Map My World Robot

Mohamed Sayed Antar

Abstract—As RTAB-Map one of the best solutions for SLAM to develop robots that can map environments in 3D, using this package to create a 2D occupancy grid and 3D octomap. By developing a new package to interface with the rtabmap-ros package and deploy a robot in a simulated environment using gazebo and RViz then using teleop to move around the environment to generate a proper map.

Index Terms—SLAM, Mapping, Localization, rtab-map, teleop, octomap, occupancy-grid-map, EKF, MCL.

1 Introduction

ROBOT localization is the process of determining where a mobile robot is located with respect to its environment. Localization is one of the most fundamental competencies required by an autonomous robot as the knowledge of the robot's own location is an essential precursor to making decisions about future actions.

localization and mapping(SLAM) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it.

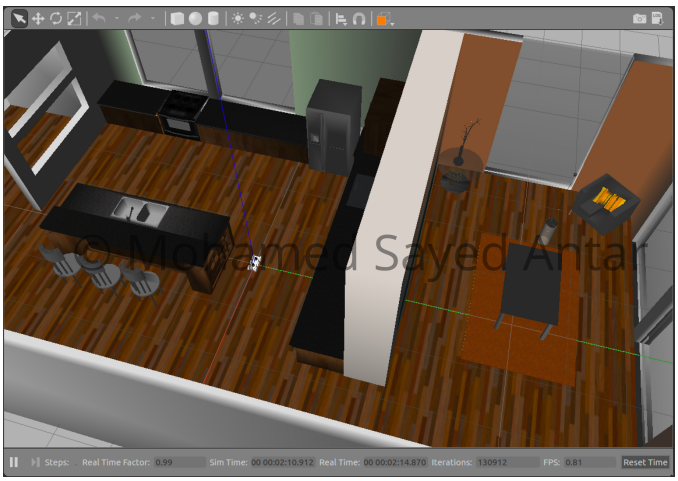

Fig. 1. Gazebo world

there are several algorithms known for solving it, at least approximately, in tractable time for certain environments. Popular approximate solution methods include the particle filter, extended Kalman filter, Co-variance intersection, and Graph-SLAM.

RTAB-Map (Real-Time Appearance-Based Mapping) is a RGB-D Graph SLAM approach based on a global Bayesian loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location. When a loop closure hypothesis is accepted, a new constraint is added to the map's graph, then a graph optimizer minimizes the errors in the map.

2 Background

In a general sense, the purpose of SLAM algorithms is easy enough to iterate. A robot will use simultaneous localization and mapping to estimate its position and orientation (or pose) in space while creating a map of its environment. This allows the robot to identify where it is and how to move through some unknown space.

In the real world, a mobile robot with a two-dimensional laser rangefinder sensor is generally deployed on a flat surface to capture a slice of the 3D world. Those two-dimensional slices will be merged at each instant and partitioned into grid cells to estimate the posterior through the occupancy grid mapping algorithm. Three-dimensional maps can also be estimated through the occupancy grid algorithm, but at much higher computational memory because of the large number of noisy three-dimensional measurements that need to be filtered out.

For two dimensional maps, describing a slice of the 3D world. In resource constrained systems, it can be very computationally expensive to build and maintain these maps. 3D representations are even more costly. That being said, robots live in the 3D world, and we want to represent that world and the 3D structures within it as accurately and reliably as possible.

2.1 Collecting data

3D lidar can be used, which is a single sensor with an array of laser beams stacked horizontally. Alternatively, a 2D lidar can be tilted (horizontally moving up and down) or rotated (360 degrees) to obtain 3D coverage.

An RGB-D camera is a single visual camera combined with a laser rangefinder or infrared depth sensor, and allows for the determination of the depth of the image, and ultimately the distance from an object. A stereo camera is a pair of offset cameras, and can be used to directly infer the distance of close objects, in the same way as humans do with their two eyes.

A single camera system is cheaper and smaller, but the software algorithms needed for monocular SLAM are much more complex. Depth cannot be directly inferred from the sensor data of a single image from a single camera. Instead, it is calculated by analyzing data from a sequence of frames in a video.

2.2 FastSLAM

The FastSLAM algorithm uses a custom particle filter approach to solve the full SLAM problem with known correspondences. Using particles, FastSLAM estimates a posterior over the robot path along with the map. Each of these particles holds the robot trajectory which will give an advantage to SLAM to solve the problem of mapping with known poses. In addition, to the robot trajectory, each particle holds a map and a local Gaussian represents each feature of the map. With this algorithm, the problem divided into a separate independent problem. Each of which aims to solve the problem of estimating features of the map. To solve these independent mini problems, FastSlam uses the low dimensional extended Kalman filter. While math features are treated independently, dependency only exists between robot pose uncertainty. This custom approach of representing posterior with particle filter and Gaussian is known by the Rao-Blackwellized Particle Filter One. With the Monte Carlo Localization(MCL) FastSLAM estimates the robot trajectory and with Low-Dimensional Extended Kalman Filter (EKF), FastSLAM estimates features of the map.

