Showcase of GraphSLAM using the RTAB-Map package in ROS

Mo Messidi

Abstract—This paper showcases a successfully implementation of robot SLAM using the GraphSLAM algorithm. A robot, custom bot, was made to autonomously map two simulation environments unknown to it and localize itself accurately in each of self-constructed maps. The project was developed using ROS. The ROS package RTAB-Map was used to implement the GraphSLAM algorithm and visualize the robot generated maps and gazebo was used to simulate the robot and its environments. The source software for this project can be found here: https://github.com/mo-messidi/RoboND-SLAM-Project.

Index Terms—Simultaneous Localization and Mapping,SLAM, GraphSLAM, FastSLAM, ROS, RTAB-Map, Robotics

1 Introduction

In the framework of robotics, Simultaneous Localization and Mapping (SLAM) is the process of constructing a map of the environment and the path of a robot using only its on board noisy sensor data as input. Given a series of sensor observations over discrete time steps , SLAM generates a map of the environment and estimates the robots location in that map.

This paper showcases a successfully implementation of robot SLAM using the GraphSLAM algorithm. A robot, custom bot, was made to autonomously map two simulation environments unknown to it and localize itself accurately in each of self-constructed maps. One of the simulation environment was given as part of the project, while the other was created independently. The robot, custom bot, that was used in this project was developed specifically to for the SLAM exercise. The project was developed using ROS. The ROS package RTAB-Map was used to implement the GraphSLAM algorithm and visualize the robot generated maps and gazebo was used to simulate the robot and its environments. The source software for this project can be found here: https://github.com/mo-messidi/RoboND-SLAM-Project.

2 Background

Simultaneous localization and mapping is a fundamental problem in robotics. The most common solution approaches to robot SLAM are FastSLAM and GraphSLAM. Given camera and depth sensor data, the SLAM algorithms can estimate the map and the location of the robot within that map as it moves and senses the environment.

2.1 FastSLAM

The FastSLAM algorithm uses a particle filter approach for SLAM. Using particles, it generates a posterior probability function for the pose of the robot and collects the parameters of the map over the robot path along with the map. Each particle holds the robot trajectory which is re-estimated after each re-sampling cycle as the robot moves and senses the environment.

After the FastSLAM algorithm estimates the robot trajectory it is then extended to Grid-based FastSLAM in order to estimate a map of the environment by assuming known robot poses from FastSLAM and using the occupancy grid mapping algorithm. Like FastSLAM, graph-based FastSLAM uses a particle filter approach. Each particle contains map feature predictions in the form of Gaussian distributions that are updated with every re-sampling cycle.

2.2 GraphSLAM

The GraphSLAM algorithm solves the SLAM problem by generating graph from the robot trajectory where the edges represent the robot path and the nodes represent map features as observed by the robot's sensors. Once inter-dependencies are observed, i.e. a number of observations are found to be related, the GraphSLAM algorithm tries to resolve these interdependencies using maximum likelihood estimation to minimize the error present in all of the constraints in the graph. This resolution results in an adjusted graph with partially eliminated sensor error.

The process of generating the dependency graph is called the front-end of GraphSLAM and the process of optimizing the generated graph is called the back-end of the algorithm. The main goal of GraphSLAM is to find the configuration of the nodes that minimizes the error introduced by the dependencies.

The GraphSLAM algorithm was implemented in this project using the RTAB-Map ROS package. RTAB-Map (Real-Time Appearance-Based Mapping) is a ROS package that implements GraphSLAM using input from an RGB-D camera and a laser sensor.

3 Configuration

3.1 Robot Configuration

The robot, custom bot, that was developed for this project has a cylindrical chassis with three balancing casters and two wheels. Its on board sensors are a front mounted

