Map my world

1. Abstract

In this project we worked on mapping problem in 3D and 2D, we used gazebo to show the environment which used to test robot mapping for 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

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. Two common types of SLAM are Fast SLAM and Graph SLAM

2.1 Fast SLAM

The fast SLAM algorithm uses a custom particle filter approach to solve the full SLAM problem with known correspondences. Using particles, fast SLAM 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. Fast SLAM uses a low dimensional extended Kalman filter to solve these independent features to estimate the map. The disadvantage of Fast SLAM 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 Fast SLAM which extends Fast SLAM to occupancy grid maps. The algorithm implements parts of the MCL algorithm along with the occupancy grid mapping algorithm.

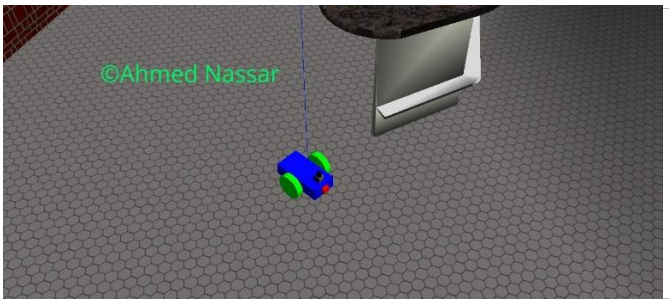
2.2 Graph SLAM

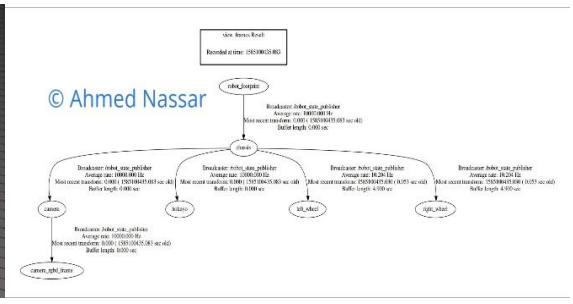
Another type of SLAM is Graph SLAM which solves the full SLAM problem. This means that the algorithm recovers the entire path and map instead of the most recent pose and map. This difference allows it to consider dependencies between the current and previous poses. Graph SLAM has the benefit of reducing the need for significant onboard processing capability and has improved accuracy over Fast SLAM. Graph SLAM 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.

4. World Creation

4.1 Robot Model

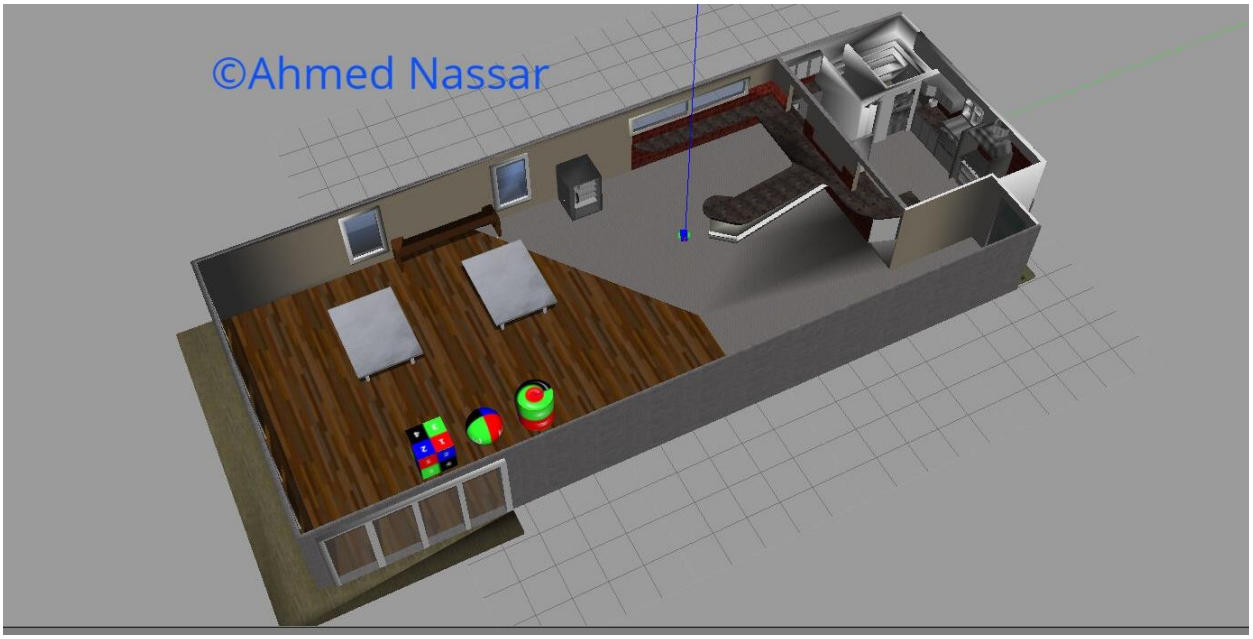
In this project we use the Udacity_bot built in localization project with the modification of the camera sensor to include depth. This is done by adding camera_rgbd_frame link to .xacro file and modify camera plugin in .gazebo file to the one of Kinect camera. The robot and its TF is shown below:





4.2 Scene Configuration

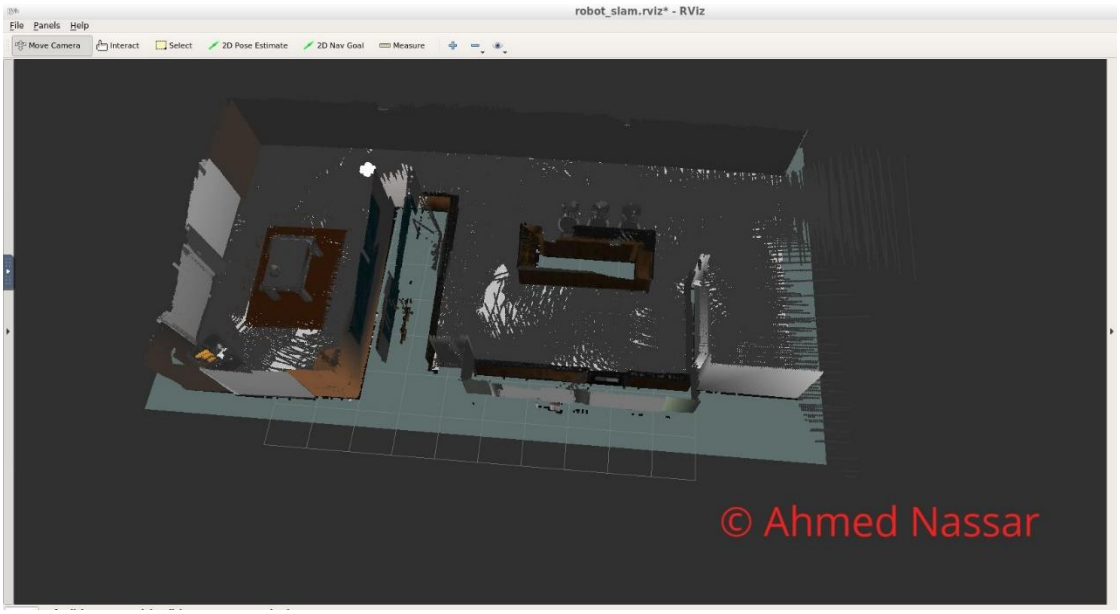
For world building, we used café world in gazebo and added some objects to it so as to represent features for the robot, we added the Tables required for the café and some. Here is our final world:



