Map My World

Mustafa Eldeeb

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. The overview of the tasks can are as follows, creating map for the given environment, create a new environment and build the map of the new environment using RTAB-Map ROS package.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, SLAM, RTAB-Map, ROS, Robot Model.

1 Introduction

The problem is to create the map of an environment while the robot is navigating.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.

2 Background

The problem is to create a map of the environment, as mentioned earlier. Furthermore, localization of the robot is a requirement to complete the task. Since the sensors are not precise, there is noise to affect the measurements coming from the sensors, and the controls are not perfect to move the vehicle to the desired location. Therefore, the algorithm should handle the noise.

2.1 Occupancy Grid Mapping

Occupancy Grid Mapping is an algorithm that assumes the location of the robot is available when constructing the map [?]. During SLAM operation, then the algorithm builds the map of the environment and localize the robot relative to it. After, the algorithm uses the exact robot poses filtered from SLAM. Then, using the known poses from SLAM and noisy measurements from the sensors generates a fit for path planning and navigation.

2.2 Grid-based FastSLAM

FastSLAM estimates a posterior over trajectory using particle filters. It solves the Full SLAM problem. Full SLAM looks for the entire path up to time t, using all of the controls and measurements. It has the advantage of mapping with known poses and utilizes low dimensional EKF to solve independent features of the map.
Grid-based FastSLAM uses Monte Carlo Localization (MCL) instead of EKF. It first estimates the trajectory, then the map by assuming known poses and applying the occupancy grid mapping algorithm.
In the Grid-based FastSLAM technique, there are three steps. First one is sampling motion. In this step, the current location of a given particle is calculated using the previous one and the current controls. In the second step, which is map estimation, with the inputs of current measurements, the particles current location and the previous map, the next map is estimated. The last step updates the weights of the particles.

2.3 GraphSLAM

GraphSLAM uses a graph representation to the SLAM problem. It constructs graphs and then matrices. They help to determine different observations showing the same landmark. [?]. The main feature of GraphSLAM is graph optimization with maximum likelihood estimation (MLE). The procedure is as follows

1) Remove inconsequential constants.
2) Convert the equation from likelihood estimation to negative lo likelihood estimation
3) Calculate the first derivative of the function and set it to zero to find extrema.

The properties of the information matrix and vector are as follows

- A motion constraint ties together two poses.
- A measurement constraint ties together one feature and one pose.
- Each operation updates four cells in the matrix and two cells in the vector.
- All other cells are zero, therefore the matrix is sparse. Sparsity helps solving equations on limited hardware.

3 Simulations

The simulation environment consists of ROS (Kinetic), Gazebo and RViz. The main ROS package is the RTAB-Map package. There are two different environments to test on the simulation. The 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.

3.1 The Kitchen Dining

The bot has created the map of the Kitchen Dining world. Figure 1 shows the world and udacity bot on Gazebo. After the mapping completed, the visualization of the database can be seen from the Figure 13.

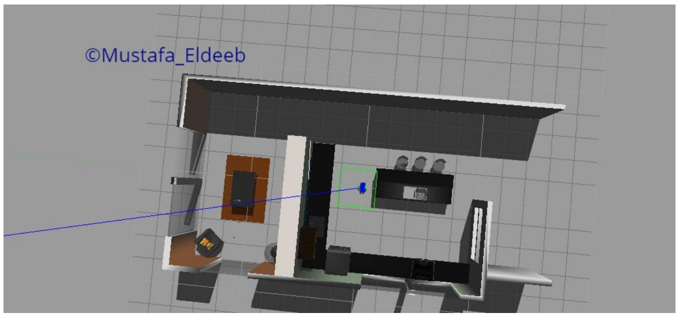

Fig. 1. Kitchen Dining in Gazebo

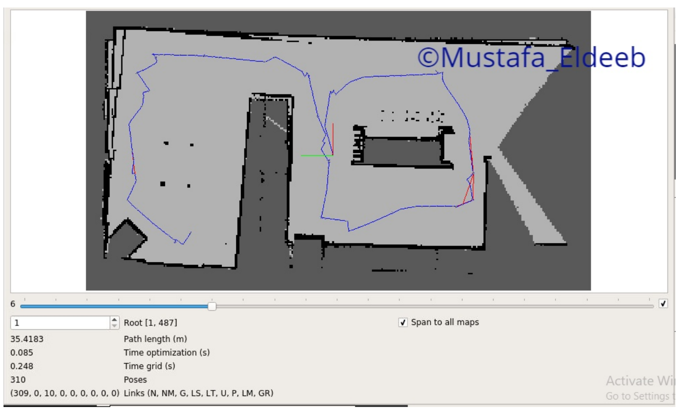

Fig. 2. 2d map in RTABMap DB Viewer

Fig. 3. number of loop closures detected

Examples of these loop closures for Udacity world are listed in the following.

