Project: Map My World Robot

Martin Papanek

Abstract—In this report, a ROS (robot operating system) package for SLAM (simultaneous localization and mapping) is developed and subsequently tested on two model environments. The SLAM algorithm used is the RTABM algorithm which is described in the background section. A test robot, equipped with an RGB-D camera and a laser range scanner, is navigated through the model worlds and performs SLAM. Finally, the results of the mapping are compared and contrasted and future improvements are suggested.

Index Terms—SLAM, ROS, RTABMap, IEEEtran, Udacity, LaTeX.

1 Introduction

Robots operating in real world environments often do not have the luxury of access to a full map of their surroundings nor do they often know their own location. SLAM is a method for giving robots the ability to compute their pose and orientation as well as map the environment. This task is of utmost importance in robotics and as part of this project the RTAB-Map SLAM algorithm is put to use on two feature rich model environments in the Gazebo physics simulator. To be able to use the RTAB-Map algorithm with Gazebo, a ROS package is developed. The responsibility of this package is to start ROS nodes. Nodes that are required for this project are teleop for tele-operation of the robot so that movement in the simulated environments can be tested. An rtabmap node for the SLAM computation. A node publishing robot transform frames. And a robot model with sensor inputs which interact with the Gazebo environment.

2 Background

To be able to perform their tasks, robots need know their location in the environment, as well as know where the objects that they interact with are with respect to them. The task of estimating the location of the robot is called localization. Localization depends on knowing the map of the surroundings so that the pose and orientation of the robot can be estimated based on the sensor readings of the surroundings and then by comparing the readings to the map. If the map is unknown, but the robot location is known, then the robot can perform mapping of the environment to build a map. However, if both the location and the map are unknown, which as said is often the case, then the robot needs to be able to at the same time map the environment and estimate its pose and orientation. This is a difficult task. In robotics this is called SLAM which stands for simultaneous localization and mapping and there are a number of different algorithms to perform SLAM.

In this paper, the focus is on the RTAB-Map (real-time appearance based) algorithm. which is a Graph based SLAM algorithm which uses data from an RGB-D camera to compute SLAM. Graph based SLAM algorithms work by keeping a graph data-structure of features and robot poses. This datas-tructure is constantly updated as the robot navigates around the world, and when objects that have been seen before are encountered again the loop closure process is responsible for detecting this and updating the graph as needed. The RTAB-Map algorithm implementation is freely available and there is a ROS wrapper to start RTAB-Map as a ROS node and to interact with this implementation through ROS topics.

Another type of SLAM algorithm, is the FastSLAM algorithm which is a grid based SLAM which means that it discretises the map into cells and performs an occupancy grid mapping at the same time as pose estimation using a particle filter approach similar to the MCL localization algorithm. In general Graph algorithms can be slow because loop closure is O(n), but the RTAB-Map algorithm utilizes a memory management technique whereby it has an upper bound on the amount of time that loop closure can take and so it is in general much faster than other graph based algorithms or grid based FastSLAM whose performance depends heavily on the number of particles it uses.

In general, SLAM algorithms are classified based on whether they are online of full. Online algorithms only estimate the current pose and orientation and the current map, whereas full algorithms estimate the entire trajectory. SLAM algorithms deal with the continuous domain which is the task of estimating continuous aspects such as pose, but also with the discrete domain which involves finding correspondences between objects that have been seen before for loop closure. In 2D SLAM only the two dimensional (x,y) pose is recovered during SLAM, in 3D SLAM additionally the height of objects is also estimated during mapping. This is necessary for drones or for example for robots which are in environments that contain tunnels that they have to pass through.

3 Scene and robot configuration

The first virtual environment used was the udacity kitchen environment. The second environment created for this project was the crime_scene environment which is quite feature rich including a pick truck, a person and a body in an alley. The environment has to be very feature rich so that loop closure can be performed multiple times without corruption of the generated 3D map. For this purpose many distinct objects like fire hydrants and traffic cones were added and also chessboard patterns were placed in

a number of places in the scene to make correspondence detection during loop closure easier.

Fig. 1. The crime scene

In this project, the udacity bot from the previous localization project was reused. It is a robot with two side wheels which are driven by means of a differential drive gazebo plugin, two castors below the chassis for stability, a hokuyo laser scanner attached to the top of the chassis, and an RGB-D camera attached to the front side of the robot so that there are no parts of the robot obscuring its view. The laser scanner, the differential drive and the RGB-D camera are all implemented using code provided as shared libraries for use with gazebo. For the crime_scene environment, the RGB-D camera plugin used a higher resolution of 1024x800 and a higher update rate of 30Hz to have better images in order to facilitate the SLAM processing of the environment. To fix a bug in RTAB-Map whereby the point clouds it generates have a the x, y and z axis are pointing in the direction of the odom frame's axes y, z and x (respectively), a fake link and joint called 'camera_link' and 'camera_link_joint' have been added to the robot's urdf which the joint applying an roll-pitch-yaw reference frame transformation of $-\pi/2, 0, -\pi/2$. The RGB-D camera controller then uses the 'camera_link' frame which fixes the orientation of the RTAB-Map point cloud.

