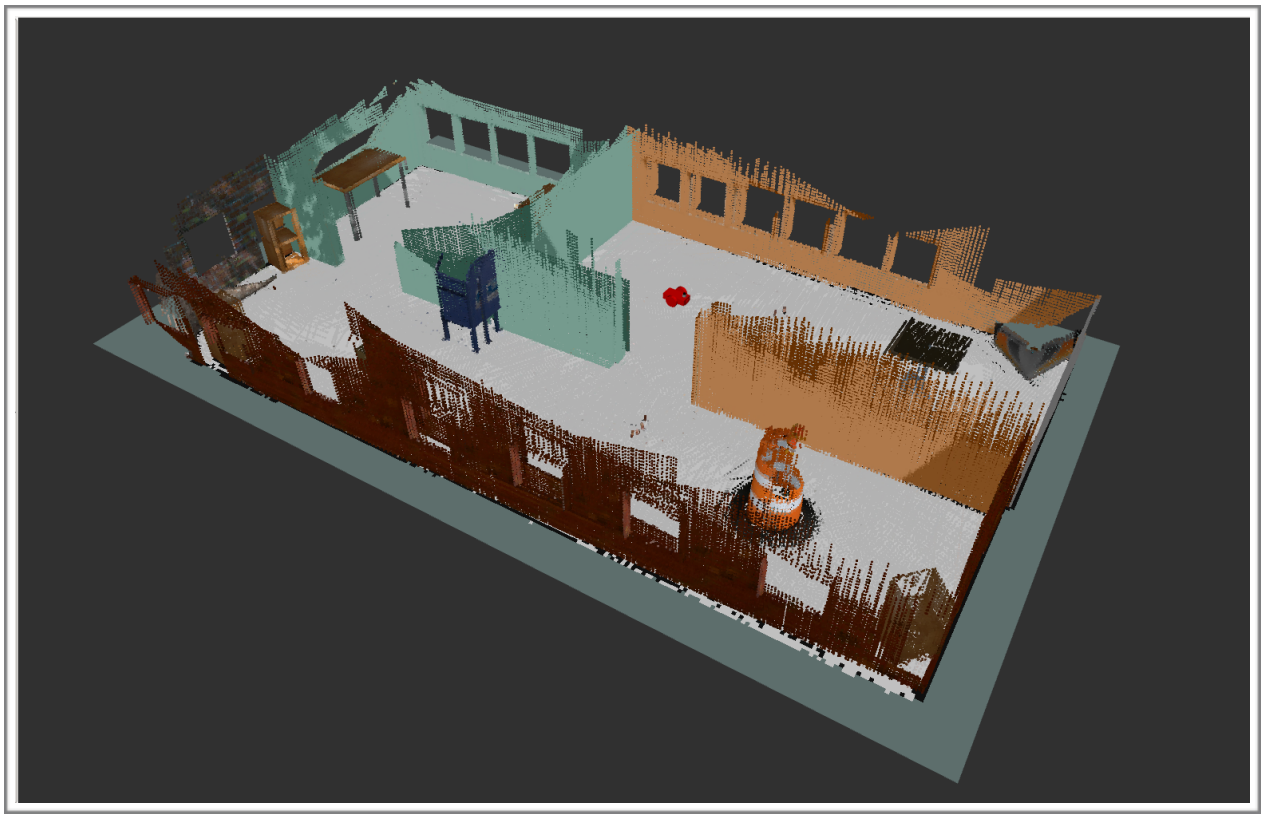
Map My World

Project Writeup
Udacity Robotics Software Engineer Nanodegree

Philipp Ludewig

March 2019



Abstract

Most mobile robot have the basic requirement to localize themselves in a given map. In reality however, an environment is often unknown or changes unexpectedly. Therefore, the ability to map an environment from scratch and localize the robot at the same time (SLAM) opens up a wide range of new applications. The goal of this project is to gain experience in using a SLAM algorithm called *RTAB-MAP* while identifying important influences on the quality of the mapping results.

1. Introduction

The simultaneous localization and mapping problem (SLAM) plays an important role for the robotics industry to expand into new and uncontrolled environments such as consumer households, warehouse applications and more. Within the scope of this project, a simulated robot will be manually steered through an unknown environment while creating 3D and 2D representations of its surroundings. By fusing a range of sensory inputs, SLAM algorithms attempt to tackle the huge challenge of combining a lot of data into an accurate outcome. This project will use a *Real Time Appearance Based Mapping* (RTAB-MAP) approach, which uses additional visual data for localization and mapping.

