Robotics SLAM

Antonio Cámara

Abstract – The robotic SLAM problem is studied in this work. A SLAM algorithm is selected and implemented. A simulated system is done with ROS and Gazebo to test the SLAM solution using their debug tools to check the correction of the model. Two different scenarios are considered for mapping in the simulated environment using RTAB-Map. Mappings of the two different scenarios are compared.

1 INTRODUCTION

Simultaneous localization and mapping (SLAM) consists of constructing and updating a map while keeping track of the robot location in the map. It is a difficult task because a map is needed for localization and a good position estimation is needed for mapping.

SLAM algorithms are mainly grouped into five categories:

- Extended Kalman Filters SLAM
- Sparse Extended information Filter
- Extended Information Form
- FastSLAM
- GraphSlam

GraphSlam algorithm is used in this project. It is an algorithm that solves the full SLAM problem. That is to say, it recovers the entire path and map. Specifically, it is used Real-Time Appearance-Based Mapping (RTAB-Map). It is a RGB-D Graph-Based SLAM approach based on an incremental appearance-based 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.

In this work SLAM is done using RTAB-Map with ROS, Gazebo an RVIZ. Two different worlds are mapped to test the algorithm results collecting a 2D and 3D maps from them.

2 BACKGROUND

SLAM is crucial for a robot to know its position and what to do next in an unknown environment. For example, in a mine where new tunnels are done every day, a robot needs to adapt its maps to a changing environment. There are many algorithms that try to solve the SLAM problem and studying them all is out of the scope of this work. Therefore, two algorithms are studied and compared.

2.1 FastSLAM

The FastSLAM algorithm solves the SLAM problem with known correspondences. FastSLAM estimates a posterior over the trajectory using a particle filter approach which gives advantage to SLAM to solve mapping with known poses. FastSLAM uses a low dimensional Extended Kalman Filter to solve independent features of the map which are modeled with local Gaussian.

2.2 GraphSLAM

GraphSLAM constructs a graph of poses and features from sensors data. It obtains the map and the robot path by resolving the constraints into a globally consistent estimate. The constraints are generally non-linear, but in the process of resolving them they are linearized and the resulting least squares problem is solved using standard optimization techniques [1].

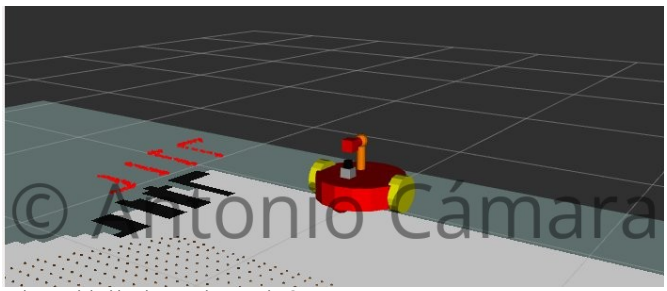
2.3 Comparison / Contrast

Due to the fact that FastSLAM is based in particle filters and there can be no particle in the most probable position, GraphSLAM has an improved accuracy compared to FastSLAM in large environments. For that reason, GraphSLAM is used in this work for mapping simulated environments.

3 MODEL CONFIGURATION

The robot created for the SLAM task has the following characteristics:

- A two wheel differential drive system with integrated odometry which allows to move the robot and measure movements for location.
- A 2D laser hokuyo sensor which allows to form a 2D map.
- A RGB-D camera with elevated position to get better floor images that allows to get a 3D map.



Robot model with odometry, laser and RGB-D sensors.

Robot description can be found in the urdf folder of the slam_project package. It is contained in two files:

- slam_bot.xacro: contains robot physical description.
- slam_bot.gazebo: contains plug-ins configuration (differential drive, RGB-D camera and laser).

Slam_project package has 5 folders with the following content:

- launch: launch files and RViz configuration.
- meshes: hokuyo sensor mesh
- scripts: teleop script to move the robot with the keyboard.
- urdf: robot description and plug-ins.
- worlds: world files with the two used scenarios.

4 WORLD CREATION

Two different scenarios have been used for mapping. First a given scenario with a kitchen_dining model. And then, a new scenario (TestScenario.world) created with the gazebo editor. This custom scenario contains many different elements to make the mapper find loop closures easily.

The custom test world has three rooms with different wall colors. They have different size windows and two different

floors. In each room there are different objects that allow the robot to differentiate locations and detect loop closures.

