Simultaneous Localization and Mapping Project: Map My World

Carlos E. Sanoja

Abstract—A two wheeled robot equipped with a RGB-D camera and a 2D Lidar sensor was designed to traverse two Gazebo world environments while performing Simultaneous Localization and Mapping (SLAM). This work aims to generate 2D occupancy grids and 3D octomaps of the provided environments using Real Time Appearance Based Maping (RTAB-Map).

Index Terms—Robot, Udacity, Localization, Mapping, SLAM, RTAB-Map.

1 Introduction

The localization problem aims to estimate the robot's pose given its odometry data and a provided map of the environment. But, in real world problems, there isn't a known map. That could be because the area is unexplored or because the surroundings change often and the map may not be up to date. In such case, the robot have to build the map.

Robotic mapping assumes that a robot knows its pose and has access to its movement and sensor data. In other words, mapping assumes an old path and estimate the environment. A map generally lies in a continuous space and as consequence, there are infinitely many variables used to describe it. In addition to, mapping involves a lot of challenges: the weather, repetitive scenarios and the geometry of the environment.

In real world scenarios, the map and the robot's pose are unknown. SLAM stands for simultaneous localization and mapping and is a concept which solves those important problems. This has been a challenge for the past years because SLAM is considered a fundamental problem for robots to become truly autonomous. This is the reason of why has been developed a large variety of different SLAM approaches and most of them, using probabilistic concepts.

In this project, two 3D world environments have to be simulated and mapped using SLAM. As there are many approaches to tackled this problem, it is used Real Time Appearance Based Mapping (RTAB-Map). RTAB-Map is a GraphSLAM algorithm implementation that uses incremental appearance based loop closure detection. As a result of this project, there are presented a 2D and 3D map of the environments with the estimation path and the features that allow to perform loop enclosure.

2 Background

SLAM is the most challenging problem, since the only information the robot has is the measurements and motions, while it has to estimate both its own poses and the map feature poses. As a result, the solution space for the SLAM problem is highly dimensional. In addition to, SLAM algorithms also need to identify object correspondences between images in order to resolve map features correctly (loop enclosures). However, the correspondence values increase exponentially over time since the robot captures more images as it explores the environment.

Sensor noise reduces the useful information content of sensor readings. Clearly, the solution is to take multiple readings into account, employing temporal fusion or multi-sensor fusion to increase the overall information content of the robots inputs. The challenges of localization do not lie with sensor technologies alone. Just as robot sensors are noisy, limiting the information content of the signal, so robot effectors are also noisy. In particular, a single action taken by a mobile robot may have several different possible results, even though from the robots point of view the initial state before the action was taken is well-known. In short, mobile robot effectors introduce uncertainty about future state. Therefore the simple act of moving tends to increase the uncertainty of a mobile robot. There are, of course, exceptions.

Because of that, localization and mapping algorithms plays important role in calculating approximate position of the robot. There are many approaches to perform SLAM such as:

- Extended Kalman Filter SLAM (EKF)
- Extended Information Form (EIF)
- Sparse Extended Information Filter (SEIF)
- FastSLAM
- GraphSLAM

The two most commonly used approaches are FastSLAM and GraphSLAM. Both of them, solve the SLAM problem well, although in different ways.

2.1 FastSLAM

The key idea of FastSLAM exploits the fact that knowledge of the robot's path $s_1, s_2, \ldots, s_t$ renders the individual landmark measurements independent. FastSLAM decomposes the SLAM problem into one robot localization problem, and a collection of K landmark estimation problems (EKF). In FastSLAM, poses are assumed to behave according to a probabilistic law named motion model with an underlying density $p(s_t|s_{t1})$.

Likewise, the measurements are governed by the probabilistic measurement model $p(z_t|s_t, \theta, n_t)$ with z_t measurement, $\theta = \theta_1, \ldots, \theta_K$ the set of landmarks, and n_t 1, ..., K the index of the observed landmark at time t (only one at a time). The ultimate goal is to estimate the posterior $p(s_t, \theta|z_t)$.

