Simultaneous Localization and Mapping (SLAM) of In-door environments

Christopher T. Angell

Abstract—For robots working in a new environment, it can be difficult to orientate the robot with respects to the environment especially when a map is not available beforehand. Known as Simultaneous Localization and Mapping (SLAM), this problem is solved by creating a map from sensor measurements and localizing the robot within the created map at the same time. Such algorithms are particularly useful for creating maps when an absolute reference frame such as in the form of GPS or landmarks, is unavailable. Here we explore using the RTAB-Map algorithm to generate maps of two indoor environments: a house and an outdoor environment.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Simultaneous Localization and Mapping, SLAM.

1 Introduction

ROBOTS moving in an environment need to have a map from which to plan paths from, and they need to be able to localize themselves within that environment so as to know where they are on the path. When a map is not available *a priori*, then it must be built by the robot through sensor measurements. Provided that absolute reference frame information is provided, such as through a GPS unit, then mapping is a separate, straight forward problem. However, when such a reference is not available, or is expected to be generated based on sensing the environment, an algorithm is needed that can both produce a map of the environment, and be able to locate the robot within the map. Such algorithms that accomplish both are known as Simultaneous Localization and Mapping (SLAM).

In this report, we overview common SLAM approaches, and use one of them, RTAB-Map, to map two virtual worlds using the Gazebo simulator. We chose to simulate a home environment and an outdoor environment. The outdoor environment presented greater difficulty as featured many poles and blank walls, giving problems for loop closure. After presenting the results, we discuss the various issues that arise when implementing a SLAM algorithm.

2 Background

SLAM is based on using sensing information about the environment combined with odometry to reproduce a map of the environment, with the robot localized in it. Sensors commonly employed are either laser range finders or visual sensors, e.g. cameras. In general, unless absolute information is known about the robot's pose, simple mapping techniques cannot be used to obtain a map of the environment, and SLAM must be used. SLAM techniques are varied, with popular techniques using either particle filters or graph based approaches. Here, we'll discuss three SLAM algorithms: Gmapping, GraphSLAM, and RTAB-Map, another implementation of a graph-based SLAM approach.

2.1 Gmapping

Gmapping uses particle filters that are updated with laser scans to create a map of the environment [1]. Particularly, it uses Rao-Blackwellized particle filters, where each particle carries a map of the environment. Re-sampling of the points is done by computing a sample distribution by evaluating the likelihood of a point by comparing scan data with that point and combining with odometry information to account for where the robot is likely to be. A grid map is updated with the combined information.

2.2 GraphSLAM

GraphSLAM is a graph based approach that tracks the robot position as nodes on a graph, and edges representing constraints between poses imposed either by odometry or by sensor measurements [2]. Whereas particle filter approaches typically solve the online SLAM problem, determining the current position of a robot in a map, graph based approaches solve the full SLAM problem, determining the position of each measurement position with respect to a map.

2.3 RTAB-Map

RTAB-Map is a specific implementation of GraphSLAM that we use here that reduces the memory load of the GraphSLAM algorithm by shifting images between short-term memory, working memory, and long-term storage [3]. Images initially go to short-term memory, and then are ranked by how long they are dwelled on, with high dwell time leading to higher rank. Then the highest ranking images are moved to working memory, and after working memory is filled up the oldest images ares moved to long term storage. This is all done to manage the memory constraint on RTAB-Map loop closure detection algorithm. Loop closure detection is done by matching similarities in images through a bag-of-words approach, where the most salient features of an image are encoded as words, and then words are compared between images. This removes the constraint for having to have identical or near-identical images to require a match. Matches are only done in working memory, so

Fig. 1. The house scene.

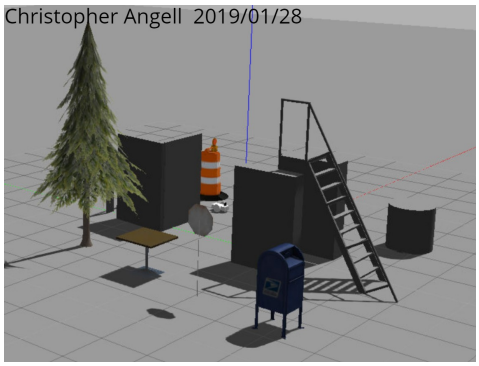

Fig. 2. The outdoor scene.

images are searched in the long-term storage and are then moved to working memory for match verification.

