# Autonomous 2D SLAM and 3D mapping of an environment using a single 2D LIDAR and ROS

**Conference Paper** · November 2017

DOI: 10.1109/SBR-LARS-R.2017.8215333



**4 authors:**

Manuel Gonzalez Ocando
Simon Bolívar University
**2** PUBLICATIONS **6** CITATIONS
SEE PROFILE

Novel Certad
Simon Bolívar University
**13** PUBLICATIONS **39** CITATIONS
SEE PROFILE

Said Alvarado
Simon Bolívar University
**2** PUBLICATIONS **6** CITATIONS
SEE PROFILE

Ángel Terrones
Simon Bolívar University
**6** PUBLICATIONS **28** CITATIONS
SEE PROFILE

# Autonomous 2D SLAM and 3D Mapping of an Environment Using a Single 2D LIDAR and ROS

Manuel González Ocando[1], Novel Certad[2], Said Alvarado[3] and Ángel Terrones[4]

***Abstract*— This paper describes an algorithm that performs an autonomous 3D reconstruction of an environment with a single 2D Laser Imaging Detection and Ranging (LIDAR) sensor, as well as its implementation on a mobile platform using the Robot Operating System (ROS). A ROS node was used to redirect the flow of data that can go to either the 2D Simultaneous Localization And Mapping (SLAM) ROS node or to the 3D Octomap ROS node depending on the operation performed at that moment, with neither of the nodes going out of sync or crashing. The autonomous algorithm is implemented with the State Machines (SMACH) library and it uses ROS interfaces such as services and actions to create a 3D model of the robot's surroundings without prior information or human intervention.**

## I. INTRODUCTION

While there are many robots that can build 3D reconstructions of an environment, very few of them are able to do so autonomously. Usually, an operator guides the data acquisition process, or there is enough data previously gathered for the robot to avoid any obstacles when navigating on a map while collecting data of the environment.

In order to make an autonomous 3D mapping solution, the problem was divided into three parts: (1) the Simultaneous Localization and Mapping (SLAM) that will allow the robot to be aware of its surroundings and it's location while it moves [1][2], (2) the 3D mapping algorithm that takes the sensor data to make a 3D model of the environment, and (3) the algorithm that operates the robot to navigate the environment avoiding any obstacles and to get all the necessary data without human intervention.

There are many implementations of 2D SLAM and 3D mapping robots with different sensor configurations. The vast majority of hardware configurations employs a dedicated sensor for 2D SLAM, along with other sensors used to gather 3D data. The solution described in this paper utilizes a 2D Laser Imaging Detection and Ranging (LIDAR) sensor as the single input to register all the data to make a digital 3D reconstruction of flat environments without previous information or human intervention.

[1]Manuel González Ocando is an undergrad student of Electronical Engineering, Universidad Simón Bolívar, Sartenejas, Miranda, Venezuela manuelgo111@gmail.com

[2]Novel Certad is with the Department of Electronical Engineering, Universidad Simón Bolívar, Sartenejas, Miranda, Venezuela ncertad@usb.ve

[3]Said Alvarado is an undergrad student of Electronical Engineering, Universidad Simón Bolívar, Sartenejas, Miranda, Venezuela said_alvarado@hotmail.com

[4]Ángel Terrones is with the Department of Electronical Engineering, Universidad Simón Bolívar, Sartenejas, Miranda, Venezuela aterrones@usb.ve

## II. HARDWARE

The LIDAR used was a Hokuyo URG-04LX-UG01. This sensor has 4 meters of range and records data in a semi-circumference of 240 degrees. It is basically a 2D plane of data, which can be converted into a 3D volume of data with the proper rotations and translations of the sensor.

A servo-controlled gimbal was designed and 3D printed to rotate the sensor to any desired orientation. By knowing the sensors pan and tilt, it is possible to extrapolate the 2D data into a 3D sphere. A servo controller with a RS232 connection made by Lynxmotion was used. Moreover, to operate the servos and relay the data obtained from the LIDAR, a Raspberry Pi 3 was used. This small card-sized computer helped bridge the data from the sensor to the PCs doing the 3D reconstruction.

Additionally, a wheeled platform was necessary to move the LIDAR with its gimbal through the environment. The platform chosen was the AmigoBot, a robot from Omron Adept MobileRobots, LLC. This robot receives commands through WiFi and provides a stable platform for the LIDAR sensor, making it wireless. A diagram of the different connections of the hardware components is presented in Figure 1, and the fully assembled robot can be seen in Figure 2.

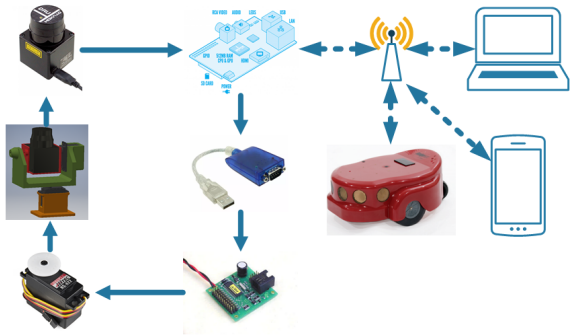

Fig. 1. Hardware connections diagram.

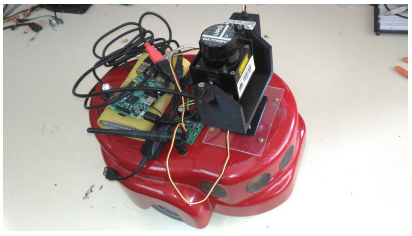

Fig. 2. Fully assembled robot.

## III. ROS SOFTWARE

The Robot Operating System (ROS) [3] framework was extensively used in this work since it allows to interconnect programs written in multiple languages and running on multiple machines in a seamless way.

Usually, nodes (ROS processes) interact with one another in the following ways:

- Messages, each sent in a specific channel called topic. This is the main form of communication in which one node passes data to many others in a non-blocking way.
- Services, which offer a one-to-one communication where a node makes a request to another node and waits for an answer.
- Actions, which are a standardized interface for pre-emptable tasks offered by server nodes. These are implemented for complex tasks that take a long time, and can be requested or canceled by a client node.

As this framework is used by many institutions worldwide, it contains node packages made by many developers. Hence, there are lots of code written to perform a specific task that are available to the public and can be used in a relatively fast and easy manner. A simplified diagram of the principal nodes implemented in this work is given in Figure 3.

### A. Third party nodes

Among the nodes created by other developers that were necessary in this work, the following are worth mentioning:

- URG Node: It is a node that takes the data from the LIDAR and broadcasts them to other nodes.
- Laser Filters: This node that takes the raw data from the LIDAR and filters out noise and outliers.
- Robot State Publisher: Node that collects information about the joints of a predefined robot and uses that data to calculate the corresponding frames of reference [4] for the sensors.
- RosAria: This node allows other programs to control the AmigoBot robot with a simple vector indicating the desired linear speed.
- Move Base: Part of the navigation stack of ROS. This node allows a robot to navigate its surroundings while avoiding obstacles to reach a fixed destination.
- Rviz: It is a visualization node used to observe the robot's 3D reconstruction of the environment.
- Smach Viewer: Node that allows viewing the current state of the State Machines (SMACH) [5] implemented with the Smach library.

Furthermore, three third party nodes must be examined in greater detail.

First, the Cartographer node. This is a SLAM algorithm developed by Google [6]. This node takes in the laser data from the LIDAR and, without additional data, it is capable of making high quality maps with loop closure, with a fairly low usage of CPU and memory.

There are many SLAM nodes in ROS [7]; however, the ability of Cartographer to use only the laser data to make its maps, without any obligatory requirement to utilize IMU or odometry data in the majority of use cases, is exceptional. Although extra data for sensor fusion is not required, it might be useful, especially in very regular environments like hallways, where that extra data comes in handy.

This node allows the robot to move through any flat environment without losing track of its whereabouts. By doing so, all the data gathered to make a 3D reconstruction can have a frame of reference to correctly position its measurements. An image of a 2D SLAM made by this robot using Cartographer node can be found in Figure 4.

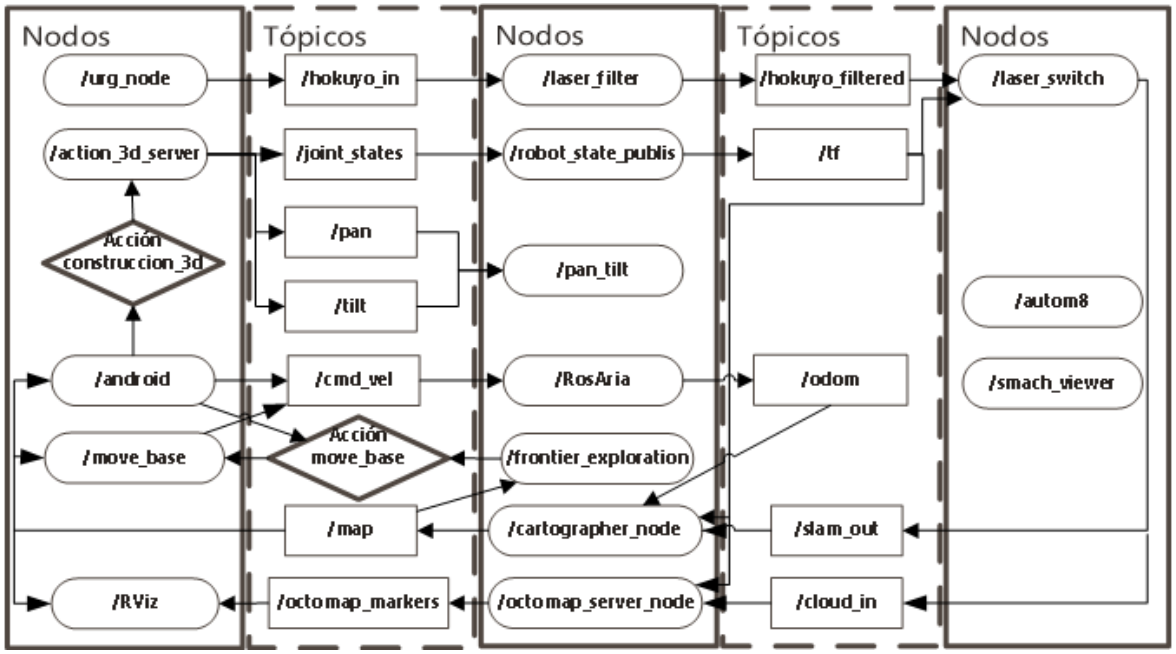


Fig. 3. ROS nodes connections.

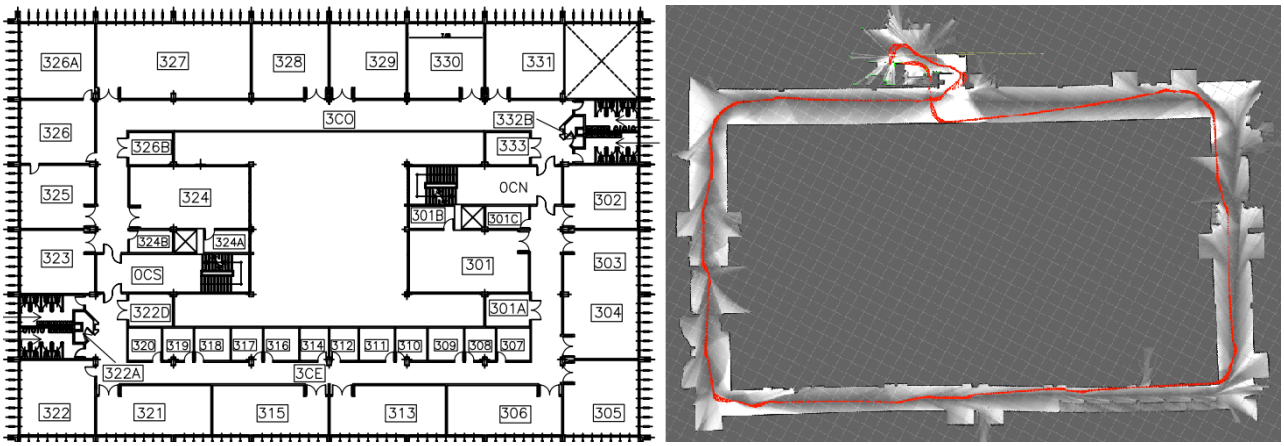


Fig. 4. Comparison of a hallway floor plan (left) against the SLAM map generated by Cartographer Node (right).

The second notable third party node is the Octomap Server node [8]. This node takes in a data point cloud and uses it to make a volumetric 3D environment model using octrees, which is called an octomap.

This node creates an octomap from data collected from the LIDAR. With these laser scans, plus the information of the frame of reference of the robots which is calculated from the servos position at any given time, the node creates a 3D map that explicitly represents not only occupied space, but also free and unknown areas.

Octomaps are an ideal way to model a 3D environment, as they offer a good balance between reconstruction fidelity and memory usage, in addition to being a probabilistic model which is very advantageous.

The third node is the Frontier Exploration node. This node implements the algorithm described in the paper by B. Yamauchi (1997) [9], which allows the node to detect the frontiers of unexplored areas in a map. It contains an action server, and when it receives an action goal to explore, it will automatically checks for missing borders of the map and give the commands to the Move Base node to go to specific locations, which in turn will put the robot into motion while avoiding any obstacles on its way.

The Frontier Exploration node is a great solution to create a 2D map of an environment, and it is already implemented. The only necessary changes to use it is to connect it to the right topics.

### *B. Developed ROS nodes*

Although ROS has a great number of generic nodes and algorithms available to the public, not all specific use cases have a ROS implementation yet. Therefore, it was imperative to create some nodes specifically for the autonomous robot of this work. [10]

First, the Pan-Tilt node. This node takes two real integers, one for the panoramic angle and another for the tilt angle, and sends a command through the RS232 interface to the servos controller to move the servos so as to reorient the gimbal to the desired inclination.

Second, the Action 3D Server, a ROS node that implements an action server to operate the servos that move the gimbal of the Hokuyo LIDAR. When an action client request a 3D scan, this node sends the angles to the servo node so that this second node moves the sensor slowly until it takes a 3D bubble of data with a diameter of twice the sensor range.

Third, the android node, a node heavily based on the android core package. It is used to initiate the automatic 3D reconstruction or to stop it in case of emergencies, and it is also useful to oversee the robot localization and environment mapping live.

Fourth, there is the Laser Switch, a ROS node capable of redirecting messages. This is the node that makes the simultaneous 2D and 3D mapping with a single sensor possible. It implements the laser geometry package of ROS, which contains functions that transform the laser scan input into a point cloud.

The feature that sets de Laser Switch node apart from the simple implementations of laser geometry is that it contains two outputs for the laser data, one for the 2D SLAM node and another for the 3D SLAM node. Also, a service offered by this node is used to determine what data is transmitted to each node in a specific moment.

There are three possible kinds of outputs to each of the two nodes: (1) relay the information coming from the LIDAR without alterations, (2) relay a particular message to a topic over and over again with variations only in the time stamp, and (3) stop sending data to an specific topic. By using the service provided by this node, the flow changes between this three settings.

With these nodes, and some careful planning, the flow of data should be indistinguishable from the original one-to-one connection, allowing the 2D and 3D algorithms to receive the data that was meant for them and filtering out the messages that would cause error to any of these algorithms.

Fifth, the Autom8 node, the autonomous mapping algorithm that implements the SMACH library for state machines. This node oversees all other nodes and handles the global objectives for the reconstruction.

## IV. Autonomous mapping algorithm

This autonomous mapping algorithm was inspired by Google Street View, a software that allows a person to visualize many parts of the world in small 3D photos spheres. These spheres are simply 360 degree pictures taken periodically from a moving vehicle. The model of the environment generated by that method is not a continuous data set, but the data spheres are close enough to offer users a good 3D visualization of panoramas in which they can navigate.

For this work, this approach was examined and then adapted to a single 2D LIDAR sensor. When the robot is moving, the 2D plane of laser data coming from the LIDAR must be parallel to the floor. That way, the 2D Cartographer node can observe the surroundings and fulfill its purpose: to locate the robot and to map the environment. While this is happening, the Octomap node does not receive any data to preserve CPU resources.

When a request is sent to the 3D Action Server to perform a 3D scan, this node carries out the following functions: first, it disables the robot motors through a ROS service so that the robot is stationary; next, it signals the Laser Switch node to change the flow of data coming from the LIDAR. As a result, all the data goes to the Octomap node instead of heading to the 2D SLAM. Then, the servos are moved to collect data from all around the vehicle. While this occurs, the SLAM node can receive (depending on the SLAM node requirements) either no new laser scan data or the last data gathered while the sensor plane was parallel to the floor with a fake timestamp.

The main advantage of this data redirecting approach is that the 2D and 3D nodes can operate seamlessly, without interruption, and without interfering with one another. The data from the tilted LIDAR that would have thrown off the SLAM node is redirected, and since the robot is not moving, once the LIDAR returns to its original position, the flow of LIDAR data is restored to the SLAM node, which will continue to operate without detecting any discontinuities.

Also, for this approach to work, the ROS tree of reference frames had to be modified. ROS uses this tree to calculate the position of the sensors and their orientation. In order to fully deceive the nodes into interpreting the data as they should, the tree of reference frames had to be divided into two branches: one is constantly operating as if the laser were always pointing forwards, and the other one is updated according to the commands given to the servos to depict the real orientation of the LIDAR, as can be seen in Figure 5.

However, this arrangement between the 2D SLAM node and the 3D Octomap node does have a drawback: all the frames of reference and the map depend completely on the SLAM node. This node is continuously rearranging the map based on new data, especially during a loop closure event. Consequently, any data regarding the reference frames of the incomplete 2D map that are taken by the octomap will be incorrectly placed. This is the reason why great care must be taken to begin the 3D data gathering only after the 2D map is complete.

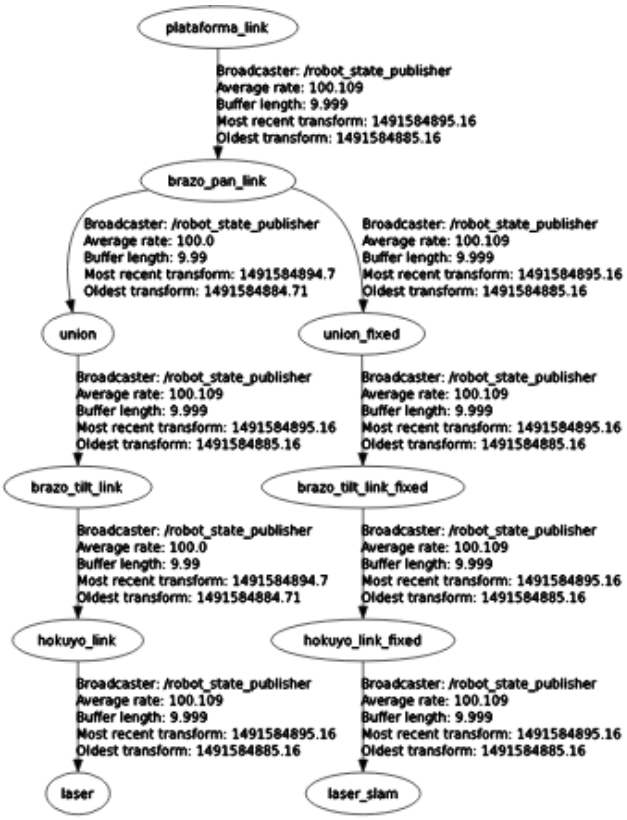


Fig. 5. Reference frame tree.

With this approach, simultaneous 2D SLAM and 3D Octomapping using a single sensor is possible, and the switch node provides a simple service for other nodes to use. Moreover, there is an action available to move the robot to any navigable location, and also an action available to carry out a 3D scan centered on the robot current location. Then, all that is left for a node to do is to use these two actions and the previously mentioned service to autonomously create a 3D model of the environment.

This is where the Autom8 node comes in. This node is a Python process that oversees the work of the rest of the nodes. It uses the SMACH library, a ROS-independent Python library to build hierarchical state machines. This facilitates the task of making a 3D reconstruction of the environment by dividing it into smaller tasks that when assembled become the autonomous module. Through this node, every action or service available in the ROS computing graph can be executed at any time. Also, it can start a node's execution or halt it to preserve computational resources.

The Autom8 process was divided into two blocks. First, the automated exploration seen in Figure 6. This oversees the 2D SLAM map generation, performing the following tasks:

- Wait for nodes initialization.
- Execute the frontier exploration node and wait for it to begin operation.
- Wait for SLAM node's correct execution.
- Signal the frontier server to begin automated explo-

ration.

- Stop the frontier node once the exploration has finished.
- Take note of the resulting map size.
- Save the robot's current position.

Each state requires the previews one to be successful. Once this block finishes, a complete 2D map of the environment should be available.

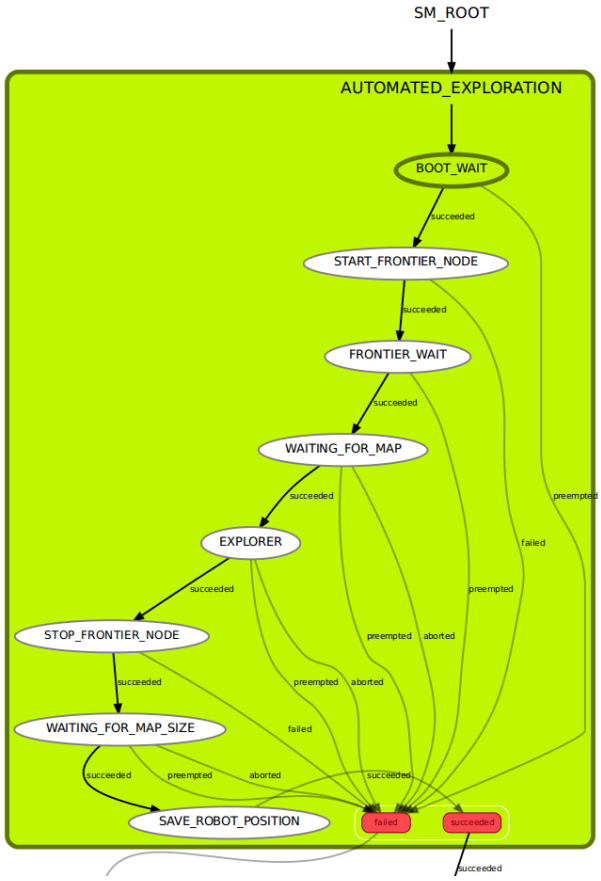


Fig. 6. Autonomous 2D SLAM.

From the data provided by the previous block, the second block begins operation. This one is in charge of the Octomap generation, as seen in Figure 7, and performs these tasks:

- Take the map size and the robot position, and use the navigation node called Move Base as a means to assess whether the robot will be able to reach a certain point in the map. This state will continuously request a navigation plan based on current data to mark a subset of equally spaced points on the map as reachable or not.
- Select the most appropriate points to perform the 3D scans once all the navigable points are known. This was done using a modified Farthest First Transversal Algorithm [11]. To make a Delone Set of the 3D scan points, as illustrated in Figure 8, the algorithm was designed so that the distance among all selected scan points was about 1.5 times the radius of the range of the laser scanner, in order to ensure that the scan bubbles are properly separated from one another.
- Initiate the Octomap Server Node.
- Select the most appropriate route to travel to all the selected points using a Traveling Salesman algorithm.
- Command the navigation node to move the robot to the scan point. If it does not succeed, it will choose another point and attempt to travel there.
- Execute the 3D Scan action if it reaches the point, and then select the next scan point.
- End the process once all the scans are done.

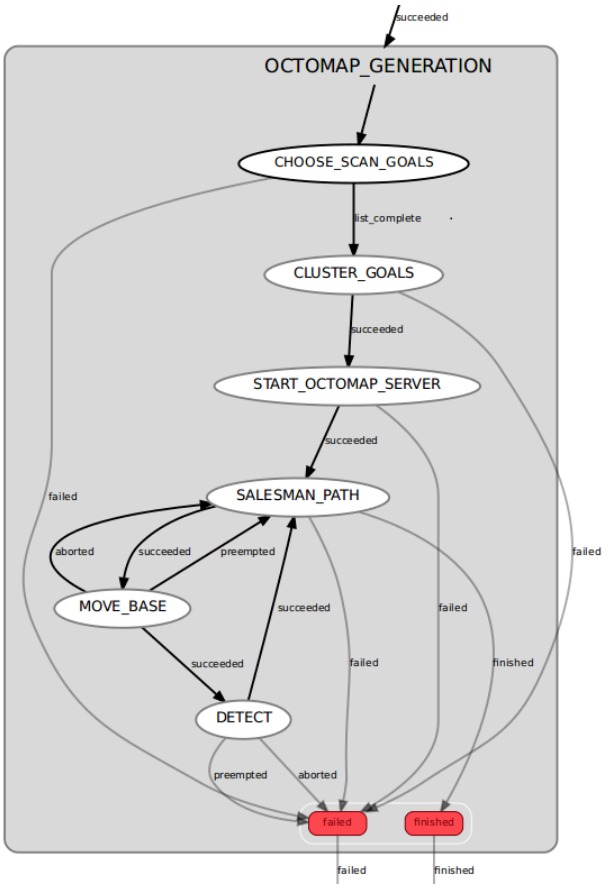


Fig. 7. Autonomous 3D Octomap generation.

This algorithm allows the robot to make 3D models of flat environments autonomously, without prior data about obstacles or the size of the area. It can do the scan by itself as long as the 2D map can find a closed contour of the environment. Therefore, it is not recommended for outdoor areas unless there are clear boundaries for the robot to detect.

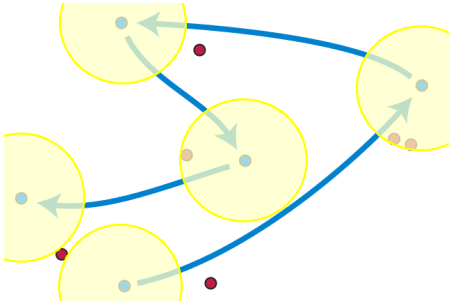

Fig. 8. Farthest First Transversal Algorithm used to make a Delone Set.

## V. Results

These algorithms were successfully tested on a Mechatronics Laboratory in the Universidad Simón Bolívar, the resulting octomaps can be seen in figures 9, 10 and 11.

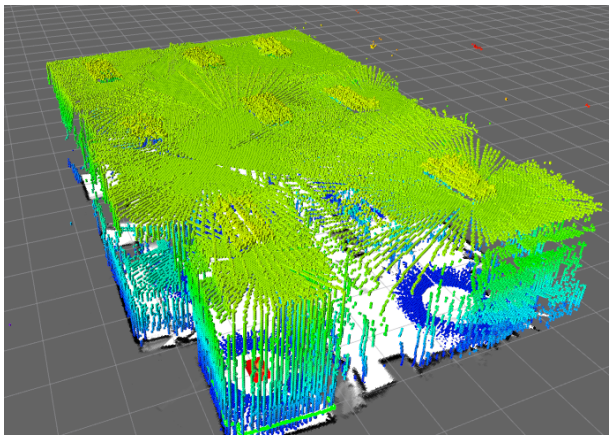

Fig. 9. Resulting Octomap of a laboratory (top view).

The rectangular shape of the building is clearly visible and straight, without discontinuities, meaning that the two stages of the process, one for 2D SLAM and another for 3D octomapping, succeeded in creating reference frames that did not shift the data along the way.

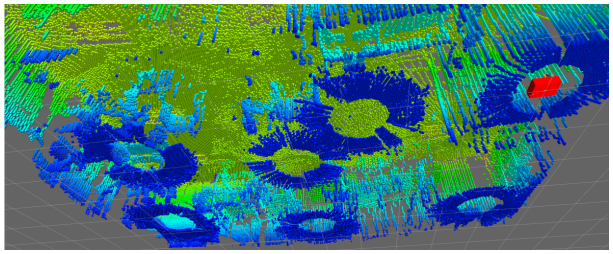

Fig. 10. Resulting Octomap of a laboratory (bottom view).

In Figure 10 the scan points are located in the empty spaces surrounded by dark blue voxels.

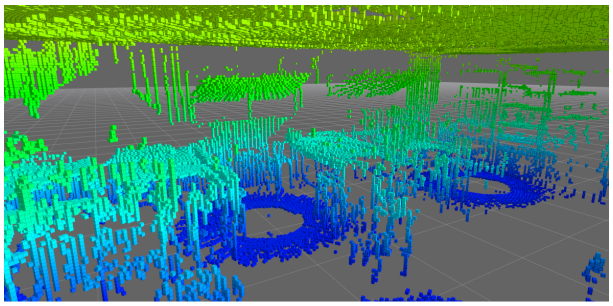

Fig. 11. Resulting Octomap of a laboratory (inner view).

The only problem faced in this work was that dark surfaces absorbed the LIDAR's light and hence they appeared invisible to it, generating blank spaces in the 2D map. To solve this problem, the surfaces had to be covered for the navigation and exploration nodes to work properly.

## VI. CONCLUSIONS

This work demonstrates that, despite being an old method, state machines are still very useful to solve robotic tasks today. The ROS framework is a remarkable tool that allows programs to interconnect in very practical ways. A generic Python version of the Laser Switch node able to work for any message type would be a good addition to the ROS topic tools. With an advanced knowledge of ROS and its tools, apparently problematic tasks such as the simultaneous 2D SLAM and 3D Octomapping through a single sensor can be very easy to solve, as long as the problem is addressed from several angles.

## ACKNOWLEDGMENT

Thanks to the "Grupo de Desarrollo e Investigación en Mecatrónica" of the Universidad Simón Bolívar.

Thanks to Dr. Holger Rapp, developer of Google Cartographer, for all the help with his SLAM package.

## References

[1] H. Durrant-Whyte and T. Bailey, Simultaneous localization and mapping (SLAM): part I The Essential Algorithms, Robotics & Automation Magazine, vol. 2, pp. 99-110, 2006.

[2] N. Certad, Diseño de algoritmo para generación de mapas con localización simultánea en ambientes estructurados para un vehículo autónomo, Master thesis, Universidad Simón Bolívar, 2013.

[3] Quigley, Morgan, et al., ROS: an open-source Robot Operating System, ICRA workshop on open source software, vol. 3, no. 3.2. 2009.

[4] Foote, Tully, tf: The transform library, Open-Source Software workshop, pp. 1-6, Apr. 2013.

[5] Jonathan Bohren, The SMACH high-level executive, IEEE Robotics & Automation Magazine, vol. 17, no. 4, pp. 18-20, Dec. 2010.

[6] W. Hess, D. Kohler, H. Rapp, and D. Andor, Real-Time Loop Closure in 2D LIDAR SLAM, in 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1271-1278.

[7] J. M. Santos, D. Portugal, and R. P. Rocha, An evaluation of 2D SLAM techniques available in Robot Operating System, 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, 2013.

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, OctoMap: an efficient probabilistic 3D mapping framework based on octrees, Autonomous Robots, vol. 34, no. 3, pp. 189-206, Apr 2013.

[9] B. Yamauchi, A frontier-based approach for autonomous exploration, Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA 97. Towards New Computational Principles for Robotics and Automation, pp. 146-151, 1997.

[10] M. González Ocando, Registro de Ambientes 3D Utilizando un Sensor Lidar 2D y ROS, Bachelor thesis, Universidad Simón Bolívar, 2017.

[11] T. F. Gonzalez, Clustering to minimize the maximum intercluster distance, Theoretical Computer Science, vol. 38, pp. 293-306, 1985.