RoboND Project 3 : Map My World

Yoonyoung Cho

Abstract—A ROS-compliant RGBD-feature based SLAM pipeline is realized through the RTABMap library; the full pipeline is verified in a Gazebo simulation against two distinct environment models encompassing wide feature variability. Overall, the pipeline achieves high-fidelity two-dimensional and three-dimensional maps, with low localization error of the mobile platform on the generated map despite low-quality initial motion estimates.

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

1 Introduction

Among the intelligent capabilities of an autonomous agent navigating in a relatively stable environment, the ability to generate and parse the map-like representation is critical to executing the mission at high performance. The existence of a map enables the robot to pre-plan optimized global trajectories while taking possible collisions, thereby reducing both the risk and cost of navigation.

Since the map is not automatically made available to the mobile base in most scenarios, it is often the task of the very vehicle to create the map of the environment in which it will operate. While generating a map with perfect positional information and measurements is a trivial problem, both of the input variables are prone to noise, which must be rectified. This process of considering both the positional estimates and the sensor inputs to generate more accurate mapping and localization information is known as Simultaneous Localization and Mapping, or *SLAM*.

In this project, the SLAM problem is approached with a Gazebo-simulated environment on a differential-drive mobile base equipped with three primary sensors: the wheel encoder, the time-of-flight lidar, and an RGB-D camera.

2 Background

Simultaneous Localization and Mapping, or *SLAM*, is the way in which the ambiguity between the errors in sensor inputs and the position estimates are be jointly resolved: in the scheme, the features in the environments are characterized and associated with a particular location which can be inaccurate; such a map is then compared with the measurements at each time step in an iterative process, and both the location of the robot at the time in which the measurements were taken and the map are updated according to the most plausible hypothesis that minimizes the conflicting terms.

2.1 2D SLAM

While SLAM itself is a problem in a general domain, there are two large domains in which SLAM is performed: in two-dimensional planar surfaces, and a general model spanning three-dimensional spaces. SLAM in two dimensions is a reduced-space optimization for general SLAM, where the operating range of the pose is constrained to a planar surface. This is a compelling optimization for ground-rovers operating in a generally planar environments, such as buildings, where the minimized problem offers a significant improvement in both speed and accuracy of the generated positional and mapping information.

2.2 3D SLAM

Such an optimization cannot be applied for mobile platforms operating in the full three-dimensional space, such as a drone. With the increasing popularity of robots equipped with flight capabilities, the three-dimensional SLAM is an incredibly compelling problem with unique challenges such as high degree of non-linearity and manifolds in the state-space introduced by rotations, as well as the high dimension of the domain itself and the stability of the measurements.

3 Model Configuration

The model was an adaptation of the prior model used for the localization project. Specifically, the mobile base was configured as a differential-drive robot with wheel encoders attached to each motor, providing basic odometry information for motion and localization estimation.

The two primary sensors for SLAM were the Hokuyo-flavored 2D lidar and the Kinect-like RGBD Camera, realized in gazebo through the *gazebo_ros_openni_kinect* plugin. The camera was raised from the platform and slightly pitched downwards from the front of the robot to include more visual fields encompassing the terrain information for more robust and persistent feature tracking, while the lidar was kept at the center of the platform to align the ray origin with the effective footprint.

Apart from this, one noteworthy aspect of the model is that the odometry information was inferred from the wheel encoder joint information, rather than the direct pose, or the *"world"* source in Gazebo. The rationale behind this decision was that it is a closer approximation of real-world scenarios, albeit providing a less accurate localization information.. As the contact coefficients were not tuned precisely to provide perfect odometry information, the robot was purposefully faced with a more adversarial condition with SLAM where the default localization performance was sub-optimal.

