Map My World Robot Writeup

Marko Sarkanj

Abstract—The focus of this project is performing Simultaneous Localization and Mapping(SLAM) in the Gazebo simulator with help of the Robot Operating System(ROS). To achieve this, the custom robot model has been created in the Gazebo simulator with two wheels, Hokuyo sensor and RGB-D camera. Sensor readings were sent to Real-Time Appearance-Based Mapping ROS node(RTAB map) to perform the RGB-D Graph-Based SLAM and save the results in the RTAB map database. The localization and mapping were performed on two different Gazebo environments, "kitchen_dining" environment and custom created Gazebo world environment, which resulted in the two different RTAB maps.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Localization.

1 Introduction

As part of the Udacity's Robotics Engineer Nanodegree program one of the assignments was to perform simultaneous localization and mapping on the two environments simulated in Gazebo. The first environment has been provided by Udacity and it contains kitchen and dining room. The second environment has been created for the project and it contains robotic workshop with SUV and firetruck parked in front of it.

2 Background

In this project there are no previous maps or robots poses based on which it would be possible to perform localization and mapping. This is why it is necessary to perform localization and mapping simultaneously, which can be achieved by SLAM. There are different approaches that can be applied to a simultaneous localization and mapping problem, some of which have been discussed in further sections.

2.1 Online SLAM vs. Full SLAM

In the case of the Online SLAM approach, no previous measurements and controls are taken into consideration when the mapping is performed, only the current ones. In the case of the Full SLAM, all of the measurements and controls from a robot's path are taken into consideration when performing SLAM.

2.2 Continuous and discrete nature of SLAM

To be able to estimate its poses, a robot needs to continuously collect odometry information. A robot needs to continuously sense the environment as well, to be able to estimate object's locations or landmarks.

As a robot moves through the environment it needs to continuously check if it has already visited the specific location to be able to perform loop closure. This is the discrete nature of SLAM.

2.3 Grid-based FastSLAM

Fast slam uses particle filter to solve the mapping and localization problem. Each particle contains a robot's pose as well as a robot map. FastSLAM estimates the robot's pose as a posterior over the robot's trajectory by using the particle filter approach. With help of low dimensional EKF, FastSLAM solves independent features of the map, which are modeled by a local Gaussian.

FastSLAM slam solves Online SLAM and Full SLAM problems simultaneously. It estimates robot's full path and each particle estimates instantaneous poses at the same time.

Grid-based FastSLAM is an extension to FastSLAM that adapts FastSLAM to occupancy grid maps. The advantage of this approach is that no known landmark positions are required, which contrasts to FastSLAM. The grid-based FastSLAM updates a map of each particle with help of the occupancy grid mapping algorithm.

2.4 GraphSLAM

GraphSLAM solves the Full SLAM problem. Its main advantage over Grid-based FastSLAM is that it doesn't use particle, so it is able to achieve higher precision in comparison to Grid-based FastSLAM. When the particle filter approach is used, there is always a possibility that there is no particle at the robot's current position. The other advantage of GraphSLAM is that it uses all the information from the robot's path to estimate the current position and the robot's pose, not just bits of information that are in form of preserved particles like Grid-based FastSLAM.

GraphSLAM constructs a graph of poses and features, defines constraints and creates system of equations to determine most likely set of poses and map features given a system observation.

2.5 RTAB-Map

RTAB-Map is one form of GraphSLAM approach that performs real time appearance based mapping. Appearance based mapping performs mapping based on data collected with help of visual sensors. To do that successfully, the process called loop closure is necessary to determine if the

robot has seen the location before and avoid large map distortions. With help of defining image attributes as bag of words and advanced memory management, RTAB-Map is able to perform loop closures in real time.

2.6 Choice of algorithm and ROS implementation

RTAB-Map has been chosen for this project based on arguments that were stated above. The mapping has been performed with help of the robot operating system(ROS) and RTAB-Map ROS wrapper. Information from the environment has been collected with the help of RGB-D camera, Hokuyo sensor and odometry movement sensors. The collected data served as an input to RTAB-Map algorithm which has been implemented as a ROS node. The result is the database containing robot's poses and map features defined as 3D point clouds.

3 Model Configuration

3.1 Robot's parameters

The robot has two wheels as actuators, box shaped chassis, RGB-D camera in front and Hokuyo sensor on the top. The robot's shape has been defined in RGBD_camera_bot.xacro file.

