# Map My World Robot

Scott Peng

**Abstract –** Simultaneous Localization and Mapping or SLAM is one important topic in the field of robot navigation. In this project, one powerful SLAM technique – RTAB-Map is implemented to the virtual robot to use for mapping and the unknown environment and localize itself. The one provided Kitchen-dinning simulation environment and another personalized environment were tested. The mobile robot model was further improved based on the previous localization project.

## 1. Introduction

Simultaneous Localization And Mapping (SLAM) is one of the fundamental and challenging problem in robotics. It is known as a "chicken or the egg problem" in robot navigation, as the robot pose are required for mapping, but the map is needed for localization. However, in most cases, both information are lacked, and the robot has to map the unknown environment while simultaneously localize itself.

SLAM problems can be categorize into two forms - online SLAM and offline SLAM. Online SLAM computes the map and pose at time instance t; whereas the offline SLAM, also called full SLAM, computes the map and pose of the entire path.

Another characteristic is the continuous and discrete nature of SLAM problems. Since the robot is continuously moving and taking measurement from the environment, the robot poses are continuous. On the other hand, whether or not the same location been visited, or an object has been seen by the robot, those questions considered as the correspondence of objects are discrete. Both continuous and discrete natures are important features while solving SLAM problems.

Several powerful algorithms were developed for SLAM problems. In this project, we will implement a virtual robot simulation with SLAM algorithm to map the created virtual environment. In the following section, we will briefly discuss the Occupancy Grid Mapping algorithm and two of the most common algorithms SLAM – FastSLAM and GraphSLAM.

## 2. Background

Before dive into the algorithm, we should realize the challenges in SLAM problems.

- Unknown map and poses - As mentioned before, the "chicken or the egg problem" of causes the first challenge. The algorithm will have to make some assumptions first and iterate through the process. During the process, one result will affect the other one.
- Large amount of data to process – due to the unknown map, there are a lot of hypothesis and large amount of variables. Depends on the map and sensor, data to be processed could exponentially growth.
- Noise – This is the nature when handling sensors or actuators.
- Perceptual ambiguity - when similar objects appear appeared, the robot might have a difficult time to distinguish and results in confusion of traveling to the same location even though it is not.
- Cycles – another problem for the noisy sensor and actuator data that the accumulated odometric error can be large after many iteration through the process.

### 2.1 Occupancy Grid Mapping

The Occupancy Grid Mapping Algorithm (OGMA) is one of the popular mapping algorithms in robotics that can be used to map arbitrary environments. The algorithm first divide the 2-dimensional space into a finite number of uniform small cells. Each of the cell holds a binary value that represent the occupancy of the location it covers – *1* indicates an obstacle was detected at the cell, *0* means no obstacle was detected. Then OGMA uses a Binary Bayes Filter to estimate the occupancy of each cell. The algorithm takes the previous occupancy values of the cells, the poses and the measurement as parameters, and loops through all the grid cells.

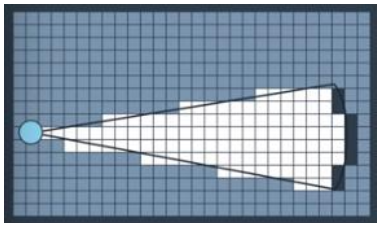

Fig. 1 Measurement Cone in Occupancy Grid Mapping

The main use of this algorithm, however, is for post-processing. Because SLAM techniques do not produce maps that are suitable for planning and navigation, occupancy grid mapping is often used after solving the SLAM problem by other means and taking the resulting path estimates given.

### 2.2 FastSLAM

FastSLAM solves the full SLAM problem with known correspondences in the following phases:

- Estimating the trajectory: FastSLAM uses a particle filter approach and solves the mapping problem with known poses.
- Estimating the map: FastSLAM uses a low dimensional Extended Kalman Filter to solve independent features of the map, which are modeled with local Gaussian.

This approach assume that the environment has known landmarks, and this is one of the disadvantages of FastSLAM. However, with the grid-mapping algorithm the environment can be modeled using grid maps without predefining any landmark position. An instance of FastSLAM is Grid-based FastSLAM which adapts FastSLAM to grid maps. Instead of tracking the landmark positions, using grid map to solve for arbitrary environment.

