# SLAM Project Writeup

Mohammed Asfour

**Abstract**—This is a writeup illustrating the results of the project of SLAM using the RTAB-MAP ROS package. It illustrates the process of map generation by looking for loop closures -similar features- of the point clouds obtained by the camera and how the robot localize itself according to the map, SLAM problem domain, future work will be discussed as well.

**Index Terms**—Robot, IEEEtran, Udacity, LaTeX, SLAM.

## 1 Introduction

The problem of robot exploration in general is very important nowadays all across different situations, from rovers exploring planets to self-driving cars sensing local short distance environments, as the robot finds itself in an unknown environment with unknown position and aims to generate the environment map and localize itself while traversing it. The robot sensors data are prone to noise, and its controls and actuations contain error. Here is where SLAM comes in, it stands for Simultaneous Localization and Mapping, it have different techniques and mainly based on Bayesian statistics. We will be using Online SLAM and GraphSLAM in this project. they will be discussed further in the paper.

Please note that all the images used here in the writeup are included in high quality in the project directory with a video of the robot passing the required task regarding the provided environment.

## 2 Background and Formulation

Solving the problem of mapping by itself requires known poses of the robot, on the other hand, the problem of localization needs the ground truth map to be solved, that's why the SLAM problem is called the chicken or the egg problem. Among the different forms to preform SLAM, the most commonly used are GraphSLAM and Grid Based FastSLAM.

### 2.1 Nature of SLAM

#### 2.1.1 Online SLAM

SLAM can be preformed on the go, while the robot is exploring and using only the data collected at that moment only, Online SLAM, which is used here. This yield the pose f the robot and the accumulated explored part of the map at each instance.

#### 2.1.2 Full SLAM

SLAM can be preformed after the robot finished exploring and using all the collected data at all time intervals, Full SLAM. this gives the full robot trajectory and the whole explored area map.

### 2.2 Forms of SLAM

#### 2.2.1 GraphSLAM

The GraphSLAM approach uses a graph to represent estimated poses and location of landmarks obtained by the sensors, and ties them together using elastic bonds -estimated distances- that connect the consequent poses due to motion and/or landmarks sensed by the robot at that time. Then, knowing that both the robot motion and the sensors data contain errors, the algorithm tries to optimize the the best configuration of these poses and landmarks from the ideally controlled robot motion's pose estimation and the sensors readings by adjusting the placement of poses and landmarks and the elastic bonds that connect them.

#### 2.2.2 FastSLAM

This form uses a particle filter to solve the full SLAM problem if known correspondences are given. Using this filter, this form will be able to estimate a posterior pose of the robot, which enables it to then solve the mapping problem. As stated before, this forms requires known correspondences ,known landmarks, so it won't be able to solve the SLAM problem for any environment in general.

#### 2.2.3 Grid Based FastSLAM

This form builds on the FastSLAM one, utilizing a grid map solution, it can solve the SLAM problem for any environment without the need for known landmarks by following these steps

1) Previous Belief
2) Sampling Motion
3) Importance Weight
4) Map Estimation
5) Re-sampling
6) New Belief

## 3 Scene and Robot Configuration

### 3.1 Robot Model

A roomba like design was used for the custom robot. For the visualization , a cylindrical base was used, with combination of half a sphere on top of it serving as the hardware, while the camera is on the front of its base shown by a rectangle. I removed the LiDAR from the robot used in the previous project. But as the cylindrical base is the biggest shape in this design, it's the only geometry included in the collision to minimize calculations.

