RoboND – SLAM *'Map My World'* Project

Abstract

In robotic mapping and navigation, **simultaneous localization and mapping** (**SLAM**) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track and estimating the pose of the robot that's navigating within it. In this project, SLAM is implemented using RTAB-Map, where a simulated robot was configured with an RGB-D camera and laser range finder to generate a map of two different environments simulated in Gazebo. The robot was integrated with Robot Operating System (ROS) packages to implement RTAB-Mapping to create a 2D occupancy grid and 3D octomap of each environment.

Introduction

The localization and mapping problem is a fundamental one for any robot navigating in unknown environments. So, any robot that needs to move about autonomously in a space needs to solve the problem of localization to know its current position in the world. In case of unknown or dynamic environments where a map is not available or constantly changing, the robot also needs to build and update the map in real time. As of now there is no general solution that works in all cases but rather algorithms that solve a specific problem for a particular use case.

In this project a robot model uses a Simultaneous Localization and Mapping (SLAM) technique called RTAB-Map (Real-Time Appearance-Based Mapping). It is a RGB-D Graph Based SLAM approach that uses incremental appearance based loop closure detection. The robot must map a provided simulated environment which is called "Kitchen Dinning", then it must map another environment that shall be designed and developed in Gazebo, this one is called "My World". The two environments shall be mapped in both 2D and 3D by using Gazebo and Rviz simulation frameworks.

The RTAB-Map ROS wrapper is leveraged with visual representation in real time via rtabmapviz. The resultant map is stored in local database that will later be explored via rtabmap-databaseViewer.

Background

SLAM or simultaneous localization and mapping, is a series of complex computations and algorithms which use sensor data to construct a map of an unknown environment while using it at the same time to identify where it is located. This of course is a chicken-and-egg type problem and in order for SLAM to work the technology needs to create a pre-existing map of its surroundings and then orient itself within this map to refine it. The concept of SLAM has been around since the late 80s but we are just starting to see some of the ways this powerful mapping process will enable the future of technology. SLAM is a key

driver behind unmanned vehicles and drones, self-driving cars, robotics, and augmented reality applications.

SLAM enables the remote creation of GIS data in situations where the environment is too dangerous or small for humans to map. Moreover, since robots live in the 3D world, and it is needed to represent that world and the 3D structures within it as accurately and reliably as possible. 3D mapping would give us the most reliable collision avoidance, and motion and path planning, especially for flying robots or mobile robots with manipulators.

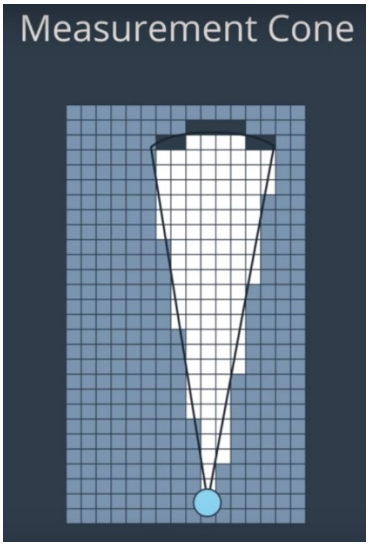
However, acquiring maps is a challenging problem due to a number of reasons:

- Size – the bigger the environment relative to the robot's perceptual range, the more difficult it gets to acquire a map
- Noise – because sensors and actuators aren't noise-free, it becomes more difficult to map an environment the larger the noise becomes.
- Perceptual ambiguity – when a robot frequently visits different places that very similar, the robot has a difficult time establishing correspondence between different locations traversed at different points in time.
- Cycles- pose another problem the robot. If a robot were to only move up and down a hallway, it can correct it odometry errors incrementally coming back. However, cycles make a robot return from different paths. When a cycle closes, the accumulated odometric error can be large.

There are different algorithms that tackle the problems in their own way. In this paper, the algorithms that will be covered are the Occupancy Grid Mapping, Grid-based FastSLAM, and GraphSLAM.

Occupancy Grid Mapping

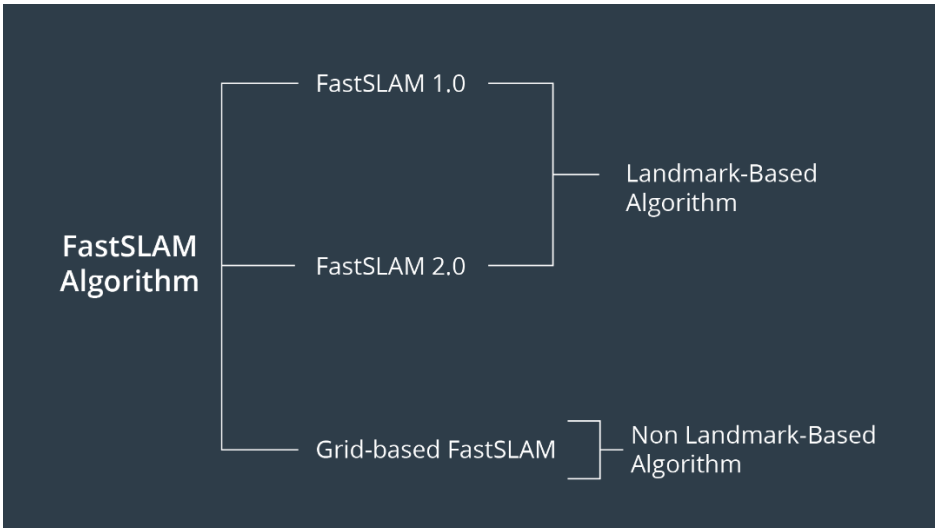
Unlike SLAM algorithms, the occupancy grid mapping addresses the issue of creating reliable maps from noisy and uncertain measurement data, under the belief that the robot pose is known. The idea of this algorithm is to represent the map as a field of random variables, organized in a uniformly spaced grid. Each variable is binary and relates to the occupancy of the location it covers.



The main use of this algorithm, however, is for postprocessing. Because SLAM techniques do not produce maps that are suitable for planning and navigation, occupancy grid mapping is often used to complement SLAM. This algorithm is used after solving the SLAM problem by other means and taking the resulting path estimates given. This method is limited to 2D mapping applications.

