Simultaneous Localization and Mapping using RTAB-Map

Shweta Ruparel

Abstract—What if a robot has to uncover the information of an unknown place like a different planet, or a home environment where the setting of the furniture changes and the prior map given to the robot is no more valid.
Since the navigation of the robot in an environment involves knowing its own location and those of its goals,this problem is more commonly known as **Simultaneous Localization and Mapping (SLAM)**, a problem solved by computing the robot pose and a map of its environment of operation at the same time.
This paper presents the techniques and comparison of different slam algorithms used in creating 2D occupancy grid and 3D octomaps from the provided simulated environment and newly created simulation environment.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, localization ,Mapping , 2d occupancy grid, 3D octompap, SLAM using RTAB Map.

1 Introduction

The problem of generating a map inside of an unknown environment is called mapping.The problem of estimating both the map and the robot poses in real-world environment is called **Simultaneous Localization And Mapping or SLAM**.

In localization problems the robot poses are estimated in a mapped environment and in solving mapping problems exact robot poses are provided and the map of the environment is estimated.In SLAM both the techniques localization and mapping are combined in a way that the environment is mapped given the noisy measurement and the robot is simultaneously localized relative to its own map given the controls.

This makes it much more difficult problem than Localization or mapping since both the map and poses are unknown.With the noise in the robot's motions and measurements the map and robots pose will be uncertain and the errors in the robots pose estimates and the maps will be correlated.The

accuracy of the map depends on the accuracy of the localization and vice versa.

SLAM is often called the **chicken or the egg problem** because the map is needed for localization and the robots pose needed for mapping.

For robots to be useful to society they must be able to move in environments they have never seen before.There are various implementation of SLAM algorithms available for different needs. Refer figure 1 to know five categories of SLAM algorithms.

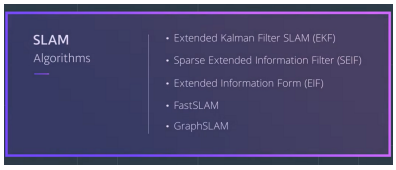

Figure 1. Various implementation of SLAM

With the ongoing work for self driving cars, Unmanned Aerial Vehicles , and autonomous underwater vehicles ,estimating the robot's pose (location, orientation) alongside mapping the environment is the most challenging task.

In this paper we will discuss *FastSLAM that uses particle filter approach alongwith the low dimensional extended Kalman Filter to solve the SLAM Problem* and then we will adapt it to the grid maps which will result in a *grid based fastSLAM algorithm.*We will then discuss *GraphSLAM that uses constraints to represent relationships between robot poses and the environment, and then tries to resolve all of these constraints to create the most likely map given the data* . This paper also presents the GraphSLAM implementation called **real time appearance based mapping or RTABMap**.

2 Background / Formulation

Lets understand what could be various mapping challenges and difficulties before we start digging into different SLAM Algorithms.

- *Challenge* - ***Unknown Map*** , When the map is unknown to us , we either have to assume known poses and estimate the map or assume unknown poses and estimate the map and poses relative to it.
- *Challenge* - ***Huge Hypothesis Space***, The hypothesis space is the space of all possible maps that can be formed during mapping.This space is highly dimensional since maps are defined over a continuous space.
- *Difficulty* - ***Size*** , Mapping large areas is hard, because of the large amount of data to be processed. The problem becomes even bigger when the size of the map is larger than the robots perceptual range.
- *Difficulty*- ***Noise***, Noise always exist in perception sensors, audiometry sensors and actuators.

With large noise mapping becomes more difficult and challenging as this noise adds up.

- *Difficulty- **Perceptual Ambiguity***,This ambiguity occurs when two places looks alike and the robot must correlate between these two places which the robot travel through at different points in time.
- *Difficulty- **Cycles***, Going in cycles, for e.g going back and forth in corridor, robots audiometry incrementally accumulate the error and at the end of the cycle the error is quite large.

