Project: Map My World Robot

Abstract

Mapping is the task of building a map of the environment of the robot, given a set of poses for the robot. It involves using the poses of the robot to estimate the map by establishing correspondences in the environment. A step beyond this is Simultaneous Localization and Mapping (SLAM). SLAM involves building a map of the environment while simultaneously localizing relative to that map. SLAM is a challenging problem, since the map requires localization and localization requires a map of the environment. By the end of this report, a successful method has been implemented to build a map of the environment by solving SLAM using the RTAB-map algorithm.

Introduction

The aim of the project is to create a robot model equipped with an RGBD camera. The model is moved around a gazebo world using keyboard teleoperation to build a map of the world using RTAB-mapping, by modifying the algorithm's parameters. The world is simulated model of a kitchen and a dining room. The second part involves building a different simulated environment in gazebo. Following this, a map of the new environment is built similarly. Further, the loop closure and the correspondences are studied using Rtabmap database viewer. Thus, the final aim is to successfully build both 2D and 3D maps of the environment.

Background

Localization is the process by which the robot can figure out where it is in the current map, what is the position and the orientation of the robot with respect to it's environment. It plays a important role in helping the robot decide where and how to move next. Mapping is similar to the reciprocal of the localization, i.e. it builds the map given a series of poses for the robot. Mapping is needed to build the environment the first time so that the robot can move around and localize itself in the world in the future. Mapping cannot be separated from localization and rhis leads to SLAM. SLAM is the task solving both these conflicting problems simultaneously.

The challenge in robot mapping is that contrary to estimating the pose, the map is highly dimensional. So, we need a lot of variables to describe the map. The uncertainties in perception also make the task difficult. Further, establishing the correspondences in the map can be challenging especially when the environment is repetitive or does not have too many distinguishing features. The difficulty in SLAM is that both the robot's motion and the mapping has uncertainties. These uncertainties are often correlated and make the task challenging.

There are five categories of SLAM algorithms:

1. Extended Kalman Filter
2. Sparse Extended Information Filter
3. Extended Information Form
4. FastSLAM
5. GraphSLAM

EKF SLAM attempts to solve the SLAM problem using extended Kalman filters, which are variants of Bayeian filters. EKFs can be used to approximate nonlinear estimation problems, using a prediction and update step. All probabilities are assumed to be Gaussian.

The EIF SLAM uses extended information filters. EIF is a similar to an EKF, but uses a different parametrization to represent the Gaussian probabilities. The EIF is a dual of the EKF. The complexity of the prediction and update steps gets inverted for the EIF, as compared to the EKF.

The sparse extended information filter is an extension of EIF. The SEIF algorithm has an additional step in which the data is made sparse, i.e. links are established between the robot's pose and only the nearby landmarks. This reduces both the time complexity and memory complexity of the SLAM algorithm, but it might lead to a drop in the quality.

FastSLAM uses particle filters with low dimensional EKFs to solve the SLAM problem. While EKF uses one high dimensional EKF filter, FastSLAM uses low dimensional EKFs, for each particle. Given the use of particles, which are resampled based on importance, this filter is more robust.

GraphSLAM builds a graph using constraints based on relationships between the robot pose and it's environment. It then tries to find a solution that resolves these constraints, and uses this solution to create the most likely map. However, this requires high memory for the computations, but it increases the accuracy of the estimation.

Scene and robot configuration

The custom gazebo world is a closed room with an entryway, and a few objects placed around the room. There is an obstacle set in the center of the room, and various objects around the rest of the room. The robot goes around the central obstacle, and maps the world using the objects around as features for correspondences and loop closures. The obstacles around the room mainly consist of common objects such as tables, cabinets and cupboards. A few other objects are added to add distinguishable features to the room.

