Map My World Robot

Joe Tanton

Abstract—This project discusses the SLAM(simultaneous localization and mapping) problem and why it is important in robotics. Issues and challenges associated with the problem are highlighted and different approaches of SLAM are identified, including FastSLAM and GraphSLAM. Using ROS and Gazebo, a simulated robot equipped with a laser rangefinder and RGB-D camera is used to implement a form of GraphSLAM called Real-Time-Appearance-Base-mapping (RTAB-map) to generate 2D and 3D maps of two different environments. Both environments are successfully mapped, however the first environment is mapped more accurately due its layout and clearer features. It was found that an environment that has a lot of recurring features throughout the map caused issues with correspondence leading to poor mapping results. It is concluded that the work carried out in this project could easily be transferred to real world robots.

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

1 Introduction

For robots to be useful in society they must be able to move in an environment that it has never seen before. As the robot moves it must construct a map of the environment while simultaneously localizing itself relative to this map. This requires simultaneous localization and mapping which is referred to as SLAM. This is more challenging than localization and mapping since neither the map or the robots poses are provided. In this project, a SLAM algorithm is implemented using ROS to simulate a robot mapping a 3D environment in a gazebo world. 3D SLAM is achieved with RTAB-Map (Real time appearance based mapping) which utilizes a RGBD camera and a laser range finder mounted on the model robot. The robot is driven around two environments multiple times to create a map using RTAB-Map that can be viewed in real-time or once the robot has completed its journey. [1]

2 Background

When solving localization problems the map is known and when solving mapping problems the exact pose of the robot is known. However, in most real world problems both the map and the robot pose are unknown. Therefore this requires simultaneous localization and mapping or SLAM. SLAM aims to map the environment and localize the robot relative to this map given noisy measurements and commanded controls. Not knowing the robot pose or the map and having to compute these makes this problem challenging due to its continuous and discrete nature. The continuous nature of the problem is having to keep track of all the many robot poses and objects in the map. As time and map size increases, the number of variables increases making the problem highly dimensional and difficult to compute. The discrete nature of SLAM is also highly dimensional, as the algorithm has to identify if a relation exists between any newly detected and previously detected objects in the environment. This is referred to as correspondence. There can be a large number of correspondence variables which can make the problem highly dimensional. Therefore due to the nature of SLAM, the algorithms chosen will have to rely on approximations using numerical solutions rather than analytical methods to reduce computational memory. Solving SLAM is one of the most fundamental problems in mobile robotics. In real world applications it is not feasible to provide the robot with a map of the exact environment as the environment is constantly changing in real time. This is certainly true in self driving cars. [1] Two common types of SLAM are FastSLAM and GraphSLAM:-

2.1 FastSLAM

The fastSLAM algorithm uses a custom particle filter approach to solve the full SLAM problem with known correspondences. Using particles, fastSLAM estimates a posterior over the robots trajectory along with the map. Each of these particles holds the robots trajectory and a map and each feature is represented by a local gaussian. FastSLAM uses a low dimensional extended Kalman filter to solve these independent features to estimate the map. The disadvantage of FastSLAM is that is assumes that there are known landmark positions which is not usually the case in real-world problems. This issue can be solved using Grid-based FastSLAM which extends FastSLAM to occupancy grid maps. The algorithm implements parts of the MCL algorithm along with the occupancy grid mapping algorithm. [1]

2.2 GraphSLAM

Another type of SLAM is GraphSLAM which solves the full SLAM problem. This means that the algorithm recovers the entire path and map instead of the most recent pose and map. This difference allows it to consider dependencies between the current and previous poses. GraphSLAM has the benefit of reducing the need for significant onboard processing capability and has improved accuracy over FastSLAM. GraphSLAM constructs and uses graphs to represent the robots poses and environment and tie these together using motion and measurement constraints. These graphs are then numerically solved to give the most likely robots path and map of the environment. [1]

2.3 RTAB-Map

Real-time appearance based mapping or RTAB-Map is a 3D GraphSLAM algorithm approach. The algorithm uses vision sensors to localize the robot and map the environment and is based on an incremental appearance-based loop closure detector. The loop closure detector uses a bag-of-words approach to determine how likely a new image comes from a previous location or a new location. When a loop closure has been found, a new constraint is added to the maps graph. As the robot travels through a expanding map, more and more images must be compared for loop closure which causes high computational load. To overcome this, the algorithm uses multiple strategies for memory management to allow for loop closure to be done in real time. [2]

3 World Creation

To demonstrate a robot performing SLAM, a robot was placed in two different gazebo worlds. The first world called Kitchen dining was provided and can be seen in Figure 1. The second world is a custom design, see Figure 2. It is a rectangular room with a number of common objects placed alongside the walls to create features in the environment to help the mapping process. It is sort of a garage/workshop with a man working on robotic arms and drinking lots of beer!