There are several mapping algorithms viz occupancy grid mapping algorithm that decomposes the world into grid cells and implement a binary Bayes filtering to estimate the occupancy of each cell individually.Unfortunately ,this does not solve the problem of mapping in the real world because both the poses and the map is unknown.

2.0.1 Simultaneous Localization and Mapping

Simultaneous Localization And Mapping is the challenge of computing maps and robot poses relative to its map when both the poses and map is unknown.

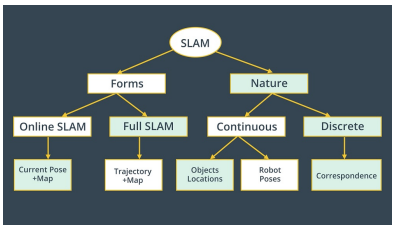

Figure 2. Nature and Form of SLAM

Refer figure 2 that describes the nature and the forms of the SLAM problem.

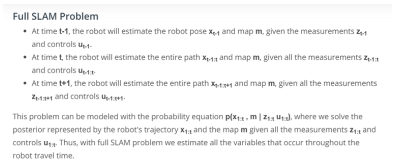

Figure 3. Full SLAM Problem

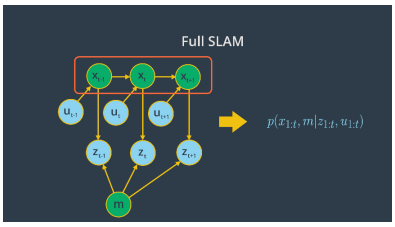

Figure 4. Full SLAM Problem

Refer Figure 3 and 4 for FULL SLAM Problem where robot estimates its entire trajectory and the map using all the measurements and controls.This problem can be modeled with the probability equation p(x1:t , m — z1:t u1:t), where we solve the posterior represented by the robot's trajectory x1:t and the map m given all the measurements z1:t and controls u1:t. Thus,

with full SLAM problem we estimate all the variables that occur throughout the robot travel time.

Computing the full posterior composed of the robot pose, the map and the correspondence under SLAM poses a big challenge in robotics mainly due to the continuous and discrete nature.

- The continuous parameter space composed of the robot poses and location of the objects is highly dimensional.While mapping the environment and localizing itself, the robot will encounter many objects and have to keep track of each one of them.Thus, the number of variables will increase with time, and this makes the problem highly dimensional and challenging to compute the posterior.
- The discrete parameter space is composed out of the correspondence values, and is also highly dimensional due to the large number of correspondence variables.Not only that, the correspondence values increase exponentially over time since the robot will keep sensing the environment and relating the newly detected objects to the previously detected ones. Even if you assume known correspondence values, the posterior over maps is still highly dimensional.

For SLAM problem, if we use the particle filter approach as used in MCL for accurate estimation of robots pose and thus localizing the mapped environment, as shown in the figure 5. In this approach,Each particle that carries the robots pose information including coordinates and theta, is added with one more variable map. This approach would fail because the map is modeled with many variables resulting in high dimensionality.

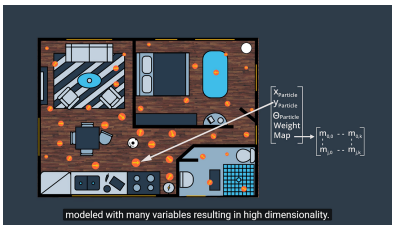

Figure 5. Full SLAM Problem

2.0.2 *FastSLAM*

The **FastSLAM algorithm** uses a custom particle filter approach to solve the Full SLAM problem with known correspondences.Using particles, fastSLAM estimates a posterior over the robot path along with the map.Each of these particles holds the trajectory which will give an advantage to SLAM to solve the problem of mapping with known poses.In addition to the robot trajectory each particle holds a map and each feature of the map is represented by a local Gaussian.With the fastSLAM algorithm the problem is now divided into separate independent problem as stated below-

- *Estimating the Trajectory:* FastSLAM estimates a posterior over the trajectory using a particle filter approach. This will give an advantage to SLAM to solve the problem of mapping with known poses.
- *Estimating the Map:* FastSLAM uses a low dimensional Extended Kalman Filter to solve independent features of the map which are modeled with local Gaussian.

