Map My World - ROS RTABMap

Snehal Gor

Abstract—Mobility is one of the basic need not only for humans or animals, but also for robots. Mobile robot requires localization of itself in an environment / map. Many a times map is available, then it's just a problem of localization of the robot with respect to Map. But many more times map is not available or environment is dynamic. In those cases we require to do Simultaneous Localization and Mapping (SLAM).Author here has explored ROS RTABMap (Real Time Appearance based Mapping). RTABMap is appearance based SLAM (GraphSLAM), whereby vision sensor used for localizing the robot and mapping the environment. Author has successfully build a map of provided kitchen dining environment. Further Author has also explored building own map environment using Gazebo, modified the robot, constructing own launch files and tuning different parameters of ROS packages. Author here is documenting his learning, findings and results.

Index Terms—Robot, Udacity, LaTeX, SLAM, RTABMap, GrapgSLAM, FastSLAM, Monte Carlo Localization, ROS, Gazebo.

1 Introduction

As age of data, artificial intelligence and robots dawn upon us, robots which can handle complex environment autonomously without any human interventions is becoming everyday reality. We see robots in every sphere of life, be it in industrial automation, entertainment, space exploration or Autonomous vehicle (and many more). Robots are making our world faster, safer, greener, smarter and enjoyable.

For robot to be mobile and autonomous in an environment, localization and mapping are primary need. Localization is relatively easier, when we have map of the environment available. But in real world where many a times robot needs to be in an unknown environment or environment keeps changing (Dynamic), we require to do Simultaneous Localization and Mapping (SLAM).

SLAM is the computational problem of constructing or updating a map of an unknown environment, while simultaneously keeping track of robot's location within it.

- Localization: Mobile robot localization is the problem of determining robot's pose (location and orientation) in an environment from sensor data.
- Mapping: Mapping is a problem of Inferring a map given a pose (location and orientation).
- SLAM: Learning a map and locating the robot simultaneously.

Given a series of sensor observations o_t and controls u_t over a discrete time stamp t, the SLAM is to compute an estimate of the robot's location x_t and a map of the environment m. In usual probabilistic form, objective is to compute [1]

$$P(m, x_t)|o_{1:t}, u_{1:t}$$

In this report, Author is documenting his exploration of SLAM using ROS RTABMap (Real-Time Appearance based Mapping). In subsequent section Author will document

- Background
- Project brief and Configuration
- Results
- Discussion
- Future Work

2 Background / Formulation

Mobile robot localization and mapping (SLAM) comes in many flavors in particular below Author has listed prominent three categories as per underlying estimation technique

- Extended Kalman Filter (EKF) and its variants
- Sparse Extended information filters (SEIF)
- Particle Filters / Monte Carlo / FastSLAM
- Least Square Error minimization / GraphSLAM

Further SLAM algorithms can be sub-divided into full and online SLAM. Full SLAM estimates the entire path and map, while online SLAM estimates most recent pose and map.

Online SLAM:

$$P(m, x_t|o_{1:t}, u_{1:t})$$

Full SLAM:

$$P(m, x_{1:t}|o_{1:t}, u_{1:t})$$

The second key feature of the SLAM problem relates to its nature. SLAM problems generally have a continuous and a discrete element.

Nature - Continuous:

During SLAM, a robot continuously collects odometry information to estimate the robot poses and continuously senses the environment to estimate the location of the object or landmark. Thus, both robots poses and object location are continuous aspects of the SLAM problem.

Nature - Discrete:

Robots continuously sense the environment to estimate the location of the objects, when doing so SLAM algorithms have to identify if a relation exists between any newly detected objects and previously detected ones. This helps the robot understand if it has been in this same location before. The answer to this question is binary - either yes or no - and thats what makes the relation between objects a discrete component of the SLAM problem. This discrete relation between objects is known by correspondence.

Below (Table 1) Author has tried to provide brief comparison of different approaches

TABLE 1
EKF vs. FastSLAM vs. GraphSLAM

	EKF	FastSLAM	GraphSLAM
Data Association	One for each Landmark	Each Particle has own hypothesis to landmark	Hypothesis and Constraints for landmarks on entire path
Online/Full	Online	Both	Full
Complexity	Quadratic	Linear	Linear (Linear-bounded for RTABMap)
Map Handling	Small	Medium	Large

RTABMap is a GraphSLAM implementation. It's an appearance based SLAM (uses vision sensor), which uses loop closure to identify previously seen location. It's optimized for large scale and long term SLAM, which uses multiple strategies like bag of words, memory management to perform SLAM in real time. Further RTABMap do not use landmark based constraints and only uses odometry constraints and loop closure (Global), that way it slightly differs from GraphSLAM.

