Map My World

Lorenzo Bermillo

Abstract – In this paper, the topic of mapping and Simultaneous Localization and Mapping (SLAM) is discussed and applied to a previously created robot to successfully generate 2D and 3D maps of the supplied simulated environment using the RTAB-Map technique. This method is again performed to a newly created environment with the same robot and the resulting maps are compared to each other. Both environments are examined in terms of mapping and graded based on the how much of the simulated environment was mapped in both 2D and 3D and how accurate were these mapped environments. The robot's new sensor addition for this task is then described along with its new configuration and lastly, further enhancements in methodologies will be examined to increase mapping performance in terms of mapped percentage and accuracy.

1. Introduction

In the previous project, the robot performed localization with the map already provided to the robot in advance, however, this isn't always the case in most real-world applications. One can say that the blueprint of a building can be provided to the robot, but this poses a problem. First, most buildings do not comply with the blueprints making it difficult for the robot to navigate. Second, even if these blueprints are 100% accurate, it doesn't account for items in the building such as furniture. These impedes the robot's ability to navigate and localize itself through a course accurately. In most cases, the robot will have to navigate through dynamic environments, thus an autonomous robot must have the ability to accurately map its environment and localize itself at the same time.

In this project, the robot is to map a given simulated environment and another one that is created in both 2D occupancy grid and 3D octomap using a SLAM algorithm known as RTAB-Map. The goal is to produce a great map of the environment with the least amount of passes possible and getting at least 3 loop closures.

2. Background

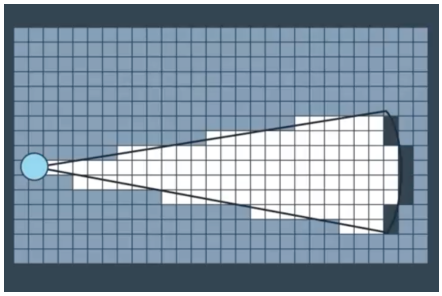
Acquiring maps is a challenging problem due to a number of reasons:

- **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.

2.1 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.

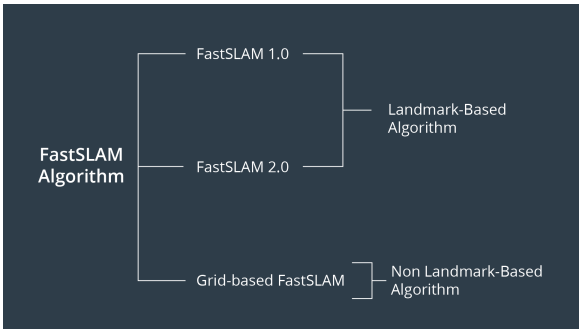


The main use of this algorithm, however, is for post-processing. Because SLAM techniques do not produce maps that are suitable for planning and navigation, occupancy grid mapping is often used to complement SLAM. This algorithm is used after solving the SLAM problem by other means and taking the resulting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

SLAM, or Simultaneous Localization and Mapping, is an algorithm that builds a map while concurrently localizing the robot relative to the map. This is more challenging to solve since neither the robot's pose nor the map is provided. The accuracy of the map depends on the accuracy of the localization and vice versa, thus SLAM is called the chicken or the egg problem. SLAM takes two forms, online SLAM and full SLAM. Online SLAM estimates the current pose of the robot and map using the present measurements and controls. On the other hand, full SLAM, or offline SLAM, estimates the robot's complete trajectory and map using all measurements and control. Both techniques solve for the full SLAM, however, only one of the two also solve for the online SLAM problem as well.

2.2 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.

