Map My World Project Writeup

Gang Wang

Abstract—In this work we set out to experiment Simultaneous Localization and Mapping (SLAM) [1] for our mobile robot in its simulated working environment. The robot was driven manually around a given world in Gazebo to construct its 2D occupancy grid map and 3D point cloud map. The goal of this project is to gain a better understanding of GraphSLAM, as well as how to leverage RTAB-Map to create a map, and to debug problems efficiently when they arise.

Index Terms—Robot, Udacity, Localization, Mapping, Simultaneous Localization and Mapping, SLAM, Real-Time Appearance-Based Mapping, RTAB-Map.

1 Introduction

MAPPING along with localization are the most fundamental tasks for a mobile robot to navigate to its goal destination autonomously. In a realistic mobile robot application, the working environment is usually unknown or ever changing. To navigate through its working environment successfully, it's essential to maintain a relatively up-to-date map for the robot.

In this project, we first created a mobile robot running in Gazebo equipped with a laser rangefinder and a RGB-D camera. It explored the world in loops to sense its running environment. The obtained sensor data were then fed into Real-Time Appearance-Based Mapping (RTAB-Map) ROS package to construct a sparse constraints graph of its surroundings. The main objective of this project is to leverage RTAB-Map to build 2d and 3d maps of two worlds (shown in Fig.1 and Fig.2) respectively.

Fig. 1. kitchen-dining world.

2 Background

The simultaneous localization and mapping (SLAM) problems arise when the robot does not have a map of the environment, nor does it know its own pose. There are two main forms of the SLAM problems. One is called *online SLAM*, which estimates the posterior over the momentary pose along with the map. Another form is known as *full SLAM* that seeks to calculate a posterior over the entire path.

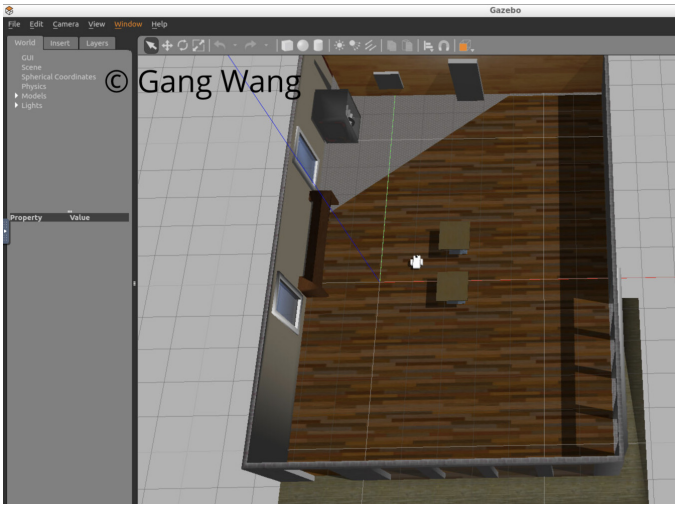

Fig. 2. custom-built world.

Various SLAM approaches have been published in academia. One of the popular full SLAM solution is GraphSLAM [2], which extracts a set of soft constraints represented by a sparse graph from observed sensing data. It then obtains the map and the robot path by resolving these constraints into a globally consistent estimate.

This project employs GraphSLAM, in particular the RTAB-Map [3] approach to generate a 2D occupancy grid map, as well as a 3D point cloud map of the working world for our robot. The basic idea of RTAB-Map is to use a RGB-D camera and a Lidar, incrementally hypothesizes *Loop Closures* from sensing images. This loop closure determines how likely a new image comes from a previous location or a new location. So once a loop closure hypothesis is accepted, a new constraint is added to the maps graph. With limited number of locations for loop closure detection and graph optimization, RTAB-Map performs in real-time for large-scale environments.

3 Scene and Robot Configuration

3.1 Robot Model Configuration

The robot in this work inherits the *udacity_bot* that's been developed in previous **Localization** project [4]. In order to perform appearance-based RTAB-Map, a Kinect RGB-D camera and a camera link, detailed in Table2 and Table1, are added to the robot. The whole robot configuration is defined in udacity_bot.xacro and the transform tree associated with the robot is shown in Fig.3.

