Simultaneous Mapping And Localization Using the Real-Time Appearance-Based Mapping Algorithm

Antti Hietanen

Abstract—This paper presents a simultaneous localization and mapping (SLAM) problem in a simulated environment which is solved using the Real-Time Appearance-Based Mapping (RTAB-Map) algorithm. During the course of the work two different worlds and a mobile robot are designed and created. The worlds try to simulate real environments from real world such as kitchen and playground. The wheeled mobile robot is equipped with two different visual sensors and an odometry sensor which are together used to construct a 2D occupancy grid and a 3D point cloud from the environment. In addition the RTAB-Map algorithm is studied and evaluated using the created environments and the robot. The performance of the algorithm is reported for both of the environments and the results show that the algorithm can succesfully solve the SLAM problem.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, SLAM, RTAB-Map.

1 Introduction

The ability of a robot to map an environment and simultaneously localize itself in it is known as *Simultaneous Localization and Mapping*, or SLAM. The topic has been hugely popular among mobile robotics community for decades and it has numerous applications ranging from spatial exploring to self-driving cars. Despite its long-term interest there still exists problems to be solved. For instance in an online applications where computation power in the terms of speed and memory is limited and the environment where the robot operates is unstructured and dynamic leverage the task difficulty.

In this work a ROS package is created to evaluate the RTAB-Map algorithm [1]. The created ROS package contains descriptions to create two different simulated environment, *kitchen_dining* and *play_ground*, and a wheeled mobile robot. The package also contains launch commands and parameter configurations for the RTAB-Map algorithm.

2 Background / Formulation

The SLAM problem is usually solved in a probabilistic fashion. The idea is to be able estimate the state of the robot (etc. 6-DoF pose) and at the same time create a map (2D grid, 3D point cloud) of the environment.

The SLAM problem has two different forms, *Online SLAM* and *Full SLAM*. In the Online SLAM problem, the most recent pose is solved independently from all the previous sensor measurements and controls and the whole problem can be modelled as:

$$P(x_t, m|z_{1:t}, u_{1:t}) \tag{1}$$

In contrast in the Full SLAM problem the posterior over the entire path along the map is estimated given all the sensor measurements and the controls up to time t. The Full SLAM problem is noted as:

$$P(x_{1:t}, m|z_{1:t}, u_{1:t}) \tag{2}$$

The second key feature in the SLAM problem is related to its nature. In SLAM there are *continues* and *discrete* elements. During SLAM, the robot continuously senses the environment with different sensors and uses the information from the sensors to estimate the robot's pose and the position of different landmarks and objects in the environment. Thus, both robots poses and objects locations are continuous aspects of the SLAM problem. In contrast the robot has to all the time infer if a newly discovered location is actually already visited location. To conclude if the place has been already visited the robot must identify if the place contains familiar objects or landmarks which the robot has stored to its database. The process of detecting an already visited the place has a binary output – either yes or no. The relation, or *correspondence* between objects forms the discrete element of the SLAM problem.

One of the earliest contributions to solve the SLAM problem was by Smith et al. [2] where the authors proposed to use an extended Kalman Filter (EKF) for estimating the robot pose along with the position of the landmarks. However the algorithm does not scale to larger environments where the computational complexity would become an issue. This is due to the fact that sensor updates require time quadratic in the number of landmarks [3].

Today there exists wide variety of different SLAM algorithms which differ from each other mainly based on the map representation and the underlying estimation technique. One of the most successful approaches are grid- and graph-based approaches.

2.1 FastSLAM and Grid-based FastSlam

The FastSLAM algorithm [3] solves the Full SLAM problem with known correspondences. In the core of the approach is the Monte Carlo Localization algorithm and the low-dimensional EKF algorithm:

Estimating the Trajectory – FastSLAM estimates a posterior over the trajectory using a particle filter approach. This will give an advantage to SLAM to solve the problem of mapping with known poses. Because the algorithm uses particles to estimate the robot pose, some roboticists consider the algorithm to solve also the Online SLAM problem.

Estimating the Map – FastSLAM uses a low dimensional Extended Kalman Filter to solve independent features of the map which are modeled with local Gaussian.

There exists several variants of the FastSLAM algorithm such as *FastSLAM 1.0*, *FastSLAM 2.0* and *Grid-based FastSLAM* each having different advantages and disadvantages.

