PROJECTMAP MY WORLD***

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 two environments - one supplied and the other student created. using the RTAB-Map technique. Students extend a previous robot creation to upgrade sensors to supply the necessary sensor messages for RTAB-Map. This leverages the laser scanner, IMU/Wheel Encoder but replaces the camera with a RGB-D camera. The ROS project is created with all links connected with appropriate naming and mapping. The robot is launched and teleoped around the room to generate a map of the environment. After successfully mapping the supplied environment, a student defined environment is created and mapped using the same technique.

---

1. Introdution

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. In most cases, the robot will have to navigate through dynamic environments, so 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.

To tackle the problems in their own way, we can use different algorithms. In this paper, the algorithms that will be covered are Occupancy Grid Mapping, Grid-based FastSLAM, Graph-SLAM and RTAB-Map.

a. 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. 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.

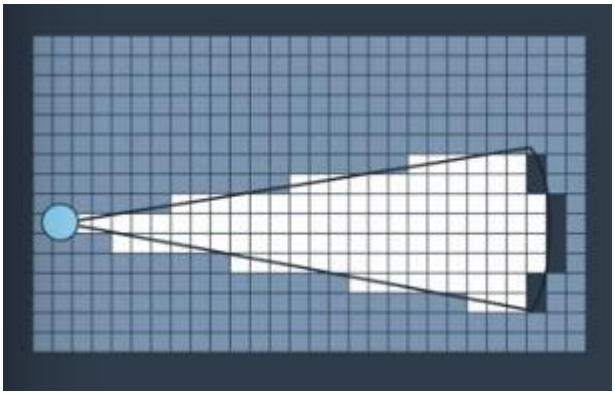

Figure 1. Occupancy Grid Mapping

b. 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.

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.

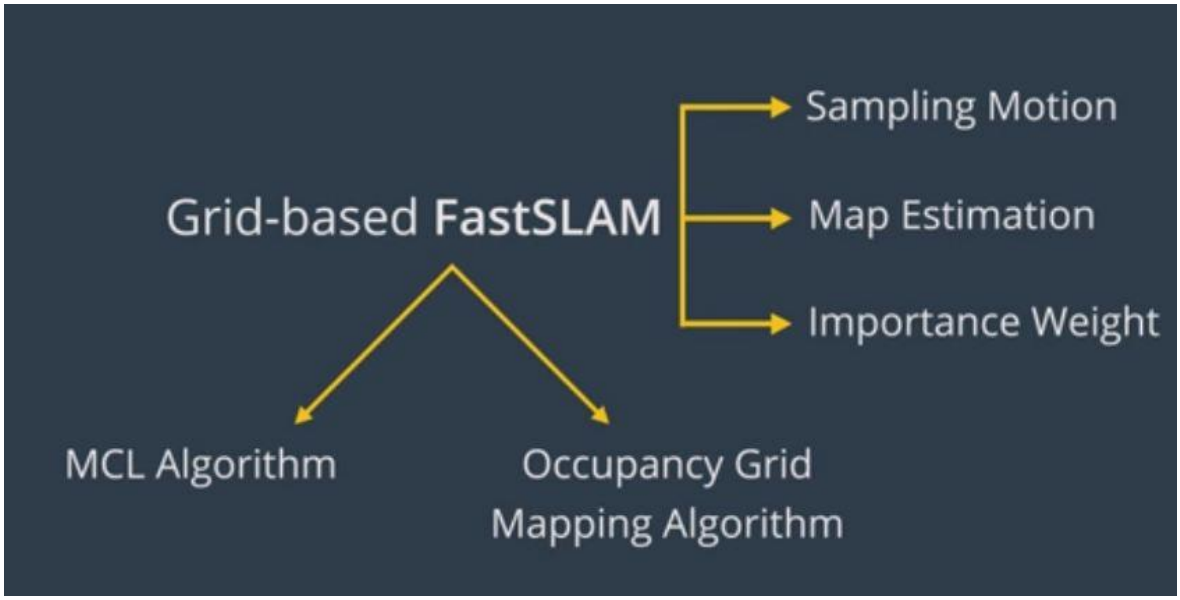

Figure 2. Grid-based FastSLAM

c. Graph-SLAM

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.