Grid-based FastSLAM

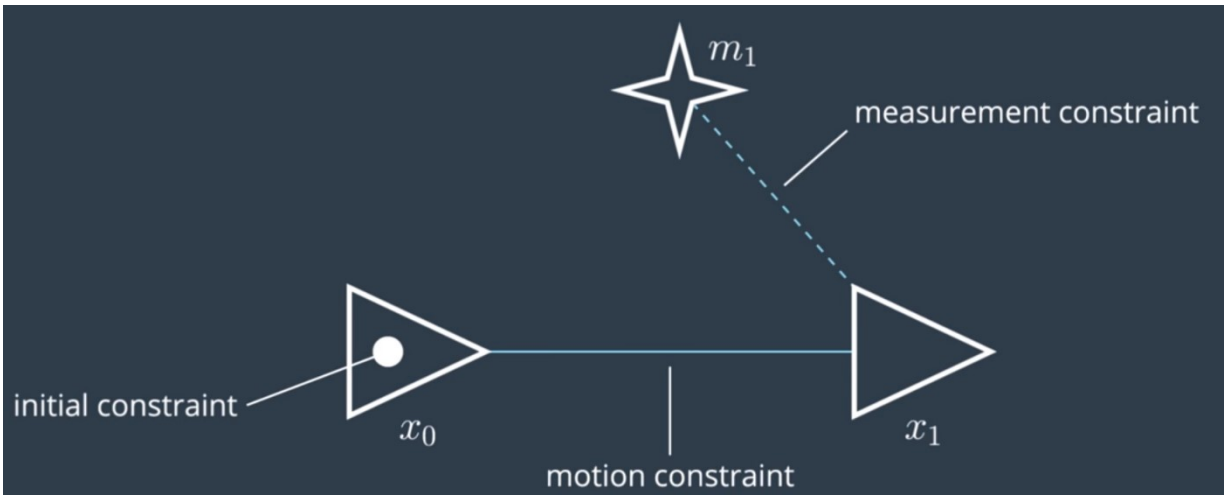
The basic idea of FastSLAM is to preserve a set of particles to approximate a posterior over the trajectory and it uses low dimensional EKF, or Extended Kalman Filter, to solve independent features of the map which are modeled with a local Gaussian. Because FastSLAM estimates for the robot's full path, this technique solves the full SLAM problem. However, each particle in FastSLAM approximates the robot's immediate pose, thus this technique also solves the online SLAM problem.



There are different types of FastSLAM, two of which are iterative upgrades of the FastSLAM algorithm, FastSLAM 1.0 and FastSLAM 2.0, and the other is an extension of the FastSLAM algorithm, Grid-based FastSLAM. The biggest disadvantage of FastSLAM is having to always assume that there are known landmarks, thus hindering its ability to model an random environment. This is where Grid-based FastSLAM fills in the gap. This technique keeps the FastSLAM's biggest advantage of using particle filter to solve the localization problem in SLAM, but it extends the algorithm to occupancy grid maps. Grid mapping algorithm can model the environment using grid maps without predefining any landmark position which solves the fundamental problem of FastSLAM.

GraphSLAM

GraphSLAM addresses the full SLAM problem, which means the algorithm recovers the entire path and map. The advantage of this algorithm is the reduced onboard processing capability and the improved accuracy compared to FastSLAM. Because FastSLAM uses particles to estimate its pose, there's a possibility that a particle doesn't exist in the most likely position. This is especially true in large environments. GraphSLAM in contrast can work with all of the data at once to find the optimal solution. GraphSLAM is quite simple. It extracts from a data set of soft constraints, represented by a graph like the one shown below.

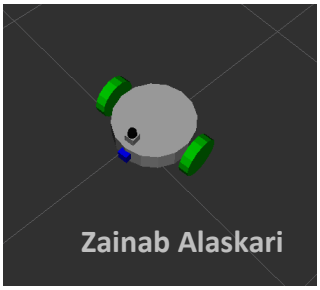


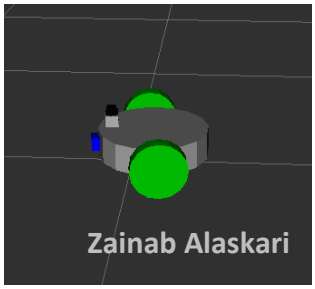
After, it recovers the map and the robot path by solving these constraints into a generally reliable estimate. Usually these constrains are non-linear, therefore, they are linearized in the process. This is keeps iterating until the optimal solution converges.

RTAB-Map (Real-Time Appearance-Based Mapping) is a RGB-D, Stereo and Lidar Graph-Based SLAM approach based on an incremental appearance-based 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. A memory management approach is used to limit the number of locations used for loop closure detection and graph optimization, so that real-time constraints on large-scale environments are always respected.

