3D Simultaneous Localization and Mapping with RTAB-Map

Rohan Paleja

***Abstract*—The simultaneous process of localization and mapping also known as SLAM is a powerful technique that provides a robot a high level of intelligence concerning its pose and surroundings. Real time appearance based mapping or RTAB mapping puts vision sensors to use alongside memory techniques to perform SLAM in real-time. This method was applied to a robot in a Gazebo environment and the results displayed the importance of loop closure and features. It was clearly seen that as the amount of features in the environment increased, the more closely the generated point cloud would look to the actual environment.**

***Index Terms*—SLAM, Robotics Nanodegree Program, Udacity, RTAB-Map, Mobile Robots.**

I. Introduction

A. Motivation for Study

Mobile robots play in key role in society today, and will have an even greater role in the future. With the ongoing work with autonomous driving cars, Unmanned Aerial Vehicles (UAVs), and autonomous underwater vehicles (AUVs), estimating the robot's pose (location, orientation) alongside mapping the environment is pivotal to safe operation and proper functionality. The extension from localization to solving the simultaneous localization and mapping problem adds a tremendous amount of data to the user environment. Performing SLAM with high accuracy, an unknown environment can be mapped almost perfectly in one pass. This information can then be used in processes of automation, obstacle avoidance, data collection, surveillance, or many other tasks that provide robots a higher functionality.

B. Problem Description

Simultaneous localization and mapping comes with several feats to overcome. The robot has to perform the task of estimating its pose and use this as an input when constructing the environment. The task to perform SLAM with a simulated robot in the Gazebo physics engine environment given that the robot has a hokuyo laser rangefinder and a RGBD camera requires choosing a specific algorithm (in this case, RTAB-Map), implementing it, and analyzing the output for high performance. In the coming sections, background knowledge will be presented that will assist in the choice of algorithm, a discussion of simulation formulation will be presented, and the results of this experiment will be explained.

II. Background

Several SLAM algorithms are available for usage in a robot equipped with a laser rangefinder and RGBD camera. Grid-based FastSLAM and RTAB-Map are few of the many algorithms available. Many of these algorithms work in a similar fashion but are able to work with different sensor inputs and use different techniques for memory management.

The inputs into these SLAM algorithms are measurements and controls received from the robot and the output is a map and the trajectory. Within the general area of SLAM, there are two smaller categories: the Online SLAM and the Full SLAM approach.

The online slam approach estimates the pose and map using the current control and measurements. Thus, each output is independent of the previous measurement. A much more powerful technique known as Full SLAM or offline SLAM estimates the entire path at once. This can be thought of as an integral of online SLAM. This way makes use of all previous measurements and control inputs. The probability functions are shown below.

$$OnlineSLAM : p(x_t, m, c_t | z_{1:t}, u_{1:t}) \tag{1}$$

$$OfflineSLAM : p(x_{1:t}, m, c_t | z_{1:t}, u_{1:t}) \tag{2}$$

where x refers to the trajectory, m to the map, c to the correspondence, z to the input measurements, u to the input controls, and the subscripts refer to the time. Correspondence describes the discrete motion of the SLAM algorithm. It is the aspect of the algorithm that identifies a relation between the previously detected objects and the current. It can be thought of as the question, "Have I been here yet?". On the other hand, the continuous aspect of the robot is more well known and is how the robot collects sensor and control data, and the estimation of pose and landmarks. Some challenges within SLAM are that the parameter space is highly dimensional. Each landmark results in a dimensional increase of the space. This results in an exponential increase of dimension over time, which is fatal to a robot where memory is limited.

A. Choice of Mapping Algorithm

Two algorithms were presented: FastSLAM and GraphSLAM. FastSLAM uses a particle filter approach to estimate the posterior and a low dimensional Extended Kalman Filter (EKF) to solve independent features and map the environment. More specifically, the particle filter used is the Monte Carlo Localization Algorithm. This solves the online algorithm, and is quick to implement. Grid-based Fast SLAM allows for the mapping problem to be reduced to mapping with known poses. The main flaw with this approach is that the landmarks must be known ahead of time, and thus, any arbitrary environment cannot be mapped. The combination of the MCL algorithm

and Occupancy grid mapping results in high accuracy if the trajectory converges. Moving to GraphSLAM, which will be the choice of algorithm in this paper due to its considerable advantages, the mobile robot can now solve the Full SLAM problem rather than the Online SLAM one. This algorithm is not landmark based and can handle changes in the environment. It has improved accuracy over FastSLAM and can handle a large number of features.

