

Map My World Robot

Robert Aleck | github.com/mnbf9rca | 21st July 2019

Abstract

In this paper, the author describes the steps taken to implement the ROS RTAB-Map [1] module in a provided and then in a customised simulation workspace provided by RViz and Gazebo, producing a 2D occupancy grid and 3D octomap of each. Good results were achieved in the provided world; the customised world proved more challenging, requiring the addition of significant modifications to meet the success criteria.

Introduction

The author adapted a 4-actuator, planar robot previously developed by the author [2] by replacing the monocular front facing camera with a top-mounted Kinect camera, using the ROS openni Kinect package (http://wiki.ros.org/openni_kinect). A small block was placed under the laser sensor to raise it up and increase clearance for depth perception above the robot and camera.

A simulated environment, `kitchen_dining.world`, was provided. The objective was to spawn the robot in to this world (by convention at the 0,0,0 coordinates, but this location is arbitrary), and navigate the world sufficiently to obtain a 2D occupancy grid, and a visually recognizable 3D map with at least 3 identified (global) loop closures.

The author was then required to generate a second environment and perform the same task.

Background

SLAM

As noted in the author's work submitted for a prior project [2], SLAM is the process of a robot incrementally constructing a worldmap simultaneous to localization within that map, to allow the robot to develop a globally consistent estimate of pose with respect to environment. Accurate pose estimation is a fundamental requirement for autonomous robotics, allowing robots to operate in previously unseen environments (such as with self-driving cars, or unmanned inspection drones for hazardous areas), and providing benefits in tasks like path planning, where priority can be given to previously unsearched areas, or to previously traversed paths therefore assumed to be "safe".

SLAM is a challenging problem, as the ultimate accuracy of the worldmap depends heavily on the accuracy of localization, and localization requires an accurate world-map, however, several techniques exist to achieve reliable outcomes in reasonable amounts of time, and with reasonable compute power, falling in to five broad categories:

- Extended Kalman Filters (EKF)
- Sparse Extended Information Filters (SEIF)
- Extended Information Form (EIF)
- FastSLAM

- GraphSLAM

SLAM solutions perform either *online SLAM*, or *Full SLAM*:

- Online – maintains an estimate over the map (which is generally assumed to be static in the algorithms outlined here), and estimates only the present robot pose
- Full SLAM – maintains a posterior over both the map and the entire robot path

Extended Kalman Filters (EKF-SLAM)

EKF SLAM makes use odometry or feature detection to estimate current robot pose (Online SLAM) in an iterative 5 step cycle:

1. State prediction: given input controls, the current pose of the robot is estimated
2. Measurement prediction: given the best estimate pose prediction, what measurements are expected to be observed
3. Measurement: capture sensor readings
4. Data association: associate measurements to specific landmarks based on measurement proximity to the landmark
5. Update: update the estimated pose given the measurements and measurement predictions.

EKF-SLAM suffers from two key problems [3]:

- Computation time is non-linear ($O(n^2)$ for n landmarks) with the number of landmarks to be measured, and
- The use of a techniques to linearize inputs leads to propagation of covariance when linearizing an inherently non-linear model (e.g. many real-world systems)

Sparse Extended Information Filters (SEIF-SLAM)

An Extended Information Filter maintains a feature matrix representing the strength of link between features (e.g. measurements over time of a feature relative to the robot's pose). On observation, it can be seen that most links in the matrix are, once normalized, close to zero, and there are a relatively small number of elements in the matrix with strong links – in short, the normalized matrix is *sparse*.

An online algorithm, SEIF-SLAM exploits this to maintain a sparse information matrix representing only strong links between nearby features. As the matrix is sparse, updates occur in constant ($O(n)$) time [4].

