My Map World Project Report

Abstract:

This paper implements the Simultaneous Localization and Mapping (SLAM) using the ROS package *rtabmap*. The paper discusses the concept of graph slam, the features of rtabmap and how the rtabmap package was used to generate the map of the environment. Furthermore, the paper describes how to create an environment in gazebo. The difficulties in using rtabmap, the mapping results of the provided environment and the newly created environment are explained. In the end, various possible applications of rtabmap for SLAM are discussed.

Introduction:

Robot localization in a map can be achieved by localization algorithms such as Adaptive Monte-Carlo Localization (AMCL). However, for using the AMCL package in ROS the map of the environment needs to be known prior to localization. In real world, this approach imposes two main limitations. This is discussed below.

Firstly, the same environment can keep on changing. For example, the position of book shelf is changed, or new items are added, etc. In such cases, the original map will keep on changing. Therefore, we cannot rely on the original map in real world applications.

Secondly, there are many applications where robot needs to explore and operate in an unknown area. In such cases, no map can be provided to the robot in advance. The problem in such cases is that the robot needs to map the environment as well as localize itself in the map simultaneously. Initially, as there is no map, the robot cannot localize and as the robot doesn't know where it is, it cannot do the mapping. This problem is often compared to the Chicken and Egg problem.

In this project, SLAM was successfully implemented using the ROS rtabmap package. RTAB-map stands for Real Time Appearance Based map. The rtabmap is a graph-based SLAM. Using rtabmap, the map of the provided environment was successfully created. Furthermore, new environment was created in Gazebo and mapped using rtabmap.

Background:

A robot in an unknown environment or changing environment has to continuously map its environment and also localize simultaneously. This problem is called as simultaneous localization and mapping (SLAM). The SLAM problem can be categorized according to its forms and its nature. The form of SLAM can either be online or offline. An online SLAM is where robot estimates its current pose and the map using the current controls and measurements. An offline SLAM is where a robot estimates its entire trajectory and map using the current controls and measurements. The nature of SLAM can either be continuous or disrcete. The continuous nature is where robot continuously senses its pose and object locations. The discrete nature is where the robot has to identify a relationship between the newly detected object and the previously detected objects, if one exists.

The main challenges in SLAM are due to large amount of data because of continuous and discrete natures. Continuous parameter space includes robot poses and object locations and therefore is highly dimensional. The robot has to keep track of all the objects and therefore as the time increases, so do the number of parameters. As a result, it is gets much more difficult to calculate the posteriors. The discrete nature parameter space is composed of correspondences and the dimensionality increases over time as more and more correspondence variables are added.

Therefore, approximations are required in order to reduce the required computational memory. A particle filter approach cannot be used due to the high dimensionality of parameters for each particle.

There are two main SLAM algorithms, viz., Grid-based FastSLAM and GraphSLAM. The Grid-based FastSLAM extends the FastSLAM algorithm to create a grid map. A FastSLAM uses a particle filter named Rao-Blackwellized filter for trajectory estimation. This particle filter approach is computationally more efficient than the MCL algorithm. For map estimation, it uses low dimensional EKF where features of the map are modeled with local Gaussian. Thus FastSLAM provides computational advantages over normal SLAM algorithms.

The GraphSLAM algorithm is a graph based SLAM approach. It uses robot poses, measurements, motion constraints and measurement constraints. The motion constraints tie together two poses whereas the measurement constraints tie together a feature and a pose. The GraphSLAM can be broken down into two parts, viz., front-end and back-end. The front-end part focuses on constructing a graph using robot's odometry and sensory measurement data. The back-end focuses on optimization of the graph. It takes into consideration the motion and measurement constraints to find the optimum graph solution, which minimizes the overall error.

RTAB-Map package uses GraphSLAM with loop closure. When a robot enters into a new environment, the images of the environment are continuously compared with the past images to check for loop closures. As the time increases, the number of images to be compared with increases. To tackle with this problem, GraphSLAM uses multiple techniques such as bag-of-words and memory management. This allows RTAB-map to run for long time in large environments.