2.4 Comparison / Contrast

The main difference between particle filter approaches and graph-based approaches is that particle filter approaches tackle the online SLAM problem, while the graph-based approaches tackle the full SLAM, also sometimes called the offline SLAM problem. Because of this key difference, with particle filters effectively throwing away information after its been used, particle filters have greater difficulty in loop detection and closure, resulting in mapping problems. Graph based approaches use an offline analysis of the graph so they can locate closures in the graph, making them more robust for mapping larger spaces with robots taking the same path multiple times.

3 Scene and Robot Configuration

The robot simulations were based on the Gazebo package in ROS. This allowed for realistic physics simulations of the robot along with simulation of sensor inputs and odometry, as well as the environment. Two different scenes were modeled, a house and an outdoor environment. The robot used is based off of that from the "Where am I?" project, as was the configuration of nodes and topics in ROS [4].

The house scene (see Fig. 1) was built with a kitchen and a living room, using unique components throughout, making it easy for the robot to perform loop closure.

The outdoor environment in contrast had only a few walls, and had many obstacles which were visible only as poles on the range finder, including a tree, and a stop sign, among others (see Fig. 2). The generally featureless walls and poles made it difficult for accurate loop closure to be performed, and it seems the loop was closed far too frequently.

3.1 Robot

The robot was a box design with dimensions of 0.4 m long, by 0.2 m wide by 0.1 m tall, and had two 0.1 m radius wheels, one at each side at the center of the robot 0.15 m from center. Two casters were placed under the robot at 0.15 m and -0.15m. The two wheel design with casters allowed for a differential drive to be used. Two sensors were included, a camera placed at the front of the robot, and a Hokuyo laser range finder placed near the front on top. The camera was changed to a Kinect, which is an RGB-D camera. A plugin was used in Gazebo, `libgazebo_ros_openni_kinect.so`, to produce the RGB-D point cloud data. The transforms for the robot are shown in Fig. 3. The ROS nodes and messages used are shown in Fig. 4. This set of nodes and messages varies from the previous project [4] in that an optional node was included to turn the RGB-D depth data into a laser scanner.

4 Results

The RTAB-Map package was used for mapping of the environment. For the house environment, it was able to make out accurate details and to build both a 3D and 2D map (see Fig. 5 and 6) based on the RGB-D camera information. The robot was driven through the environment three times, and achieved 13 global loop closures (see Fig. 7). Because of conflict with using the Hokuyo laser range finder in the provided house environment, the 3D map and 2D map were obtained separately. The 3D map was obtained using the RGB-D camera, and the 2D map was also obtained using the RGB-D camera but using the node `depthimage_to_laserscan` to turn the RGB-D image into a laser scan topic rather than the Hokuyo. The RGB-D image and `depthimage_to_laserscan` node could not be used simultaneously due to an unresolved issue with transforms were adjusting the transform for the RBG-D image to be correctly realized in RViz meant that the inferred laserscan had the incorrect transform, and vice-versa.

The map for the outdoor environment was similarly made, with 77 loop closures found (see Fig. 8). The 3D map is shown in Fig. 9 and the 2D map is shown in Fig. 10. The high number of loop closures came from repetitive features. Increasing the required number of features to map in order to close a loop would decrease this number, and improve the quality of the map. However, if the map is too repetitive there may not be a good threshold for number of features to achieve this. The 3D map and 2D map were generated at the same time as the Hokuyo laser scanner had no issues with being used in the outdoor environment.

