SLAM Project: Map My World Robot

Dilmuratkyzy Zerde

Abstract—This report studies Simultaneous Localization and Mapping (SLAM) problem using Real-Time Appearance-Based Mapping (RTAB-Map) in a Gazebo environment. A robot model from the previous localization project was used with an added RGB-D camera to map both predefined and custom made environment. With data from robot sensors and odometry data , using RTAB-Map ROS library database of environments were built, and both 2D and 3D maps were generated.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, SLAM, RTAB-Map, ROS.

1 Introduction

SIMULTANEOUS localization and mapping, SLAM, is the problem of constructing a map of an environment while simultaneously localizing a robot relative to this map, where neither the map or position of the robot are provided.

This report showcases simultaneous localization and mapping on the two simulated Gazebo environments. First environment *kitchen_dining* was provided as a part of the project, and the second one is created independently using Gazebo. Both environments were successfully mapped using robot model, equipped with RGB-D camera and Hokuyo Lidar, and rtab-map library, 2D and 3D maps of environments were generated.

2 Background

SLAM is fundamental problem in mobile robotics, because in order for robots to be useful for society, they must be able to move around in unknown environments. In real life, our surrounding is often changing and not static, thus it is not convenient or often impossible to update map for robots every time a change happens in its environment. SLAM is very challenging task, since with noises in robot's motions and sensor data, the map and robot's pose will be uncertain, and the accuracy of the map and the accuracy of the localization depends on each other. SLAM algorithm generally fall into five categories:

- Extended Kalman Filter SLAM
- Sparse Extended Information Filter
- Extended Information Form
- FastSLAM
- GraphSLAM

Following sections describes the two of the commonly used SLAM algorithms: Grid-based FastSLAM and GraphSLAM

2.1 Grid-based FastSLAM

The FastSLAM algorithm solves the SLAM problem with predefined landmark positions using particle filter approach . Each particle holds the robot trajectory and a map. FastSLAM divides SLAM into two separate independent problems, estimating the trajectory and estimating the map. FastSLAM estimates a posterior over the trajectory using a particle filter approach, and uses a low dimensional Extended Kalman Filter to solve independent features of the map which are represented with local Gaussian.

Grid-based FastSLAM is extension of FastSLAM, which adapts FastSLAM into grid map. With grid-based algorithm the environment can be modeled using grid maps, without any predefined landmark positions, thus it can solve SLAM problem in an arbitrary environment. To adapt FastSLAM to grid mapping three different techniques are used:

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

The sampling motion and importance weight techniques can be solved with the MCL algorithm, whereas the map estimation technique can be solved with the occupancy grid mapping algorithm.

2.2 GraphSLAM

GraphSLAM solves the Full SLAM problem, it uses nodes to represent robot poses or landmarks, and constraints to represent relationships between nodes and the environment, and then resolve all of these constraints and data into a map.Its advantage over Grid-based FastSLAM is that it doesnt use particle, when the particle filter approach is used, there is always a possibility that there won't be any particle at the robots actual position.

Graph based SLAM problem can be broken up into two sections: the front-end and the back-end. Front-end is constructing graph using motion and sensory data collected by the robot. It is consisted of interpreting sensory data, building the graph, adding new data to it as the robot moves around, also the front-end solves the data association problem. The back-end takes the complete graph with all the constraints that have already been built in the previous step as a input and results the most probable configuration of the robot poses and map features. The front-end step can vary greatly from application to application, where back-end is a lot more consistent across various applications.

Real-Time Appearance-Based Mapping or RTAB-Map is graph-based SLAM approach that uses data collected from vision sensors to localize the robot and construct the map of environment. In this project RTAB-Map is used to map the environment.