2.1.1 Grid-based FastSLAM

With grid mapping algorithm is possible to model the environment using grid maps without predefining any landmark position. So by extending the FastSLAM algorithm to occupancy grid maps, now is possible to solve the SLAM problem in an arbitrary environment. With this extension each particle holds a guess of the robot trajectory. In addition, each particle maintains its own map. The grid-based FastSLAM algorithm will update each particle by solving the mapping with known poses problem using the occupancy grid mapping algorithm.

To do so, the algorithm perform three main techniques:

1) **Sampling Motion:** $p(x_t x_{t1}{}^k, u_t)$: Estimates the current pose given the k-th particle previous pose and the current controls u
2) **Map Estimation:** $p(m_t z_t, x_t{}^k, m_{t1}{}^k)$ Estimates the current map given the current measurements, the current k-th particle pose, and the previous k-th particle map
3) **Importance Weight:** $p(z_t x_t{}^k, m^k)$ Estimates the current likelihood of the measurement given the current k-th particle pose and the current k-th particle map.

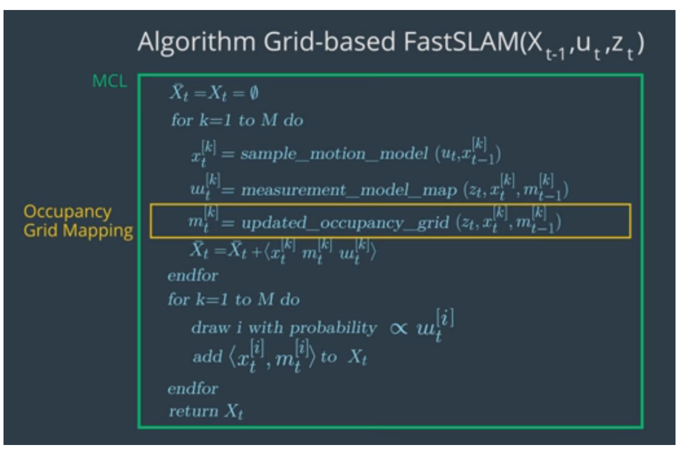

Fig. 1. FastSLAM algorithm.

2.2 GraphSLAM

A graph-based SLAM approach constructs a simplified estimation problem by abstracting the raw sensor measurements. These raw measurements are replaced by the edges in the graph which can then be seen as virtual measurements. More in detail an edge between two nodes is labeled with a probability distribution over the relative locations of the two poses, conditioned to their mutual measurements. In general, the observation model $p(z_t|x_t, m_t)$ is multimodal and therefore the Gaussian assumption does not hold.

This means that a single observation z_t might result in multiple potential edges connecting different poses in the graph and the graph connectivity needs itself to be described as a probability distribution. Directly dealing with this multi-modality in the estimation process would lead to a combinatorial explosion of the complexity. As a result of that, most practical approaches restrict the estimate to the most likely topology. Thus, one needs to determine the most likely constraint resulting from an observation. This decision depends on the probability distribution over the robot poses.

So far, the GraphSLAM algorithm represents the SLAM problem as a graph where the poses of the trajectory and the measured locations are the nodes, and the links are the estimated motion and measurement distances represented as constraints. The algorithm solves for the best configuration of the graph to satisfy the constraints.

The Real-Time Appearance-Based Mapping algorithm is a GraphSLAM approach and it will be used in this project to perform SLAM. This algorithm uses data collected from sensors to localize the robot and map the environment. In RTAB-Map a process called loop closure is used to allow the robot to determine if the location has been observed before.

2.2.1 Loop Closure

The loop closure detection occurs against working memory to constrain the number of images interrogated. Working memory can be transferred and retrieved from long term memory to reduce complexity. RTAB-Map uses a memory management technique to limit the number of locations considered as candidates during loop closure detection. This technique is a key feature of RTAB-Map and allows for loop closure to be done in real time. The algorithm used for loop closure detection is SURF (Speeded Up Robust Features).

SURF is a detector and a high-performance descriptor of the points of interest of an image, where the image is transformed into coordinates, using a technique called multi-resolution. It consists of making a replica of the original image of the Gaussian Pyramidal or Laplacian Pyramidal form, and obtaining images of the same size but with reduced bandwidth. In this way a fuzziness effect is obtained on the original image, called Scale-Space. This technique ensures that points of interest are invariant in scaling.