5 Discussion

The performance of mapping largely depends on the ability for accurate loop closure to occur. As can be seen from

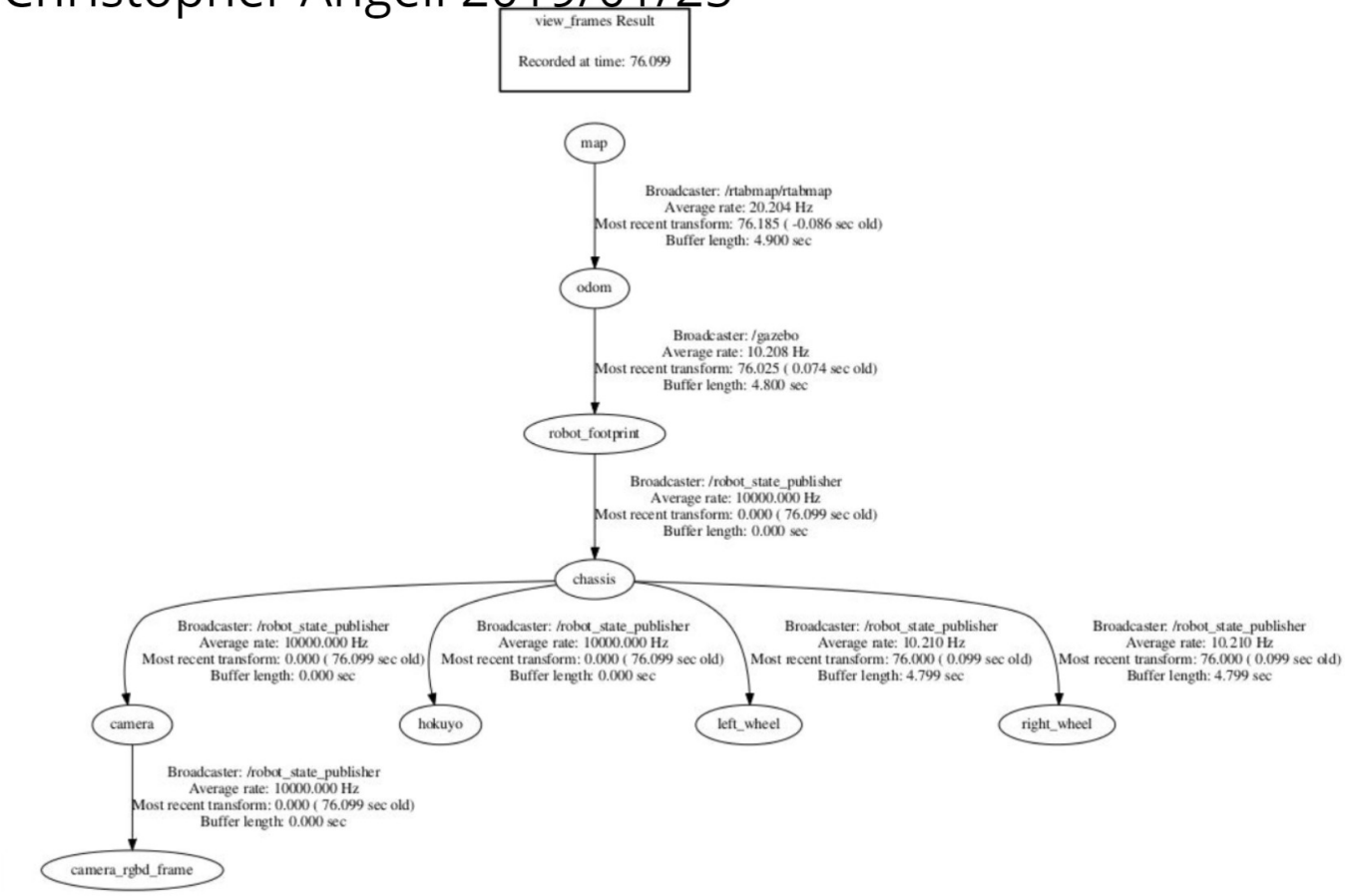

Fig. 3. The transforms for the robot. The second camera transform is needed to re-orient the camera correctly so that it could put the Gazebo point cloud in the correct orientation for visualization in Rviz.