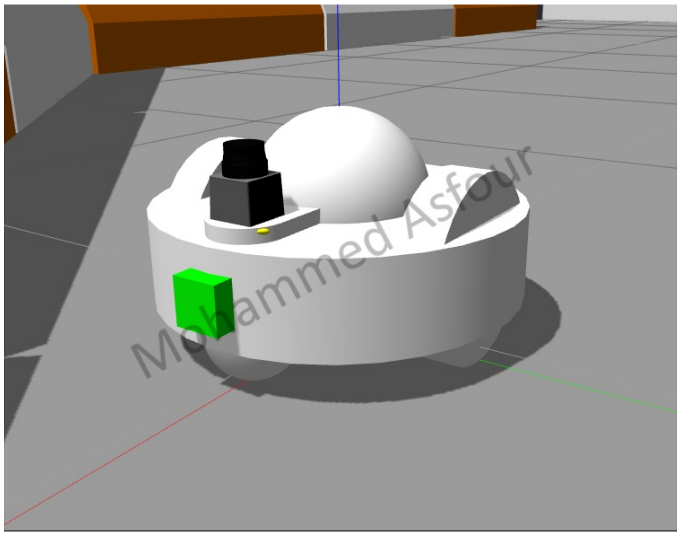


Fig. 1. Old Robot Model

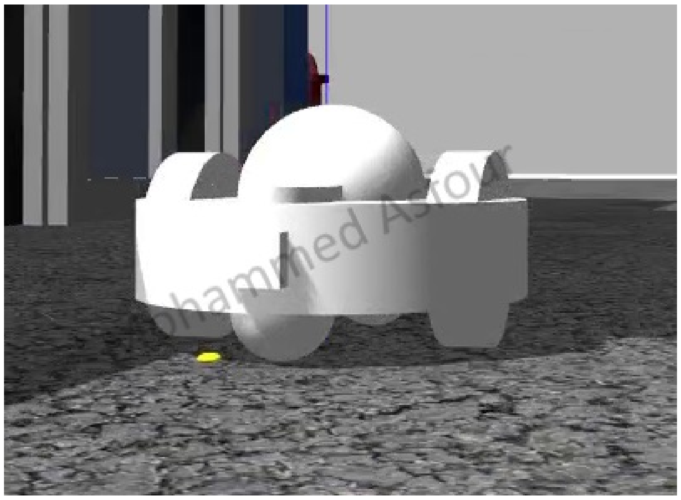


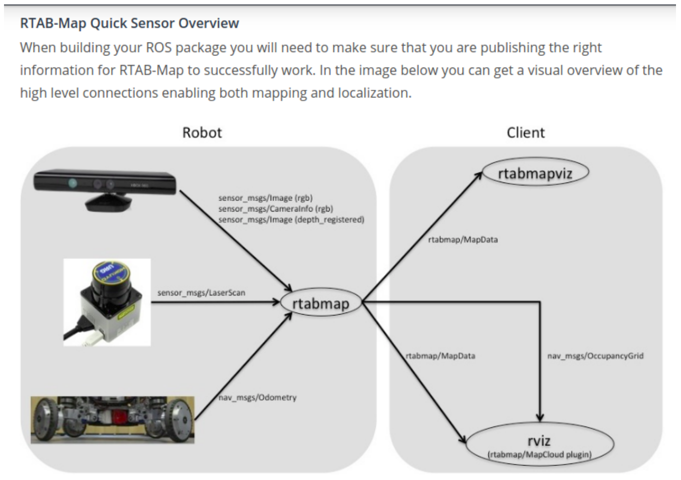
Fig. 2. New Robot Model

## 3.2 Sensors Selection

Even though the rubric says that a laser sensor, didn't specifically say LiDAR, the project sections explain what is wanted is laser sensory data provided to the rtab-map package alongside the camera feed to perform SLAM. The sections go on and explain two ways to achieve that

### *3.2.1 RGB-D Cam with LiDAR*

This is the old sensors configuration I implemented last project. I kept the code for the LiDAR in the xarco file but commented it out, as I won't be using this method because it doesn't provide anything new regarding the robot model.



**RTAB-Map Quick Sensor Overview**

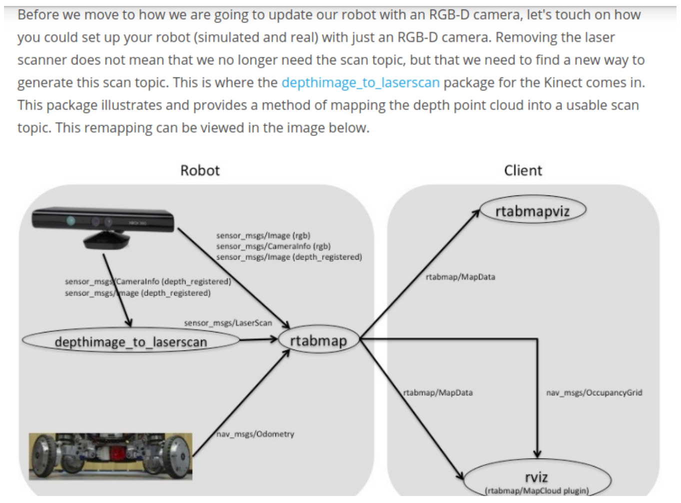
When building your ROS package you will need to make sure that you are publishing the right information for RTAB-Map to successfully work. In the image below you can get a visual overview of the high level connections enabling both mapping and localization.

Fig. 3. RGB-D with LiDAR

### *3.2.2 RGB-D Cam Only By Using Depth to Provide Laser Scan Data*

This is a more convenient method for me as I want o implement this project in real life, and I can't access a LiDAR, and it's a more challenging way to provide the Laser Scan. This is the method I used in this project.