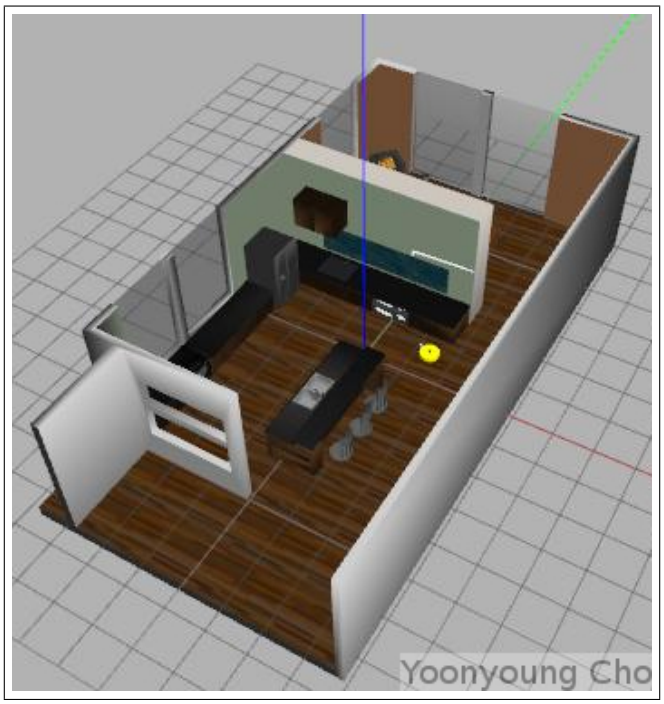


Fig. 1: View of the mobile base construction; note that semi-transparent render highlights the driven and caster wheels embedded at the bottom. The white cube marks the pose of the RGBD Camera, and the mesh at the center of the base is the lidar.

4 World Creation

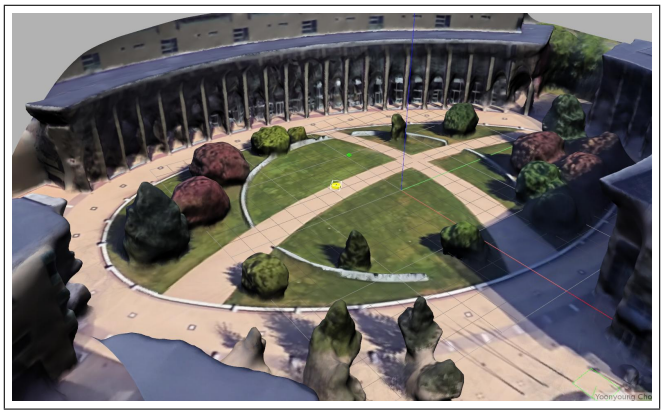


Fig. 2: Simulation of the Olin Oval environment in Gazebo, with medium-fidelity texture and mesh representations.

The world creation was a fairly involved process, in order to create a compelling simulation environment for the robot. As the supplied environment was primarily an enclosed indoors environment, it was desirable to test the configuration on a more open world, where the trackable visual features are more sparse and the overall scale is large. To this end, the most intuitive approach was to essentially *imporr* a real-world environment into the Gazebo Simulation.

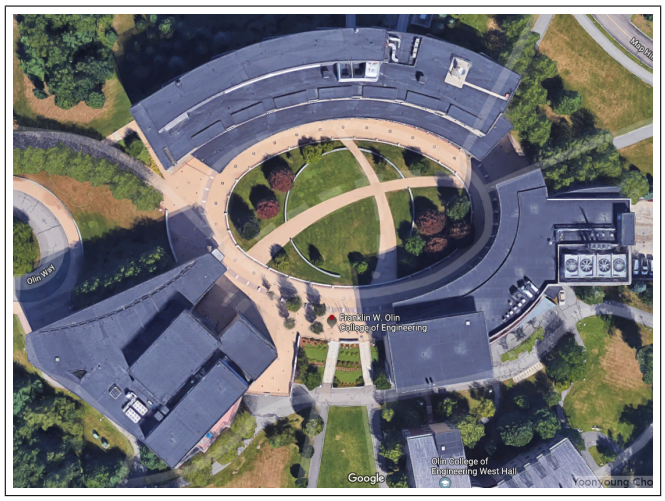


Fig. 3: Top-down view of the target area satellite image at Olin College; a representative sample from the image set

4.1 Data collection

The operating environment was a virtual re-creation of Olin College Oval, a relatively open area enclosed by three large buildings. This was done through 3D Reconstruction of the scene geometry based on Google Earth-based satellite images. While Google Earth maintains an internal three-dimensional model of the area, the mesh is not accessible and thus only obtainable through indirect means. In order to obtain a satisfactory resolution and quality of the mesh, a set of approximately 110 images were collected in *Google Earth* at various view-points.

