAM** a visual SLAM software that aims to use multiple freely moving cameras to simultaneously compute their egomotion and the 3D map of the surrounding scenes in a highly dynamic environment. [5]

In this project RTAB-MAP which is a Graph-Based SLAM approach based on an incremental appearance-based loop closure detector will be used to test SLAM algorithm in simulated indoor small size environment.

3 Robot Model Configuration

Simulated robot model was created in Gazebo and equipped with sensors that are required to solve SLAM problem. Microsoft XBOX 360 Kinect sensor was added as the RGBD camera that will provide both RGB image and Depth image. Hokuyo lidar was added as a laser scanner that will provide 2D laser scan. Hokuyo sensor position was elevated to be higher and behind the RGBD camera to avoid any impact on the laser scanning beams. Robot is having two wheels with differential driving mechanism. Robot Shape is as illustrated below:

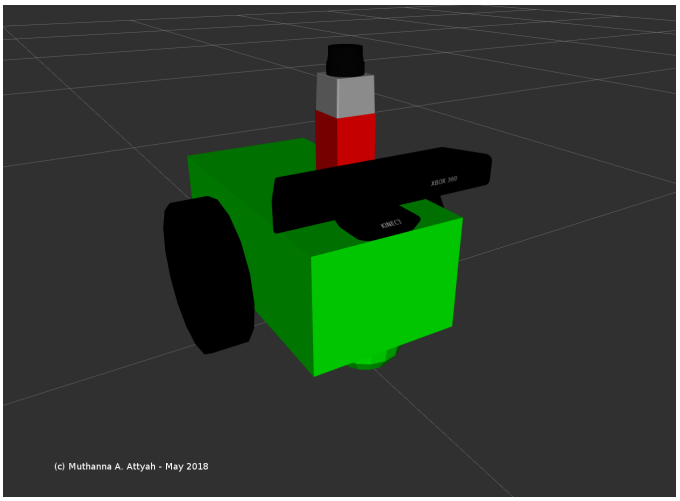

Fig. 1. robot

The tf frames map of the different robot joints are as showing in below tf map. One additional joint was added to the RGBD camera to fix the orientation of the generated depth point cloud which is by default is generated along the Z axis.

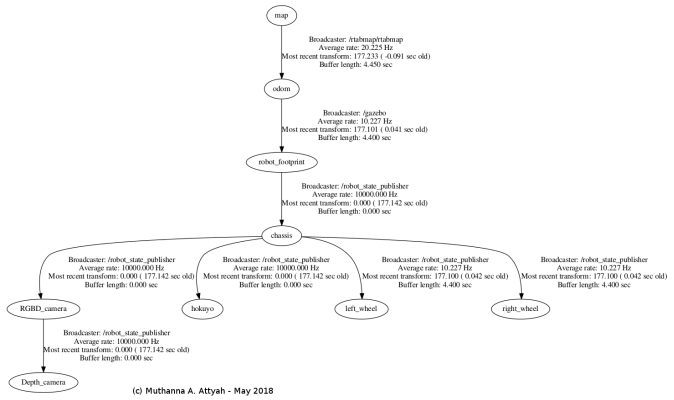

Fig. 2. TF Frames Map

3.1 Gazebo Plug-ins used in Robot Model

The following Gazebo plug-ins were used in the robot model:

- **libgazebo_ros_diff_drive.so** : to simulated the differential wheel driving mechanism.
- **libgazebo_ros_openni_kinect.so** : to simulated Microsoft XBOX 360 Kinect RGBD camera.
- **libgazebo_ros_laser.so** : to simulate Hokuyo laser scanner sensor.

3.2 ROS Packages used in Robot Model

The following ROS Packages were used in the robot model:

- **gazebo_ros** : Gazebo Simulator Package to spawn the robot and publish robot and joint states.
- **rviz** : RViz package to visualize the robot, sensors data, and simulated world.
- **slam_project teleop_key** : Teleoperation node to control robot movement using keyboard.
- **rtabmap_ros** : ROS wrapper (API) for interacting with RTAB-Map.

Below ROS graph is showing all ROS Nodes generated from above packages, topics, and subscriptions that are enabled in the robot model.