The custom approach of representing the posterior with particle filter and Gaussian is known by the *Rao-Blackwellized particle filter* approach.

Three different instances of fastSLAM Algorithms are stated as below:

- *FastSLAM 1.0*
  The FastSLAM 1.0 algorithm is simple and easy to implement, but this algorithm is known to be inefficient since particle filters generate sample inefficiency.
- *FastSLAM 2.0*
  The FastSLAM 2.0 algorithm overcomes the inefficiency of FastSLAM 1.0 by imposing a different distribution, which results in a low number of particles. Both of the FastSLAM 1.0 and 2.0 algorithms use a low dimensional Extended Kalman filter to estimate the posterior over the map features.
- *Grid-based FastSLAM*
  The third instance of FastSLAM is really an extension to FastSLAM known as the grid-based FastSLAM algorithm, which adapts FastSLAM to grid maps.

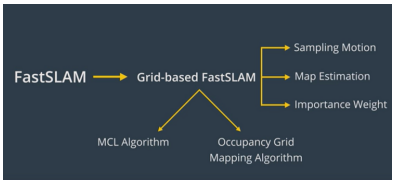

Figure 6. Grid Based Full SLAM Problem

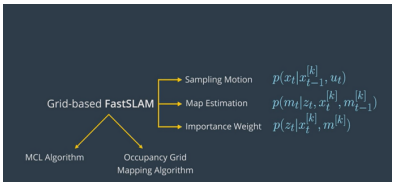

Figure 7. Grid Based Full SLAM Problem

Refer figure 6 and 7 that shows Grid Based FastSLAM implementation technique and its representation in probability equation.

The main advantage of the FastSLAM algorithm is that it uses a particle filter approach to solve the SLAM problem. Each particle will hold a guess of the robot trajectory, and by doing so, the SLAM problem is reduced to mapping with known poses. But, in fact, this algorithm presents a big disadvantage since it must always assume that there are known landmark

positions, and thus with FastSLAM we are not able to model an arbitrary environment.

With the grid mapping implementation technique as shown above, the environment is modelled using grid maps without predefining any landmark position. So by extending the FastSLAM algorithm to occupancy grid maps, the SLAM problem can be solved in an arbitrary environment.

2.0.3 Grid-based FastSLAM

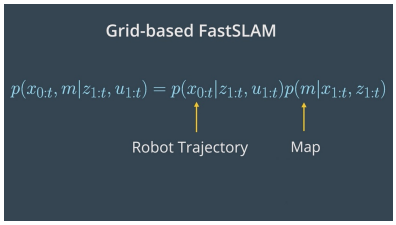

Figure 8. Grid Based Full SLAM Technique

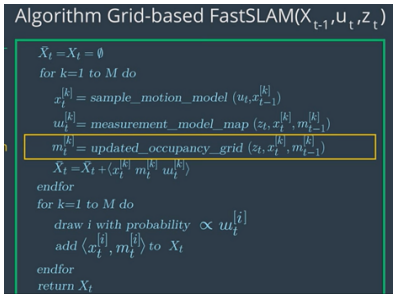

Figure 9. Grid Based Full SLAM Algorithm

Refer figure 8 and 9 for Grid based fastSLAM Algorithm representation. To adapt FastSLAM to grid mapping, we need three different techniques:

- Sampling Motion: Estimates the current pose given the k-th particle previous pose and the current controls.
- Map Estimation: Estimates the current map given the current measurements, the current k-th particle pose, and the previous k-th particle map
- Importance Weight: Estimates the current likelihood of the measurement given the current k-th particle pose and the current k-th particle map.