There are three key techniques are used to implement Gird-bases FastSLAM:

- Sampling Motion: Estimate pose given the k-th particle previous pose and current control
- Map Estimation: Estimate map given the measurements, k-th particle pose and the previous k-th particle map
- Importance Weight: Estimate the likelihood of the measurement given the k-th particle pose and k-th particle map

### 2.3 GraphSLAM

GraphSLAM is another approach to solve the full SLAM problem by using a graph representation. The graph is constructed nodes and edges. The nodes represent the robot poses or map features. The edges connecting those nodes are constraints that characterize their relations between poses (motion) and/or between the surrounding environments (measurement). The goal of GraphSLAM is to find a configuration of the nodes that minimize the error introduced by the constraints, using a principle known as Maximum Likelihood Estimation (MLE).

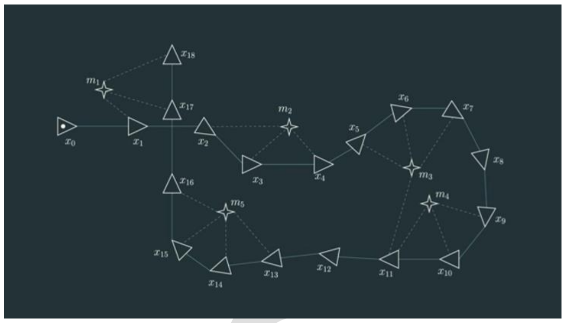

Fig. 2 graph with poses, map feature and constrains

GraphSLAM algorithm can be considered as two sections:

- Front-end: continuously added new nodes and constraints to the graph
- Back-end: optimize the most probable configuration of the poses and map features, and update the graph

In this project, we use RTAB-Map, which is a GraphSLAM approach that uses loop closure detection, a process indicating that a location has been visited before. Which in turn uses a bag-of-words approach to extract features from images, compare them, and associate some sort of linkages to these images that have a number of common features above a predefined threshold number.

### 3. Scene and robot configuration

The personalized virtual environment was created by using the “building editor” in gazebo package. Based on the experience using the provided *kitchen_dining.world*, complex world model consumes a lot of computational power and results in significant delay when operating the robot. Thus, a very simple and fully enclosed room with two separated areas was created. In addition, a few obstacles, such as coffee table, cabinet…etc., were added to the environment to provide enough features for the robot detection.

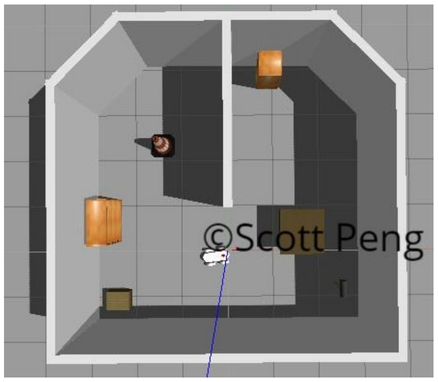

Fig. 3 personalized room with robot

The robot model created from the previous project was used with a few improvements. The main change was to use Kinect RGB-D sensor instead of the original non-depth camera, which was a necessary upgrade for the RTAB-Map algorithm.

The project structure is also organized based on the previous project with a few new files added. Main sub-folders and files are briefly described below:

**slam-project**

- **Launch**: containing launch files for different tasks (world, robot-description, mapping, teleop, rviz-confg)
- **Meshes**: containing the robot mesh model
- **Scripts**: containing the provided scripts (rtab-run and teleop)
- **Urdf**: containing the robot urdf file
- **Worlds**: containing the world files for the testing virtual environements
- CMakeLists.txt
- Package.xml

To execute the package, simply run the provided *rtab-run* file and follow the dialog to select world file and enable mapping.

