Map My World: Simultaneous Localization and Mapping with RTAB-Map

Cheshta Dhingra

Abstract—Simultaneous localization and mapping (SLAM) is implemented in this paper. A Real-Time Appearance-Based Mapping approach is utilized to create both 2D and 3D map representations from both a provided simulated environment, as well as our own created simulated environment. We find that RTAB-Map can be an efficient solution to the SLAM problem.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Localization.

1 Introduction

There are many robotic applications in which neither the map of the environment nor the robot's position within it is known. SLAM is the computational problem of building a map of an unknown environment while simultaneously locating an agent's path within it, given a set of (noisy) controls and measurements from sensor observations. SLAM is known as a ”chicken or the egg problem” in robotics, as the robot poses are needed for localization, but the map is needed for localization. However, there exist several algorithms that can solve the problem approximately, such as the Extended Kalman Filter, Sparse Extended Information Filter, Extended Information Form, FastSLAM and GraphSLAM. In this paper, two simulated environments were leveraged to test the RTAB-Map algorithm, an implementation of RGB-D Graph-based SLAM.

2 Background

”Mapping with Known Poses” involves generating a map under the assumption that the robot poses are known and non-noisy. This problem can be solved by the Occupancy Grid Mapping Algorithm, which can estimate the posterior map given noisy measurements and known poses. However, in SLAM, the problem changes from mapping with known poses, to mapping with unknown poses. During SLAM, the robot must build a map of the environment and localize itself relative to it. After SLAM, the Occupancy Grid Mapping Algorithm uses the exact robot poses filtered from SLAM. Then, with the known poses from SLAM and noisy measurements, the mapping algorithm generates a map fit for path planning and navigation. The following subsections provide an overview of the Occupancy Grid Mapping Algorithm, the Grid-Based FastSLAM algorithm and finally, GraphSLAM.

2.1 Occupancy Grid Mapping Algorithm

The Occupancy Grid Mapping Algorithm (OGMA) is one of the popular mapping algorithms in robotics that can bes used to map arbitrary environments. To estimate the posterior map, the OGMA will uniformly partition the 2-dimensional space into a finite number of grid cells. Each of these grid cells will hold a binary random value that corresponds to the location it covers. Based on the measurements data this grid space will be filled with 0s and 1s. If there is an obstacle in a space, the cell will be occupied with a value of 1, while free spaces will have a value of 0. The number of maps that can be formed out of n grid cells is 2 raised to the power of n.

2.1.1 *Binary Bayes Filter*

The OGMA implements a binary Bayes filter to estimate the occupancy value of each cell. Initially, the algorithm takes the previous occupancy values of the cells, the poses and the measurement as parameters. the OGMA now loops through all the grid cells. In each cell, the algorithm tests if the cell is currently being sensed by the range finder sensor.

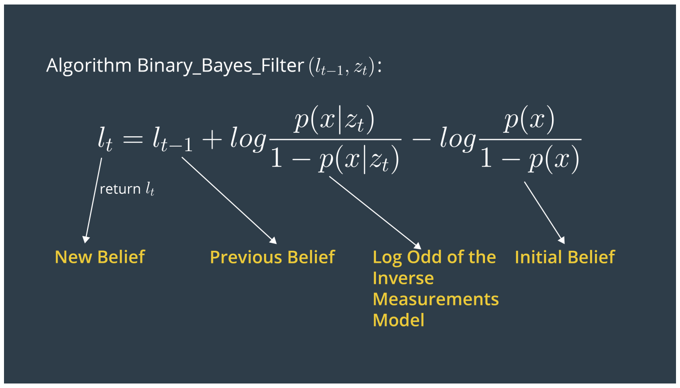

Algorithm Binary_Bayes_Filter (l_{t-1}, z_t):
$l_t = l_{t-1} + log\frac{p(x|z_t)}{1-p(x|z_t)} - log\frac{p(x)}{1-p(x)}$
return l_t
New Belief
Previous Belief
Log Odd of the Inverse Measurements Model
Initial Belief

Fig. 1. The Binary Bayes Filter Algorithm.

