A Mobile Robot Localization Using the Adaptive Monte Carlo Algorithm

Antti Hietanen

Abstract—This paper presents a localization problem in a simulated environment which is solved using the Adaptive Monte Carlo Localization (AMCL) algorithm. During the course of the work two different mobile robots are designed, created and evaluated in a simulated environment containing barrier obstacles. The created robots are equipped with two different cameras and an odometry sensor which are together used to monitor the environment and to collect information about the robot actions. In addition two different localization algorithms for mobile robots from the literature are reviewed and optimal parameters for the open-source implementation of the AMCL algorithm for navigation task are searched and reported in the document.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, deep learning.

1 Introduction

In mobile robotics applications that require autonomous capabilities, navigation is usually regarded as the most fundamental problem. Successful navigation requires four building blocks: *perception* - the robot must extract meaningful information from the environment; *localization*- the robot must determine its position in the environment; *cognition*- the robot must decide how to act to achieve its goals; and *motion control* - the robot must modulate its motor outputs to achieve the desired trajectory. This work focuses on the localization block which has gained considerable attention during the recent years in academia and industry.

Indoor mobile robot localization is still considered a challenging problem due to the lack of accurate GPS which can be used in outdoors to extract precise location of the robot. With wheel-based mobile robot the movements of the robot can be estimated using an odometry, such as wheel encoder, which measures the angular positions of a wheel. However due to slipping and sliding of the wheels the measurement are noisy and the estimation error accumulates fast over time if the robot location is not updated using other localization methods.

A natural choice to extend the localization capabilities is to use vision sensors which are today lightweight and cheap. The vision-based localization is not trouble-free approach either due to changes in the dynamic environment and often the mobile robots have to solve complex perception tasks.

2 Background / Formulation

A robot localization refers to the problem of tracking the location of a robot over time using measurement which are usually noisy. This is important task for every autonomous mobile robot, ranging from aerial robots to self-driving cars. One of the most well known localization algorithms are the Kalman filter [1] and the Monte Carlo Localization (MCL) algorithm [2]. Both have been utilized in many real world applications successfully.

2.1 Kalman filter

Kalman filter has proven to work well on a noisy environment and still be able to estimate the robot location accurately. In its essence, the algorithm describes the belief of its state using a Gaussian distribution. In this representation the mean value is the expected state (e.g. the robot location) and the covariance matrix is the uncertainty of this belief. The algorithm runs in a two-step process. In the prediction step, the filter produces estimates of the current state variables. When a new measurement arrives (usually corrupted with noise), the state is updated using a weighted average, with more weight being given to estimates with higher certainty. However the default Kalman filter has two major limitation based on the filter assumptions:

- The filter assumes that the system and the observations are both linear, which is not true in real life scenarios.
- The state belief is assumed to be Gaussian distributed.

The drawbacks narrow the possible user cases for the default Kalman filter and thus an Extended Kalman Filter (EKF) was introduced.

The main purpose of the EKF algorithm is to extend the default Kalman filter to nonlinear systems. To linearize a nonlinear motion or measurement function one have to utilize multi-dimensional Taylor series. After linearizing the state and measurement equations the standard Kalman filter formulas are applied to the linearized equations.

2.2 Particle Filters

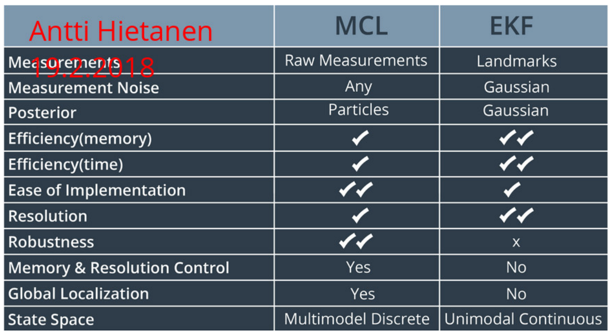
A particle localization approach uses a set of particles to estimate a pose of an robot. A single particle can be thought as a resemble of the robot, where the particle has a position and orientation and represent an estimation of the robot location. By default predefined number of particles are randomly spread around the initial position of the robot. While the robot moves each of the particles is resampled based on the measurement updates and hopefully while the robot has moved for a while in the environment each of the particles have converged to the robot actually position.

