Map my world Project Write-up

1. Abstract

In this report, we discuss robot mapping problem. An explanation of the importance of both 2D and 3D mapping is made. We show the environment we built in gazebo to test robot mapping module. Then we address our work to build “rtabmap”. Mapping work includes mapping of both given udacity environment and our built one. We use rtabmap database viewer to show map features which are discussed later with comments on the results and suggestions for performance enhancement.

2. Introduction

In this project we build a 2D and 3D map for robot working environment. We use the differential drive mobile robot model that has been built in localization project. The robot model is built using URDF file and simulated in gazebo and Rviz. We use rtabmap_ros package to build the map of two environments, the given udacity environment and our built one. The main contribution in this project is to integrate between different components and to build an API for using RTAB mapping package in ROS. A teleop ROS node is used to command the robot in gazebo environment and sensor measurements which are laser scans and RGB-D images are sent to rtabmap_ros node. Our results are analyzed to check mapping accuracy.

3. Background

Robot mapping is one of the most important problems to be studied in any robotic system. On the contrary of robot localization where the map is given and it is desired to estimate robot pose, we know robot state over time and want to create a map for the environment where the robot move. This task is important in exploration applications. In reality, localization and mapping tasks are not simply decoupled since we have uncertainties or unknowns in both the environment around the robot and robot state itself. In a real world application we only know robot measurements and actions with some level of uncertainty. How to use these data to map our world and estimate our pose relative to it is addressed by SLAM problem which is highly challenging by nature. Some of SLAM challenges comes from the fact that SLAM algorithm should identify correspondences between objects to extract map features in a correct manner. Also, data association should be accurately performed to determine which samples to be added to our consideration. There are two categories of SLAM problem: Online SLAM and Full SLAM. Each category have its distinguishing properties and algorithms. In Online SLAM, the current pose of the robot and the map are estimated based on the last control action (leading to current state) and last measurement (at current state). Each time step the robot take an action and make a measurement will estimate its pose and update its belief about the environment. The Online SLAM can be formally described as: $P(x_t, m|u_{1:t}, z_{1:t})$ where x: represents state, m: represents the map, u: represents the control action, and z: represents the measurement. On the other hand, Full SLAM (also known as offline slam) aims to estimate the entire path taken by the robot up to current time t. Each time step the robot take a control action and make a measurement it estimates the current pose, adjust the estimate of the previous poses, and update its belief about the map. The Full SLAM can be formally described as: $P(x_{1:t}, m | u_{1:t}, z_{1:t})$. There are many different algorithms that solve Full SLAM problem such as FastSLAM and GraphSLAM.

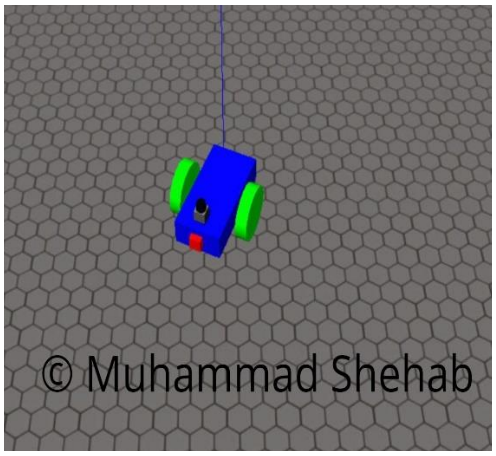
FastSLAM is a landmark based SLAM based on particle filter algorithm. It extends Monte Carlo Localization to estimate both features from the map and trajectory from robot poses. On the other hand, GraphSLAM represents the SLAM problem as a graph with robot poses and map feature locations as nodes, and distances corresponding to control actions or measurements between robot and landmarks as edges. The idea is to optimize edge links and consequently adjust node poses to get the best configuration of the graph which represent the estimated robot trajectory and map feature locations.

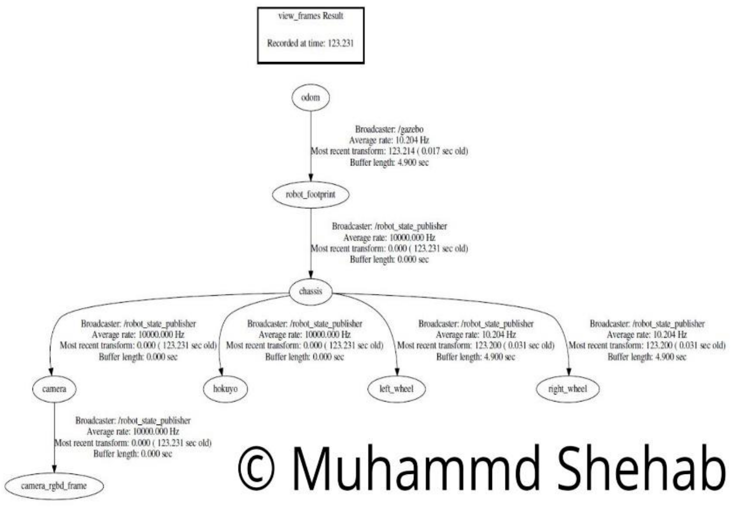
In this project we use the Real Time Appearance Based Mapping (RTAB-Map), which is a graph based SLAM approach. It uses data collected from vision sensor. The idea of RTAB is to use the concept of loop closures to determine whether the robot has seen this location before or not. Loop closures determination requires comparison between large sets of images which is computationally expensive. RTAB-Map is optimized to allow loop closures to be performed offline after trajectory execution.