2.2 SLAM

There are two distinct forms of SLAM: online SLAM and full SLAM. The online SLAM problem computes a posterior over the current pose along with the map and the full SLAM problem computes a posterior over the entire path along with the map. The second key feature of the SLAM problem relates to its nature. SLAM problems generally have a continuous and a discrete element. Both robots poses and object location are continuous aspects of the SLAM problem. The answer to the question Have I been here before? is binary - either yes or no - and thats what makes the relation between objects a discrete component of the SLAM problem. This discrete relation between objects is known as correspondence.

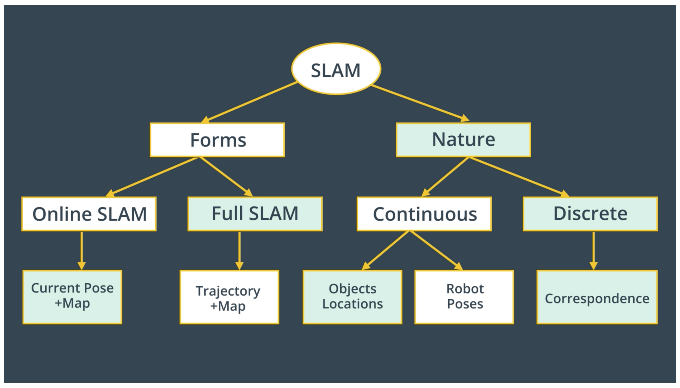


Fig. 2. Nature and Forms of the SLAM problem.

2.2.1 *FastSLAM and Grid-Based FastSLAM*

The FastSLAM algorithm solves the Full SLAM problem with known correspondences using the following key steps:
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 FastSLAM 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. However, by extending the FastSLAM algorithm to occupancy grid maps, the SLAM problem can be solved in an arbitrary environment.
Robot Trajectory
Just as in the FastSLAM algorithm, with the grid-based FastSLAM each particle holds a guess of the robot trajectory.
Map
In addition, each particle maintains its own map. The grid-based FastSLAM algorithm will update each particle by solving the mapping with known poses problem using the occupancy grid mapping algorithm.

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 u
- 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

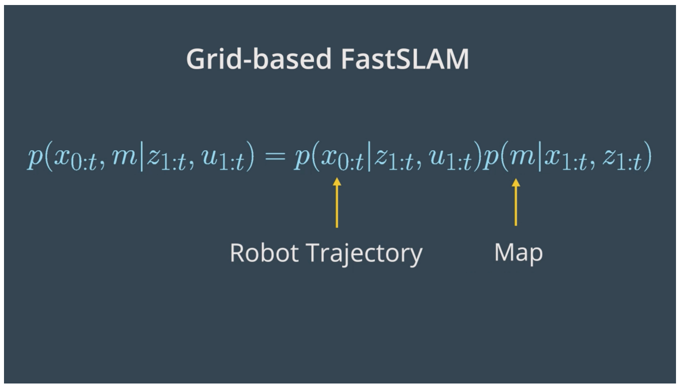

Grid-based FastSLAM
$p(x_{0:t}, m|z_{1:t}, u_{1:t}) = p(x_{0:t}|z_{1:t}, u_{1:t})p(m|x_{1:t}, z_{1:t})$
Robot Trajectory
Map

Fig. 3. Grid-Based FastSLAM.

