Map My World

Ghee Chong Foo

Abstract—In this write up, Simultaneous Localization and Mapping (SLAM) technique is used to perform mapping. In particular, the Real Time Appearance Based Mapping (RTAB-Map) which utilizes Graph-SLAM algorithm. A Udacity provided kitchen environment was mapped, followed by a custom environment. In both cases a robot equipped with depth camera was driven around manually to perform mapping with at least 3 loop closures on the rtabmap database.

Index Terms—Robotics, SLAM, RTAB-MAP.

1 Introduction

Simultaneous localization and mapping (SLAM) is a technique in robotics to construct and update a map of an unknown environment while at the same time keeping track of whereabout of the robots [1]. This appears to be a chicken-and-egg problem and resolved through approximation using method such as particle filter, extended Kalman filter and Graph-SLAM. In this project RTAB-Map (Real-Time Appearance-Based Mapping) that utilizes RGB-D Graph-SLAM approached will be deployed, which is based on global Bayesian loop closure detector. The loop closure detector uses bag-of-words approach to determine the likelihood of new image comes from a previous location or a new location [2]

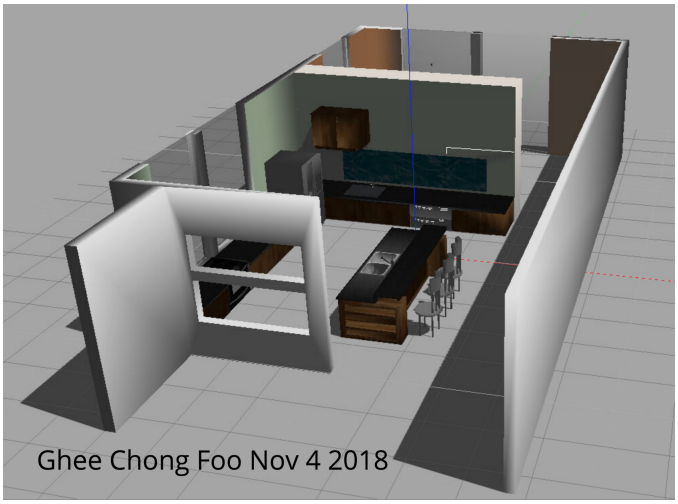

Fig. 1. Kitchen World

2 Background

In localization, the problem is estimating the robot's poses given some landmarks, while in mapping the landmarks is estimated given the robot's poses. SLAM take a step further by trying to estimate both the robot's poses and landmarks **simultaneously**, and hence the name. Although this has been perceived as chicken-and-egg problem, it is fundamental requirement for robot to be truly autonomous and the basis for most navigation systems.

Two general approach to SLAM problem are Full SLAM and Online SLAM. In Full SLAM the algorithm tries to estimate the entire path, while Online SLAM seeks to recover only the most recent pose. The differences between the algorithms is how they differentiate already detected features to keep the map consistent, i.e. solving loop closure problem.

One of the popular SLAM algorithm is FastSLAM, which decomposes the SLAM problem into localization and landmark estimation problem that are conditioned on robot pose estimate. FastSLAM uses a modified particles filter for estimating posterior over robot paths. By representing maps as a grid, one can get away with the landmarks based approach and estimate a map for any arbitrary environment. It is a combination of robot pose estimation using particle filter and occupancy based mapping algorithm with known poses.

In this project, Graph-SLAM algorithm will be employed. The algorithm represent the SLAM problem as a graph with poses of the trajectory and measurement locations as nodes as well as using estimated motion and measurement distances as links. The links in the graph are the constraints between the robot poses and its environment represent their relationship. Graph-SLAM then tries to resolve all of these constraints by creating the most likely map on the given data. In particular, the Real Time Appearance Based Mapping (RTAB-Map) is used, which is Graph-SLAM approach that uses bag of words approach to identify correspondences between frames using visual similarity for optimization. The camera image is compared against already known map features allowing the algorithm to map trajectories with repeated locations and thus reducing constraint complexity.