B. GraphSLAM

GraphSLAM uses a graph-based approach to represent poses, features, and constraints in the environment. There are two types of constraints, motion constraints and measurement constraints. Motion constraints tie together 2 poses and measurement constraints tie together a feature and a pose. As the robot moves, the number of constraints will quickly grow. Every constraint will pull the estimated pose to a certain state and the goal is to find the state where most constraints are satisfied.

This algorithm can be split into the front-end and back-end. The front-end of this algorithm is how to construct a graph using the odometry and sensor data. It assesses the features and adds the nodes to the graph. The back-end of this algorithm takes the completed graph with constraints and outputs a most likely configuration of poses and map features. This requires the use of graph optimization. Here, the idea of maximum likelihood can be used to find the parameter that best explains the outcome. This helps to identify the state and feature locations given motion and measurement observations. Examples of a one-dimensional system and associated cost function are easy to visualize but as dimensions increase, the process of obtaining an analytical solution are impossible. Thus, a numerical solution can be found much faster using known algorithms such as gradient descent, etc. Using the concepts of likelihood estimation alongside multi-dimensional data structures, the best estimate for the path and map can be recovered. The data structures, known as the information matrix and information vector, represent the certainty in a relationship between two poses, a pose and a feature, or two features. As features and poses are estimated, this ends up a matrix with over a million cells. This tensor can be sparse, and may have many zeroed elements after the robot travels through in environment. Making use of several techniques, such as variable elimination, the computation can be less intense. The steps, of constructing a graph, defining constraints, and solving a system of equations while linearizing poses is a very hard problem. Thus, a subset of GraphSLAM known as RTAB-Map is proposed.

C. Real Time Based Appearance Mapping

Real-time appearance based mapping or RTAB-Map uses vision sensors alongside loop closure for high quality mapping. Loop closure refers to a comparison between a previous image and the current. Thus, as the map grows, the number of images that must be compared grows as well. This growth is linear and must be optimized to be done in real-time for large environments. The visual odometry can come from any camera. In this case, an RGBD camera is used to calculate geometric constraints. A laser scanner can be used to refine this constraint.

1) Loop Closure: Loop closure is of the utmost importance when mapping an environment. Every loop closure results in improved accuracy and results. There are two types of loop closure, local and global. Many algorithms use local, where loop closure is done for a limited map region. Size and location of this region is determined by the uncertainty of position. If uncertainty is too high, SLAM will ultimately fail. Global loop closure is where the current location is compared with all previous location. If there is no match, a new location will be added to the map. If this process of matching takes longer than the acquisition time, then this map and process will become ineffective. RTAB-Map uses the global loop closure technique, which has higher accuracy for large environments than the local version.

2) Bag of Words: RTAB-Map uses a Visual Bag of Words technique. This is a method for obtaining features from an image. Each set of pixels in an image creates a feature, which is a unique descriptor of an image. RTAB-Map uses SURF or Speeded Up Robust Features to detect these features. Going more in depth, SURF compares small sub regions of an image and the corresponding pixel intensities. These regions are clustered repeatedly till they represent a visual set of words. Thus, now matching of an image can be quantified into how many words they have in common. Then, a standard Bayesian filter can be used to evaluate scores. If a certain threshold is passed, loop closure is detected.

3) Memory Management: The special memory management technique of RTAB-Map is vital to performing global loop closure in real time. RTAB-Map does this by keeping recent positions in working memory and others in long-term memory storage. A breakdown of this process is generalized below.

When an image is acquired, a new node is created in the short term memory. As this node is created, a bag of words is created for this node. Then, the nodes are assigned a weight based on how long the robot spent in that specific location. When this short term memory reaches its maximum predefined size, the oldest node is moved into the working memory. This is where nodes are tested for loop closure detection. After a certain period of time, nodes will be transferred to the long term memory, thus keeping the size of the working memory constant. The nodes in the long term memory is not used for loop closure detection and graph optimization. If a neighboring node of one that is in the long term memory is detected, then these nodes may be retrieved. Thus using this method, the complexity becomes constant as time increases. This allows for a highly optimized system that can grow very large in scale.