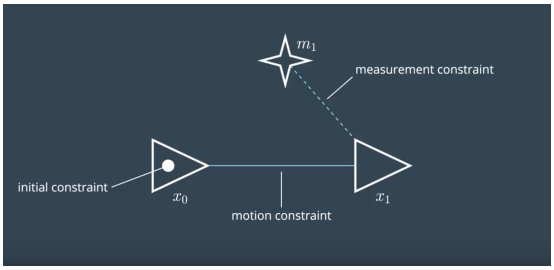


There are different types of FastSLAM, two of which are iterative upgrades of the FastSLAM algorithm, FastSLAM 1.0 and FastSLAM 2.0, and the other is an extension of the FastSLAM algorithm, Grid-based FastSLAM. The biggest disadvantage of FastSLAM is having to always assume that there are known landmarks, 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.

2.3 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.

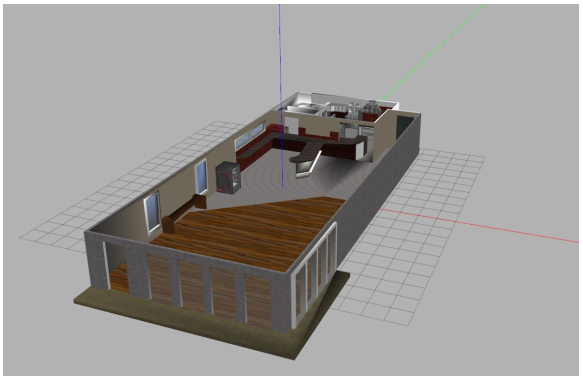


After, it recovers the map and the robot path by solving these constraints into a generally reliable estimate. Usually these constrains are non-linear, therefore, they are linearized in the process. This is keeps iterating until the optimal solution converges.

RTAB-Map, real-time appearance-based mapping, is a graph-based SLAM method that uses data gathered from vision sensors to localize the robot and map the environment in real time. This algorithm uses a concept called loop closure to establish if a robot has seen a location before. RTAB-Map is optimized for large-scale and long-term SLAM by using numerous strategies to allow loop closure to be done in real time.

3. Scene and robot configuration

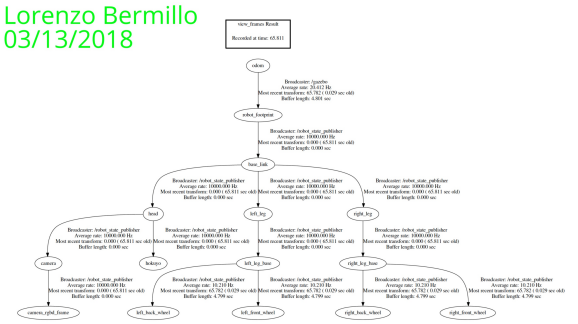
In the creation of a new world in gazebo, it was elected to use a predefined environment in gazebo, however, more objects were added into the world to create a feature rich environment. The predefined environment chosen was a café provided by gazebo.

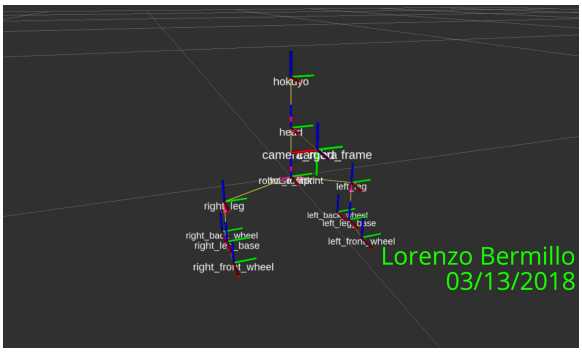


The café had a big space in the front where the wooden floor was located. So, to utilize this space, objects such as tables, a tree, and a gazebo were then added to create the desired environment resulting into the layout below.

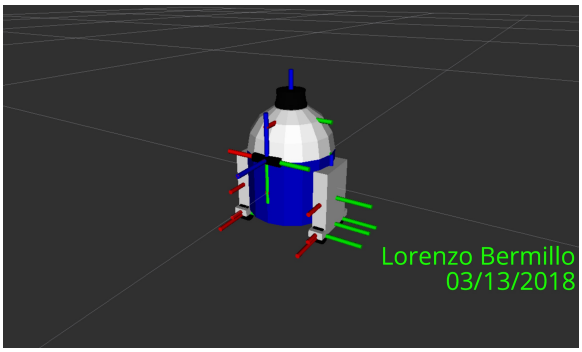
This will help the robot distinguish features and decide whether it has seen a feature before or not.