3 Scene and robot configuration

The initial environment can be reproduced by running "create_workspace.bash" script in Github depo, which consists of following commands:

```
#!/bin/bash

mkdir -p ~/catkin_ws/src
```

```
cd ~/catkin_ws/src
catkin_init_workspace
cd ~/catkin_ws
catkin_make; source devel/setup.bash
cd ~/catkin_ws/src
git clone
https://github.com/ggnohc/RoboND-MapMyWorld.git
cd ~/catkin_ws
catkin_make; source devel/setup.bash
```

The udacity provided kitchen dining world and custom world, and the relevant services can be launch through following:

- roslaunch udacity_bot udacity_world.launch
- (or) roslaunch udacity_bot Office.launch
- roslaunch udacity_bot teleop.launch
- roslaunch udacity_bot mapping.launch
- roslaunch udacity_bot rviz.launch

5 launch file has been created/provided as below:

1) **udacity_world.launch**: Launches the kitchen dining world and the robot model
2) **Office.launch**: Launches the custom model (to be elaborated later)
3) **teleop.launch**: Launches the teleop movement to control robot movement manually
4) **mapping.launch**: Launches RTAB-map services to providing mapping and closure
5) **rviz.launch**: Launches RViz for robot_slam.rviz configuration

3.1 Scene configuration

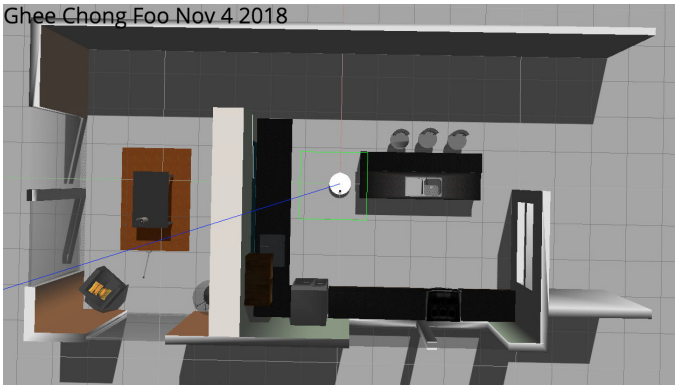

Fig. 2. Kitchen World

Figure 2 depicts top view of Kitchen Dining World as part of student material.

Figure 3 depicts custom created Gazebo world, which is based on Gazebo OSRF Elevator model, and populated with miscellaneous items such as standing person, cafe table, dumpster etc. for mapping purpose.

3.2 Robot configuration

The robot is modelled after house hold vacuum cleaning robot, having a cylinder shape and depth camera mounted at the top and camera mounted at the front, as in Figure 4.

Figure 5 shows the ROS Graph for the robot model.

