Map My World

Sandra Schuhmacher

Abstract—In this project Simultaneous Localization and Mapping (SLAM) is used to map environments. It uses the Real Time Appearance Based Mapping (RTAB-Map), a GraphSLAM algorithm.
For the experiment, the same robot model is used as in the previous *Where Am I?* project, but extended by a depth camera. First, a provided kitchen environment is mapped, then a custom environment is constructed and mapped.
The goal of the project is to explore the capabilities of the RTAB-Map algorithm and map both environments with at least three loop closures.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, deep learning.
Repository: https://github.com/yulivee/RoboNDMapMyWorld

1 Introduction

In real world situations of mobile robots environments are constantly subject to change, resulting in few applications where a fully featured map can be provided for navigation in advance. The robot needs to solve a kind of chicken-and-egg problem: mapping the environment and localize itself at the same time. Fortunately, this problem is solved by Simultaneous Localization and Mapping (SLAM) which is explored in this project to map a simulated environment. The robot exploring the world is the same as in the previous *Where Am I?* project, extended by an RGB-D camera to provide depth input for the mapping. On successful application of the RTAB-Map GraphSLAM algorithm, the robot produces a map with identifiable features of its surroundings.

2 Background

In robot mapping, the surrounding environment is estimated and the path of the robot is assumed to be in the past. A lot of challenges present themselves while mapping like

- infinite variables to describe a map as a continous space
- uncertainty in perception
- feature-richness of the space to be mapped

SLAM takes this challenge a step further: additionally to constructing a map of the environment, the robot simultaneously localizes itself relative to its map. Its a basic feature for autonomous mobile robots. Neither the map nor the robot poses are provided and both will inhabit noise from the sensors measurements. This could lead to an uncertain map and errors in the pose estimates. The problem is approached in two variations: *Online SLAM* estimates current pose and map from previous measurements and motions, *Full SLAM* estimates the entire trajectory and map features at the same time.

A SLAM algorithm has to identify correspondences between images to correctly resolve map features. Basic SLAM works like this: The robot starts with measurement zero on its initial pose and includes it into the map. Then the robot moves and takes another measurement, calculates its new position from the current map and includes the new unseen features into the map. This way, the map is incremented continually. The differences between the algorithms is how they detect already seen features to keep the map consistent - this is called the *loop closing* problem.

The SLAM method used in this experiment is a GraphSLAM algorithm. These types of algorithms represent the SLAM problem as a graph with poses of the trajectory and measurement locations as nodes as well as estimated motion and measurement distances as links. The links in the graph are the contraints between the robot poses and its environment and represent their relationship. GraphSLAM tries to resolve all of these constraints to create the most likely map on the given data. In this project, the algorithm *Real Time Appearance Based Mapping (RTAB-Map)* is used. It is an upgraded version of GraphSLAM which uses a *bag of words approach* to identify correspondences between frames using visual similarity. Every frame, the camera image is compared against already known map features allowing to map trajectories with repeated locations by reducing constraint complexity.

3 Scene and robot configuration

3.1 Scene configuration

A new environment is build from scratch in Gazebo by using the Wall-Tool to create the outer room structure and adding some models from the publicly availiable model database. It is meant to resemble a robot workshop and the robot is meant to resemble a vaccuum robot which is supposed to clean the workshop. The robot workshop world is presented in 1. The transform-tree can be found at the last page of this document.

3.2 Robot Configuration

The robot model is inherited from project *Where Am I?*. A new camera link `camera_link_optical` and joint were added to the camera frame as the image of the depth camera needed to be rotated by 90 degrees due to some internal misconception of the kinect depth camera driver (see this link to ROS Answers for further details). The robot model

