Robotics Nanodegree SLAM Project

Sebastian Castro — sebas.a.castro@gmail.com

Abstract—This paper discusses the design and implementation of real-time appearance-based simultaneous localization and mapping (SLAM) of mobile robots. This is implemented using a simulated mobile robot with a 2D lidar sensor and an RGB-D camera that can generate a 3D point cloud. The results of SLAM, which include a 2D occupancy grid, a 3D map including color information, and a robot pose trajectory posterior, are tested and compared using two simulated scenes: a indoor environment and and outdoor environment.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Localization, Mapping, SLAM

1 Introduction

Robotic navigation algorithms rely on information about the robot and the environment it must navigate. *Localization* relates to finding the pose of a given a known environment, whereas *mapping* relates to finding information about the environment given a known robot pose.

In practicality, neither the robot pose nor the environment is known a priori. Therefore, robotic systems often employ some mechanism to solve the Simultaneous Localization and Mapping (SLAM) [1] problem.

Depending on the environment in which a robot operates, different sensors may be available for localization and mapping [2]. Proprioceprive (self-sensing) sensors such as inertial measurement units (IMU) and wheel encoders can be used to track the location of a robot relative to a previous known or estimated locations. Exteroceptive (externally sensing) sensors, such as cameras, wheel encoders, and line-of-sight sensors (radar, lidar, etc.). Different sensors have different advantages and disadvantages, and it performing *sensor fusion* to combine estimates from multiple sources can lead to better results.

The work presented in this paper explores one approach of solving the SLAM problem using a simulated robot. This robot, equipped with a known set of sensors, is manually driven through two virtual environments. The design parameters, approach, and results of SLAM for each environment, which include an estimated final map of the environment and trajectory the robot took within that environment, will be discussed and compared in subsequent sections.

2 Background

SLAM problems typically consists of a *front-end* and a *back-end*.

The front-end relates to specifics of the robot itself. Given a mathematical model of the robot, and information about its sensors, what information can be extracted from taking an action and collecting measurements? Specifically, how can these be used to make estimates about the robot pose and map?

The back-end, on the other hand, relates to how this information is used. Using the estimated pose and map information over time from a front-end, a back-end usually puts this information together with the goal of getting a better estimate of the entire robot pose and map posterior for the entire duration of the SLAM task.

There are several types of SLAM back-end methods. Some common ones include:

- **Extended Kalman Filter (EKF) SLAM** – one of the earliest SLAM methods, works well for linear systems and often requires a set of identifiable landmarks or features for a measurement model.
- **Extended Information Filter (EIF) and Sparse Extended Information Filter (SEIF) SLAM** – Computationally equivalent to EKF SLAM, but uses an inverse information representation of covariance matrices.
- **FastSLAM** [3] – Combines particle filters and low-dimensional EKFs for improved performance. Particle filters (or Monte Carlo Localization) are used to estimate the robot trajectory given sensor measurements, and the low-dimensional EKF for solving for map features such as landmarks.
- **GraphSLAM** [4] – Incrementally builds a graph of robot pose and map posteriors that is continuously optimized to maximize the probability of a pose and map posterior given known actions and measurements.

Kalman and Information Filter SLAM methods, at their core, are designed for linear systems with Gaussian noise distributions. Model nonlinearity and non-Gaussian noise can be handled to some extent through Extended and Unscented variations. Additionally, these approaches do not scale well computationally with high-dimensional map features (though Information form empirically scales better).

FastSLAM and GraphSLAM approaches, on the other hand, more easily have the ability to use higher-fidelity map representations such as occupancy grids. This is important when the map cannot easily be represented by a relatively small set of landmarks.

FastSLAM uses particle filters, each particle with its own map representation and low-dimensional EKF for pose estimation. This means there is high memory usage and a strong dependency on number of particles and their initial distribution for results to converge.