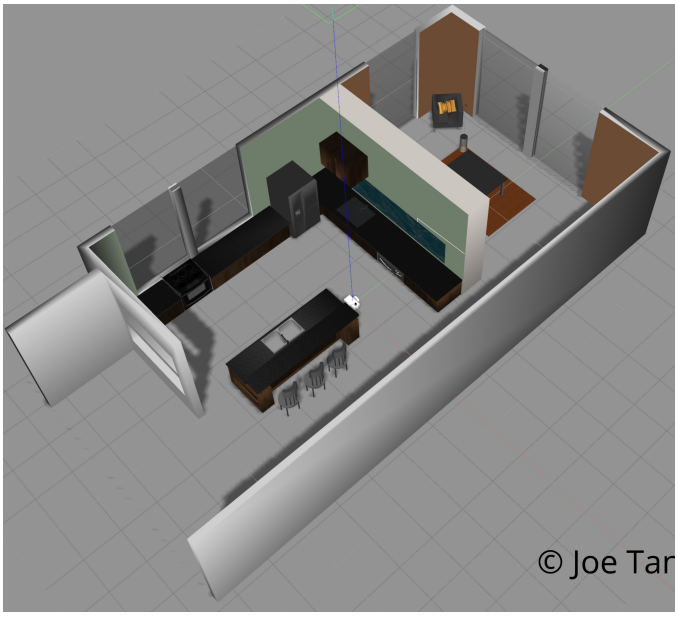

Fig. 1. Kitchen dining world

Fig. 2. Garage world

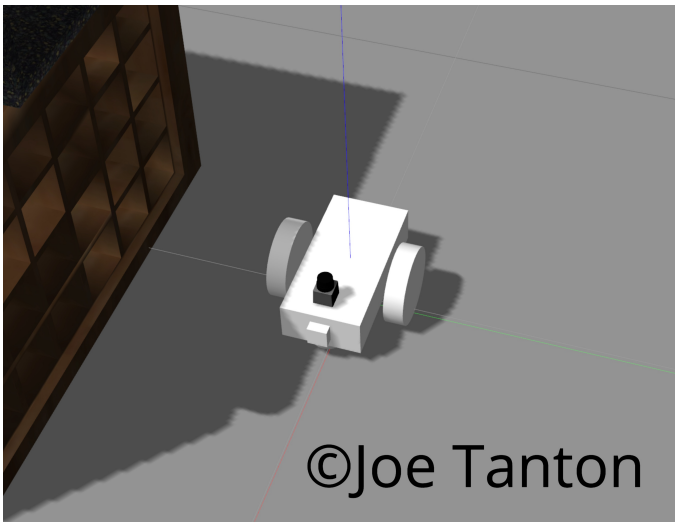

Fig. 3. Udacity_bot

4 Model Configuration

The robot model used in the simulation was carried over from the previous project, Udacity_bot. The only change to the model was the introduction of a RGB-D camera positioned in the same place as the original camera. The laser rangefinder and odometry sensors are unchanged. See figures 3 and 4 for robot model and the transform frames for the robot links.

The simulation uses a custom ROS package called slam_project. The package structure is as follows:- xarco, gazebo, world files to create the world and robot. launch files to spawn the robot in the gazebo world, start the teleop node to drive the robot and launch the mapping node to implement RTAB-Map. The necessary topics were remapped and all parameters were unchanged from the ones provided.

5 Results

Figure 5 and 8 show the mapping process. In each map the robot was driven around the map 3 times. In both environments the robot was delayed/slow to drive around the map due to the computation load and hardware limitations. The

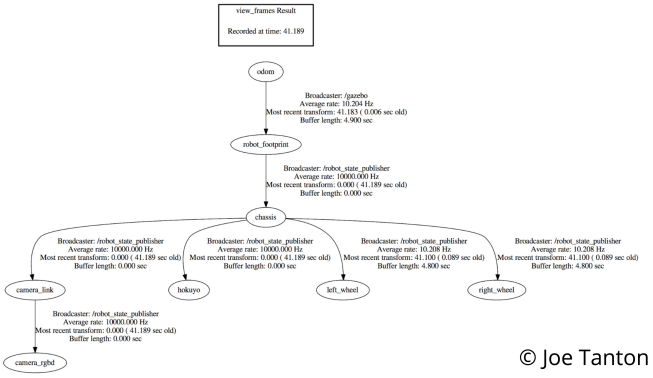

Fig. 4. Transform Tree

garage map was particularly slow and steering was very delayed when running the simulation on the Jetson TX2.