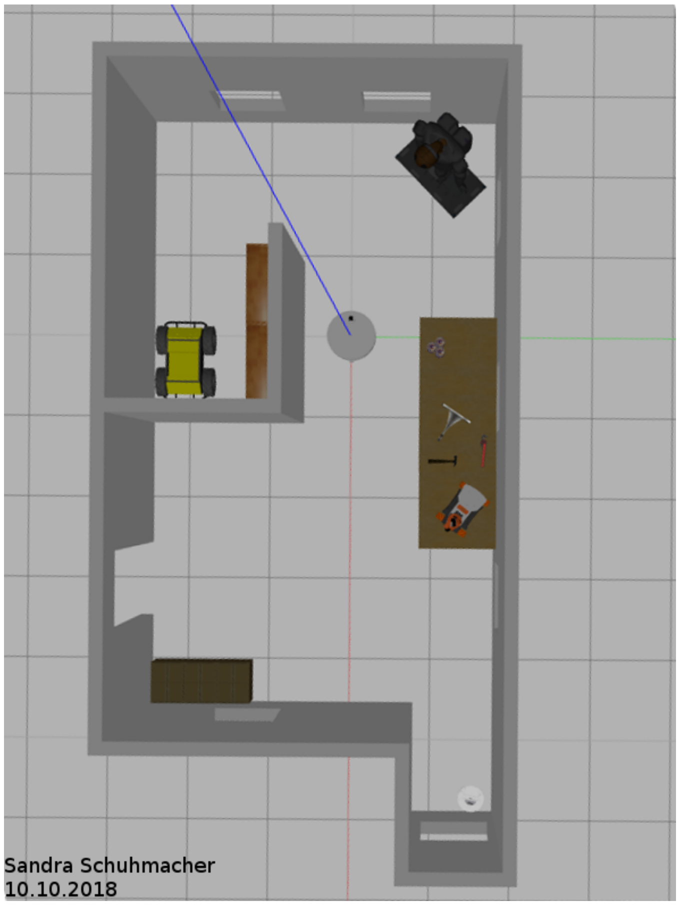

Fig. 1. Robot in the Workshop World

is presented in 2. The transform-tree can be found in the repository at `report/frames.pdf`.

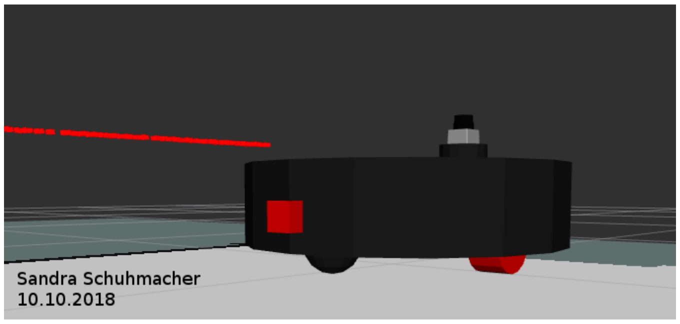

Fig. 2. Robot Model

4 Results

4.1 RTAB Map Loop Closures

An example of an RTAB-Map loop closure can be seen in 3. The robot model is driven around the environment by using manual teleop controls. During this, rtabmap creates a database where the mappings are recorded. By inspecting the database after the run, the mapping performanced can be observed. When the robot revisits a position several times and rtabmap identifies similar poses, a loop closure is generated (see the pink circles in 3. During the mapping, 9 loop closures were detected for the kitchen world and 12 loop closures for the robot workshop world. Detailed images of the features can be found in the repository folder `report/images`, images `mapping-kitchen-world.png`, `mapping-workshop-world.png` and `mapping-session.png`

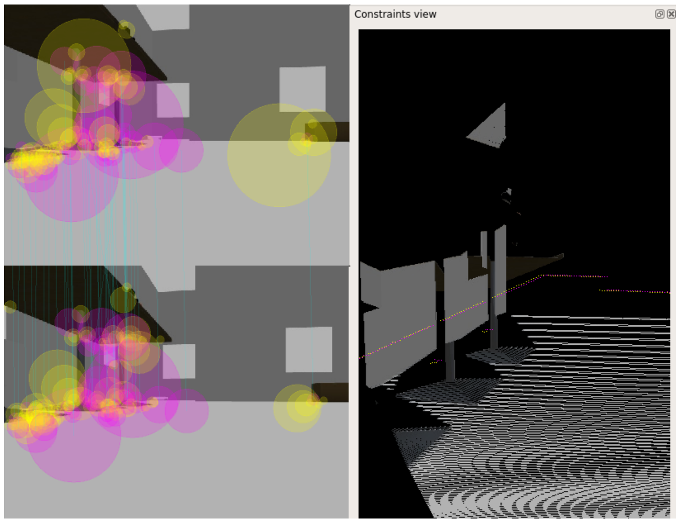

Fig. 3. Example loop closure from the rtabmapdb of the workshop world

4.2 Mapping Accuracy

4 shows the 2D maps of both environments. The movable areas are display well with some noise in the straight edges. The 2D maps are produced by the hokuyo laser range finder, which is mounted on top of the robot (see 2) and as a result, obstacles smaller than the robots height are not visible in the map. It is especially apparent in the robot workshop world, where three desks are standing at the longest wall, but they are only visible as small black dots as the desks main stand is beneath the laser sensor.