Scene and Robot Configuration:

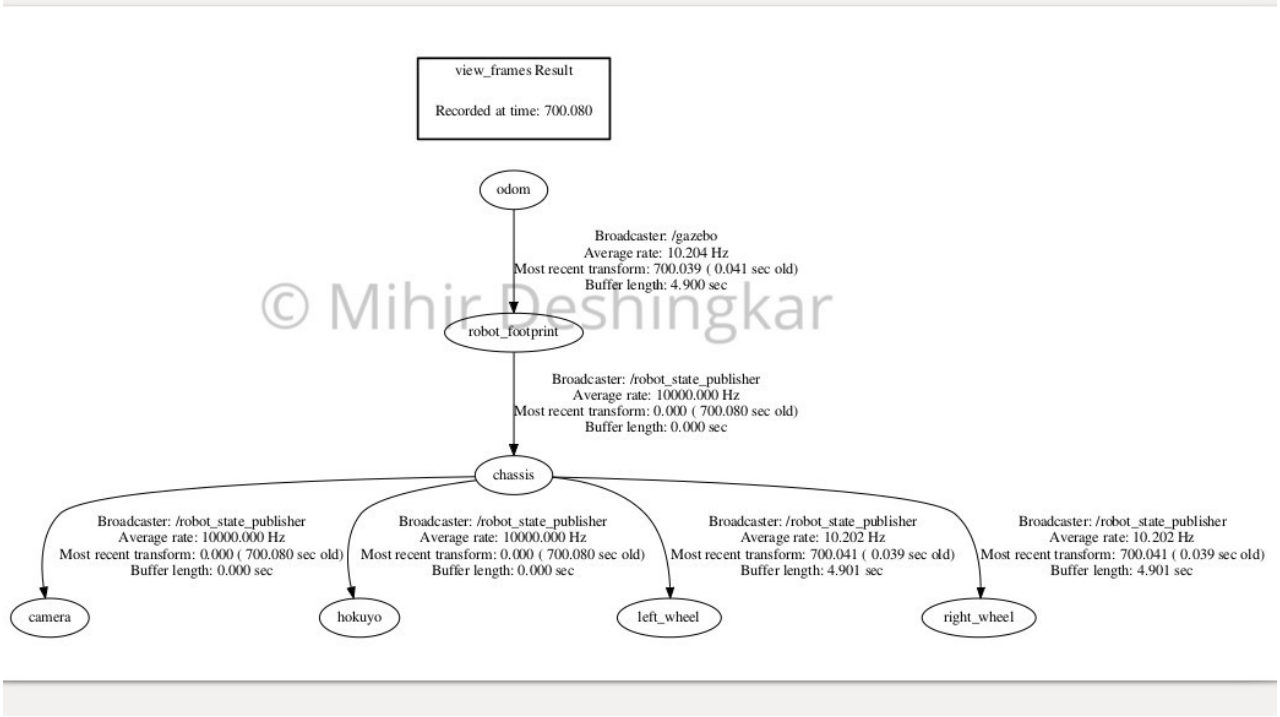
The newly created gazebo environment has a 4 walled room in which various items are place. The 4 wall room was created using a custom model. This custom model was created using 'Building Editor' in Gazebo. Below, the list of items present in this world is provided -

1. Book shelf
2. Cafe table
3. drc practice orange jersey
4. drc practice white jersey
5. Fountain
6. Number2
7. Car weel
8. Beer
9. Table

The Udacity bot was used with one main modification. The RGB camera in Udacity bot Gazebo model was replaced with Kinect, which is an RGBD camera. It is a two wheeled differential drive robot with two (front and back) caster wheels. The robot has Hokuyo laser sensor. The robot tf tree is given on the next page.