5.1 Kitchen dining world

The below figures show the results of the mapping process for the Kitchen dining world. Figure 6 shows the 2D map and displays that 113 loops closures were detected. Figure 7 shows the 3D map.

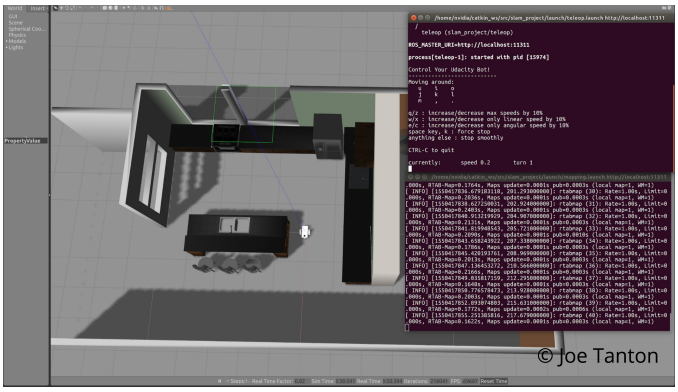

Fig. 5. Mapping Kitchen dining

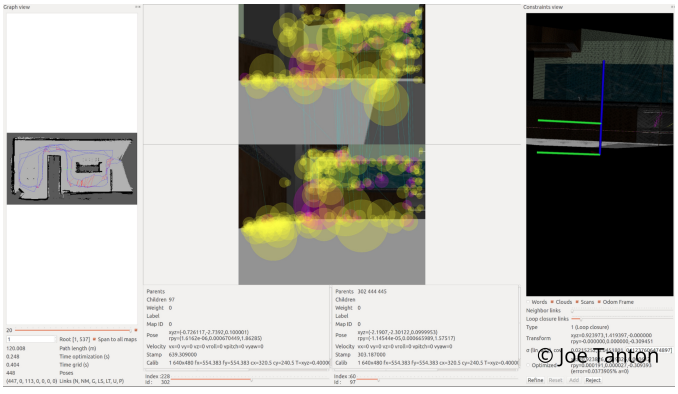

Fig. 6. Kitchen dining results

5.2 Garage world

The below figures show the results of the mapping process for the Garage world. Figure 9 shows the 2D map and displays that 59 loops closures were detected. Figure 10 shows the 3D map.