The occupancy grid maps in 5 are similar to the 2D map. They are also based in the laser scans and have the same drawbacks.

5 Discussion

Mapping the kitchen is easier than mapping the self constructed environment. The kitchen world is a very feature-rich model, with lots of different objects like chairs, various furniture, a fridge, a bookshelf etc. After getting all the ROS-parameters right to actually start mapping, the default settings already produce accurately enough mapping results.

The robot workshop initially contained less items, the walls are just grey and feature-less and the empty three desks were kind of repetitive. Sometimes false positives occured when facing either side of the row of tables. It took several experiments to get the mapping result usable and part of it was adding more items to the room to generate more features. By placing the robot puppet behind the desks and putting small items on top of the desks, the false positives vanished.

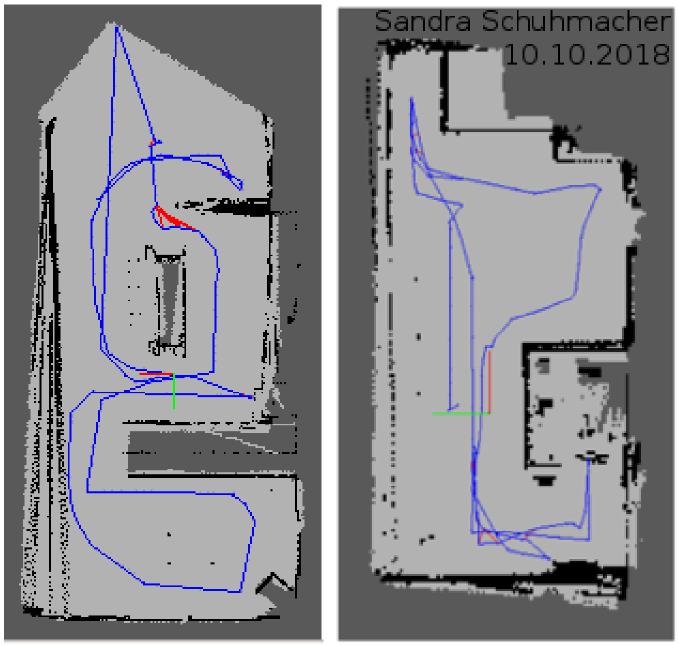

Fig. 4. 2D Maps of kitchen (left) and workshop (right) environments

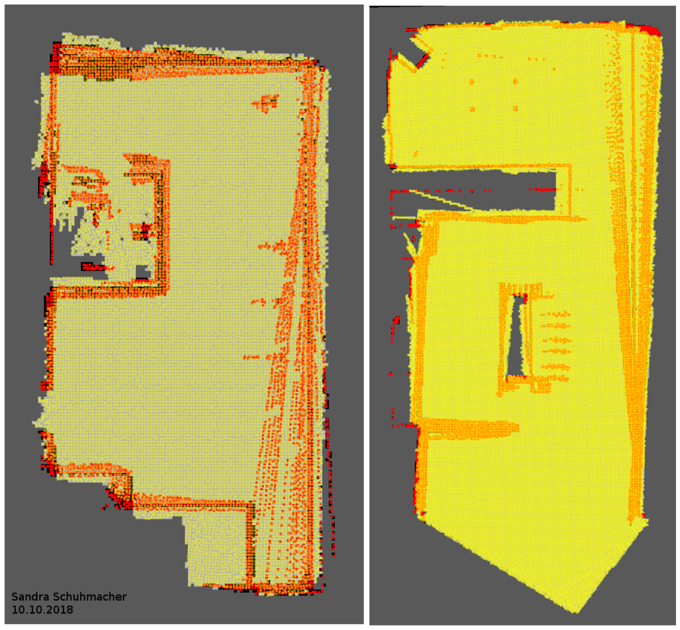

Fig. 5. Occupancy grids of kitchen (left) and workshop (right) environments

6 Future work

RTAB-Map is an interesting choice of a mapping algorithm and provided quick results without much need for extensive parameter tuning. An interesting next step would be to use it in an actual real world scenario with a real camera feed instead of a simulated environment (e.g. on NVidia Jetson TX2 Hardware). Possible exploration areas would be how well it deals with camera noise or different lighting settings. A challenge to be dealt with is going to be that RTAB-Map is expecting a static environment, so an additional component will be required to solve that issue.