The 'urdf' folder in slam_project package contains the URDF files for this robot. The 'meshes' folder contains the .dae file for Hokuyo laser sensor. The topic '/udacity_bot/laser/scan' was remapped to '/scan' in the 'world.launch' file. All the launch files are stored under 'launch' folder. The scripts folder consists of teleop.py code which is used to drive the robot using keyboard keys. The config file consists of the saved 'rviz' configuration, which subscribes to the required topics.

The ‘worlds’ folder consists of the world files of the provided ‘kitchen_dining’ world and the created ‘my_home1.world’ world file. All these folders follow the ROS file structure hierarchy.



Udacity bot TF tree

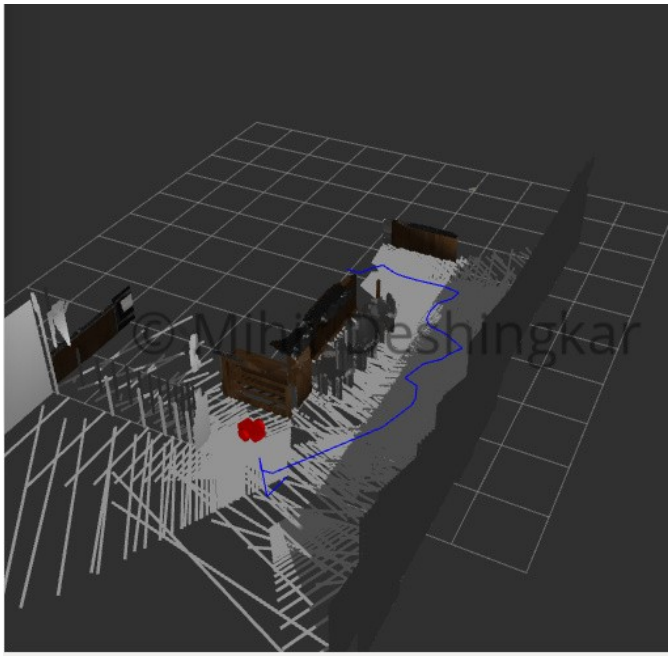
Results:

A good map of both environments was created. Below are the final results.

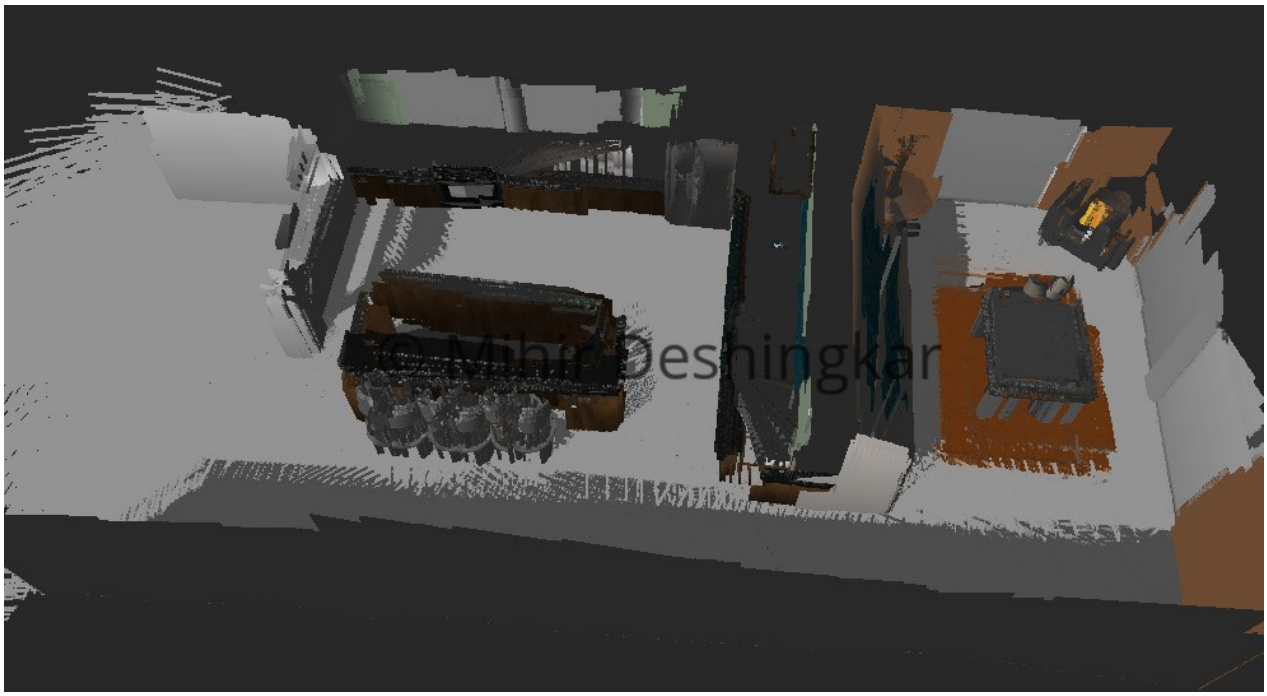
Mapping result of provided map of kitchen and dining room -