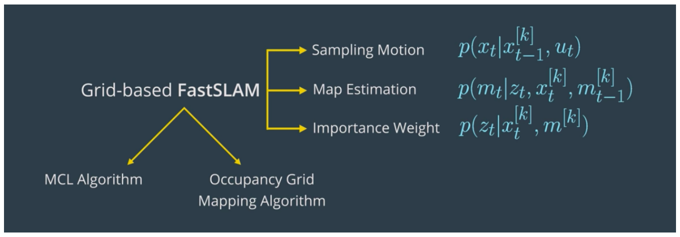

Grid-based FastSLAM
Sampling Motion
$p(x_t|x_{t-1}^{[k]}, u_t)$
Map Estimation
$p(m_t|z_t, x_t^{[k]}, m_{t-1}^{[k]})$
Importance Weight
$p(z_t|x_t^{[k]}, m^{[k]})$
MCL Algorithm
Occupancy Grid Mapping Algorithm

Fig. 4. Grid-Based FastSLAM.

2.3 GraphSLAM and RTAB-Map

GraphSLAM uses a graph to represent the Full SLAM problem. A graph is constructed in which nodes represent robot poses or landmarks and edges between two nodes symbolize a sensor measurement that constrains the connected poses. Motion constraints tie together two poses while measurement constraints tie together a feature and a pose. The goal of GraphSLAM is to find a configuration of the nodes that minimize the error introduced by the constraints, using a principle known as Maximum Likelihood Estimation (MLE).
GraphSLAM can be viewed as a 5 step process:

- Construct a graph
- Define the constraints
- Optimize to solve the system of equations
- Linearize as most motion and measurement constraints are non-linear
- Iterate

In this project we use RTAB-Map, which is a GraphSLAM approach that finds loop closure with Visual Bag-of-Words. The algorithm used for loop closure detection is Speeded Up Robust Features (SURF). When a loop closure hypothesis is accepted a new constraint is added to the graph. A graph optimizer then minimizes the errors in the map.

3 World and Robot Configuration

3.1 World Configurations

The provided world consisted of a Kitchen and Dining room setup and was a pre-built model available through gazebo.

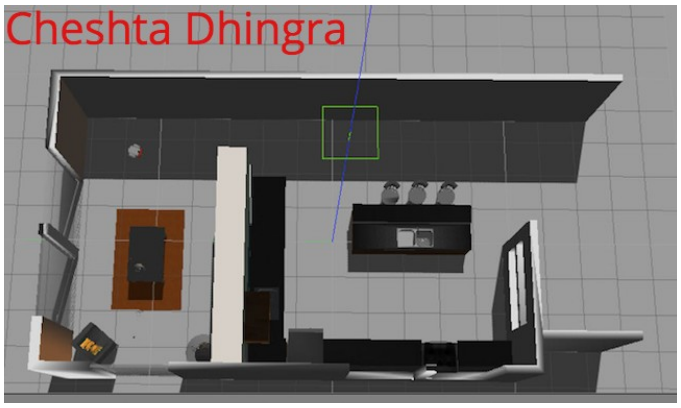


Fig. 5. Kitchen Dining World.

The second world consisted of a playground scene and was created for this project using the gazebo model building features. Elements were added to the scene using the model repository in gazebo. The scene consists of a set of playground equipment, 2 people, 2 trees, a fire hydrant and a trash can. The limits of the playground are defined by the brick walls that surround it.

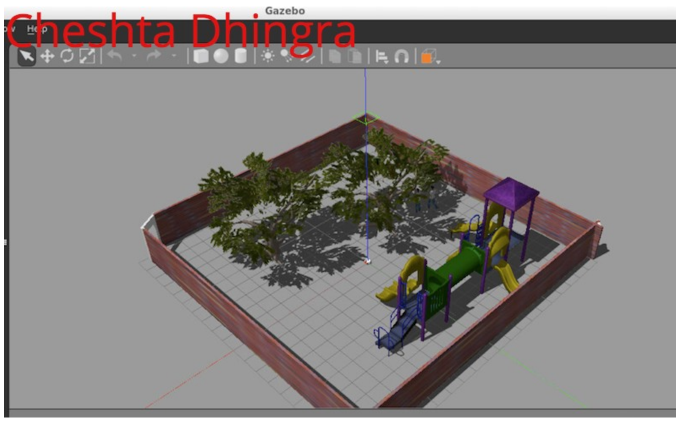


Fig. 6. Playground World.