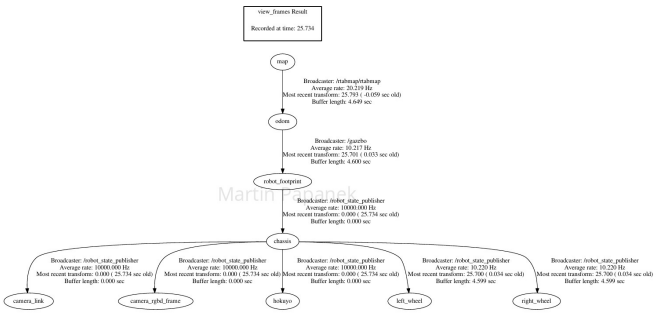

Fig. 2. Transform frames

The package structure that was followed in the project has a folder called 'urdf' for the robot definition files and the gazebo plugin file. A 'worlds' folder which contains the two virtual models of environments. A 'meshes' folder for the hokuyo mesh. A 'scripts' folder which contains the 'teleop.py' script which is started as part of 'teleop.launch' to make the robot receive commands. A 'config' folder for the rviz config which is used by the 'rviz.launch' file and finally a 'launch' folder which contains all the necessary launch files to bring up the simulated world and robot in gazebo, start rtabmap, start teleop and rviz. This flat package structure was chosen because it was used in the localization ROS package.

For mapping of the kitchen world the 'mapping.launch' file from udacity was used. This uses SURF descriptors and a visual only Reg/Strategy. For the mapping of the crime scene these parameters performed miserably and so the configuration given in the RTAB-Map tutorial was used instead. It uses BRIEF descriptors and ICP which includes scan.

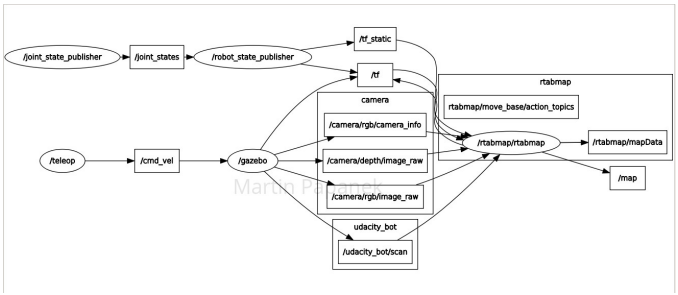

Fig. 3. ROS nodes and topics

4 Results

The results of performing mapping in the kitchen environment are shown below in **??** and 5. As can be seen the robot fills the occupancy grid correctly and manages to do 256 global loop closures. From the 3D map it can be seen that most of the space that was in range of the robot's camera was mapped, however some loop closures especially around the chair and the desk in the small room were performed incorrectly and the objects seem to sit on top of each other multiple times.

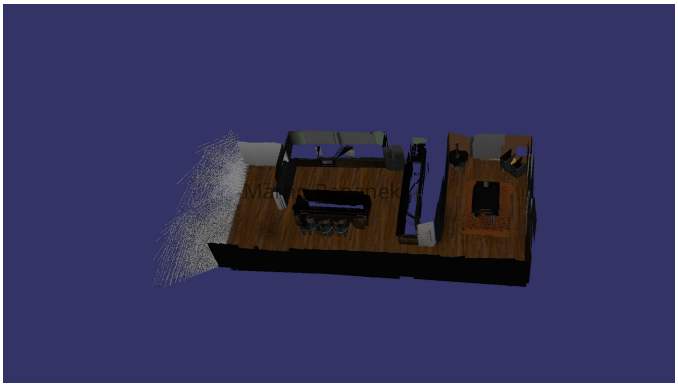

Fig. 4. Kitchen 3D map

The results for 3D mapping the crime scene environment are shown in 6 and 7. As can be seen, since this environment contained many objects, not all parts were accessible to the robot and therefore some corners remain un-mapped.