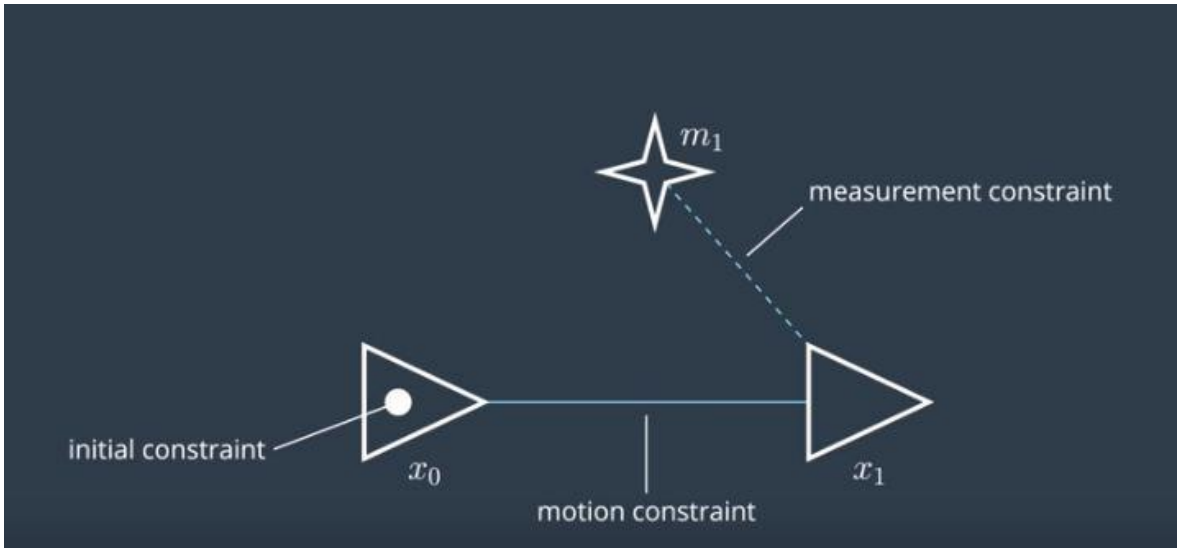

Figure 3. Graph SLAM

d. RTAB-Map

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.

The loop closure detection occurs against working memory to constrain the number of images interrogated. Working memory can be transferred and retrieved from long term memory to reduce complexity. The algorithm used for loop closure detection is SURF (Speeded Up Robust Features). The possible outputs of RTAB-Map are 2D occupancy grid map, 3D octomap or a 3D point cloud.

Robots are of varying dimensions inclusive of height. Whilst mapping a 2D environment may show where fixed walls etc are it does not take into account height. A robot, that is propelled on the floor, may be able to navigate under some obstacles but not others eg a chair vs a large table. Hence the need to understand the environment from a 3D perspective. However building a 3D map is more costly then a 2D map. This is not only in terms of Compute & Data costs but also in the

cost of the sensors required. However, simple sensors such as a single camera may be cheaper but the algorithms required can be more complex.

3. Robot Model Configuration

The robot model used was based on the my_bot created in the previous project. But this project, I didn't use gripper for my_bot to my_bot can go easily. And the camera was removed and replaced with a kinect leveraging the openni_camera ros package with the gazebo controller Openni Kinect. No changes were made to the hokuyo laser range finder.