Name	Origin	Parent link	Child link
optical_joint	xyz="0 0 0" rpy="-pi/2 0 -pi/2"	camera_link	Kinect

TABLE 1
Changed Gazebo model joints

Name	Origin	Geometry
camera_link	xyz="0 0 0" rpy="0 0 0"	box="0.05 0.05 0.05"
camera_link_optical (Kinect)	xyz="0 0 0" rpy="0 0 0"	box="0.05 0.05 0.05"

TABLE 2
Changed Gazebo model links

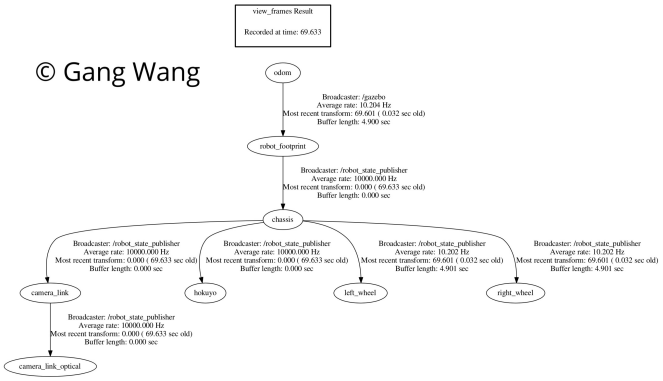

Fig. 3. Transform Tree Frame.

3.2 Customized World

The customized world was created in Gazebo based on the cafe model. Since the robot can not enter the kitchen, a customized wall with a door is erected before the kitchen, which also narrows the mapping world in effect. Two cafe tables are put in the middle. Fig.4 shows how it looks like from above.

4 Results

We drove our mobile robot around the worlds in Gazebo manually. The robot navigated through the entire working world to collect as many surrounding feature images as possible. By the time a full 3D environment has been generated in Rviz, we had following results.

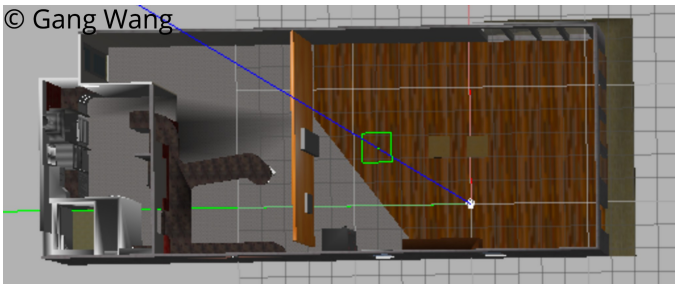

Fig. 4. An overview of the customized world.

4.1 Kitchen-Dinning world results

A 2D occupancy grid map of the *Kitchen-Dinning* world is shown in Fig.5. Corresponding 3D point cloud map is shown in Fig.6. The RTAB-Map database is put in Goole drive [5].

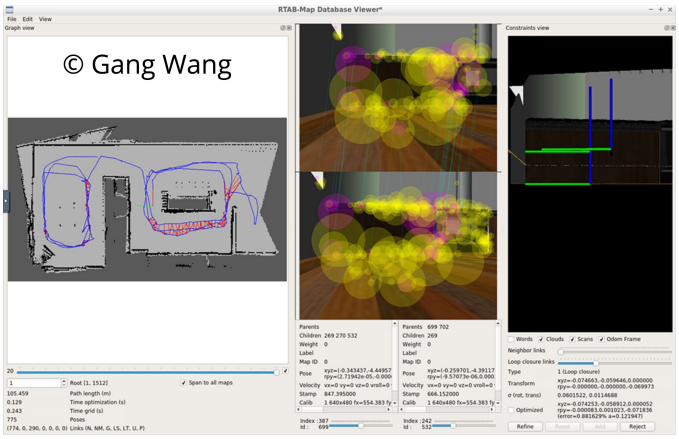

Fig. 5. 2D occupancy grid map of the kitchen-dinning world.

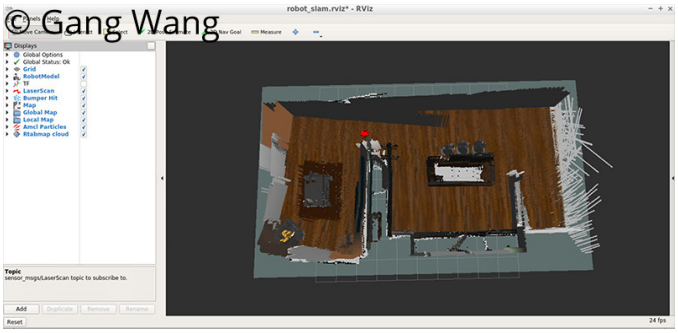

Fig. 6. 3D point cloud map of the kitchen-dinning world.

4.2 Customized world results

A 2D occupancy grid map of the *custom-built* world is shown in Fig.7. Corresponding 3D point cloud map is shown in Fig.8. The RTAB-Map database is put in Goole drive [6].