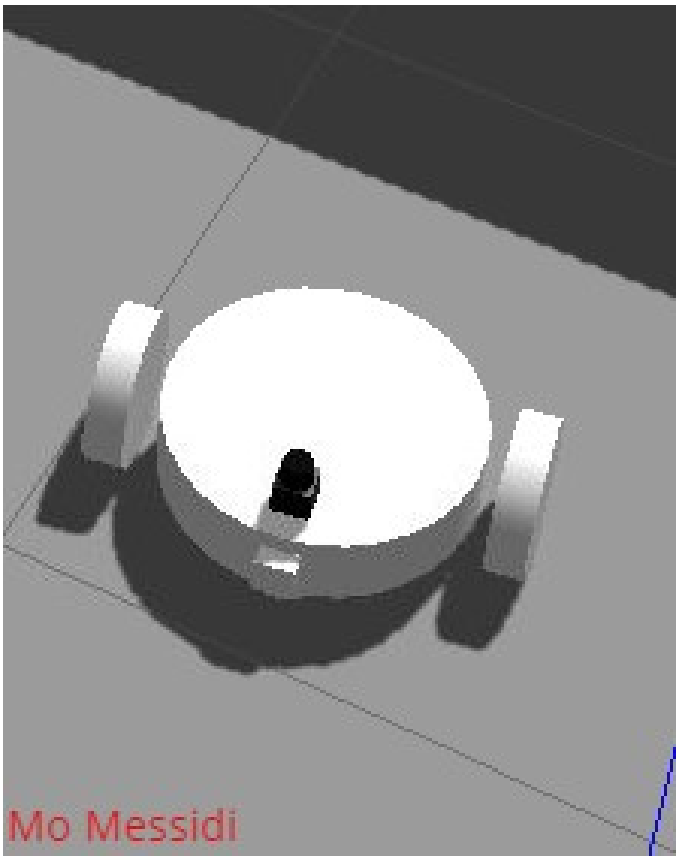

Fig. 1. Custom Bot Model

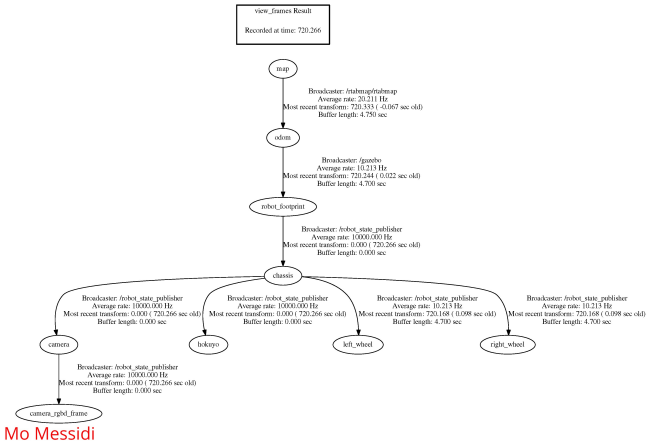

Fig. 2. Custom Bot Frames

laser sensor, a front mounted RGB-D camera, and wheel encoders. Figure 1 shows the robot model simulated in gazebo and figure 2 shows the all the links of the robot and its joints.

3.2 Environment Configuration

Two simulation environments were mapped in this project. Both environments were unknown to the robot before mapping. One of the simulation environment, Kitchen and Dining, was given as part of the project, while the other, MyWorld, was created independently. Both environments

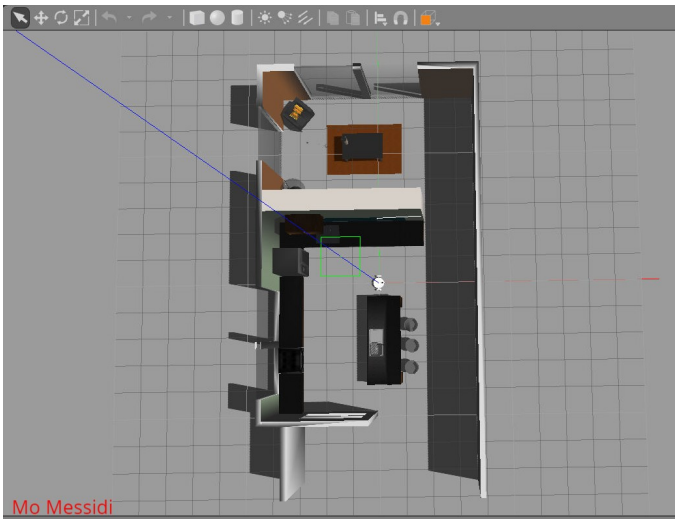

Fig. 3. Kitchen and Dining Environment

were built and simulated using Gazebo. The Kitchen and Dining environment shown in figure 3 is an unchanged sample gazebo world and MyWorld environment shown in figure 4 is a slightly customized environment based on the sample gazebo world cafe.world but with several additional objects such as a brick wall, a granite table and a wooden slab added to it.

4 Results

The robot was able to successfully map both environments it was exposed to and localize itself in each. Figures 5 and 6 show the robot mapped kitchen and dining environment and figures 7 and 8 show the robot mapped MyWorld environment.