Extended Information Form (EIF-SLAM)

Although similar to EKF-SLAM, EIF-SLAM is not incremental, calculating a full SLAM posterior on each execution. Because the entire dataset is known at the point of calculation, it's possible to improve the linearization of the model, and as the EIF-SLAM iterates map construction, it is less susceptible to errors, and produces maps of higher accuracy than EKF. Finally, EIF-SLAM be applied to maps of many orders of magnitude larger than EKF-SLAM [5].

FastSLAM [6]

FastSLAM first estimates the robot's estimated pose (as a distribution over all possible robot paths) using a modified particle filter similar to that used in Monte Carlo Localization. This path estimate is then used to estimate landmark location relative to the robot's path using a low dimension EKF to calculate a full posterior over the position of the measured landmarks over the

robot's entire estimated path. FastSLAM resolves both the *online* estimation challenge (by estimating an instantaneous pose), and the *full SLAM* problem (by estimating the robot's entire path). By storing the resultant probability distribution in a tree-based data structure, can operate in $O(M \log K)$ time, where M is the number of particles in the filter, and K the number of landmarks measured. Further iterations (FastSLAM 2.0) reduces the number of particles to attempt to make

GraphSLAM [7]

A highly efficient full SLAM algorithm, GraphSLAM generates a graph comprising of nodes representing robot poses at different points in time, and edges representing information constraints between those poses (such as a environment or odometry measurements), recorded as a probability distribution over the relative transformations between the poses represented by the nodes. The path and map is then solved to determine the configuration that best fits the constraints through a technique like Most Likely Estimation, SGD, or others. The problem of cyclical graphs can be reduced through variable elimination whereby duplicated constraints (e.g. observations of the same landmark at different times) are removed to simplify the network.