Finally, GraphSLAM methods do not require sampling and tracking particles, but rather continuously performing

a maximum-likelihood estimate of the pose and map posteriors as new actions and measurements are added to the graph.

The front end can be done with different sensors, most commonly proprioceptive (odometry, IMU, GPS) and exteroceptive (lidar, image features, or 3D point clouds.

Finally, worth discussing loop closures, or using similar measurements. This reduces the size of the graph and improves robustness of the final map.

3 Approach

3.1 Model Configuration

The simulated robot is a differential-drive robot with wheel encoders for odometry, as well as a front-facing RGB-D camera and a 2D horizontal lidar sensor with full rotational field of view. Simulation is performed using the Gazebo simulator.

Figure 1 and Table 1 describe the details of the robot model kinematics, dynamics, and sensor configuration.

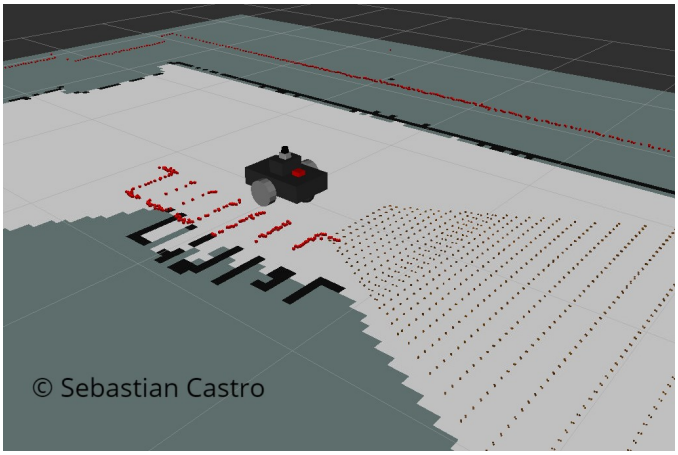

Fig. 1. Robot model with sensor display

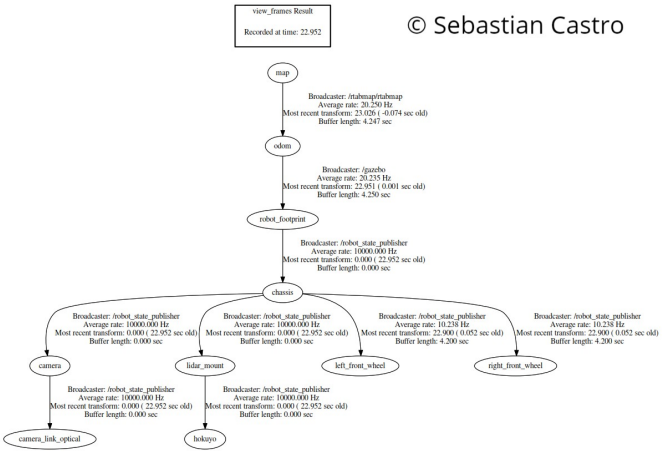

Fig. 2. ROS transformation tree showing robot model coordinate frames

TABLE 1
Robot Model Parameters

Total Mass	25.2 kg
Wheel radius	75 cm
Wheelbase	45 cm
Camera location [XYZ]	[0.1 0 0.5] m
Camera field of view	60 deg
Camera depth range	0.1 - 20 m
Lidar location [XYZ]	[0 0 0.15] m
Lidar range	0.25 - 20 m
Lidar angles	720 samples, -180 to +180 deg

3.2 World Creation

3.2.1 *Benchmark World*

Figure 3 shows the benchmark simulated world in Gazebo. This is an indoor environment consisting of two separate rooms: a kitchen and a dining room.

Fig. 3. Benchmark simulation world

3.2.2 *Custom World*

Figure 4 shows a custom developed simulated world in Gazebo. This is an outdoor environment consisting of a building front and a small yard containing walls, barriers, and assorted obstacles.