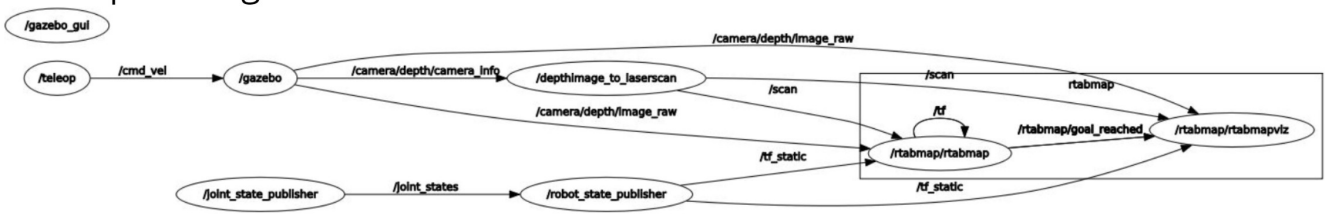

Fig. 4. ROS nodes and topics.

Fig. 6, when a highly differentiated environment is available, mapping is made easier. A relatively high number of loop closures is completed, and the entire map stabilizes to an accurate description of the environment. In the case of the outdoor scene, where the layout is largely made up of flat, non-descript surfaces with a few poles, the algorithm does a poorer job at building an accurate map of the environment as seen in Fig. 10. A scene layout was attempted with descriptive stone walls, but the algorithm would recognize different portions of the walls as the same, and would close the loop at inappropriate locations, so the walls were removed.

The biggest challenge in the case of low details is how to successfully construct at map and close the loop at appropriate locations. Improving odometry is one way. If highly accurate odometry were available, then loop closure could be done by detecting if the robot comes back to the same physical position. Additionally, if external reference were available, such as high-precision GPS, or position reference systems such as the VIVE light-house system as used in virtual reality equipment, then the problem could be taken out of the SLAM domain and fully into the mapping domain.

Fig. 5. A 3d map generated by the RTAB-Map package for the house environment.

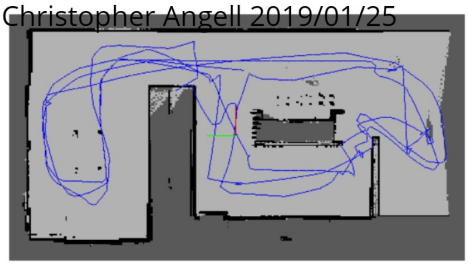

Fig. 6. A 2d map generated by the RTAB-Map package for the house environment.

As such systems may not be fielded, the environment could be enriched, and diversity added such as adding paintings to the walls. Particularly, QR codes could be added to the environment which would code locations, and the robot could learn to recognize the QR codes and reference its position based on such object markers.

6 Conclusion / Future work

In conclusion, SLAM algorithms can be used to make a map of the environment. Modern SLAM algorithms are sufficiently evolved that reliable mapping capabilities are even extended to 3d maps. Here, we only simulated the use of the RTAB-Map SLAM algorithm in Gazebo. The application could be readily extended to implement on real-world hardware by changing where the ROS topics are being broadcast from. A particularly relevant platform would be the Roomba home vacuum cleaner which already includes SLAM as part of its built in functionality to navigate a persons home.

One particularly relevant area of future research would be implementing SLAM in outdoor environments as opposed to indoor, as well as natural environments such as forests as opposed to urban environments. An urban environment with building walls and other man made features should be readily tractable using current SLAM implementations as such environments contain highly distinguishable features. For robots moving in natural environments, the use of GPS as an aid in detecting loop closure should be pursued as such environments may not of themselves provide enough distinguishing features to be accurately mapped using SLAM.

References

[1] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, 2006.
[2] S. Thrun and M. Montemerlo, "The graphslam algorithm with applications to large-scale mapping of urban structures," *Intl. J. Robotics Research*, 2006.
[3] M. Labbè and F. Michaud, "Rtab-map as an open-source lidar and visual slam library for large-scale and long-term online operation," *J. Field Robotics*, 2018.
[4] C. T. Angell, "Localizing two types of mobile robots." Report written in requirement of the Udacity Robotics Nanodegree certificate.