The Grid-based FastSLAM is an extension to the original FastSLAM algorithm where the environment is modelled using grid maps without predefined any landmark positions. This allows the Grid-based FastSLAM algorithm to solve the SLAM problem in an arbitrary environment. As with the default version of the algorithm the robot trajectory (e.g. robot pose) is solved using the particles where each of the particles resemble the current pose of the robot. In addition, each particle maintains its own map. The Grid-based FastSLAM algorithm will update each particle by solving the mapping with known poses problem using the occupancy grid mapping algorithm. Three different techniques are required for adapting the FastSLAM to to grid-based mapping:

Sampling Motion – Estimates the current pose given the k-th particle previous pose and the current controls u.

$$P(x_t|x_{t-1}^{[k]}, u_t) \tag{3}$$

Map Estimation – Estimates the current map given the current measurements, the current k-th particle pose, and the previous k-th particle map.

$$P(m_t|z_t, x_t^{[k]}, m_{t-1}^{[k]}) \tag{4}$$

Importance Weight – Estimates the current likelihood of the measurement given the current k-th particle pose and the current k-th particle map.

$$P(z_t|z_t, x_t^{[k]}, m^{[k]}) \tag{5}$$

2.2 GraphSLAM

As the FastSLAM algorithm GraphSLAM also solves the Full SLAM problem. The algorithm uses graphs to represent robot poses and the environment and the algorithm consists of four main components represented in the graph, *Poses*, *Features* (extracted from the objects in the environment), *Motion constraints* (tie together two poses) and *Measurement constraints* (tie together a feature and a pose). At the core of GraphSLAM is graph optimization process. GraphSLAM uses a optimization routine to minimize the error in all the constraints in the graph using *maximum likelihood estimation* (MLE) to structure.

Rtab-Map (Real-Time Appearance-Based Mapping) is a Graph-based SLAM approach and in this work it is used to solve the SLAM problem. The approach uses bag-of-words to determine how likely a new sensor measurement is from a new location or from previously visited location. A novel memory management is used in the approach to achieve real-time performance on a large scale-environment.

3 Scene and Robot Configuration

3.1 Robot model

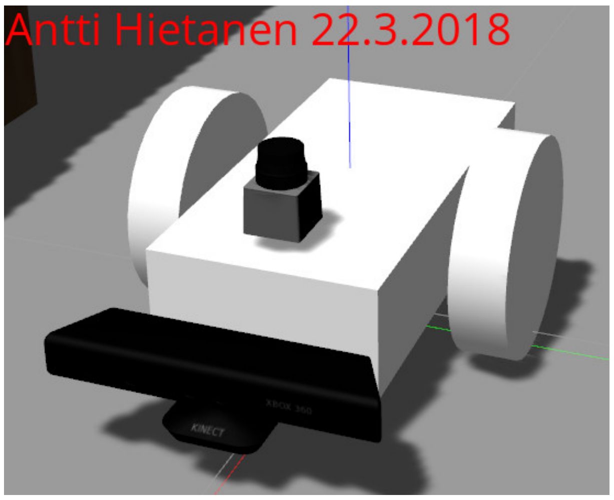

Fig. 1. Udacity_slam_bot.

In this work the wheeled mobile robot from the localization task was adopted and modified for the SLAM task and it is shown in Fig. 1. The model consists of the robot base chassis and two wheels. In addition the robot was equipped with a laser scanner and a RGB-D camera which were used for mapping and localization. The sensors were placed in front of the robot so that they had unblocked view of the environment.

3.2 Worlds

Two world models were created for this work, *kitchen* and *play_ground*. The kitchen environment simulates an indoor dining room whereas as the play_ground is a outdoor environment for kids to play around. The world models are illustrated in Fig. 2.

4 Results

The Rtab-MAP algorithm was tested in both environments and in the following sections results of the localization and mapping process is illustrated. In both experiments the robot was operated using the *teleop* node.

4.1 Dining room

The RTAB-Map algorithm was able to successfully to solve the SLAM problem in the kitchen_dining world and the final reconstructed 3D and 2D views of the environment are shown in Fig. 3. In the experiments default parameters given by Udacity program were used. In total of 5 loop closures were found, most of them were identified from images which contained the oven. This indicates that the oven can provide meaningful features (SURF keypoints) and can be used reliably to detect loop closure.