The grid-based fastSLAm algorithm consists of two main for loops. The first section includes the motion , sensor and map update steps and the second one includes the re-sampling process.At each iteraion , the algorithm takes the previous belief or pose , the activation command and the sensor measurements as input.Here are the steps that are performed

- Initially the hypothetical belief is obtained by randomly generating M particles.Each particle then begins by implementing the sampling technique in the sample motion model to estimate the current pose of the k particle.
- Next, in the measurement update step,each particle implements the important weight technique in the measurement model map function to estimate

the current likelihood of the k particle measurement.

- In the map update step , each particle will implement the map estimation technique into updated occupancy grid map function to estimate the current k particle map.
- The newly estimated k particle pose map and likelihood of the measurements are all added to the hypothetical belief.
- Next the re-sampling process happens through the re-sampling wheel. Here , particles with measurement values close to the robot's measurement values survive and are redrawn in the next iteration, while the others die.
- The surviving particles poses and map are added to the system belief.Finally the algorithm outputs the new belief and another cycle of iteration starts implementing the newly completed belief, the next motion and the new sensor measurements.

2.0.4 Disadvantage of fastSLAM

FastSLam and its instances uses custom particles to estimate the robots most likely pose.However at any point in time, its possible that there isn't a particle in the most likely location.Infact, the chances are slim to none especially in large environments.

To solve this challenge **Graph SLAM** is used.*This SLAM algorithm solves the full slam problem. This means that the algorithm recovers the entire path and map , instead of just recent pose and map. This difference allows it to consider dependencies between current and previous poses*. Since GraphSLAM solves the full SLAM problem this means that it can work with all the data at once to find the optimal solution.

2.0.5 GraphSLAM

Lets understand few notation used in representing GraphSLAM Algorithm

- **Poses** are represented with triangles.
- **Features** from the environment are represented with stars.
- **Motion constraints** tie together two poses, and are represented by a solid line.
- **Measurement constraints** tie together a feature and a pose, and are represented by a dashed line.

Refer figure 10 for the notation.As the robot move around more and more nodes are added to the graph and over time the graph constructed by the mobile robot become very large in size as shown in figure 11

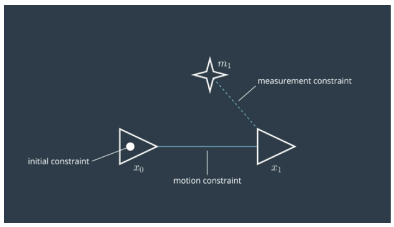

Figure 10. Graph SLAM Notation

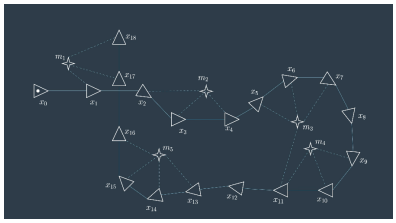

Figure 11. Graph SLAM notation for a navigating robot

Refer figure 12 for graph SLAM at glance.

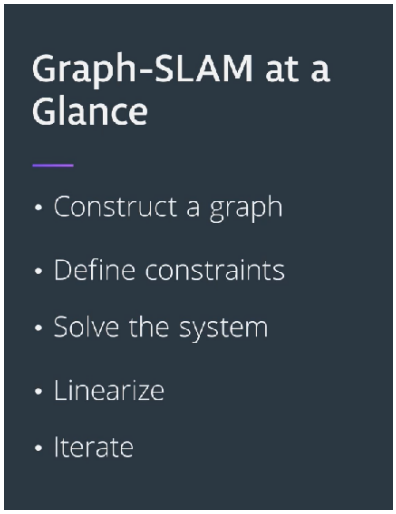

Figure 12. Graph SLAM at glance

The Goal of GraphSLAM is to create a graph of all robot poses and features encountered in the environment and find the most likely robot's path and the map of the environment. This task can be broken into two sections mentioned as below:

- *Front-End* - The front-end of the graphSlam looks at how to construct the graph using the odometry and sensory measurements collected by the robot.This includes interpreting the sensory data, creating the graph and continuing to add nodes and edges to it as the robot traverses the environment. The front-end also has the challenge of accurately identifying whether the features in the environment have been previously seen.
- *back-end* - The input to the back-end is the completed graph with all of the constraints. The output is the most probable configuration of robot poses and map features.The back-end is the optimization process that takes all of the constraints and find the system configuration that produces the smallest error.