5 Discussion

As we can see in the resulting maps, especially in Fig.5, when our robot explored the world with many rotations,

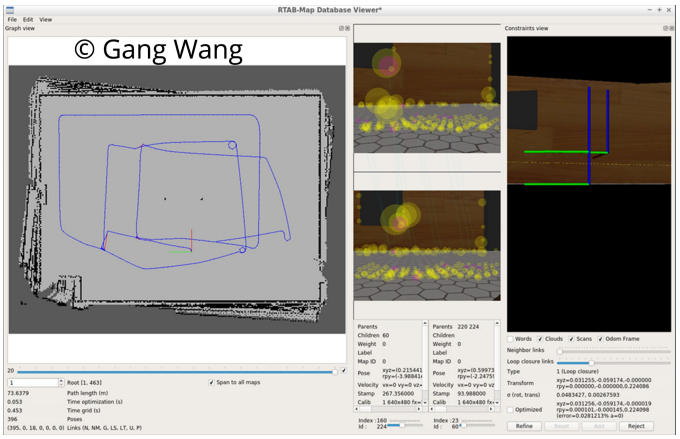

Fig. 7. 2D occupancy grid map of the custom built world.

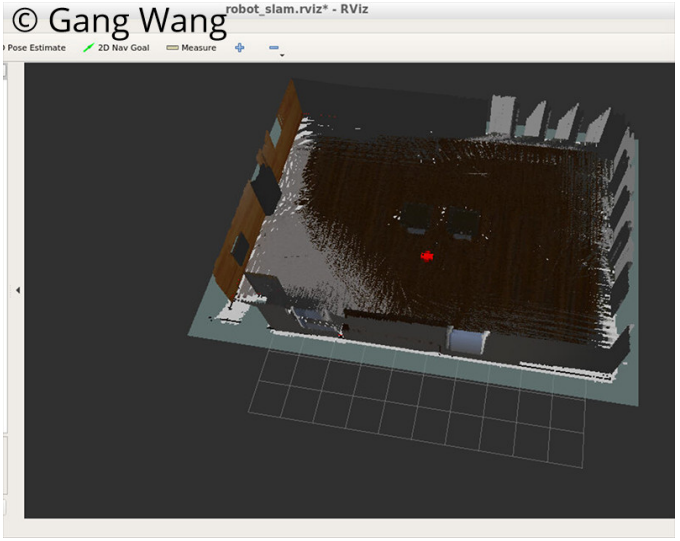

Fig. 8. 3D point cloud map of the custom built world.

our final maps also rotated accordingly (depicted in RED trajectory in Fig.5 and Fig.7). This suggests a moderate surveying motion, in particular avoiding sharp turning, should be employed to improve mapping accuracy.

We experimented this project in a relatively lower-end laptop. One obvious observation is it requires more time to collect images in order to map the whole world. And the resulting maps are highly screwed, meaning they suffered from much more map rotations to the extend that they are barely discernible. We are tempted to conclude that SLAM in general is a computation intense process and more computational powers such as faster a CPU/GPU, a higher volumes of memory are preferable.

Last but not least, compared our customized world to the given kitchen-dinning world results, distinctly feature-rich environment improves the final results a lot. One possible explanation might be more distinctive features makes an easier loop closure detection, considering the burden of computationally is negligible.

6 Future Work

One interesting future work would be to enable localization for our mobile robot, making it navigate to a goal position autonomously.

Because RTAB-Map can generate a 2D and a 3D point cloud map, one potential application would be to detect, classify objects from the 2D image, then using the 3D point cloud data to estimate object pose. Given the opportunity, one is tempted to try it out on a Pick & Place robot, to see if it would have any help in detecting objects and estimating end-effortor pose accurately and efficiently.

References

[1] Wikipedia, "Simultaneous localization and mapping." https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping. [Online].
[2] S. Thrun and M. Montemerlo, "The graphslam algorithm with applications to large-scale mapping of urban structures." http://robots.stanford.edu/papers/thrun.graphslam.pdf, 2006. [Online].
[3] M. Labb, "Real-Time Appearance-Based Mapping." http://introlab.github.io/rtabmap/. [Online].
[4] G. Wang, "Where Am I project." https://github.com/samguns/RoboND-Project6-WhereAmI/. [Online].
[5] G. Wang, "Resulting RTAB-Map file of kitchen-dinning world." https://drive.google.com/open?id=15z01hD7aNLq56pKdXdZk7b2ff7YXMvrf. [Online].
[6] G. Wang, "Resulting RTAB-Map file of customized cafe world." https://drive.google.com/open?id=1mjG2zMGmf-4ITe-xosGx1H5fAldC6OX9. [Online].