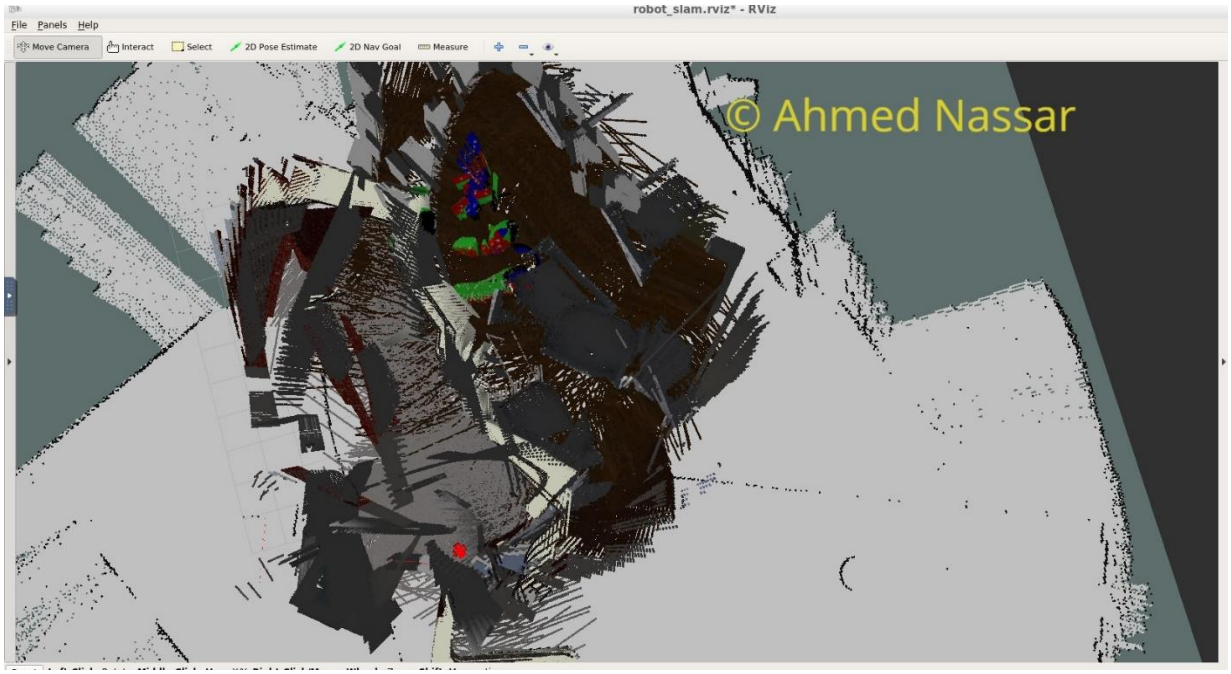
5. Results

5.1 Environment Maps

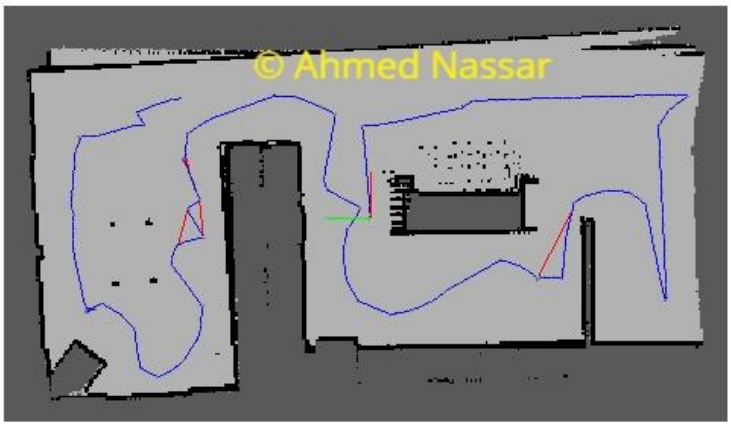
3D map for both environments are shown below:

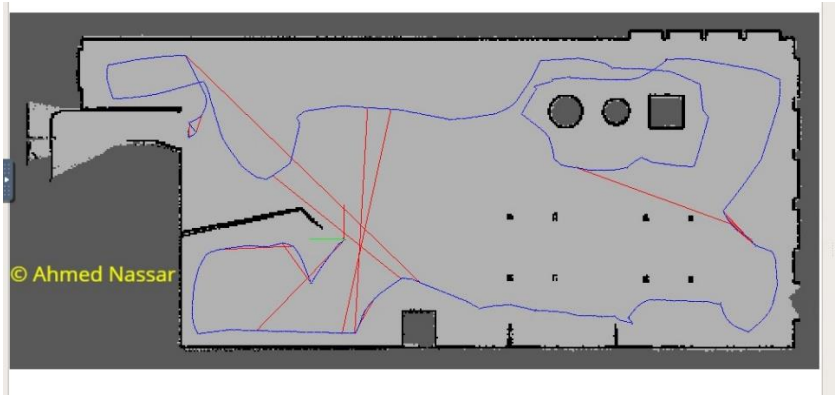


The 3D map is not so good and requires more iterations in my world since the world have similarities and the shadow of tables distract the camera. Laser measurements constructing 2D maps are more accurate.