3.2 Robot Model Configuration

The robot model chosen was that created in our Udacity localization project. The major modification was the addition of a Kinect RGB-D camera to this robot.

3.3 Package Structure

The key components of the slam_project package are the 4 launch scripts:

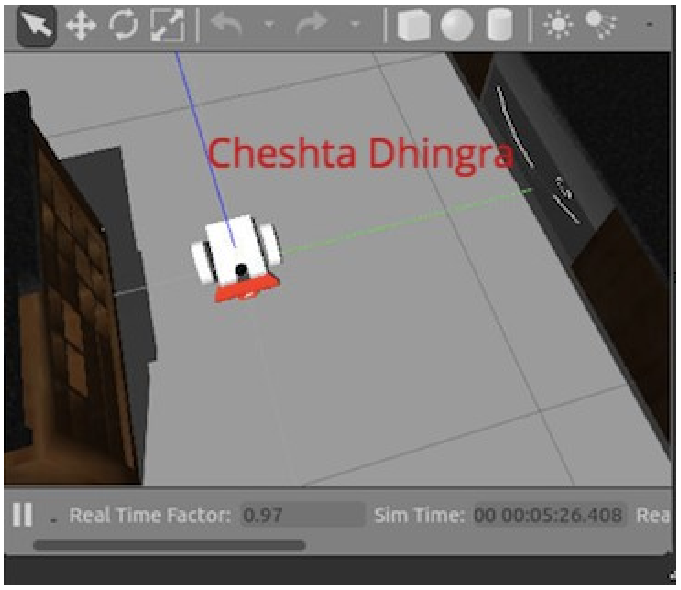


Fig. 7. Robot Model.

- dining.launch / playground.launch: includes the specific world file to launch, the robot description launch file, the Depth-Image-To-Laser-Scan Nodelet, and spawns the robot in the specified world in gazebo
- mapping.launch: key launch file that launches RTAB-map server to produce the map and find loop closures
- teleop.launch: launches the teleop node to allow keyboard control of the robot
- rviz.launch: launches RViz with the custom robot_slam.rviz launch configuration

The transform tree and rqt-graph are shown below.

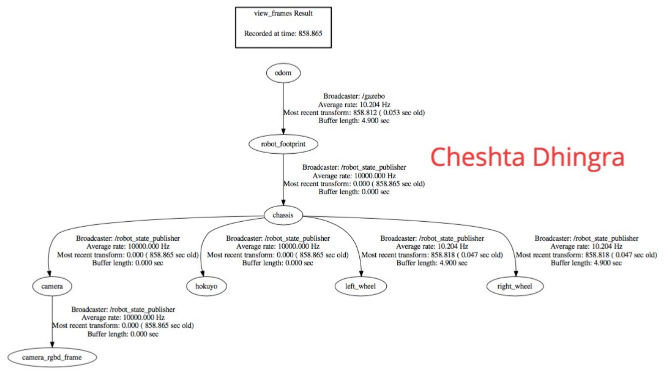

view_frames Result
Recorded at time: 858.865
odom
Broadcaster: /gazebo
Average rate: 10.204 Hz
Most recent transform: 858.812 (0.053 sec old)
Buffer length: 4.900 sec
robot_footprint
Cheshta Dhingra
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (858.865 sec old)
Buffer length: 0.000 sec
chassis
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (858.865 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (858.865 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 858.818 (0.047 sec old)
Buffer length: 4.900 sec
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 858.818 (0.047 sec old)
Buffer length: 4.900 sec
camera
hokuyo
left_wheel
right_wheel
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (858.865 sec old)
Buffer length: 0.000 sec
camera_rgbd_frame

Fig. 8. Transform Frame Tree.