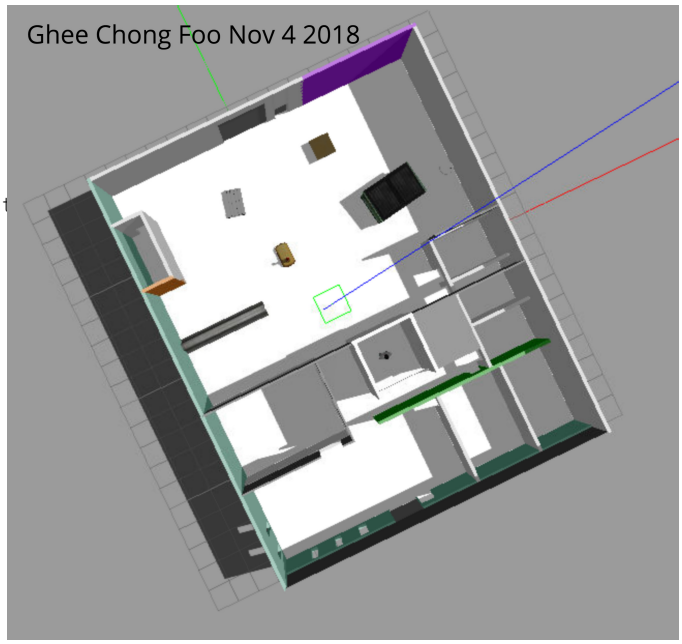

Fig. 3. Office World

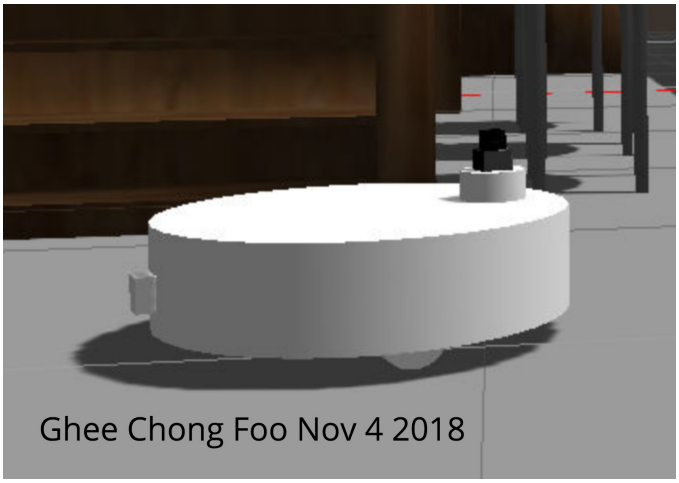

Fig. 4. Robot Model

4 Results

The robot was driven manually through teleop control, often passing through the same area multiple times to make sure the area was detected by observing the point cloud in Rviz. In both world the minimum required of 3 loop closure was achieved.

The RTAB-MAP database is archived in https://drive.google.com/open?id=1Orfo77AVvk9UFrvDMlLDa_U68EGVkIPb

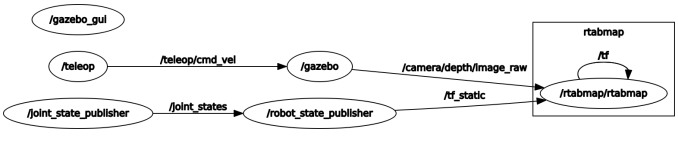

Fig. 5. rosgraph for robot model

4.1 Kitchen Dining World

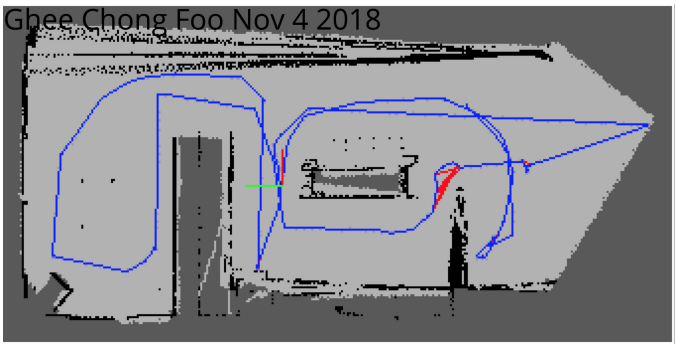

Fig. 6. Kitchen 2D

Figure 6 shows the path that robot has pass through while mapping the Kitchen world.

Fig. 7. Kitchen 3D

Figure 7 shows the surrounding mapped by the robot presented in 3D.

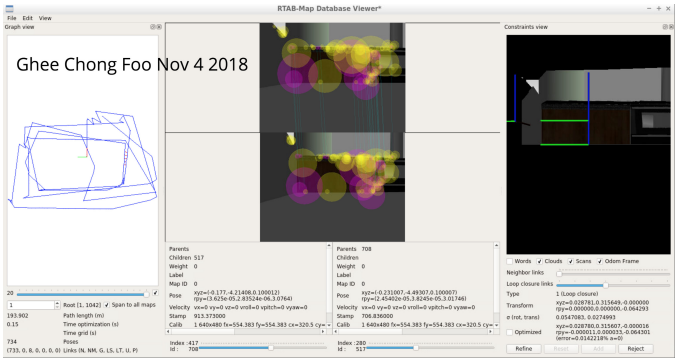

Fig. 8. Kitchen RTAB Map

Figure 8 shows the output of RTAB Map database viewer. 8 loop closure has been achieved in this case.

4.2 Custom Model

Figure 9 shows the path that robot has pass through in Office world.

Figure 10 shows the Office world mapped by the robot presented in 3D.

Figure 11 shows the output of RTAB Map database viewer. 7 loop closure has been achieved in this case.