By using the depth info the depth-to-laserscan package can pass the required laser sensory data to the /scan topic to the rtab-map package.



Before we move to how we are going to update our robot with an RGB-D camera, let's touch on how you could set up your robot (simulated and real) with just an RGB-D camera. Removing the laser scanner does not mean that we no longer need the scan topic, but that we need to find a new way to generate this scan topic. This is where the depthimage_to_laserscan package for the Kinect comes in. This package illustrates and provides a method of mapping the depth point cloud into a usable scan topic. This remapping can be viewed in the image below.

Fig. 4. RGB-D Only

## 3.3 Rqt Graph

The following is the rqt-graph for the whole project. As seen in the graph, it complies with the second method mentioned by using only a RGB-D camera. With the aid of the depth-to-laserscan package, the laser data is provided

## 3.4 TF Tree

This tree shows the frames and how they relate to each other in the robot. I used a separate frame originating from the chassis frame for the depth-to-laserscan package

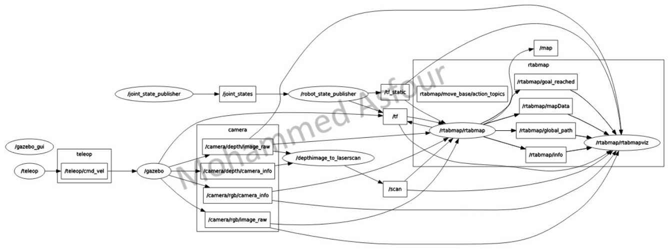


Fig. 5. Rqt Graph

to use as a frame to publish the laser data from to look like the same structure if LiDAR was used. I made another rame under the camera called camera-compensation-frame, tha's because the rtab-map package published the 2D map correctly but the point cloud obtained were rotated, so the frame is rotated to correct the orientation of the point clouds.

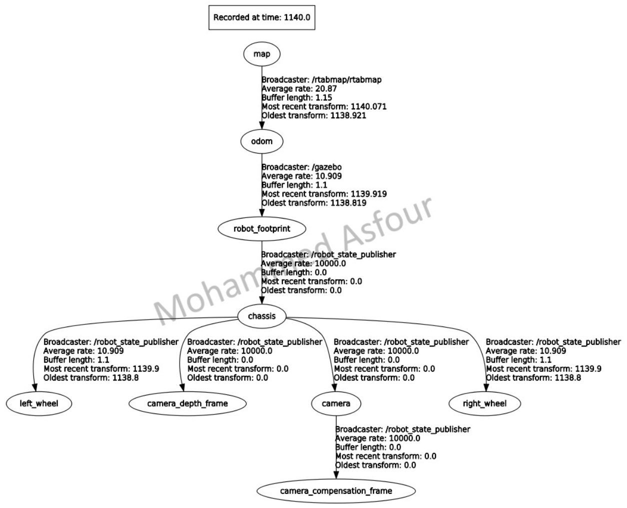


Fig. 6. TF Tree

### 3.5 Custom World

Instead of mapping inside environment again, I decided to try out an outside environment, that's why I choose a regular size building with a pickup truck outside of it. The models used are

- Post office
- Walking human
- Lamp post
- Pickup truck
- 45 degree ladder
- Dumpster
- Fire hydrant

### 3.6 Altered Parameters

For the customized world, the robot incorrectly found loop closures which resulted in wrong distorted map so by altering

- The number of inliers needed to detect a loop closure

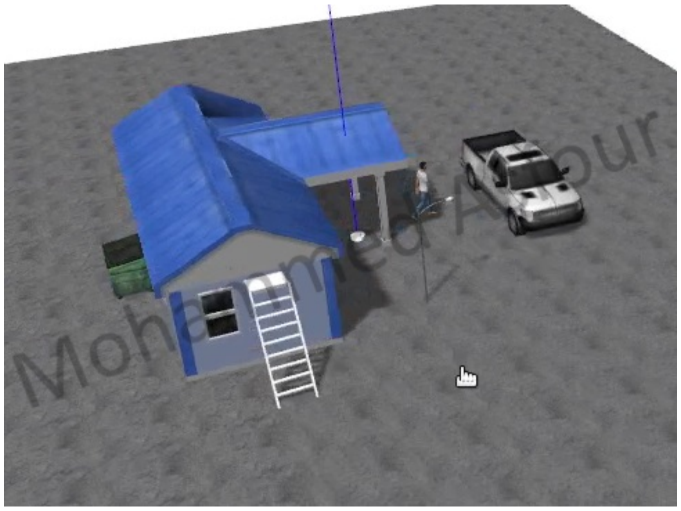


Fig. 7. Custom World Inside Gazebo

- Enabling the laser scan data in mapping

I was able to achieve good results. Both of these parameters are found in mapping.launch file I have also altered the node code in depth-to-laserscan package because it subscribed to a wrong topic by default, thus I have include this package in my submission.