4 Results

4.1 Dining World

Figure 9 shows the setup used to perform SLAM. The Gazebo display is on the right, followed by the RViz visualization and the RTAB Map window that carries out realtime feature detection, loop closures, and other relevant information to the mapping process. The pose displayed in

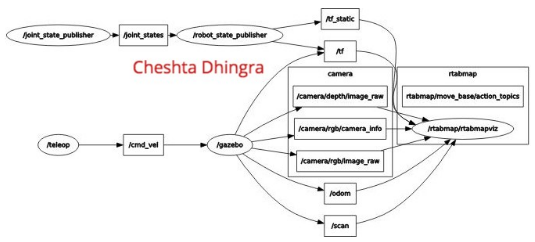


Fig. 9. Transform Frame Tree.

this figure is towards the beginning of the SLAM process where the robot has just started to explore the kitchen environment.

Figure 10 shows the point where half the environment has been mapped. Fig 11 is the final 3D map of the kitchen dining environment as well as the path taken by the robot during SLAM. The left side of the screen shows odometry data as well as loop closure detection.

Figure 12 and 13 display the final 3D and 2D maps as captured in RViz.

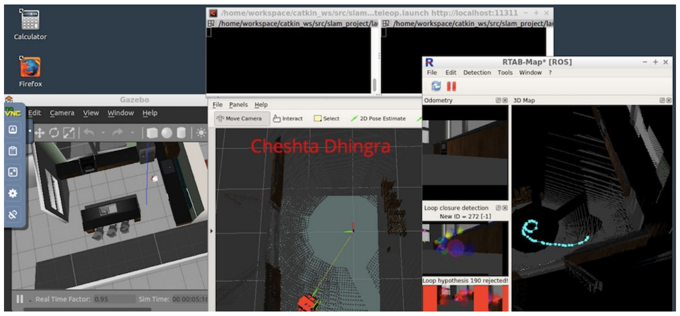


Fig. 10. SLAM Setup - Gazebo, RRViz, RTAB Map.

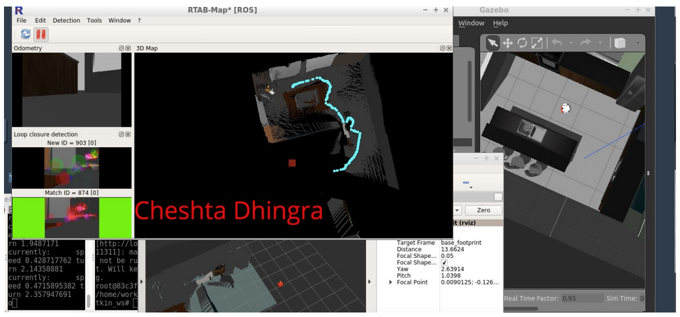


Fig. 11. SLAM Setup - Gazebo, RRViz, RTAB Map.

4.2 Playground World

The figures below each represent a different aspect of the playground world as the robot mapped it. In Fig. 13 the robot maps the slide set in the playground, in Fig. 14 it maps the fire hydrant and the trash can. Similarly in Fig. 15 the central tree is mapped and in Fig. 16 the robot is maneuvering through the slide set mapping some of its

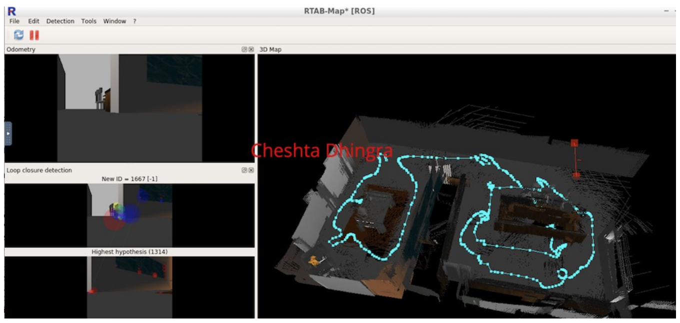


Fig. 12. Final 3D Map - RTAB Map.