2. Background

While it is relatively simple to localize a robot in a known environment, a much more complex problem lies within exposing it to an unknown environment with an unknown pose. Therefore, SLAM aims to localize a robot in exactly this type of situation while mapping the new environment at the same time. In some cases, 2D mapping of an occupancy grid may be sufficient, other circumstances require full 3D mapping.

With the complex SLAM algorithms, several challenges arise. Large spaces for example can lead to huge amount of data and therefore pose computational challenges. Additionally, environments with similar looking areas can cause matching conflicts for the algorithm. Further, cyclic mapping behavior of the robot can lead to sensor error accumulation. Although there are many algorithms attempting to solve the SLAM problem, this project focuses on the following.

2.1 FastSLAM

This algorithm uses a particle filter approach along with a low dimensional extended Kalman filter (EKF) to solve the problem. It estimates the posterior over the trajectory using a modified particle filter, known as the *Rao-Blackwellized* approach. Independent features of the map are solved with the EKF.

2.1.1 Grid based FastSLAM

A variation, the *Grid based FastSLAM,* does not require known landmarks and therefore allows to map an arbitrary environment by creating a grid map. While this approach is computationally efficient, a 2D mapping solution might not be sufficient to solve a problem.

2.2 GraphSLAM

This solution has several advantages over the previous approach.

GraphSLAM uses graphs to represent robot poses and environmental features over time. By constantly creating so called *motion* and *measurement* constraints between nearby graphs, the algorithm computes the most likely *full* trajectory of the robot using *maximum likelihood estimation.*

2.2.1 RTAB-Map

The Real Time Appearance Based Mapping is a GraphSLAM approach using additional visual data for localization and mapping. It allows to create even more accurate maps by trying to match all previously seen features (global loop closure) to correct for accumulated error.

To solve the problem of the linearly increasing amount of matching computations over time, RTAB-Map uses a combination of working and long term memory. Further, the algorithm allows to apply graph optimization to further improve the output.

3. Scene and Robot Configuration

3.1 Scene Configuration

To customize the project, a simple world was built using gazebo. A base footprint of the building was created using the building editor. Following, the environment was populated with a range of database models such as tables and other objects (Fig. 1).

Fig. 1: Custom Environment Overview

3.2 Robot Configuration

As robot model, the Udacity model (Fig. 2) from a previous project was used. Consisting of two wheels attached to the cuboid shaped chassis and a caster wheel added for stabilization. An rgbd camera and lidar sensor were added for sensor input.

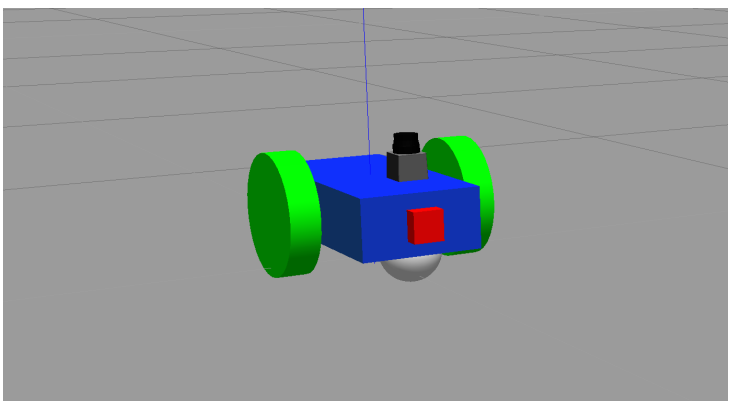


Fig. 2: Udacity Robot Model (source: Udacity)

3.3 Parameters and Package Structure

The package structure was adopted from the previous project. Besides adapting the Gazebo file, RTAB-Map required a separate frame in the URDF file to integrate the RGBD camera. Further, topics were partially remapped according to RTAB-Map documentations.

The full transform tree can be seen below (Fig. 3).