4.2 Post-processing

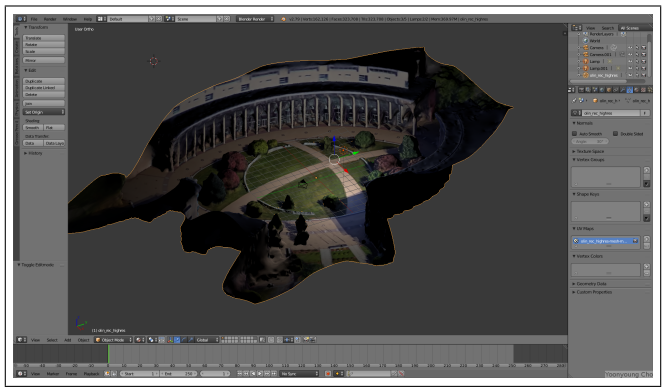


Fig. 4: Blender UI, displaying the processed model with smoother and simplified geometry for the simulation.

After the data collection process, the images were imported into *Autodesk ReCap*, which created the raw mesh. The resultant file was further processed in *Blender* in order to clean up possible artifacts, such as overly bumpy terrain. Specifically, in order to maintain the two-dimensional assumption on which the SLAM algorithm operated, the ground plane was flattened flush with the z-plane, while the geometry of wall-like obstacles were preserved.

5 Package Configuration

Overall, the package was configured in compliance with ROS best practices, where each logical unit is separated to a dedicated directory.

```
robond_map (root)
├── CMakeLists.txt
├── config
│   └── robot_slam.rviz
├── launch
│   ├── localization.launch
│   ├── mapping_2d.launch
│   ├── mapping.launch
│   ├── rviz.launch
│   ├── spawn.launch
│   ├── teleop.launch
│   └── world.launch
├── media
├── models
├── package.xml
├── scripts
│   ├── loc_eval.py
│   ├── rtab_run
│   └── teleop
├── urdf
│   ├── common.xacro
│   ├── robot.gazebo
│   └── robot.xacro
└── worlds
    ├── kitchen_dining.world
    └── olin.world
```

Fig. 5: Package directory structure; note that auxiliary entries were omitted for clarity.

Specifically, the *scripts* folder contained all executable nodes including *loc_eval.py* for evaluation of localization performance, which will be described in more detail in later sections. The model description was primarily developed in xacro to provide a more succinct, generalized specification of the robot with minimal code-duplication, where commonly used macros and parameters were stored in *common.xacro*.

Two operating environments were provided: the default *kitchen_dining.world* and the custom *olin.world*. The *world.launch* was configured to accept the file as a parameter so that the user could easily select the options during runtime.

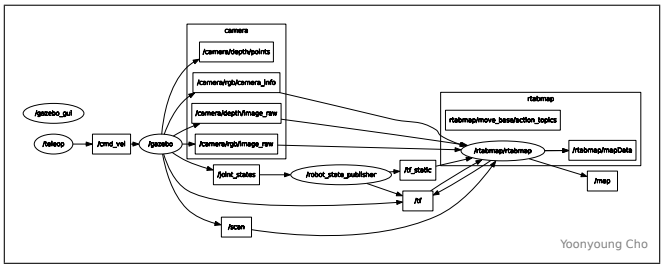


Fig. 6: ROS Graph visualization of the processing nodes and the intermediary topics.

As reference on how the nodes interact with each other, the data-flow style graph visualization is presented in Fig. 6. Note that all sensory inputs are channelled from the Gazebo simulated environment, and RTABMap consumes the data to produce the localization transformations and mapping information.

6 Results

6.1 Mapping

6.1.1 RTABMap

The primary package for slam and localization was RTABMap, or Real-Time Appearance Based Mapping. The algorithm operated on inputs from the laser scan and the rgbd image provided by the onboard sensors, as well as the wheel encoder odometry which was parametrized to accumulate significant drift over time.

