Autonomous Intelligent Navigation for Mobile Robots in Closed Environments

Steven Silva Mendoza[1(✉)], Dennys F. Paillacho Chiluiza[2], David Soque León[2], María Guerra Pintado[2], and Jonathan S. Paillacho Corredores[2]

[1] Facultad de Ingeniería en Mecánica y Ciencias de la Producción - FIMCP, ESPOL Polytechnic University, Escuela Superior Politécnica del Litoral, ESPOL, Campus Gustavo Galindo, P.O. Box 09 -01 -5863, Guayaquil, Ecuador
sasilva@espol.edu.ec

[2] Facultad de Ingeniería en Electricidad y Computación - FIEC, CIDIS, ESPOL Polytechnic University, Escuela Superior Politécnica del Litoral, ESPOL, Campus Gustavo Galindo, P.O. Box 09 -01 -5863, Guayaquil, Ecuador
{dpaillac,disoque,magaguer,jspailla}@espol.edu.ec

Abstract. Providing a map is mandatory for Autonomous Mobile Robots to be able to complete localization and navigation tasks, known as SLAM. Several SLAM algorithms which provides different quality maps have been proposed before but still issues related to map quality can appear while for accurate navigation high mapping performance is desired, therefore to be used in areas regarding health care through delivery and indoor control. For that reason, although several SLAM methods are available, the one provided by Cartographer ROS has been chosen for being one of the most recent, updated ones and has been taken into test with respect to the map quality provided. To accomplish that objective, the implementation of a simulation and experimental environment have been constructed in order to contrast between both mapping, localization and navigation results by using Turtlebot3 and Arlo Parallax platforms including LiDar and encoder sensors, with which the map created by the simulation would be the most optimum map as possible. As a result by using an RPLiDar A1, an acceptable map from the experimental procedure related to the optimized one was acquired. With which could be concluded that Cartographer ROS algorithm is satisfactory to be used for intelligent autonomous navigation purposes by providing high fidelity and effective maps even while demanding affordable computational power.

Keywords: Autonomous mobile robots · Cartographer ROS · Robotics operating system · MicroPython · ESP32 · Social navigation

1 Introduction

Several packages, navigation stacks and solution for AMR[1] have been proposed throughout both previous decade, some of them have been left outdated while

[1] Autonomous Mobile Robots.

M. Botto-Tobar et al. (Eds.): ICAT 2020, CCIS 1388, pp. 391–402, 2021.
https://doi.org/10.1007/978-3-030-71503-8_30

some of them updated continuously. An AMR corresponds to any platform that is aware of its environment and able to navigate through without colliding and finding the right trajectory for different coordinates goals without having to be overseen directly by a operator. Since it is needed for the AMR to recognize its environment, certain resources are requested by it regarding surrounding space features. These features which correspond to a map, distance array structures, and encoder values have its own quality or accuracy, from which one of the most important is a well made map. In the current project, only 2D Mapping is considered [15].

In the case of ROS[2], a map can be created by using different SLAM[3] algorithms and techniques like Gmapping, Hector SLAM and RTAB, each giving a different approach of a map solution but not a definitive one to be used without question. However not too many years ago Cartographer, a Google project which is a system that provides real-time SLAM in 2D and 3D across multiple platforms and sensor configurations was ported to ROS [6]. While seeking for one of the most optimum methods of creating a map using ROS and then use it for autonomous navigation, it was stated as an objective to test one of the SLAM methods. From the ones we researched, found Cartographer ROS to be one of the best and most promising ones, since it uses a global map optimization cycle and local probabilistic map updates which makes the system more robust to changes in the environment [2]. Having in mind that what is most important is to be able to create the map of the space experimentally, it was proposed to analyze Cartographer ROS experimental performance comparing it with simulation, considering the last one as the most precise results. In order to accomplish that, an experimental and simulated environment was needed and in both cases a 2D map were gotten with which navigation was tested.

With a definite algorithm or SLAM package with which a map can be constructed, it could enable easier AMRs implementation for indoor environments a good navigation results that conclude in solutions of automation uses for hospitals, restaurant attention, etc.

2 State of Art

Nowadays there are several SLAM methods and every of them are a different approach of a solution for autonomous navigation, some of them are dependent of some sensors like depth cameras, others might be dependent of what is called laser scan provided by a LiDar. There are many of these SLAM packages that have been with time outdated or left without any optimization. In the case of Cartographer ROS, it looks to be a well optimized and promising method to be used with other components like Move Base[4]. As a stack which is updated continuously. In terms of the results of other SLAM methods, it provides a

[2] Robotics Operating System [3].
[3] Simultaneous Localization and Mapping.
[4] A package provides an implementation of an action (see the actionlib package) that, given a goal in the world, will attempt to reach it with a mobile base [9].

really good algorithm in which the map that is given by it is updated by the use of submaps when it is still running, making it a real-time SLAM package capable of providing effective mapping results, as seen in other investigations and categorized as one of the most accurate [1]. In Fig. 1, it can be seen Cartographer Node structure.

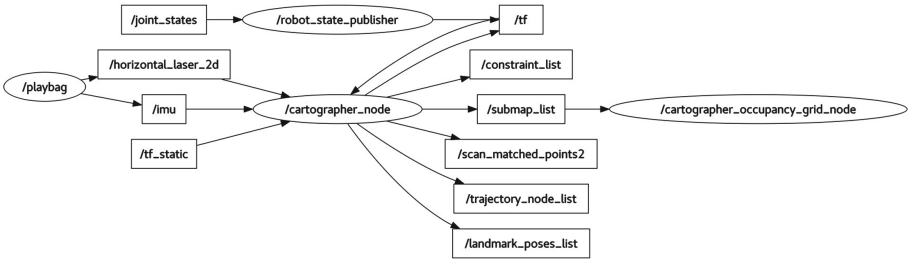

Fig. 1. Cartographer node for 2D SLAM [6].

In order to evaluate the efficiency or quality of the map, there are qualitative and quantitative ways to do it. If quantitative evaluation is requested, GT[5] data coming from simulation needs to be provided as well as the odometry from the trajectory gotten in the experimental proves, with which RMSE error can be calculated between both results as well as absolute and relative pose error [16].

Indirectly the efficiency of movement of a robot is affected by how good the map and the trajectory that in terms of ROS is here is accomplished with AMCL and Move Base. Nowadays there are already some software packages capable of this kind of analysis, one of them is a python module called EVO, a python package which can give several insight about the quality of a trajectory [4].

Qualitative analysis on the other hand can also be accomplished even of it is not as reliable as a quantitative method, and actually in the current project the maps are only analyzed in a qualitative way. One of them, corresponds to the proportion of the map, the blurrier the walls or outline of the map, the worst it is, meaning a well made map should be pretty sharp, as seen in Fig. 2.

Also, another one is the amount of corners the map has, a map with a huge amount of corners is most likely to be inconsistent unless the ones shown are actual corners. This is also related to the amount of enclosed areas, the most enclosed areas the map has, means the map is could fail for navigation in the future since the AMR shouldn't be enclosed in only one single space, taking as enclosed area a space in which the robot is surrounded completely by occupied cells. Mostly it is important to do the analysis with the before stated parameters when no ground truth is available and it is important to consider that these parameters can also be affected because of bad data sensing like odometry which could cause map shifting, for example when turning [8,11].

[5] Ground Truth: corresponds to the most precise trajectory or odometry recording of the robot or moving platform.

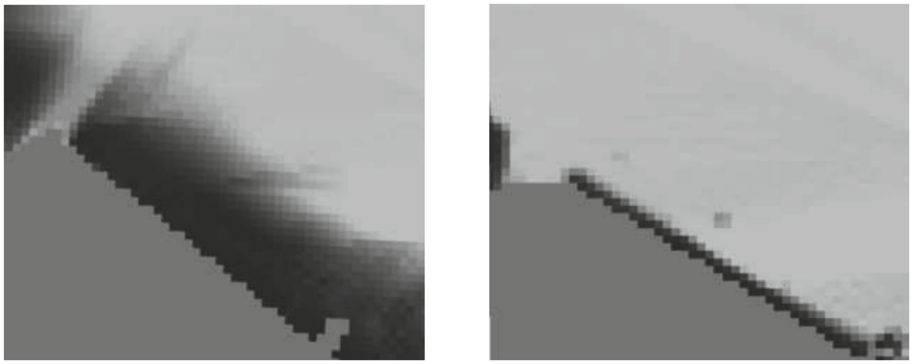

Fig. 2. A map with different proportions [1].

3 Methodology

The project was carried out in two parts; for the first part a simulation with which mapping was made using a simulated environment in Gazebo and a second part in which the same technique was implemented in the commercial ArloBot platform.

For the complete project, a repository in GitLab was established as amr-navegación. Also, in order to accomplish lots of part related to the configuration of the project, an urdf defining the ArloBot platform [7] and a wiki about Cartographer ROS with Ouster LiDar [12] were used as resources.

3.1 Simulation

Mapping Environment Setup. A navigation scene was recreated, specifically the interior of a house, trying to add details of significant size such as walls, pillars, furniture doors, dining table and chairs; by using Onshape, CAD software for collaborative drawing, supporting 3D design. Subsequently, it was necessary to carry out a conversion of the final assembly of the drawing from stl to sdf, which is one of the extensions that Gazebo handles in the software used by ROS to simulate scenarios and the interaction of the robot with it. In order to accomplish that conversion, Onshape to Robot was used. Once the stage is exported into the gazebo as a model, the settings are made to select this model as the world of our simulated robot. The resulting simulated environment exported with Onshape to Robot from Onshape can be seen in Figs. 3 and 4, according to the experimental one in Fig. 5.

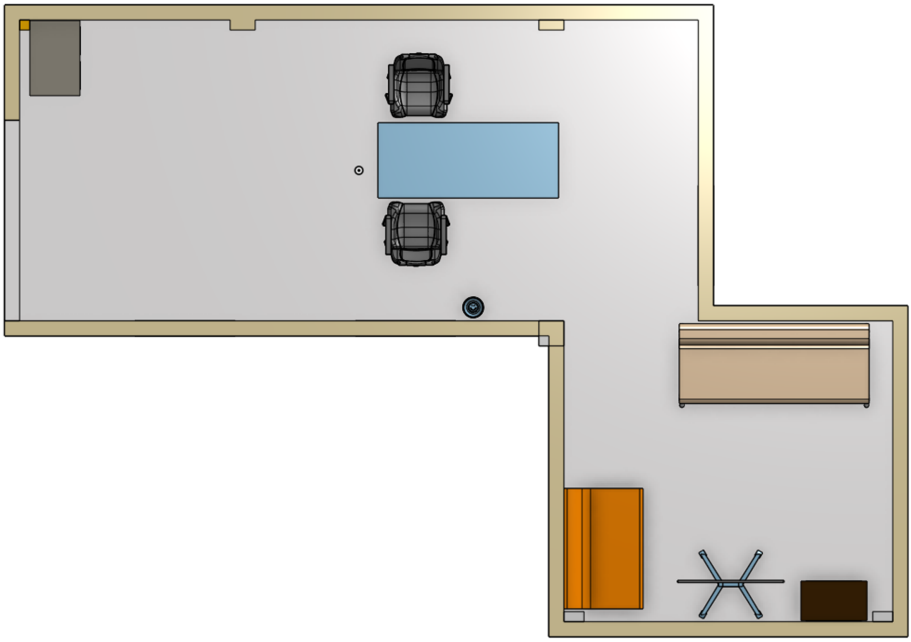

Fig. 3. Superior view of the simulated environment.

Cartographer and Robot Setup. The installation of the Cartographer ROS project was made in the workspace intended for both simulated tests and implementation. The necessary steps to follow are indicated in the GitLab repository.

Afterwards, Turtlebot3 was used as the simulated robot for the simulation. Since there wasn't a package in order to simulate ArloBot in gazebo, Tutlebot3 Burguer was considered for the purpose since it was the most similar to ArloBot.

Also other packages like `differential_drive`, `move_base` and `rplidarROS` were installed in the workspace in order to provide the data with defined messages type that Cartographer Node requests.

Launch and Config Files. In order to start the mapping, different launch files were created in which every needed node for the simulation was included. In between these is the `gazebo spawn node`, `rviz`, `move base` and `amcl`. Also `.lua` configuration files were stated to have the best result from cartographer. All the last components were needed in order to recreate, visualize, move and localize the AMR in simulation and experimental procedures [5,10].

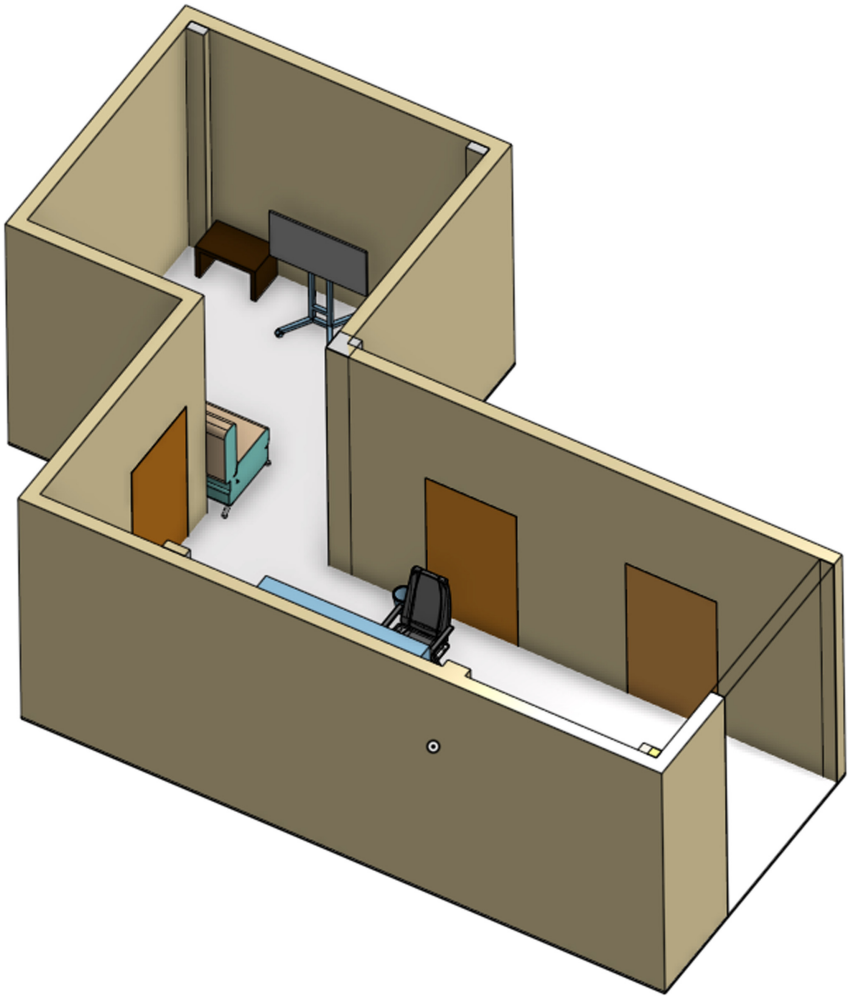

Fig. 4. Recreated experimental environment.

The structure defined in Fig. 6 was the one used in order to define all the project regarding the simulation. It was using Draw. Io including every package and middleware needed in order for the simulation to work properly. It is worth saying that red arrows correspond to *Publishers* and green arrows to *Subscribers*.

Fig. 5. Experimental physical environment.

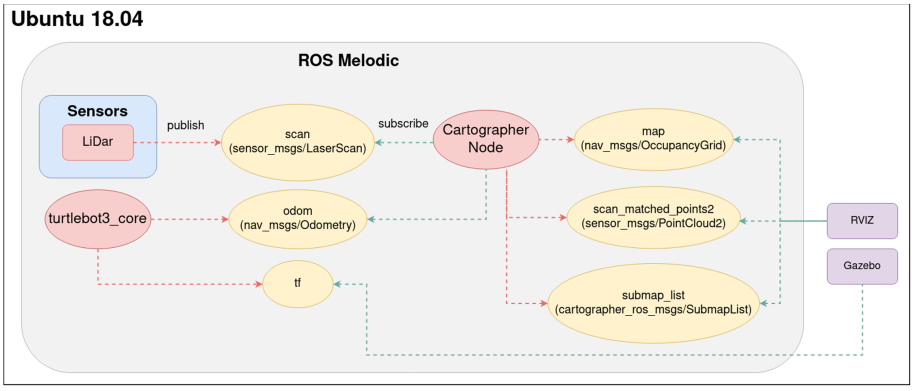

Fig. 6. Simulation architecture.

3.2 Experimental

For the experimental part, most of the launch and configuration files were copy pasted since ROS lets to use almost the same file and packages structure for simulation and experimental practices. However, it is still important to point

out as defined Fig. 7, where other packages needed to be included in order to let the interaction with hardware happen.

ESP32 and MicroPython Setup. The most significant difference between the simulated and experimental part, is that in the experimental one it is needed to program a microcontroller to communicate with ROS master in order to provide information about certain sensors like encoders but also control the platform, in this case, the driver for the motors which is a DHB-10.

The ESP32 was used for this task for being a really robust and efficient microcontroller. In regard to MicroPython, it was used to rely on only one programming language since ROS is also mostly used with Python and additionally because it is a fairly easy language to use. In order to establish communication with ROS from the microcontroller, *rosserial* was proposed, with which uros[6] was used. And to control the DHB-10 driver a module called uPyArlo[7] was installed as well, which would let encoder readings and speed commands writings.

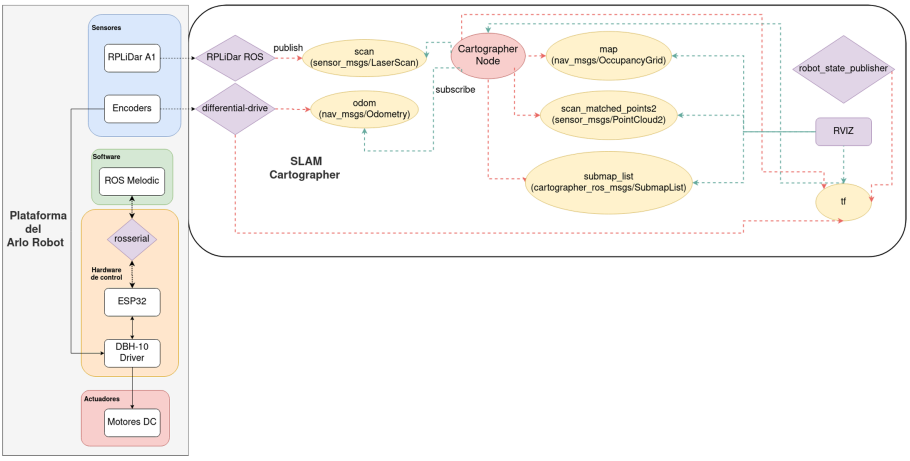

Fig. 7. Experimental architecture

4 Results and Discussion

4.1 Simulation

The resulting map obtained from simulation environment can be seen in Fig. 8. It was obtained by using `map_saver` command from the package `map_server` after

[6] A MicroPython module developed to be used with rosserial [14].
[7] A MicroPython module developed to control DC motors through DHB-10 driver [13].

going through the mapping process. Looking at its outline it is a fairly good result of a map. Considering the width of the walls in the map, corresponding to proportion, they show to be pretty thin, well defined and sharp, which is a good sign and helps with the accuracy of a good navigation. Some parts of the outline show to have certain anomalies in the outer part of the map, but they shouldn't affect the navigation since the robot can't actually go through that area.

Now considering the amount of corners presented in the map, there are almost none of them, the corners that are presented are mostly because the environment is actually cornered, they are really well closed and defined sufficiently. It is worth considering that some spaces were left opened because the laser from the LiDar couldn't go in such small spaces, which is then limited by hardware and not the SLAM algorithm. These opened areas should be left opened however in the current testing it is almost impossible to accomplish that matter since it is mostly a matter of how the environment was defined, for example, to not have those opened areas, the sofas from the environment should be put as close as possible to the walls.

In regard to the enclosed areas, it also shows a good sign, no enclosed areas are presented and the platform is able to go through all the map without restrictions. That is because the map doesn't show any shifting anomaly in any way. It could be said that it represents a perfect map in the context of others maps resulting from other SLAM techniques.

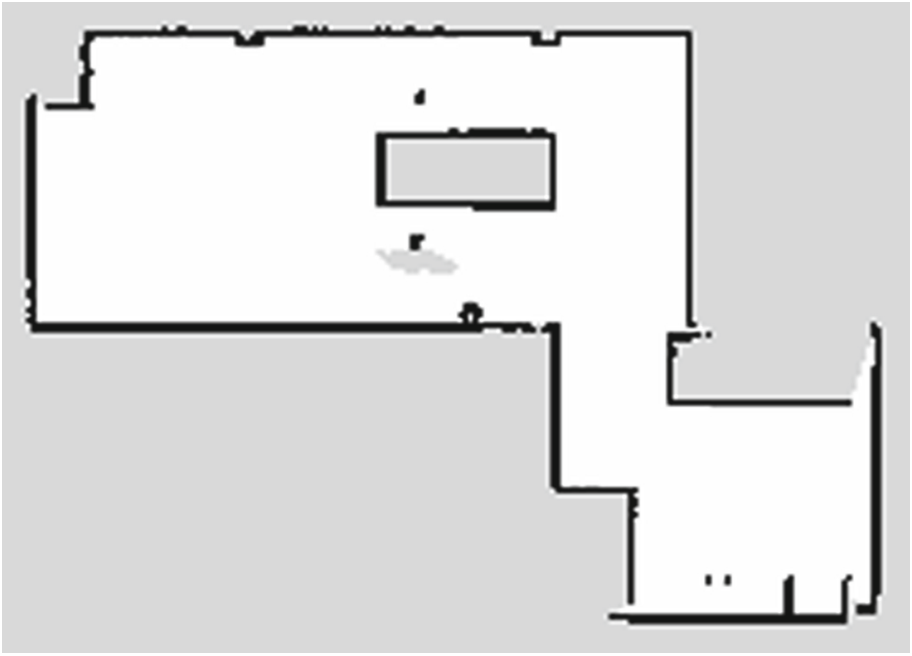

Fig. 8. Map obtained from simulated environment.

4.2 Experimental

In the case of Fig. 9, exported by using `map_server` as well, the first thing to take into account is that the map is not actually aligned with the simulated one, which is basically because of how the AMR started the mapping process. Now in terms of proportion, it is clearly that the experimental map is not as good as in the simulation, there are certain parts of it that not that neat and perfectly defined. Some of the causes could be the colors and certain disturbances from the walls, like irregular surfaces. Other than that, is shows a good sign since either way the width of the outline is thin. It should also be considered, that in this case a real LiDar was used which has a lot more errors and less accuracy than a simulated one and still the proportion of the outline is satisfying enough.

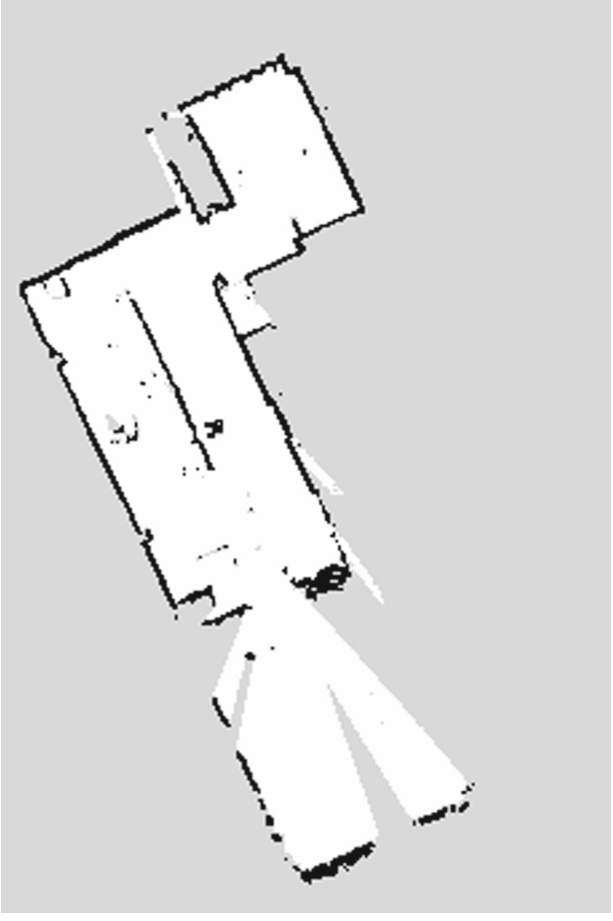

Fig. 9. Map obtained from experimental environment.

Regarding the amount of corners, this map has more corners than Fig. 8, but it does not reduces the quality of the map significantly because there aren't too much of them. Most of them can be spotted in the outline of the walls, where little spikes are shown. Also the actual corners from the environment were mapped and are shown convincing.

Related to the enclosed areas, none were created, as it can be seen from the map, it is completely open and the AMR can drive freely through it as in the simulated one. Although the map fulfills the parameters for a qualitative analysis, it has some anomalies that should be pointed out. One of them is at the inferior part, there seems to be an open part in which the map is not complete, that error is because at that area a complete window replacing a wall is placed which wasn't sensed by the LiDar leading to an open section. Another one can also bee seen at the right part, where unoccupied cells as a wide line outside the outline is shown, an error caused because of little open edges in between the door. Apart from that, the map is accurate and comparable to the simulated one, the edges and irregularities it were caused because of the hardware used and some differences between the simulated and experimental environment.

5 Conclusion

Both simulated and experimental results for mapping by the use of Cartographer ROS were shown and while analyzing the maps provided by qualitative methods, it could be concluded that Cartographer ROS is able to provide a really well made maps even when using not the most expensive hardware resources and that the efficiency doesn't vary significantly in between the simulation and experimental procedures. While using a RPLiDar A1 sensor, a considerably cheap LiDar, still no much difference was seen considering that in simulation, the laser sensor plugin from Gazebo is as accurate as possible, as well as in the best condition.

By looking at the results, it can easily be seen the maps are not of the same quality, but that doesn't mean the experimental solution can't be used nor that it can't compete with the simulated results, actually it is a really satisfying approach while the errors gotten are not enough to reduce its usefulness at all.

It is possible for the reason that this SLAM method works with submaps made while it goes through the mapping process, it is able to update the current published map and optimizing already mapped areas continuously with previous Laser Scan data samples, and without using much CPU power. Meaning that Cartographer ROS is one of the best approaches to be used when implementing a SLAM technique in order to acquire autonomous navigation.

References

1. Anton, F., Artyom, F., Kirill, K.: 2D slam quality evaluation methods, August 2017
2. Filipenko, M., Afanasyev, I.: Comparison of various slam systems for mobile robot in an indoor environment, August 2018

3. Foundation, O.S.R.: Ros documentation. https://wiki.ros.org/
4. Grupp, M.: evo: Python package for the evaluation of odometry and slam (2017). https://github.com/MichaelGrupp/evo
5. Hershberger, D., Gossow, D., Faust, J., Woodall, W.: Rviz. https://wiki.ros.org/rviz
6. Hess, W., Kohler, D., Rapp, H., Andor, D.: Real-time loop closure in 2D lidar slam. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1271–1278 (2016)
7. Lofland, C.: Arlobot. https://github.com/chrisl8/ArloBot
8. Longhi, R., Fabro, J.: Ros navigation: concepts and tutorial, February 2016
9. Marder-Eppstein, E.: Move base. https://wiki.ros.org/move_base
10. Marder-Eppstein, E.: Navigation. https://wiki.ros.org/navigation
11. Santos, J.M., Portugal, D., Rocha, R.P.: An evaluation of 2D slam techniques available in robot operating system (2013)
12. Selby, W.: Building maps using google cartographer and the os1 lidar sensor. Technical report, October 2019. https://ouster.com/blog/building-maps-using-google-cartographer-and-the-os1-lidar-sensor/
13. Silva, S.: Micropython arlorobot. https://github.com/FunPythonEC/uPyArlo
14. Silva, S.: Micropython rosserial. https://github.com/FunPythonEC/uPy-rosserial
15. Wonnacott, D., Karhumaa, M., Walker, J.: Autonomous navigation planning with ros
16. Yagfarov, R., Ivanou, M., Afanasyev, I.: Map comparison of lidar-based 2D slam algorithms using precise ground truth, November 2018