

RoboND Map My World Robot

Kevin James

3 October 2018

Abstract

Simultaneous Localization and Mapping (SLAM) is conducted using the *rtabmap* package in ROS. This project takes two gazebo environments without maps and uses an exploring robot with SLAM to discover the occupancy grid and map while simultaneously localizing. The goal was to demonstrate loop closures showing that the algorithm is able to recognize locations that it has encountered previously.

Introduction

Robots often operate in *a priori* unknown environments. When this is the case, if it wishes to operate effectively, a robot must both map its environment and localize itself within the environment. This is the domain of SLAM (Simultaneous Localization and Mapping) algorithms. The problem is complicated by the robot's need to know its relative location during the mapping process in order to construct an accurate map. The robot does not know its location but its control inputs enable computation of where it should be relative to where it started. Since the effect of controls is statistical, the estimated location accumulates error over time. Additional information is needed to close the loop and correct the error. One solution uses optical feature recognition and distance. Of course, since the map is not known in advance, a camera collects data as the robot explores and later correlates images with previously seen features. Distance to detected objects can be determined using an RGBD camera or LIDAR.

Background

In contrast to structured environments such as product shipping warehouses, for which maps can be constructed in advance, most environments into which a robot might be deployed do not have pre-existing maps. Because of this, autonomous robots require the ability to construct their own maps in real time. SLAM algorithms solve this problem, enabling general purpose autonomy. This project constructs 2D and 3D maps of an unknown environment that could arise in a context such as a robot vacuum cleaner for a home. The algorithm used here for this purpose is *rtabmap*, an implementation of GraphSLAM that solves the *full* SLAM problem: it recovers the entire path and map, as opposed to *online* SLAM algorithms that estimate only the current pose and the map.

The graph constructed by the algorithm links nodes with edges corresponding to the robot's *motion* and corresponding to *features* sensed in the environment from specific poses. *Motion* between poses has expected value given the control inputs, and noise. Feature detection has both noise and the requirement to associate known features using *loop closure*. Graph edges between poses and features both constrain the solution, and minimizing the set of constraints produces the most likely actual path and map.

GraphSLAM improves on FastSLAM, which uses a particle filter and Extended Kalman Filter. Each particle contains a hypothetical pose and map. EKF is used for feature association, and feature measurements determine the particle weights. Since particles are expensive, some work on FastSLAM has involved reducing the number of particles, but this reduction may result in no particle that accurately estimates the pose and map.

Current algorithms, including GraphSLAM, face the challenge of non-linearity in robot motion. As in other contexts such as EKF, this limitation is addressed by linearizing around the operating point.

Scene and Robot Configuration

The robot used to scout the environments is the *udacitybot* from the previous project, updated with an RGBD camera. The robot also maintains its Hokuyo laser range sensor. The customized gazebo world is based on a gazebo cafe model. Added to this open-roof building model are multiple objects that are intended to be differentiable so that the robot can establish non-aliasing correspondences during the localization and mapping process.

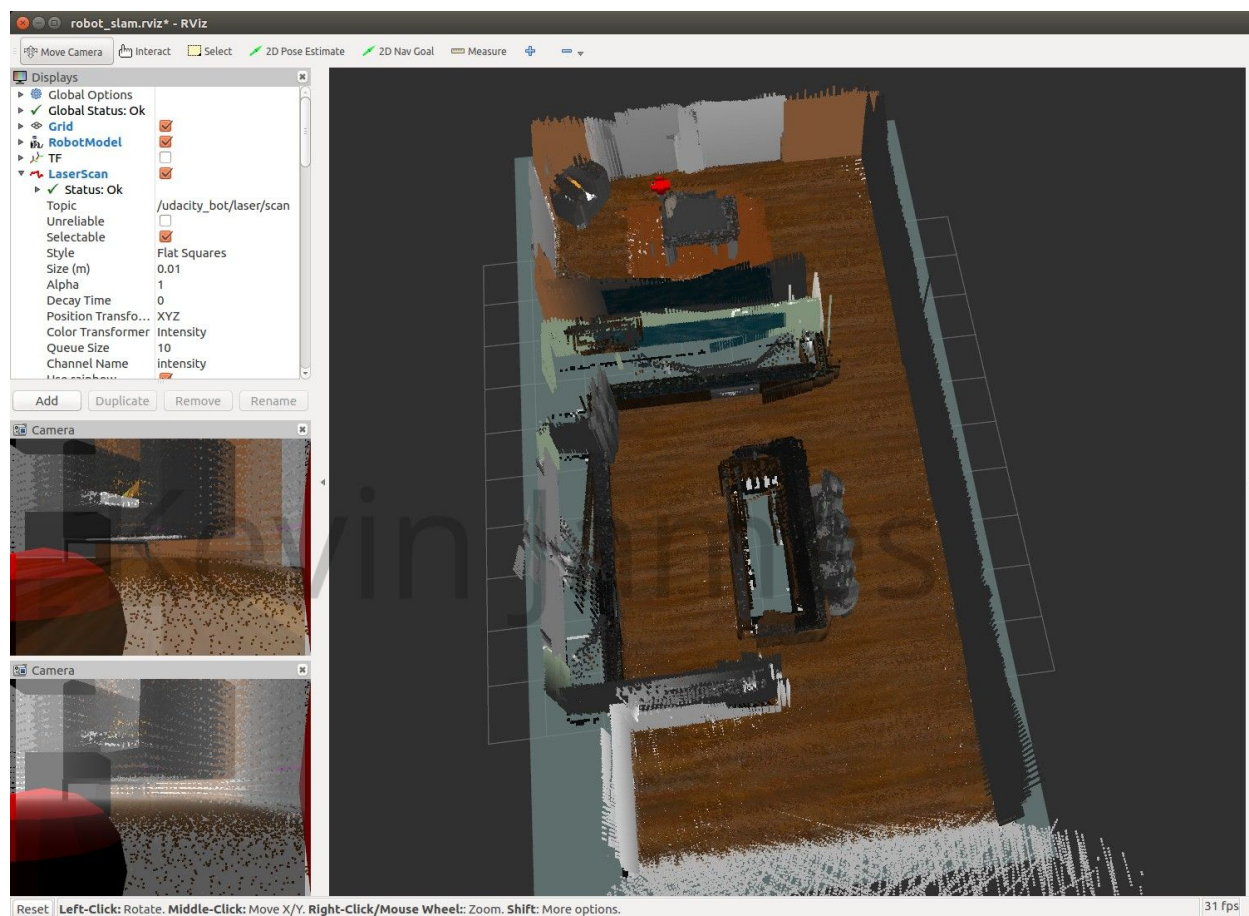
The package structure for launching the world and robot in ROS are derived from the previous project. Added are the launch files for mapping and teleop. The basic files were provided for the project, but some topic name changes were required to connect the laser and camera sensors and TF frames correctly.

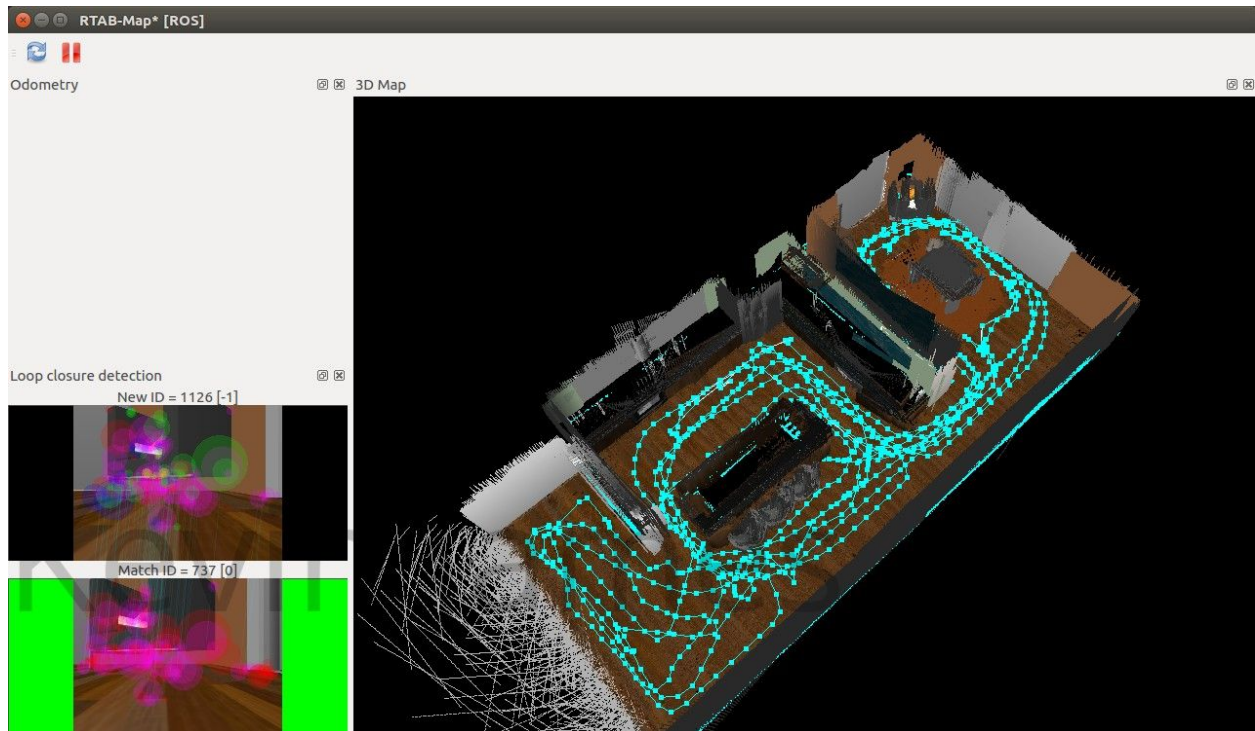
Results

Kitchen Dining World

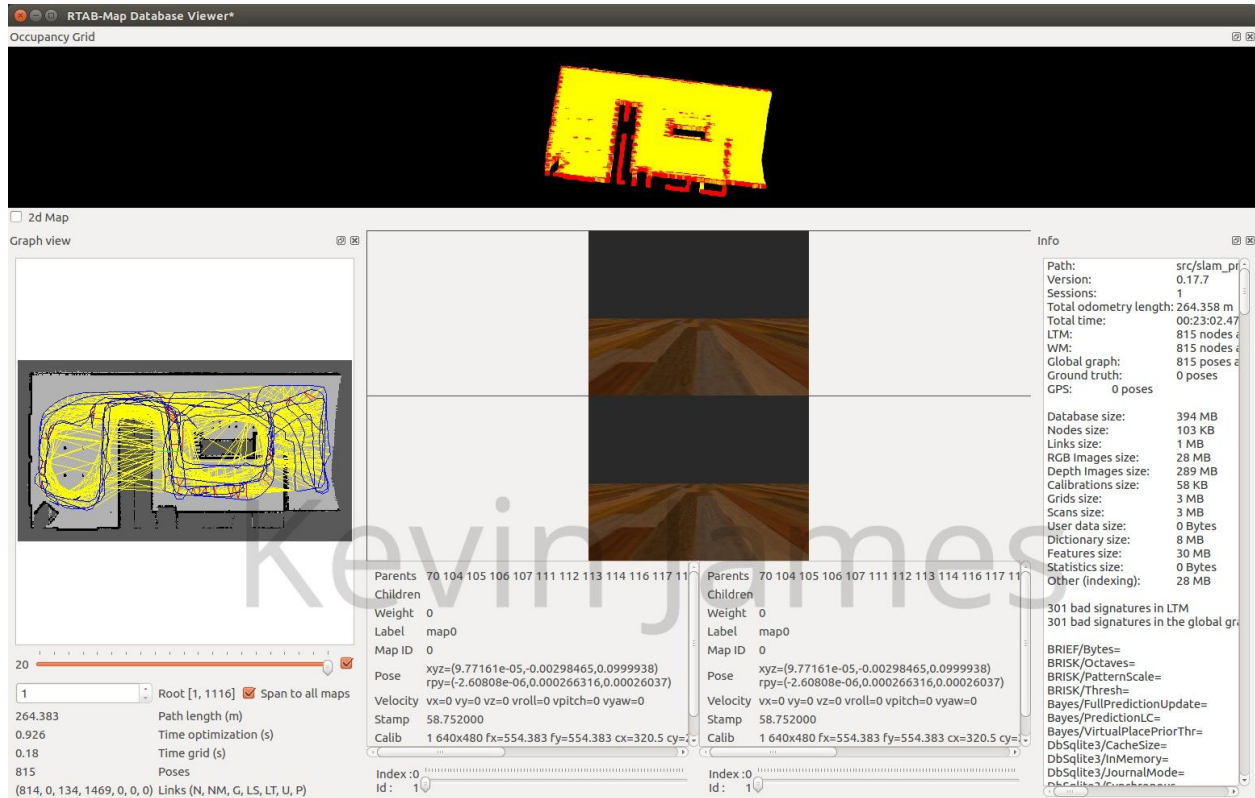
Shown below are images of the Udacity world mapping, occupancy grid, path, and loop closure. Note that images may have been generated from different explorations of the environment.

The image below shows the 3D map of the world in rviz. Following is the map and path shown by the *rtabmap* visualizer.

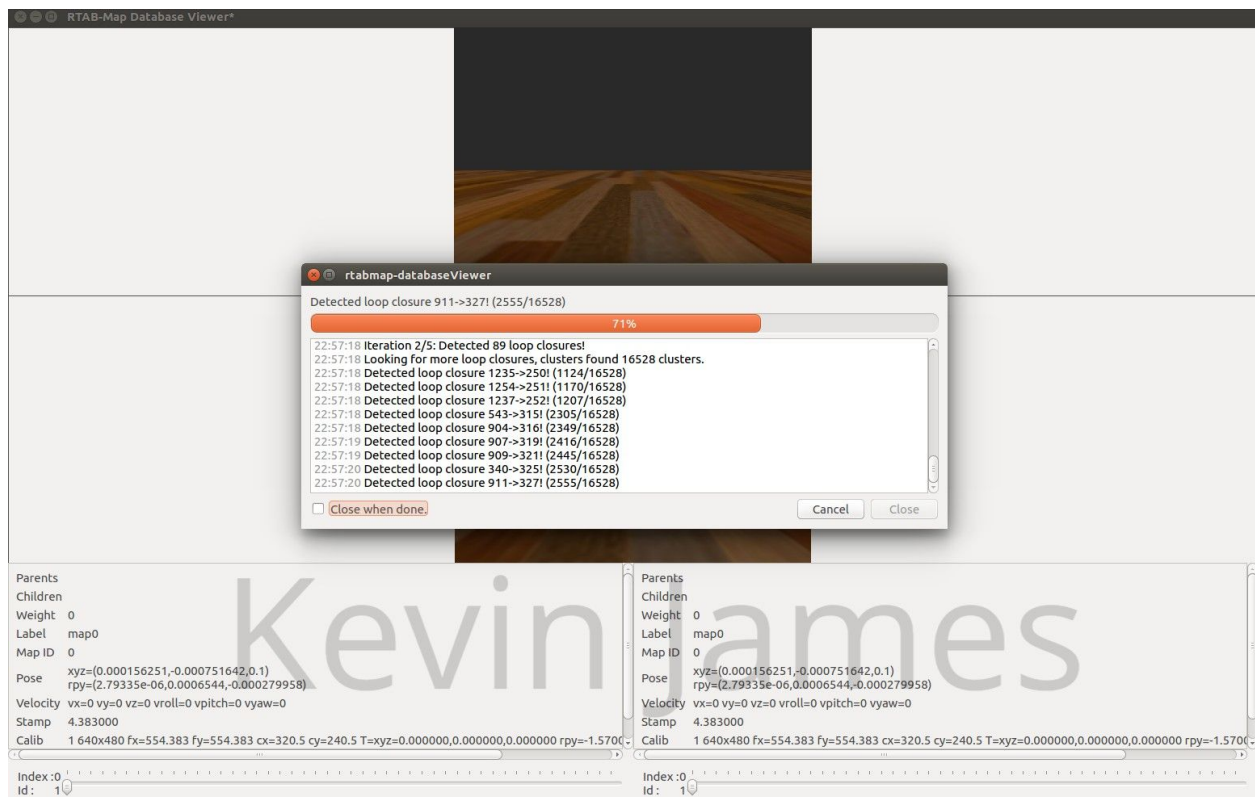


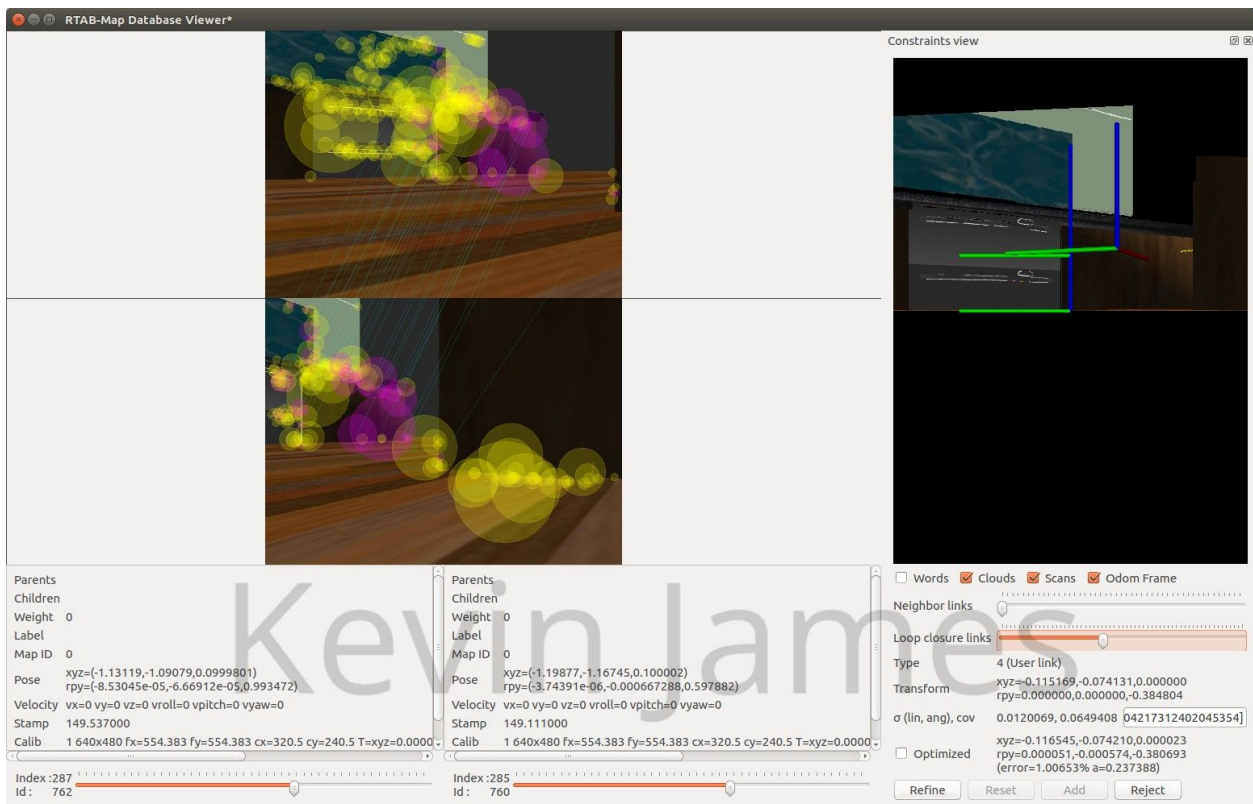


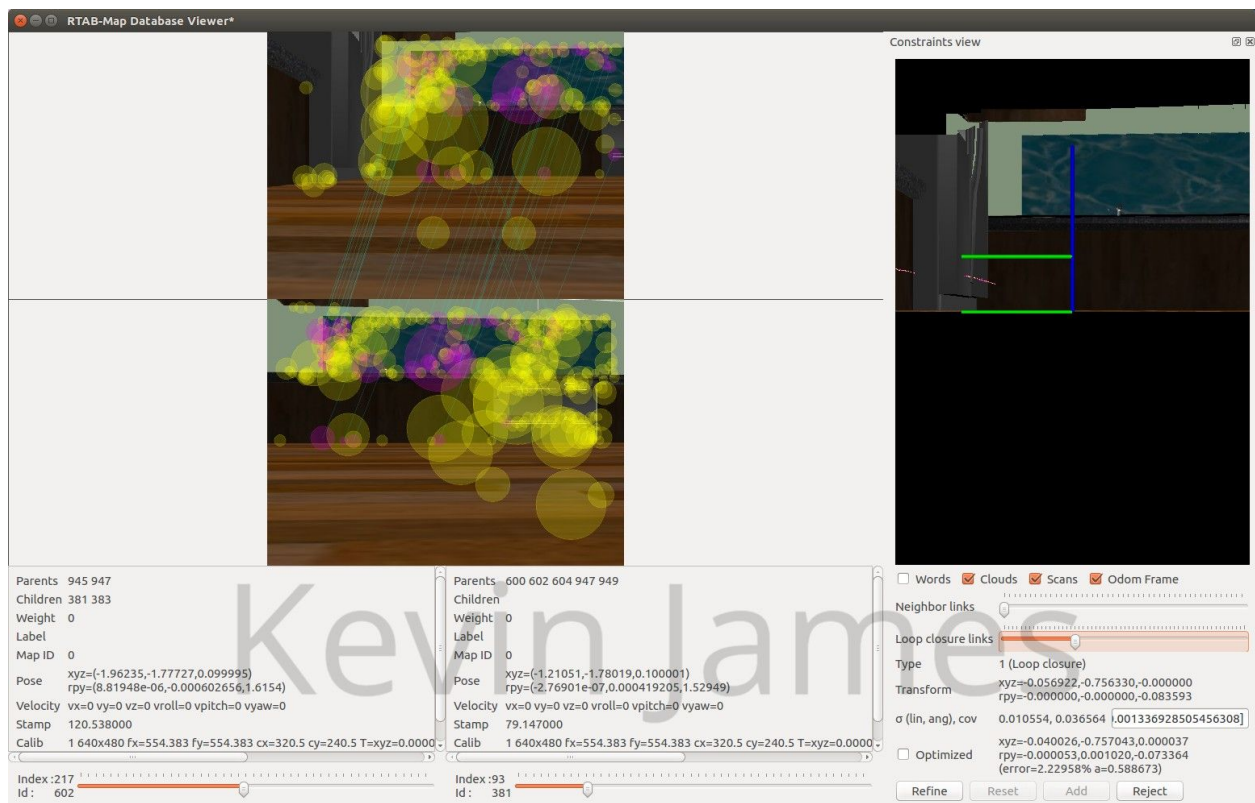
The occupancy grid and graph view are shown in *rtabmap-databaseViewer*.



The following images show loop closures detected in *rtabmap-databaseViewer*.



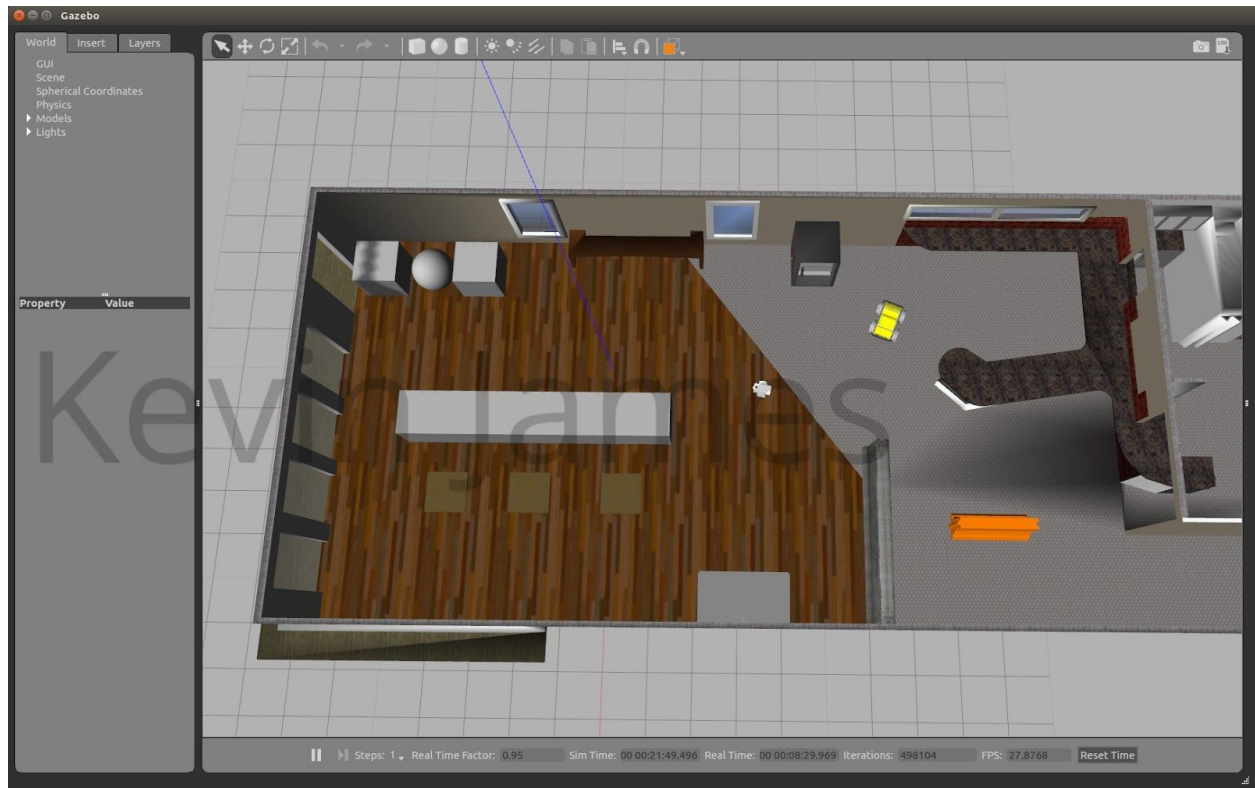




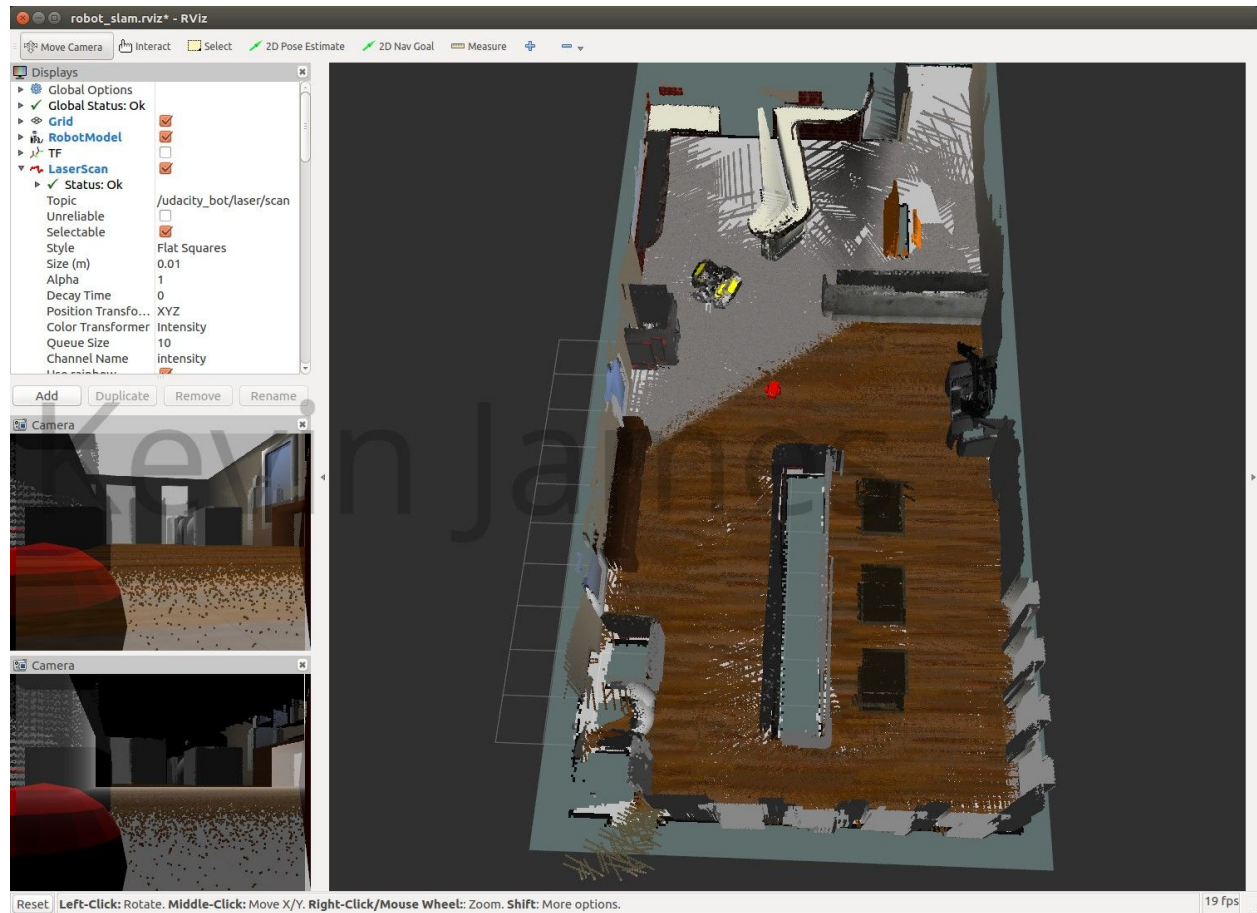
Custom World

The images below show results for the customized world. Rtabmap_viz is used to show the loop closures because in this case Reg/Strategy = 2 (ICP and Visual) was required to get good correlation results. With Reg/Strategy = 0 (Visual) the map was severely corrupted and the algorithm was clearly unable to get consistent localization.

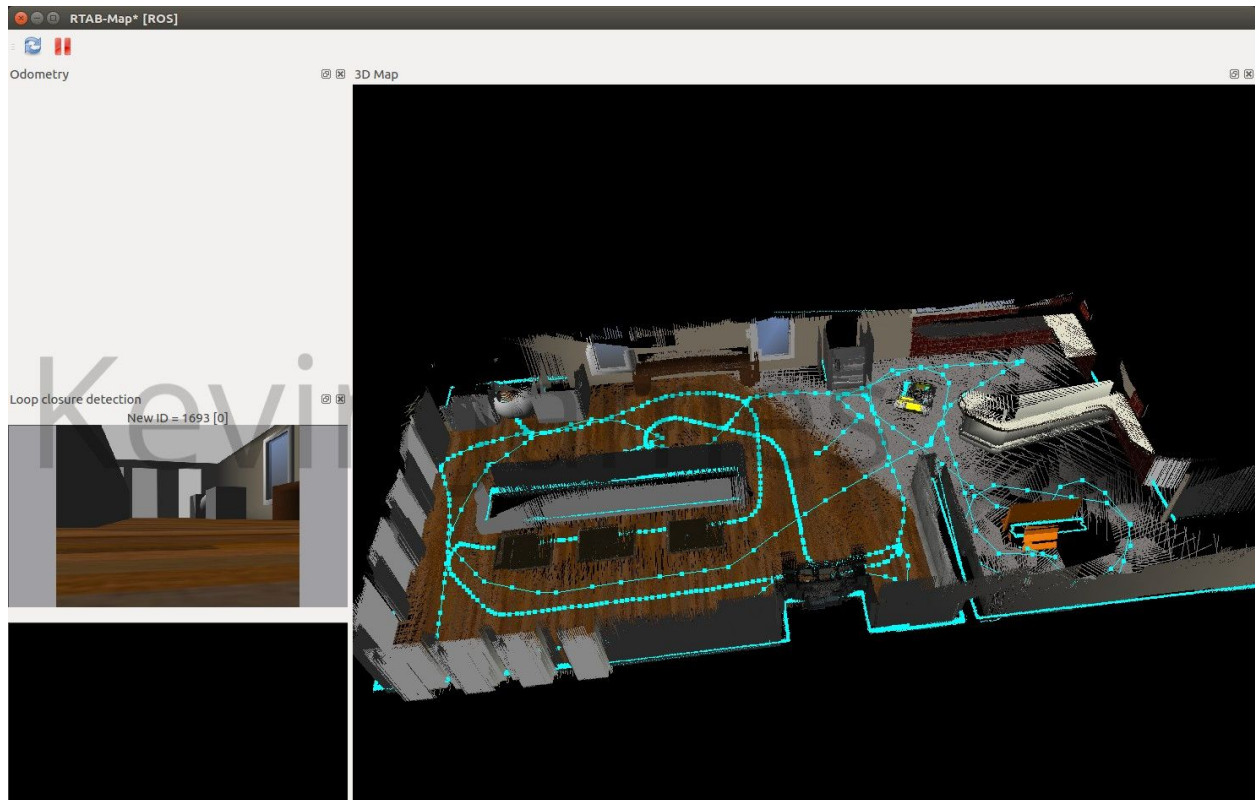
The custom gazebo world is shown in the following image.



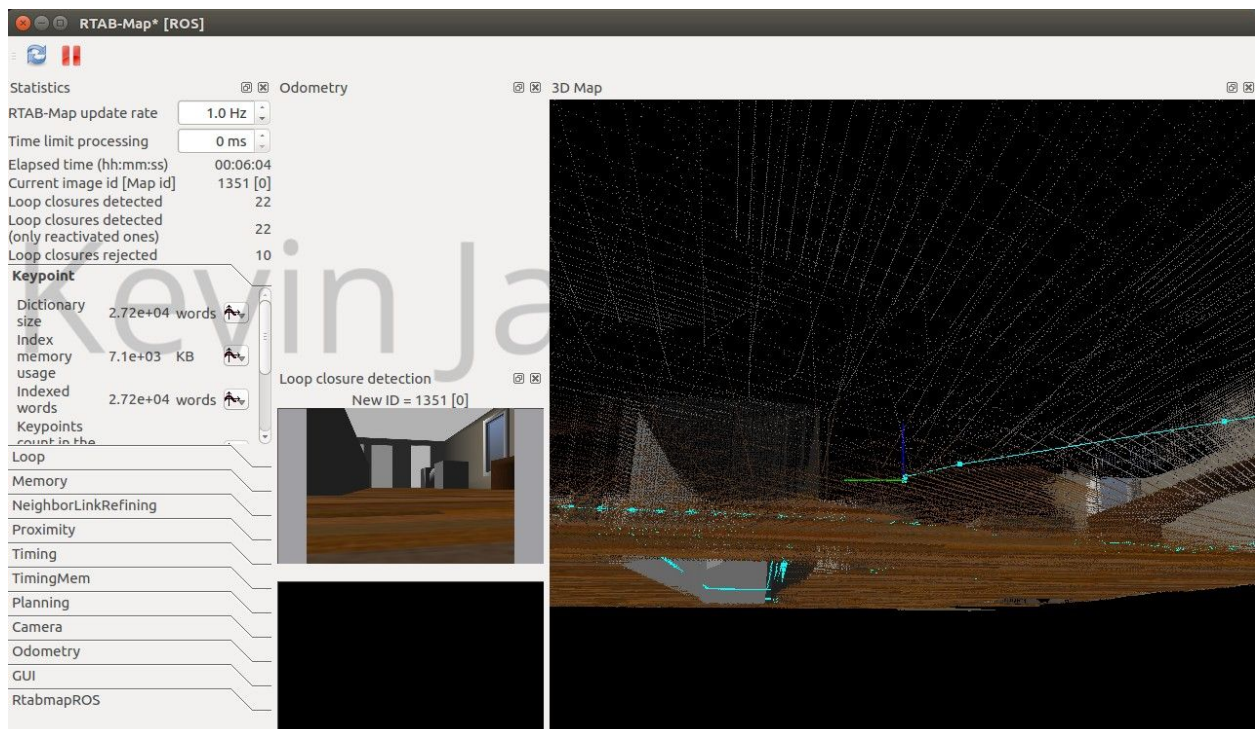
3D map for the custom world is shown in rviz in the following image.



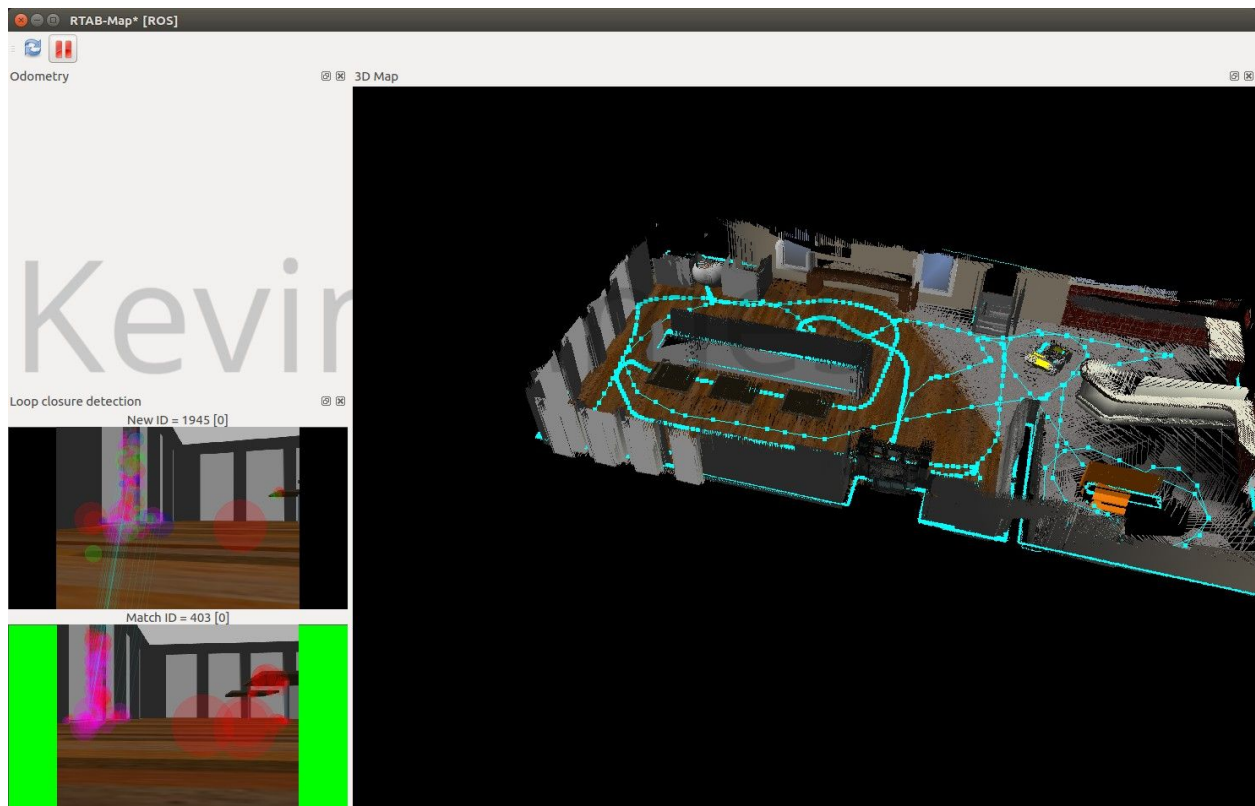
The path is shown in *rtabmap_viz* below.

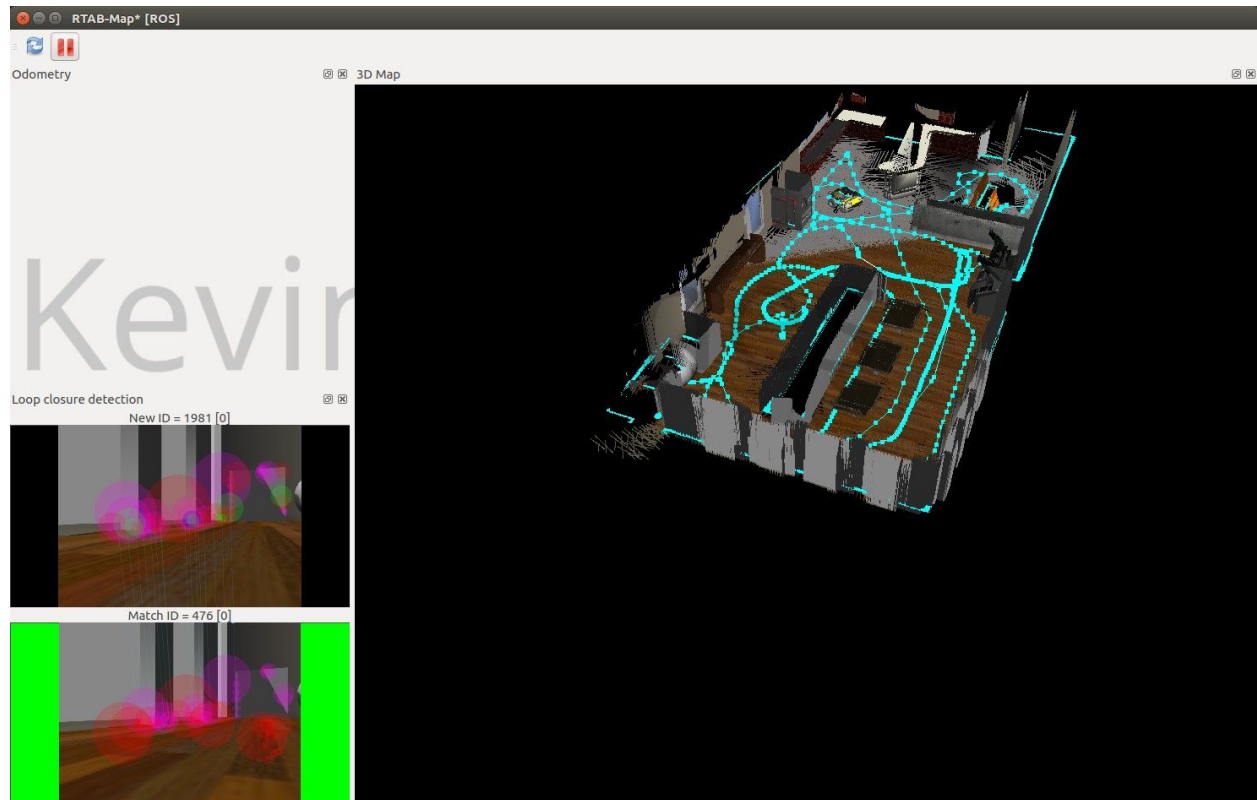


Statistics column shows *Loop closures detected* (22).



The following three images show loop closures detected live on *rtabmap_viz*.





Custom world occupancy grid is shown in *rtabmap-databaseViewer* below.



Discussion

There were four problems encountered on this project. First was likely caused by a missed setup step. Laser scans were not detected on the Udacity world unless the sensor orientation was incorrect. Absence of laser scans resulted in poor map quality. While attempting to observe loop closures, **Reg/Strategy** rtabmap param was changed to 1 (ICP) to make use of the laser scan. It later turned out that ICP is not supported in *rtabmap-databaseViewer* as 0 loop closures were reported (<http://official-rtab-map-forum.67519.x6.nabble.com/Cannot-detect-more-loop-closures-offline-td4777.html>). Setting this parameter to 2 (ICP and Visual) produced good maps, but attempting to detect loop closures caused the database viewer to crash. Thus, loop closures shown in the database viewer for the Udacity world used Visual only (0).

Creating the custom world involved some learning about gazebo. First issue was that only some of the objects are static, and others can be moved by contact with the robot, or apparently can just shift around for no reason. Mapping performance was mediocre with the world shifting. The final issue was probably due to inadequate visually distinctive features in the custom world. Mapping performance was quite bad using Visual loop closure. Changing this parameter back to ICP and Visual resulted in marked improvement, and this setting was used for the results presented. Because the parameter setting crashes the database viewer, loop closures were shown in the visualizer instead.

Aside from the problems mentioned, *rtabmap* produced good occupancy grids and 3D maps with modest required number of passes through the environment. However, while the Kitchen and Dining world was amenable to mapping with visual correspondence, behavior in the custom world demonstrates limitations to using only visual information for loop closure. This world likely has more ambiguous visual features and the result was a map that appeared to suffer from location aliasing: the map tended to evolve into a severely distorted non-rectangular form. Using the laser range measurements in addition to visual features dramatically improved the mapping performance. Final results for the two worlds were of similar quality but without the laser sensor the custom world might not have converged.

Future Work

The next step here is clear enough: deploy on a real robot and see how *rtabmap* performs in a real-world environment. The author has a long term plan to construct a *cat harassment robot*. This robot will naturally be required to seek and harass the target (cat) in arbitrary rooms and houses. Using the SLAM algorithm evaluated here the robot will be able to learn the layout of the house and conduct organized searches for its target. Having constructed an accurate and current map, the robot *should* also be able to segment the anomaly that is the cat and establish a target. Accomplishing this extension is the subject of future work.