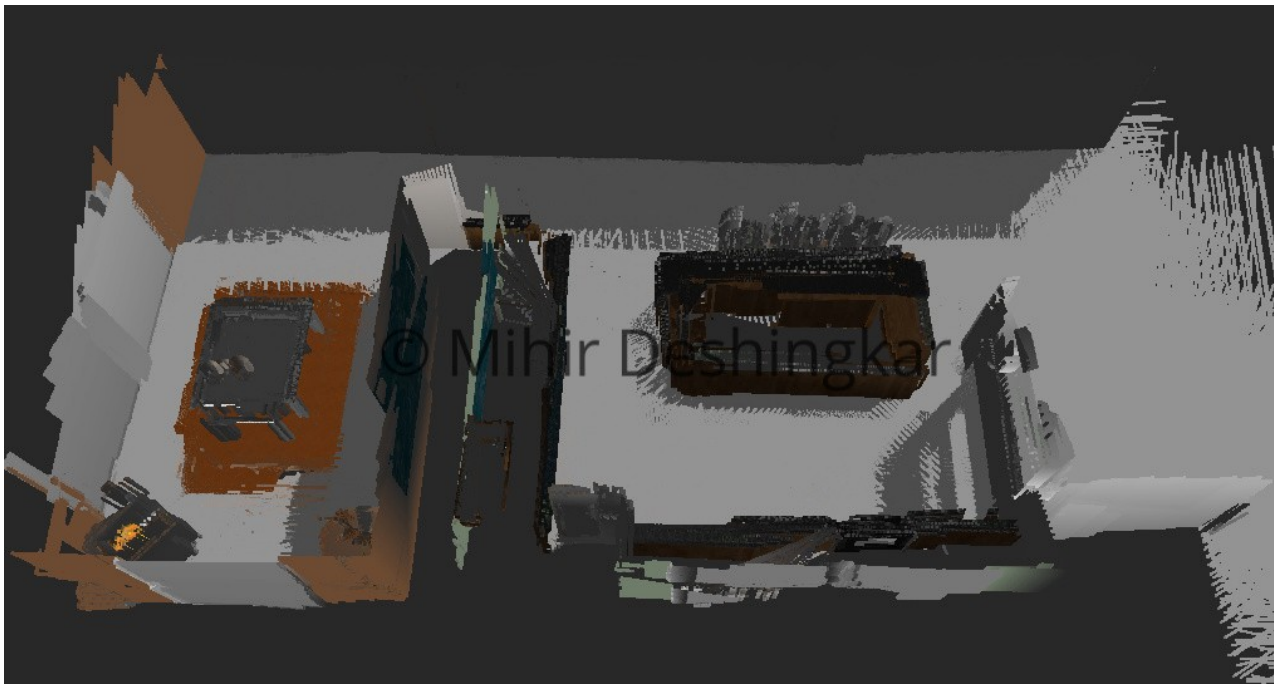
Below is the 3D map generated using the rtabmap-databaseViewer tool.