3 Project Brief and Model Configuration

Author used Gazebo and different ROS packages to implement a package for RTABMap SLAM. The package launches a custom robot with RGBD camera and laser sensor in a Gazebo world and utilizes RTABMap for SLAM. Following is the brief explanation

- Author's package name is *mapworld_sg*
- worlds folder: Contains Udacity provided *kitchen_dining*, and other two world files constructed by Author - *gas_station_sg* and *room_sg*
- urdf folder: Contains a xacro file and gazebo file, related to the Robot. Author modified the xacro files for addition of RGBD camera link, with transformation to correct the image. Author modified the gazebo file with addition of RGBD camera plugin and related parameters. Also topics are correctly named.
- script folder: Udacity shared teleop script being copied in this folder
- meshes folder: Udacity shared Hokuyo dae file copied here
- launch folder: Author added following launch files
  - robot_description: Sends URDF to parameter server, sends fake joint values and sends robot states to tf
  - slam_world: Spawns the robot in gazebo world (Udacity kitchen_dining)
  - mapping: Launches rtabmap node and for visualization rtabmapviz node. Author modified topics here to map to correct published topics - RGB raw, RGB depth, RGB info, Scan, Map, etc. Also parameters like Hessian threshold and detection rate modifed.
  - rviz: Launches rviz
  - teleop: Launches teleop node
  - sg_slam_world: Spawns the robot in gazebo world (Author's created *room_sg*)
  - sg_mapping: RTAB Mapping related launch file for Author's world. Hessian threshold changed.

Below Fig 1 shows overall setup [2]

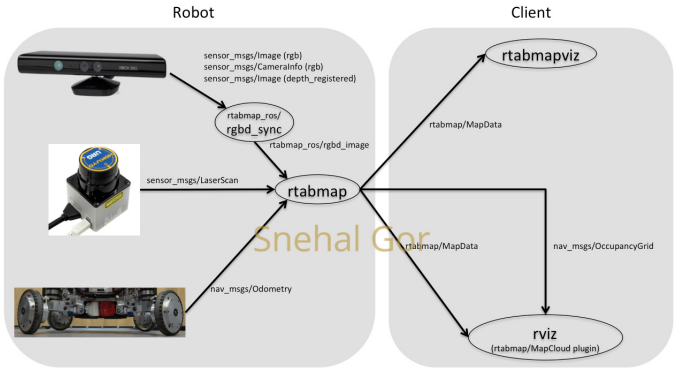

Fig. 1. Overall Setup with tf.

Below Table 2 provides brief the robot setups, while Fig 2 shows simple graphical representation of the same.

TABLE 2
Brief details of Author's Robot setup

	Robot (Udacity)
Sensors	RGBD Camera and Hokuyo Range Finder
RGBD Camera Position	In front on chassis
Laser Range Finder	In front on chassis
Wheels	4
Number of Joints	4 (2 wheels and 2 sensors)
Actuation	Two wheel Differential

Further Author mapped all topics across different launch and xacro files. During debugging Author realized that, camera image is not coming correctly. To overcome the problem Author added dummy link with transformation *rpy="-1.57075 0 -1.57075"*. Below Fig 3 shows overall transform frames

3.1 Differential Drive [3]

One important part of Author's robot setup is Differential drive and corresponding kinemetics model. Below Author is sharing his understanding and learning. A differential drive is based on two separately driven wheels and doesn't require additional steering motion. For balancing the robot with differential drive, additional wheels are required. Author used two caster wheels for the Robot.

In differential drive, robot's motion vector is sum of independent wheel motions.

- Straight line motion is produced by moving both the wheels at the same rate in same direction.