2.3 GraphSLAM

GraphSlam is another SLAM algorithm that solves the full SLAM problem. This means that the algorithm recovers the entire path and map, instead of just the most recent pose and map. This difference allows it to consider dependencies between current and previous poses.

3 Scene and Robot Configuration

3.1 Scene

first using the provided environment in gazebo for simulation, rtabmap package to solve slam problem, rviz, and teleop package to move around the environment to map it.

then using the gazebo models database to create the new environment from scratch by adding the ground, cars and other elements to make the environment similar to a small street.

Fig. 2. Gazebo world 2

3.2 Robot

using the robot from localization project without any changes, this robot has a RGB-D camera and two laser sensors with six wheels.

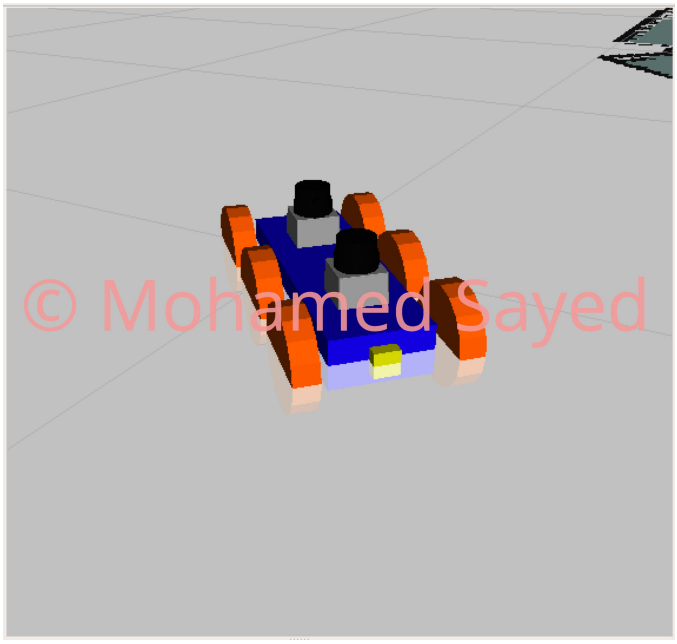

Fig. 3. Robot

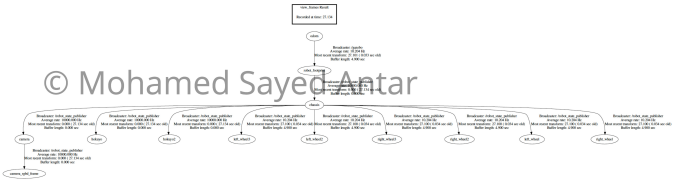

Fig. 4. Frames

3.3 Packages

eventually by creating a package named slam-project this package contain the following main folders:

- worlds folder
- launch folder
- urdf folder
- meshes folder
- config folder

the worlds folder contain the different (.world) files for the environments.

the launch folder contain the launch files.

the urdf folder which contain the robot xacoro file and the gazebo file.

the meshes contain the (.dae) files for any robot sensor or device.

the config folder contain any required configuration files.

4 Results

in this section the output maps in shown as 2D maps and 3D maps.

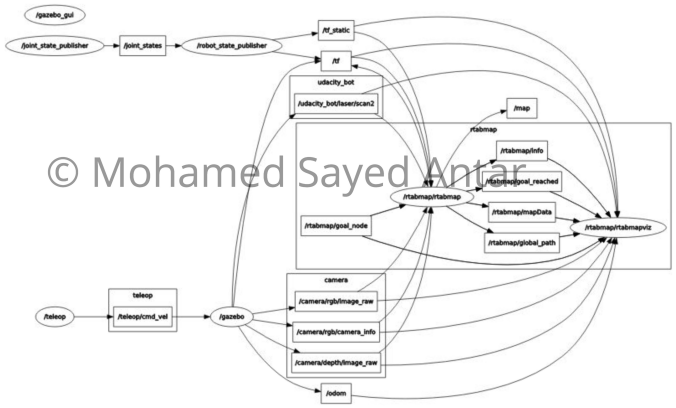

Fig. 5. rqt-graph

4.1 2D Maps

for the 2D maps for both the given environment and the new environment.