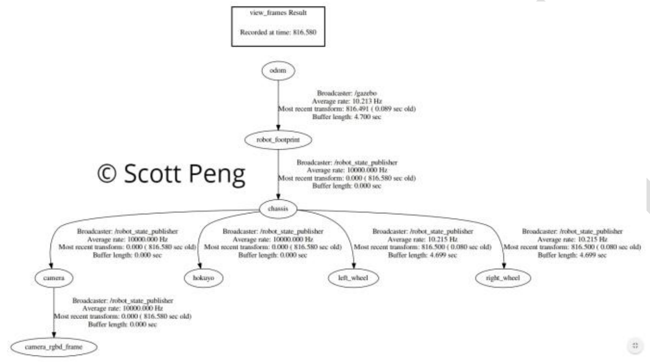
view_frames Result
Recorded at time: 816.580
odom
Broadcaster: /gazebo
Average rate: 10.213 Hz
Most recent transform: 816.491 ( 0.089 sec old)
Buffer length: 4.700 sec
robot_footprint
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 ( 816.580 sec old)
Buffer length: 0.000 sec
chassis
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 ( 816.580 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 ( 816.580 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10.215 Hz
Most recent transform: 816.500 ( 0.080 sec old)
Buffer length: 4.699 sec
Broadcaster: /robot_state_publisher
Average rate: 10.215 Hz
Most recent transform: 816.500 ( 0.080 sec old)
Buffer length: 4.699 sec
camera
hokuyo
left_wheel
right_wheel
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 ( 816.580 sec old)
Buffer length: 0.000 sec
camera_rgbd_frame

Fig. 4 ROS tf frame

## 4. Results

### 4.1 Kitench_dining

Fig. 4 below shows the gazebo simulation with robot in the kitchen_dining world environment. Fig 5 shows the RViz and RTAB-Map tool display real time images captured from the sensors, the 2D and 3D results during robot maneuvering.. Fig. 6 shows the detected data in the Database Viewer, which indicates 128 global loop closures in the end.