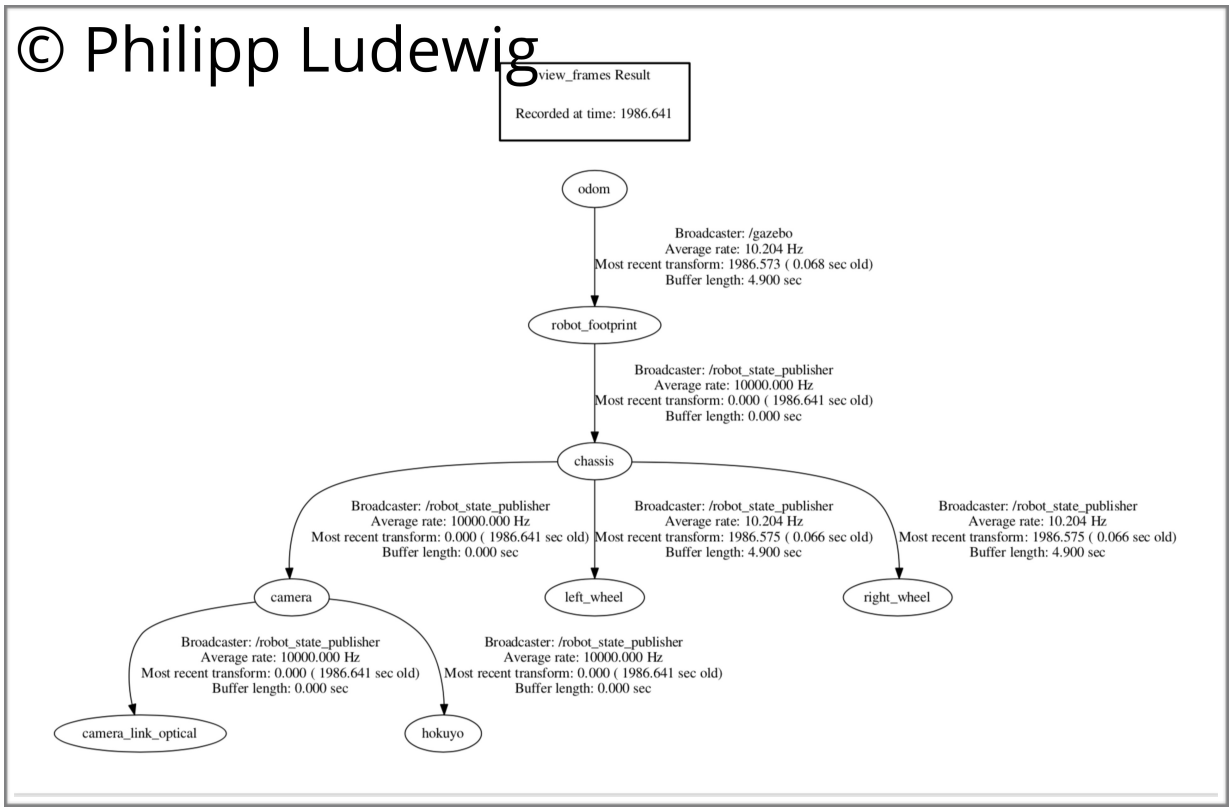


Fig 3: TF Tree

4. Results

4.1 Custom Environment

The robot model was able to map the environment and localize itself. No serious problems occurred during the mapping process. One negligent mismatch between object point clouds was discovered (Fig. 6).

Besides a 2D occupancy grid map including the robot path, 3D point cloud maps were created.