3 Configurations

The Gazebo and Rviz environments are setup and the robot is launched inside the environment. The appropriate launch files are created.

3.1 Provided Simulated Environment

The Kitchen and Dining Scene world was provided and will be used to be mapping (See Fig. 2).

3.2 Custom Environment

A new scene is built from scratch in Gazebo. The models in the scene are all from the model database as is shown in Fig. 3

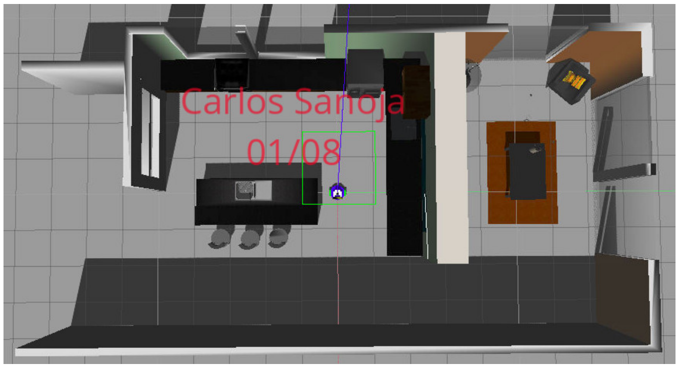

Fig. 2. Kitchen and Dining world.

Fig. 3. Custom world.

3.3 Robot configuration

In Fig.4 is shown the udacity bot. A mobile robot created to navigate and localize itself through a know map. Keeping ROS as framework, the model was developed using the Unified Robot Description Format (URDF). Its structure consists in (Fig. 5):

- main body which is 0.4x 0.2 x 0.1 box
- two wheels with 0.1 radius attached to left and right sides
- front and back spherical caster (with radius 0.05) underneath the robot body for stability.

The robot is also equipped with a front facing camera mounted to the front of the robot and a Hokuyo LIDAR attached to the top.

3.4 RTAB-Map Configuration

RTAB-Map (Real-Time Appearance-Based Mapping) is a RGB-D Graph SLAM approach based on a global Bayesian loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location. When a loop closure hypothesis is accepted, a new constraint is added to the map's graph, then a graph optimizer minimizes the errors in the map.

A memory management approach is used to limit the number of locations used for loop closure detection and

Fig. 4. Explo Bot Model.

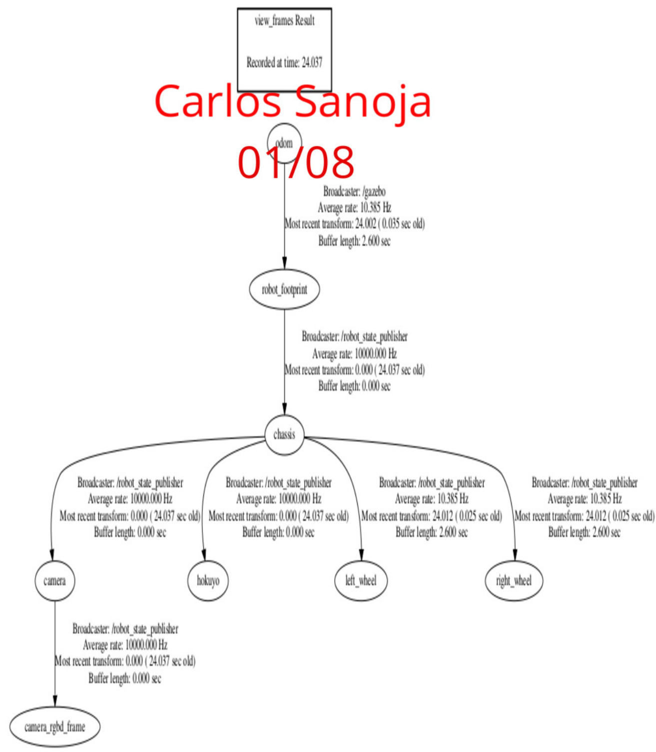

Fig. 5. Frames of Explo Bot Model.

graph optimization, so that real-time constraints on large-scale environnements are always respected. For this project, RTAB-Map will be used alone with a handheld Kinect a robot equipped with a laser rangefinder.