4. Scene and Robot Configuration

4.1 Robot Model

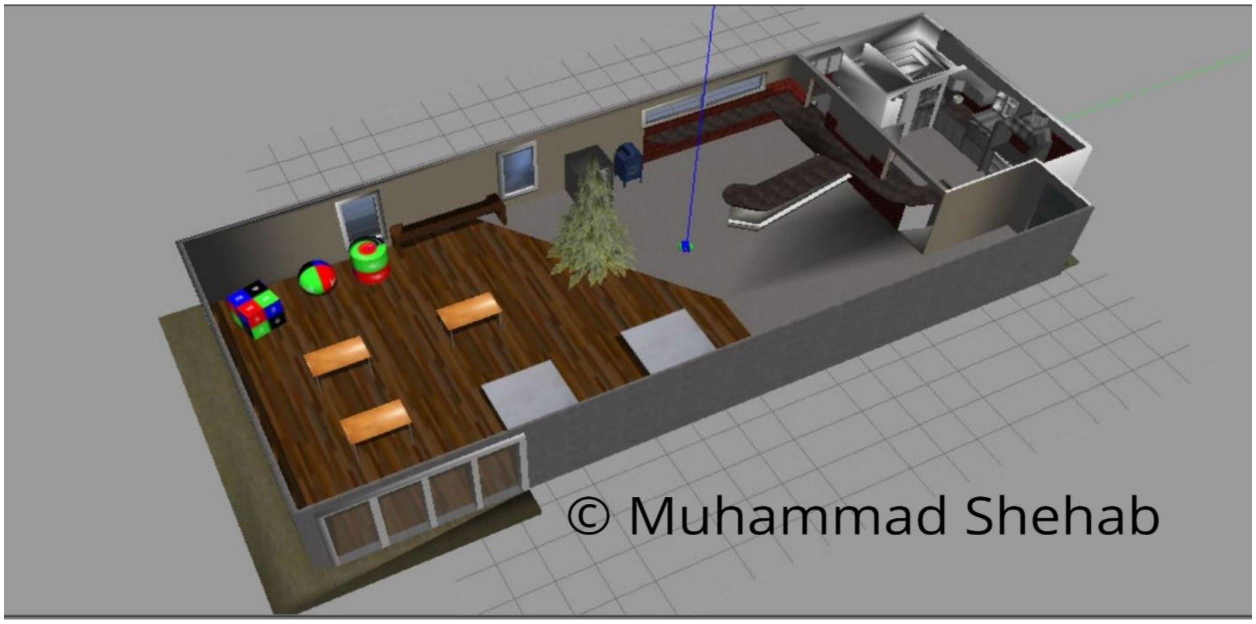
In this project we use the udacity_bot built in localization project with the modification of the camera sensor to include depth. This is done by adding camera_rgbd_frame link to .xacro file and modify camera plugin in .gazebo file to the one of Kinect camera. The robot and its TF is shown below:




view_frames Result
Recorded at time: 123.231
odom
Broadcaster: /gazebo
Average rate: 10.204 Hz
Most recent transform: 123.214 (0.017 sec old)
Buffer length: 4.900 sec
robot_footprint
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (123.231 sec old)
Buffer length: 0.000 sec
chassis
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (123.231 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (123.231 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 123.200 (0.031 sec old)
Buffer length: 4.900 sec
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 123.200 (0.031 sec old)
Buffer length: 4.900 sec
camera
hokuyo
left_wheel
right_wheel
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (123.231 sec old)
Buffer length: 0.000 sec
camera_rgbd_frame
© Muhammd Shehab