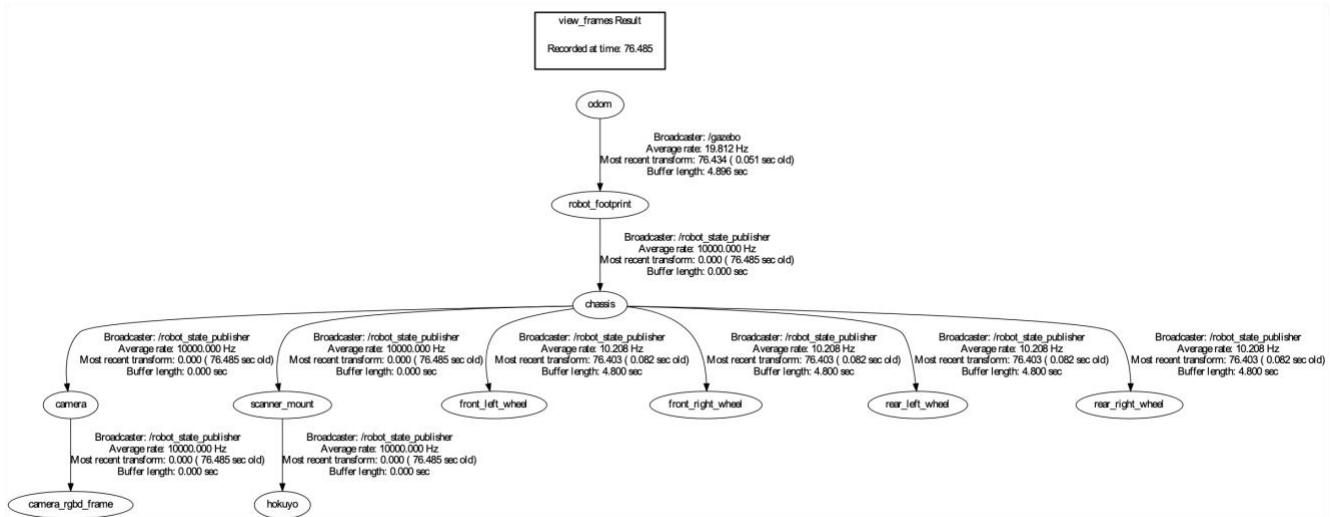
RTAB-Map

Real Time Appearance Based Mapping, or RTAB-Map, uses information from odometry and visual sensors to perform localization and mapping. Building on GraphSLAM, rather than detecting landmarks, RTAB-Map encodes visual (including optionally depth) information using bag-of-words to detect global loops and perform loop closure (i.e. identifying images which represent areas of the map which the robot has seen before). As the algorithm operates over long periods of time, the number of images which must be matched increases, causing a linear increase in processing time, but RTAB-Map uses various graph optimization techniques to ensure that loop detection happens in roughly $O(1)$. RTAB-Map solves both the online SLAM problem and Full SLAM.

Scene and Robot Configuration

Robot model

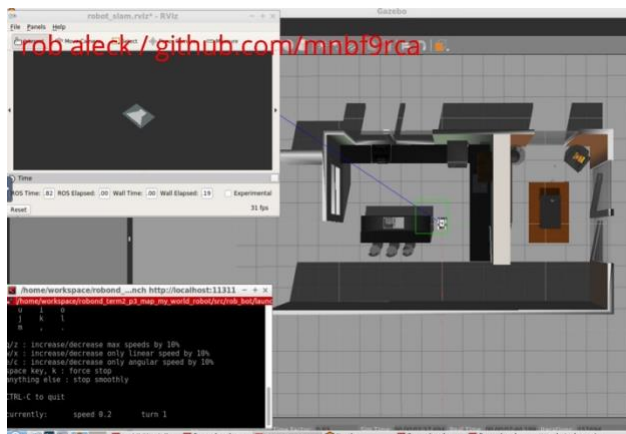
The robot model was based on a previously designed model rob_bot, created for a prior project submission. The robot was modified to remove the front-facing monocular camera and replace it with a top-mounted front-facing RGBD camea modelled on a Kinect device powered by Openni2 drivers. The hokuyo laser finder was raised by approximately 10cm to ensure the beams were not interrupted by the Kinect camera. The resulting frames are as follows:



Frame map for rob_bot

Custom scene generation

A training scene, called `kitchen_dining.world`, was provided, and is shown in **Error! Reference source not found.** as seen on initial launch. A custom scene was generated using an out of the box GazeboSim model called "cafe". The model was further extended by adding furniture and people. During tuning (see results), it became apparent that the floor texture was highly repetitive across much of the model space. The first attempt at resolving this was to include additional wall textures, but this was unsuccessful and so additional textured shapes in alternative directions were added to the floor to provide additional visual differentiation for word selection. This world is shown in **Error! Reference source not found.** and provided in the worldfile



`kitchen_dining.world` viewed from above

`robs_cafe.world`.



`robs_cafe.world` viewed from above

Directory Structure

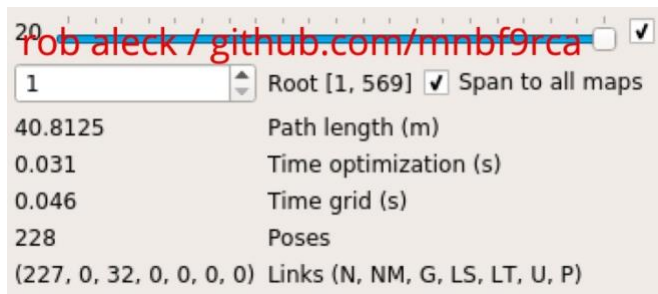
The following diagram depicts the directory structure of the package, which is based on previously used structures and known gazebo/ros best practice:

```
launch
├── mapping.launch
├── robot_description.launch
├── rviz.launch
├── teleop.launch
└── world.launch
meshes
├── hokuyo.dae
└── kinect.dae
package.xml
rviz
└── robot_slam.rviz
src
└── teleop
urdf
├── rob_bot.gazebo
└── rob_bot.xacro
worlds
├── kitchen_dining.world
└── robs_cafe.world
```