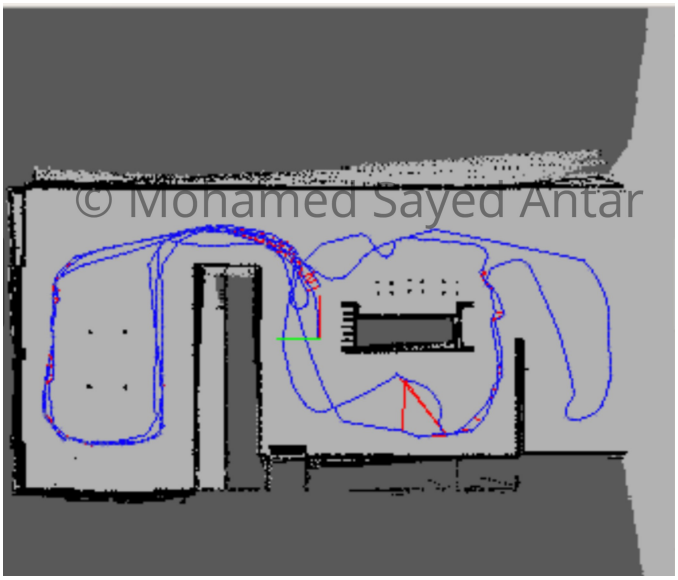

Fig. 6. 2D-fistr.Env

4.1.1 3D maps

for the 3D maps for both the given environment and the new environment.

4.1.2 RViz

for RViz output for both the given environment and the new environment.

4.1.3 Database Analysis

The rtabmap-databaseViewer is a great tool for exploring your database when you are done generating it. It is isolated from ROS and allows for complete analysis of your mapping session. This is how you will check for loop closures, generate 3D maps for viewing, extract images, check feature mapping rich zones, and more.

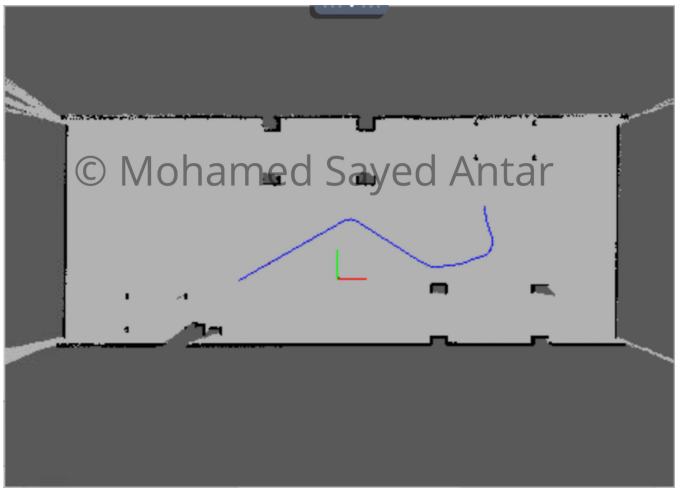

Fig. 7. 2D-second.Env

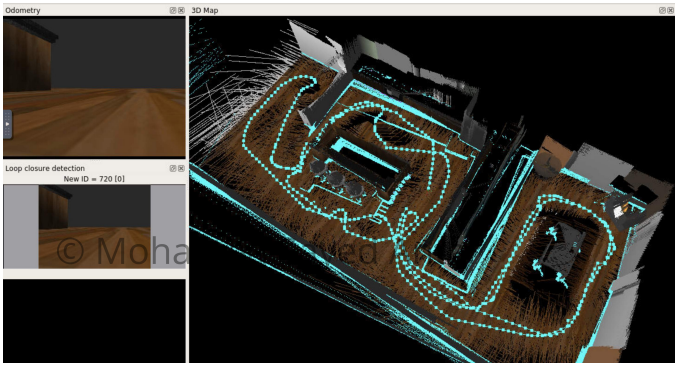

Fig. 8. 3D-first.Env

5 Discussion

As the robots camera in a low height from the ground this lead the robot for weak discovering the far areas around him it need to be near to any thing on the floor to identify it's color and so on and this is not good, so the camera should be a little higher from the ground for better discovering the around areas, so the ground of the 3D map is not recognized well even the robot was near to this ground part. another bad thing is the accuracy of the robot sensors, even these errors is small but after moving around the room or the