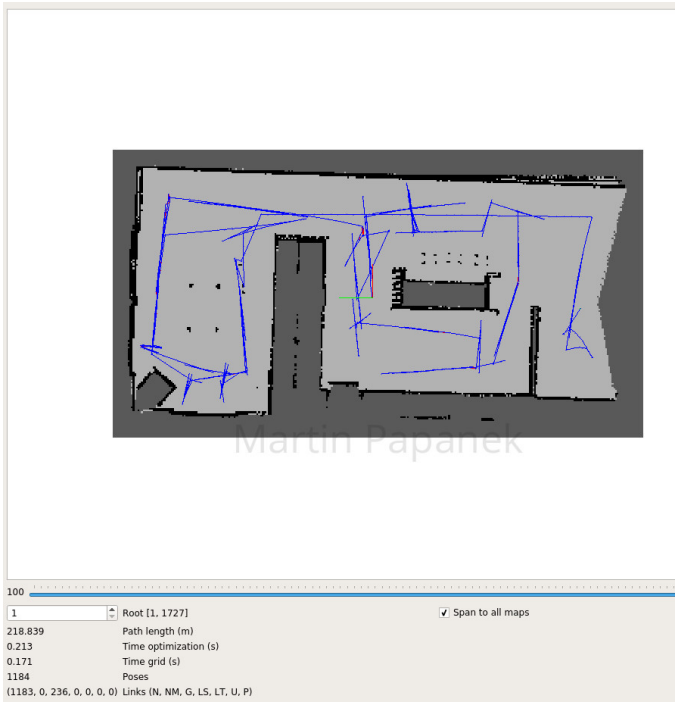

Fig. 5. Kitchen 2D map

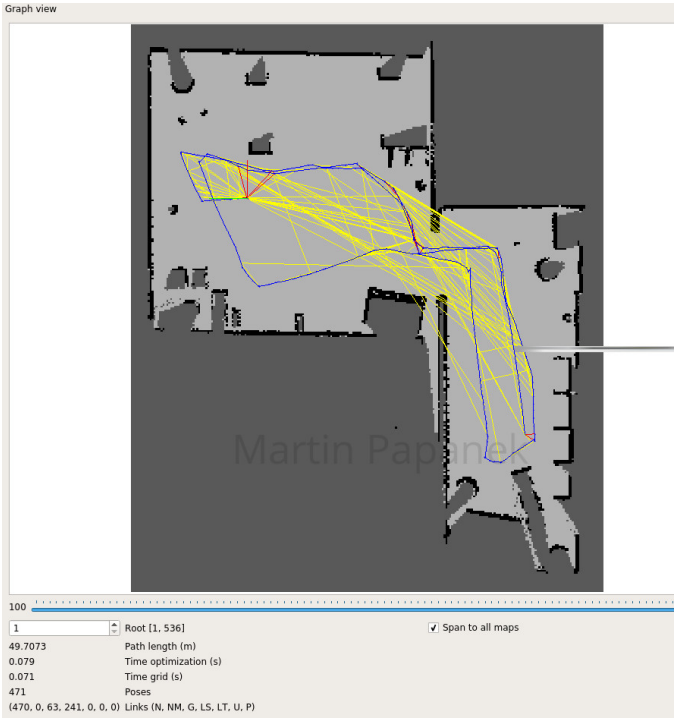

Fig. 7. Crime scene 2D map

However, the map is quite recognizable if compared to the image from gazebo in 1. From the 2D map we can see that the robot performed 63 global closures. Again some distortions can be seen especially around the car and the orange traffic cones, where loop closure failed, but not as many due to few sharp turns in the route taken by the robot.

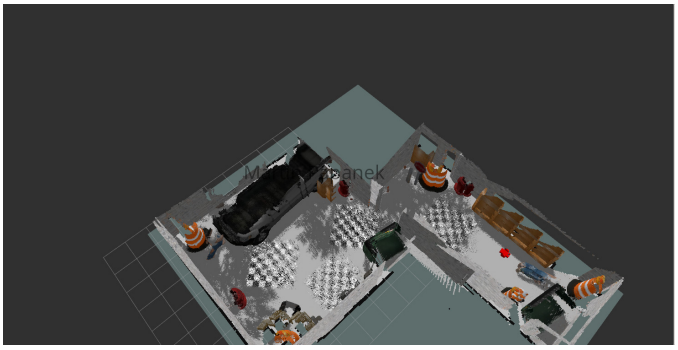

Fig. 6. Crime scene 3D map

5 Discussion

The quality of the produced map depends heavily on the parameters provided to RTAB-Map but these are not documented very well and there are so many that it is difficult to try all. The safest approach turned out to be to drive the robot very slowly and attempt to not turn it too much. Once the map gets corrupted by an incorrect loop closure, it never seems to fix itself (perhaps due to some unknown parameter setting) and so it is easy to end up in a situation where multiple copies of the same object are visible in the map in many different places and the robot then struggles to localize itself in the map. Driving very slowly and not turning works the best.

The mapping of both the crime scene and the kitchen contains many distortions, especially around objects which look quite different when seen from different perspectives (like most real world objects) such as the chairs or the car in the crime scene world. The crime scene contains fewer distortions because as can be seen from 7 the robot's route through the map doesn't contain that many turns. In both maps objects are registered correctly as obstacles in the occupancy map but their exact shape is often quite distorted. Surprisingly, the detail of the object, such as whether it is quite feature-rich will not prevent these distortions from happening, as can be seen from the traffic cones in the crime scene map.

6 Conclusion / Future work

In the future, it would be interesting to see if RTAB-Map performs better with real world images instead of images taken in the simulated environments. Real world images will have considerably more detail that can be utilized in the classification of images into a bag-of-words by RTAB-Map so it stands to reason that the generated environments should have fewer distortions. On the other hand real world objects are also much more complex so it isn't obvious how well it will perform. It will be also good to learn more about the parameters of RTAB-Map and explore them more in depth, which wasn't part of this project. The parameters are very important on the quality of the result and there is a large amount of different parameters and thus a very large number of parameter combinations to try. It is quite likely also that real world mapping may benefit more from parameters which don't work well in simulation.