Map My World Robot

Robotics Software Engineer Program

December 23, 2018

Abstract

In this report is presented an introduction to Mapping and Simultaneous Localization and Mapping (SLAM), by implementing them on a simulated robot to generate maps of specific scenarios. This was possible using a Graph based SLAM approach, using the Graphic Real Time Appearance Base Mapping (RTAB-Map) technique to localize the robot and map the environments based on data collected from the robot visual sensors. The techniques were tested in two different scenarios, proving the effectiveness of the algorithms under different circumstances. Results obtained show that the techniques were able to localize the robot while generating 2D and 3D maps for the mentioned scenarios. Nonetheless, the accuracy obtained in the presented results can be improved if different parameters are used, however, as being an introduction the results presented are enough to understand the mentioned algorithms.

I. Introduction

Localization proved to be a great technique to determine the robot's position on a well defined map, in a previously presented report. However, not in all cases the map is known, and therefore, it needs to be found using different techniques in order for the robot to knows its location. Robot Mapping is a technique similar to localization, however, instead of known the map that defines the environment to estimate the position of the robot, robot mapping estimates the environment based on the robot's position. The problem of generating a map from an unknown environment is called mapping.

Nevertheless, it is important to mention that different variables can increase the complexity of the mapping problems. For example, in a repetitive environment the complexity increases due to difficulties identifying similar objects as different, in order to be used as references to determine the robot's specific location. Additionally, a map generally lies in a continuous space, as a result an infinite number of variables can be used to describe it. Fortunately, there exist algorithms that help us to solve the mapping problem, such as the Occupancy Grid Mapping algorithm.

There exist a third problem in which nor the map or the robot's position is known. This problem is called Simultaneous Localization and Mapping (SLAM). SLAM, estimates the robots position while generates a map of the robot's location by exploring its environment. Many techniques have been developed to solve this problem, such as EKF SLAM [1], SEIF SLAM [2], EIF SLAM [3], FastSLAM [4] and GraphSLAM [5]. However, this report is based in the implementation of GraphSLAM, to be more specific, it is based on the use of the Graphic Real Time Appearance Base Mapping (RTAB-Map) technique.

More details about the mentioned techniques can be found in next section. Furthermore, details about the robot and the scenarios used can be seen in sections III and IV, respectively. The obtained results are presented in section V and the Discussion and Future work are presented in sections VI and VII, respectively.

II. Background

i. Occupancy Grid Mapping

As mentioned before, the continuous space for the generation of maps increase the complexity while solving the mapping problem, due to the large number of variables related

with the space and because the mapping problem is solved based on data acquired from visual sensors. Uncertainties are generated from large signal to noise ratios, which increase the complexity of the mapping problem solution. Lastly, perceptual ambiguity is the last problem to consider when solving the mapping problem, objects that looks alike generate problem when determining the current position of the robot.

The Occupancy Grid Mapping algorithm [6] is used to solve the mapping problem. This technique presents a discrete approximation of the environment as a finite number of grid cells, used to determine which cells are being occupied or not. The technique uses the previously occupancy values of the cells, the poses and the sensors measurements as parameters to determine the current status of the cell. In figure 1, it can be observed the representation of the Occupancy Grid Mapping algorithm, where white cells represent cells that falls under the sensors range while dark cells represent those that are not being measured by the sensors. Then, the Occupancy Grid Mapping algorithm is used to find the current occupancy status of the white cells.

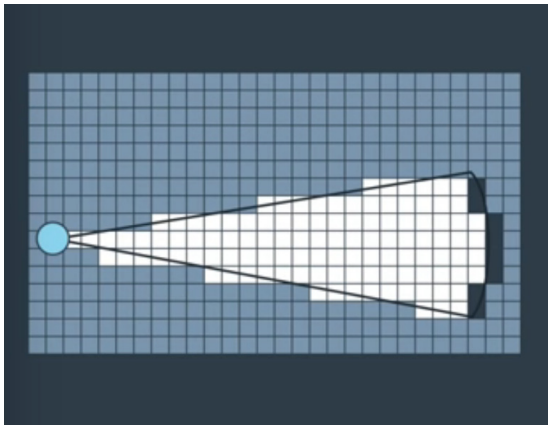

Figure 1: *GridMapping measurement range.*

Nonetheless, the problem of generating a map under the assumption that the robot poses are known and non noisy is referred as mapping with known poses. Then, Occupancy Grid Mapping algorithm can estimate the posterior map. In most robotics applications the robot poses are unknown, however, mapping usually happens after SLAM, which will be explained in more details in the next section. Once SLAM is implemented, the known locations for the robot are used to solve the mapping problem for path planning and navigation.

ii. SLAM

SLAM is used to estimate the map and robot pose in real-worlds environments. The difference between SLAM and mapping is that in mapping the map is estimated by knowing the robotâĂŹs poses, while in SLAM the map and robot pose are unknown. However, as stated previously, mapping depends on SLAM for real world environments, as can be seen in figure 2.

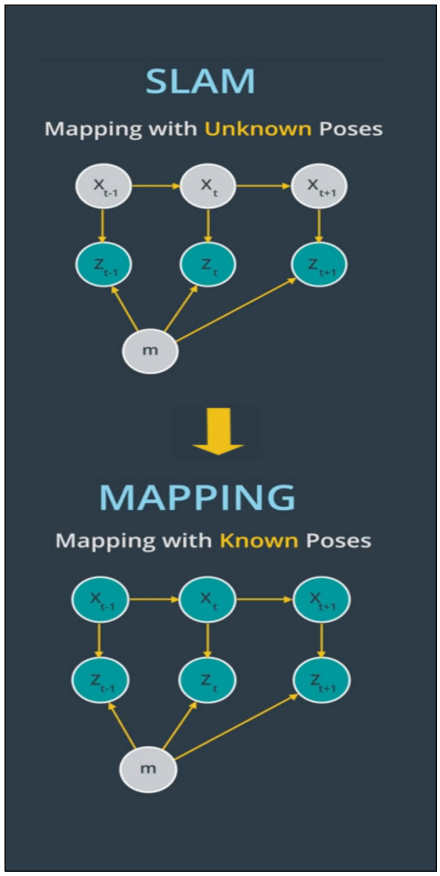

Figure 2: *SLAM and Mapping relationship.*

where, x, z and m are used to represent the robot poses, the sensors measurements and the map, respectively. In the case of the robot poses and sensors measurements the time instants are used to represent previous, current and the posteriori poses and measurements. Moreover, the green circles represent known values while gray circles are used to represent unknown values.

Thus, in SLAM the idea is to map the environment giving noisy measurements and to localize the robot relative to its own map giving the controls, as shown in figure 3, where u is used to represent the control actions at specific time instants.

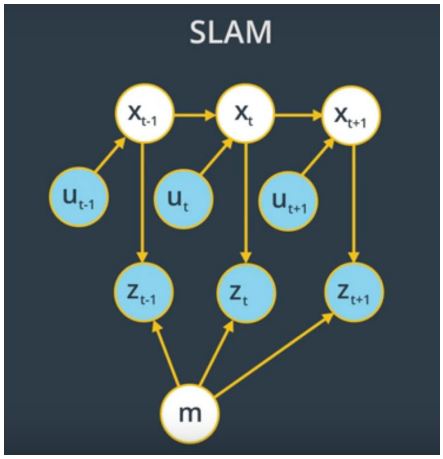

Figure 3: *SLAM problem.*

There exist two SLAM problems that are important in robotics, online SLAM and Full SLAM or offline SLAM. In online SLAM, the robot estimates its current pose and the map giving its current measurements, without considering the previous measurements and controls actions. On the other hand, in Full SLAM the robots estimates the entire path, x_{t-1} to x_{t+1}, and the map giving all measurements and action controls.

However, there exist additional characteristics of the SLAM problem to consider for its solution as is its nature. The SLAM problem generally have continuous and a discrete element, continuous since the robot continuously acquire data from the sensors and a discrete element given that the robot continuously ask itself if it has been at a specific position before. This discrete relationship between objects detected by the robot and its current pose is known by correspondence. Then, both problems, online and offline SLAM problems, need to consider the correspondence, as presented in equations 1 and 2 for online SLAM and Full SLAM, respectively.

$$p(x_t, m, c_t | z_{1:t}, u_{1:t}) \tag{1}$$

$$p(x_{1:t}, m, c_{1:t} | z_{1:t}, u_{1:t}) \tag{2}$$

The previous equations are used to model both SLAM problems as probability equations, where p represents the probabilities of being at specific poses within a map m and correspondence c giving specific sensors measurements z and control actions u.

FastSLAM is an algorithm used to solve the SLAM problem, using a particle filter approach [7] to solve the problem with known correspondences. Using particles, fastSLAM estimates the posterior pose of the robot over the robot path along with the map. This is possible since each of the particles holds the robot trajectory which helps SLAM to solve the mapping problem with known poses. The fastSLAM algorithm is based on the Rao-Blackwellized Particle Filter approach [8].

FastSLAM 1.0, FastSLAM 2.0 and Grid-based FastSLAM are the most common algorithm used to solve this problem. However, FastSLAM 1.0 and 2.0 have the disadvantage that they need to know a landmark position on order to solve the problem. Grid-based FastSLAM does not have that problem, and then it is possible to model arbitrary environments.

The Grid-based FastSLAM algorithm is based on the Sampling Motion, Map Estimation and Importance Weight techniques. Sampling motion is used to estimate the current pose of the robot given $k-th$ particles previous pose and the current controls u, while Map Estimation is about the estimation of the current map given the current measurements, the current $k-th$ particle pose and the previous $k-th$ particle map. Moreover, Importance

Weight estimates the current likelihood of the measurement given the current $k-th$ particle pose and the current $k-th$ particle map.

iii. Graph based SLAM approach

Graph based SLAM approach, or GraphSLAM, is a SLAM algorithm that solve the full SLAM problem. The Graph name of the algorithm is because it uses graphs to represent the robot's poses and its environment. As can be seen in the following figure, where robot poses are represent by triangles, while features from the environment are represent with stars. Additionally, motion constraints tie two poses and are represented by a solid line. Moreover, measurements constraints tie together a feature and a pose and are represented by a dashed line.

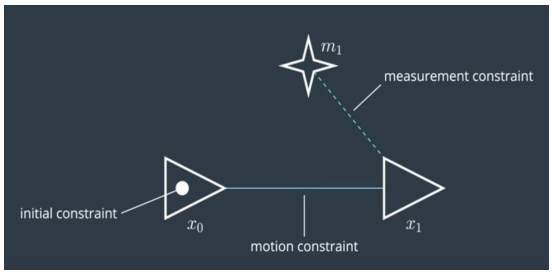

Figure 4: *Graph based SLAM approach.*

Graph SLAM is used to create a graph of all robots poses and features found in the environment, while finding the most likely robot's path and map of the environment. An example can be seen in the following figure:

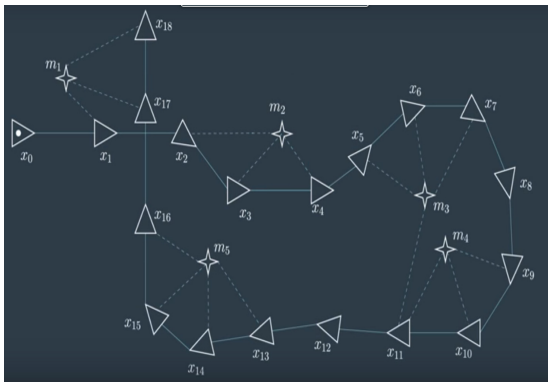

Figure 5: *Graph SLAM map.*

The Graph based SLAM approach can be divided into the Front-End and Back-End sections. The Front-End part of the GraphSLAM algorithm involves the construction of the map graph using odometry sensors and sensory measurements acquired by the robot. The Back-End part of the GraphSLAM algorithm uses all available information obtained in the Front-End part of the algorithm in order to find the system configuration that produce the smallest error, it finds the most probable configuration of the robot poses and map features.

Real-Time Appearance-Based Mapping or RTAB-Map approach is an algorithm that uses data collected from visual sensors to localize the robot and map the environment. In this algorithm a process known as loop closures is used to determine where the robot has seen a location before. However, the complexity of this algorithm increase linearly as the robot move to new areas of the environment, since the number of images that are compared increases.

Nonetheless, the RTAB-Map algorithm is optimized for large-scale and long-term SLAM problems, by using multiple strategies to allow for loop closures to be done in real-time.

III. Model Configuration

The robot used for this project is based on the model of a robot implement in a previous project. The robot is composed by a main frame in which is located a rgb-d camera and a hokuyo laser scan. Additionally, below the frame is locate an Odometry sensor and two wheels are located on each of the side of the robot allowing its movement through the environment. More details about the robot model can be observed in figure 6.

The sensors used on the robot were selected for the implementation of the previously mentioned algorithms. For example, in the project based for the robot model it was used a rgb camera, but that was changed for a rbg-d camera in order to use the GraphSLAM algorithm.

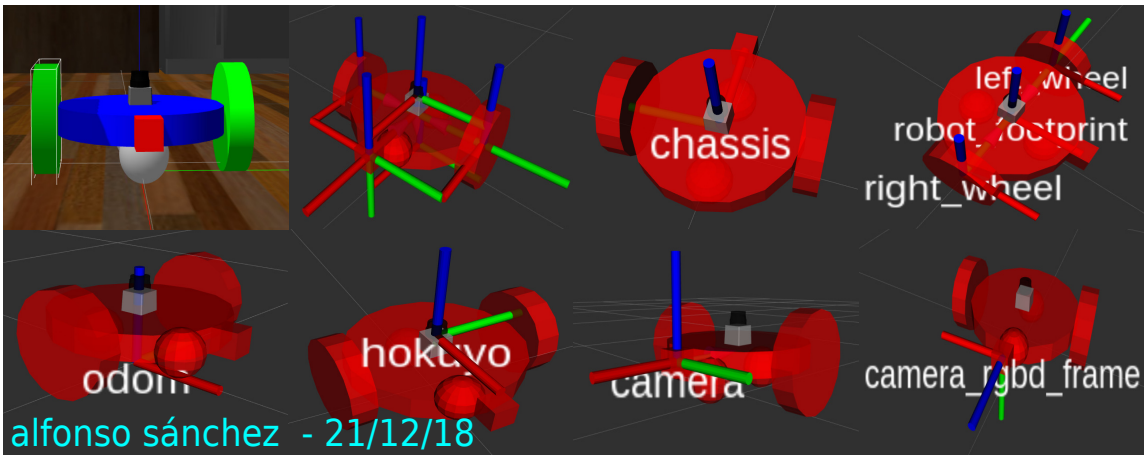

Figure 6: *Model of the robot used in the project. From left to righ: model of the robot, frames, chasis, wheels and main frame, odometry sensors, laser sensor, camera frame and rgbd camera sensor.*

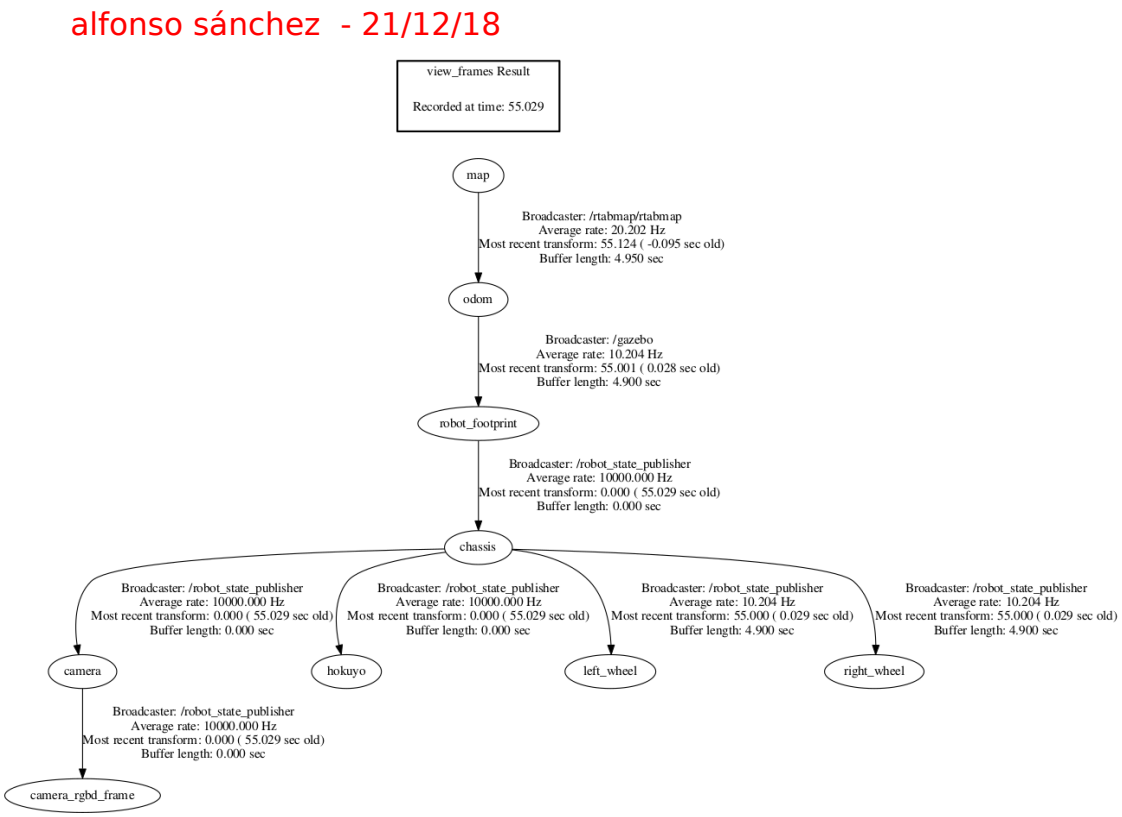

Figure 7: *Coordinate frames for the implemented robot, or **tf** tree.*

The frames for the robot model used can also be observed in figure 7, generated using the ROS tf library. It can be observed the same frames as in figure 6 and their relation that exist between them.

IV. World Creation

The mapping and SLAM algorithms tested in this project were implemented in two different environments, one provided from Udacity and one created specifically for this project. The first environment is composed by a kitchen and a small living room, while the second scenario is a parking lot composed by trees and vehicles.

Figure 8: *From top t bottom: world used for the first scenario and world used for the second scenario.*

The specific world created for this project contains two pick-ups, one polaris-ranger vehicle, several pine trees, one oak tree, a dumpster and many constructions items, located over an asphalt plane and delimited by walls, as can be seen in the previous figure. The repetition of items was to test the perceptual ambiguity effect in the implementation of the SLAMS algorithms. The result presented in the following sections shows that the SLAM algorithm works correctly in the chosen design for the environment.

V. Results

Result are presented for each of the environment used to implement the SLAM algorithms. In both cases a description of the navigation through the environment using the ROS-teleop package is presented, followed by the obtained 2D and 3D maps obtained using the graph-SLAM algorithm by running the mapping launch file. The obtained maps are also presented using rviz. Furthermore, the number of global loop closures are presented as obtained in the rtabmap-databaseViewer.

i. Kitchen_dining world

The navigation through this world is presented in figure 9, obtained by using the teleop package to navigate through the gazebo world while implementing the RTAB-MAP algorithm.

Figure 9: *3D map generated using the RTAB-MAP algorithm and the teleop ROS package.*

An obtained 2D map of the mapped environment is shown in figure 10, obtained in the rtabmap-databaseViewer.

Figure 10: *2D map generated using the RTAB-MAP algorithm and the teleop ROS package.*

In the following figure is presented an example of a loop closure found:

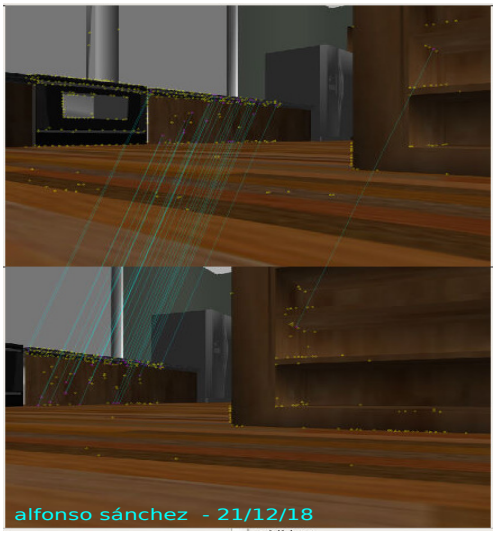

Figure 11: *Example of a found loop closure.*

In figures 12 and 13 are presented the amount of loop closures found in the rtabmap-databaseViewer and the map generated while mapping in rviz, respectively.

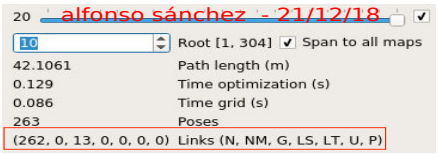

Figure 12: *Number of found loop closures.*

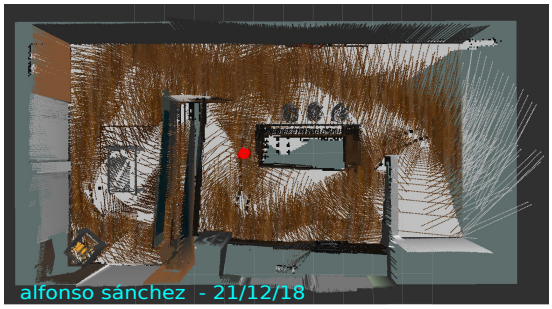

Figure 13: *Rviz map generated using the RTAB-MAP algorithm and the teleop ROS package.*

ii. Parking_lot world

The navigation through the world created specifically for this project is presented in figures 9 and 15, obtained using the teleop package to navigate through the gazebo world while implementing the RTAB-MAP algprithm and the rtabmap-databaseViewer, respectively.

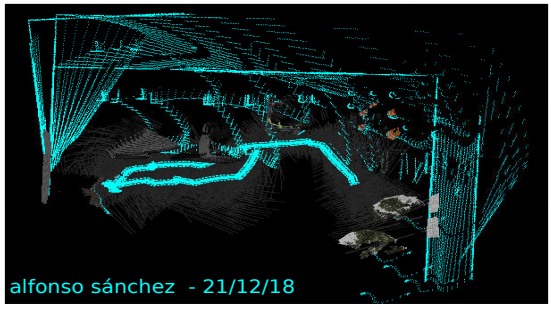

Figure 14: *3D map generated using the RTAB-MAP algorithm and the teleop ROS package.*

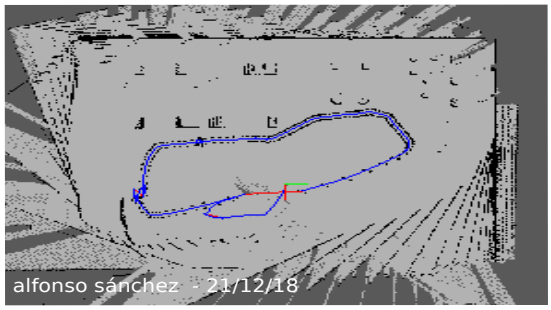

Figure 15: *2D map generated using the RTAB-MAP algorithm and the teleop ROS package.*

In the following figure is presented an example of a found loop closure:

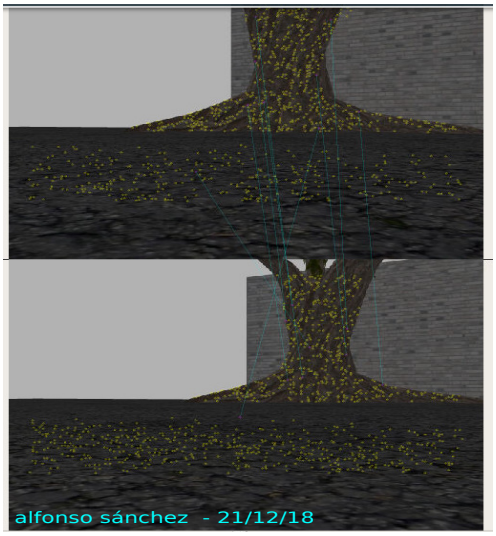

Figure 16: *Example of a found loop closure.*

In figure 17 is presented the amount of loop closures found in the rtabmap-databaseViewer while mapping the parkinglot world.

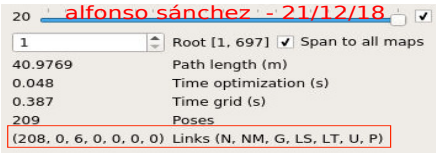

Figure 17: *Number of found loop closures.*

All results presented until this point were obtained using an Online Virtual Machine with ROS provided from Udacity. However, given the poor results obtained for the second scenario, a local implementation of the SLAM algorithm for the second scenario was implemented, with the limitation that the 2D map couldn't be obtained due to missing package of the ROS installation available. However, the rest of the results presented were obtained correctly.

In figures 18 and 19 are presented the obtained 3D maps from the graphSLAM algorithm and using Rviz, respectively, while in figures 20 and 21 are presented the total number of loop closures found and an example of a loop closure, respectively.

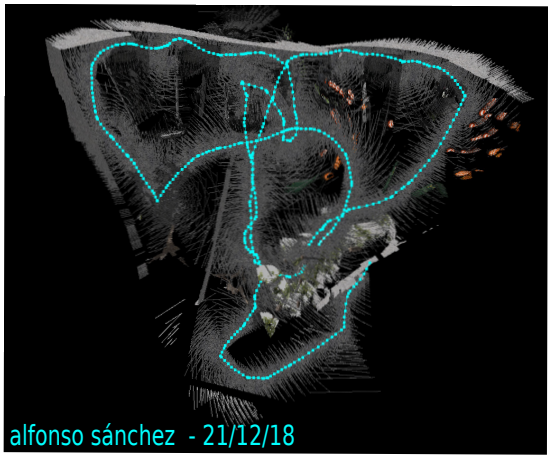

Figure 18: *3D map generated using the RTAB-MAP algorithm and the teleop ROS package.*

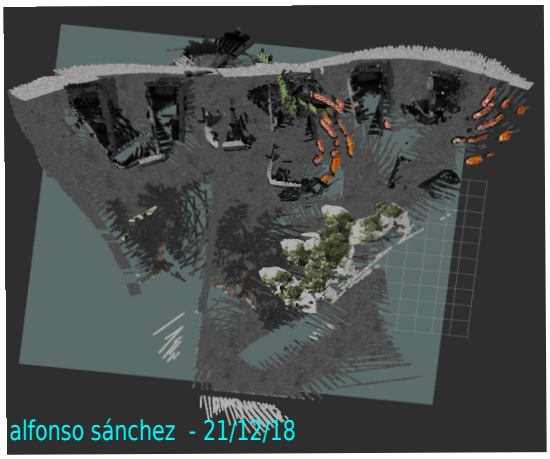

Figure 19: *Rviz map generated using the RTAB-MAP algorithm and the teleop ROS package.*

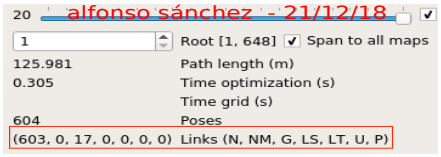

Figure 20: *Number of found loop closures.*

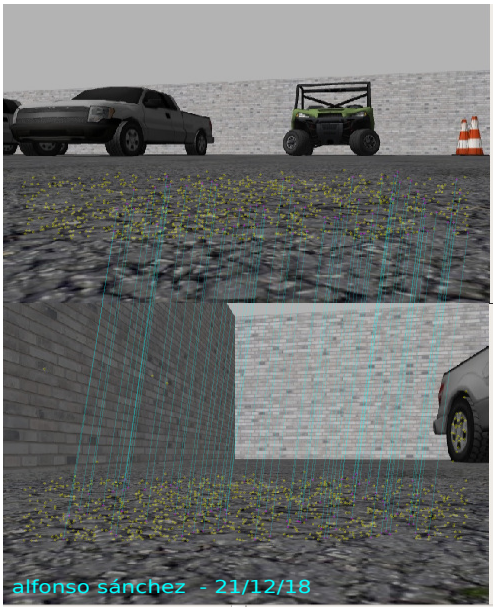

Figure 21: *Example of a found loop closure.*

VI. Discussion

Accordingly with the obtained results, the characteristics of the environment need to be considered when applying the grapSLAM algorithm. This, because of the perceptual ambiguity and the size of the environment. If the environment have similar objects in it, the graphSLAM algorithm gets confuse and misinterpreted the current pose of the robot, failing to map the environment correctly. The same effect happens when the environment it too large, especially if the environment has similar patterns on the walls or on the floor, as was the case for the second presented scenario. This effects can be reduced by mapping the environment several times.

The first introduced scenario didn't present the previously mentioned problem. The results obtained showed that the algorithm was able to localize the robot correclty and to map the environment with enough accuracy. Moreover, more than 3 loop closures were found during the mapping process.

VII. Future Work

The future work related with this project is about the improvement of the graphSLAM implementation, in order to decrease the effects of the perceptual ambiguity generated by repetitive patterns which cause the algorithm to fail mapping the environment correctly. Additionally, results obtained using a graphSLAM algorithm can be compared with results obtained using non-graphical algorithms such as any FastSLAM technique, to have a better understanding of the algorithms limitations.

Moreover, the implementation of the RTAP-MAP algorithm in a Nvidia Jetson TX2 board is an important future work to consider, especially since it will allow a stand alone implementation of the algorithm and therefore, realistic implementations in devices such as RC cars, drones or robots such as sphero. Examples of realistic implementations to consider are the Standford autonomous car used to map environments for autonomous navigations [9] and the TPR-Robina robot which maps a museum every time is reconfigured [10].

References

[1] S. Huang and G. Dissanayake, "Convergence analysis for extended kalman filter based slam," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 412–417, IEEE, 2006.

[2] S. Thrun and Y. Liu, "Multi-robot slam with sparse extended information filers," in *Robotics Research. The Eleventh International Symposium*, pp. 254–266, Springer, 2005.

[3] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based slam," *The International Journal of Robotics Research*, vol. 26, no. 4, pp. 335–359, 2007.

[4] T. Bailey, J. Nieto, and E. Nebot, "Consistency of the fastslam algorithm," in *Proceedings 2006 IEEE International Conference*

on Robotics and Automation, 2006. ICRA 2006., pp. 424–429, IEEE, 2006.

[5] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time for pose graph slam," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1611–1626, 2013.

[6] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous robots*, vol. 15, no. 2, pp. 111–127, 2003.

[7] S. Thrun, "Particle filters in robotics," in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pp. 511–518, Morgan Kaufmann Publishers Inc., 2002.

[8] S. Särkkä, A. Vehtari, and J. Lampinen, "Rao-blackwellized particle filter for multiple target tracking," *Information Fusion*, vol. 8, no. 1, pp. 2–15, 2007.

[9] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments.," in *Robotics: Science and Systems*, vol. 4, p. 1, Citeseer, 2007.

[10] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.