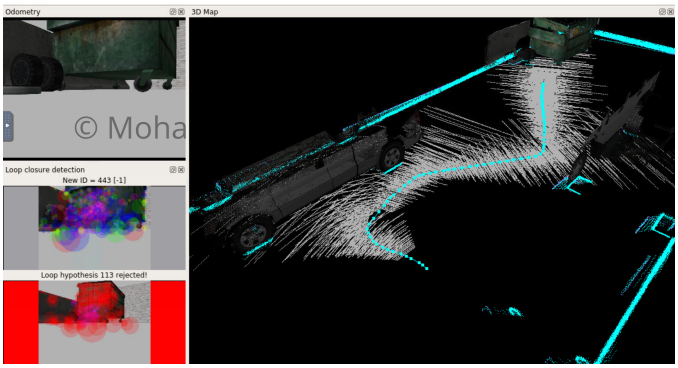

Fig. 9. 3D-second.Env

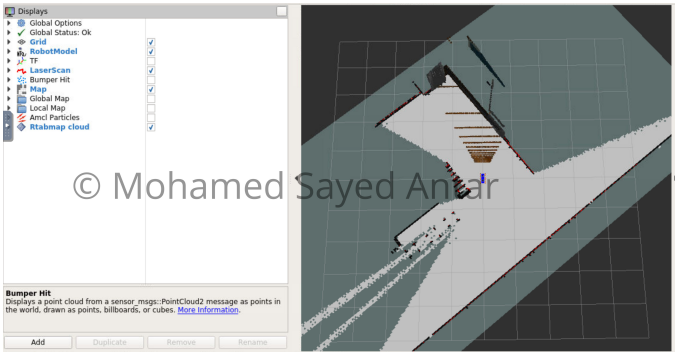

Fig. 10. first-RViz

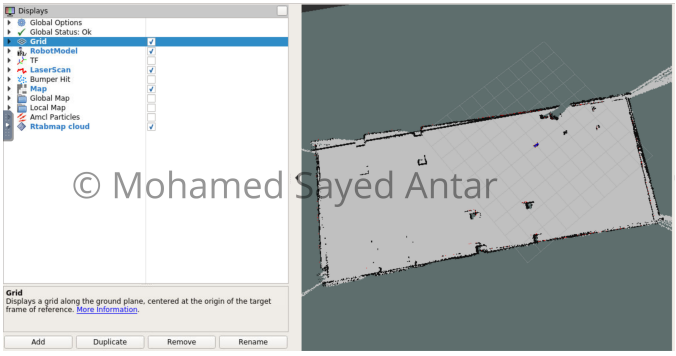

Fig. 11. second-RViz

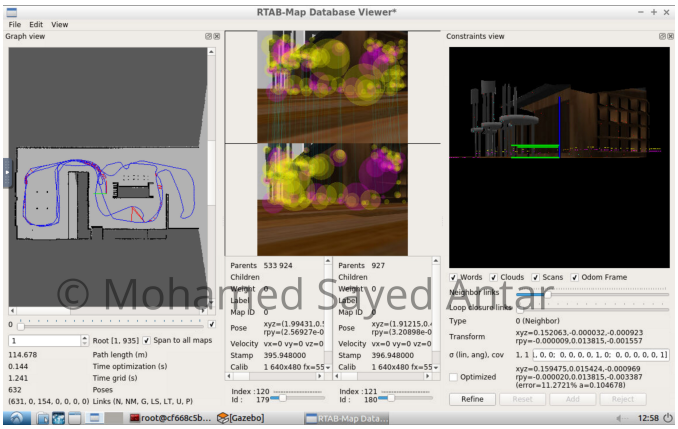

Fig. 12. first-Env

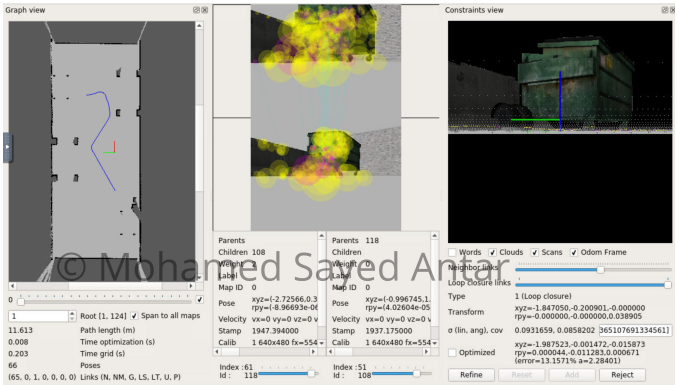

Fig. 13. second-Env

specific place several times this small error increase with time too so after several minutes the total error is big and this is very bad thing, for this reason as the robot go through the same place again it will build another map this map in not on the same position of the previous one with respect to the robot poses, reducing the error of the measuring sensors will reduce the error but not as expecting, as increasing the moved distances accumulate the error to eventually affect the total map.

6 Conclusion / Future work

putting the camera in a suitable place on the robot to be in a proper distance from the ground is very important to make it easy for the robot to discover the around world clearly.