Map My World Robot

Christoph Doerr

Abstract—This study shows the implementation of a GraphSLAM algorithm, to generate a 2D occupancy grid and a 3D octomap of two simulated environment in Gazebo. A cleaning robot as known from industry applications equipped with a RGB-D camera and a 2D Lidar sensor was simulated to traverse two Gazebo world environments, one provided, and another custom build environment. The robot maps the environment using the Real-Time-appearance-Based Mapping (RTAB-Map) GraphSLAM algorithm. The created maps were both evaluated on accuracy.

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

1 Introduction

It is essential for a mobile robot to map its dynamic surrounding, while localizing itself with noisy sensor data and control inputs. 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 study shows SLAM on two different simulated Gazebo environments. The kitchen dining environment was provided by Udacity as part of the project and the second one was created independently using the Gazebo build-in environments and objects. Both environments were mapped by a robot model, equipped with a RGB-D camera and a Hokuyo Lidar. The maps were generated with the rtab-map library.

2 Background

Given that the map is needed for localization as well as the pose of the robot is needed for generating a map, this problem is mostly called the chicken or the egg problem of SLAM. This is a challenging problem, because it has to quickly solve two related problems simultaneously and back-to-back. Extended Kalman Filter SLAM, Sparse Extended Information Filter, Extended Information Form, FastSLAM and GraphSLAM are one of the current used SLAM algorithms. The following sections will give a short overview about FastSLAM and GraphSLAM.

2.1 Grid-based FastSLAM

The FastSLAM algorithm uses a traditional particle filter approach to solve the Full SLAM problem with known correspondences. With these particles, the FastSLAM evaluates a posterior over the robot path along with the map. Each of these particles holds the robot trajectory which will give the advantage to SLAM to solve the problem of mapping with known poses. Additionally, each particle holds a map and each feature of the map is represented by a local Gaussian. FastSLAM has a big disadvantage since it must always assume that there are known landmark positions and this with FastSLAM we are not able to model an arbitrary environment. Grid-based FastSLAM additionally adapts FastLAM into a grid map. The environment can be modeled with grid-based algorithms, without any predefined landmark positions. Therefore, it can solves the SLAM problem in an arbitrary environment. The different techniques are used to adapt FastSLAM into a grid map as the following list shows:

- 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 the 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 with nodes, which represent robot poses or landmarks as well as constraints which represent relationships between nodes and the environment. It uses these constraints and data to create a map. The absence of particles in this approach is the main advantage of the GraphSLAM, because there is always a probability that there are no particles at the actual position of the robot, while using particle filters. The GraphSLAM approach can be divided into two parts:

- **Front-End** is the construction of a graph using motion and sensory data collected by the robot. These data is used to build the graph. Furthermore, it solves the data association problem
- **Back-End** takes the complete graph with all the constraints that have already been built in the Front-End step. This step is way more consistent compared to the Front-End step.

RTAB-Map is a graph-based SLAM solution, which uses data collected from vision sensors to localize the robot and construct the map of the environment [1].

3 Scene and Robot Configuration

To perform a SLAM task, the robot needs an environment to map and localize itself. Two different environments are used during this study to generate a 2D and a 3D map of each world.

3.1 Kitchen and Dining Scene

The kitchen scene was provided by Udacity and will be used as a first environment to perform the SLAM task. It is shown in figure 1