Results

Provided world

A single pass around the room was sufficient to achieve 32 global loop closures across 228 poses, as can be seen below, and the resulting 2D and 3D maps are clearly recognizable.



rob aleck / github.com/mnbf9rca

20	<input type="checkbox"/>	<input checked="" type="checkbox"/>
1	Root [1, 569]	<input checked="" type="checkbox"/> Span to all maps
40.8125	Path length (m)	
0.031	Time optimization (s)	
0.046	Time grid (s)	
228	Poses	
(227, 0, 32, 0, 0, 0, 0)	Links (N, NM, G, LS, LT, U, P)	

*Summary of loop closures in provided world
from rtabmap-databaseViewer*

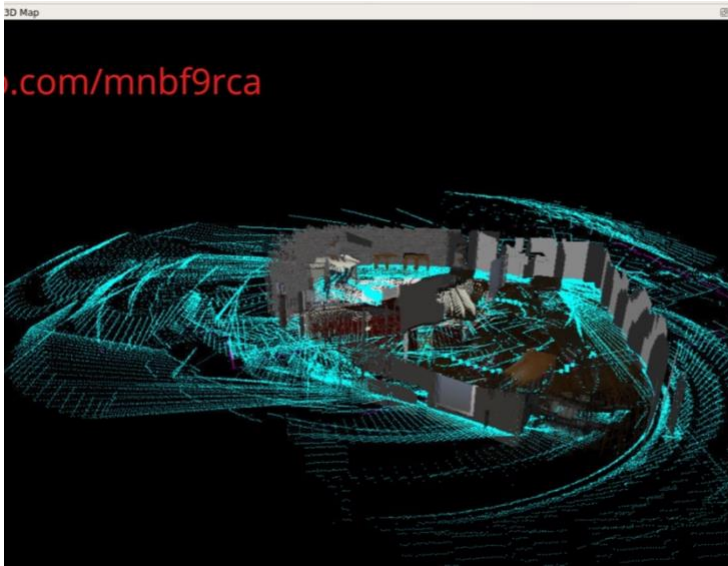


2D map of kitchen_dining.world visualised by RViz

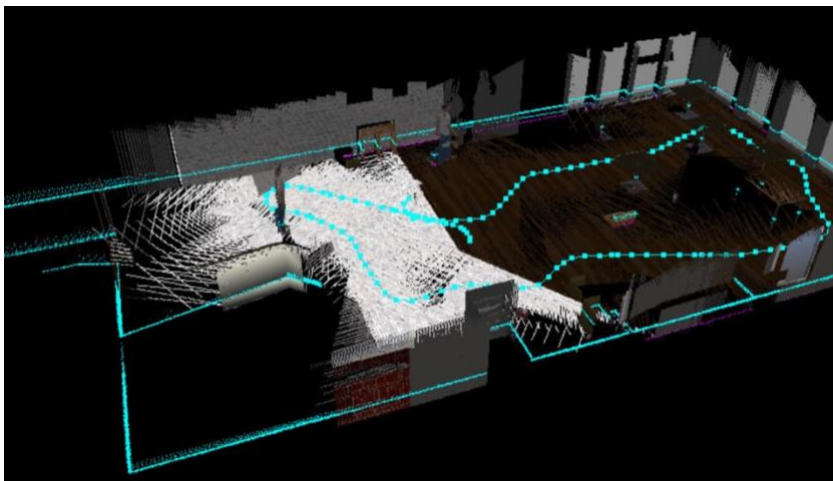


3D map of kitchen_dining.world visualized by rtabmapviz

robs_cafe.world



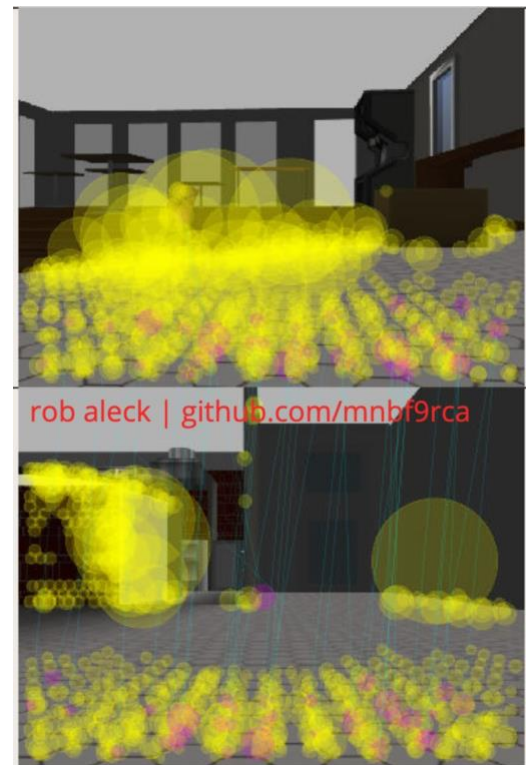
Overlapping 3D map in robs_cafe.world



Recognizable 3D world map in robs_cafe.world

20	rob aleck / github.com/mnbf9rca	<input checked="" type="checkbox"/>
1	Root [1, 203]	<input checked="" type="checkbox"/> Span to all maps
40.6525	Path length (m)	
0.017	Time optimization (s)	
0.131	Time grid (s)	
128	Poses	
(127, 0, 6, 0, 0, 0, 0)	Links (N, NM, G, LS, LT, U, P)	

Summary of loop closures in robs_cafe.world



erroneous loop detection on floor patterns

Discussion

The provided world was mapped relatively simply. Improved loop detection was seen when lowering the `Vis/MinInliers` parameter to 10 in `mapping.launch`'s `rtabmap` node. Very little other modifications or tuning were required, although it was required to enable the ROS `gpu_ray` laser module (http://gazebosim.org/tutorials?tut=ros_gzplugins#GPULaser).