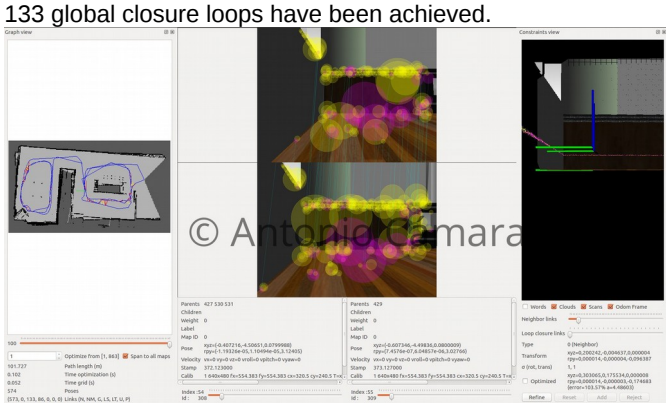
Custom scenario with three rooms and different objects and walls.

5 RESULTS

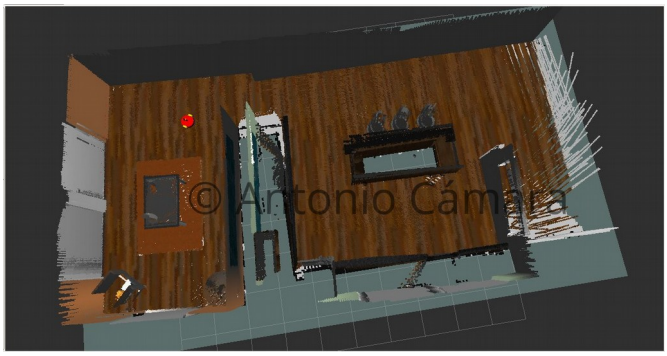
Both scenarios can be launched using the same commands. In first place the world and robot are loaded in gazebo with the following command:
roslaunch slam_project world.launch
To use the custom world it should be indicated as a parameter:
roslaunch slam_project world.launch world_file:=TestScenario.world
In second place the teleop module should be launched to control the robot movements:
roslaunch slam_project teleop.launch
In third place the mapping module is launched:
roslaunch slam_project mapping.launch
The last step is launching RViz to show real time results:
roslaunch slam_project rviz.launch

5.1 First world results

After mapping the first scenario, a 2D and a 3D maps are obtained where objects are clearly distinguishable. Using rtabmap-databaseViewer a 2D map can be seen and a total of 133 global closure loops have been achieved.

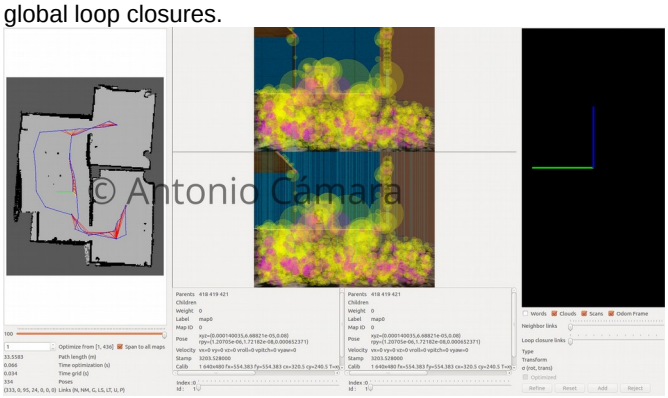


The 3D map shown in RViz displays in detail the scenario.



5.2 Second world results

The custom scenario is a bit more complicated, having three different rooms to map. They have many different objects and colors to allow the robot to differentiate every location.
Rtabmap-databaseViewer shows a good 2D map and 95 global loop closures.



RViz shows the 3D map obtained.

6 DISCUSSION

The first scenario was easier to map. It is simpler and the algorithm estimates the location and map with less mistakes. The robot went several times over the same path creating the maps without big errors. The second scenario is more complex and it was more difficult to get a good map. The map obtained has only one pass over the path getting good details rotating the robot in stationary positions to avoid localization errors. Trying to map the second world like the first one (with continuous movement around the path) gives bad results with a map different from reality.
Another way to map a difficult scenario like the custom world is mapping parts of it and then joining other parts to form a bigger map with multi-session mapping [2].

7 FUTURE WORK

Robotics SLAM problem has been studied in this work. Two different scenarios have been mapped using RTABMap algorithm in a simulation environment with ROS, Gazebo and RViz. Good mapping results have been obtained but it could be difficult with complex scenarios.
After analyzing SLAM, there are many robotics applications where it can be applied. Every navigation problem requires localization and mapping. Even in static environments there could be unexpected objects that change our map and our movement strategy. Therefore, SLAM should be used in all real world navigation problems to achieve a successful robotic navigation.
As future work, the algorithm can be deployed in a real robot and tested in similar situations to compare the obtained results with what was done here.

REFERENCES

[1] Sebastian Thurn, Michael Montemerlo. The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures.
http://robot.cc/papers/thrun.graphslam.pdf

[2] RTABMap Multi session.
https://github.com/introlab/rtabmap/wiki/Multi-session