Refer figure 13 and figure 14 for front-end and back-end tasks.

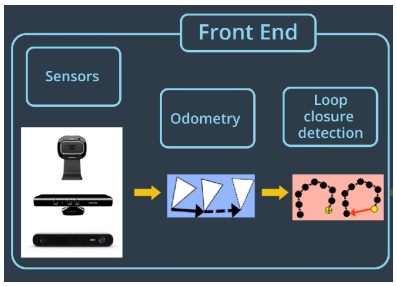

Figure 13. front-end

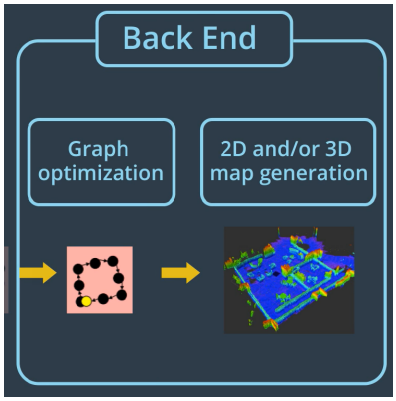

Figure 14. back-end

Let's consider a specific graph-based SLAM approach called RTAB-map.

2.0.6 RealTime Appearance-Based Mapping

Appearance-based SLAM means that the algorithm uses data collected from vision sensors to localize the robot and map the environment.In Appearance-based methods, a process called loop closure is used to determine whether the robot has seen a location before. 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.As a robot travels to new areas in its environment, the map is expanded,and the number of the images that each new image must be compared to increases, this causes the loop closure to take longer with the complexity increasing linearly. RTAB-Map is optimized for large scale and long term SLAM by using multiple strategies to allow for loop closure to be done in real time

RTAB-Map can be used alone with a handheld Kinect, a stereo camera or a 3D lidar for 6DoF mapping, or on a robot equipped with a laser rangefinder for 3DoF mapping.

3 Scene and Robot Configuration

The ros package slam_project is created and rtabmap is deployed to perform SLAM on two simulated environments.The first environment named Kitchen-dining was provided as a part

of project.The second project was created from scratch using Gazebo. A simulated environment for a Gas Station with people and vehicles was created.

3.1 Test Environment

Udacity workspace was used to test the robot in simulation.This online tool includes GPU access when needed as well as ROS, Gazebo, and other packages.While Gazebo is a physics simulator, RViz can visualize any type of sensor data being published over a ROS topic like camera images, point clouds, Lidar data, etc. This data can be a live stream coming directly from the sensor or pre-recorded data stored as a bag file.

RTAB-map a ros package for computing SLAM Challenges was deployed and tested to create a 2D occupancy grid and 3D octomap in the simulated environment like kitchen-dining room provided by Udacity and on a newly created environment for Gas Station with people and vehicle. The mapping was done by the moving the robot controlled by the keyboard using a teleop package. In order to be able to have more than 3 loop closure detection, which was the project's benchmark, the robot was required to navigate the whole environment at least 2-3 times to collect more images and map the environment accurately.

3.1.1 Kitchen-Dining Test environment

Refer figure 15 to check out the kitchen-dining environment provided for the project.

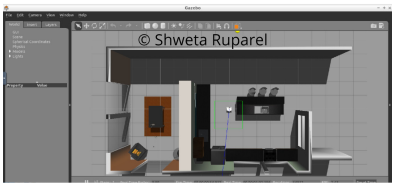

Figure 15. Kitchen Dining Test environment

3.1.2 Gas Station Test environment

Refer figure 16 to check out Gas Station environment created using Gazebo.