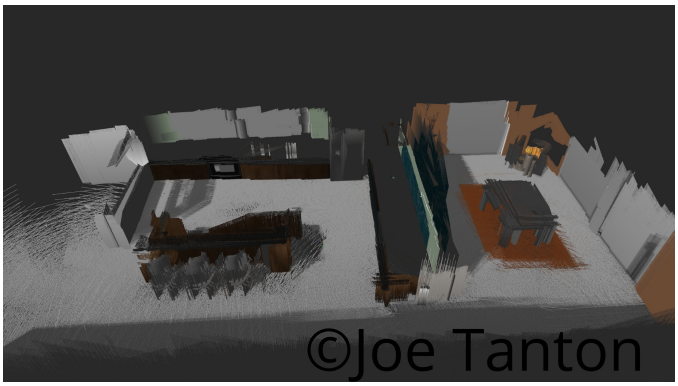

Fig. 7. Kitchen dining 3D map

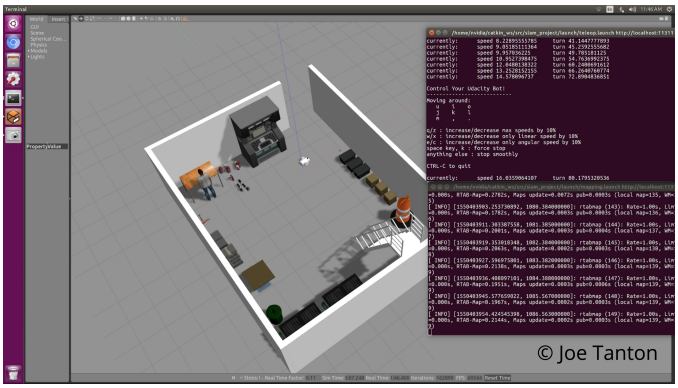

Fig. 8. Mapping Garage world

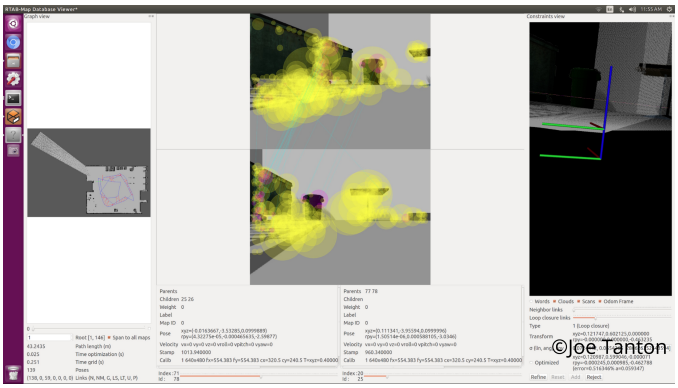

Fig. 9. Garage world results

6 Discussion

As can be seen from the results, both environments were successfully mapped and are recognizable from the 2D and 3D generated maps. The 2D maps represent the environments accurately and clearly display the outline of the map from a top down perspective. The 3D maps on the other hand were less accurate but still represent the environment. It can be seen that features in the final 3D maps are repeated/offset, representing the error between

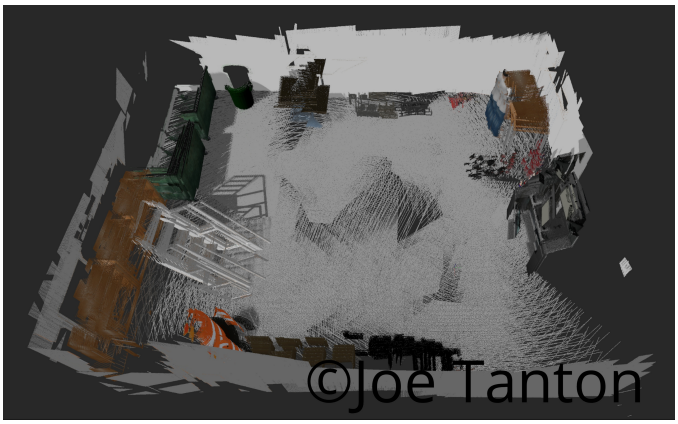

Fig. 10. Garage world 3D map

each pass. Mapping the kitchen dining environment was easier and more successful than mapping Garage world. Although the kitchen dining map is bigger and looks more complicated, there is a lot more distinguishable features and it is a denser environment. This facilitates in more successful loop closures improving the quality of map. This is evident from the results with the 115 loop closures for the Kitchen dining world and 59 loop closures for Garage world. When creating Garage world a lot of trial and error with the design took place to allow the robot to map the environment successfully. Initially, the walls and floor had a stone texture/detail which made the mapping process difficult due to there being multiple similar features throughout the map. This caused a lot of false positive loop closures confusing the mapping process and also increasing computational load. When mapping Garage world, the mapping accuracy may have been affected by the repeated objects in the world, i.e the row of cardboard boxes. Avoiding these features could improve the mapping process. Other improvements to Garage world could be made by increasing the number of objects and making areas more distinguishable. Overall mapping accuracy for both maps could be improved by exploring and tuning the parameters for RTAB map algorithm. Additionally, multiple cameras could be mounted on the robot to not only increase the accuracy but decrease mapping times.

7 Conclusion / Future work

The work demonstrated in this project could be transferred to real world projects. The RTAB-map mapping algorithm could be deployed on a Jetson TX2 and could be mounted on any real world mobile robot with a suitable RGB-D camera. For example a robot vacuum cleaner or a robot lawnmower. Ones feels that the work done in this project could easily be used in one of these applications without much modification. More demanding SLAM problems such as self driving cars could utilize the RTABmap algorithm but would require further development. This is due to the complexity and nature the problem presents. For example, driving on a long road with a continuous tree line would cause many false positive loop closures in the RTABmap algorithm.

References

[1] *Udacity RoboND classroom material*.
[2] *http://introlab.github.io/rtabmap/*.