3.2 The deeb World

The deeb World was built as the custom environment. To increase mapping accuracy, different types of objects were distributed to the map. Figure 7 shows the custom world in Gazebo.

Examples of these loop closures for deeb world are listed in the following.

Fig. 4. udacity world loop closure example 1

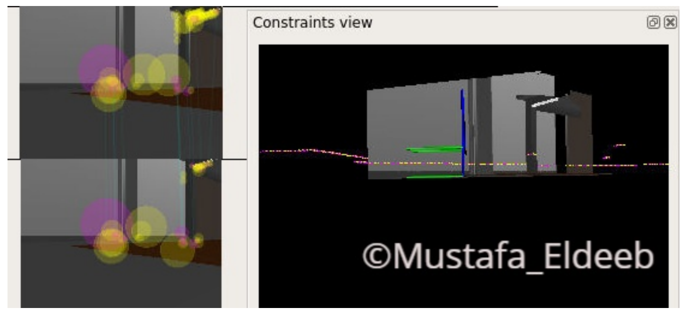

Fig. 5. udacity world loop closure example 2

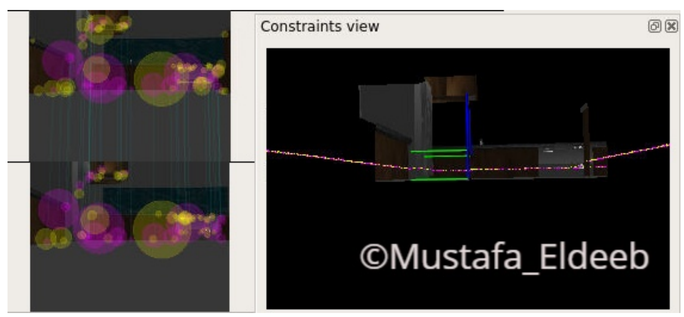

Fig. 6. udacity world loop closure example 3

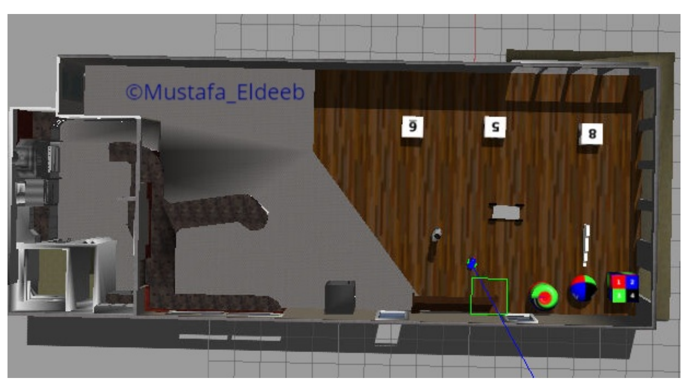

Fig. 7. deeb World in Gazebo

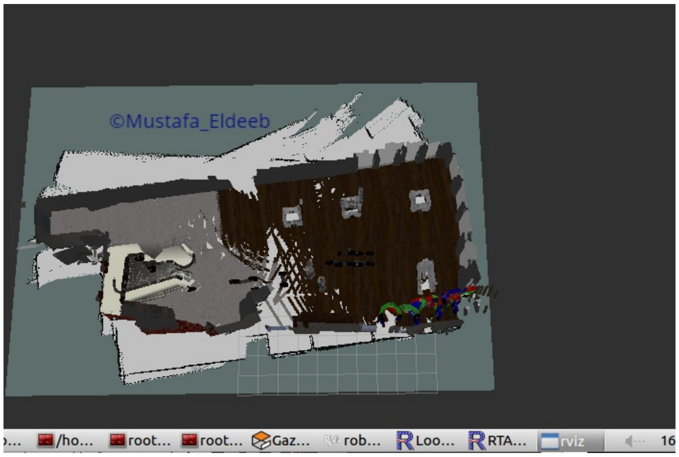

Fig. 8. deeb World in RViz

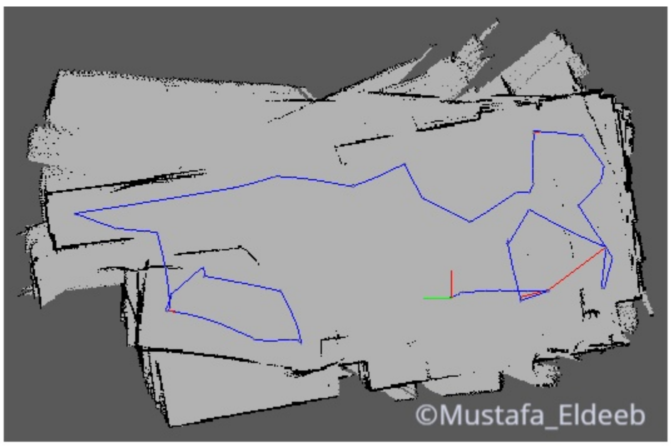