Fig. 4. Custom simulation world

3.3 SLAM Algorithm and Parameters

Our approach involves a variation of GraphSLAM known as real-time appearance-based mapping (RTAB-Map) [5] [6]. RTAB-Map employs a visual bag-of-words approach to reduce dimensionality, as well as memory management technique that limits the number of locations used to perform GraphSLAM (pose graph optimization and loop closure detection).

RTAB-Map is available as a ROS package that can integrate with the robot and environment simulations presented. It allows the use of different sensor configurations, which may include odometry, 2D lidar scans, and a depth-sensing vision system such as an RGB-D camera or stereo camera pair.

As discussed previously, our simulated robot measurements consist of wheel odometry, a 2D lidar, and an RGB-D camera. Then, RTAB-Map is configured with the following options:

- Nodes are added to the pose graph at a rate of 1 Hz.
- 2D SLAM is performed, meaning that pose graph optimization and loop closure detection is done on a 2D map. This also means that the estimated map is represented as a 2D occupancy grid.
- Loop closure is determined by matching laser scans using an iterative closest points (ICP) algorithm [7], although matching visual features is also required to accept a loop closure.
- The visual feature descriptors used are Speeded Up Robust Features (SURF) [8], requiring a minimum of 15 inliers with a minimum Hessian threshold of 100 to accept a match.
- Each collected image can have a maximum of 400 visual words in the bag-of-words model. More visual words can improve results, but decrease computational performance.

3.4 Software Architecture

The above approach is contained in a ROS package named `slam_project`, which can be downloaded from this GitHub repository. This package provides the following launch files to reproduce the results presented in the following section.

- `world.launch` - Starts the Gazebo simulator and robot model, as well as RViz for sensor visualization
- `mapping.launch` - Starts the RTAB-Map node for performing SLAM
- `teleop.launch` - Allows the user to manually move the robot in the environment using keyboard controls

The list of ROS nodes and topic names can be found in Table 2 and in the `rqt_graph` snapshot in Figure 5.

TABLE 2
ROS Topic Names

ROS transform information	`\tf`
Odometry	`\odom`
Camera information	`\camera\rgb\camera_info`
Camera color image	`\camera\rgb\image_raw`
Camera depth image	`\camera\depth\image_raw`
Laser scan	`\scan`
Velocity command	`\cmd_vel`
2D occupancy map	`\map`
3D map cloud	`\rtabmap\mapData\costmap`

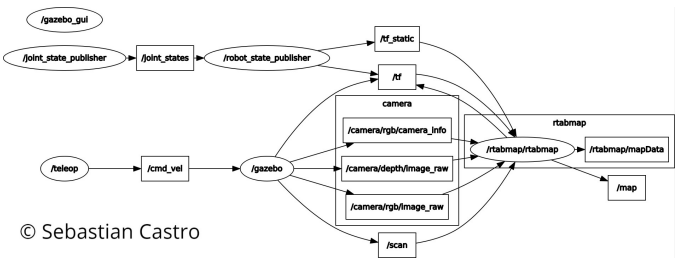

Fig. 5. ROS graph showing topics and node interactions

4 Results

4.1 Benchmark World

In the benchmark world, the robot was driven around each room twice with the goal of identifying loop closures. Figure 6 shows the estimated 3D map matching closely with the ground truth map in 3.

Figure 7 shows the estimated 2D occupancy map, as well as shows mapping information. The pose graph consisted of 601 nodes with 121 global loop closures. Most of these loop closures appear to have been detected when traversing the south side of the kitchen room.

Fig. 6. 3D map of benchmark simulation world

4.2 Custom World

In the custom world, the robot was driven in a loop twice with the goal of identifying loop closures and trying to create a sufficiently accurate 3D map of all the objects scattered in the environment.

Figure 8 shows the estimated 3D map matching relatively closely with the ground truth map in 4. There are some artifacts in the front of the post office building which can be seen in the "Post Office" sign, wooden briefcase, and traffic cone appearing as if there were two objects.