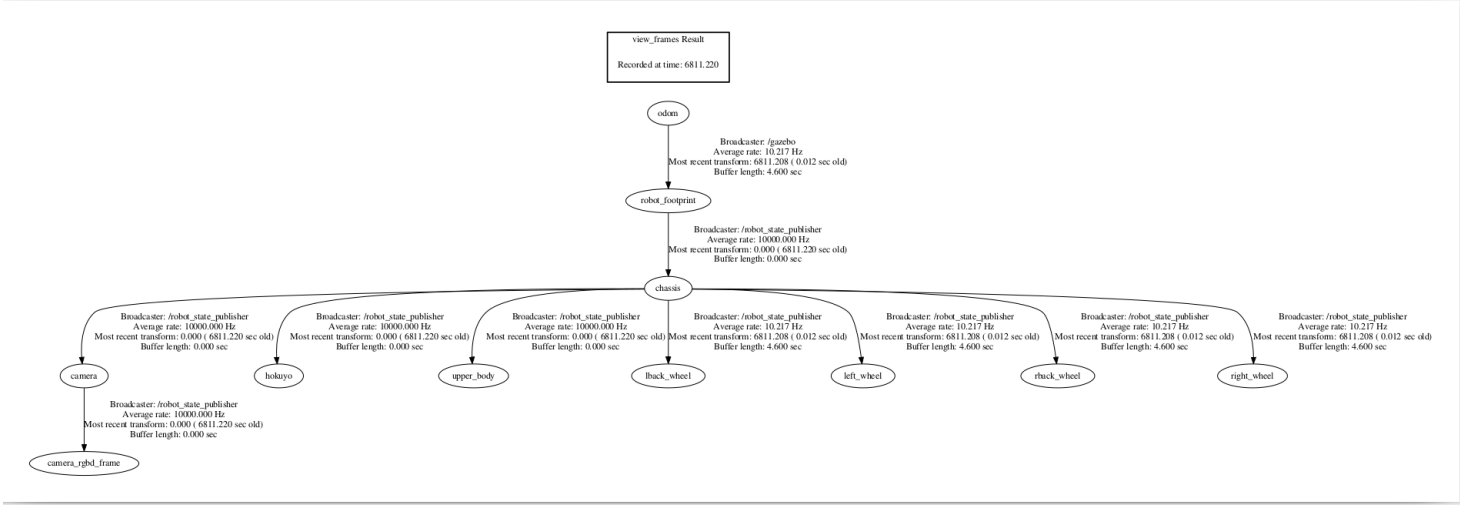
The robot model is a four wheeled skid steer robot, with a 2D Lidar and an RGBD Kinect camera. The robot has a dimension of 0.4 (length) by 0.6 (breadth) by 0.45 (height). The camera is located at around 0.2 m above the ground and it is slightly set in compared to the outermost point of the robot. The lidar is located on the top of the robot.

The package is based on typical ROS package structures. The package folder named slam_project has folders called:

1. launch: Contains all the launch files for all the processes needed to run all the processes. Also contains a config subfolder, which contains the rviz config files.
2. meshes: Contains the mesh file for the 2D lidar
3. urdf: Contains the gazebo and xacro file for the robot model
4. worlds: Contains all the gazebo world files