One of the most popular algorithms utilizing particles is the Monte Carlo Localization algorithm. The algorithm consists of two main processes, 1) motion and sensor update and 2) resampling process. The default version of the MCL algorithm consists of the following steps:

1) The algorithm receives the previous belief of the robot state, the actual command and the sensor measurement. Initial belief is generated by randomly generating M particles.
2) Hypothetical state is computed using the previous state and the actuation command. After that each particle weight is updated using the generated hypothetical state and the sensor measurement. Both updates are then added to the previous state belief.
3) In the resampling process, the particles with high probabilities survive and are re-drawn in the next iteration while the others are removed.

In this work the robot was localized using the Adaptive Monte Carlo Localization algorithm where the number of particles dynamically changes. The number of particles is adapted on-line, and large number of particles is only invoked when necessary .

2.3 Comparison of Extended Kalman Filter and MCL


Antti Hietanen
19.2.2018

	MCL	EKF
Measurements	Raw Measurements	Landmarks
Measurement Noise	Any	Gaussian
Posterior	Particles	Gaussian
Efficiency(memory)	✓	✓✓
Efficiency(time)	✓	✓✓
Ease of Implementation	✓✓	✓
Resolution	✓	✓✓
Robustness	✓✓	x
Memory & Resolution Control	Yes	No
Global Localization	Yes	No
State Space	Multimodel Discrete	Unimodal Continuous

Fig. 1. Comparison between MCL and EKF algorithms in different domains.

In a linear system Kalman Filter is an optimal solution. In a system that is nonlinear the particle filter may give better results but in the cost of more CPU cycles. The particle filter transforms a set of points which represent the robot location using a known nonlinear transformation and combines the results to estimate the mean and covariance of the state. However the extended version of Kalman Filter can be also used in nonlinear systems. The differences between EKF and MCL are summarized in Fig. 1. For the work an adaptive version of the MCL was selected.

3 Results

In Fig. 2 is illustrated the result for Udacity_bot. On the left image is shown the robot in the default position when it is summoned into the simulator having 500 particles randomly sampled around the robot. On the right the same robot has reached the goal state. From the image we can see

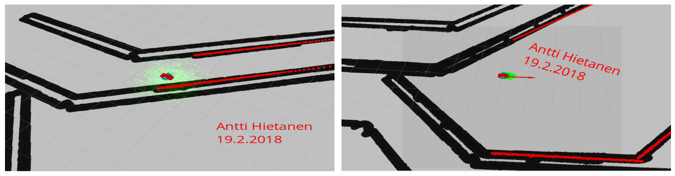


Fig. 2. The Udacity_bot in the start position (left) and in the goal position (right).

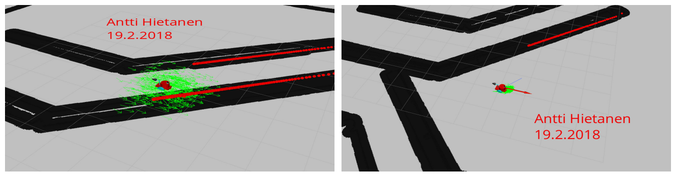


Fig. 3. The Udacity_bot_v2 in the start position (left) and in the goal position (right).

that the particles have converged and they closely resemble the robot current position. This means the AMCL algorithm can successfully estimated the robot position using laser sensor and odometry information.

The results for Udacity_bot_v2 are shown in Fig. 3. Like with Udacity_bot the version two was also successful to navigate to the goal state. In addition all the particles were converged close to the robot real state.