Scene and robot configuration

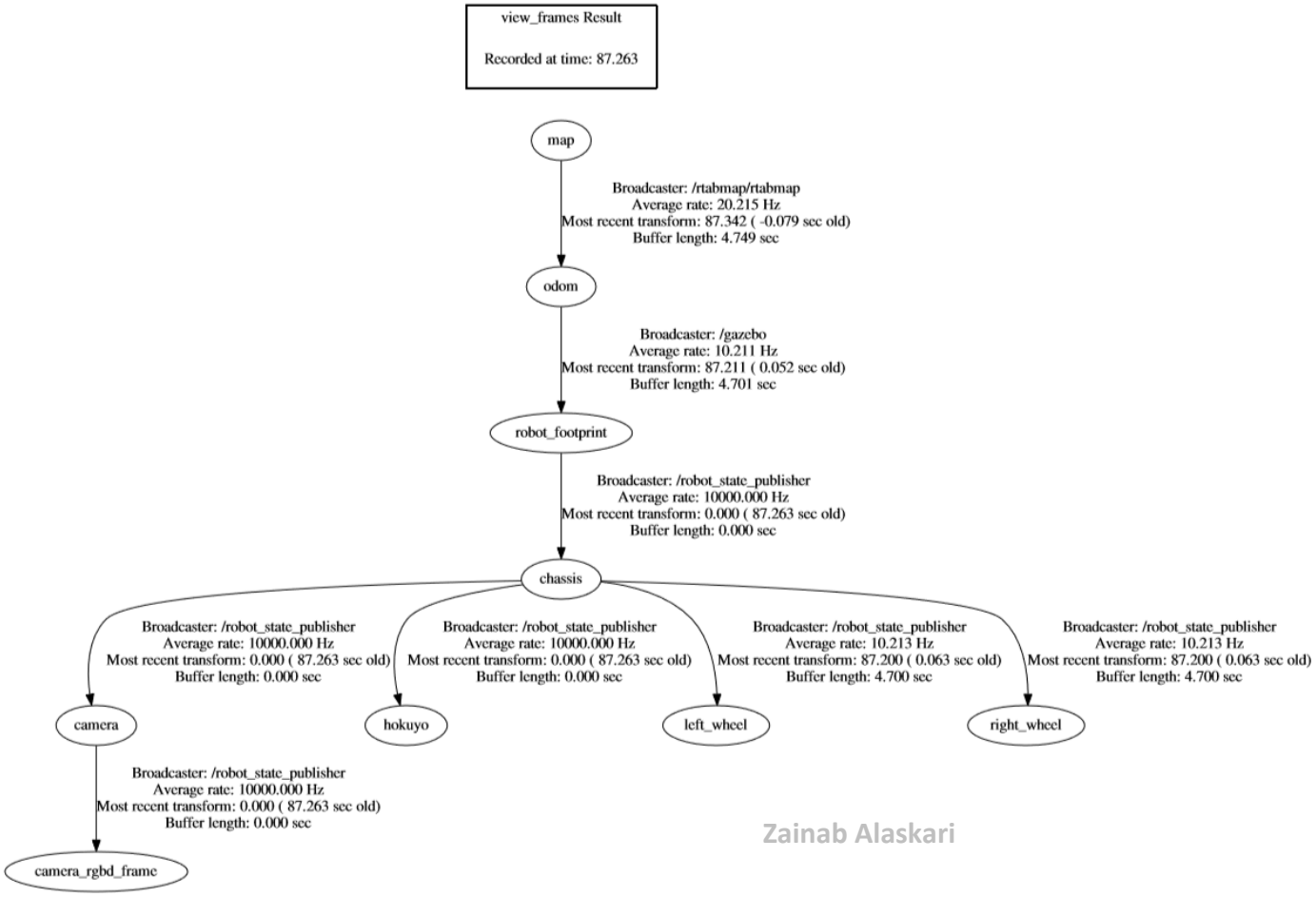
Model Configuration





For this project, the robot creation from the previous localization project which has a cylindrical shape, with two cylindrical wheels, and to spherical casters was extended by upgrading the constructed RGB camera to an RGB-D one which allows it to measure the depth of its environment. This, along with its existing 2D laser rangefinder (Hokuyo) sensor, allows the robot ROS package slam_project to leverage the rtabmap-ros package to perform SLAM. Since manual mapping is to be performed the slam-bot package communicates with a teleop package that allows the user to move and steer the robot.

The backend configuration of the robot can be better appreciated in the Transform tree as seen in the following graph.

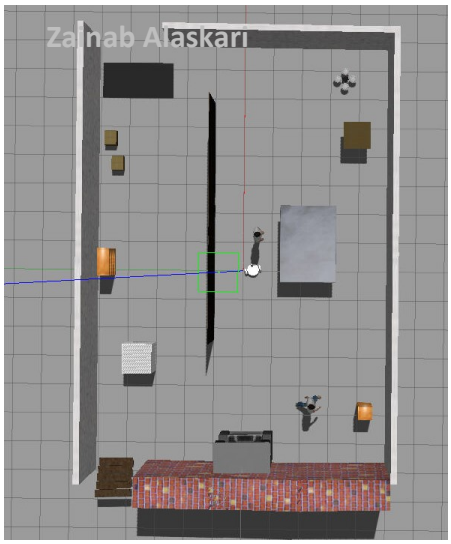


World Creation

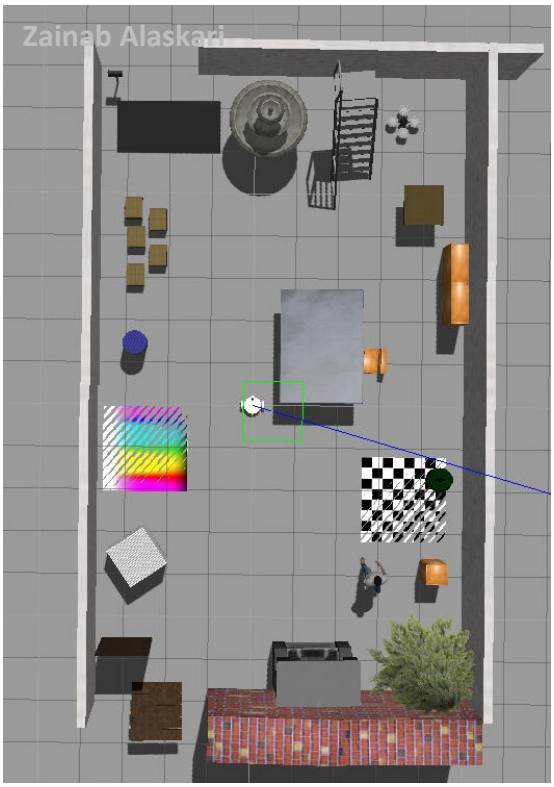
Many simulated Gazebo world environments were created and tested.

The below figures show few examples of the Gazebo world creations.

This was the first attempt at a simulated environment. The environment is a room with four walls and small objects such as tables, boxes, ect. Along with two human models.



At the second attempt, more objects were added to enhance the mapping by adding a ladder, a stair case, a tree, etc.



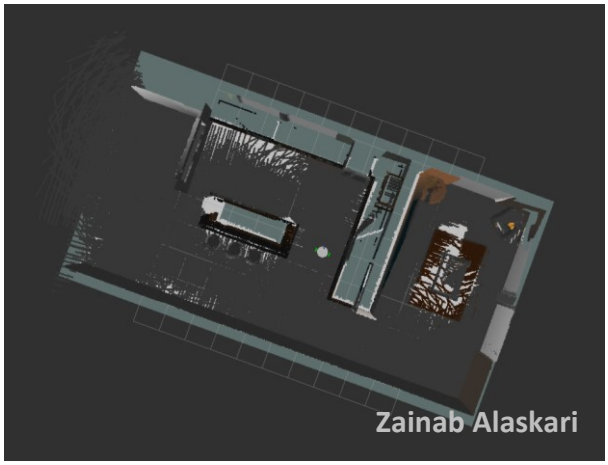
More attempts were done to enhance the mapping by trying to add more features in the room environment next to each wall to support the loop closure detection of the mapping algorithm. The size of the room was also reduced.