Fig. 9. deeb world 2d map in RTABMap DB Viewer

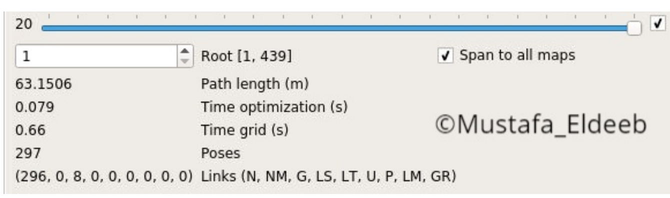

Fig. 10. number of loop closures detected in deeb world

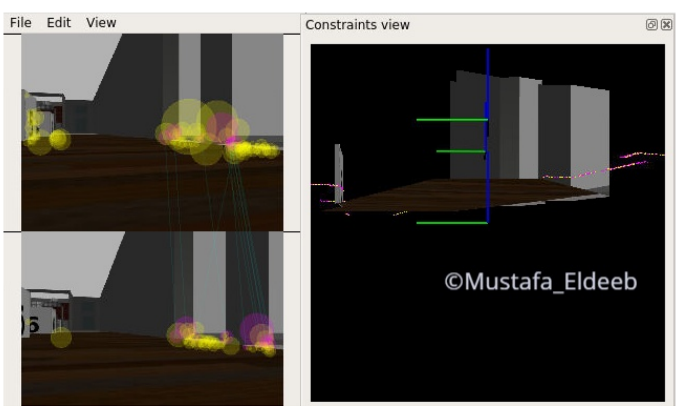

Fig. 11. deeb world loop closure example 1

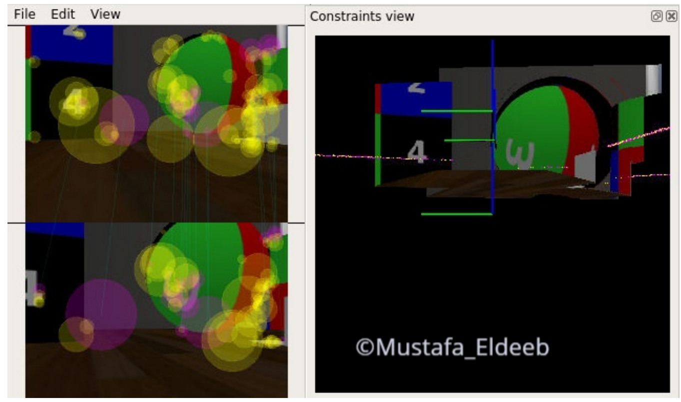

Fig. 12. deeb world loop closure example 2

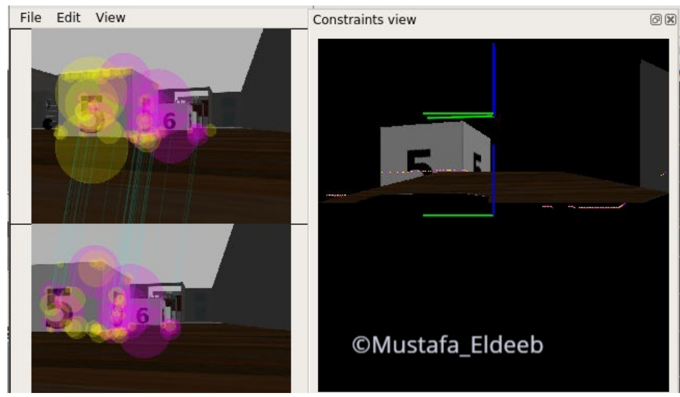

Fig. 13. deeb world loop closure example 3

3.3 Achievements

The algorithm was able to create maps for both of the environments, however, the maps were not perfect. There were little rotations and noises on the maps. Figure 14 shows the frame, and Figure 15 shows the topics and the ROS graph.