4.2 Scene Configuration

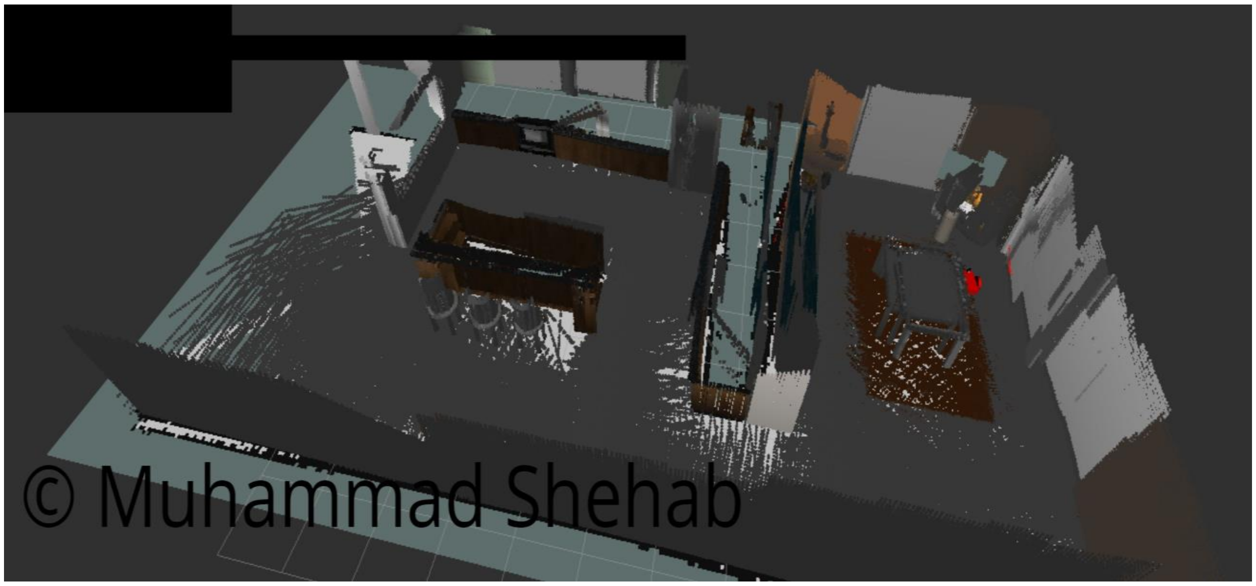
For world building, we used café world in gazebo and added some objects to it so as to represent features for the robot, we added the Tables required for the café and some decorations like textures, bookshelves, and a tree. Here is our final world:



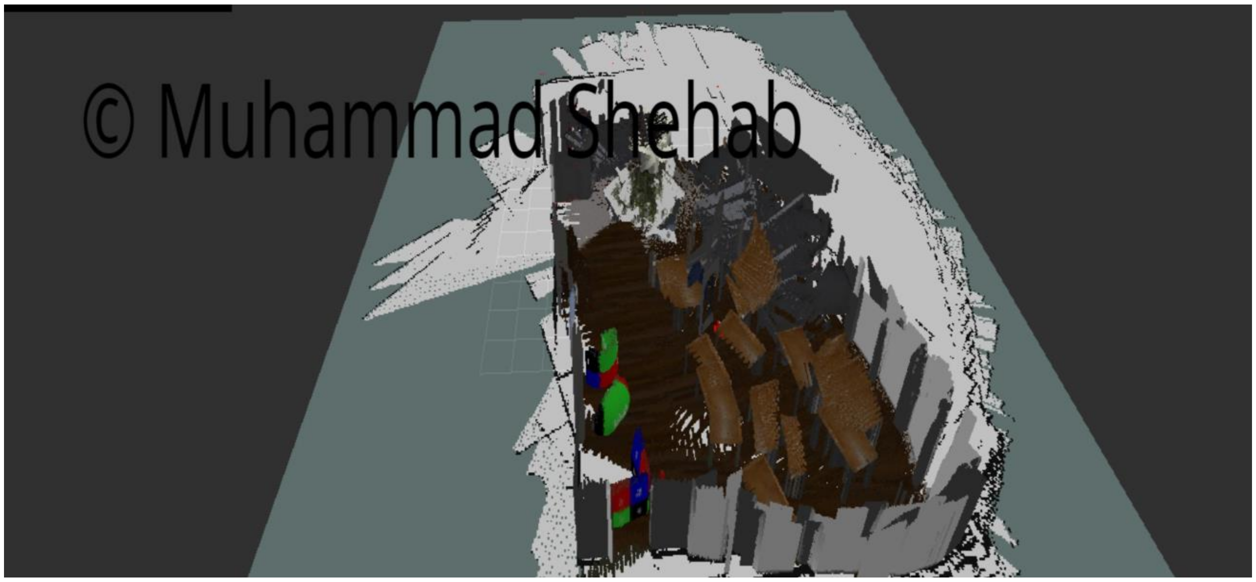
5. Results

5.1 Environment Maps (2D/3D)

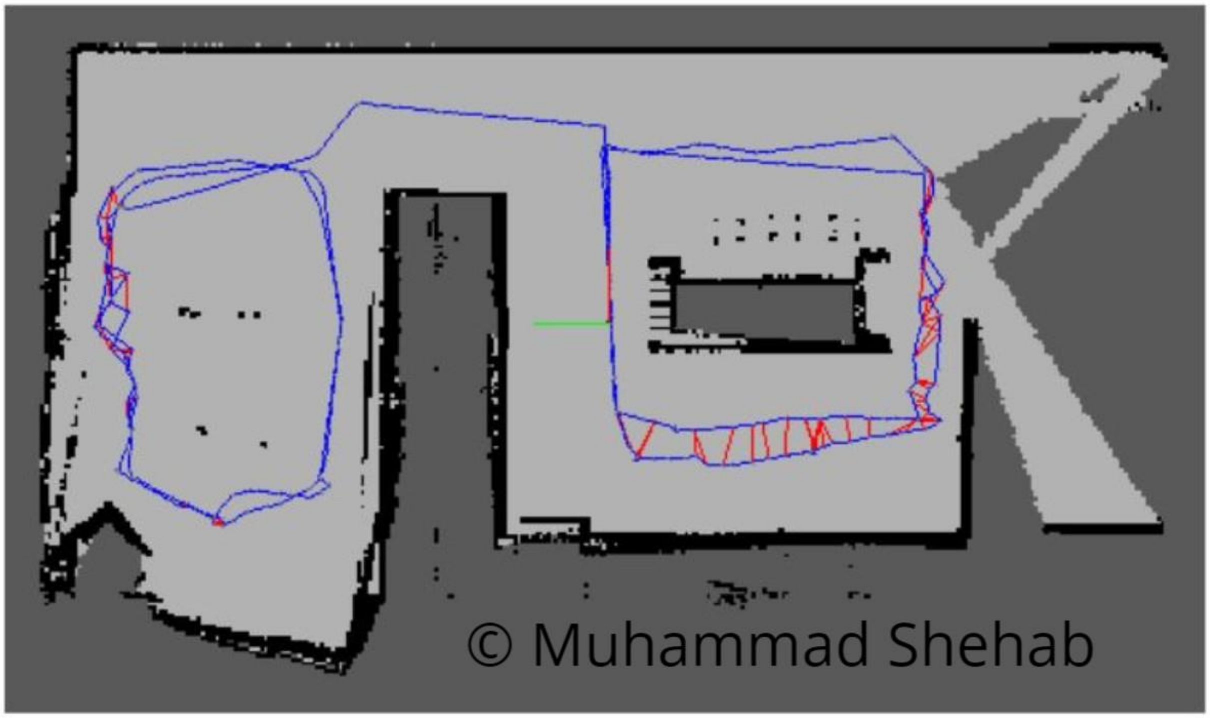
3D map for both environment saved from Rviz are as shown:



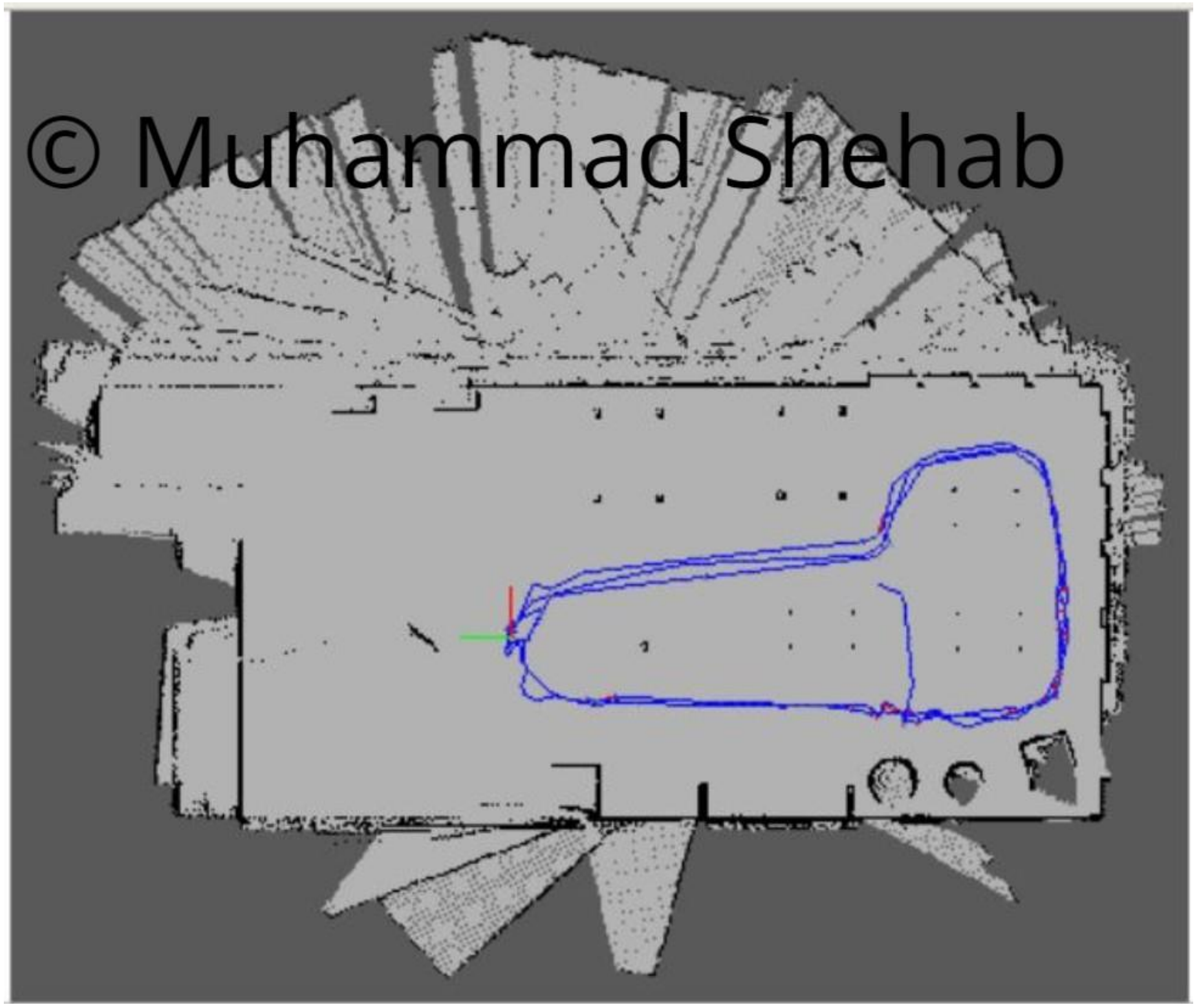
The 3D map is not so good and requires more iterations in my world since the world have similarities and the shadow of tables distract the camera. Laser measurements constructing 2D maps are more accurate.