3 Scene and robot configuration

3.1 Robot configuration

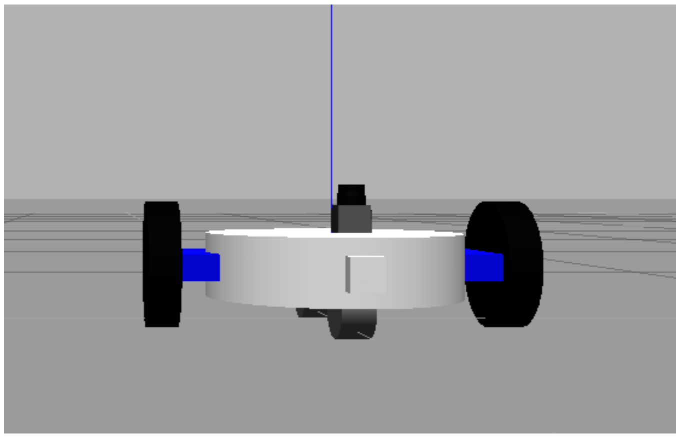


Fig. 1. Robot Model

Robot model is based on robot model from previous localization project. It is consisted of base_cylinder with radius 0.2 and length 0.1, base_box with size 0.15x 0.5 x 0.04 and same origin as the base_cylinder, two wheels with 0.1 radius attached to left and right sides of base_box, and front and back wheels with radius 0.0499 length 0.05 underneath the robot body for stability. The robot is equipped with Hokuyo Lidar that is attached to the top of the robot base, and RGB-D camera mounted to front of the cylinder facing front.

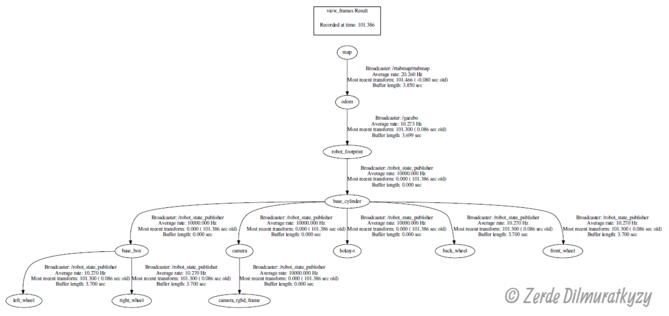


Fig. 2. Robot model configuration and frames

3.2 Packages Used

RTAB-Map ROS library is used to build the map, launched by mapping.launch file. It takes laser scan data from Hokuyo Lidar, image and depth data from RGB-D camera and odometry movement data to construct 2D and 3D maps of the environments . mapping.launch file also launches real time mapping visualization tool rtabmapviz *teleop* script is used to move the robot around using keyboards.

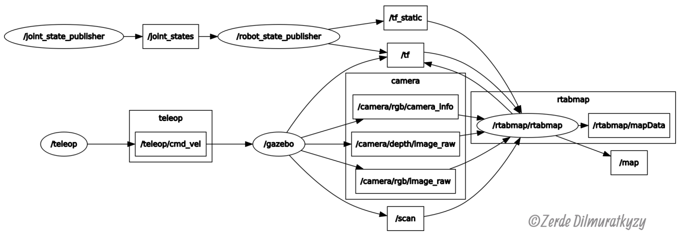


Fig. 3. ROS graph

3.3 Worlds

Mapping is performed to two gazebo environments. First one is predefined *kitchen_dining* (Fig.4) world provided by udacity. Second world *myworld* (Fig.5) is constructed using gazebo library items. Different types of objects with different texture, size, and shape were added to test out how different types of objects will be mapped and to add more features to the environment.