III. Simulations

To perform SLAM in a gazebo environment with the rtabmap-ros package, several steps were conducted. A mobile

robot, gazebo environment, and corresponding ROS package were built. After this, several outputs of the RTAB-Map including a 2D occupancy grid map, a 3D octomap, and a 3D point cloud were analyzed. Each of these outputs come with various benefits and negatives dues to their size and computation.

A. Building the Mobile Robot and Gazebo Environment

The benefits of creating mobile robots in simulation are that one has the freedom to quickly prototype different robots with different sensors and one does not have to worry about parts breaking due to crashes (if the algorithm or sensor fails, this will result). To create a mobile robot, a Unified Robot Description Format (URDF) file is created. This URDF specifies the visual, inertial, and collision properties including the location, shape, and origin. The mobile robot used contained two wheels, two casters, and a chassis. Sensor are then added to the vehicles. This vehicle makes use of a laser scanner and a RGBD kinect camera. Gazebo does have preexisting sensors, but since ROS is being utilized, plugins for each sensor must be used. Thankfully, many of these plugins can be found online and thus, one is created for each wheel joint for motion,one for the camera, and one for the rangefinder. The rtabmap package requires laser scan data, depth data, RGB image data, and odometry. A method can be used to remove the need for a laser scanner by converting the depth image to a laser scan, but in this case, the scanner will be used to output laser scans. A transform tree portraying the sensor and link relations is shown below.

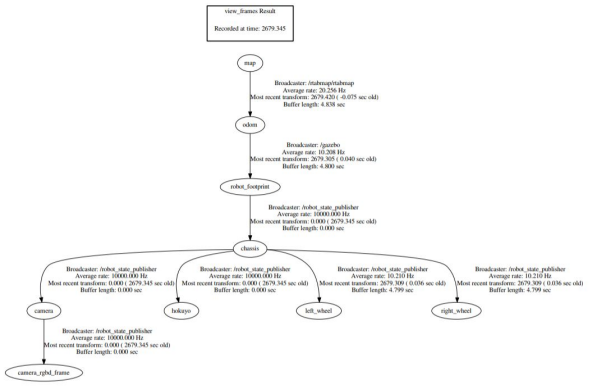

Fig. 1. Transform Tree of the Mobile Robot

Two environments were mapped throughout this experiment, a kitchen created by Udacity and a user-created environment. Pictures of the robot in each environment is shown below. The user-created environment was created by taken the default cafe model in Gazebo and adding several fixtures to it including two white blocks placed on the ground randomly, a bookshelf, and a cinder-block placed floating in the air. This world tested the RTAB-Map package performance in feature lax zones. This map had the characteristic as the robot may go large distances without seeing a significant number of different features.

Fig. 2. Vehicle in Udacity-Created Gazebo Environment

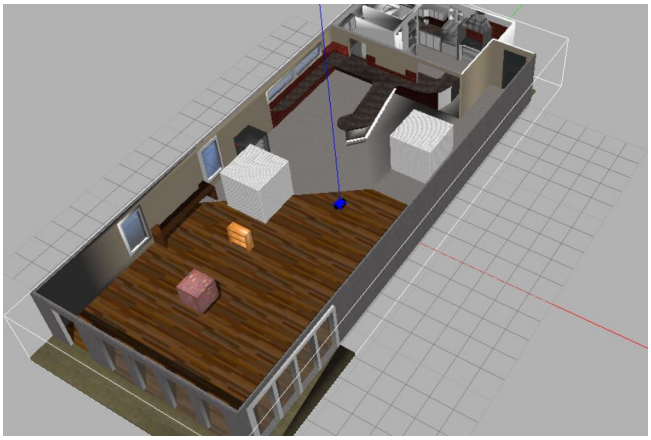

Fig. 3. Vehicle in User-Created Gazebo Environment

B. Launch Files

Several launch files were given by Udacity to help in the quick start-up of this project. A world launch file was created by the user to launch a gazebo world alongside a mobile robot. At this point, all the topics were mapped to their correct destination and this launch file would initiate them. After this, a mapping launch file was given by Udacity that would initiate the RTAB-Map package and open a RTABMap visualization window. After this, an RVIZ launch file will launch RVIZ and a teleop node will be launched to communicate velocity commands to the robot. A picture of the ROS node map after everything is launched is shown below.