2D Maps are shown on rtabmap database viewer. They are identifiable for both environments.

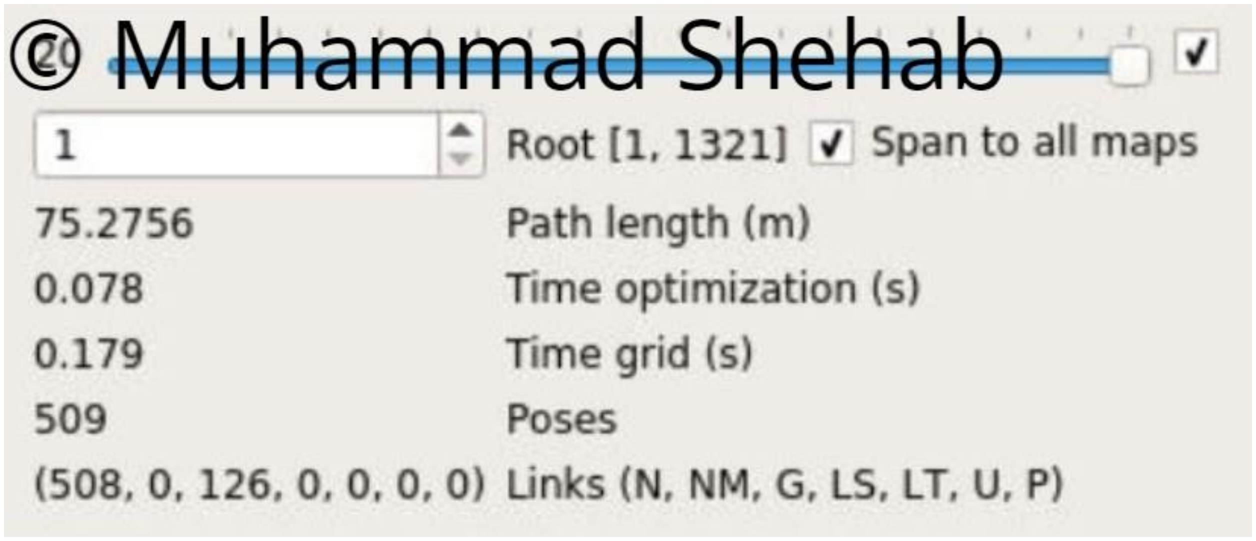


And here's the 2D map for my environment. The Legs of tables only detected by the Laser scanner but the map is more accurate than 2D. Some Noise outside the rectangle due to glass walls.