Results

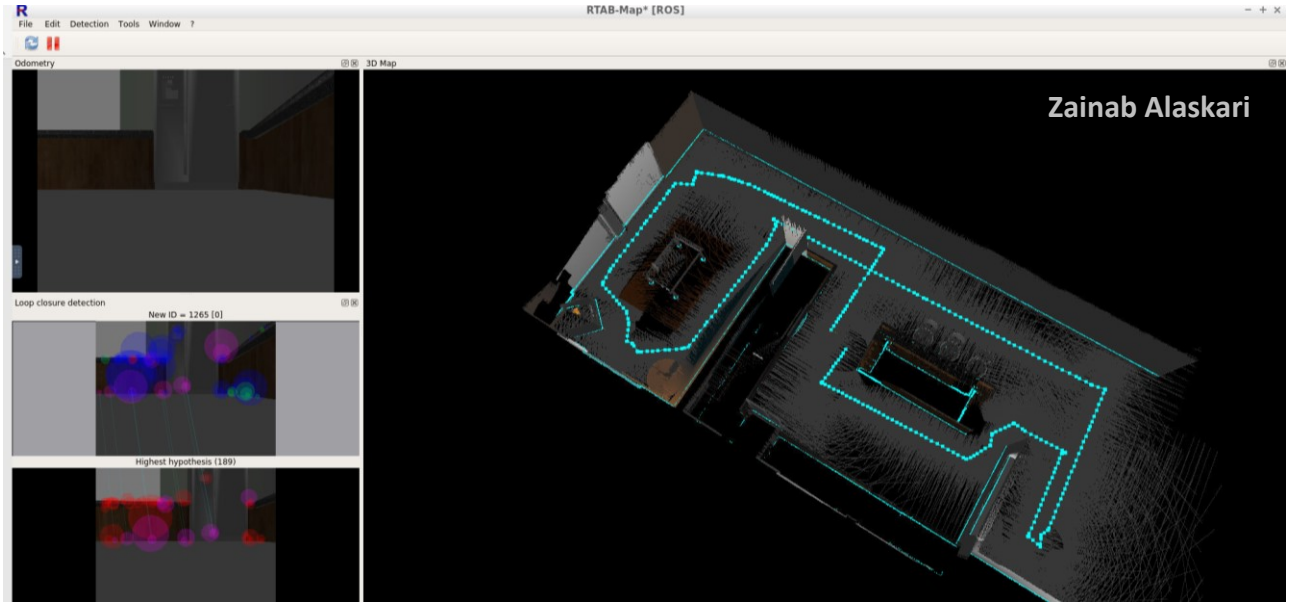
Kitchen dining scene

Upon launching the Kitchen and Dining scene simulation, and launching the mapping node, the my_bot started mapping the environment in its immediate vicinity. After traversing and navigating around the entire world, the my_bot was able to generate an accurate map of the world that was provided. As can be seen from the rtabmap database, several loop closures were observed, and both a 2D occupancy grid map and a 3D octomap were generated for the provided environment.

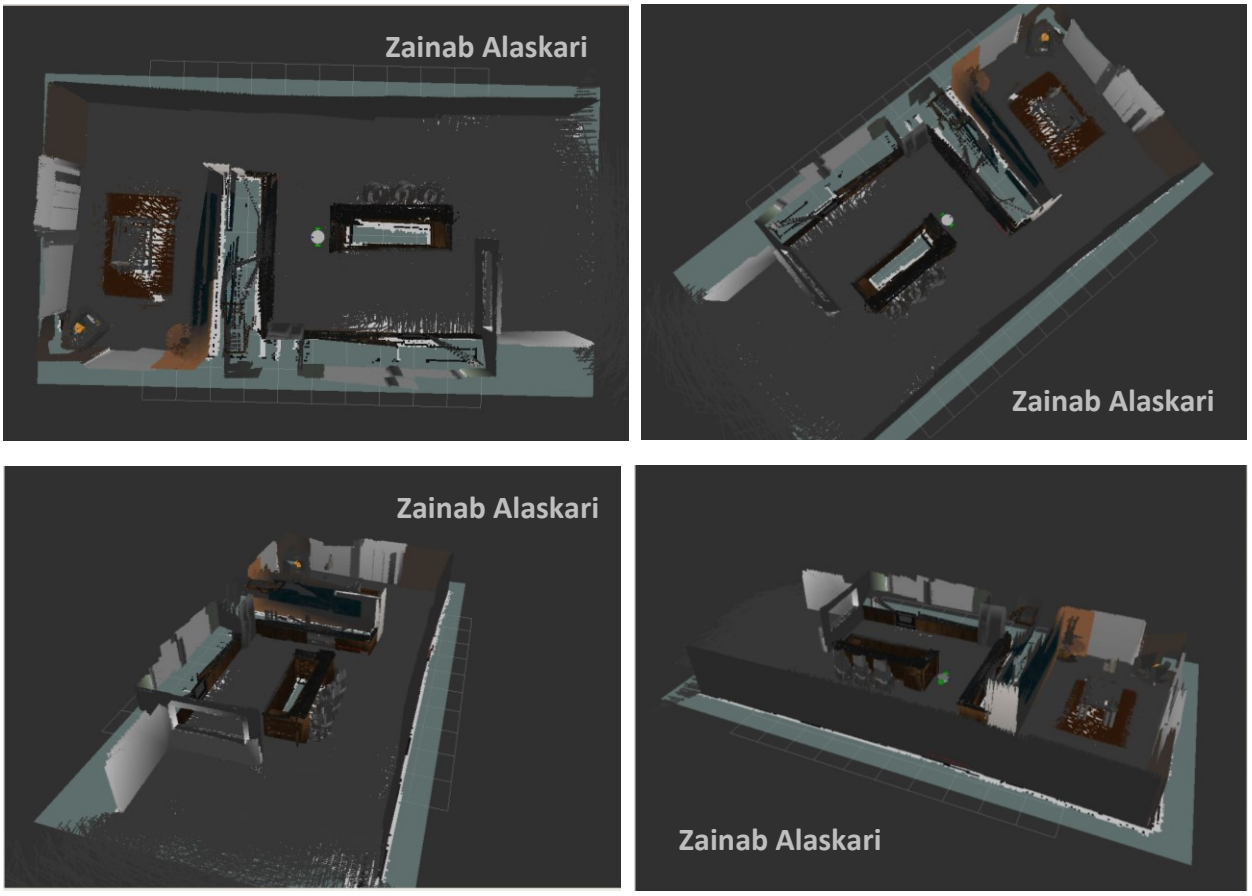
3D Visualizations in Rviz of the world as the robot makes its first pass through the environment.