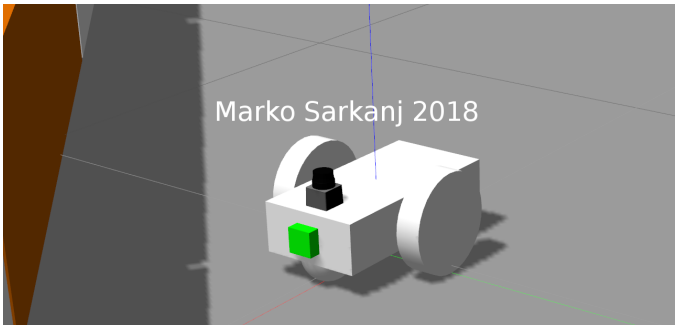

Fig. 1. Design of the robot

3.2 Sensor features

Three different sensors have been used in this project: Hokuyo sensor, odometry readings from the differential drive controller and RGB-D camera. Differential drive controller was used to simulate the robot's movement by moving left and right wheels. RGB-D camera was used to take depth pictures of the environment and Hokuyo sensor was used for the laser scan of the environment. Hokuyo controller publishes scan topic, RGB-D camera rgb/image_raw, depth/image_raw, depth/points, rgb/camera_info and depth/camera_info topics. Differential drive controller publishes odom topic and receives cmd_vel topic.

The RGBD camera frame(camera_rgbd_frame) was rotated for -1.58 radians on roll and pitch axis so that the RGB-D point cloud would be presented correctly in the RViz simulator, as well as mapped correctly by the RTAB-Map node to the RTAB-Map database.

3.3 Package structure

The central node of the project is RTAB-Map mapping node(rtabmap_ros package) described in the mapping.launch file. In the mapping.launch file RGB-D camera and scan(Hokuyo sensor) topics that are subscribed by the RTAB-Map node have been defined, as well as the chassis frame id, odometry frame id and the database path. RTAB-Map output topic "/map" has been defined as well. In addition to that, mapping.launch file contains parameters that configure RTAB-Map GraphSLAM approach, such as loop closure detection strategy(SURF), rate at which new nodes are added to map(10 Hz), maximal number of visual words per image represented as bag-of words(200) and minimal visual inliers to accept loop closure(15). All those parameters have been defined based on experimentation except rate at which new nodes are added to a map, which has been defined based on the camera capture rate of 10 Hz.

Other packages used are gazebo_ros used in udacitly_world.launch file to spawn the robot, rviz package used in rviz.launch file to launch RViz, tf package to perform previously described static transformation of the RGBD camera frame and the robot_state_publisher package to publish robot's states.

4 Gazebo World Creation

The custom world created in Gazebo contains the robotic workshop with SUV and firetruck parked in front of it. In robotic workshop numerous small objects have been added, like drone, robot model, table, boxes and traffic sign. This has been done to test how objects of different sizes, colors and shapes are mapped with the RTAB-Map approach. Two larger objects, SUV and fire truck were added to test how larger objects get mapped as well, especially to test the detection of loop closures in the case of larger objects.

5 Results

As the result of the previously described steps, following images have been produced:

Figure 2.- Screenshot of the Gazebo environment containing kitchen and dining room

Figure 3. and 4. - RTAB-Map of the kitchen and dining room environment

Figure 5. - Kitchen and dining occupancy grid

Figure 6. - Screenshot of the custom Gazebo environment

Figure 7. and 8. - RTAB-Map of the custom Gazebo environment

Figure 9. - Custom Gazebo environment occupancy grid

6 Discussion

The main challenge of the project was correctly connecting RTAB Mapping ROS node to all of the published states from the sensors. The second challenge was rotating depth images received from the RGB-D camera so that they can be processed correctly by the RTAB Mapping ROS node. After the described steps one change of the configuration of teleop.py was necessary, to publish commands sent to robot to the correct state cmd_vel, which can be processed by the differential_drive_controller Gazebo plug-in.