5 Discussion

The robot was navigated via the keyboard in both environments and features where added to the feature bank using the bag-of-words methodology. Once the robot revisited a previously known area in the map a loop is detected and loop closure is performed to account for senor errors and optimize the self-generated map and location. In order for improve loop closure detection the parameter MinInliers that sets the threshold of common features that need to exist for a loop closure to occur was set to 10 from the default of 15. This parameter change had a significant effect on loop closure detection, especially in the myworld environment that was less feature rich than the kitchen and dinning environment.

6 Conclusion / Future work

This paper showcased a successful implementation of the GraphSLAM algorithm to allow a robot to map two distinct environments and accurately localize itself within those environment simulations. A reasonable future expansion for the project would be to implement the same SLAM methodology in a real world environment rather than a simulated one and compare the results of each.

Fig. 4. MyWorld Environment

Fig. 6. Mapped Kitchen and Dining Environment Grid

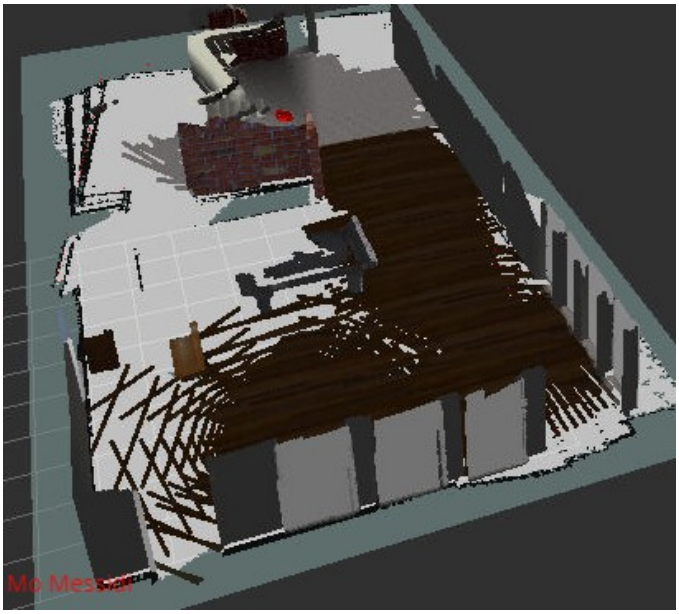

Fig. 7. Mapping MyWorld Environment

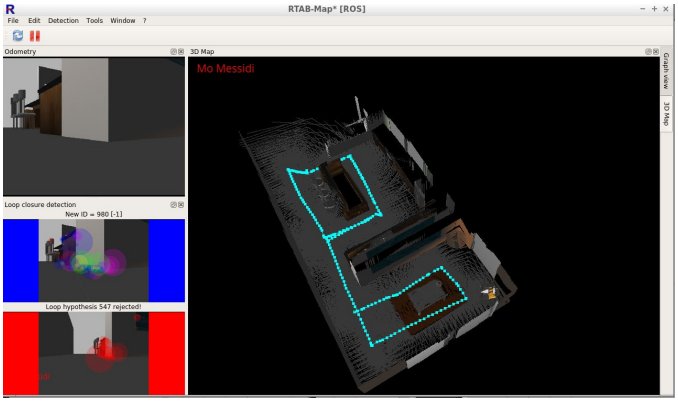

Fig. 5. Mapped Kitchen and Dining Environment

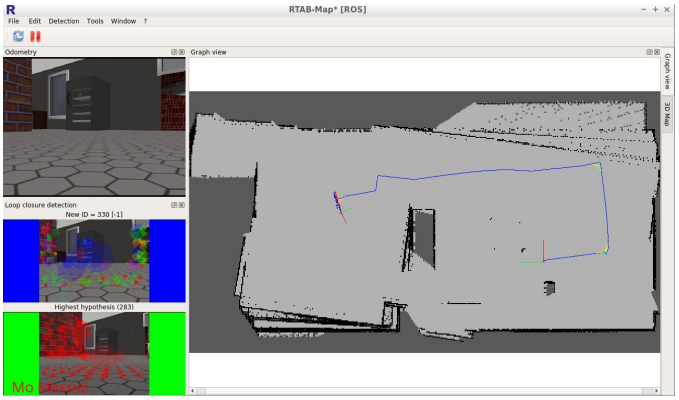

Fig. 8. Mapping MyWorld Environment Grid