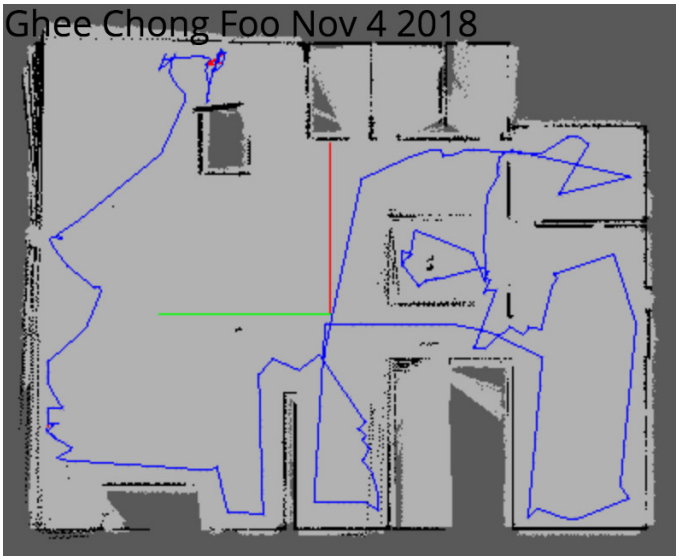

Fig. 9. Office 2D

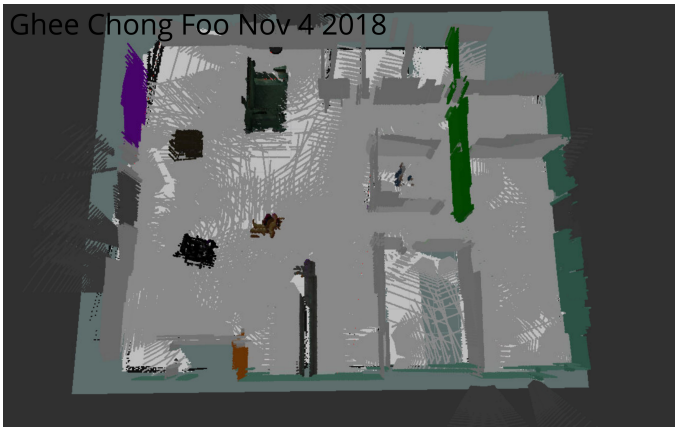

Fig. 10. Office 3D

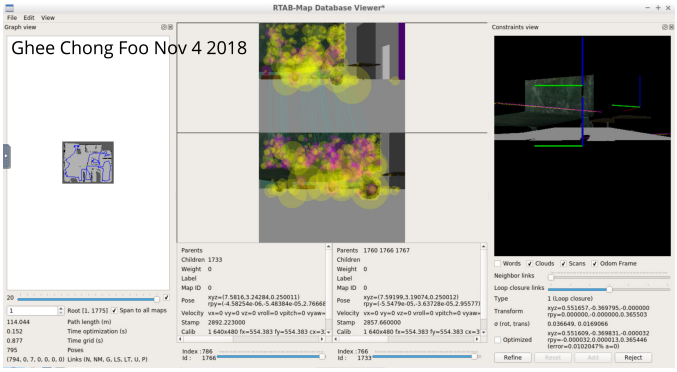

Fig. 11. Office RTAB Map

5 Discussion

The Kitchen world was easier to map as it contain object with different sizes and shapes, which is feature rich enough for the RTAB-Map algorithm to identify the environment. In contrast for the custom office environment, initially it was hard for the robot to map out the environment as it was repetitive and feature-less, which confuses the algorithm and not able to recreate the actual environment. Once more unique objects are placed then only the robot was able to perform the mapping correctly.

The map for both world can be further improved by going through the environment several times, by covering angles that was previously missed. However one will quickly discover that the size of the RTAB-Map database grows quickly and might not scale in a large and complex environment.

6 Future work

RTAB-Map is an interesting application of GraphSLAM algorithm. Future work would include running it in a actual mobile robot instead of simulation, which has to deal with real world noise such as those coming from camera and actuator due to imperfect sensor and environment. While RTAB-Map works well in mapping static and feature rich environment, more works are needed to deploy the robot in a dynamic environment, such as busy hospitals with people constantly moving around versus static walls or rooms.

References

[1] "Simultaneous localization and mapping," 2018. Last edited 2018-09-15.

[2] M. Labbe, "Real-time appearance-based mapping," 2018. Last edited 2018-05-06.