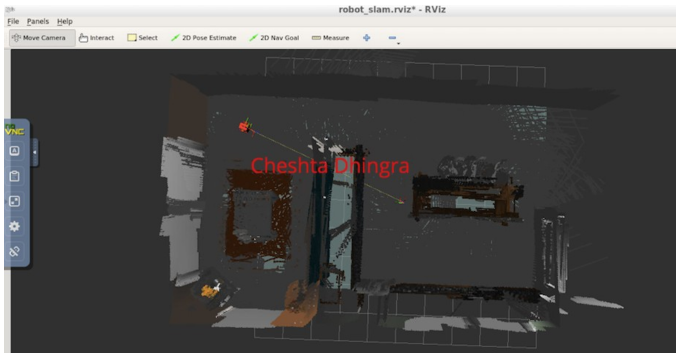


Fig. 13. Final 3D Map - RViz.

intricate details.

5 Discussion

Overall, the Graph SLAM algorithm is successful at mapping an unknown environment and localizing itself within it. Crucially, the detection of loop closures is performed using the RTAB-Map algorithm, allowing the robot to remember features that have already been encountered. In order to fully leverage the loop closure detection functionality, the robot was frequently rotated in position, allowing it to capture the 360 degrees surrounding it.
Both worlds performed reasonably well. However, in the playground world, the large size of the environment combined with the relative lack of features made it much more difficult to perform SLAM. There were some issues with the tools crashing when very similar features were encountered,

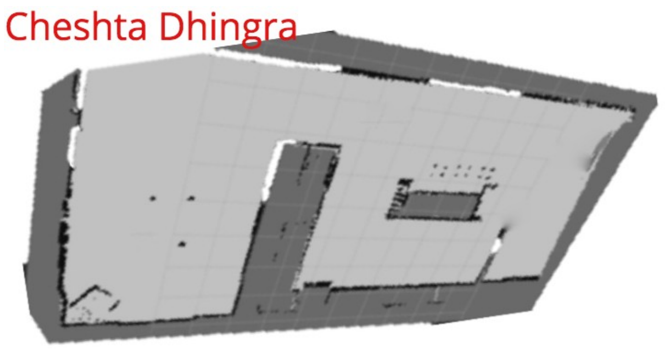
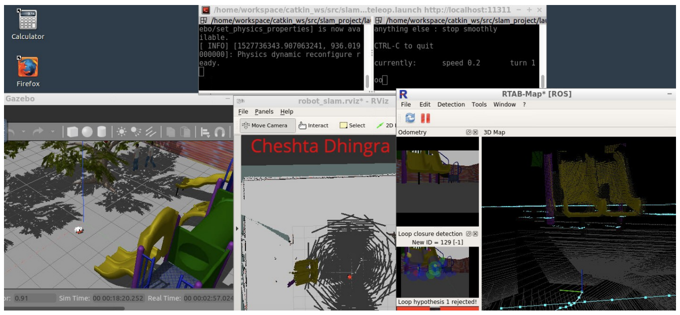


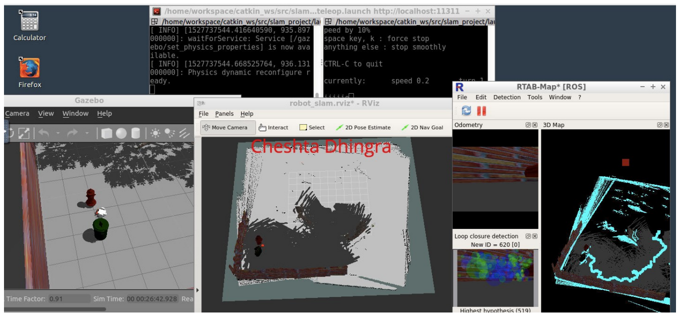
Fig. 14. Final 2D Map.



```
ebo/set_physics_properties] is now ava
ilable.
[ INFO] [1527736343.907063241, 936.019
000000]: Physics dynamic reconfigure r
eady.
```

```
anything else : stop smoothly

CTRL-C to quit

currently:      speed 0.2       turn 1

oo
```