Initial passes in `robs_cafe.world` resulted in multiple, overlapping loop closures and a 3D visual map which contained recognizable features, but clearly included overlapping, repeated frames, as shown in above. Further investigation of the loop detection using `rtabmap-databaseViewer` made it obvious that floor features were being detected. Looking at the floor in the part of the room experiencing overlapping loops, the floor is covered with hundreds of identical hexagons which, even when required word count was increased, were leading to false positives (see above). Adding additional textures to the floor of the café in the relevant areas stopped this erroneous loop detection.

As it was significantly more feature rich, mapping of the `kitchen_dining.world` model performed significantly better than `robs_cafe.world`.

Future Work

Future work would include:

- Using other try other loop closure techniques to see if this reduces false positives in highly repetitive textures.
- Attempting to generate depth data using the depth sensor of the Kinect and removing the laser scanner
- Implementation on a physical robot

References

- [1] M. Labbé, "RTAB Map - Real-Time Appearance-Based Mapping," 2017. [Online]. Available: <http://introlab.github.io/rtabmap/>. [Accessed 21 07 2017].
- [2] r. aleck, "github/mnbf9rca/robond_term2_p2_where_am_i," 2019. [Online]. Available: https://github.com/mnbf9rca/robond_term2_p2_where_am_i.
- [3] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593-598, 2002.
- [4] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *The International Journal of Robotics Research*, vol. 23, no. 7-8, 2004.
- [5] S. Thrun, W. Burgard and D. Fox, "11: THE EXTENDED INFORMATION FORM ALGORITHM," in *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, MIT Press, 2005, pp. 267-268.
- [6] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," *In Proceedings of the AAAI National Conference on Artificial Intelligence*, Vols. 593-598, 2002.
- [7] G. Grisetti, R. Kummerle, C. Stachniss and W. Burgard, "A Tutorial on Graph-Based SLAM," *IEEE Transactions on Intelligent Transportation Systems Magazine*, vol. 2, no. December, pp. 31-43, 2010.