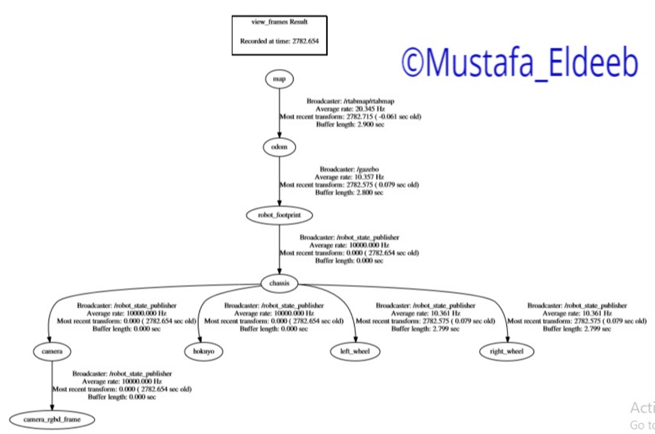

Fig. 14. TF Frames

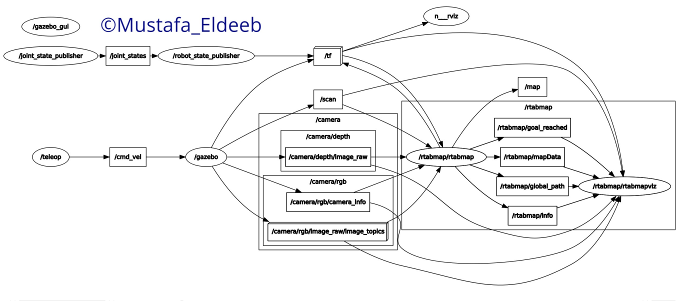

Fig. 15. ROS Graph and Topics

4 Results

4.1 Mapping Results

4.1.1 The Kitchen Dining

3D map for both environments are:

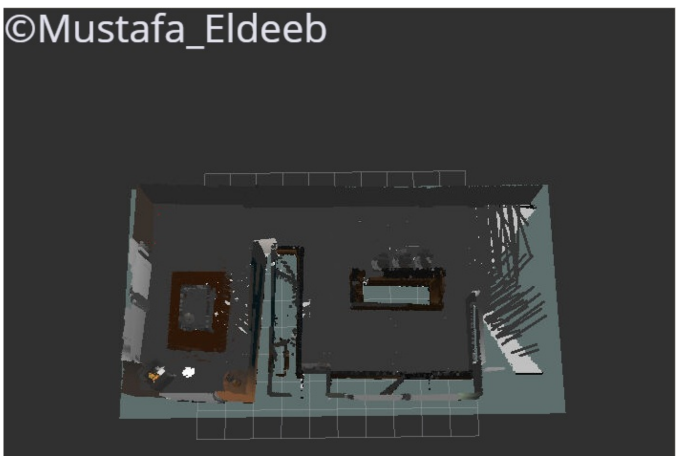

Fig. 16. Kitchen Dining Map in Rviz

4.1.2 deeb bot

The map is skewed from the horizontal middle line.

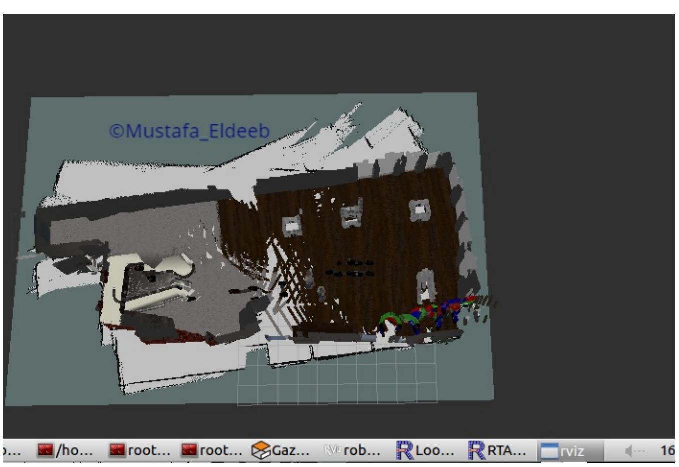

Fig. 17. deeb World Map

4.2 Technical Comparison

The first world is a more realistic world. It has objects to serve as landmarks which lead to a better mapping for the algorithm. There are several objects in the custom map, but still, it is harder to create the map of the deeb world.

5 Discussion

The algorithm was able to create a map for each environment. The robot navigated for almost two laps in both of the worlds. Both of the map images look warped. The parameters for the algorithm should be optimized, but due to time limitations, the tunning operation would be future work.

6 Conclusion / Future work

It is required to tune the parameters for mapping. Besides, a more complex environment should be created for further tests.

Testing the algorithm on the Jetson TX2 and comparing the results with the current results is a waiting task for the future work. Upon receiving successful results, the algorithms in the project can be used on a real robot such a vacuum cleaner or an autonomous forklift.