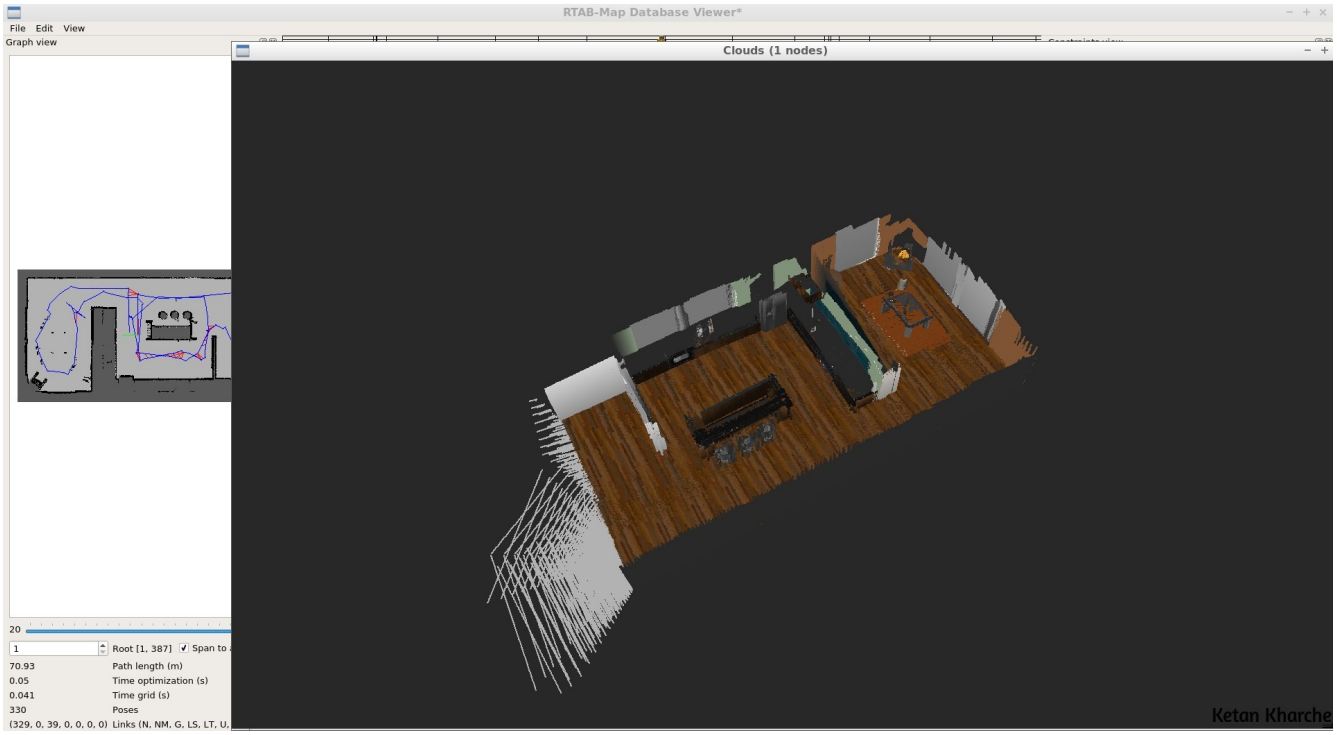
Results

TF tree

