Map my World Project

Kevin Fructuoso

Abstract—The purpose of this report is to apply the robotic Mapping principles in order to generate a perceived mapping of a robot's environment. There are many different methods of mapping that can be implemented. This report focuses on Real-Time Appearance-Based Mapping (RTAB-Map) based on camera feedback from an RGB-D sensors. In this report, a simulated robot was configured with an RGB-D camera and laser range finder and was then configured to generate a map of two different environments. The robot was integrated with Robot Operating System (ROS) packages to implement RTAB-Mapping to create 2D and 3D maps of each environment.

1 Introduction

MAPPING is a core component of many applications in robotics, especially for mobile robots. Mapping is the method for a robot to determine its environment given the robots poses. While it is possible to have compiled and downloaded a map to the robot before deployment, reality prevents this from being a viable solution. At any point after deployment, obstacles around the map may shift or be readjusted by other external forces, rendering the preconceived mapping incorrect. This is the primary reason mapping is a critical feature in robotic applications. A valid map vital to applications that involve motion planning. This report will focus on the use or RTAB-Mapping using an RGB-D camera and laser range finder.

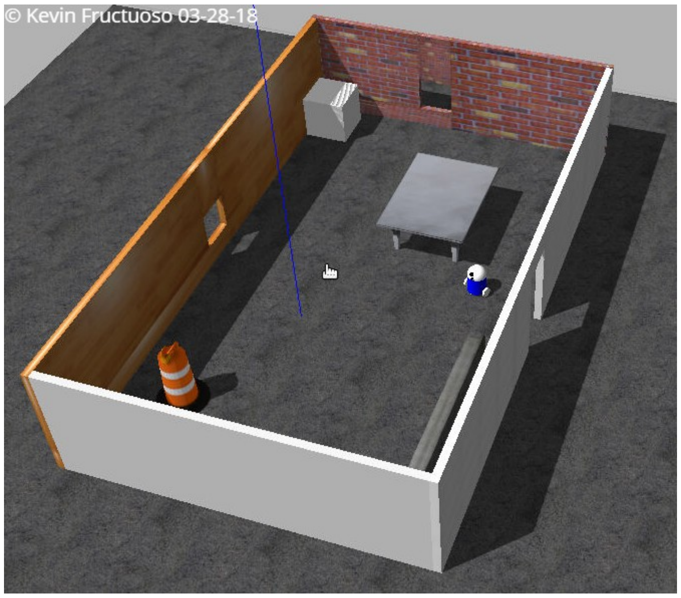


Fig. 1. Custom world with robot

1.1 Project Goal

The goal of this project was to apply RTAB-Mapping in simulation in order to create valid 2D and 3D visualizations of a robot's environment. In this project, two different environments - one given and one created by the author - were mapped, one of which is shown in the figure above.

2 Background

There are many different types of mapping algorithms that can be employed to determine a robots environment. These include Occupancy Grid Mapping, Fast-SLAM, and Graph-SLAM to name a few. Each come with their own pros and cons. This report focuses on the Graph-based SLAM approach with RTAB-Mapping for performing 3D SLAM.

2.1 SLAM Challenges

There are numerous challenges for SLAM algorithms to address. As with any robotics application, large amounts of noise may introduce irreparable error into the algorithm. Specifically for SLAM applications, algorithms will tend to have difficulty with large environments that have multiple areas that appear similar. These produce incorrect consolidations of the created map. Proper tuning of algorithm parameters will mitigate this issue.Additionally, some algorithms may reach memory management issues. This is more prevalent in larger environments. The required memory to accurately map the environment may become costly for the algorithm to maintain. This issue can be addressed architecturally by selecting the most suited SLAM method for the robot and environment. Environments with unique landmarks and smaller spaces are mapped more easily.

2.2 SLAM Methods

Occupancy Grid Mapping is a mapping algorithm that only generates a 2D map of the environment. It implements a Binary-Bayes filter to estimate occupancy of all the grid cells within its vision. This estimate creates a map identifies which cells are likely occupied. This method is limited to 2D mapping applications.

Fast-SLAM solves the SLAM problem using a custom particle filter approach. Using particles,it estimates the map using "known" poses of the robot. It estimates features of the map with Gaussians using Extended Kalman Filters. By using known poses, the SLAM problem is easier to solve. However, it may be less accurate than the next method as no particle may be in the actual location of the robot, causing some error in the mapping.

Graph-based SLAM solves the full SLAM problem by recovering the entire path and map. This allows it to consider dependencies between the current and previous poses.

This means that the algorithm uses all of the information available to reach the optimal mapping solution. A greaty benefit of this method is the significantly reduced requirement of on-board processing. Also, it is much more accurate compared to Fast-SLAM.

2.3 RTAB-Mapping

RTAB-Mapping is a real-time method that uses appearance-based SLAM algorithms with input from vision-based sensor inputs to perform mapping to perform a process called ”loop closure.” [1] This process determines if the robot has ”seen” the location before and adjusts the map accordingly based on the path loop it completed. This concept is critical to performing proper mapping of the environment. As the robot travels to new areas of the environment, the map is expanded. The more areas of the map that have been explored, the more comparisons the algorithm must make to determine whether or not a loop closure should occur. This causes the computations to take longer as the algorithm loses efficiency.

RTAB-Mapping is optimized for large-scale and long-term SLAM by using multiple strategies to allow for loop closure to be done in real-time. Image loop closure occurs fast enough to produce results before the next image is obtained. From this, the algorithm assembles an occupancy grid to generate 2D and/or 3D maps. RTAB-Mapping uses global loop closure where new images are compared to all previously viewed locations. If no match is found, it is added to memory. As the map grows with new locations being added, the amount of time to perform loop closure checks increases linearly. When the time to perform the comparison becomes longer than the acquisition time of a new image, the map starts to become ineffective. However, RTAB-Map implements a special memory-management technique to ensure that the loop closure process happens in real-time.

3 Scene and Robot Configurations

3.1 Robot Model

The robot model chassis is a combination of a cylinder and with a dome on top. The cylinder radius is 0.2 meters wide and 0.4 meters tall. The dome was created with a sphere of the same 0.2 meter radius with its origin placed at the top of the cylinder such that only the top half of the sphere is visually exposed. This robot has two wheels set 0.6 meters apart. It also has two casters to help prevent from tipping over frontwards or backwards. The camera and laser range finder were moved to the dome of the robot. The camera is located (0.15, 0.0, 0.3) meters in the xyz-planes from the origin of the cylindrical chassis. The laser-range finder is located (0.2, 0.05, 0.25) meters in the xyz-planes from the origin of the cylindrical chassis. This model was specified with a weight of 2 kilograms. The robot can be found in figure 2.

During development of the mapping project, it became apparent that the RGB-D camera frame required rotation and translation to properly re-orient the measured point cloud axes for mapping purposes. To accomplish this, a specific transform node was created in the

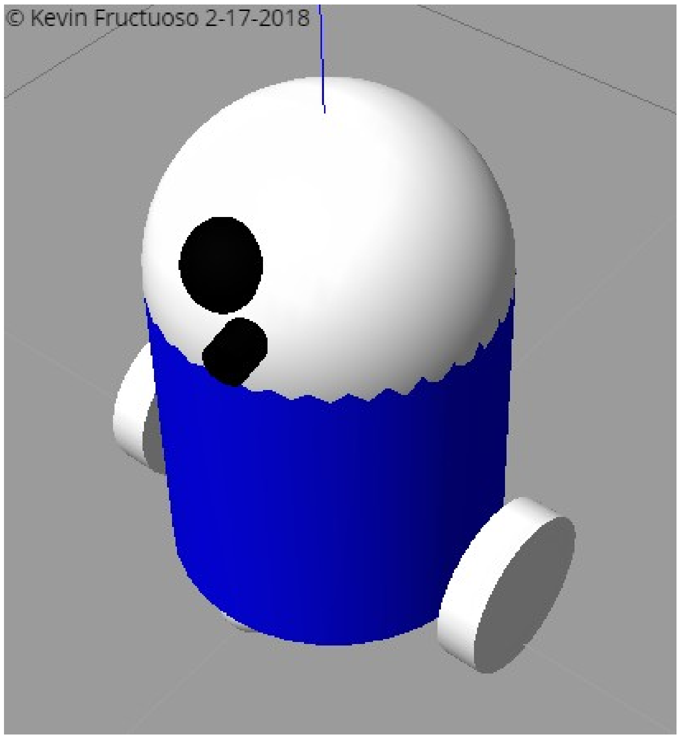


Fig. 2. Custom robot

”robot_description.launch” file. The ”TF” frames can be found in Figure 3.

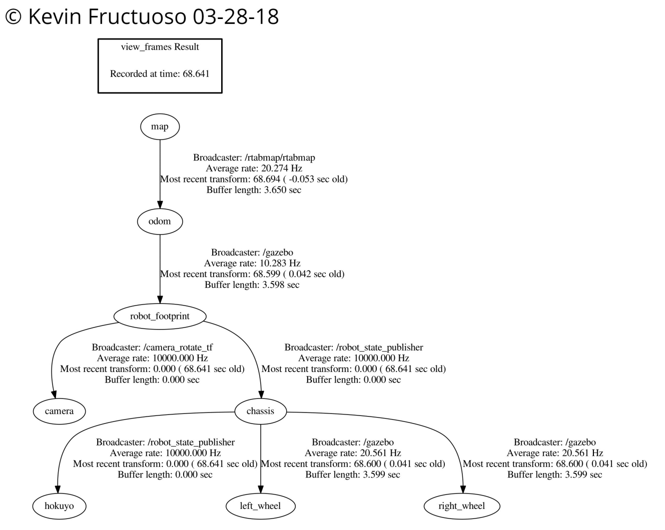

© Kevin Fructuoso 03-28-18
view_frames Result
Recorded at time: 68.641
map
Broadcaster: /rtabmap/rtabmap
Average rate: 20.274 Hz
Most recent transform: 68.694 (-0.053 sec old)
Buffer length: 3.650 sec
odom
Broadcaster: /gazebo
Average rate: 10.283 Hz
Most recent transform: 68.599 (0.042 sec old)
Buffer length: 3.598 sec
robot_footprint
Broadcaster: /camera_rotate_tf
Average rate: 10000.000 Hz
Most recent transform: 0.000 (68.641 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (68.641 sec old)
Buffer length: 0.000 sec
camera
chassis
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (68.641 sec old)
Buffer length: 0.000 sec
Broadcaster: /gazebo
Average rate: 20.561 Hz
Most recent transform: 68.600 (0.041 sec old)
Buffer length: 3.599 sec
Broadcaster: /gazebo
Average rate: 20.561 Hz
Most recent transform: 68.600 (0.041 sec old)
Buffer length: 3.599 sec
hokuyo
left_wheel
right_wheel

Fig. 3. Robot TF frames

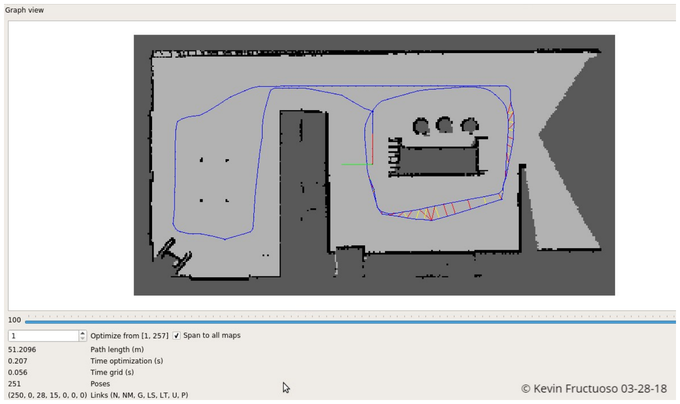
3.2 Scene Configuration

The designed custom world can be seen in Figure 1 on page 1. It consists of a rectangular room with a few objects, windows, and a door leading outside. There is a white box in one corner, a marble table and traffic barrel to navigate around, and a cement jersey barrier in another corner of the room. Each wall was created in a different material with windows or doors to aide in properly identifying which direction (i.e. which wall) the robot is facing. Unique landmarks were placed such that the robot can see an object when turned in any direction as long as it did not directly face a wall. This also supports the loop closure detection of the mapping algorithm.

4 Results

4.1 Kitchen Dining Area

The generated mapping of the Kitchen Dining area resulted in twenty-eight global loop closures. This is quite high which highlights that the mapping parameters are not fully optimized. The 2D map, 2D occupancy grid, and 3D visualization can be found in Figures 4-6.



```
1                      Optimize from [1, 257] ✓ Span to all maps
51.2096                Path length (m)
0.207                  Time optimization (s)
0.056                  Time grid (s)
251                    Poses
(250, 0, 28, 15, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)
```

Fig. 4. 2D map of kitchen dining area

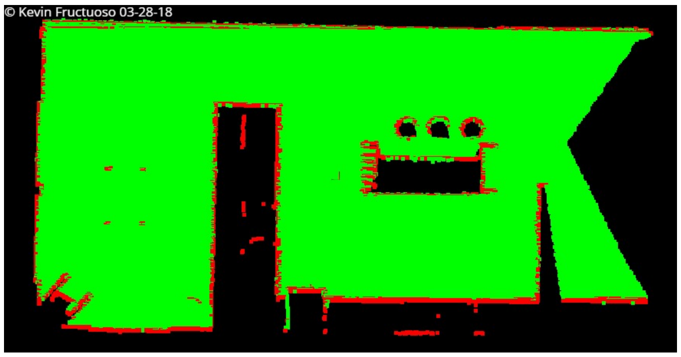


Fig. 5. Occupancy grid visualizing 2D map of kitchen dining area

The robot was able to correctly map the boundaries very well. The missing chunk on the right side is likely due to that side of the room being unbounded. From the the 2D map, one can see where the kitchen island and chairs lay as well as the chair in the corner of the next room over. However, it should be pointed out that the robot failed to map 2D boundaries for the tables in the next room over. This is due to the height of the laser scanner - it was not level with the horizontal portions of the tables and could not map them as boundaries. One way to improve this might be to use the RGB-D camera and use the ROS depthimage_to_laserscan package. The generated 3D map very clearly resembles the external surfaces of the rooms and objects within them. The robot had trouble filling the insides of the table and kitchen island. This is expected as the camera images can only perceive external surfaces of the objects.

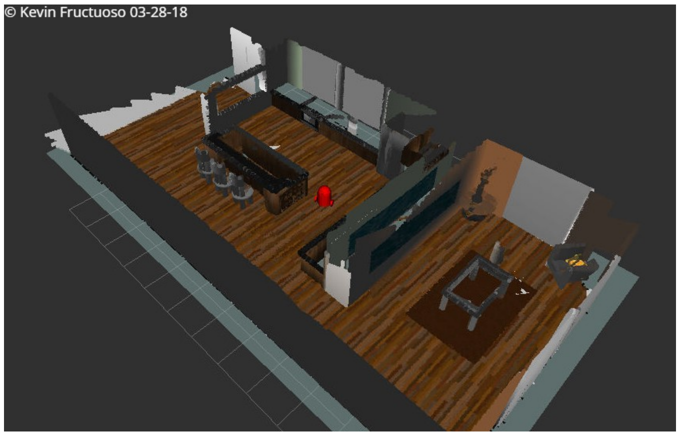


Fig. 6. 3D visualization of kitchen dining area

4.2 Custom World

The generated mapping of the custom world resulted in six global loop closures. The 2D map, 2D occupancy grid, and 3D visualization can be found in Figures 7-9.

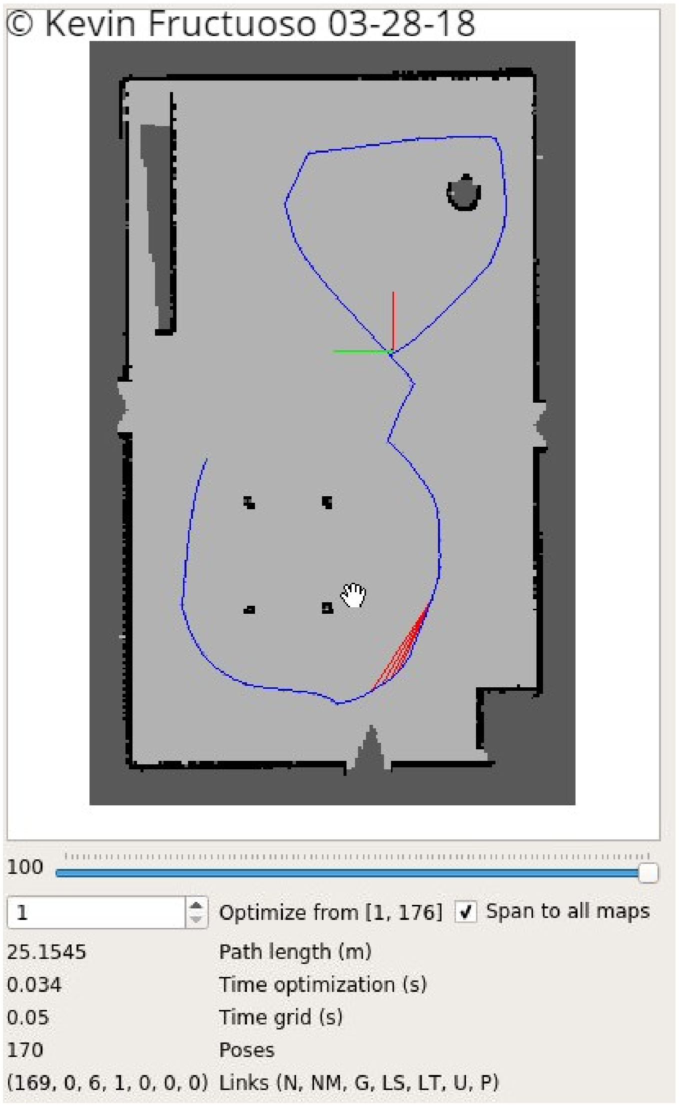
As seen in the Kitchen Dining Area, the boundaries are mapped very well and the flat portion marble table was too high to map in 2D. From the 2D occupancy grid, it can be seen in the upper left corner (near the jersey barrier) that objects in the corners affect ability to generate a totally accurate map in that section. The 3D map shows some difficulties mapping high regions of the walls. This is presumably due to the short path not having scanned the entire section of the wall. Also, due to the objects in the corners, the remaining walls could not be seen.

5 Discussion

The performance of the robot's mapping for each environment have a few similarities. Both were capable of clearly mapping 2D map boundaries, except for the unbounded section of the Kitchen Dining area. The robot was able to generate 3D maps of each world that were very recognizable of the actual environment. However, both struggled with 2D and 3D mapping the tables due to the height and angle of the camera and laser sensors.

Also, the The real-time performance of the mapping was marginal and inconsistent as it was very sensitive to a few specific things. The path taken and the distances from other objects often affected loop closure detection such that it would incorrectly close the loop prematurely. More work needs to be done to tune the mapping parameters to optimize loop closure detection.

One apparent difference between the performances is the completeness of the walls in the 3D map of each environment. The Kitchen Dining Area resulted in full, apparent walls while the custom world had sections missing. This is partially due to failing to scan the entire area methodically. The path taken in the custom world was much shorter and only resulted in one physical loop. It is possible that not all areas of the walls were properly scanned. Future test trials of this mapping implementation should take care to survey the complete area.



```
1              Optimize from [1, 176]  ✓ Span to all maps
25.1545                 Path length (m)
0.034                   Time optimization (s)
0.05                    Time grid (s)
170                     Poses
(169, 0, 6, 1, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)
```

Fig. 7. Occupancy grid visualizing 2D map of custom world

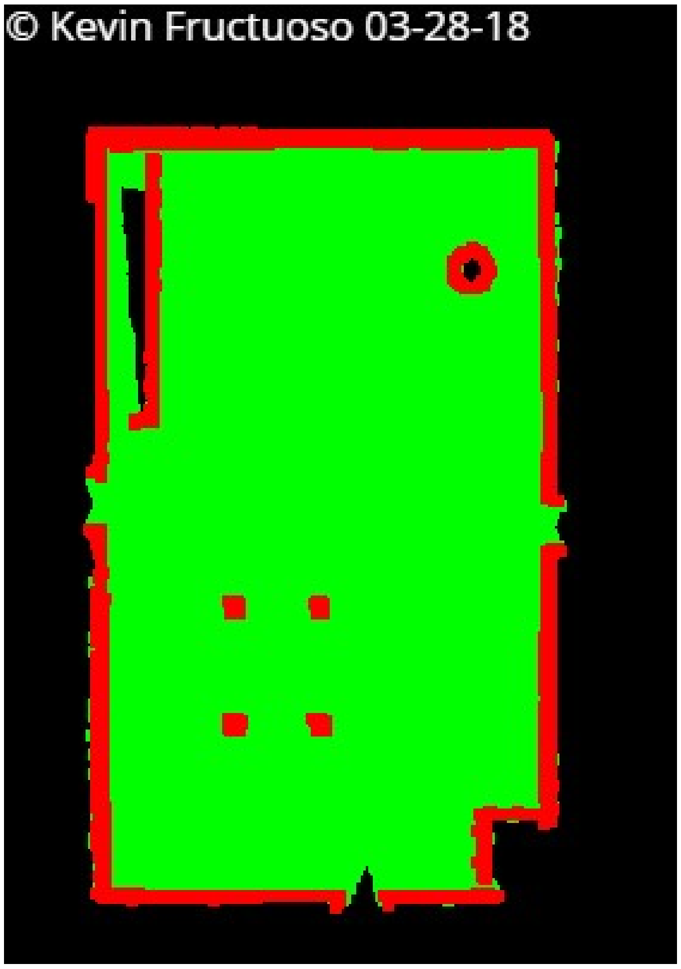


Fig. 8. Occupancy grid visualizing 2D map of custom world

6 Conclusion / Future work

A ROS package [2] was generated and successfully performed 2D and 3D mapping using RTAB-Map. The robot was able to generate quality maps of the kitchen dining room and average maps of the custom world. However, there is still much room for improvement in this SLAM implementation. Further work is required to optimize the SLAM package to be more robust and reliable. Also, the robot may be expanded to perform more autonomous behavior. Newer configurations should be tested to attempt to address shortcomings noted in the Discussion section.

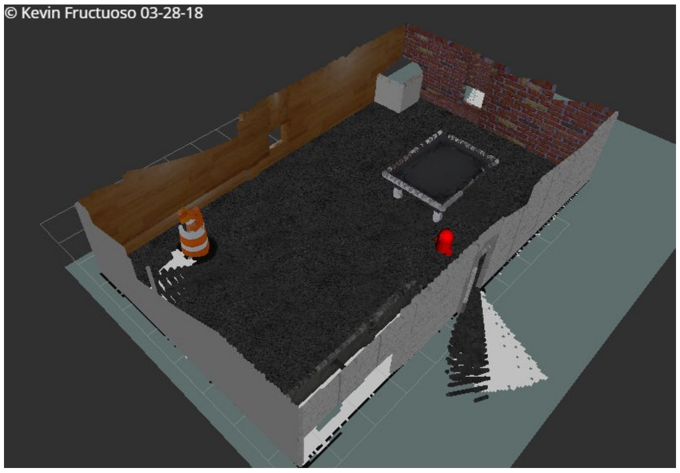


Fig. 9. 3D visualization of custom world

References

[1] IntroLab, “Rtab-map overview,” 2017.
[2] Kevin Fructuoso, “slam_bot mapping project repository,” 2018.