The robot used to perform was the same robot used in the previous project for localization, however, few adjustments had to be made to create a project ready robot. The main adjustment was the addition of an RGB-D sensor to also measure depth. This was a necessary camera upgrade for RTAB-Map. Then the camera object was lengthened and moved down to view more of the ground rather than just what's in front of it. Next a new frame was added to the robot's transform to compensate for the problem of RGB-D point clouds pointing up.



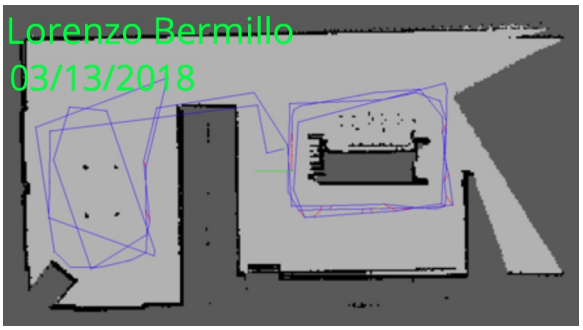


This is due to the depth cloud being generated along the z axis and therefore this needs to be rotated in order to visually correct. Adding this new faux frame connected to the RGB-D camera allows the rotation of the point clouds without having to modify the camera's orientation. This frame is rotated by -90° on both the x axis and z axis to correctly visualize the point clouds in the project.



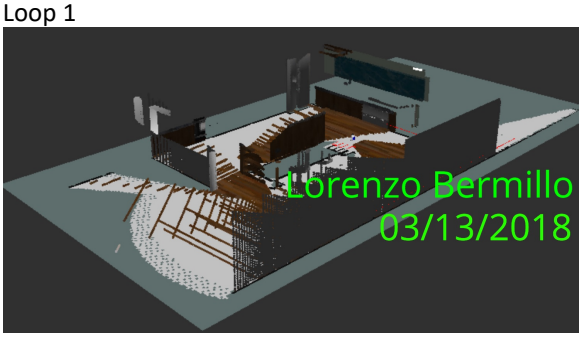
4. Results

To map the provided world, the decision was to loop around each room 3 times, first looping through the kitchen area then the dining area as shown by the 2D map below.



The speed of the robot when mapping in the linear x direction is 0.1 while the angular z direction is 0.5. Below, the first two loop progressions of the kitchen area is shown for each loop the robot makes.

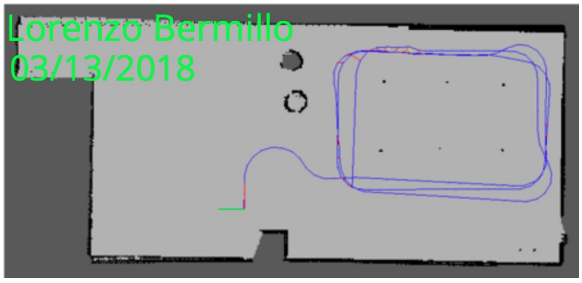
Loop 1



Loop 2

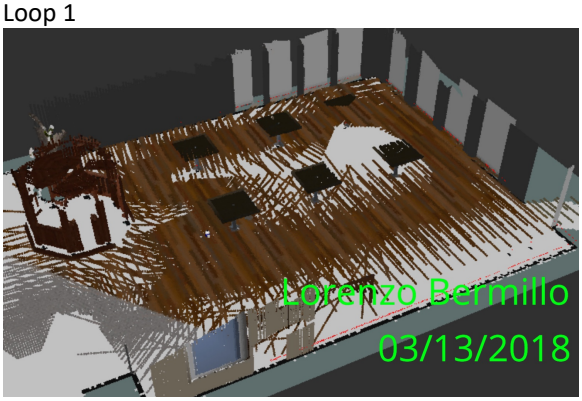
Final 3D map

For the created map, the decision was for the robot to loop around the tables in the front of the café only. Here, the robot's speed was 0.3 linearly and 0.5 angularly. A 2D map of the environment with the robot's trajectory is shown below



Like before, the progression of the map is shown below, only the front part of the café is mapped, for reasons explained in the next section.