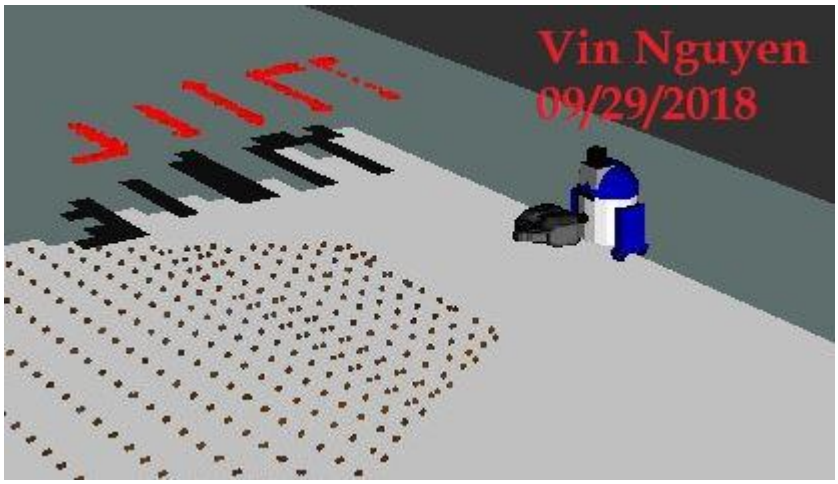

Figure 4. My_bot for previous project

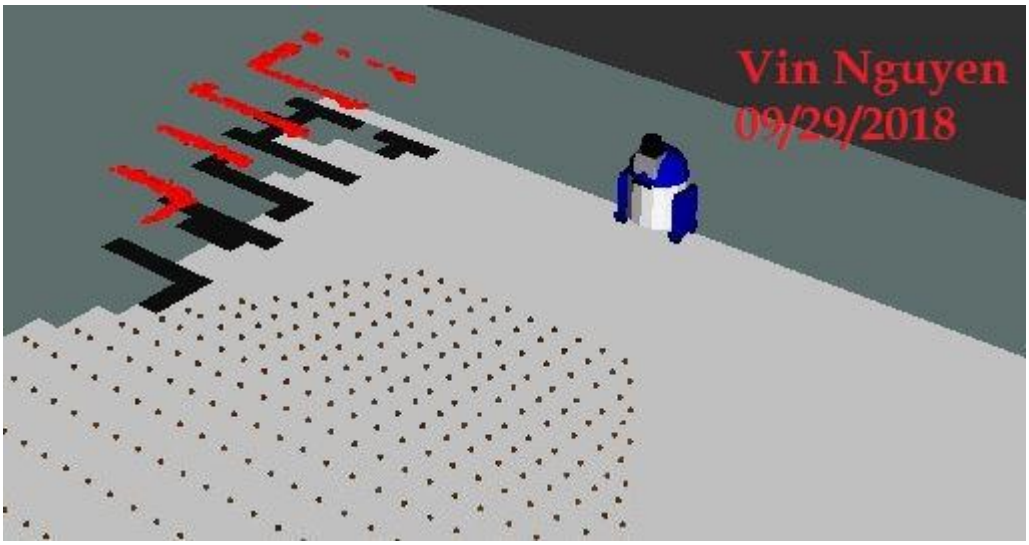

Figure 5. My_bot for this project

The my_bot configuration files can be found under the urdf directory. Visualization of the frames follows:

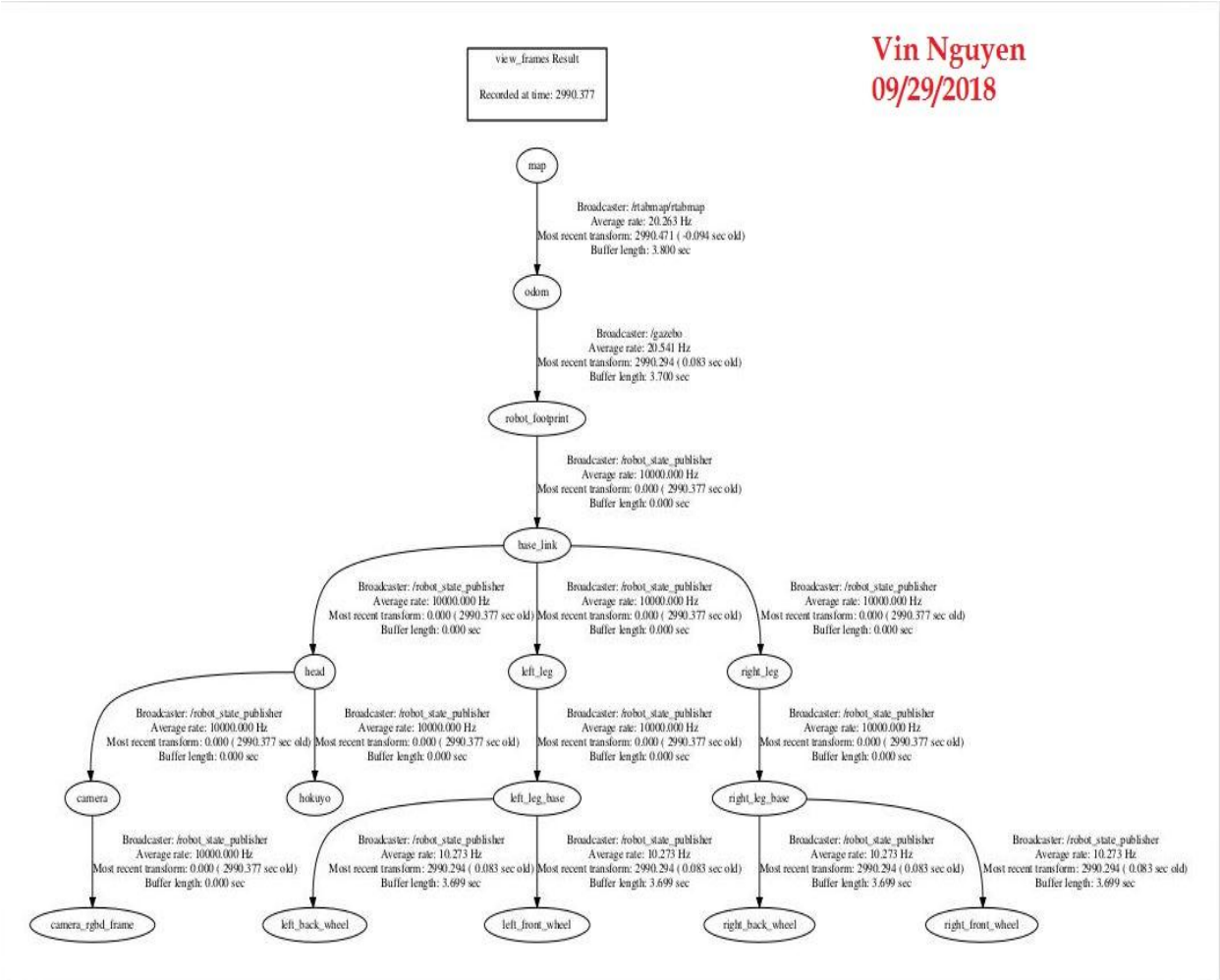

Figure 6. Visualized frames of my_bot

4. World Creation

Two worlds were created in gazebo - one supplied as kitchen_dining.world:

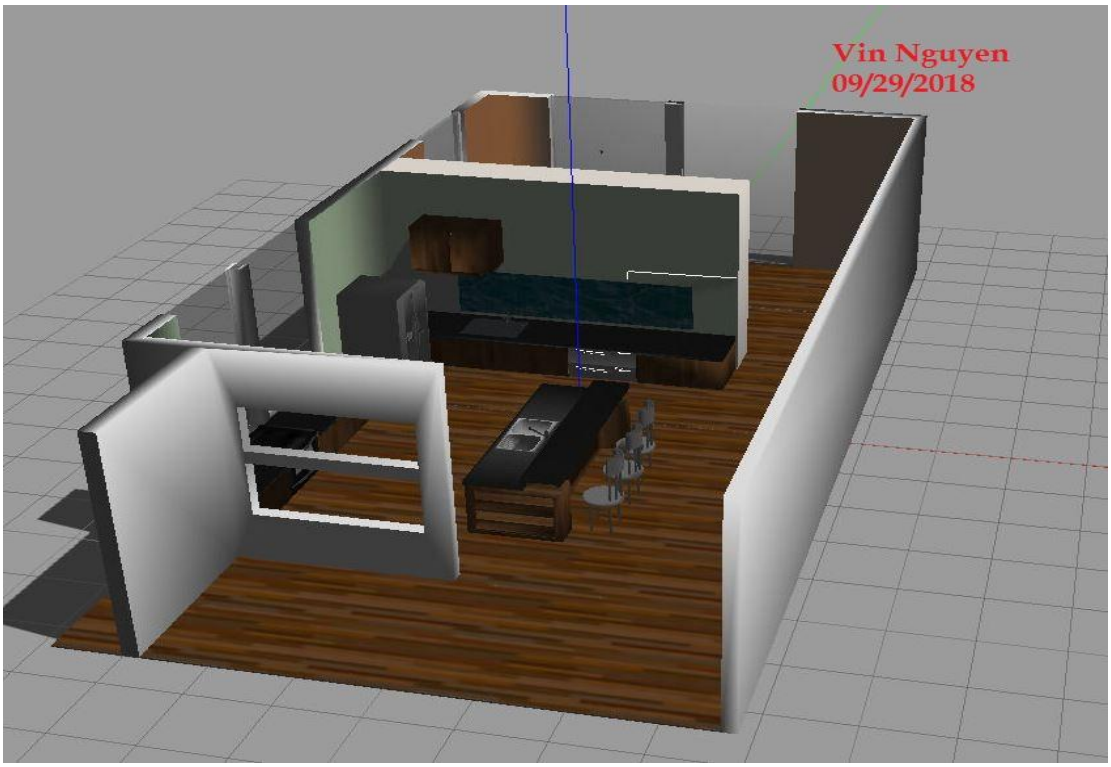

Figure 7. Kitchen_dining.world

And the other student customised cafe_world.world:

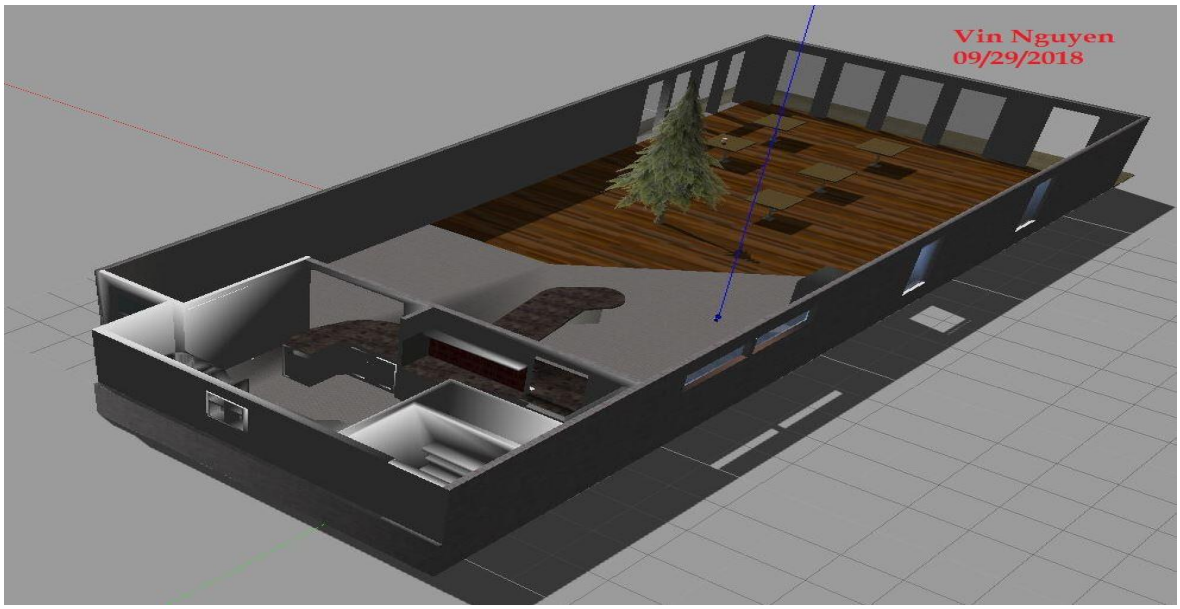

Figure 8. Cafe_world.world

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 cafe provided by gazebo. The cafe had a big space in the front where the wooden floor was located. So, to utilize this space, objects such as tables, a tree, beer,.. This will help the robot distinguish features and decide whether it has seen a feature.

5. Results

a. Provided world

To map the provided world, the decision was to loop around each room 3 times, first looping through the dining area then the kitchen area:

Figure 9. Map 2D the provided world

The speed of the robot when mapping in the linear x direction is 0.14 while the angular z direction is 1. Below, the 3 loop progressions of the kitchen area is shown for each loop the robot makes.