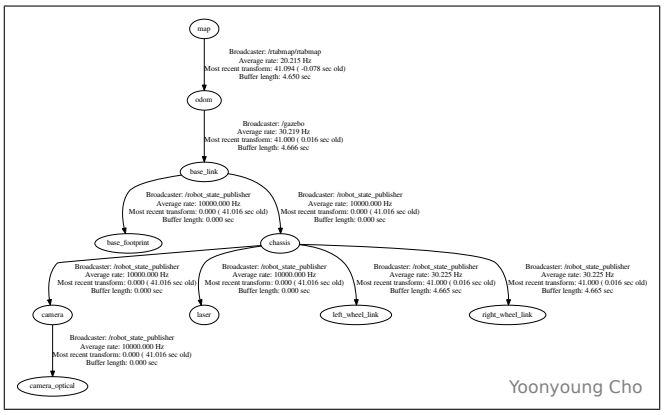
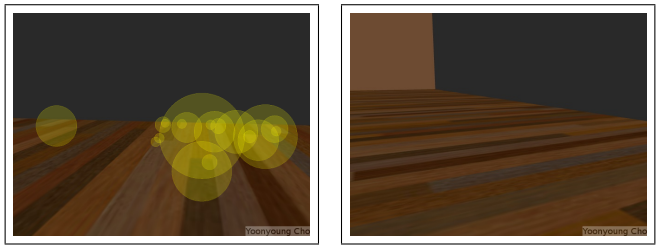


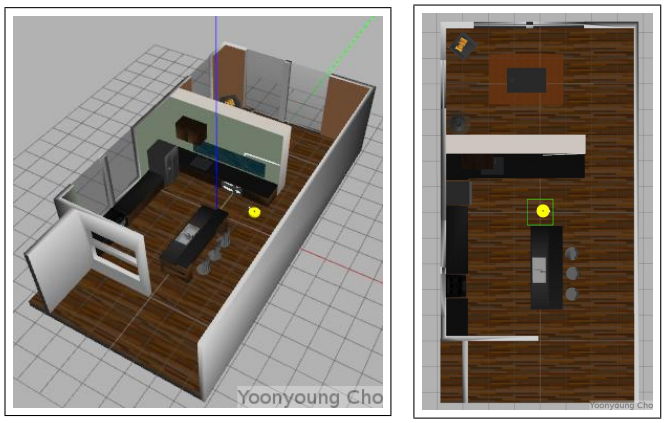
Fig. 7: TF Transformation setup in the current configuration; note that RTABMap is responsible for *map* to *odom* transformation, the final stage in the robot pose localization.

Overall, the performance of RTABMap in both simulated environments were fairly compelling. One particular challenge in the provided world configuration was the passage between the inner- and outer-rooms, which posed a significant challenge for the SLAM algorithm due to the spatial disconnect and the lack of persistent, trackable visual features in the region to correct for possible errors accumulated while traversing this region.



(a) Frame 9 (b) Frame 11

Fig. 8: Feature deprivation visualization in the corridor section; no trackable features are identified on the walls, and subsequent frames have little to no features and which leads to lack of computed correspondence.

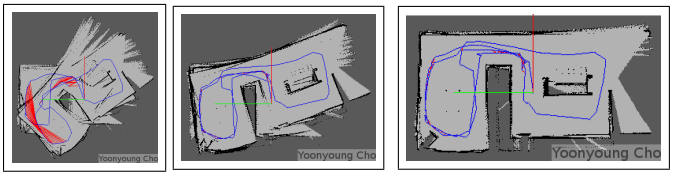


(a) Perspective view (b) Top-down view

Fig. 9: Ground truth kitchen environment display from Gazebo.

6.1.2 Kitchen

The supplied kitchen environment, as seen in Fig. 9, was a primarily indoor, planar world that was relatively straightforward in construction. In order to reveal the tiled terrain features without modifying the collision properties of the model, the visual element of the ground-plane was omitted to preserve the simulated dynamics while providing reasonable surface-level visual features.