Fig. 2. Screenshot of the Gazebo environment containing kitchen and dining room

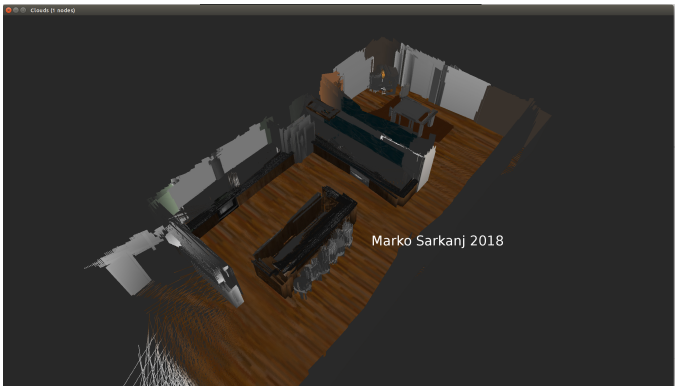

Fig. 3. RTAB-Map of the kitchen and dining room environment

Fig. 4. RTAB-Map of the kitchen and dining room environment

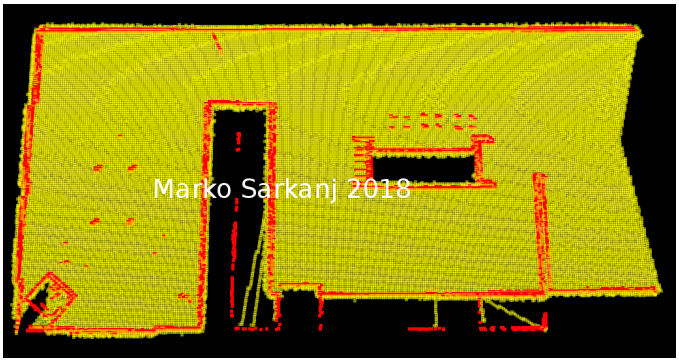

Fig. 5. Kitchen and dining occupancy grid

Fig. 6. Screenshot of the custom Gazebo environment

Fig. 7. RTAB-Map of the custom Gazebo environment

Fig. 8. RTAB-Map of the custom Gazebo environment

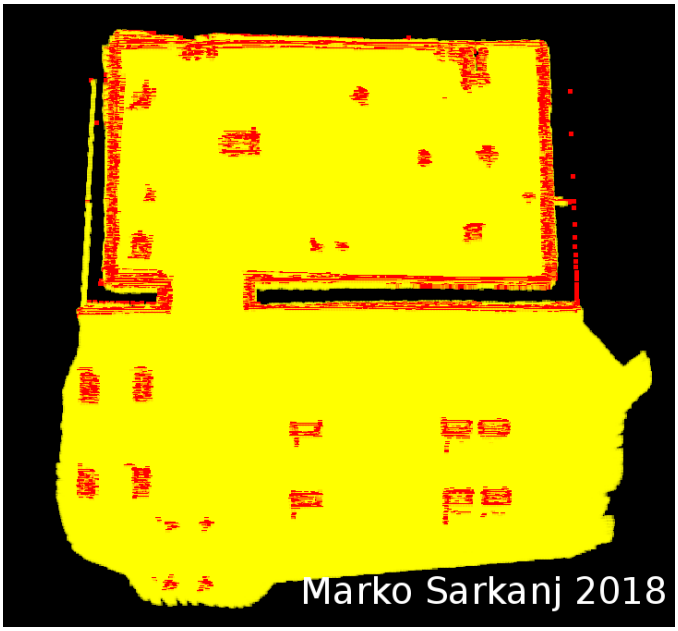

Fig. 9. Custom Gazebo environment occupancy grid

One additional step of the project was creating custom Gazebo environment. This has been done by adding various 3D objects to Gazebo world.

After all the described configuration has been completed, the mapping of the environment was straightforward. It was necessary to drive the robot through the Gazebo world and track the mapping progress in RViz. After that the point cloud map has been exported with the RTAB map database viewer(rtabmap-databaseViewer).

RTAB Mapping of the Udacity provided environment performed better that the RTAB Mapping of the custom Gazebo environment. The reason for this was that the custom Gazebo environment contains many small objects that are sometimes hard to recognize from the surface because of the similar colors. Another reason were very large objects in the custom Gazebo environment as well(firetruck and SUV), which were hard to map completely as the robot didn't had enough distance from the objects to perform the mapping correctly.

Another problem of the RTAB Mapping approach is that the loop closures are still not perfect, some objects can have distorted shape after few robot loops around the map even though SURF local feature detector and descriptor has been implemented.

One of possible future improvements would be increasing camera resolution which would help in detecting smaller objects but also increase hardware requirements because of the increased computation time.

7 Conclusion / Future work

RTAB Mapping approach performs reasonably well if the objects are not too small and they have distinct color from the background. Algorithm can be applied for robot orientation and movement in space and eventually for picking up objects and delivering them on specific shelf or location. As future work it would be interesting to apply RTAB mapping approach to the real world. It would be interesting to experiment with some other sensors like lidar and create RTAB map based on them as well.