In process -



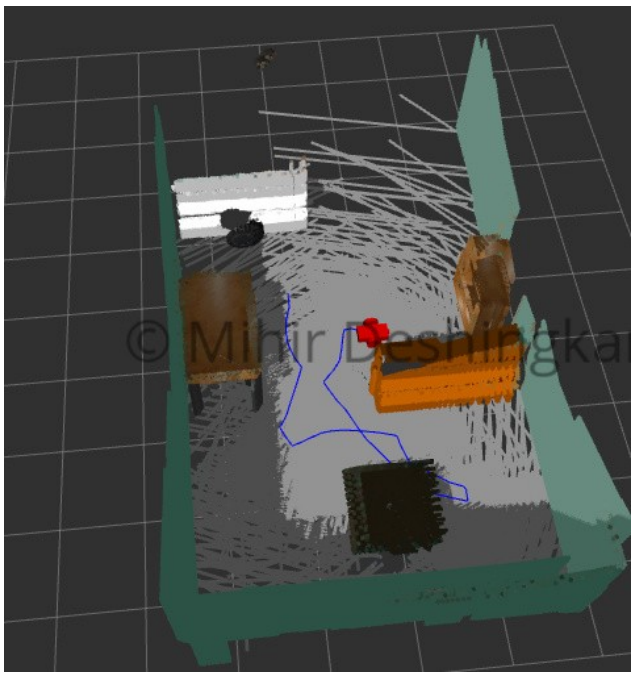
Completed map -

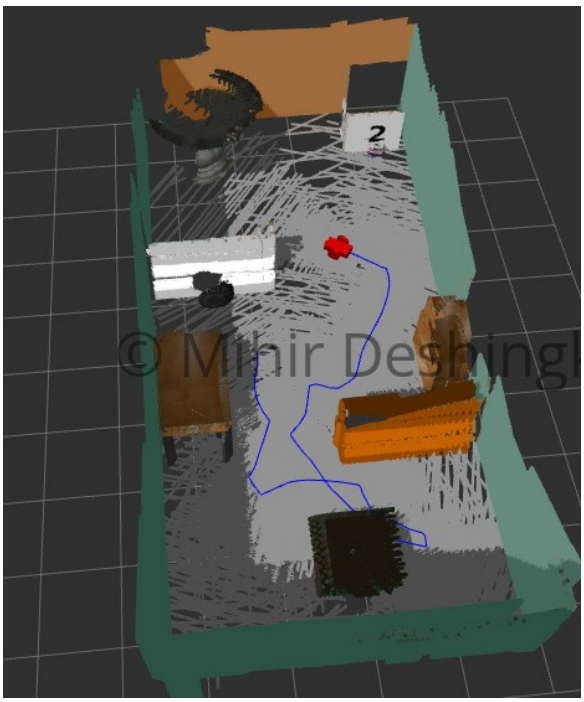




The mapping result of newly created map -

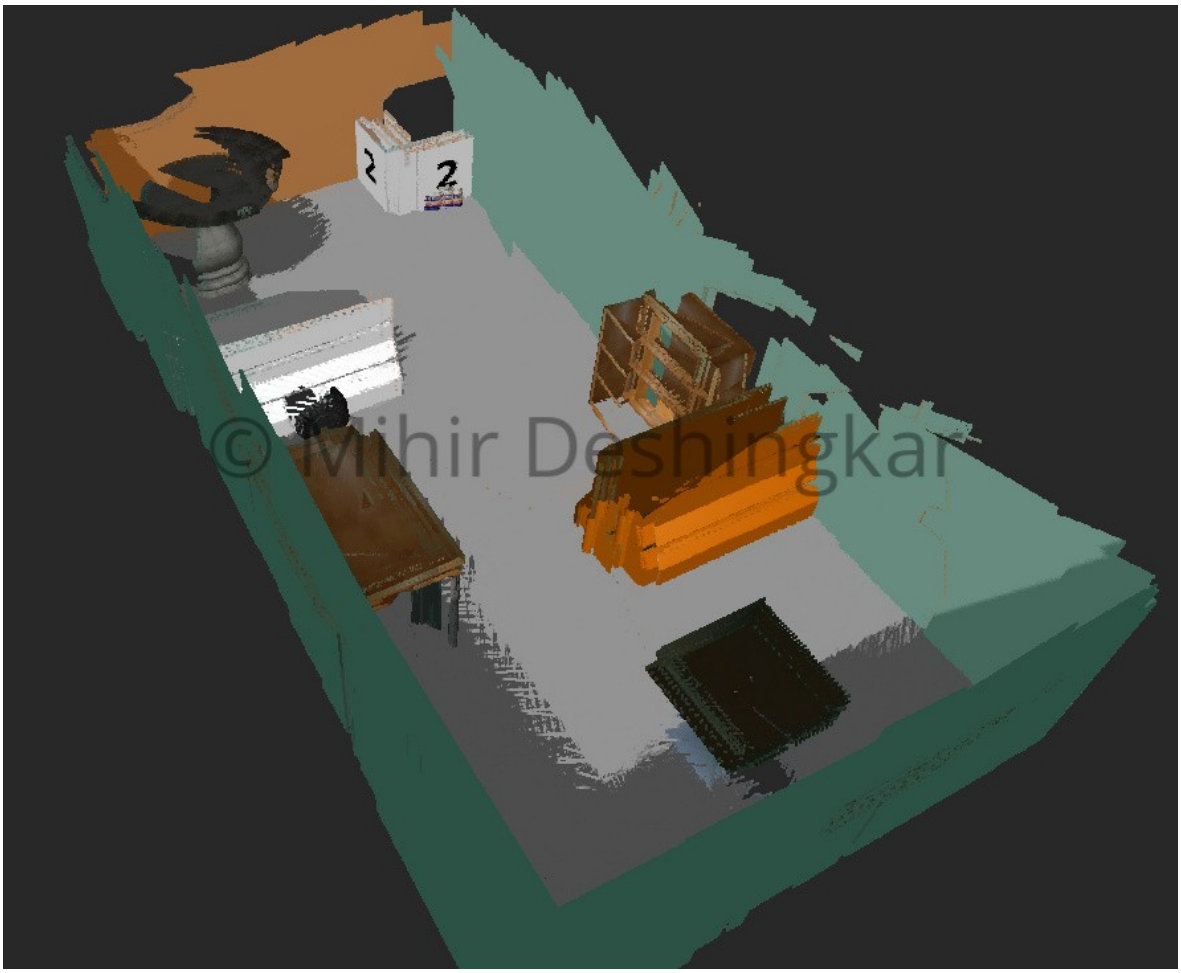
In process -



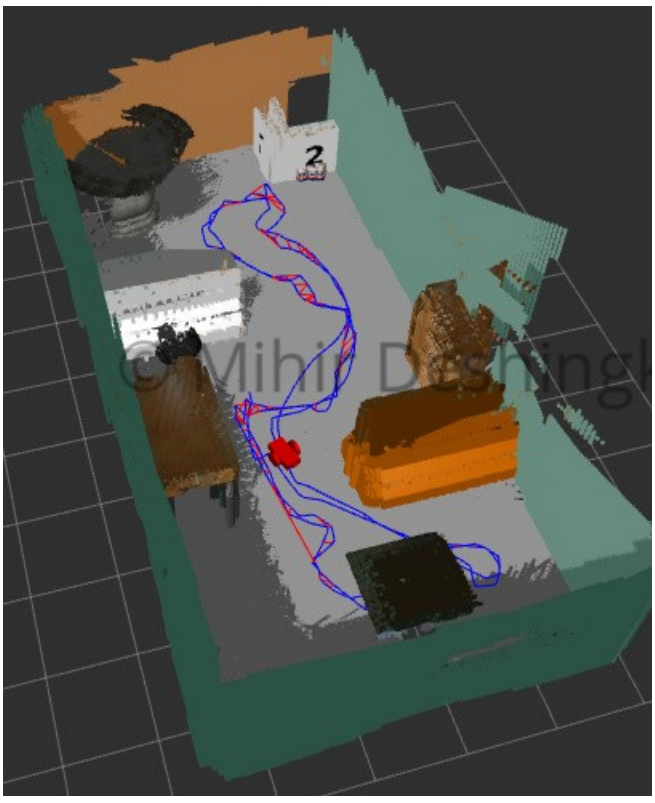


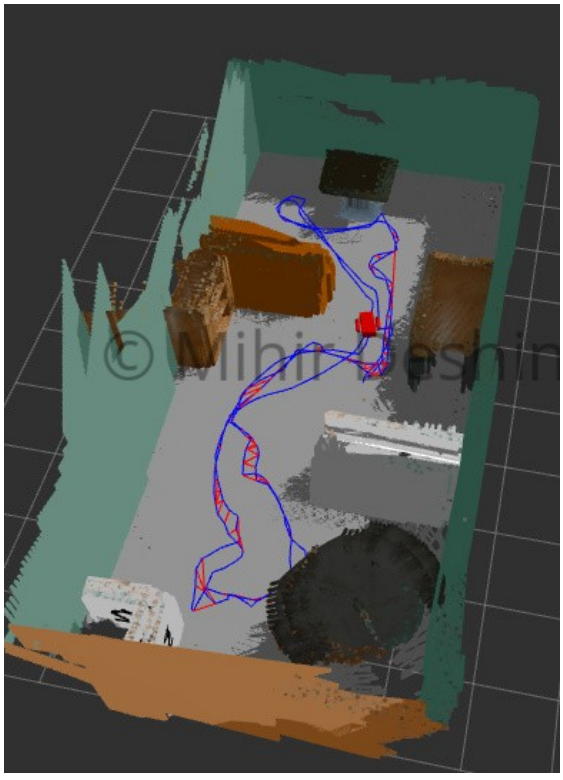
2D map -

Completed map -



From Rviz -





2D map -

Discussion:

The main footprint of both maps looks sufficiently accurate. For the provided map of kitchen and dining room, the map is quite accurate and important features are captured. Total of 128 global loop closures were observed. However, there are some features in the map which are mapped somewhat incorrectly. For example, all the table and chair legs have multiple representations nearby.