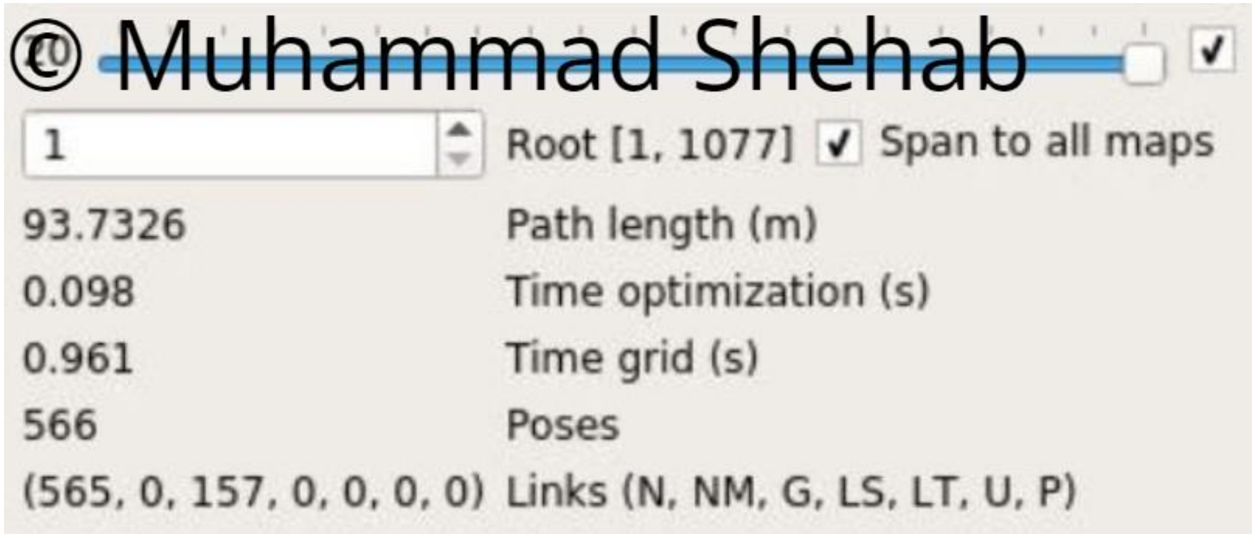


5.2 Loop Closures

Loop closure requirements are achieved for both environments as the path taken by the robot passed through the same locations (i.e. features) more than one time. Here's a screenshot from rtabmap database viewer showing the number of loop closures detected in udacity world and my world respectively.