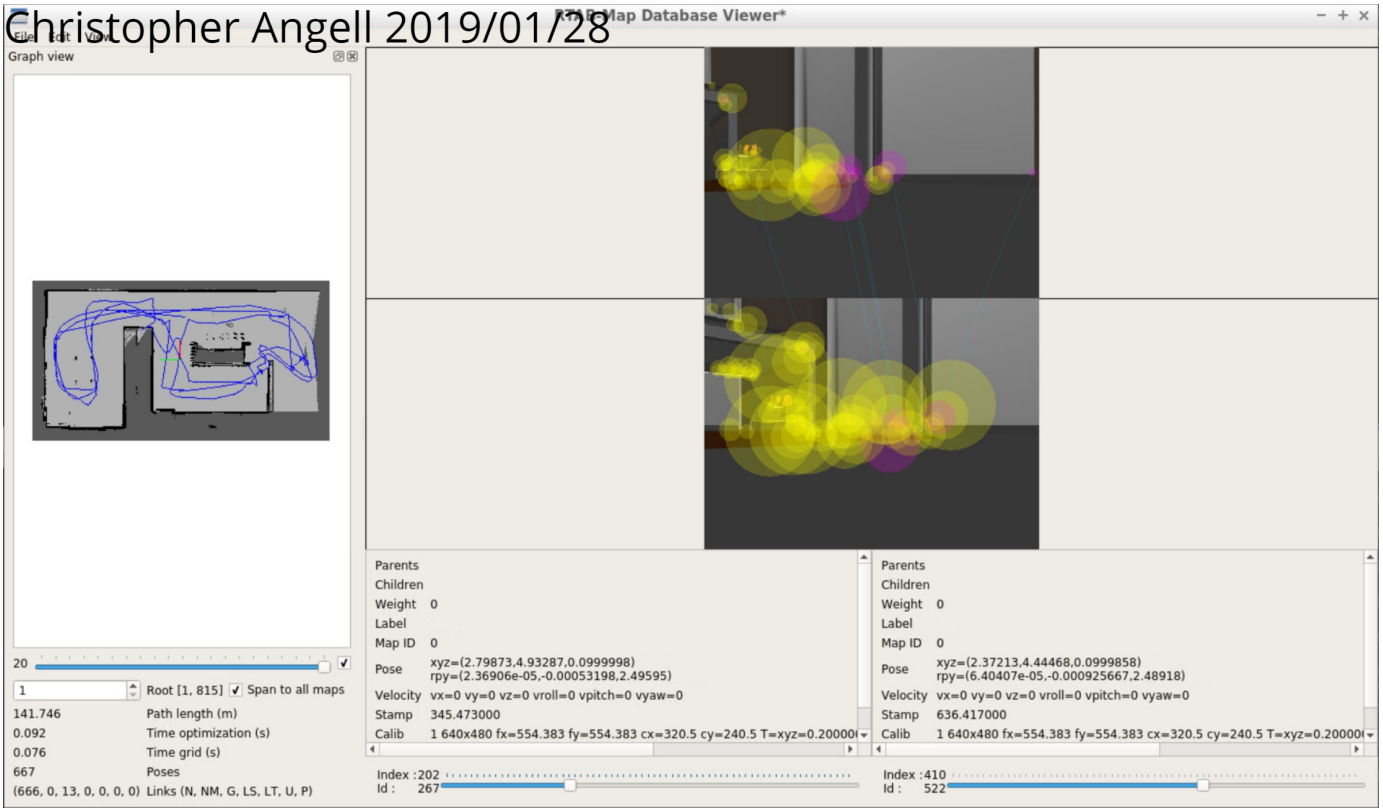

Fig. 7. The database file for the house scene open in the database viewer demonstrating the number of loop closures.