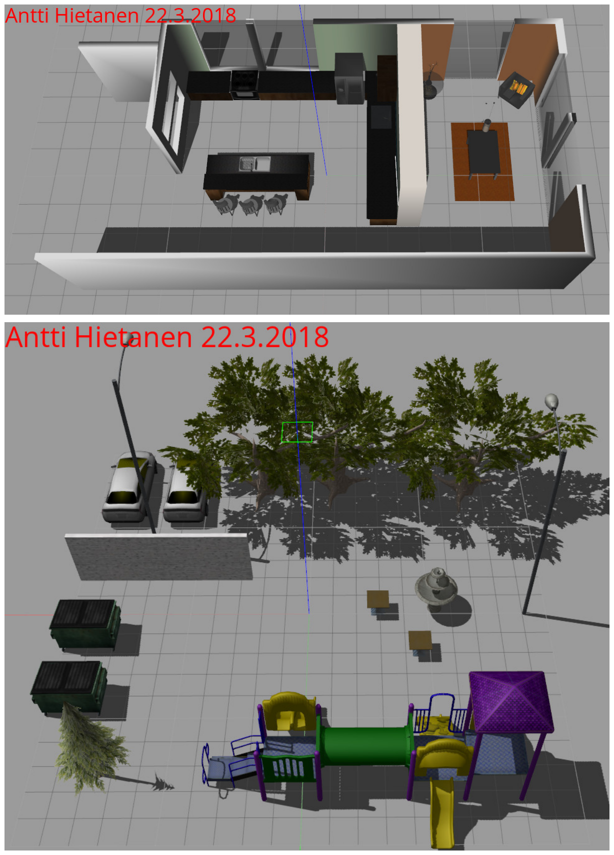

Fig. 2. Two simulated environments, dining room (top) and playground (below).

4.2 Playground

The RTAB-Map algorithm was successfully evaluated in a simulated outdoor environment mimicking a kids playground and the results are shown in Fig. 4. In total of 3 loop closures were found from near the playgound obstacle which was the most textured object in the environment. As with the dining room the default parameter were used which were noted to work well in both environments.

5 Conclusion / Future work

In this work the RTAB-Map algorithm was evaluated on a SLAM problem. For the work a wheeled mobile robot and two simulated environments were designed and created. Using the RTAB-Map algorithm and the onboard sensors such as the odometry, the laser scanner and the RGB-D camera the robot was able to localize itself and simultaneously create a map of the surrounding environment. However, during the experiments it was noted that the algorithm can fail if the environment contains repetitive and textured surface which can easily lead to wrongly assigned correspondences during loop closure (Fig. 5). In addition the algorithm might have difficulties in an environment which contains lots of dynamic objects such as a busy street. Future work includes testing other SLAM approaches in the environments.

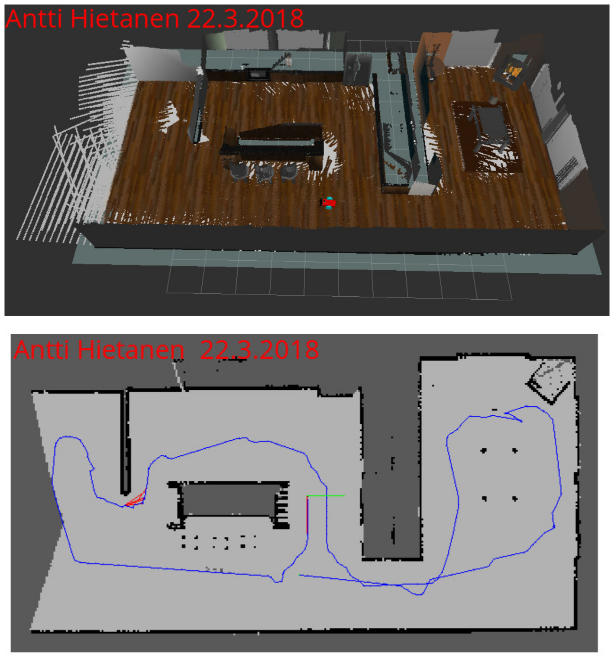

Fig. 3. Constructed views of the environment. A 3D map of the dining room using point cloud data and known camera poses (top) and 2D grid map (below).