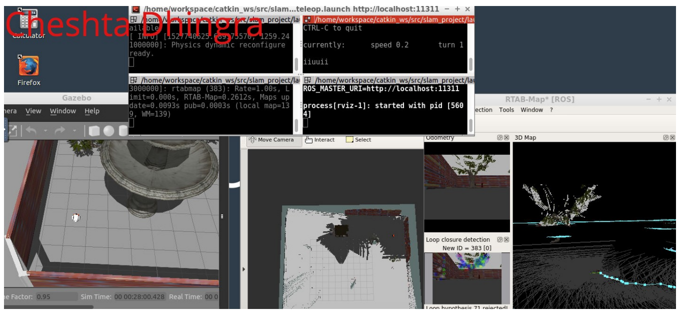
Fig. 15. Mapping the Slide Set in the Playground Environment.



```
[ INFO] [1527737544.416640590, 935.897
000000]: waitForService: Service [/gaz
ebo/set_physics_properties] is now ava
ilable.
[ INFO] [1527737544.668525764, 936.131
000000]: Physics dynamic reconfigure r
eady.
```

```
peed by 10%
space key, k : force stop
anything else : stop smoothly

CTRL-C to quit

currently:      speed 0.2
```

Fig. 16. Mapping the Fire Hydrant + Trash Can in the Playground Environment.


Cheshta Dhingra
/home/workspace/catkin_ws/src/slam...teleop.launch http://localhost:11311
Gazebo
RTAB-Map* [ROS]
Odometry
3D Map
Loop closure detection

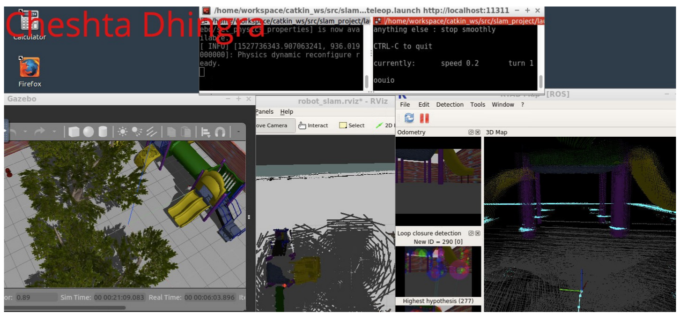
```
ailable.
[ INFO] [1527740625.689175570, 1259.24
1000000]: Physics dynamic reconfigure
ready.
```

```
CTRL-C to quit

currently:      speed 0.2       turn 1

iiuuii
```

```
3000000]: rtabmap (383): Rate=1.00s, L
imit=0.000s, RTAB-Map=0.2612s, Maps up
date=0.0093s pub=0.0003s (local map=13
9, WM=139)
```

```
ROS_MASTER_URI=http://localhost:11311

process[rviz-1]: started with pid [560
4]
```

Fig. 17. Mapping the tree in the Playground Environment.



```
ebo/set_physics_properties] is now ava
ilable.
[ INFO] [1527736343.907063241, 936.019
000000]: Physics dynamic reconfigure r
eady.
```

```
anything else : stop smoothly

CTRL-C to quit

currently:      speed 0.2       turn 1

oouio
```

Fig. 18. Mapping the inside of the Slide Set in the Playground Environment.

such as the people in the environment, who looked quite similar despite one being ”standing” and the other ”walking”.

The placement of the sensor may be modified to improve detection of features, especially those that are relatively tall. It would be valuable to have a much more flexible RGB-D camera setup that can access remote areas that the robot itself may not be able to.

6 Future Work

In the future, we will apply the Graph SLAM algorithm to a physical robot. It would be interesting to create a vacuum robot that maps an indoor environment and navigates around furniture.