The map of the newly created world has about 109 global loop closures. This map also includes all the important features in the room. However, this map also has incorrect representation of table legs. Moreover, the quality of the map is a little less compared to the previous map. For example, the book shelf is not mapped very accurately and has multiple representations.

The quality of both maps can be further increased by adding new features. Also, once can use a custom AMCL package for localization, which can improve the map quality. Integrating IMU sensor for better odometry data can also improve the map quality.

Future Work:

The ‘RTABMAP’ is a great tool in ROS for performing SLAM and mapping the unkown environment. It provides many parameters that can be tweaked by the user. Some of these are the map update rate, loop closure method (such as SIRF, SIFT etc.), loop closure strategy (from RGB image or laser scan), etc. Properly tuning these parameters can in fact increase the quality and accuracy of the map. These parameters will be very important when mapping an outdoor environment. For example, if the laser data is noisy due to interference, one can use RGB loop clusure strategy instead of laser scan. Map quality and localization can be improved by integrating with IMU sensor for better accuracy and AMCL package for localization.

This tool can also be leveraged to map a real enviroment. Once mapped, various path planning algorithms can be used to autonomously navigate the robot in the map. For example, one can create

a home cleaning robot. The robot can use RTABMap to map the house. Using this map, the robot can autonomously clean the entire house.

Reference Links:

http://official-rtab-map-forum.67519.x6.nabble.com/rtabmap-ros-Could-not-get-transform-from-camera-link-to-zed-rgb-optical-frame-td828.html

http://wiki.ros.org/