Loop 1



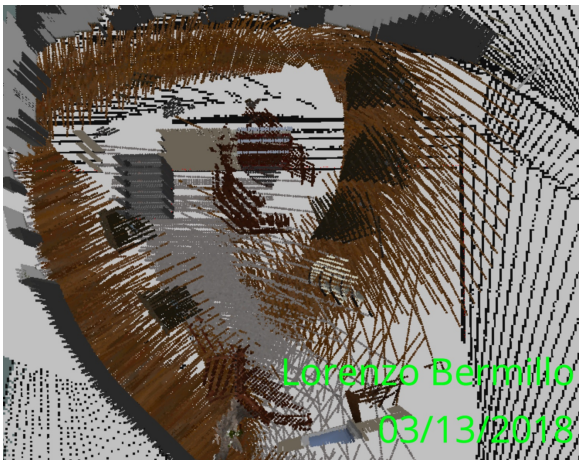
Loop 2

Final 3D map

5. Discussion

Overall mapping the environments for both was successful. To start, the 2D maps for both worlds were mapped very well. Mapping the worlds in 2D took at least one or two passes to complete them and was a fairly an easy task to do. The real challenge lied in mapping the 3D worlds.

The supplied kitchen and dining world was slow to map; however, this was found to be the best method because increasing the speed took more passes to create a recognizable 3D map. On the other hand, the created café world was much easier to map for it was able to map more in less passes, so the speed was increased. As seen in the first loops in the Result section, it can be observed that the robot was able to map more in the café compared to the kitchen. As a result, the café was faster to map. From the rtab-databaseViewer, the kitchen and dining world had a total global loop closures of 18 while the café world had 24 indicating that the café world had more features. However, although the café seems to be the better world, the reason as to why only the front of the café was mapped was due to when the robot maps the back of the café, the kitchen area, after mapping the front part of the café, the robot is unable to localize itself and distorts the map as seen below.



Even if the robot maps from the back of the café first, the end result is the same. It seems that RTAB-Map doesn't localize well when the environment changes.

Mapping the café can be improved once the café gets renovated and the same floor is used throughout the café and the kitchen area uses a similar theme as the front of the café. On the other hand, the kitchen and dining world didn't have any issues other than it maps slow. For this, adding more features will help the robot map the environment faster and help increase the number of global loop closures. The robot can also benefit from upgraded processing hardware to process the environments much quicker.

6. Future Work

Mapping and localization is very essential in robotics. It is used in many applications in robotics from vacuuming a room to self-driving cars. This technology is especially crucial for areas where humans can't enter due to dangerous situations such as natural disasters. Mapping can be applied in critical missions involving disasters where it is difficult or too dangerous for humans to traverse an affected area. Drones can construct 3D maps of the affected areas to provide first responders information of victim locations and also gives them a chance to plan how to approach the area. This same concept was used after Hurricane Harvey in Texas.

This technology can also be used in a fun way. In a future project, this tool will be very handy for the creation of objective games such as capture the flag, that involves physical robots in opposing teams placed into an unfamiliar environment. These robots will have to map the environment where the map information will be distributed amongst team members and they must work as a team to complete the objective and win the game. This is similar to modern console games played today but in a physical form using robots.

Works Cited

Dieter, Fox, Sebastian Thrun and Wolfram Burgard. *Probabilistic Robotics*. 2005.
Udacity. *www.udacity.com*. n.d. 03 2018.