This package can be used to generate a 3D point clouds of the environment and/or to create a 2D occupancy grid map for navigation.

RTABMAP is the main node of this package. It is a wrapper of the RTAB-Map Core library. This is where the graph of the map is incrementally built and optimized when a loop closure is detected. The online output of the node is the local graph with the latest added data to the map. To get a 3D point cloud or a 2D occupancy grid of the environment, subscribe to *cloud_map, grid_map or proj_map topics*.

To tunned this package, their parameters should be changed accordingly. For this project was as follow:

- **queue_size** (int, default: 10): Size of message queue for each synchronized topic. It was set to 30 to avoid issues about sync with topics publishing rates.
- **Kp/MaxFeatures**: Maximum visual words per image (bag-of-words). set to 550.
- **SURF/HessianThreshold:** Used to extract more or less SURF features. Set to 100.
- **Kp/DetectorStrategy:** set to 0 to use SURF as algorithm for feature detection
- **Vis/MinInliers:** Minimum visual inliers to accept loop closure. Set to 15

4 RESULTS

Both environments were mapped satisfactorily. To accomplish that it was necessary to map accordingly the topics over ROS environment. In Fig. 6 are shown the topics and how are they connected.

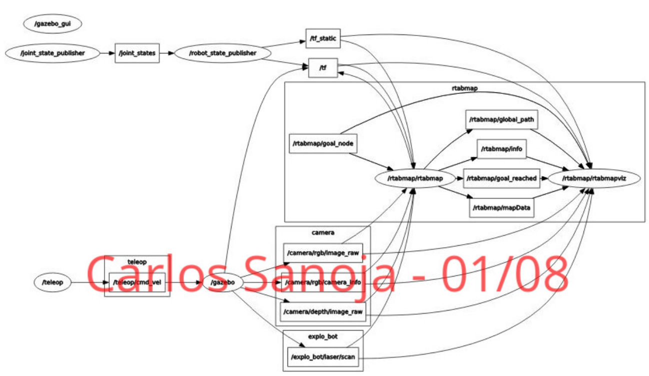

Fig. 6. Topics relationship.

As a measure of the possible errors of the system, it's possible to analyses how well the algorithms performed through loop closure detection. In Fig. 7 and Fig. 8 are shown the map created for every environment and the possible loop closure detection (for the robot position) and the highest hypothesis of that position in the map. It is visible that the algorithm perform very well and the features found were correlated with older ones to do the loop closure satisfactorily.

4.1 Kitchen and Dining World Results

By analyzing the map created, it could be observed that there are many loop closures detected. Since the robot revisited every point in the world at least twice, the loop closures correctly identify when the poses around there are similar.

In Fig. 9 and Fig. 10 are shown the 2D and 3D maps respectively.

4.2 Custom World Results

In Fig. 11 and Fig. 12 are shown the 2D and 3D maps respectively.

Finally, the rtab-models are store in the following link because they are big: ***https://goo.gl/spprLx***

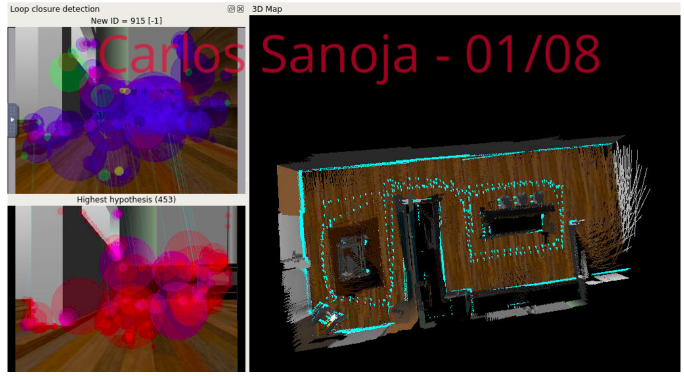

Fig. 7. Loop closure detection in kitchen and Dining world.