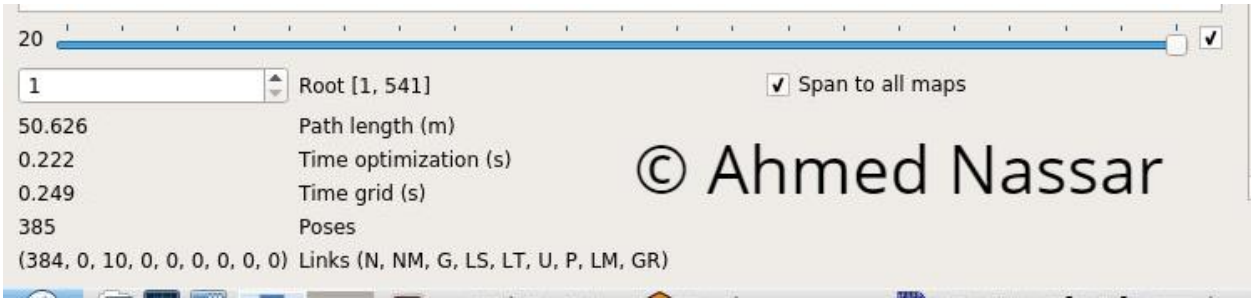
2D Maps are shown on rtabmap database viewer.