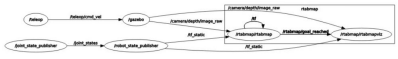

Fig. 4. ROS Node Map Created by Rqt Graph Package

IV. Results

Several experiments were conducted using these gazebo environments. Tests were completed where the robot would

go around once and the map would be analyzed. Another type of test was where the robot would perform multiple runs of an environment. The last time of type where the best results were obtained was where after every 3 meters the robot would complete a full spin. During this full spin, at least 1 loop closure would occur and thus, increase the accuracy of the map. This process was very lengthy. Simulations were first done on the Nvidia Jetson TX2, which proved extremely slow and unable to simulate the system without high latency. Thus, testing was moved to the Udacity classroom. Here, latency was still seen but was not nearly as bad as the Jetson. Below are the mapping results for several simulations. Note all the results are 3D point clouds. The 2D Contour is very similar to these and can be thought of as its projection onto the 2D plane. Pictures were taken in a time order to portray the growth of the point cloud. The results below portray the third method of taking data where the robot spins every three meters. The picture

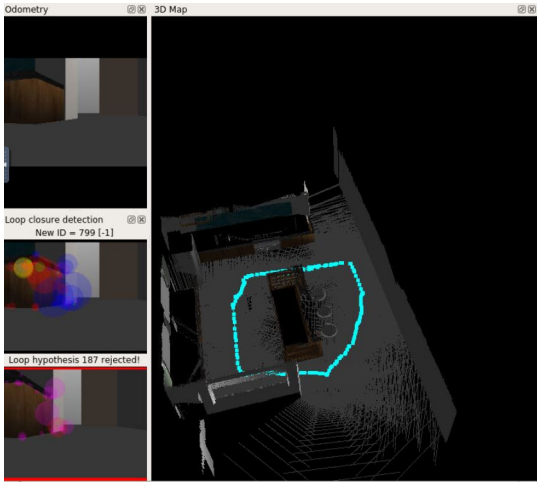

Fig. 5. 3D Map after the First Room has been Traversed by the Robot

above shows the RTAB-Map Visualization tool output after a loop around the table in the first room. It can be seen that little to no information is known about the second room. Next, one full loop around the second room is completed and the map is displayed again. While this looks like a very good map, subtle details such as sharpness or the edges and noise should be noted. The number of IDs can also be seen to be 768. After this, both rooms were traversed again and the map is shown again. After this, the ID is at 1834. The environment also looks a little cleaner with some object data more recognizable. This result is the most accurate of the three. It can be generalized that as more loops are conducted, the accuracy of the map is likely to increase. A visualization of the rviz display is also shown below. It is very similar to the output of the RTAB-Map Visualization tools.

This full simulation took around thirty minutes due to the latency which can be visualized by the teleop launch terminal. The keys that were not inputted into gazebo would be shown in this terminal and after the experiment, there was at least 200 characters displayed. The user-created environment was