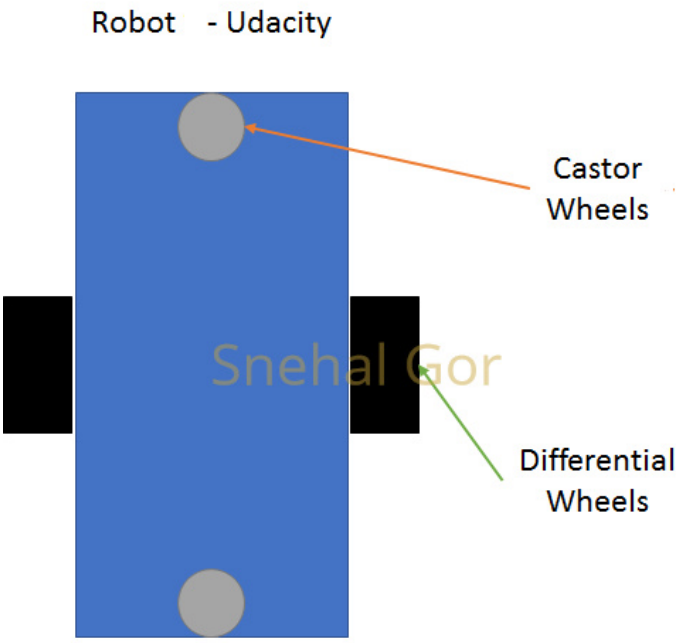

Fig. 2. Author's Robots Setup

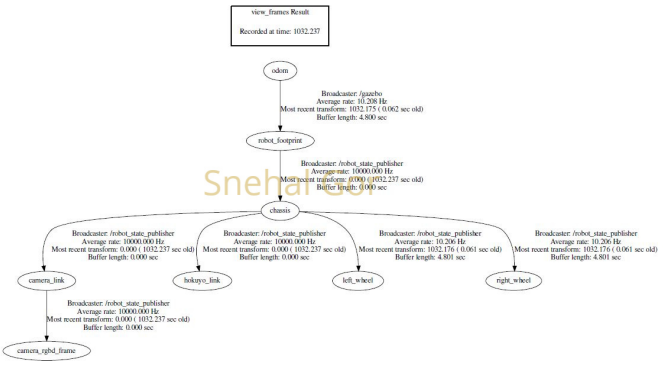

Fig. 3. Overall Transformation Tree.

- In place rotation is produced by moving both the wheels at the same rate in opposite direction
- Other variations like turning left / right while forward / back-word moving, is obtained varying speed and/or direction of the wheels

3.2 Author's Environment

Author used Gazebo model editor, to construct 2 environments

- Gas Station - Fig 4
- Room - Fig 5

3.3 Configurations / Parameter Tuning [2]

Below Author is providing insights into different parameters tuned by Author.

Fig. 4. Output Snapshot - Author's constructed Gas station Environment.

Fig. 5. Snapshot - Author's constructed Room Environment.

- RTABMap
  - Author changed DetectionRate to 2 from 1, to increase the number of loop closure found / constraints with possibility of getting better map
  - For Author's environment (sg_room), Author increased *'SURF/HessianThreshold'* to 600. As Author observed lot many SURF features on the wall, affecting the correspondence. Which in turn identifying false loop closure, and affecting the map quality. By increasing the Hessian threshold, Author has rejected those not so strong features / corners.
  - Author kept laser model parameters to default, with maximum range to 30. As Author wanted in wider space in a map, robot should be able to localize itself. (Further please refer discussion Section 5 below)

- Overall - Author updated different parameters to map topics correctly - Camera raw image, Camera depth image, hokuyo scan, Odom frame, map, etc.

4 Results

Author got a good result, with Udacity provided and Author constructed Room environments. Also mapping accuracy is quite good. Snapshots of the both environments have been provided below

- RViz Snapshots of Kitchen Dining Mapped Environment - Fig 6
- RViz Snapshots of Kitchen Dining Occupancy Grid - Fig 7
- RViz Snapshots of Author's Room Mapped Environment - Fig 8
- RTABMap DB Viewer Snapshot for Kitchen Dining Environment - Fig 9
- RTABMap DB Viewer Snapshot for Author's Room Environment - Fig 10

Author couldn't achieve good result with Author's gas station environment. Learning while mapping the gas environment, further shared in the Discussion section 5.

Fig. 6. Output Snapshot - Mapped Udacity provided Kitchen Dining Environment.