(a) Input (b) Optimized (c) Refined

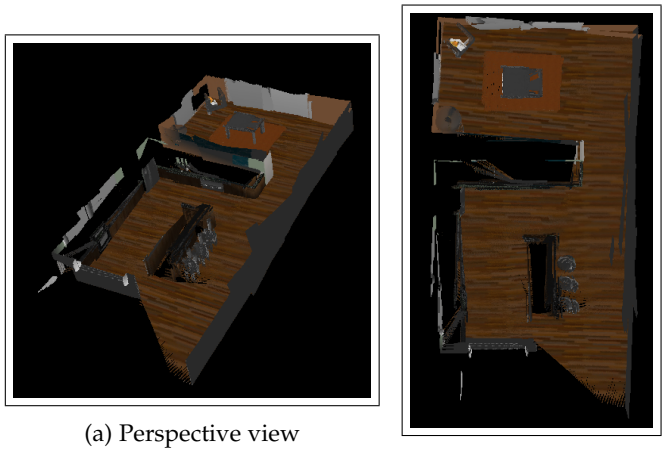
Fig. 10: Visualization of progressive improvement on the map trajectory.

Demonstration of the optimization performance with RTABMap-based SLAM is addressed in Fig. 10. While the figures were produced based on the generated database after the trajectory was run inside gazebo, the performance was similar to that seen during runtime. In particular, high degree of error and drift in odometry is noticeable in the original input image, while clean robot trajectory and perpendicular map features are observable after graph optimization and neighbor-link and loop closure refinement steps.

As a qualitative assessment, the generated 3d point cloud of the map is shown in Fig. 11. The noteworthy points are the continuity of visual features, lack of duplicate objects (which, when present, are indicators of poor loop closure performance), and how well the apparent map aligns with the ground truth simulation environment as seen in Fig. 9.

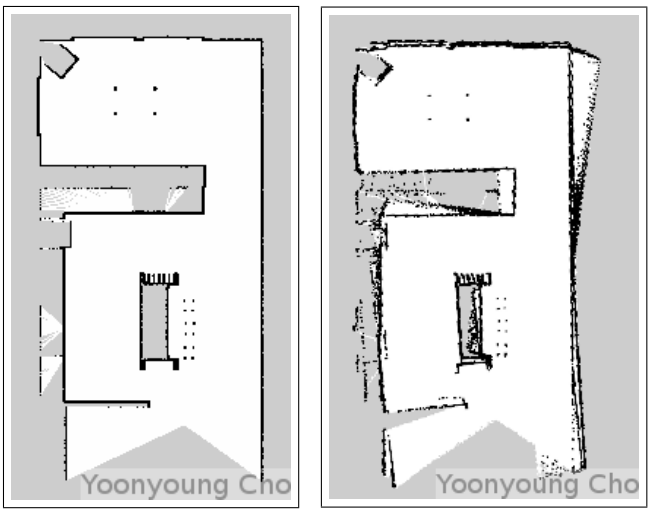
6.1.3 Hector 2D Mapping

As an addendum to the RTABMAP-based SLAM algorithms, the *hector_slam* package was also evaluated on the default environment, which does not take visual features into consideration during the mapping process of the robot.



(a) Perspective view (b) Top-down view

Fig. 11: RTABMap-generated map of the kitchen environment.



(a) Hector-Slam Results (b) RTABMap Results

Fig. 12: Final Map output comparison on the supplied environment; while the large features are similar, significantly higher quality map is generated with hector_slam.

What's intriguing about the generated map and the localization performance is that the produced map is observably cleaner and higher-fidelity to the environment than the RTABMap based solution.

6.1.4 Olin Oval

The Olin Oval environment is a *rectified* outdoor environment, where the wall-like features are less prominent and there are fewer depth features to track in general apart from the terrain information. The textures on the meshes are relatively low-resolution, which poses difficulty on RTABMap as the features to track are not very well-defined. Furthermore, the map is relatively large despite the fact that it is a 35%-scaled version of the actual locale.