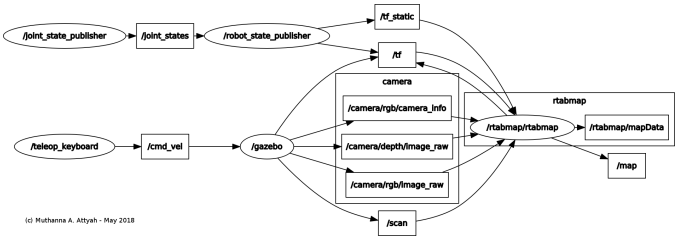

Fig. 3. ROS Graph

rtabmapviz is an additional node that can be used for real time visualization of feature mapping, loop closures,

and more rtab-map parameters. However its not recommended to use this tool while mapping in simulation due to the computing overhead. inputs and outputs to this node are similar to inputs and outputs to rtabmap_ros node.

3.3 Parameters

When deciding the configuration parameters for all above mentioned plug-ins and packages following points was taken into consideration:

- Hokuyo laser 2D scanner will be publishing **/scan** topic. It's name should match what will be subscribed by rtabmap node.
- RGBD camera will be publishing **rgb/image**, **depth/image**, and **rgb/camera_info**. It's names should match what will be subscribed by rtabmap node.
- **frame_id** needs to be set to match the name of the robot foot print link id name (**robot_footprint in our case**.
- Two Loop Closure Detection methods were tested during the simulation **0=SURF** and **4=FAST/BRIEF**. SURF was not performing well when the simulated world was having less details and running on a relatively slow PC, FAST/BREIF gave better results in this case.
- **Vis/FeatureType** was added and set to value similar to the value of **Kp/DetectorStrategy** to suppress a warning message that used to appear every time rtabmap node was launched.
- there are many other parameters that was left as per default settings. ROS Wiki is a good reference that can be used to further fine tune parameters when scenario is changed. [5]

4 World Creation

Two simulated worlds were used during the test of RTAB-Map algorithm using a single robot model:

4.1 Supplied Gazebo World

First world is supplied by Udacity and representing a 14m x 8m kitchen and dinning area (in door) with good level of details/features in every location and angel. Kitchen World is as shown below.

Fig. 4. Kitchen World

4.2 Personal Gazebo World

Second world was created in Gazebo, representing a 8m x 8m laboratory with less number of features in compare to the kitchen world. Personal World is as shown in below image:

Fig. 5. Personal World

5 Results

Both occupancy grid and 3D map for the kitchen world are as shown below:

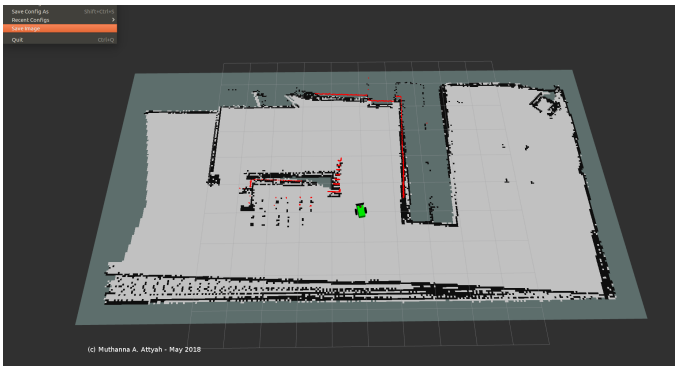

Fig. 6. Kitchen 2D Map

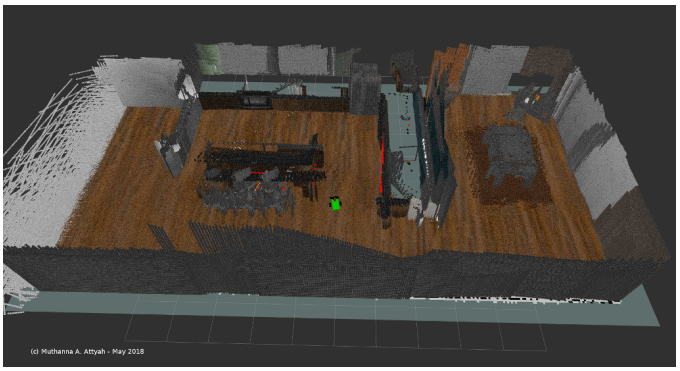

Fig. 7. Kitchen World 3D Map

A recorded video for the 3D map can be accessed on this link

occupancy grid and 3D map for the personal world are as shown below:

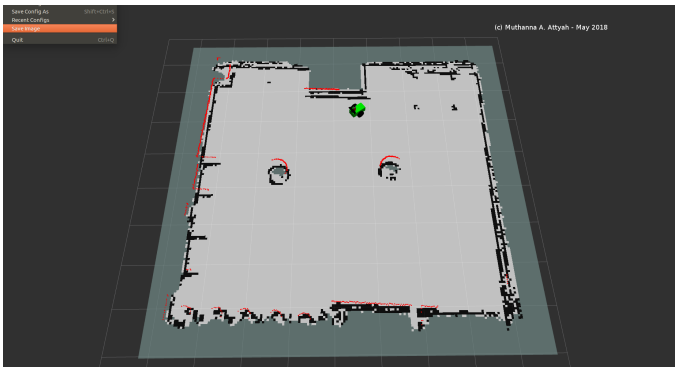

Fig. 8. Personal 2D Map

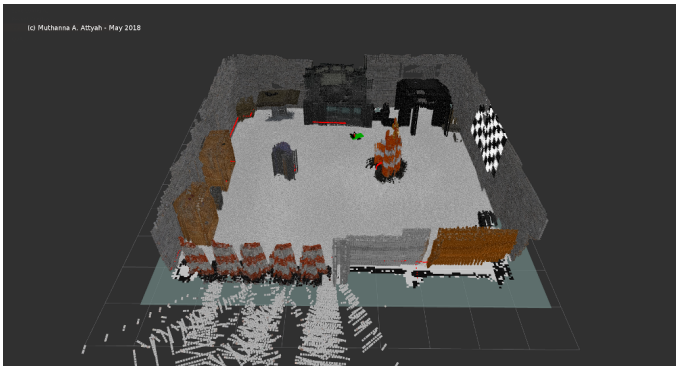

Fig. 9. Personal World 3D Map

Generated RTAB-Nap Databases for both kitchen world and personal world can be downloaded from the following URL rtapmap databases.

6 Discussion

After having all parameters properly configured taking into considerations all points mentioned in parameters section above; kitchen world mapping was straight forward and it generated good results for both 2D occupancy map and 3D map. however process was slightly slow because of the relatively slow PC used in simulation which was also lacking GPU accelerations. Udacity workstation was not used because of the slow Internet link available at the time of this test.

Personal world that was created in the first attempt was 12m x 12m representing an outdoor scene with much less features than what is available in kitchen world. Robot was not able to generate any acceptable 3D map or 2D map due to the lack of features which are required to get loop closure detection. Second attempt was done by reducing the size of the world to 8m x 8m and adding few more features to the scene such as construction cones, tables, cylinders, etc. Results of second attempts were much better. Third attempt included more features such as adding a checker board on the wall and computer workstation then mapping method was changed from SURF to FAST/BREIF which is much less computing intensive then the results was much better and reached the level that was obtained from kitchen world.

Conclusion was that RTAB-Map is a great algorithm to do SLAM and obtain 2D/3D maps in multiple scenarios. It has good number of parameters that will add to its flexibility allow the user to adjust it as required by the specific robot application. However when comparing it to other algorithms such as FastSLAM, RTAB-Map is more computing intensive and it will require GPU acceleration to give satisfactory results. Using NVidia Jetson TX2 platform with RTAB-Map is a good combination to overcome this algorithm limitation.

7 Future Work

RTAB-Map algorithm has a great capability that can be utilized in a much wider scope than what was tested in this project. below are few examples that can be considered.

- A simpler/cheaper robot can be used by eliminating the 2D Laser scanner which is usually comes with high cost and removing the wheel encoders then depending on RTAB-Map algorithm to generate both a visual odometry and 2D scanning (using depthimage_to_laserscan package). This approach can be good for applications where less accuracy and cheaper robots are required.

- RTAB-Map can be combined with other ROS navigations packages to autonomously drive the robot and detect/avoid obstacles that are identified with RTAB-Map algorithm

- Algorithm can be used in flying robots to augment the odometry/IMU sensors and improve its accuracy.
- Algorithm can be used to create a hand held 3D scanner that will generate 3D models for objects for further processing/3D printing.

References

[1] M. Labb and F. Michaud, "Real-time appearance-based mapping github repo,"
[2] G. S. S. Denis F. Wolf, "Localization and mapping in urban environments using mobile robots,"

[3] t. f. e. Wikipedia, "Simultaneous localization and mapping,"
[4] J. M. M. M. D. G.-L. Raul Mur-Artal, Juan D. Tardos, "Orb-slam2 github repo,"
[5] D. Zou and P. Tan, "Coslam: Collaborative visual slam in dynamic environments," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2013.