Figure 9 shows the estimated 2D occupancy map, as well as shows mapping information. The 2D occupancy map

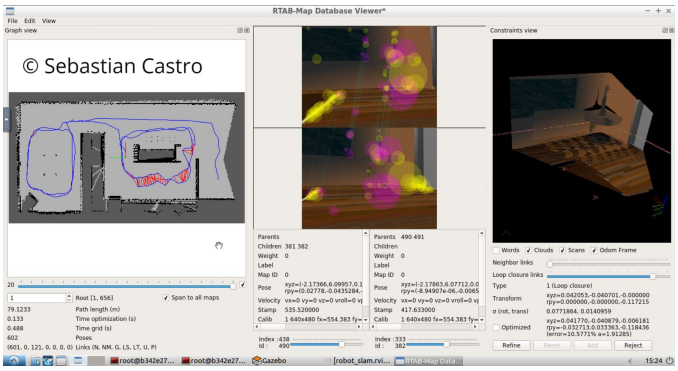

Fig. 7. 2D map and mapping information of benchmark simulation world

contains significant noise in the representation of walls and obstacles, especially compared to the results from the previous environment. The pose graph consisted of 271 nodes with 149 global loop closures spread evenly throughout the robot trajectory.

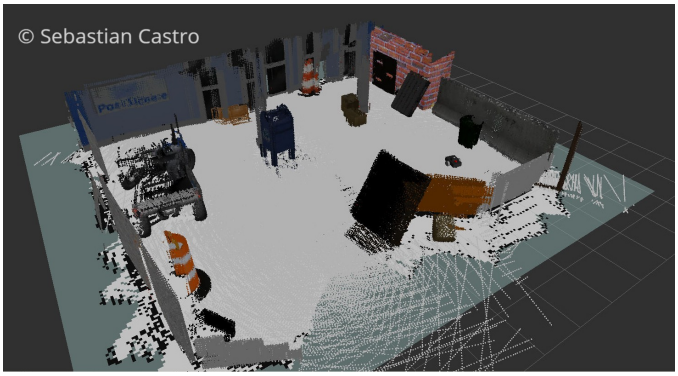

Fig. 8. 3D map of custom simulation world

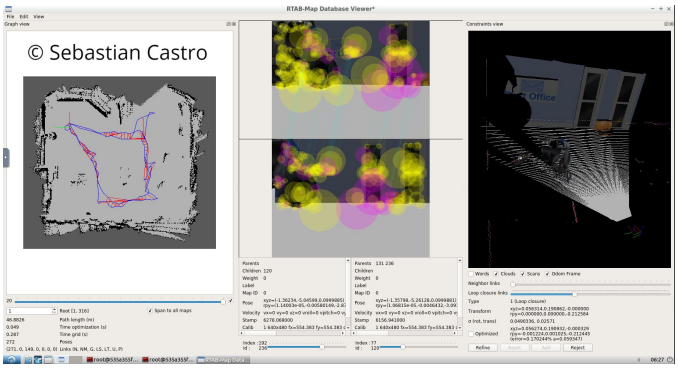

Fig. 9. 2D map and mapping information of custom simulation world

5 Discussion

As can be seen from the results above, the SLAM results were significantly more accurate for the benchmark world compared to the custom world.

The benchmark map pose graph had more than double the nodes, yet had slightly less loop closures. This may indicate there were more false loop closures in the custom map, which created artifacts in both the 2D and 3D maps that made certain walls and obstacles look like multiple objects staggered on top of each other.