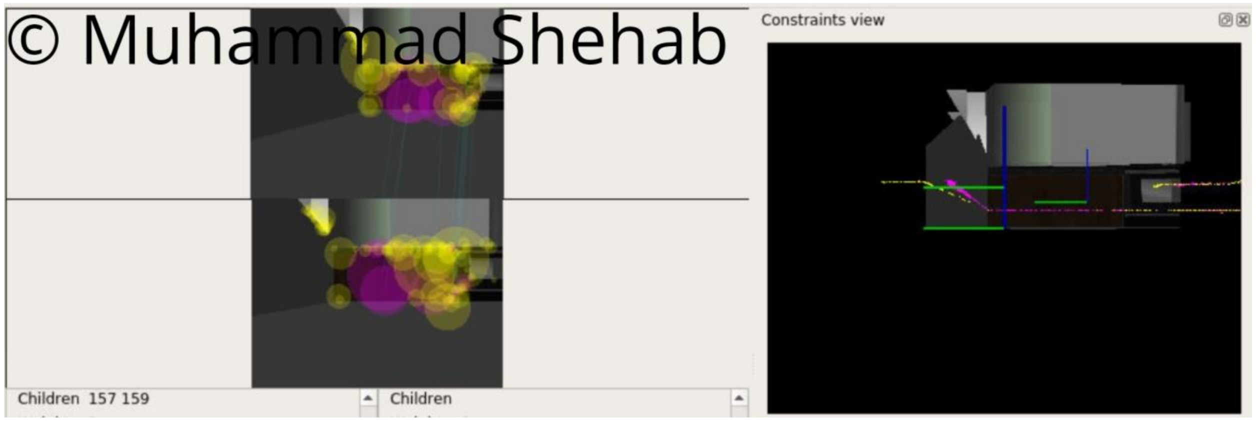


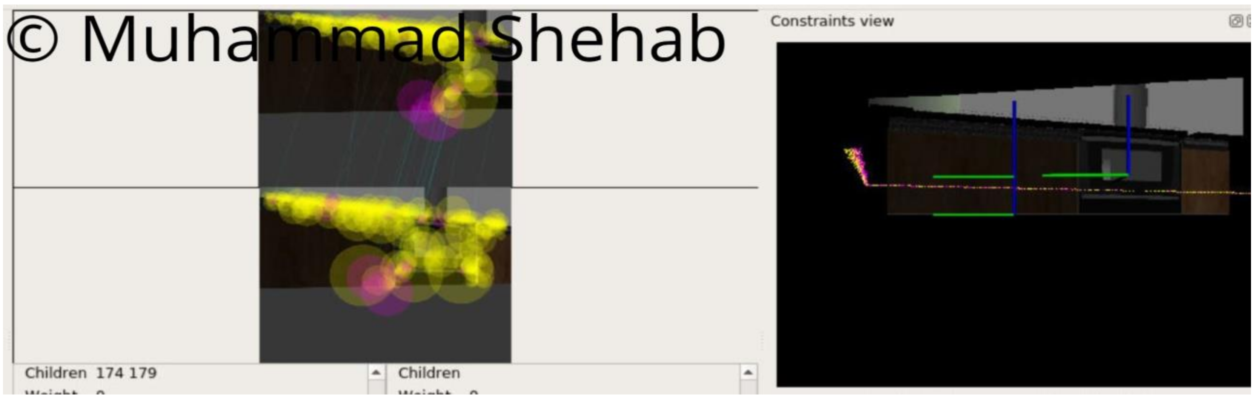
1	Root [1, 1321] ✓ Span to all maps
75.2756	Path length (m)
0.078	Time optimization (s)
0.179	Time grid (s)
509	Poses
(508, 0, 126, 0, 0, 0, 0)	Links (N, NM, G, LS, LT, U, P)

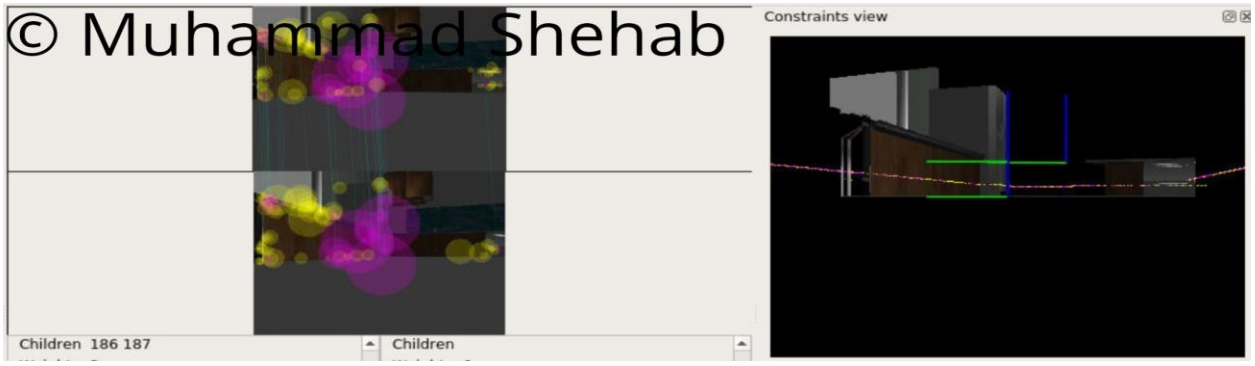


1	Root [1, 1077] ✓ Span to all maps
93.7326	Path length (m)
0.098	Time optimization (s)
0.961	Time grid (s)
566	Poses
(565, 0, 157, 0, 0, 0, 0)	Links (N, NM, G, LS, LT, U, P)

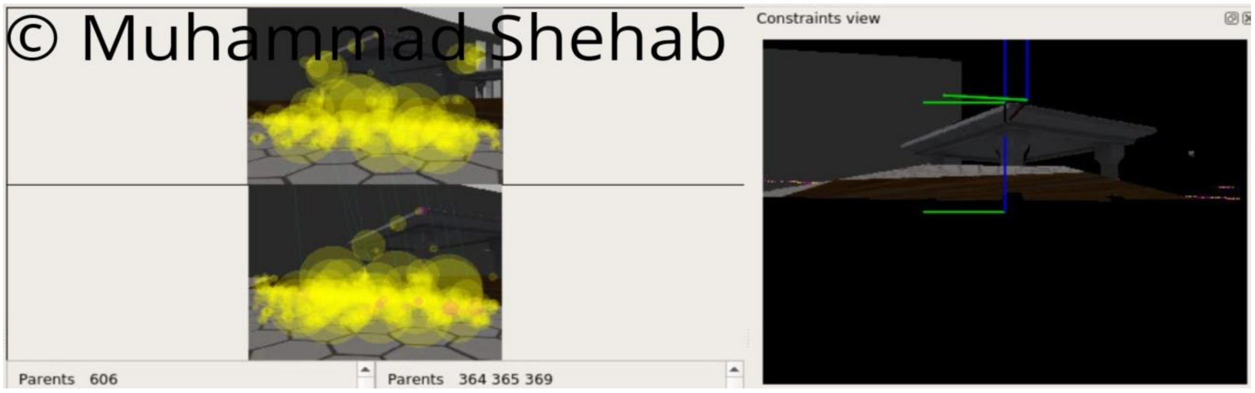
Examples of these loop closures for udacity world are listed below.