### 3.7 Launch files

- **deploy-env-provided.launch** Launches the provided world in gazebo and rviz, as it includes rviz.launch, and loads the model.
- **mapping.launch** Launches the rtab-map package and the its visualization tool for mapping.
- **slam.launch** Launches the rtab-map package and the its visualization tool for SLAM.
- **teleop.launch** Provides keyboard keys control of the robot.

## 4 Results

#### *4.0.1 Provided World*

I have obtained very good results in this world, and I have included a video of the SLAM process of this world in the submission folder. As 3 loop closures are required in this world, the robot achieved as high as 29.

#### *4.0.2 Custom World*

I got good results in this world after altering the parameters I mentioned above. The 3D map is cut at a certain level simply because of the robot camera low position .

Other result images and a video is included in the submission folder, as there are many images to include.

## 5 Discussion and Future Work

As said before, some parameters needed to be altered in the mapping launch file and enabling the laser scan results as well. The robot detected a lot of wrong loop closures before altering these parameters. This project is almost ready to be implemented on a real robot, but will need some more information about the camera used the robot parameters to have correct simulation results.

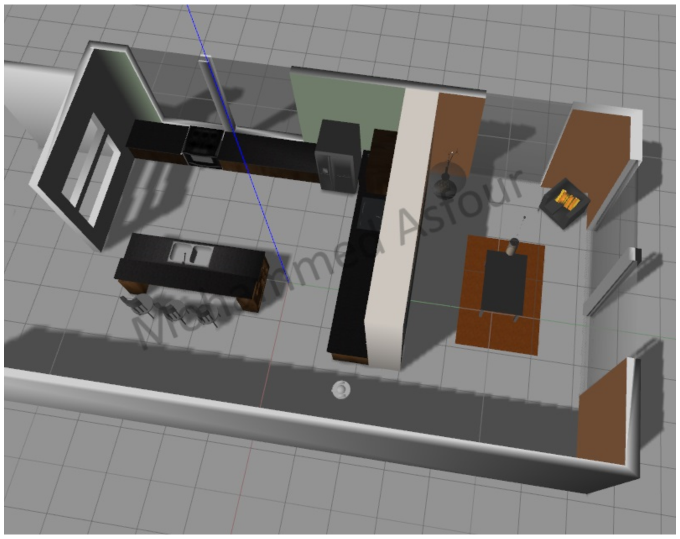
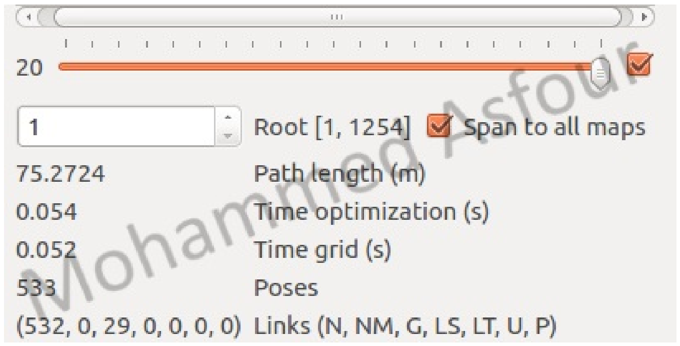


Fig. 8. Provided World Inside Gazebo



```
20
1                                  Root [1, 1254]  ✓ Span to all maps
75.2724                            Path length (m)
0.054                              Time optimization (s)
0.052                              Time grid (s)
533                                Poses
(532, 0, 29, 0, 0, 0, 0)           Links (N, NM, G, LS, LT, U, P)
```