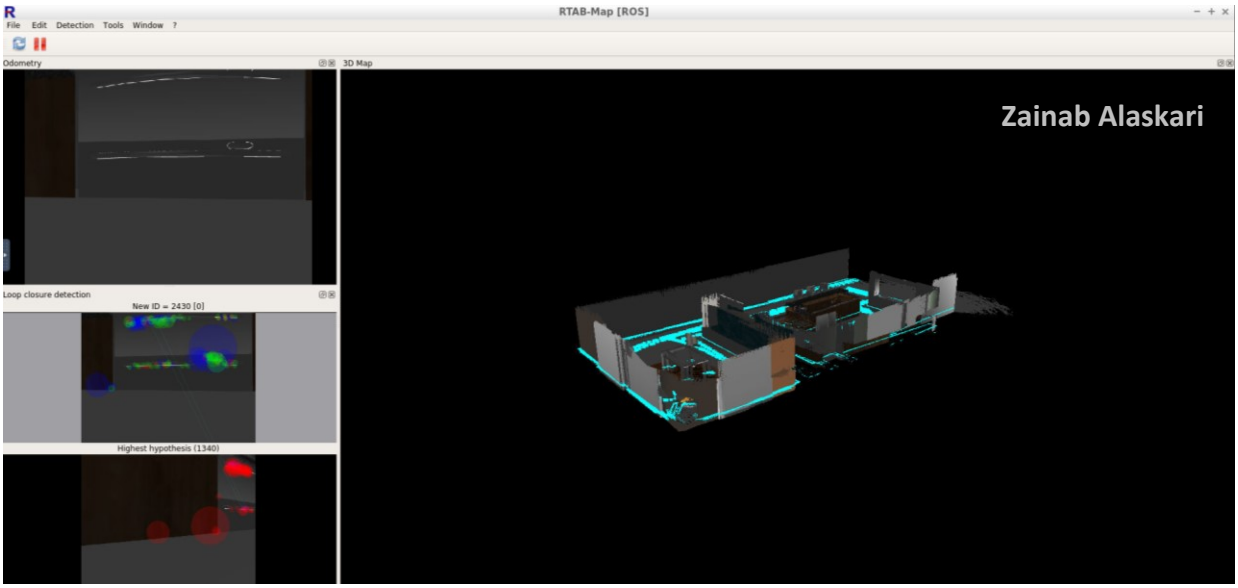
Live 3D visualization of the generated map in rtabmapviz showing realtime feature detection and loop closures as the robot makes its first pass through the environment.

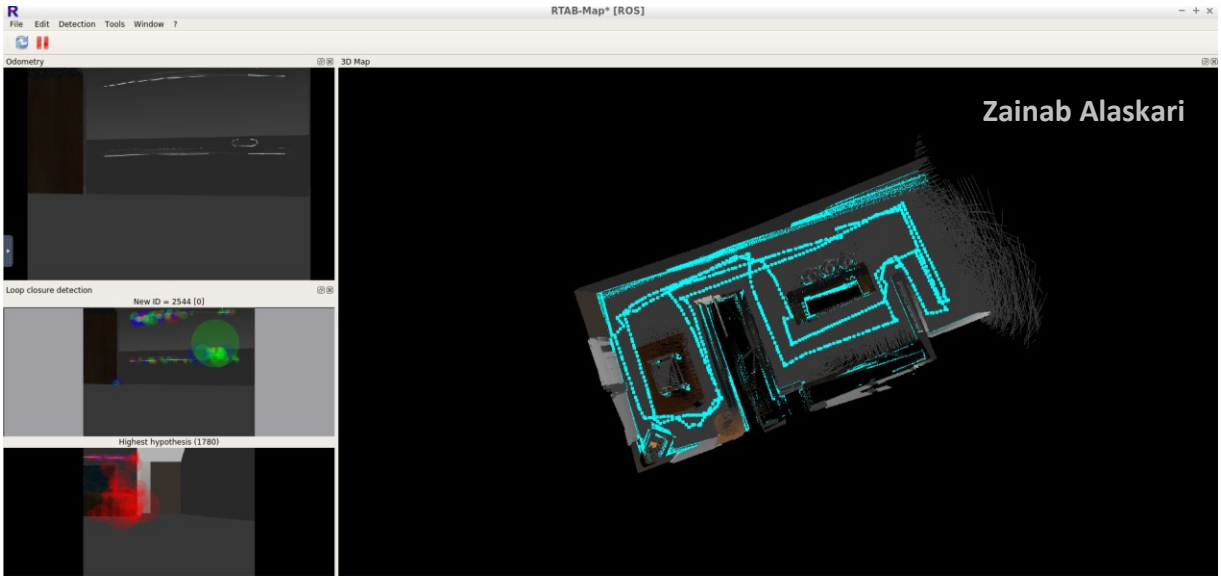


3D Visualizations in Rviz of the world as the robot makes its last pass through the environment.

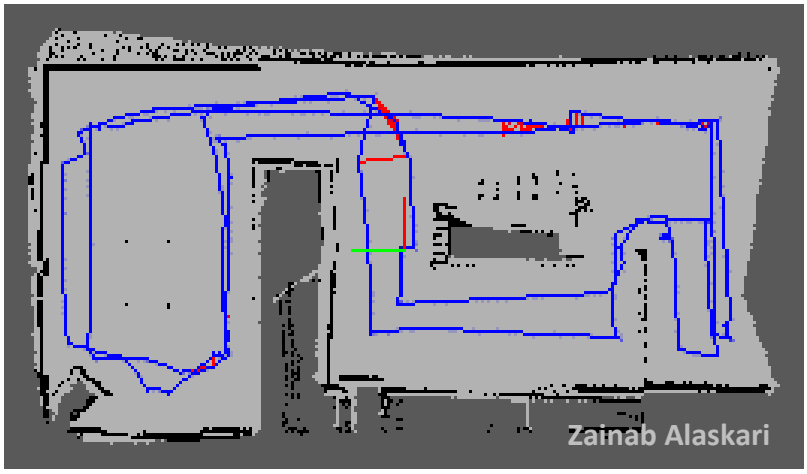


Live 3D visualization of the generated map in rtabmapviz showing realtime feature detection and loop closures as the robot makes its last pass through the environment.