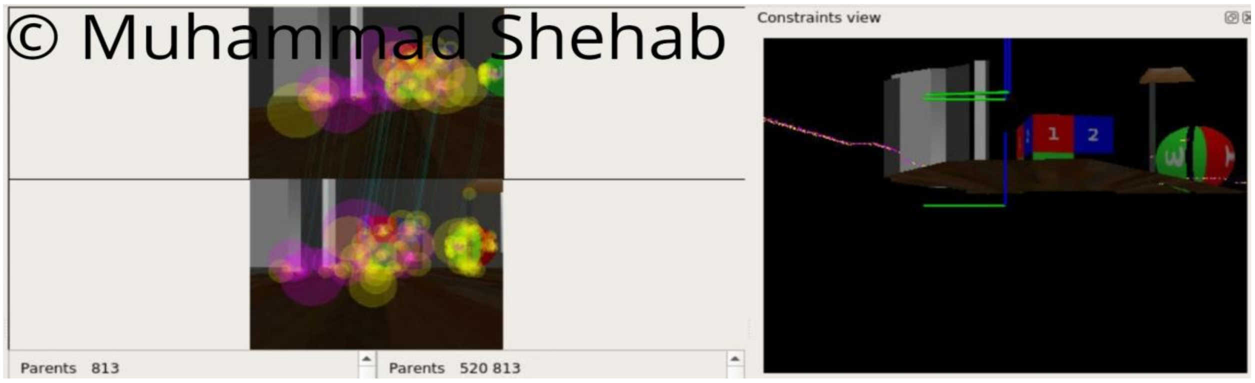


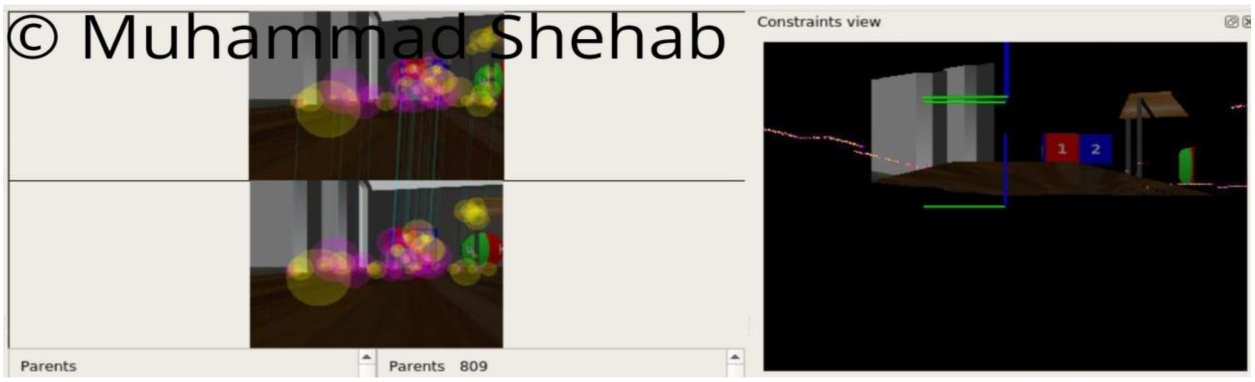




The same results are achieved for our built world.







5.3 RTAB-Map

My constructed RTAB-Map are so large to be uploaded on github. So I uploaded them on google drive and here's the link from which you can download:

https://drive.google.com/drive/folders/180oDppGCv79i1bwU0M1tCZN-HBlMOat_?usp=sharing

6. Discussion

In this project we achieved requirements for mapping an environment with at least 3 loop closures detected in the generated RTAB-Map. It is obvious that RTAB-Map requires huge computation. For accurate mapping we need to path through the same room more than one time to get more loop closures. This requires time and memory for rtab-map database. RTAB-Map SLAM depends on detecting loop closures based on feature correspondence. Hence, environments with rich features (i.e. distinguished landmarks at different locations) are easier to be mapped using image matching techniques. Also, Environments with solid walls can be properly mapped by laser scanner. In our project, mapping of supplied environment is more accurate since its features are properly distributed and are not repeated. This is slightly different in case of café environment which contains a large hall with some tables on it. We also notice that in both environments, glass walls affect the quality of mapping in these regions.

7. Future Work

All work done in this project only represents a first step in slam problem issues to be solved. For real world application this work should be extended to include:

1. Sensor Calibration to estimate measurement noise ranges.
2. Real application of control action to evaluate model uncertainty.
3. Use of more powerful Hardware to perform required computations effectively.
4. Make a mode of the real world where the robot works and then compare between maps created based on motion in simulation and those created based on real world motion.