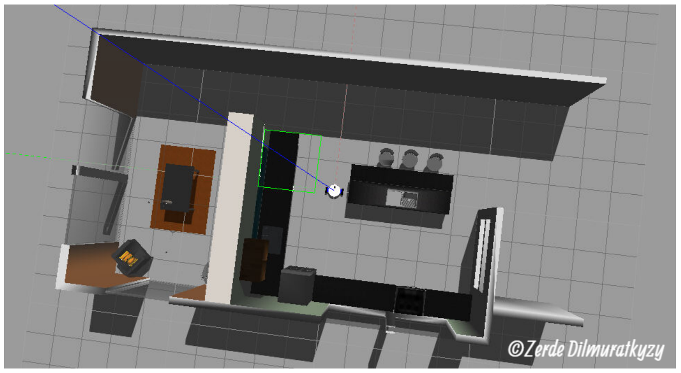


Fig. 4. kitchen_dinig world

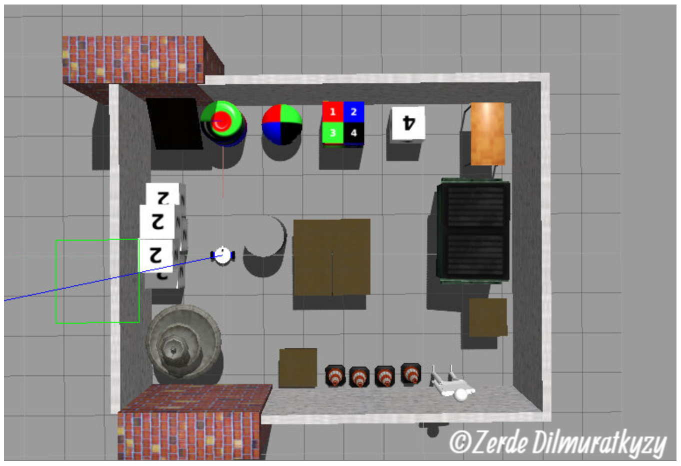
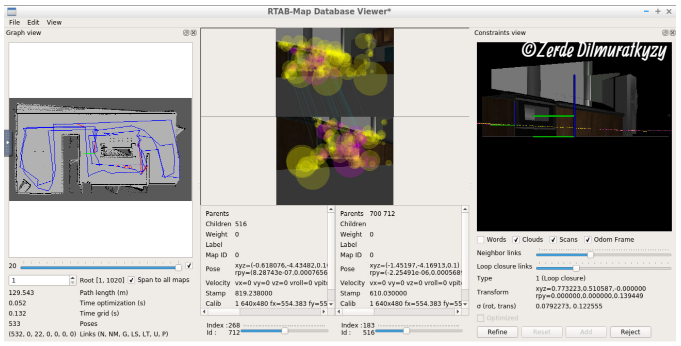


Fig. 5. myworld

4 Results

After moving robot around the word in several loops the map of both environments were successfully built. Both 2d occupancy grid map and 3d map were constructed. Following images showcases mapping results for both worlds.



```
20
1   Root [1, 1020] ✓ Span to all maps
129.543              Path length (m)
0.052                Time optimization (s)
0.132                Time grid (s)
533                  Poses
(532, 0, 22, 0, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)

Parents
Children  516
Weight    0
Label
Map ID    0
Pose      xyz=(-0.618076,-4.43482,0.1
          rpy=(8.28743e-07,0.0007656
Velocity  vx=0 vy=0 vz=0 vroll=0 vpit
Stamp     819.238000
Calib     1 640x480 fx=554.383 fy=55
Index :268
Id :   712

Parents   700 712
Children
Weight    0
Label
Map ID    0
Pose      xyz=(-1.45197,-4.16913,0.1)
          rpy=(-2.25491e-06,0.000568
Velocity  vx=0 vy=0 vz=0 vroll=0 vpit
Stamp     610.030000
Calib     1 640x480 fx=554.383 fy=55
Index :183
Id :   516

☐ Words ✓ Clouds ✓ Scans ✓ Odom Frame
Neighbor links
Loop closure links
Type             1 (Loop closure)
Transform        xyz=0.773223,0.510587,-0.000000
                 rpy=0.000000,0.000000,0.139449
σ (rot, trans)   0.0792273, 0.122555
☐ Optimized
```

Fig. 6. rtab-map databaseViewer kitchen_dining

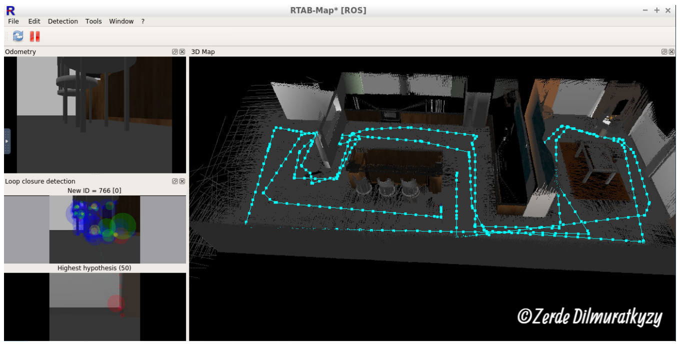
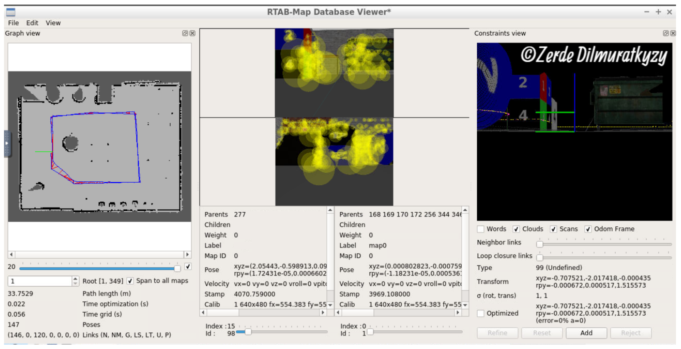