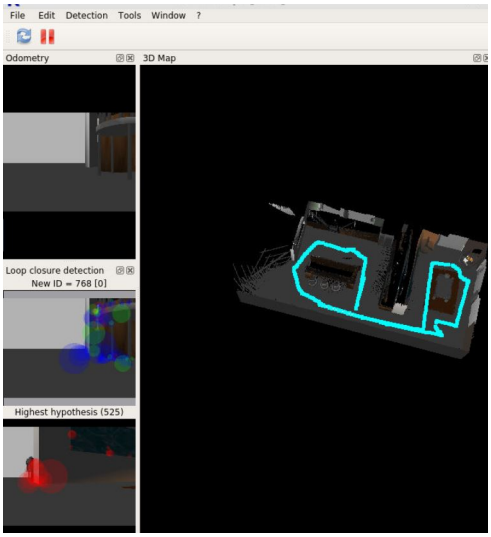

Fig. 6. 3D Map after the Full Environment has been Traversed by the Robot

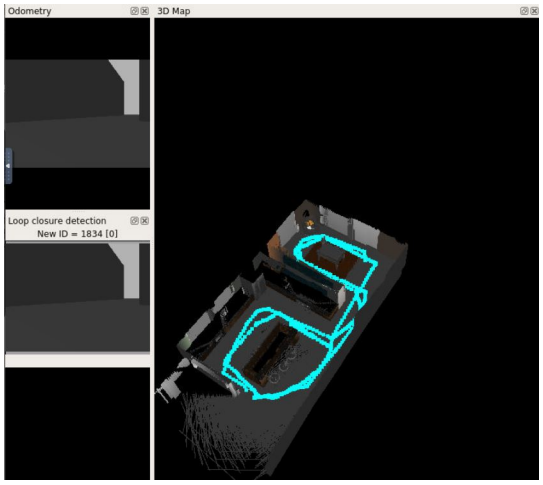

Fig. 7. 3D Map after the Full Environment has been Traversed by the Robot Twice

then tested using the same launch files with a slight change to the world launch file. The experiment was only conducted using the third method, where the robot would traverse three meters and then, perform a full spin and continue along a path. This simulation had more latency than the simulation of the kitchen. This may be due to the higher complexity of the cafe. The robot was spawned into an area with relatively many objects. After moving some distance, the map below was obtained. This map is fairly accurate. The rectangle shape of the cafe is kept intact and there is no considerable noise on the edges. At a point near this step, the setup at which the experiment is conducted is shown. This portrays the missed inputs by the teleop package. As the robot moves towards the other side of the cafe, where the color of the floor is nearer to that of the walls and objects, the accuracy seems to plummet. At this point, the accuracy began to decrease and the map gets worse and worse. Thus, mapping was stunted here and analysis

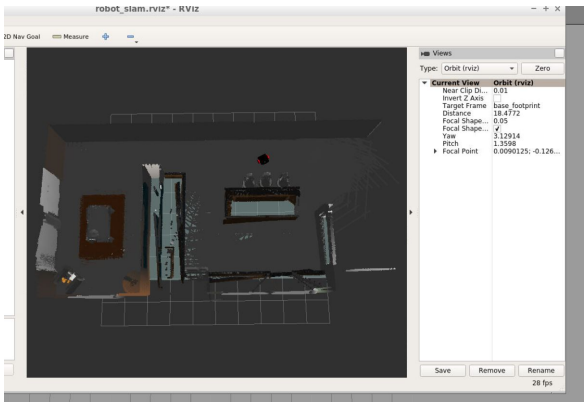

Fig. 8. 3D Map shown in Rviz after the Environment has been Traversed by the Robot Twice

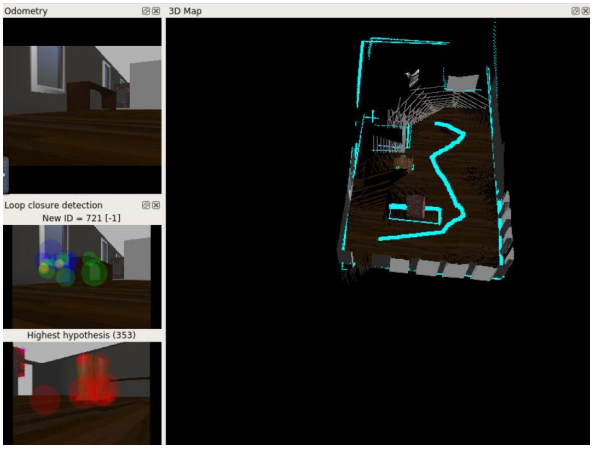

Fig. 9. 3D Map after Robot has Traversed some Distance in the Cafe.

was obtained. RTAB-Map cannot map monotone environments with little to no features. In areas with less features, odometry must carry much weight and odometry is prone to drift. An idea to explain the large inaccuracies could be RTAB-Map is unable to handle quick turns in feature-less environments. An example of these large inaccuracies due to repeated failure of loop closure is shown below.