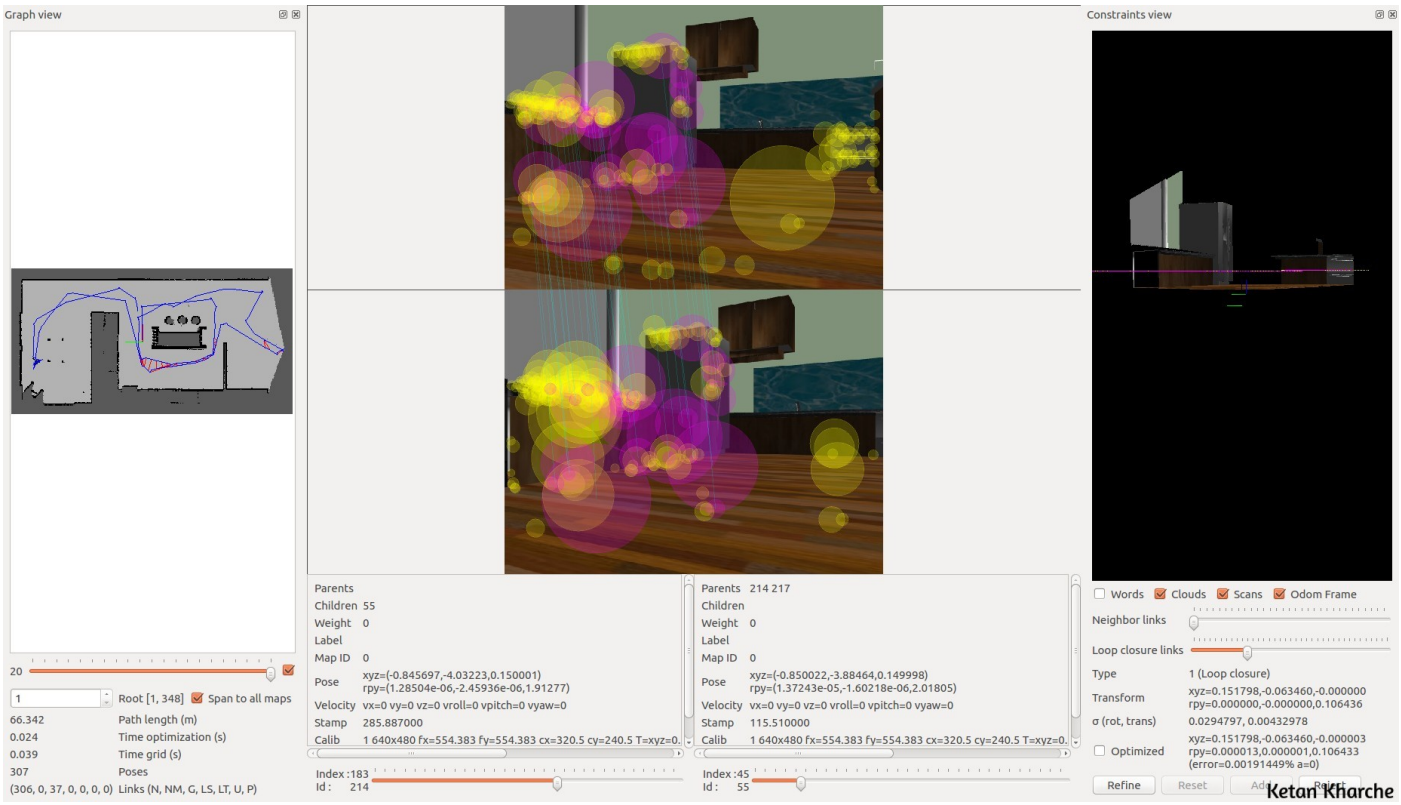


2D map of kitchen & dining world:

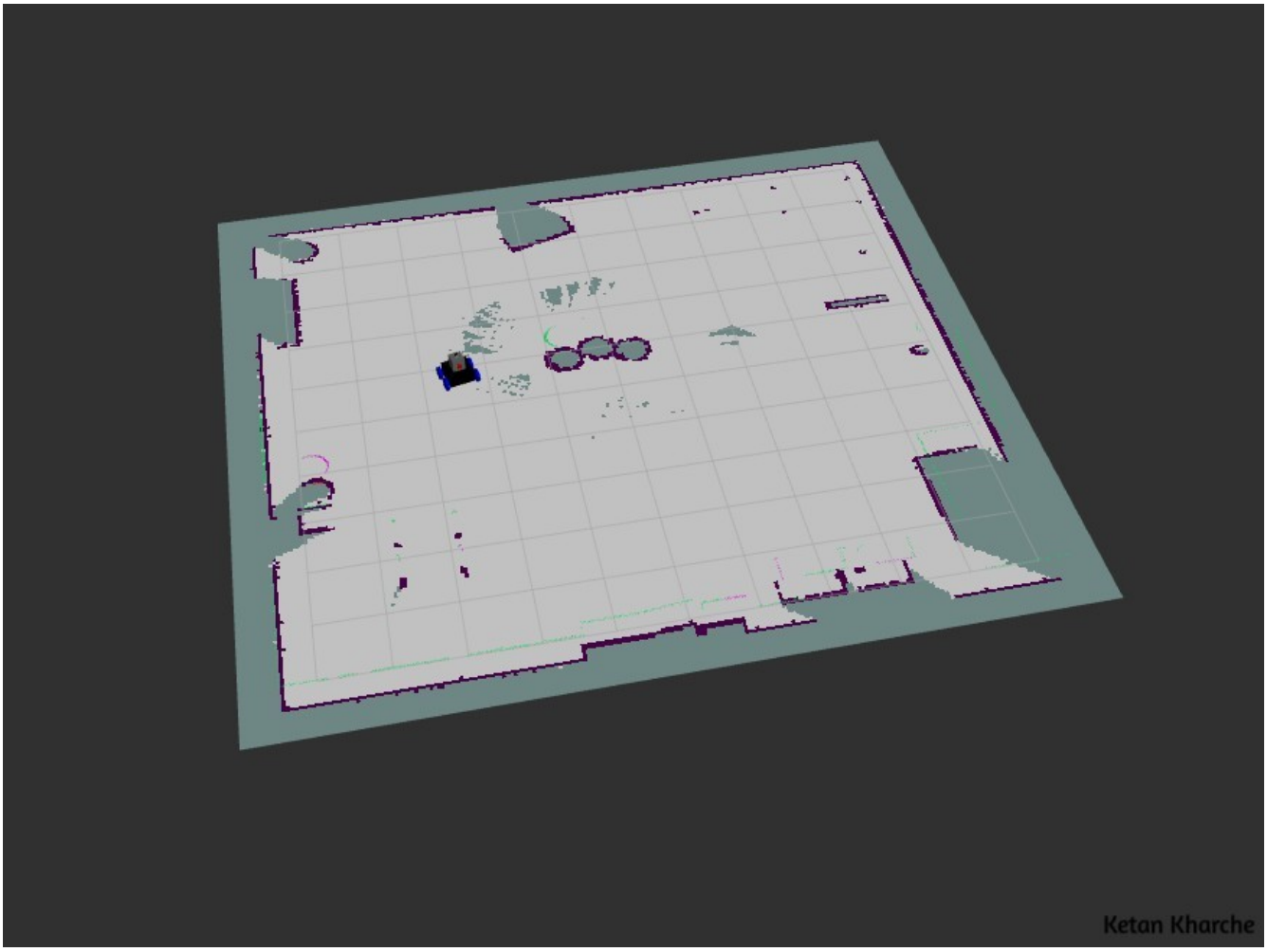
3D map of kitchen & dining world:



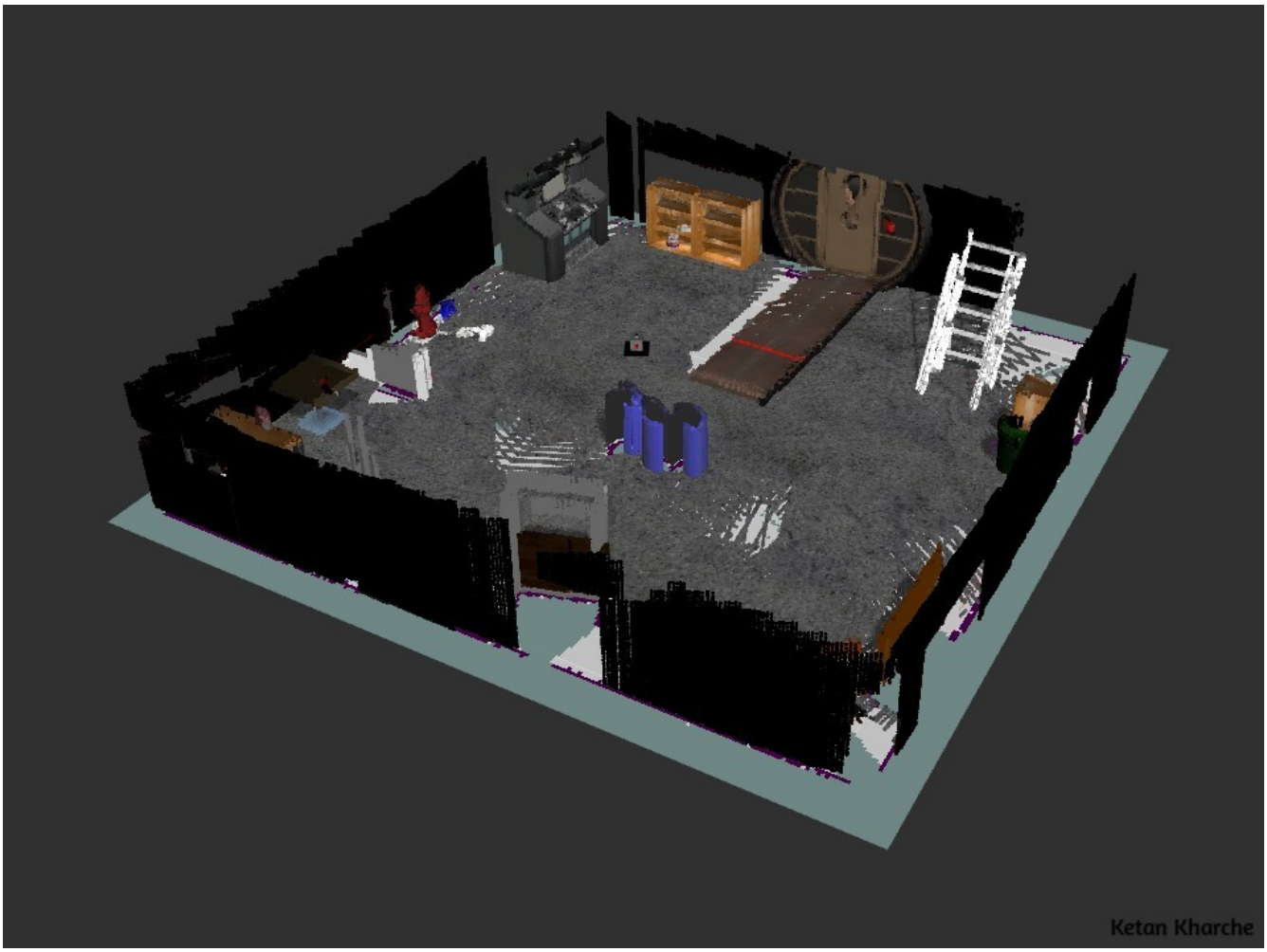
RTAB-map database for kitchen & dining world:



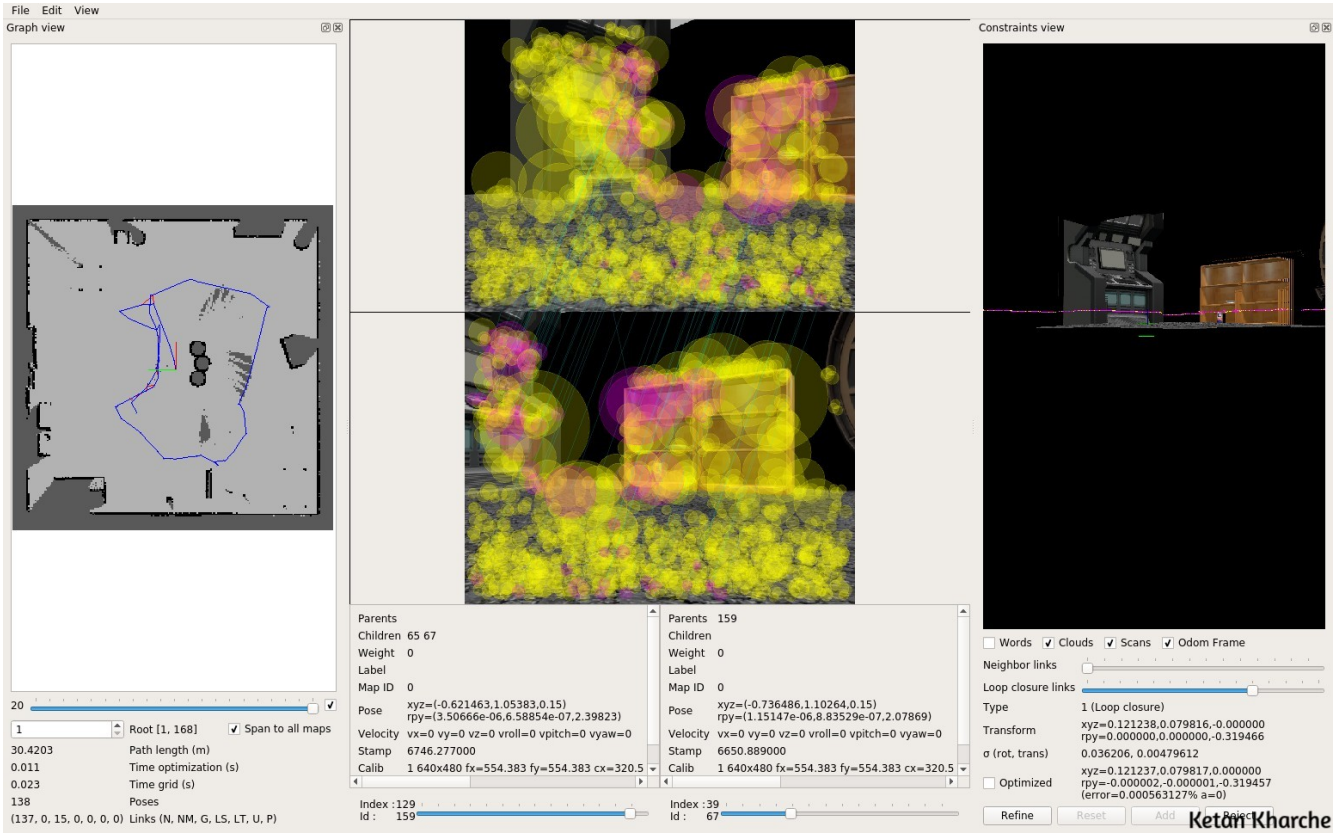
2D map of custom world:



3D map of custom world:



RTAB-map database for custom world:



Discussion

The skid steered robot is able to move around easily using the keyboard teleop, and is successfully able to navigate through the environment.

The mapping works well for both the kitchen dining world and the custom built world. The kitchen dining world is feature rich and not too repetitive. That helps with establishing correspondences and loop closures, and the parameters have to be adjusted accordingly. SURF was used for feature detection in rtabmap. For the environment with distinct features, the number of inliers can be set to 20. In some cases of incorrect matches, the map can shift and turn by a large amount, and ruin the map. That was fixed by reducing the maximum error that was allowed in mapping, by adjusting the OptimizeMaxError parameter, and setting it to 0.5.

The custom world however has more repetitive features, and as a result, there are more false matches than the kitchen dining world. A lot of false matches get registered due to the patterned floor. Initially, this led to the map closing the loop at incorrect places multiple times. As a result, the number of inliers had to be increased to 40 for the new world. The max error was also set to 0.1. This improved the results significantly by reducing the number of false loop closures. The mapping failed multiple times, but it worked after the parameters were tuned.

The performance is definitely better in the kitchen dining world, as the environment has more distinct features. The robot does not get lost as easily, and is able to localize itself more easily.

I was also able to implement the localization using rtabmap. A modified launch file, with the localization parameters from the localization project, worked well, and the robot was able to localize itself and move to given goal positions.

Future Work

The future work involves improving the RTAB-map algorithm to make it more adaptable to environments with fewer distinct features. The next plan is to implement mapping for an outdoor environment, such as a parking lot, or a public park. These environments have fewer consistent and distinct features. However, a successful implementation would be a good trial run for an outdoor robot. I also plan on trying out other feature recognition methods such as ORB or FAST/FREAK or FAST/BRIEF, to see if they perform better in such worlds.

References

1. https://infoscience.epfl.ch/record/146805/files/ekf_fastslam_comp.pdf

2. http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam07-seif-slam-4.pdf

3. http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam06-eif.pdf