Fig. 11. No. of Detected Loop Closures

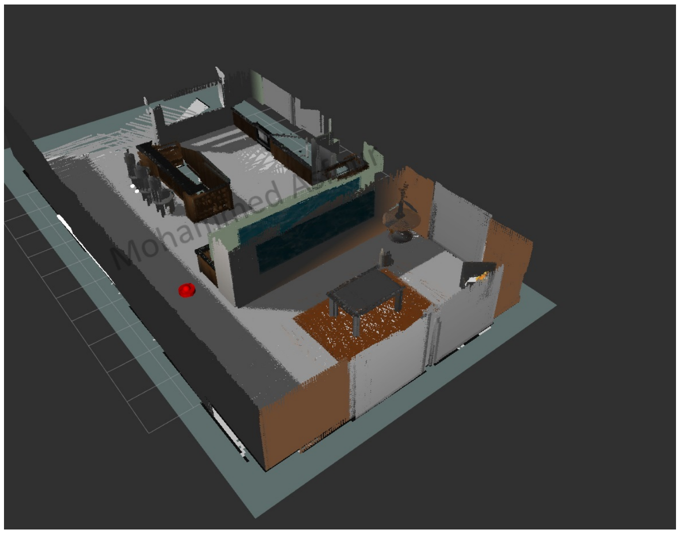


Fig. 9. 2D and 3D Maps for Provided Environment in Rviz

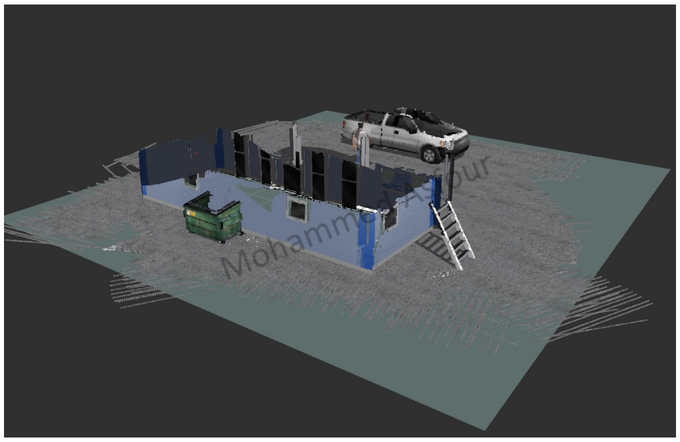


Fig. 12. 2D and 3D Maps for Custom Environment in Rviz

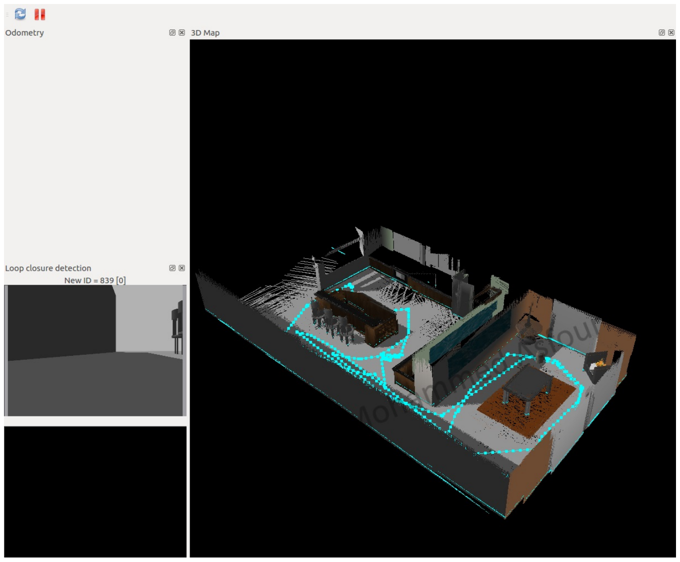


Fig. 10. Mapped Provided Environment in Rtab-map Visualization

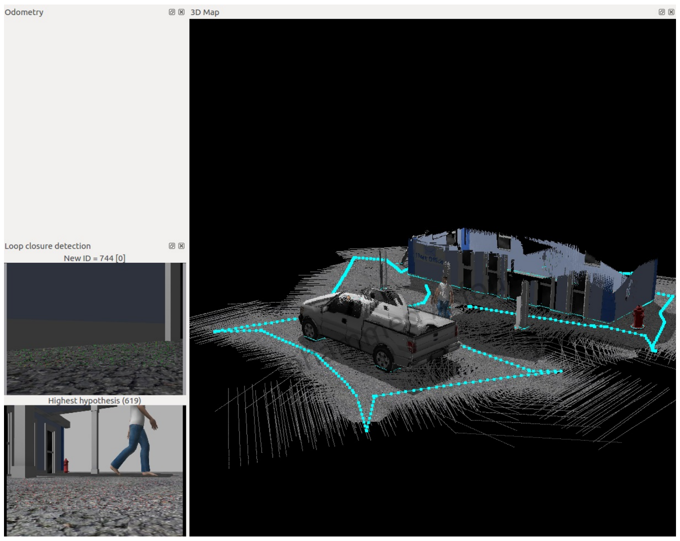


Fig. 13. Mapped Custom Environment in Rtab-map Visualization