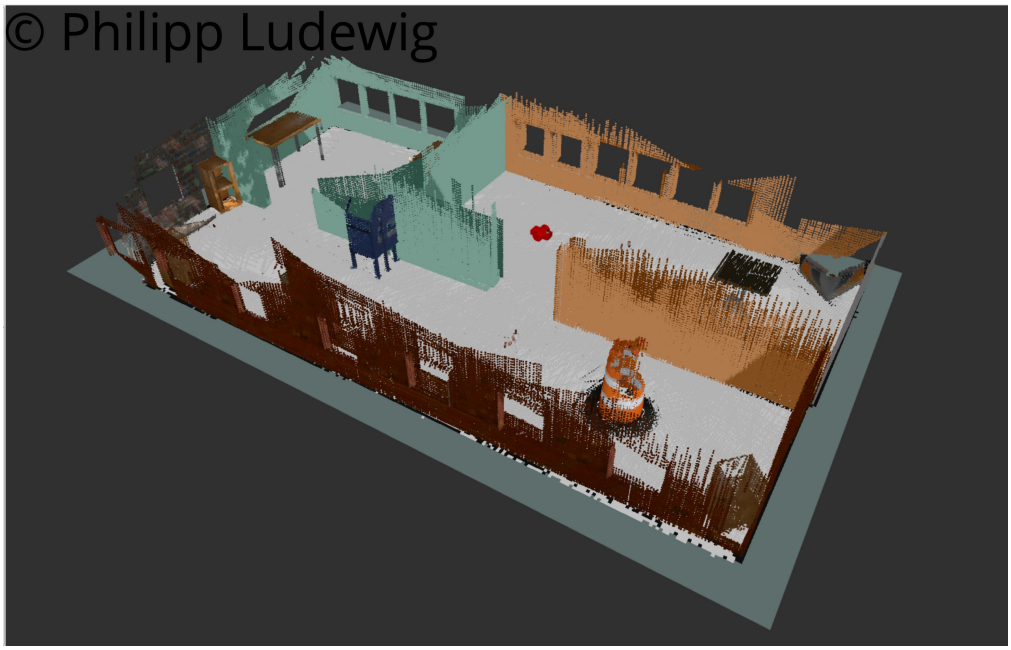


Fig 4: 3D Point Cloud of Custom Environment

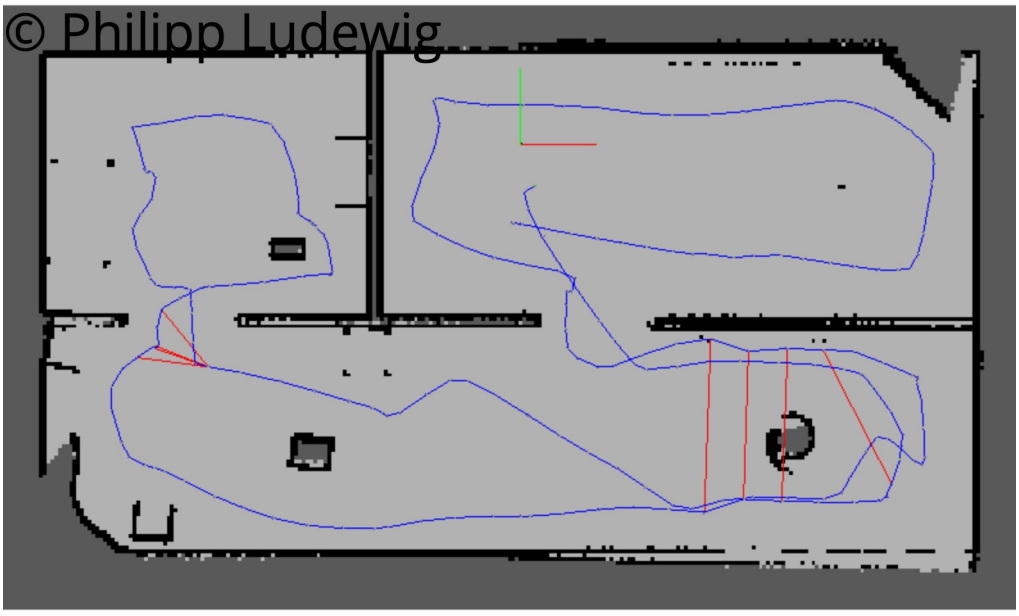


Fig 5: 2D Occupancy Grid Map with Robot Path

Fig 6: Matching Error

4.2 Demo Environment

A demo environment was provided to compare the results. However, no significant differences in accuracy were found. The results can be seen below.

Fig 7: 3D Point Cloud of Demo Environment

Fig 8: 2D Occupancy Grid Map of Demo Environment

5. Discussion

Overall, the mapping results were found to be sufficiently accurate and are discussed further below. Three passes of the demo environment lead to 92 global loop closures.

5.1 Comparing Environments

Both environments did not pose any major issues for the RTAB-Map algorithm. However, in order to increase the amount of loop closures, many distinct features were particularly helpful in the custom environment. Despite fewer passes here, the amount of global loop closures accounted to nearly twice as many as with the demo environment.

5.2 3D Maps

The resulting 3D maps have turned out to have a few matching errors present. While point cloud accuracy seemed accurate during the first pass, the second and third pass would sometimes lead to error corrections due to detected global loop closures. These corrections could sometimes result in point cloud errors, however, the problem was negligent.

5.3 2D Maps

The occupancy grid map was created without noticeable errors. No immediate improvements could be made here.

5.4 Notes and improvements

In general, the RTAB-Map default parameters delivered sufficiently accurate results. Besides diving further into parameter tuning, a few improvements could potentially lead to even better mapping.

For example, a higher placement of the RGBD camera could have possibly resulted in a better mapping quality in higher areas of the map (e.g. kitchen countertops).

Further, mapping at large speeds turned out to cause greater matching errors upon loop closure detection. Low speeds are therefore recommended.

Also, the robot should navigate on similar paths with each pass. This improves image comparison and loop closure detection.

In conclusion, the RTAB-Map package has turned out to be a simple tool, requiring only odometry, RGBD and lidar data as input.

6. Future Work

The RTAB-Map algorithm was so far only used in a simple 2D robot movement. Several variations exist to even use visual odometry as input. This would allow applications on drones or other applications were traditional odometry is not feasible.

Additionally, the algorithm could be tested in environments that are much less feature rich as the ones tested within this project.