Figure 16. Gas Station Test environment

A new environment depicting a **Gas Station** was created using Gazebo by following the steps below:

- Open Gazebo
- Go to Insert - Model Database
- Browse through the available models and find model for Gas Station. Also look for SUV, a Standing person, a Walking person and a bus.
- Drag the selected objects into the environment and situate them.

- Export the model: File - Save World As, then navigate to the custom package, and then the 'models' directory, and save the file with a .world extension.
- When done, test the newly created environment by launching it as, gazebo GasStation.world

Different launch files are created to launch, the simulated environment(Gazebo World), teleop node and mapping node.Different debug tools like **tf** to view the links and connections of the robot used, **roswtf** to check if all running nodes are connected and communicating properly, or if any nodes have died,**rqt_image_view** and **rqt_graph** visualise camera images and robots connection respectively.

The **rtabmap-databaseViewer** is a great tool for exploring database when mapping is done generating . It is isolated from ROS and allows for complete analysis of mapping session.

Another tool **rtabmapviz**,for real time visualization of feature mapping, loop closures, and more.It is great to deploy on a real robot during live mapping to ensure that you are getting the necessary features to complete loop closures.

3.2 Robot Model

Slam bot is a small light weight cylindrical based robot created to perform simultaneous localization and mapping to create 2d occupancy grid and 3d mapping using RTAB-map SLAM.Slam bot is a small cylindrical based robot with two caster wheels , two driving wheels , a RGB-D camera and a Laser scanner. Refer figure 17 to check for the robot model called slam bot.

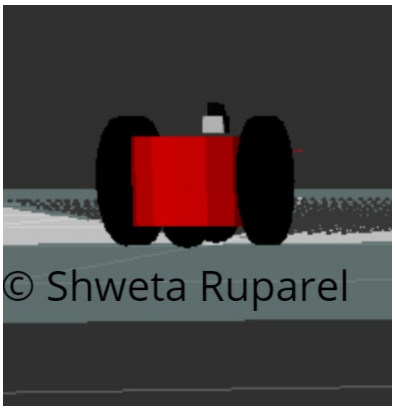

Figure 17. Slam Bot

4 RESULTS

The goal is to create a great map with the least amount of passes as possible. Getting 3 loop closures will be sufficient for mapping the entire environment. loop closures can be maximized by going over similar paths two or three times. This allows for the maximization of feature detection, facilitating faster loop closures!

4.1 Slam Bot mapping the Kitchen-Dining area

Refer figures to see the different results observed in gazebo ,Rviz, Rtabmap and rtabmap database viewer.

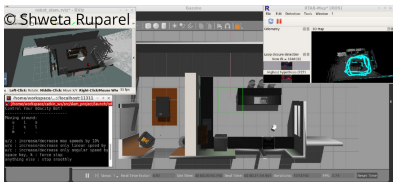

Figure 18. kitchen dining Test environment with teleop, rtabmap,rviz and gazebo windows

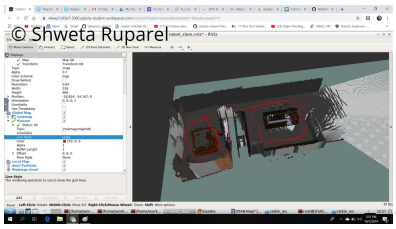

Figure 19. Kitchen Dining Test environment shows octomap

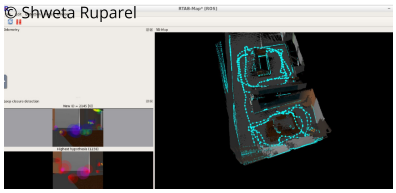

Figure 20. Kitchen Dining Test environment in rtabmap showing Loop closures and map