V. Discussion

In this experiment, a robot was navigated using a keyboard around two environments. One common problem was that the robot would not move forward, and occurred during loop closure. To fix this, the number of max features could be reduced and the minimum inliers could be reduced. It was also seen that rotating the robot resulted in regaining the ability to move. The results in the kitchen environment was considerably better than the cafe environment. This was due to the excess amount of features in the kitchen world. With chairs, tables, and windowed walls, the robot was able to calculate features with ease and thus have much better visual odometry. The cafe world did have some objects but gaps were left in some areas to test how the robot would perform when features were lax. Also, the cafe had some areas were the colors of the

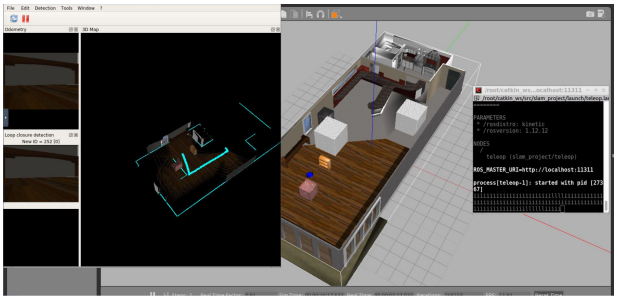

Fig. 10. Setup of Experiment.

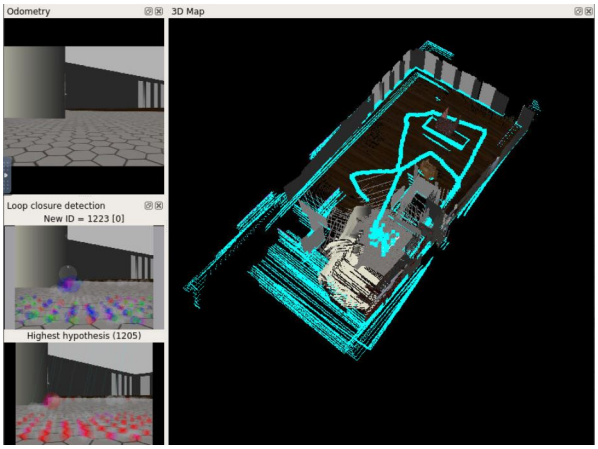

Fig. 11. 3D Map after Robot has Traversed into a monotone region of the Cafe.

surfaces were similar and this seemed to also deteriorate the results. It was seen that the kitchen model performed better and portrayed that rich features are vital in the operation of RTAB-Map. It was also seen that incorrect loop closure quickly leads to an experiment failure and further uncertainty.

VI. Conclusion / Future Work

The operation of SLAM is a very difficult and sensitive technique. The environment must fit the criteria of the SLAM algorithm for it to run at its highest performance. RTAB-Map proved to be very viable in feature rich environments and produce very meaningful results in a robot equipped with a laser scanner, an RGBD camera, and odometry. As previously stated, the laser scanner can be removed, and then this robot would be able to perform SLAM for a very cheap price. This type of SLAM would be very useful in mapping unknown areas for intelligence. Furthermore, much of this data could be processed and tested against Lidar datasets for object recognition as previously done in a Udacity Project. This type of SLAM could also be applied to a real household robot using the NVIDIA Jetson TX2. It was found difficult to implement simulations but it may be a worthwhile experiment to lower the computation and attempt to map a feature rich area.

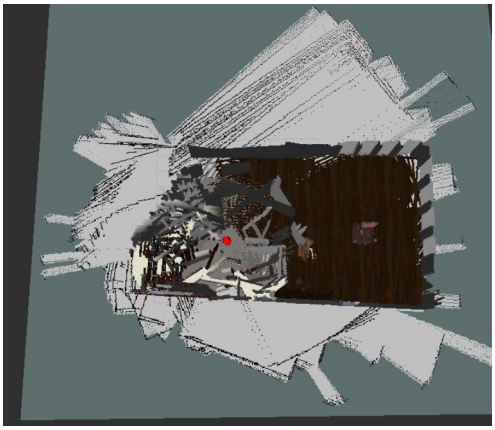

Fig. 12. Failure in Mapping due to Incorrect Loop Closure.

REFERENCES

[1] Thrun, Sebastian. Probalistic Robotics.. (n.d.). Retrieved April 14, 2018,

[2] Rtabmapros. (n.d.). Retrieved April 17, 2018, from http://wiki.ros.org/rtabmapros

[3] Grisetti, G., Kummerle, R., Stachniss, C., Burgard, W. (2010). A Tutorial on Graph-Based SLAM. IEEE Intelligent Transportation Systems Magazine, 2(4), 31-43. doi:10.1109/mits.2010.939925