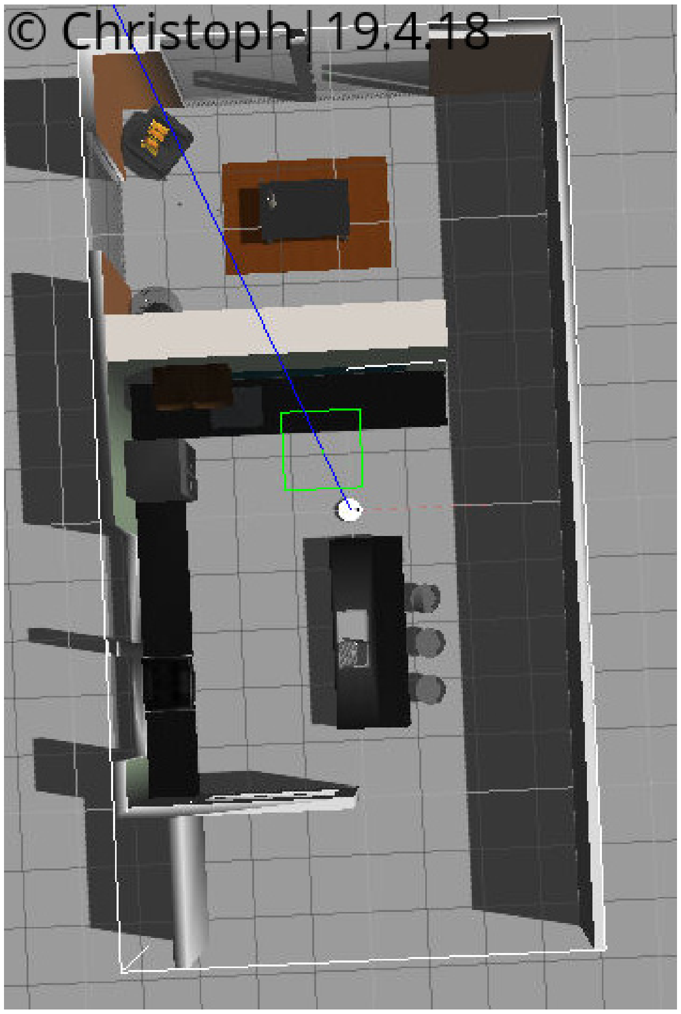


Fig. 1. Udacity's kitchen world in Gazebo

3.2 My World

The my wold setup was created with Gazebo and its build in environments and objects. This scene was created with the SLAM task in mind, therefore it is rich of objects and features to make the mapping and localizing problem as easy as possible for the robot. Figure 2 represents the created world.

3.3 Robot Configuration

The robot configuration was created for the previous localization project. It was reequipped with a 2D Hokuyo

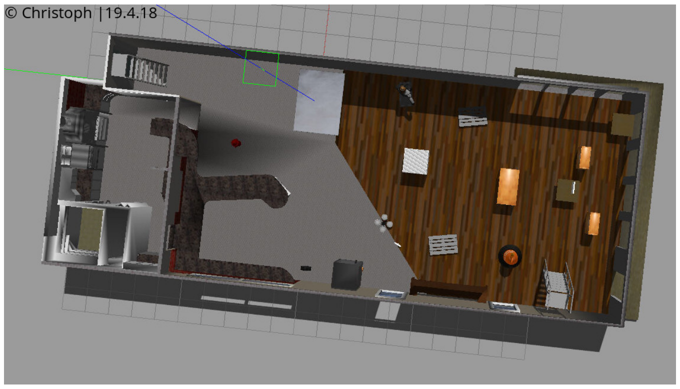


Fig. 2. My World in Gazebo

Lidar and a RGB-D kinect camera to perform the SLAM task. With this RGB-D camera it can detect the depth of the environment. This sensor setup allows the robot to perform SLAM with the rtabmap-ros package . The configuration of the robot can be seen in figure.

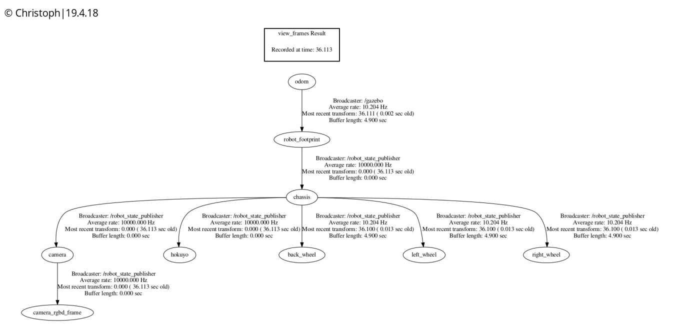

© Christoph|19.4.18
view_frames Result
Recorded at time: 36.113
odom
Broadcaster: /gazebo
Average rate: 10.204 Hz
Most recent transform: 36.111 (0.002 sec old)
Buffer length: 4.900 sec
robot_footprint
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (36.113 sec old)
Buffer length: 0.000 sec
chassis
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (36.113 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (36.113 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 36.100 (0.013 sec old)
Buffer length: 4.900 sec
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 36.100 (0.013 sec old)
Buffer length: 4.900 sec
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 36.100 (0.013 sec old)
Buffer length: 4.900 sec
camera
hokuyo
back_wheel
left_wheel
right_wheel
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (36.113 sec old)
Buffer length: 0.000 sec
camera_rgbd_frame

Fig. 3. The tf tree of the robot

4 Results

After moving the robot through the mentioned world, the 2D occupancy grid map and 3D map could be constructed. The mapping of the own created map was more difficult, because it was larger and the objects were not fixed to the ground and moved by crashing into them with the robot. Following figures show the mapping results of the kitchen and the own created world.