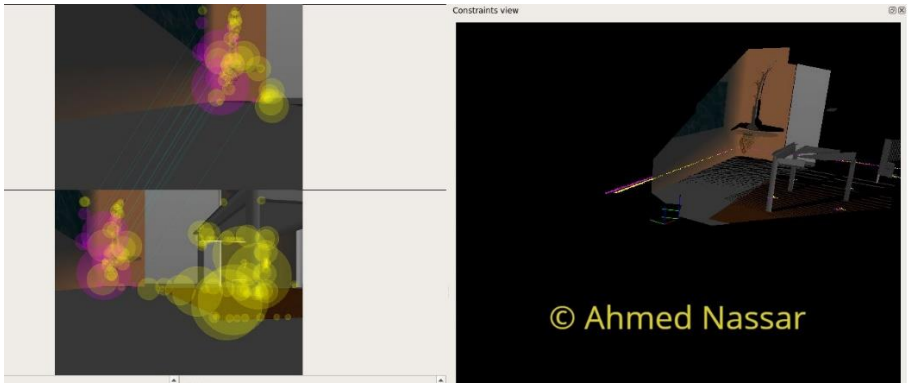


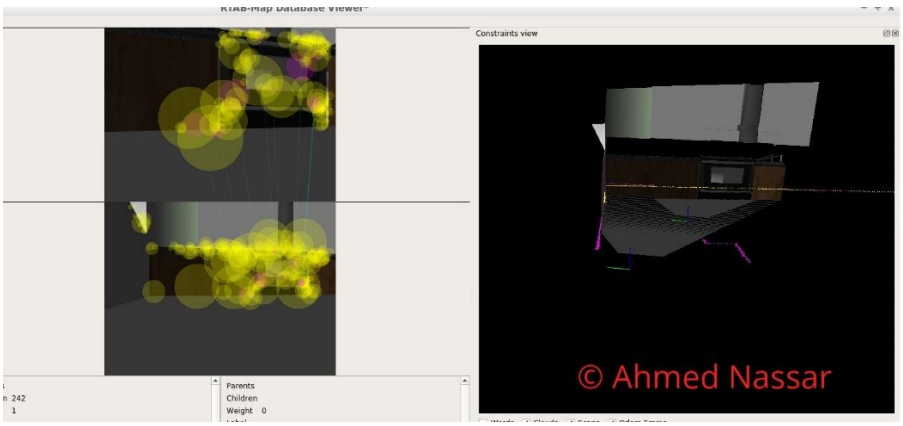
5.2 Loop Closures

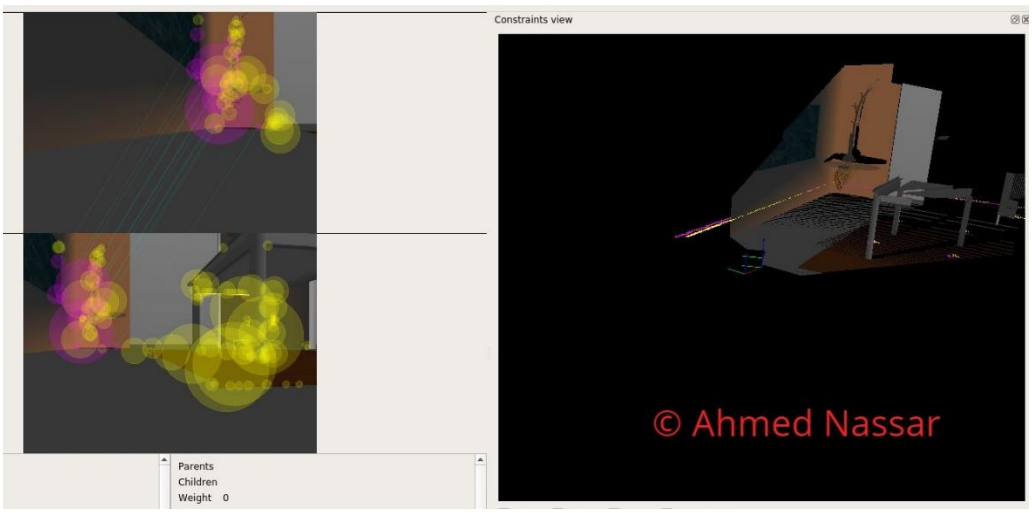
Loop closure requirements are achieved for both environments as the path taken by the robot passed through the same locations (i.e. features) more than one time. Here's a screenshot from rtabmap database viewer showing the number of loop closures detected in Udacity world and my world respectively.



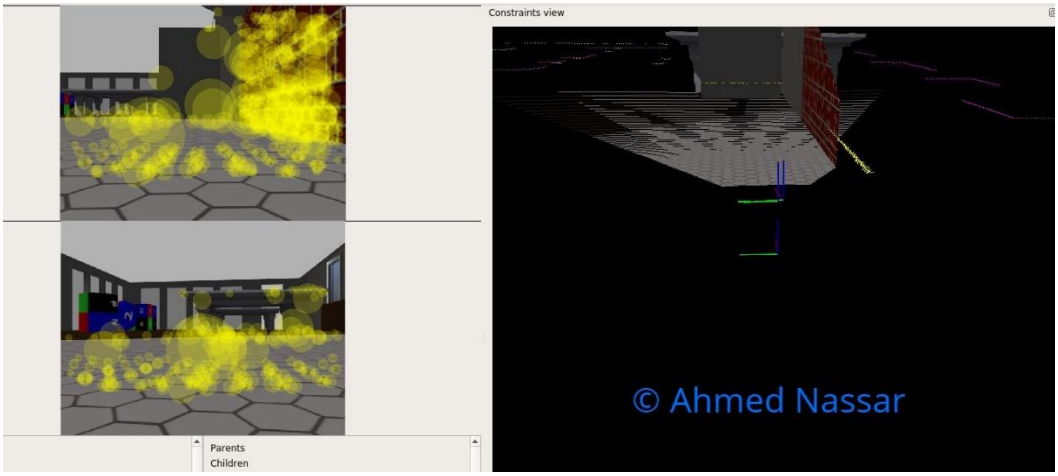
Examples of these loop closures for Udacity world are listed below.







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/open?id=1sP_PTsRJwsY-P_j76hmPddYXmyS2c70R

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 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 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.