Figure 10. Loop 1 times

Figure 11. Loop 2 times

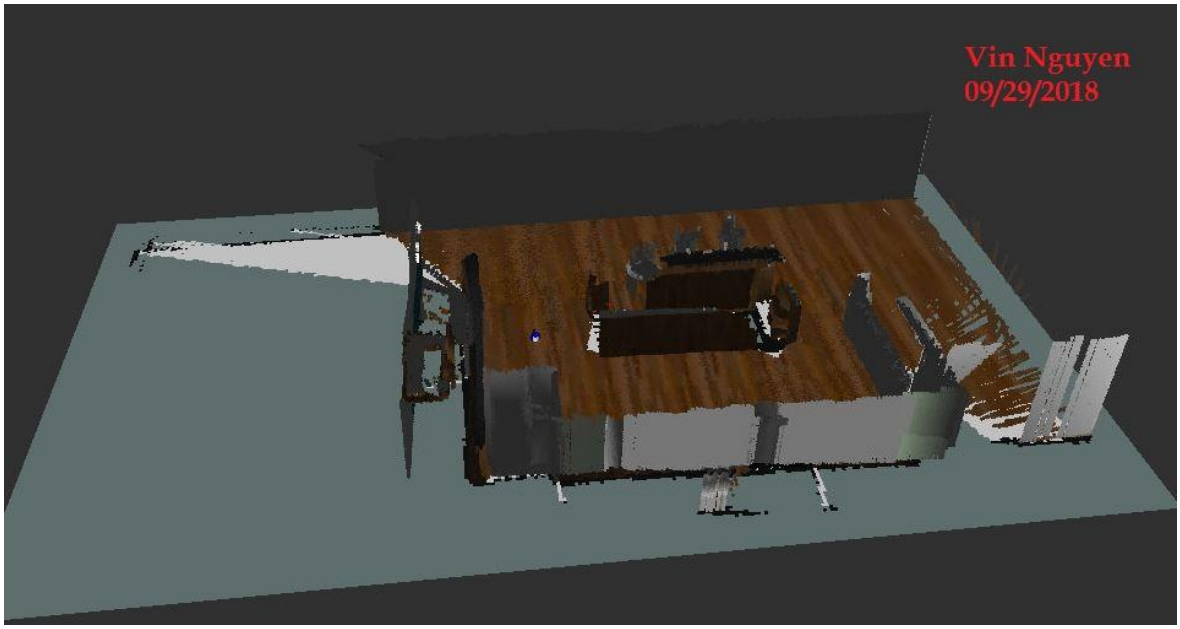

Figure 12. Loop 3 times

Figure 13. Final 3D kitchen world

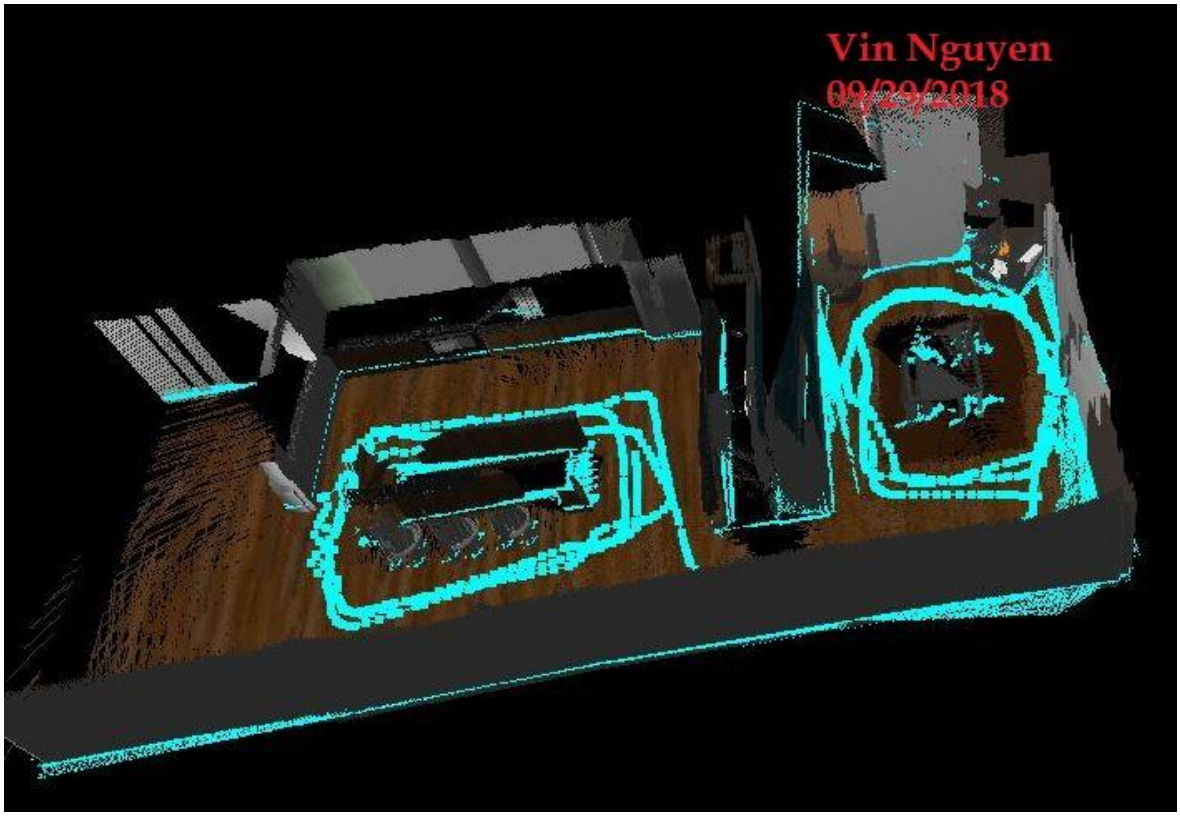

Figure 14. Final 3D kitchen world

b. Created world - cafe world

For the created map, the decision was for the robot to loop around the tables in the front of the cafe only. Because when the robot maps the back of the cafe, after mapping the front part of the cafe, both the kitchen area, after mapping the dining erea, the robot is unable to localize itself and distorts, the map as seen below.

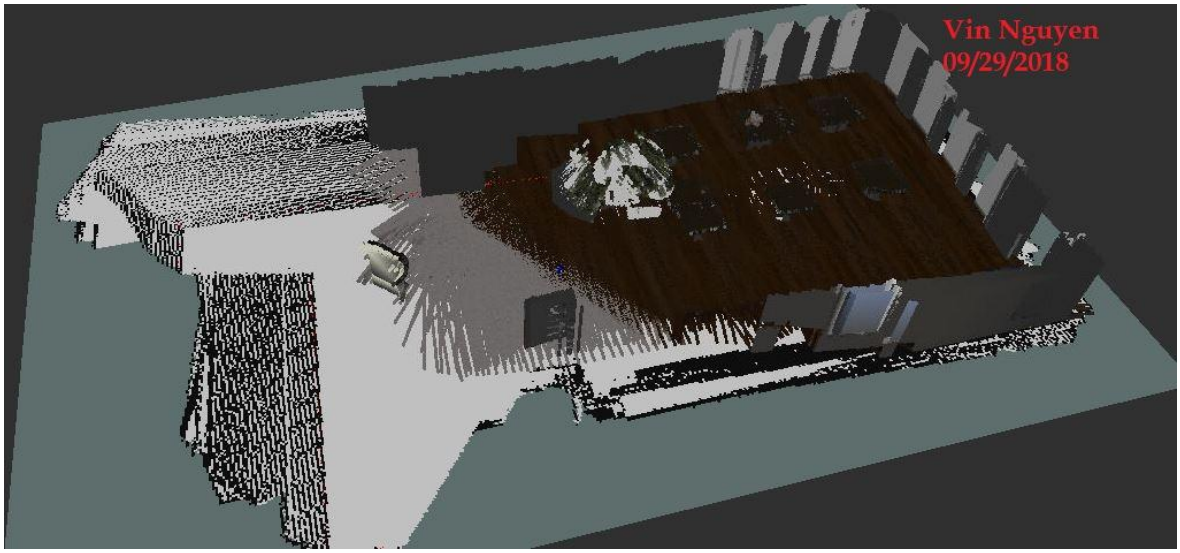

Figure 15. Final 3D cafe world

Here, the robot's speed was 0.2 linearly and 1 angularly. A 2D map of the environment with the robot's trajectory is shown below:

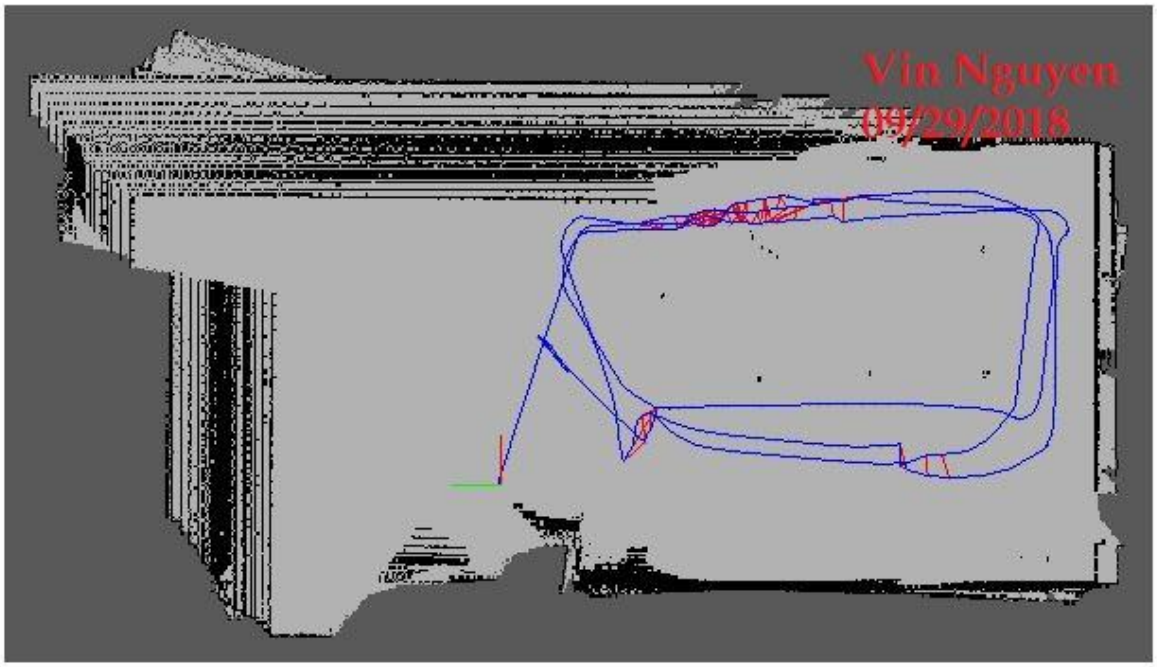

Figure 16. Final 2D dining world

Below, the 3 loop progressions of the cafe area is shown for each loop the robot makes.