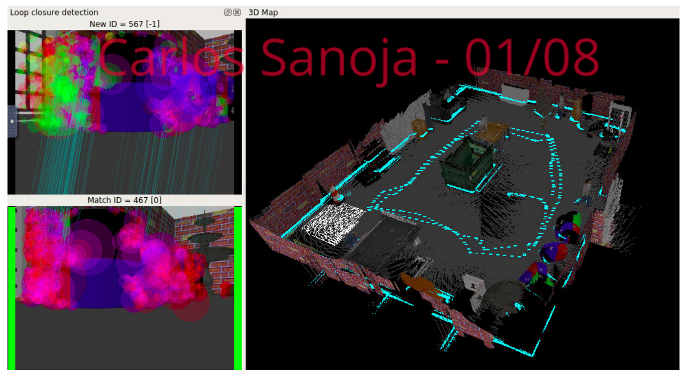

Fig. 8. Loop closure detection in custom world.

Fig. 9. 2D occupancy grid for Kitchen Dining world.

Fig. 10. 3D map for Kitchen Dining world.

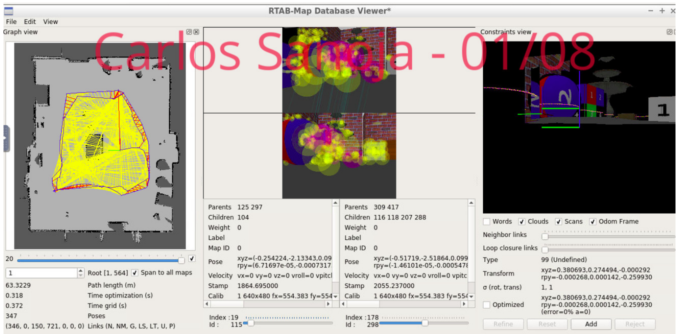

Fig. 11. 2D occupancy grid for Custom world.

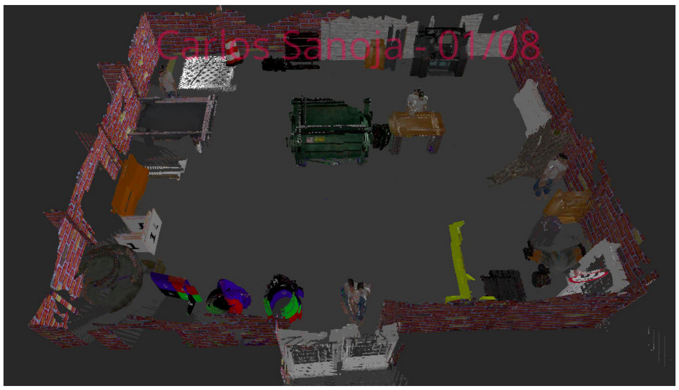

Fig. 12. 3D map for Custom world.

5 Discussion

The task of mapping the provided environment was easier than mapping the custom world. It is capable to map accurately enough in its first try through the rooms. By the other hand, it took at least twice rounds of completed navigation to the custom world to obtain a considerable result. That happens because the provided map contains a lot of feature-rich objects that allow to perform the loop closure quickly and precisely.

The goal was achieved satisfactorily and the environments were mapping (2D and 3D) correctly. The robot is controlled by keyboard, not autonomously. thus, the performance of mapping in both worlds goes well in case that the robot is controlled without collisions and following similar paths to ensure feature detections.

6 Conclusion / Future work

A 2D occupancy grid and 3D octomap are created from a provided simulated environment. It is shown how RTAB-Map algorithm work and how it is the best possible solution to tackle SLAM problem in real-time processing.

Possible future work includes developing a wheeled robot equipped with the Jetson TX2 and a kinect RGBD camera sensor to map an entire real-life environment.

7 Reference

1) https://es.wikipedia.org/wiki/SURF
2) A Tutorial on Graph-Based SLAM. Giorgio Grisetti
3) The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. Sebastian Thrun
4) EKF SLAM vs. FastSLAM A Comparison. Michael Calonder, Computer Vision Lab
5) SLAM for Dummies. Sren Riisgaard and Morten Rufus Blas
6) Graph-Based SLAM. Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Diego Tipaldi, Luciano Spinello
7) http://wiki.ros.org/rtabmap_ros
8) http://wiki.ros.org/rtabmap