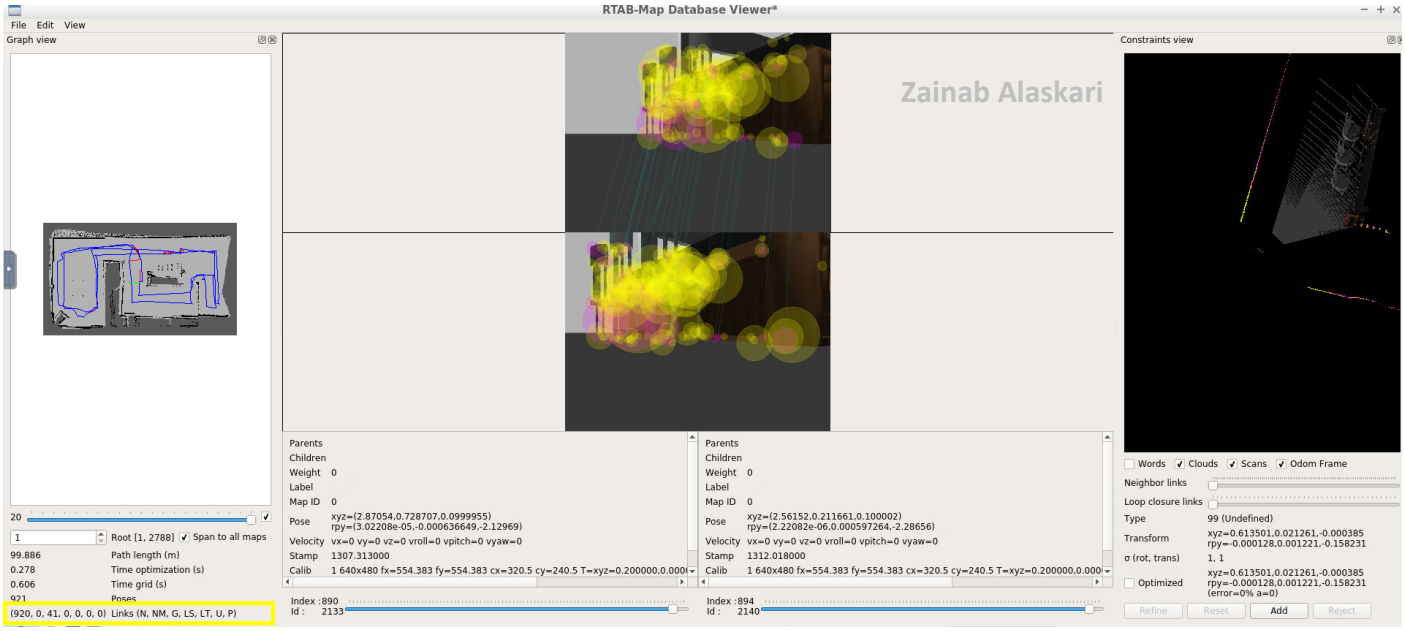


This is the generated 2D occupancy grid map.



rtabmap-databaseViewer tool used to explore the generated database.

On the left, the 2D grid map in all of its updated iterations and the path of your robot can be seen. In the middle different images from the mapping process can be seen with all of the features from the detection algorithm. These features are in yellow. The pink indicates where two images have features in common and this information is being used to create neighboring links and loop closures. On the right the constraint view can be seen. This is where it is identified where and how the neighboring links and loop closures were created.

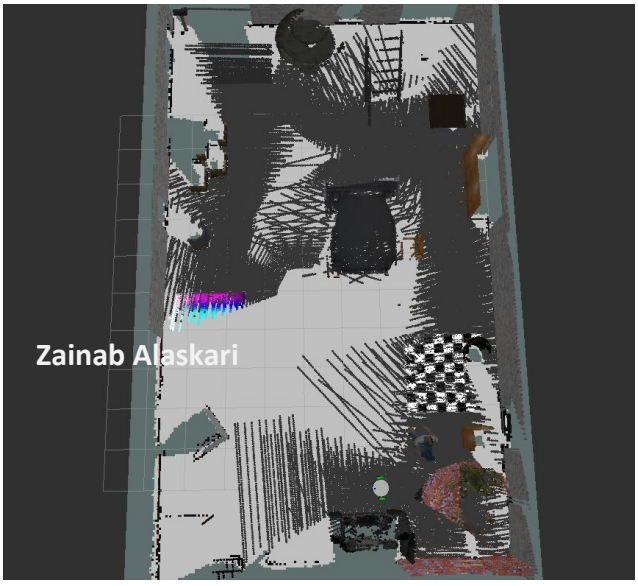


The total number of loop closures detected is 41, this value can be seen in the bottom left of the image.