However, the performance of RTABMap on this environment was quite remarkable, as seen in Fig. 15; there exist little to no distortion in the final produced map, and no

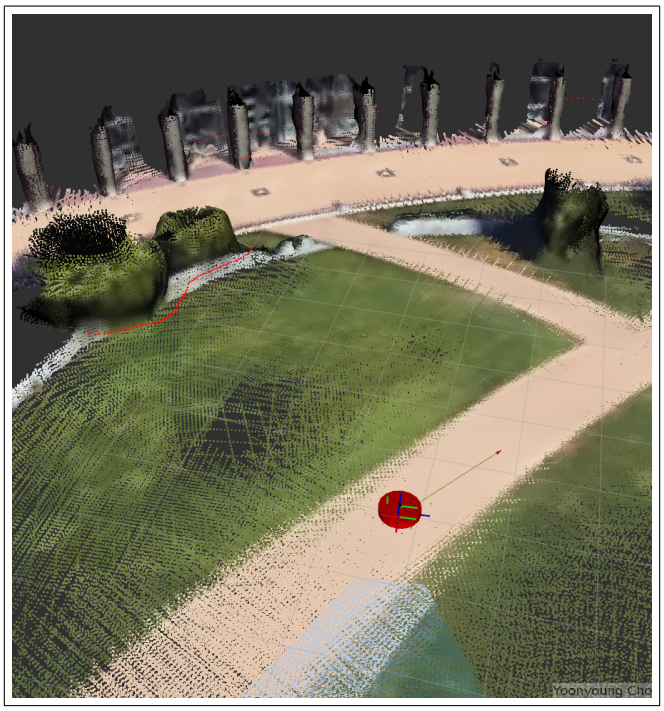
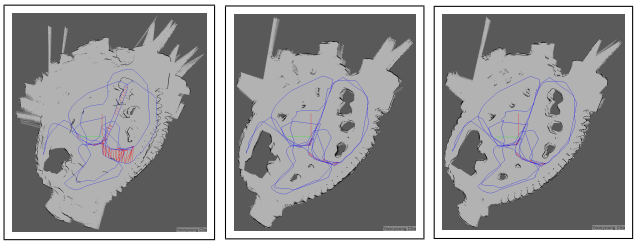


Fig. 13: Close-up view of the robot during the mission, visualized atop the reconstructed point cloud from RTABMap.



(a) Input (b) Optimized (c) Refined

Fig. 14: Visualization of progressive improvement on the map trajectory.

prominent artifacts from un-detected loop closures can be found.

For completeness, the two-dimensional map of the environment is also presented, while the non-linear features of the environment render the visualization harder to interpret. One noticeable feature of the generated map is the lack of apparent *bleed-through* features that often arise from incorrect loop closure incidents.

6.2 Localization

Motivated by the success in the mapping procedure, localization performance on the Olin Oval environment was also evaluated with a similar heuristic from the previous AMCL project, where the positional and rotational error at each synchronized frame is computed for an estimate of localization error over time.

One critical distinction here is that the plot represents the deviation from the inferred relative coordinates based on the initial localized points, in order to resolve the discrepancy

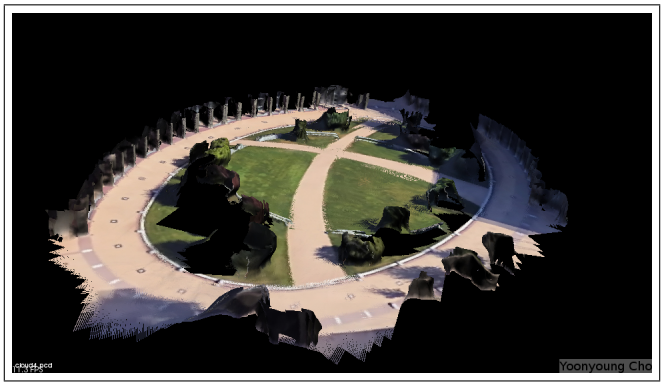


Fig. 15: RTABMap full reconstruction results in the *Olin Oval* environment.