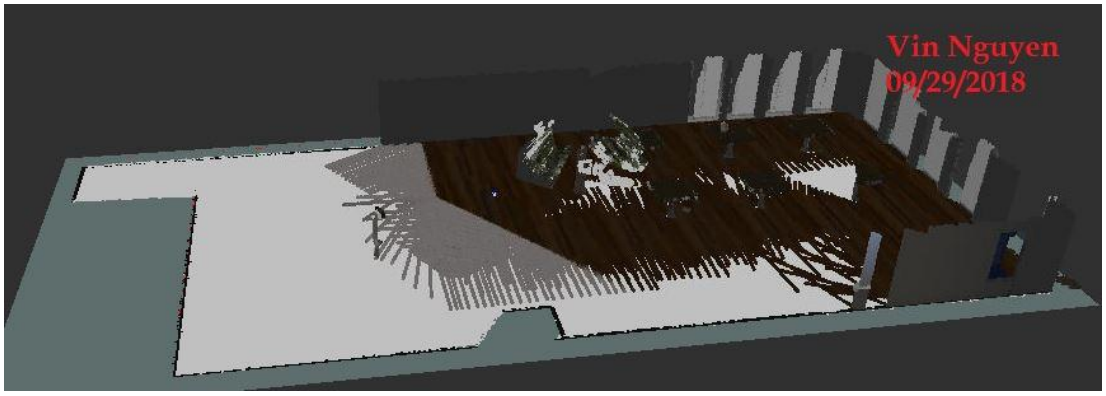

Figure 17. Loop 1 times of cafe world

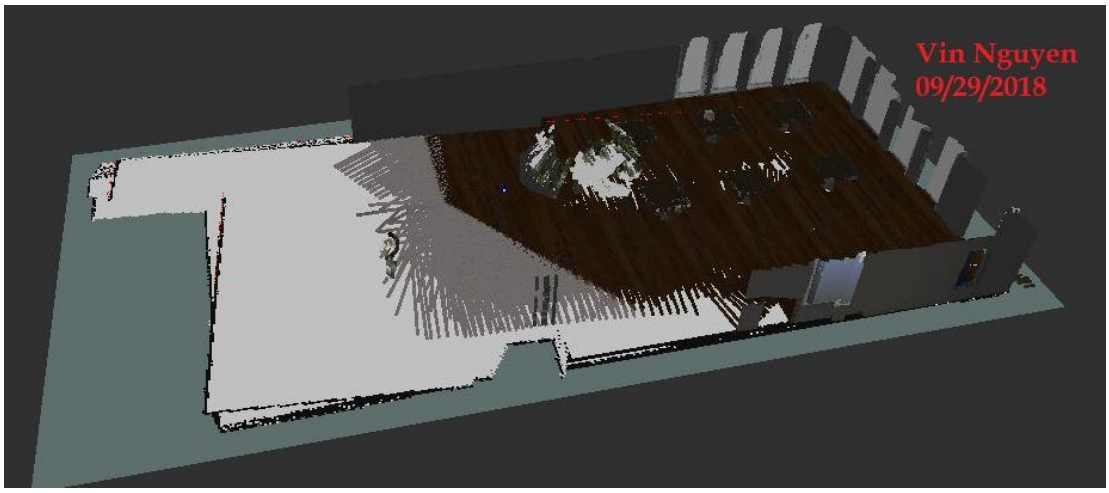

Figure 18. Loop 2 times of cafe world

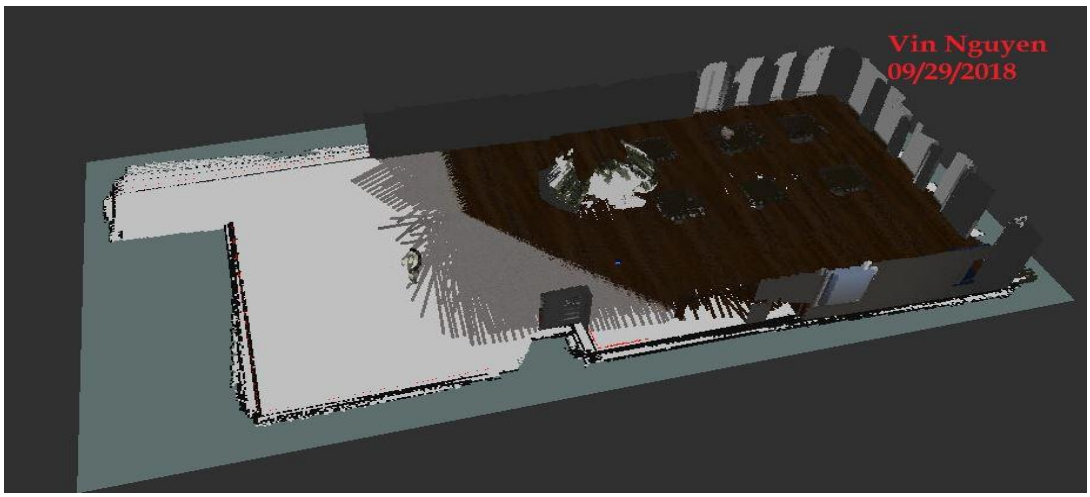

Figure 19. Loop 3 times of cafe world

6. Discussion

Overall mapping the environments for both was successful. However, the supplied kitchen and dining world was slow to map. The created cafe world was much easier to map for it was able to map more in less passes, so the speed was increased.

For the created world, the decision was for the robot to loop around the tables in the front of the cafe only. Because when the robot maps the back of the cafe, after mapping the front part of the cafe, both the kitchen area, after mapping the dining erea, the robot is unable to localize itself and distorts or not good. I think, seems that RTAB-Map doesn't localize well when the environment changes. Mapping the cafe can be improved , the same floor is used throughout the cafe.

As seen in the first loops in the result section, it can be observed that the robot was able to map more in the cafe compared to the kitchen. I think, because, cafe world had more features. 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.

7. Future work

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.

Mapping is important to help understand the world. There are a plethora of sensors and of interest is the about to arrive solid state lidars. As the price pointof these sensors continues to drop it will open up opportunities to create richerand more realistic 3D maps at a cheaper price point.

Being able to map an environment cost effectively to create a replicated virtual world will increasingly be important to allow for the training of deep learning model.