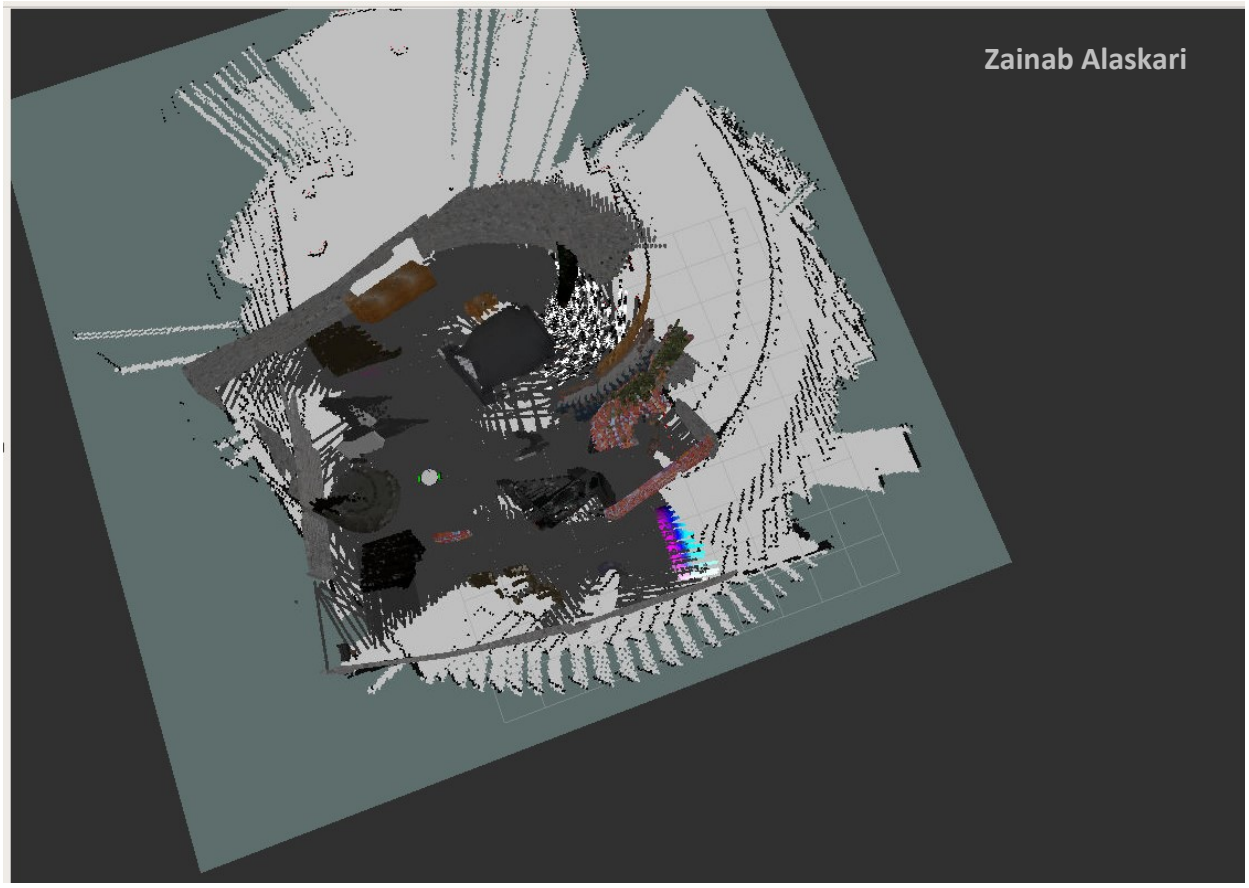
Custom world scene

Many custom worlds were created and tested as was mentioned earlier.

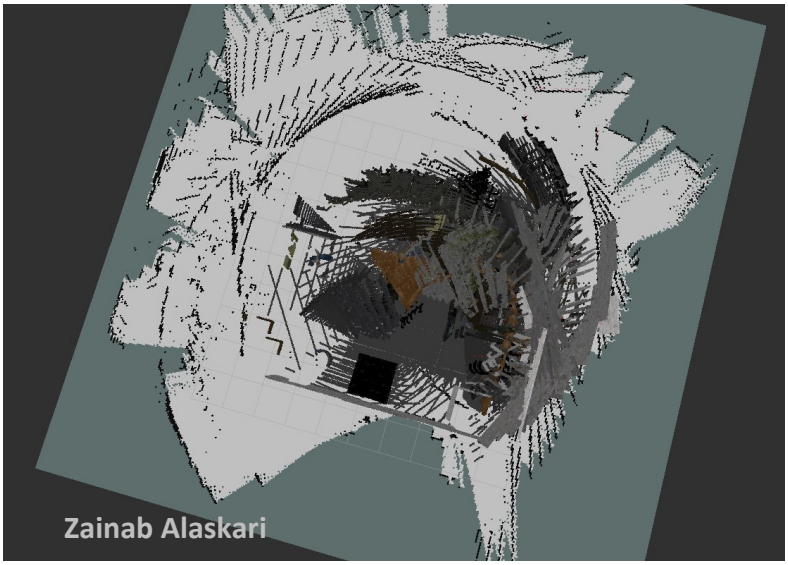
This world creation gave a good mapping results initially.

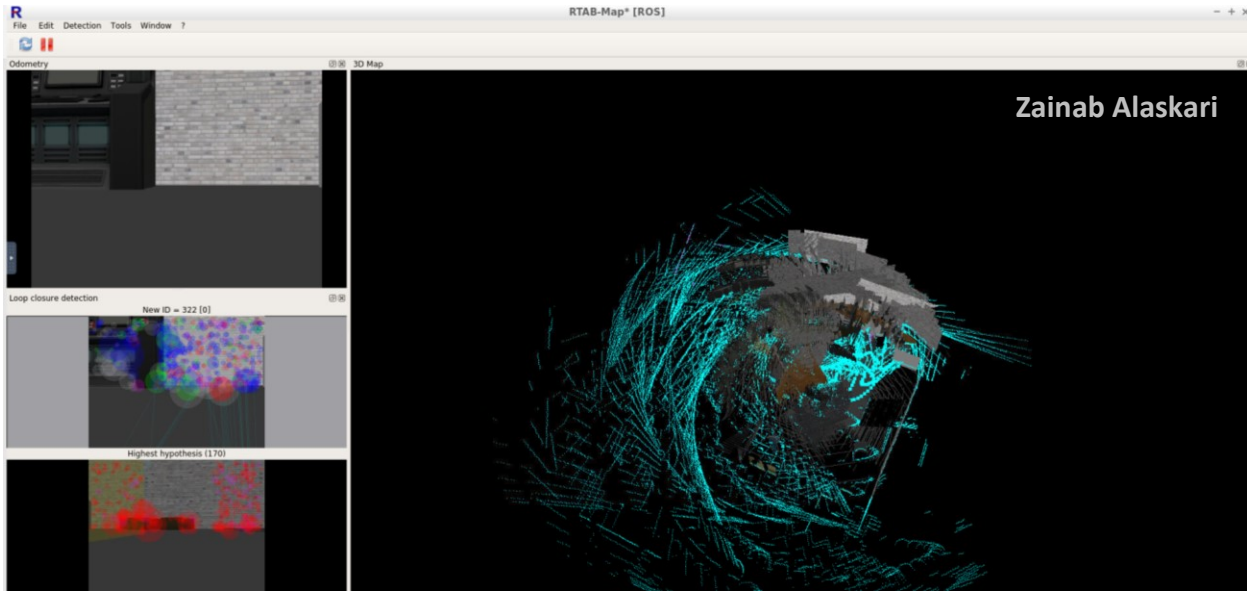


As the my_bot navigated through the environment, the map got distorted due to the lack of features in the environment.