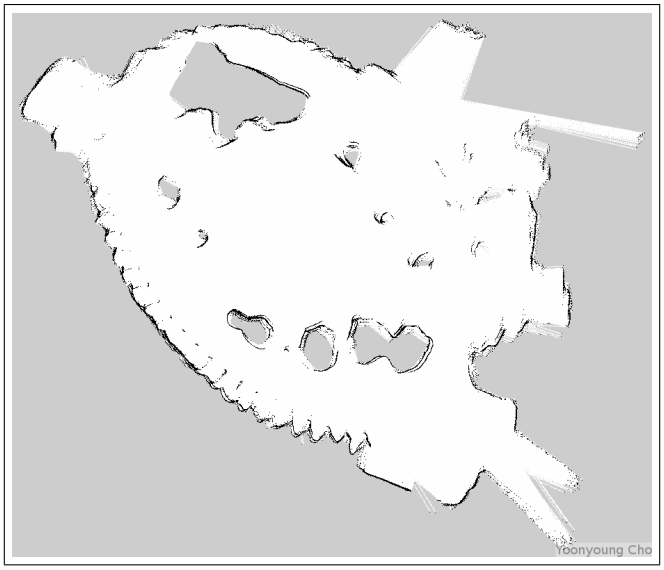


Fig. 16: Two-dimensional occupancy grid abstraction of the *Olin Oval* environment generated by RTABMap during runtime.

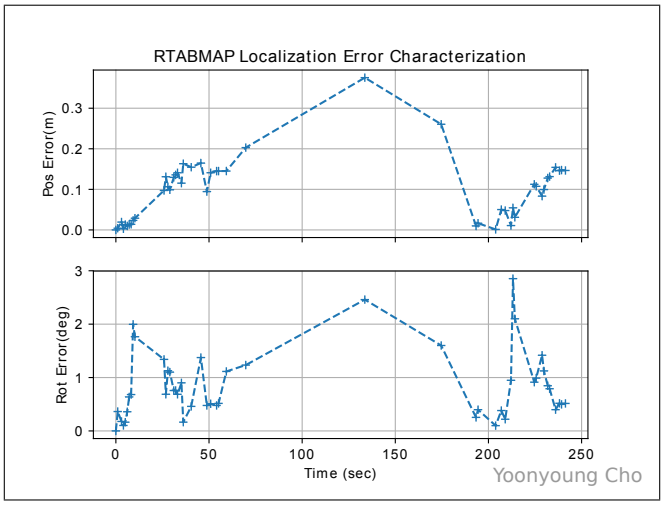


Fig. 17: Estimated deviation of localization over a 4-minute long trajectory on the custom environment. Cross-marks represent data points, and the dotted lines are extrapolated estimates in the time-frame.

where the origin of the RTABMap-generated map and the simulation map are not coincident.

The plot indicates high-quality localization over time, where the maximum error in a fairly large map of about 30m in each direction is only 0.3m; the rotation error also stays within a controlled region of within 3 degrees maximum deviation.

7 Discussion

Overall, RTABMap-based SLAM algorithm on RGBD images and laser-scan data was able to demonstrate compelling performance in both simulated environments. One aspect of the output maps that's intriguing is the high apparent quality of both the 3D and 2D map generated on the *Olin Oval* environment, which was intended to be more difficult considering the relative size, sparsity of features and the low-resolution, blurry texture on the object mesh.

Furthermore, a large part of the terrain were either nearly visually homogeneous or presented visual patterns which may have induce ambiguities when discerning between two successive poses. On the other hand, the *Kitchen Dining* environment was significantly smaller, composed of more distinct depth features with well-defined linear geometries surrounding the area at all times.

One hypothesis for the success of RTABMap on the *Olin Oval* environment is the fact at no point during the data collection process the robot was extremely close to a particular obstacle. This is partly aided by the fact the the environment itself is quite large, but one implication of this is that there is no significant instant in which RTABMap completely loses track of correlated features across frame-captures. The largest error-inducing points in the *Kitchen Dining* environments were the corridor between the two rooms which were deprived of visual features in turning, and many of the turns forced the robot to face directly to the flat-colored wall which lacked any visual markers.

8 Future Work

RTABMap demonstrates close to state-of-the-art performance in RGBD mapping and localization tasks, which is where its primary value is drawn. One of the future projects that is currently under consideration is leveraging RTABMap's high quality localization and loop closure capabilities to perform kinematic calibration of robot arms and actuators, where reliable characterization of position can yield robust association between the control inputs and the output behavior of the platform.