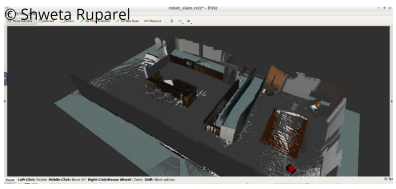

Figure 21. Kitchen Dining Test environment showing full mapped environment

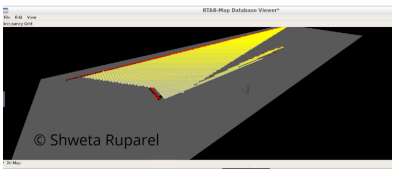

Figure 22. Kitchen Dining Test environment showing 2d occupancy grid

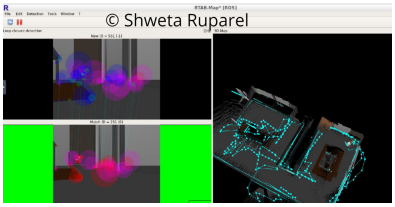

Figure 23. Kitchen Dining Test environment showing 3d map. Here we can see the table and chairs in the kitchen and other obstacles. In the other section, we can see a rug on the floor , a table and a couch and few other obstacles.

4.2 Slam bot mapping the Gas Station

Refer figures to see the different results observed in gazebo ,Rviz, Rtabmap and rtabmap database viewer.

Figure 24. Gas Station Test environment with teleop, rtabmap,rviz and gazebo windows

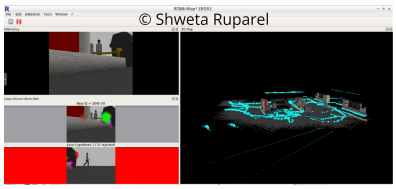

Figure 25. Gas Station Test environment in rtabmap showing Loop closures and map

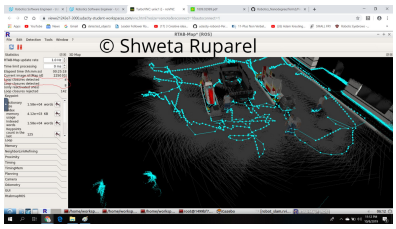

Figure 26. Gas Station Test environment showing full mapped environment

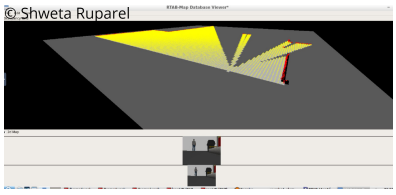

Figure 27. Gas Station Test environment showing 2d occupancy grid

Figure 28. Gas Station Test environment showing 3d map. Here we can see the Standing and walking persons an SUV, the trash cans and the Fuel pumping machine and other obstacles and walls. The walking person is mapped twice

5 Discussion

Both the robots were able to create the 3D map of the environment.Mapping the provided environment for kitchen-dining was more accurate than mapping the gas station. For more accurate mapping the gas station environment the tele-operation needs to be done more accurately by moving near to the obstacles to be able to get the features accurately and navigating the area multiple times. In gas station environment the bus was not captured completely. Also the loop closure detection produce false positives for mapping for a walking person. it is shown two times in the map, refer figure X

6 Conclusion / Future work

This paper has described the SLAM problem, the essential methods for solving the SLAM problem and has summarised key implementations and demonstrations of the method.

Tuning Visual Odometry parameters may directly affect the performance of rtab map.Few parameters like changing maximum features , changing the frame rate of camera and resolution, changing feature distance can greatly affect the performance. An interesting future work would be to explore the RTABMap package's visualization section in more details. A potential area would be Wifi signal strength mapping. This feature allows

the user to visualize the strength of the robots WiFi connection. This can be important in order to determine where the robot may lose its signal, therefore taking the necessary actions so that the data is not lost.Also an implementation on real robot and deploying SLAM to localize and map a real environment like a house or the periphery of the house would be very interesting.

REFERENCES

[1] FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association
[2] REAL-TIME 2D AND 3D SLAM USING RTAB-MAP, GMAPPING, AND CARTOGRAPHER PACKAGES
[3] The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures