Map My World Robot ?

Miguel A. Colmenares Barboza

Abstract — **The project presented in this document focuses on implementing simultaneous localization and mapping (SLAM) using two sensors (Lidar RGB-D sensor and Kinect camera) installed in a two-wheeled robot. The environments that will be mapped are two different worlds that are simulated in Gazebo. The robot transits each environment separately and maps the environment using the GraphSLAM approach to mapping based on appearance in real time (RTAB-Map).**

Index Terms —Robot, SLAM, RTAB-Map, Udacity, IEEEtran.

I. INTRODUCCIÓN

For a mobile robot that transits a predefined zone is it essential to know where it is? and where should it go? To solve this location problem, SLAM is implemented. This technique solves the problem of localization to map the environment and at the same time maps the environment to solve the localization problem. In this project two 3D environments are simulated in Gazebo and mapped in Rviz implementing SLAM with the help of the RTAB-Map tool. The first environment was named "Kitchen_dining" and the second room was named "Bot_world".

II. BACKGROUND.

SLAM is often called the egg or chicken problem because the map is necessary for the location, and the position of the robot is necessary for mapping, therefore, it is a challenging problem. There are many approaches to performing SLAM, but until now, the two most useful approaches for SLAM are FastSLAM based on grid and GraphSLAM.

FastSLAM basado en cuadrícula.
The FastSLAM algorithm uses a traditional particle filter approach to solve the Full SLAM problem with known correspondences. Using particles, FastSLAM estimates a posterior over the robot path along with the map. Each of these particles holds the robot trajectory which will give the advantage to SLAM to solve the problem of mapping with known poses. In addition to the robot trajectory, each particle holds a map and each feature of the map is represented by a local Gaussian. FastSLAM has a big disadvantage since it must always assume that there are known landmark positions, and thus with FastSLAM we are not able to model an arbitrary environment.

Grid-based FastSLAM extends the FastSLAM algorithm and solve the SLAM problem in term of grid maps, thus you can now solve the SLAM problem in an arbitrary environment. Specifically, Grid-based FastSLAM algorithm estimates the robot trajectory using the MCL. Then, the Grid-based FastSLAM algorithm estimates the map by assuming known poses and using the occupancy grid mapping algorithm.

GraphSLAM.
GraphSLAM is a SLAM algorithm that solves the full SLAM problem. This algorithm recovers the entire path and map, which allows it to consider dependencies between current and previous poses. GraphSLAM has a few advantages over FastSLAM. GraphSLAM improves upon the need of onboard processing capability, while still improving accuracy over FastSLAM. Since GraphSLAM is able to retain information from past locations, it proves an advantage over FastSLAM which uses less information and has a finite number of particles.

The Real-Time Appearance-Based Mapping algorithm is a GraphSLAM approach and 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. While the robot continues to traverse through its environment the map continues to grow. For other Appearance-Based methods, the robot continues to compare new images to past images to identify whether it has been at that location before. This, over time, increases the number of images for comparison, causing the loop closure process to take longer, proportionally increasing complexity along the way. However, RTAB-Map is optimized for large-scale and long-term SLAM, allowing loop closure to be processed fast enough to be known in real-time without dramatically affecting performance.

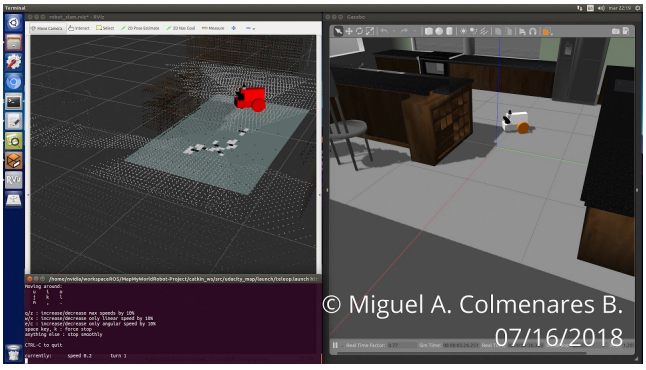

Fig 1. Simulated robot in a gazebo environment implementing SLAM.

III. CONFIGURATION.

In this project, the robot model and the environment model were developed from scratch. Both elements were built, simulated and tested in Gazebo.

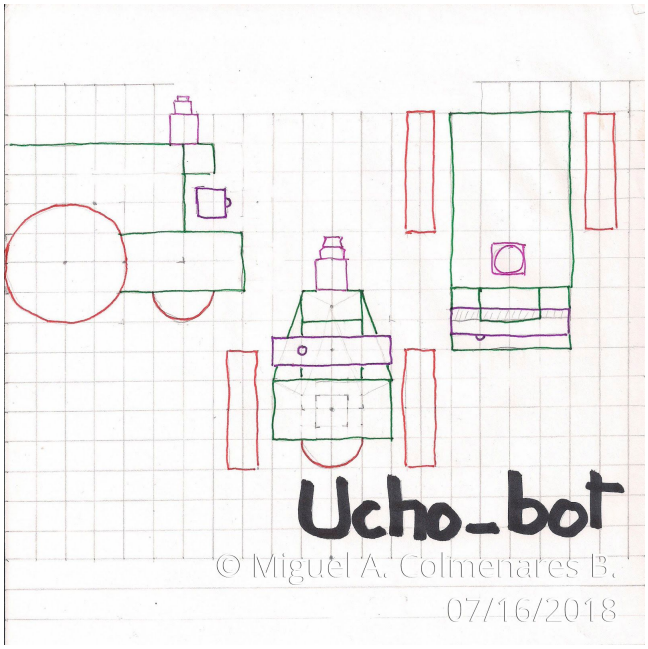

Fig 2. Boceto del Robot.

Robot configuration.
The model of the robot consists of a base, two wheels, a Hokuyo laser sensor located on the top of the robot and a Kinect camera located in the front of the robot.

The files that define the robot model are two files deposited in the URDF folder. The ucho_bot.xacro file provides the description of the shape of the robot in macro format, while the ucho_bot.gazebo provides definitions of the controller of the unit, the RGB-D camera, the laser scanner and the controller of these sensors for Gazebo.

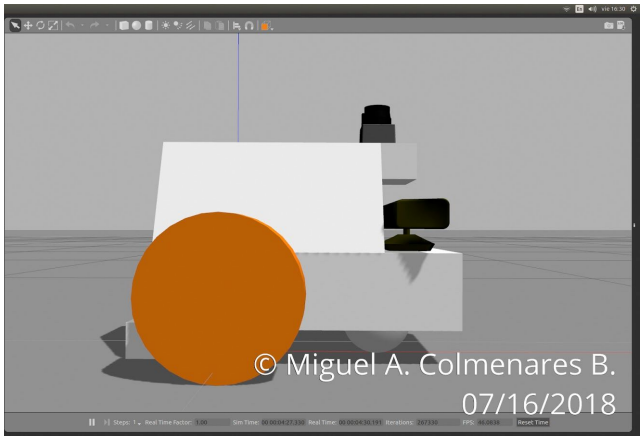

Fig 3. Side View of the Robot.

Fig 4. Front View of the Robot.

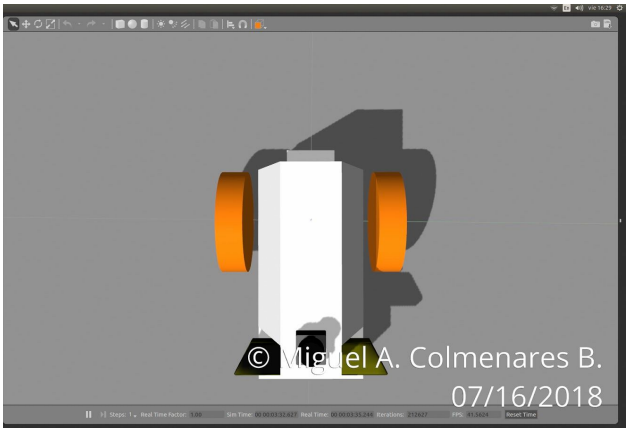

Fig 5. Top View of the Robot.

The robot was developed from Gazebo in an empty world. After having the model of the robot ready, the robot is located in the test environment and the following command is executed:

```
$ rosrun tf view_frames
```

The above command generates a PDF where you can see the coordinate frame of the robot junctions, along with the additional coordinate frame for the world and the odometry.

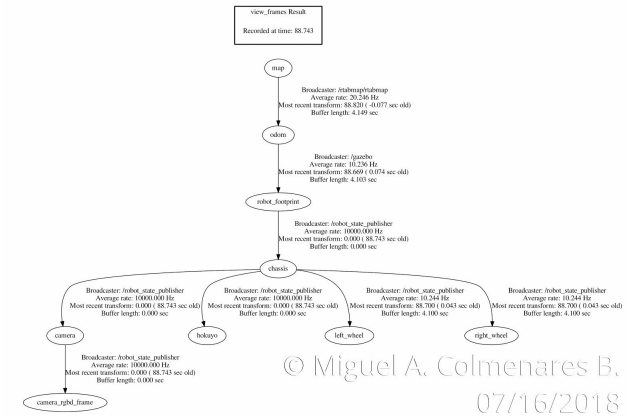

Fig 6. Coordinate frames Union, Map, Odometry, etc.

Configuration of the Environment.
The robot performs the SLAM implementation test in two different environments. The first environment to be evaluated (worlds / kitchen_dining.world) is provided by a third party, it is a kitchen and a dining room. The second environment is simpler. The second environment (worlds / udacity.world) and with very varied elements.

Fig 7. Environment kitchen_dining.world

Fig 8. Environment udacity.worlds

IV RESULTS.

To start the simulation of any of the environments (kitchen_dining / bot_world), four nodes are executed:

```
$ roslaunch udacity_map world.launch
$ roslaunch udacity_map teleop.launch
$ roslaunch udacity_map mapping.launch
$ roslaunch udacity_map rviz.launch
```

Respectively in that order and waiting for a prudential time between execution so that each node is loaded correctly.

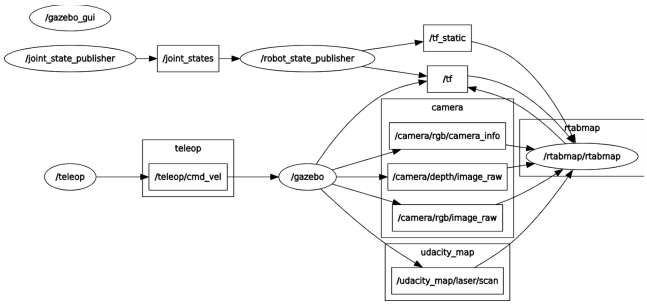

Fig 9. ROS graph of the package udacity_map.

Kitchen and Dining Scene.

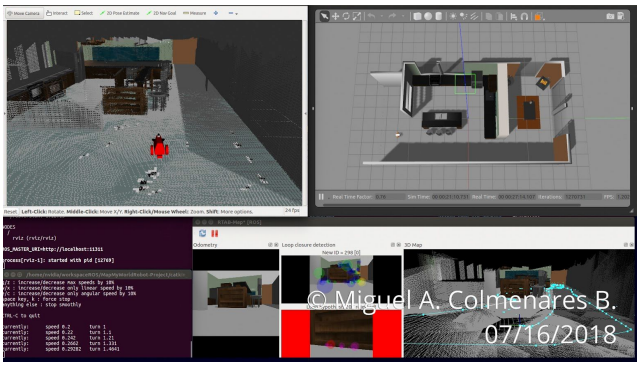

Fig 10. Mapping the environment Kitchen_dining.

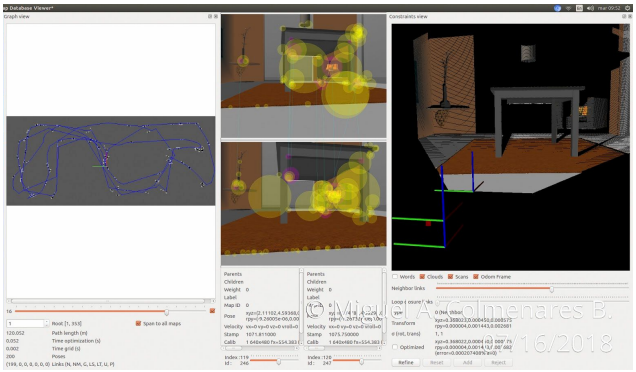
Fig 11. Analysis of the database rtabmapKitchen.db.

Bot World Scene.

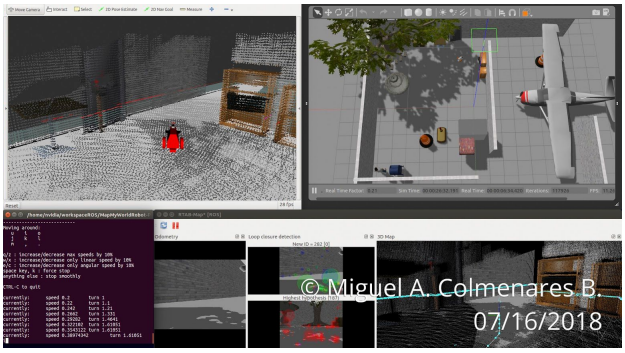

Fig 12. Mapping the environment Bot world.

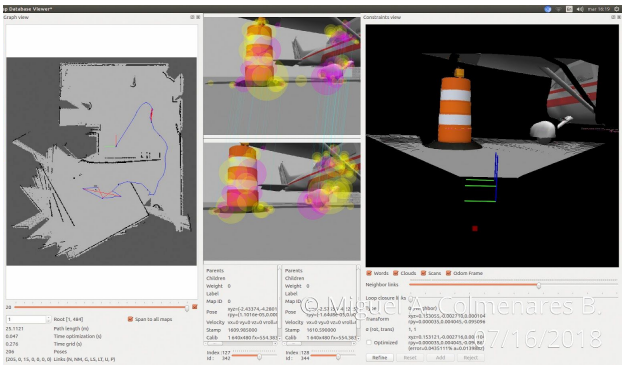
Fig 13. Analysis of the database rtabmapBotWorld.db.

After loading each node, a Gazebo window with the scene being tested was opened on the screen, a Rviz window showing the mapping made by the robot, a window of the RTAB-Map tool and a terminal with the teleop node. loaded to control the robot from the keyboard.

After traversing and navigating around the world Gazebo, the robot was able to generate an accurate map of the scene (Kitchen dining / bot_world). Then the command was executed:

```
$ rtabmap-databaseViewer ~/.ros/rtabmap###.db
```

To open the database rtabmap ###. Db (rtabmapKitchen.db for the world Kitchen dining and rtabmapBotWorld.db for the world Bot world). With this tool you can verify the closing of loops, generate 3D maps to see, extract images, verify rich areas of function maps and much more!

V DISCUSSION.

The built robot model was suitable for mapping tests. The robot transit well by both environments. The Kitchen dining scene was ideal for the robot to implement SLAM, from this scene good results were obtained and the robot was able to map the environment. The only problem was getting a closed loop of the robot's travel throughout the scene. The Bot world scene was more complicated and had its difficulties. The first difficulty was the lack of experience managing the Gazebo program, so it took time to become familiar with the program. The second difficulty was to handle the robot by the scene that turned out to be very big for the robot, for which the robot took more time to travel the scene. And the third difficulty was that during the test with the second scene, on several occasions the robot slid and could not recover. Otherwise, the robot did not have major problems in mapping both scenes successfully.

VI FUTURE WORK.

As future work, there are three specific points to be addressed:

- Build a robot with two track wheels, equipped with the Jetson TX2 and a Kinect RGBD camera sensor to map an entire real-life apartment.
- Add a motion planning algorithm.
- Add a collision detection algorithm.