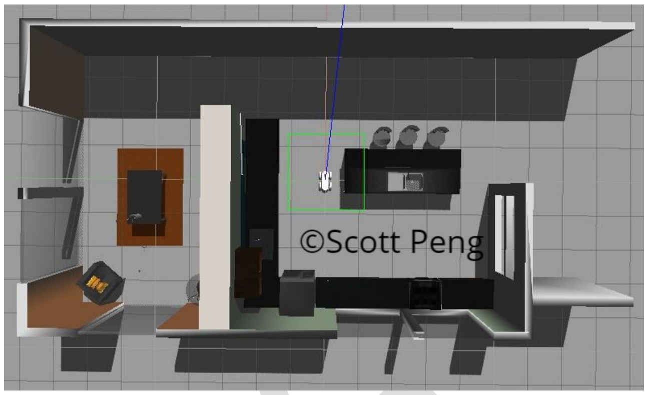

Fig. 5 kitchen_dining world with robot

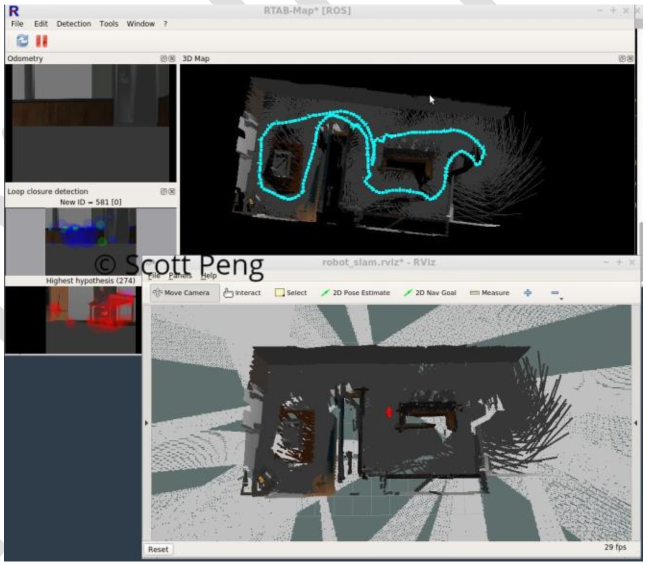

Fig. 6 RTAB-Map for kitchen_dining world

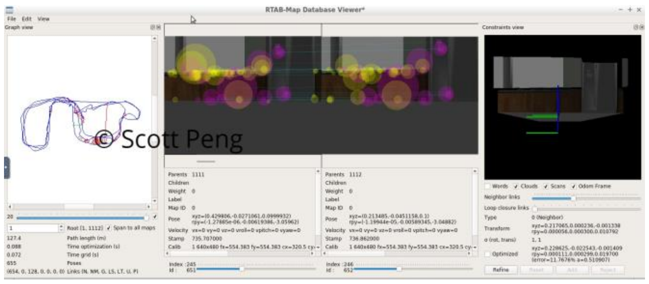

Fig. 7 RTAB-Map Database Viewer with kitchen_dining.world

### 4.2 Personalized room

Fig. 7 below shows the gazebo simulation, the RViz, and RTAB-Map tool for the personalized room. Fig. 8 shows the detected data in the Database Viewer. It also shows the Occupancy Grid for the detected map. In the end, there are five global loop closures were found from the collected data.

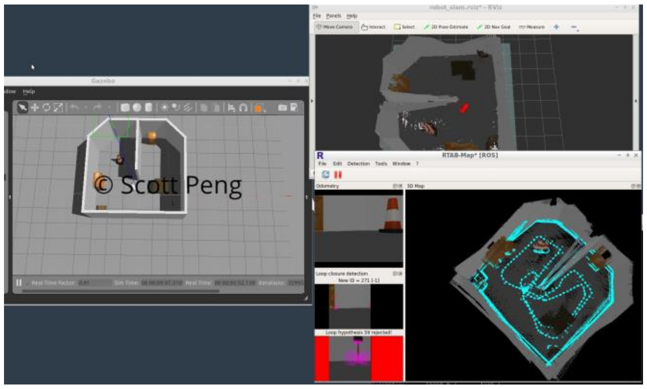

Fig. 8 RTAB-Map for personalized room

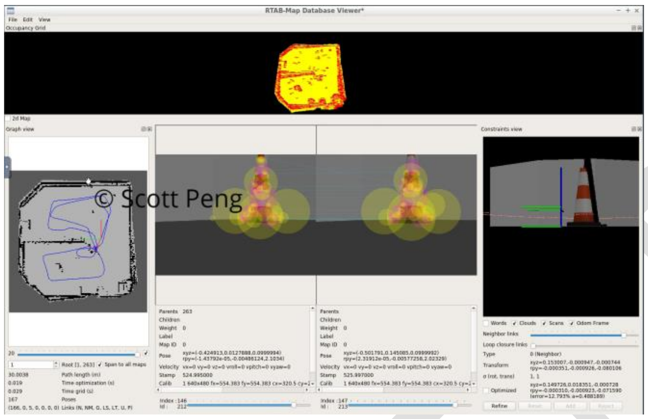

Fig. 9 RTAB-Map Database Viewer with personalized room

## 5. Discussion

Overall, the Graph SLAM algorithm is successful at mapping an unknown environment and localizing itself within it. However, due to the computation power, significant delay causes robot operation in the complicated environment such as the kitchen-dining world very difficult. Simplifying the world model greatly improves the simulation efficiency but also reduces the environment features captured.

Another issue during the mapping is that, in some cases, location discrepancy occurred randomly, causing the mapping results "jumped" and overlapped to each other. One suspected reason might be one of the update frequencies in one of the packages. This happened more often when turning the robot in place using key "J" or "L" rather than using key "U" and "O".

Another important factor affecting the mapping result is the sensor location. The RGB-D camera and laser scanner have to be placed at the locations can provide enough perception area. Unlike the previous localization project, where the camera can set to look at the bottom of walls to provide good tracking result, in this project, environmental features from all height are very important.

## 6. Future Work

Based on the foundation built in this project, we could use the same RTAB-Map on a real mobile robot. Firstly to deploy on a mobile rover or a drone to explore a room, an apartment, or a building since those are closer to the simulation setup, such as enclosed space, relative stable lighting condition. We might need to repeatedly adjust hardware setup, such as the camera position, and many of the configuration parameters, such as the particle amount, potential map size… etc. Until being successful in handling the algorithm in the buildings, we can then test the hardware in open space with more unknown objects.

Mapping and localization both are very important techniques for mobile robots navigation. Especially, since SLAM could handle unknown environment, it extends possible application in the areas where environmental information was not available ahead. For example, exploring the unknown terrain in the hidden valley, cave, or deep-sea, executing rescue activities in the dangerous environment where human could not access. Eventually this technique could be one of the core package in any mobile robot application.