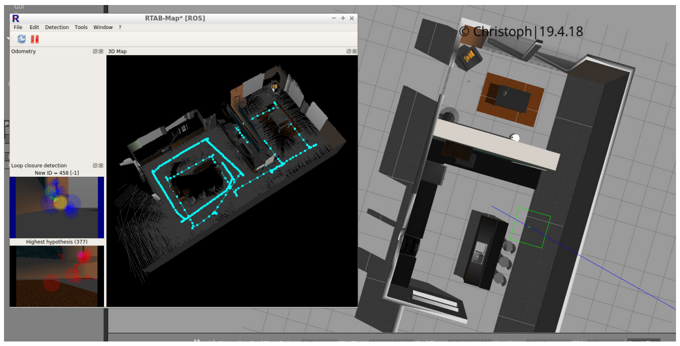
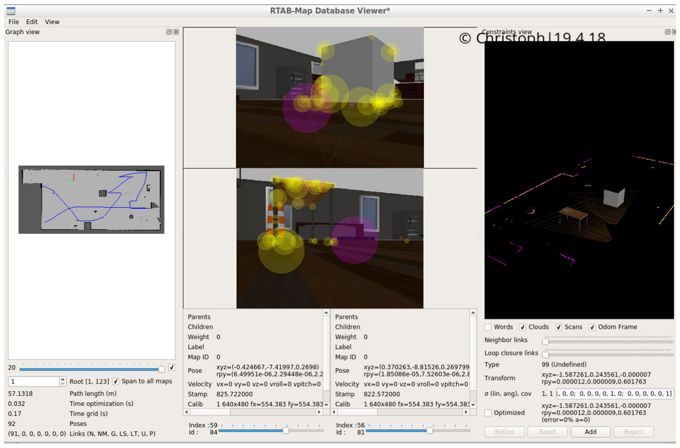


Fig. 4. The kitchen environment in Gazebo and the 3D map



```
20
1            Root [1, 123]  ✓ Span to all maps
57.1318      Path length (m)
0.032        Time optimization (s)
0.17         Time grid (s)
92           Poses
(91, 0, 0, 0, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)

Parents
Children
Weight    0
Label
Map ID    0
Pose      xyz=(-0.424667,-7.41997,0.2698)
          rpy=(6.49951e-06,2.29448e-06,2.2
Velocity  vx=0 vy=0 vz=0 vroll=0 vpitch=0
Stamp     825.722000
Calib     1 640x480 fx=554.383 fy=554.383
Index :59
Id :   84

Parents
Children
Weight    0
Label
Map ID    0
Pose      xyz=(0.370263,-8.81526,0.269799
          rpy=(1.85086e-05,7.52603e-06,2.8
Velocity  vx=0 vy=0 vz=0 vroll=0 vpitch=0
Stamp     822.572000
Calib     1 640x480 fx=554.383 fy=554.383
Index :56
Id :   81

Neighbor links
Loop closure links
Type            99 (Undefined)
Transform       xyz=-1.587261,0.243561,-0.000007
                rpy=0.000012,0.000009,0.601763
σ (lin, ang), cov  1, 1  l, 0, 0;  0, 0, 0, 0, 1, 0;  0, 0, 0, 0, 0, 1]
Optimized       xyz=-1.587261,0.243561,-0.000007
                rpy=0.000012,0.000009,0.601763
                (error=0% a=0)
```