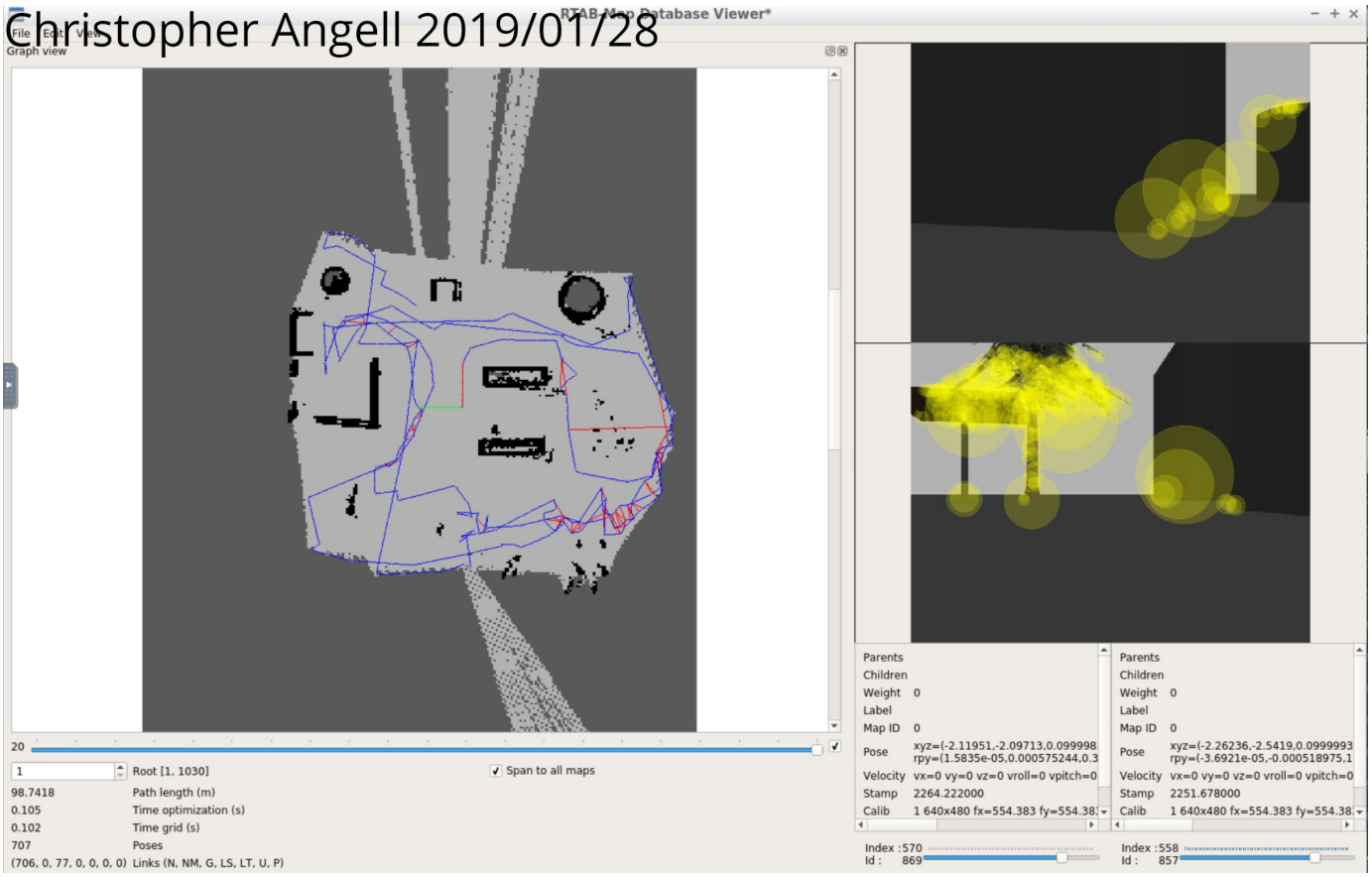

Fig. 8. The database file for the outdoor scene open in the database viewer demonstrating the number of loop closures.

Fig. 9. A 3d map generated by the RTAB-Map package for the outdoor environment.

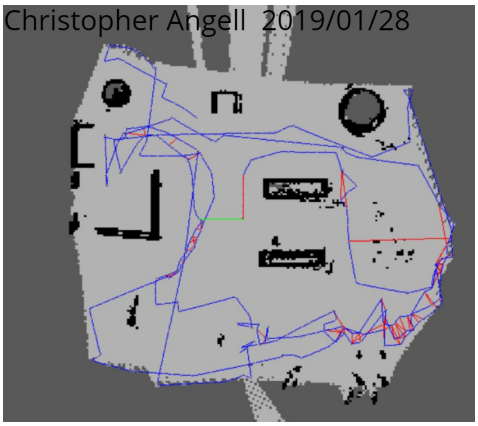

Fig. 10. A 2d map generated by the RTAB-Map package for the outdoor environment.