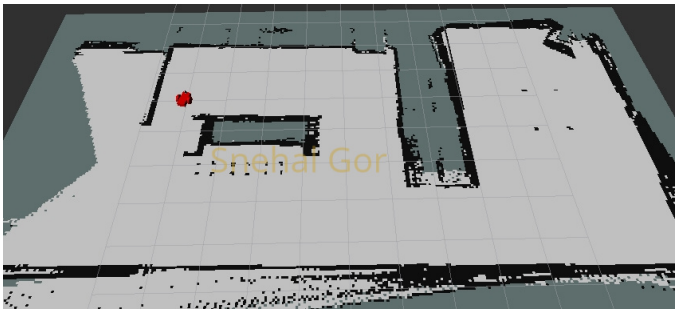

Fig. 7. Snapshot - Occupancy Grid for Udacity provided Kitchen Dining Environment.

5 Discussion

Author could successfully map two environment. In Author's opinion following are few of the learning / reasons of getting a good performance

- Simplicity of the environment : Map environment which has been constructed by Author in Gazebo, is static with no moving obstacles or complex reflection or patterns
- Ideal sensor models, where almost no effects of environment or inherent noises
- RTABMap (GraphSLAM) is one of the popular approach for SLAM, which can do real-time SLAM with global loop closure

Fig. 8. Output Snapshot - Mapped Author created Room Environment.

Fig. 9. RTABMap DB Viewer Snapshot for Udacity provided Kitchen Dining Environment.

Author faced some challenges. Below Author is sharing learning

- Initial camera image not coming correctly. Rviz helped in identifying and debugging. To correct the problem, Author added dummy link with transformation *rpy="-1.57075 0 -1.57075"*.
- Author faced problem with initial mapping, where quality of map observed was very dull. Author increased the update rate which slightly improved the quality. But still it was no where near the quality of snapshot shared on Udacity project page. After going through online resources and Udacity forum, Author learned that problem is due to the collision for the Udacity provided world is not enabled. Author

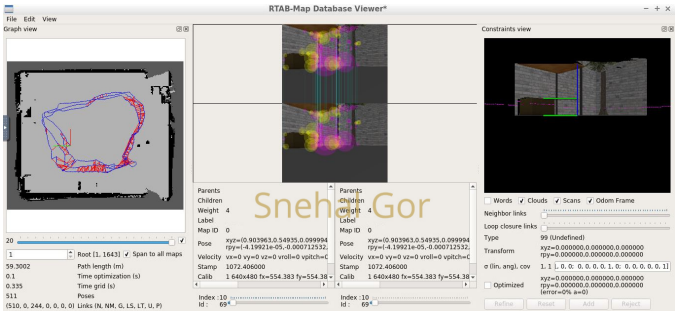

Fig. 10. RTABMap DB Viewer Snapshot for Author created Room Environment.

curled the new model using following command

```
curl -L https://s3-us-west-1.amazonaws.
    com/udacity-robotics/Term+2+Resources
    /P3+Resources/models.tar.gz | tar zx
    -C ~/.gazebo/
```

- Author tried to map Gas station environment constructed by him, which didn't go as expected. Author below listing potential reasons for the same
  - Gas station environment was quite large with open spaces.
  - Repetitive patterns combined with large open spaces and shorter sensor ranges, frequently gave false loop closures
  - Author kept Hessian threshold for SURF feature low, which affected the correspondence
- Learning from unsuccessful mapping of Gas station environment, helped Author while he faced false loop closure in a Author constructed room environment. Author increased Hessian threshold for SURF feature, which improved the performance.
- Author also learned / explored different SLAM approaches [4], GraphSLAM [5]

In Author's opinion RTABMap / GraphSLAM is a robust approach towards SLAM. Author faced problem in localizing in an environment having repetitive patterns.

6 Conclusion / Future work

Author has successfully constructed own world and performed SLAM with in two environments: Building own package, Mapping different topics, Tuning different parameters of ROS packages. With respect to RTABMap, Author would like to explore it further. Author would love to explore object and obstacle detection along with autonomous exploration and navigation. Author also sees RTABMap can be used for mapping of

- Simulated indoor environments like houses, factories, etc.
- For outdoor environments like campuses, urban environments, etc.

References

[1] *Wikipedia : Simultaneous$_l$ocalization$_a$nd$_m$apping*.
[2] *Setup RTAB-Map on Your Robot*.
[3] *Mobile Robot Kinemetics*.
[4] *SLAM Course - University of Bonn by Cyrill Stachniss*.
[5] S. Thrun and M. Montemerlo, "The GraphSLAM algorithm with applications to large-scale mapping of urban structures," *International Journal on Robotics Research*, vol. 25, no. 5/6, pp. 403–430, 2005.