Fig. 5. the kitchen in the trabmap dataviewer

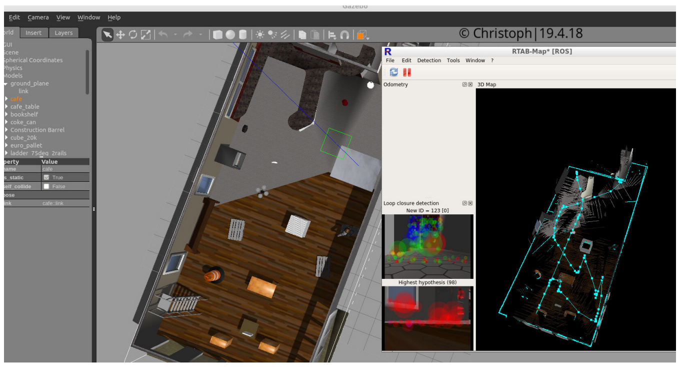
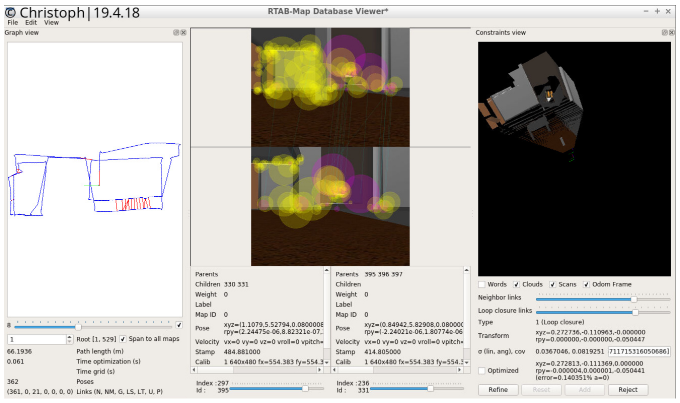


Fig. 6. My World in in the rtabmap dataviewer

5 Discussion

6 Conclusion / Future work

Future work could be improving the mapping and adding localization, as well as exploring other features of rtabmap library such as object detection and obstacle detection.This RTAB-Map could be implemented on a Jetson TX2 as well as a kinect RGBD camera on a small mobile robot to map real world apartments. Furthermore, this robot could be developed to a vacuum cleaner or with further implementations on an aerial mapping robot.



```
8
1            Root [1, 529]  ✓ Span to all maps
66.1936      Path length (m)
0.061        Time optimization (s)
             Time grid (s)
362          Poses
(361, 0, 21, 0, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)

Parents
Children  330 331
Weight    0
Label
Map ID    0
Pose      xyz=(1.1079,5.52794,0.0800008
          rpy=(2.24475e-06,8.82321e-07,
Velocity  vx=0 vy=0 vz=0 vroll=0 vpitch=
Stamp     484.881000
Calib     1 640x480 fx=554.383 fy=554.3
Index :297
Id :   395

Parents   395 396 397
Children
Weight    0
Label
Map ID    0
Pose      xyz=(0.84942,5.82908,0.080000
          rpy=(-2.24021e-06,1.80774e-06
Velocity  vx=0 vy=0 vz=0 vroll=0 vpitch=
Stamp     414.805000
Calib     1 640x480 fx=554.383 fy=554.3
Index :236
Id :   331

Neighbor links
Loop closure links
Type            1 (Loop closure)
Transform       xyz=0.272736,-0.110963,-0.000000
                rpy=0.000000,-0.000000,-0.050447
σ (lin, ang), cov  0.0367046, 0.0819251  711715316050686]
Optimized       xyz=0.272813,-0.111369,0.000000
                rpy=-0.000004,0.000001,-0.050441
                (error=0.140351% a=0)
```

Fig. 7. The kitchen in the rtabmap dataviewer

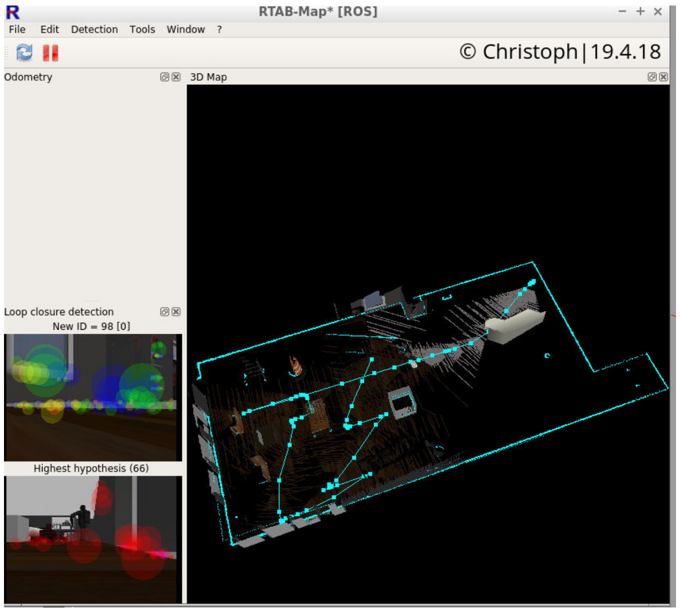


Fig. 8. Live 3D map of my world

References

[1] Udacity, *Udacity's Robotics Softwareengineer Nanodegree, SLAM Lessons*. Udacity, 2019.