Based on these results, it can be seen that the RTAB-Map configuration parameters selected seemed better suited for indoor mapping. Some issues identified (and partially solved) while performing mapping on the custom world, include:

- **Mismatch in sensor vs. obstacle height** – Notice that the 2D occupancy map for the custom world does not appear to have as many obstacles as the ground truth map would indicate. For some obstacles like the vehicle and mailbox, the robot sensor seems to only pick up small areas near the ground, such as wheels and legs. So, this affects specifically how much data the laser scanner has to detect neighboring points and loop closures. Contrast this with the benchmark world, in which most obstacles are solid walls or furniture elements.
- **Wide-open spaces were insufficiently feature-rich**, both for images and scans, so they could easily match to other wide open spaces elsewhere. Originally, the custom map had an asphalt floor, but this created too many SURF features and created false loop closure detections. So, the ground texture was removed.
- **Large objects with repetitive textures** in the custom world contained features that looked the same from significantly different viewpoints, thus affecting results. This was evident in the brick walls, as well as the post office building front which has several columns/windows along its span. This was partially resolved by placing unique objects along to the walls. While this fixed several issues with mapping, it affects how realistically the simulated world represents a true outdoor environment.
- **Seeing objects from multiple points of view** – For instance, notice that most objects are pressed against walls. If the robot sees what seems like the same object view from the front and the back, this can cause false loop closures. An example of this was the orange Jersey barrier, which looks the same from the front and the back. This was partially solved by propping a cardboard box against one of the two sides of the barrier. Another example had to do with the cylindrical traffic cones. If these were placed near the center of the map so the robot could drive around them, this affected the map building results.

That being said, there were some feature-rich objects in the custom simulated environment that could be tracked reliably. These included the vehicle in front of the building, the "Post Office" sign (specifically the mailbox logo and text), and the metal plate with holes stood against the red brick wall.

6 Conclusion / Future work

In this paper, the RTAB-Map graph-based SLAM approach was implemented and tested on two virtual worlds with a simulated robot.

The resulting estimated map and robot trajectory were shown to be more accurate in obstacle-dense indoor environments where there are multiple lidar scan readings and visual features to perform pose graph optimization and detect loop closures.

Below are some additional ways to ensure better mapping performance on the outdoor simulation world.

- **Using different parameters for indoor vs. outdoor mapping** – Since the outdoor scene has a lower concentration of obstacles and more repetitive visual features, then other visual feature descriptors, as well as tighter thresholds for loop closure and visual feature strength, could be explored.
- **Modifying the robot sensor configuration** – Repositioning existing sensors, adding more sensors, or using sensors with higher resolution, longer range, wider field of view, and/or less simulated noise could improve mapping results. However, practical limitations of real-world sensors should be considered.
- **Taking smaller steps between sensor readings** – Whether this means driving the robot slower or performing additions to the pose graph more frequently, having less time and/or distance between readings can improve mapping results at the expense of more computation or more time to complete driving the intended trajectory.

Some other tasks to consider in future for extending this work include:

- Once a map has been generated, it can use used to purely perform localization. This can then be combined with other navigation algorithms to automatically navigate the robot within the map.
- If the map is not static, i.e., it contains dynamic obstacles, then new sensor readings could be used to update the existing map in an online fashion during autonomous navigation
- Deploying the RTAB-Map SLAM approach on real robot hardware to see how results differ when there are lighting conditions, more complex environments, and more noise in motion and measurements with real actuators and sensors.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard, "Simultaneous localization and mapping: Present, future, and the robust-perception age," *CoRR*, vol. abs/1606.05830, 2016.

[2] J. Borenstein, H. R. Everett, L. Feng, and D. K. Wehe, "Mobile robot positioning: Sensors and techniques," *J. Field Robotics*, vol. 14, pp. 231–249, 1997.

[3] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593–598, AAAI, 2002.

[4] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, winter 2010.

[5] M. Labbe and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *Robotics, IEEE Transactions on*, vol. 29, pp. 734–745, 06 2013.

[6] M. Labbe and F. Michaud, "Memory management for real-time appearance-based loop closure detection.," in *IROS*, pp. 1271–1276, IEEE, 2011.

[7] E. Mendes, P. Koch, and S. Lacroix, "Icp-based pose-graph slam," *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 195–200, 2016.

[8] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In ECCV*, pp. 404–417, 2006.