Fig. 7. Mapped kitchen_world



```
20
1   Root [1, 349] ✓ Span to all maps
33.7529              Path length (m)
0.022                Time optimization (s)
0.056                Time grid (s)
147                  Poses
(146, 0, 120, 0, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)

Parents   277
Children
Weight    0
Label
Map ID    0
Pose      xyz=(2.05443,-0.598913,0.09
          rpy=(1.72431e-05,0.0006602
Velocity  vx=0 vy=0 vz=0 vroll=0 vpit
Stamp     4070.759000
Calib     1 640x480 fx=554.383 fy=55
Index :15
Id :   98

Parents   168 169 170 172 256 344 34
Children
Weight    0
Label     map0
Map ID    0
Pose      xyz=(0.000802823,-0.000759
          rpy=(-1.18231e-05,0.000536
Velocity  vx=0 vy=0 vz=0 vroll=0 vpit
Stamp     3969.108000
Calib     1 640x480 fx=554.383 fy=55
Index :0
Id :   1

☐ Words ✓ Clouds ✓ Scans ✓ Odom Frame
Neighbor links
Loop closure links
Type             99 (Undefined)
Transform        xyz=-0.707521,-2.017418,-0.000435
                 rpy=-0.000672,0.000517,1.515573
σ (rot, trans)   1, 1
☐ Optimized      xyz=-0.707521,-2.017418,-0.000435
                 rpy=-0.000672,0.000517,1.515573
                 (error=0% a=0)
```

Fig. 8. rtab-map databaseViewer myworld

5 Discussion

The main challenge of the project was setting up all the topics and nodes correctly. After debugging and tweaking using debugging tools such as tf library, roswtf mapping process was launched. When moving robot around the world sometimes robot will stay at one place ignoring the moving commands, this might be caused by launching the real time visualization of mapping process and increasing computational cost , and when forming loop closures, then Kp/MaxFeatures parameter was reduced by half and robot moved around at a normal speed.

Also distortion of map was observed when mapping the custom world if after moving the robot around the room

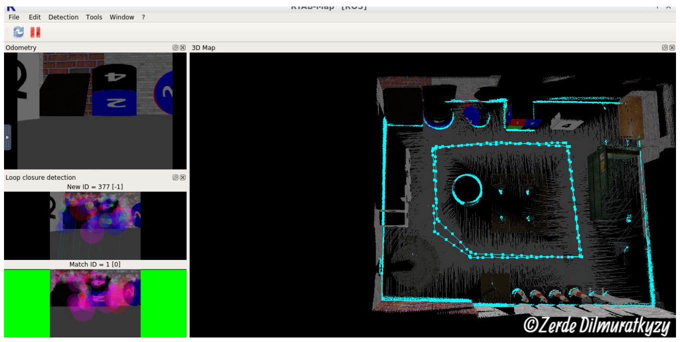


Fig. 9. Mapped myworld

first time in one direction and moving the robot in another direction the next round. When myworld contained smaller number of similar objects, results were poor due to the similarity of different parts of the environment. After adding more objects with different textures, sizes and shapes quality of resulting map was improved.

6 Future work

For this project future work would be improving mapping and adding localization , as well as exploring other features of rtabmap library such as object detection and obstacle detection.

RTAB-Map mapping performs reasonably well in environments containing good amount of distinctive features. Simultaneous mapping and localizing algorithms can be applied to real world robots such as hotel room service robot, or robots in big warehouses that moves goods from one point to another, as in this scenarios surrounding environment is constantly changing with people moving around all the time, and change of positions of furnitures or shelves.