References

[1] M. Labbe and F. Michaud, "Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2661–2666, Sept 2014.
[2] R. Smith, M. Self, and P. Cheeseman, "Autonomous robot vehicles," ch. Estimating Uncertain Spatial Relationships in Robotics, pp. 167–193, New York, NY, USA: Springer-Verlag New York, Inc., 1990.
[3] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

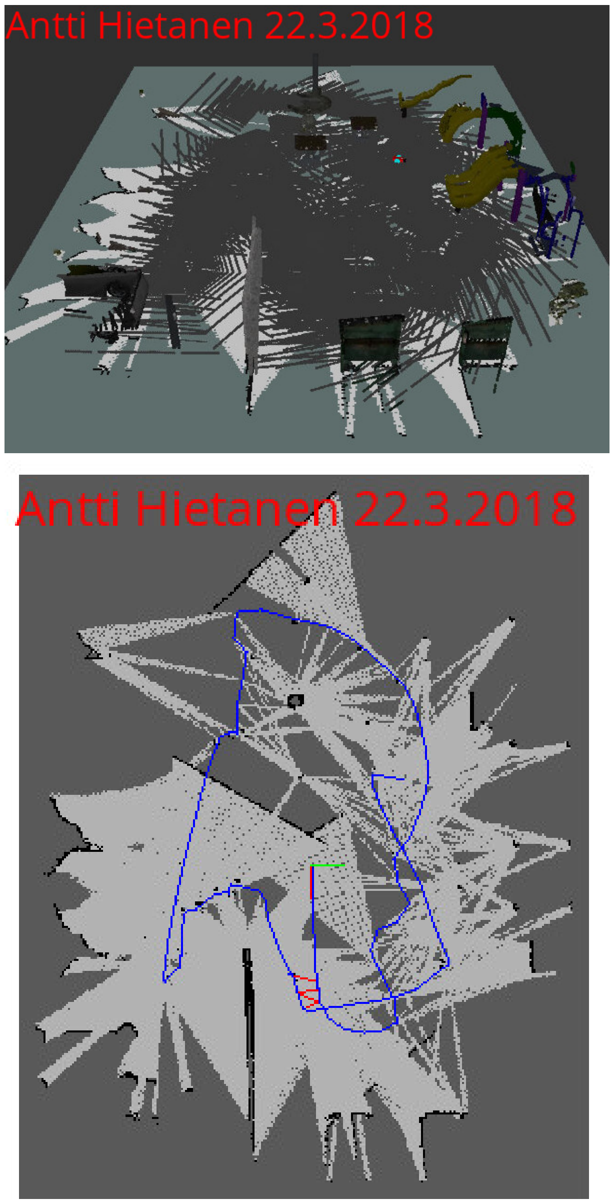

Fig. 4. Constructed views of the environment. A 3D map of the playground using point cloud data and known camera poses (top) and 2D grid map (below).

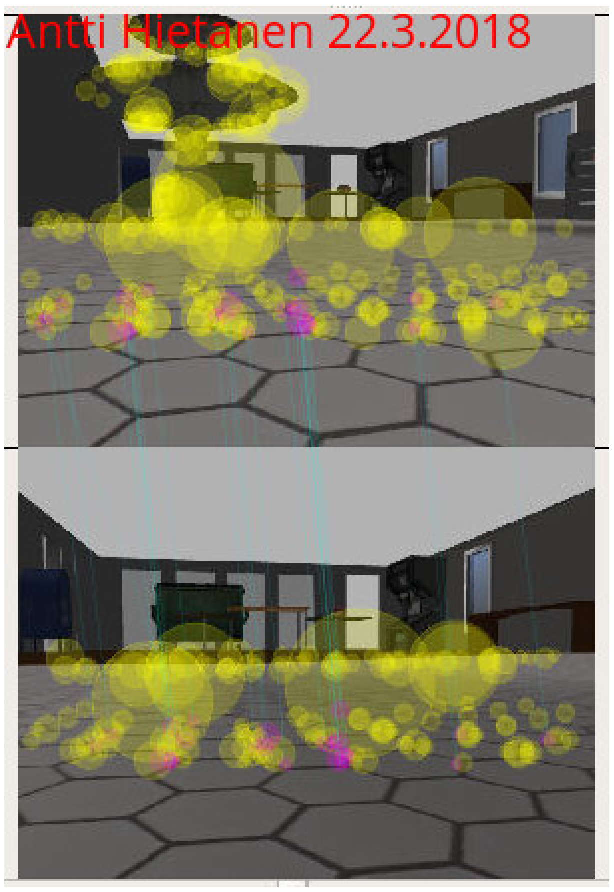

Fig. 5. Incorrectly assigned correspondences between two frames.