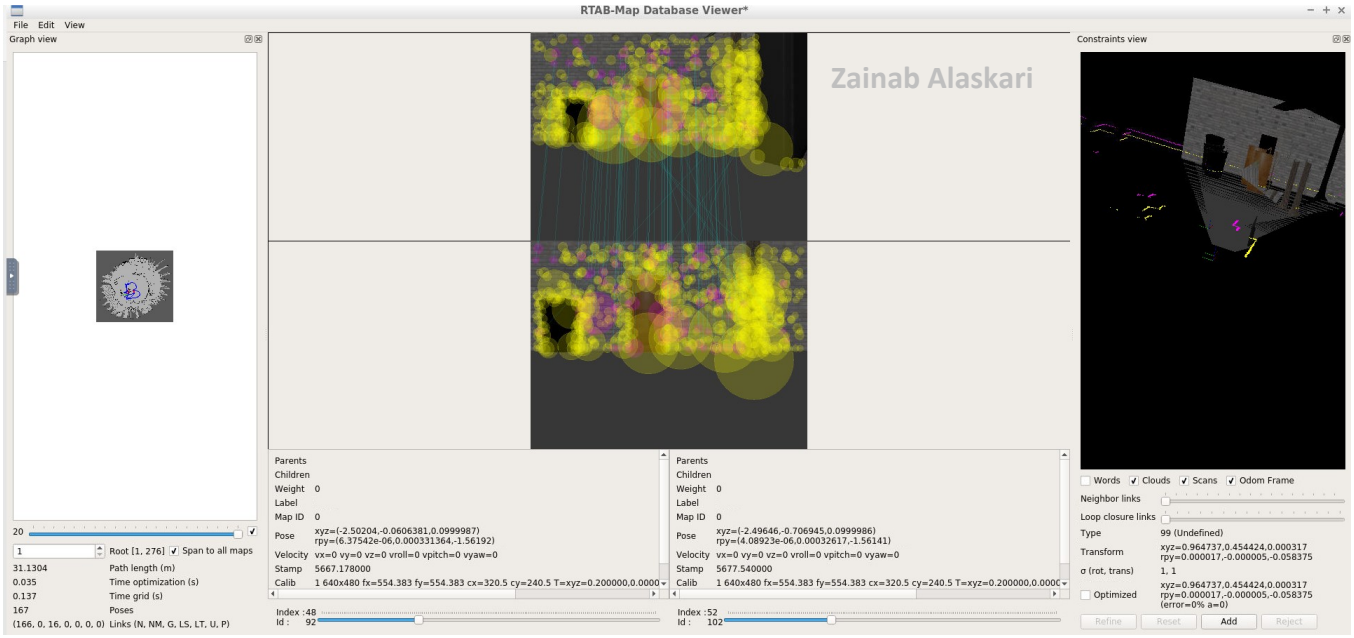


This was the last world creation that was tested, and these were the results.





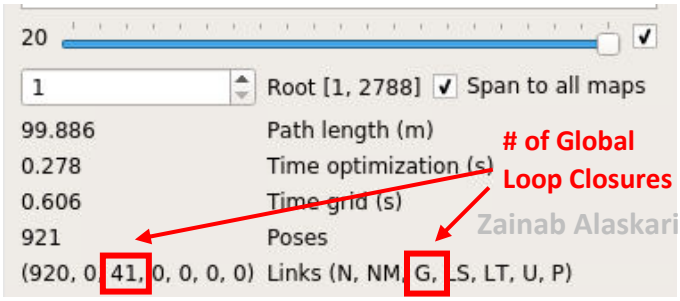
rtabmap-databaseViewer showing the poor 2D occupancy grid map generated along with the number of loop closures detected which is 16. Many of them are false-positives, hence the bad accuracy mapping.



Discussion

Mapping the Kitchen dining scene was successfully accomplished, two robot passes were made in order to generate the map.

rtabmap-databaseViewer shows that 41 loop closures were detected. This shows that the robot has been over similar paths more than once which allows for the maximization of feature detection, facilitating faster loop closures.



The custom world was not mapped properly because the walls of the simulated world were not feature rich enough. Unique landmarks should have been placed such that the robot can see an object when turned in any direction as long as it did not directly face a wall. This would help support the loop closure detection of the mapping algorithm and hence increase mapping accuracy.

Comparing the mapping of the two environments, the accuracy of the mapped kitchen dining scene is much higher than the custom scene due to its unique and complex landmarks and feature rich walls. Mapping the kitchen dining scene was achieved on the first try while mapping the custom designed world took sever trials and tests and was still not achieved successfully. The robot was unable to localize itself and distort the map because the RTAP-Map produced false-positives of loop closures.

Future Work

RTAB map uses the Bag of Words approach to identify reoccurring objects. However, the method needs tuning and sometimes produce false positives of loop closures. For example, in the “My World” scene, the robot often identifies different sections of the wall as the same section, or two similarly shaped objects as the same one. When such false positives occur, the created map would be heavily distorted and unable to be used. Object detection and classification algorithms with better precision, such as deep neural network, might be able to reduce false positives and enable the robot to map similarly well in less feature-rich environments. In addition, the RTAB map does not perform well when there are dynamic objects in the environment. Since the RTAB map assumes the objects to be static in the environment, when it detects an object appearing at different locations, it will be unable to produce the correct constraints and result in a low-quality map. A more advanced detection algorithm would be required to distinguish a moving object from a dynamic object.

It is believed that RTAB Map algorithm can be implemented at home for vacuum cleaner or lawn mower robots, where these environments can stay static for some time to allow for efficient mapping. In general, RTAB Mapping can be used with any autonomous vehicle that moves in a static environment.

Moreover, SLAM is central to a range of indoor, outdoor, in-air and underwater applications for both manned and autonomous vehicles. SLAM in general can be applied for the following applications:

- Air: surveillance with unmanned air vehicles
- Underwater: reef monitoring
- Underground: exploration of mines
- Space: terrain mapping for localization