4 Model Configuration

4.1 Robot models

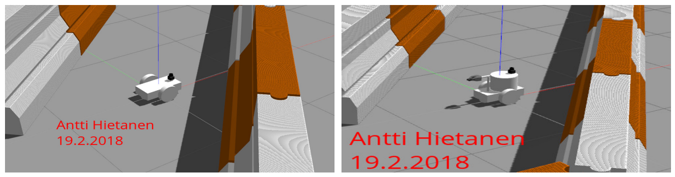


Fig. 4. Two different robot models used in the work, Udacity_bot (left) and Udacity_bot_v2 (right).

In this work two different wheeled mobile robots were designed and are shown in Fig. 4. The first model consists of the robot base chassis and two wheels. In addition the robot has two sensors which are used for localization. In the modified version (Udacity_bot_v2) the laser scanner was positioned higher using a cylinder object which was placed on top of the base chassis. In addition a robotic arm was added at back side of the robot. Visual components from the *urdf_tutorial* ROS package were used for the gripper.

4.2 Hyper-parameters

In this section the hyper-parameters which had the most influence regarding the particle converge and the robot ability to reach the goal position are described.

transform_tolerance – The parameter specifies the delay in transform (tf) data that is still acceptable. The parameter is a trade-off between the accuracy and the robustness of the system and in the work it was tuned to 0.2. With the default value of 0.0 the robot did not move.

number of particles – For the AMCL algorithm the min and the max boundaries for the particle count can be adjusted. In the work the max particle number was reduced to 500 from the default 5000. Reducing the particle count increased the overall efficiency of the algorithm without sacrificing too much from the accuracy.

odom_alphaX – Five different odom_aplha parameters model different kind of noise from the odometry measurements. The parameters were one of the key elements to increase the converge time of the particles. By default the parameters had too high values and the particles were unable to converge at all when the robot explored the environment. By reducing all the odom_alpha parameters the performance of the algorithm increased a lot.

obstacle range – The parameter defines maximum distance from the robot at which an obstacles will be inserted into the cost map in meters. In the work the parameter value was set to 5.0 meters.

raytrace_range – The maximum range in meters at which to raytrace out obstacles from the map using sensor data. In the work the value was set to 8.0 meters.

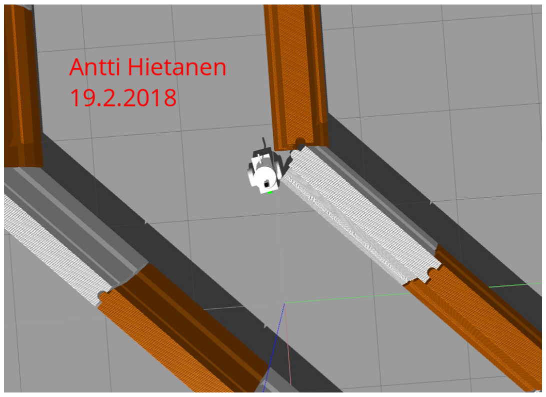


Fig. 5. The Udacity_bot_v2 stuck on an obstacle.

robot_radius – The parameter describes the radius of the robot in meters. In the ROS documentation the parameter is stated only be useful for circular robots but in the experiments with the Udacity_bot_v2 it was noted that using too small value the robot could get stuck on the walls (Fig. 5). However using too big value the robot usually stopped due to the fact that there was not big enough passage for the robot.

yaw_goal_tolerance – The parameter describes the rotation tolerance in radians around z-axis which is acceptable between current position and the goal position. In the work the parameter value was decreased to achieve more accurate pose in the goal state.

5 Discussion

Both robots were able to reach the goal position in the experiments after careful parameter tuning of the AMCL algorithm. Without any modifications the robots did not even start to move. In both experiment the two different robots took almost identical paths to reach the goal position. However in both cases the selected route was only a suboptimal and in both experiments the traveling time was approximately 6 minutes and 30 seconds. The traveling time could be reduced significantly with more careful tuning of the parameters, especially the parameters related to the namespace *base_local_planner::TrajectoryPlannerROS*.

The version 2 of the *Udacity_bot* performed almost identically as the initial version of the bot. However the robot could get stuck more easily on the obstacles on the route which might be due to the fact that the laser was positioned higher than in the first version. The white/orange barriers were narrowing towards the top which made the localization algorithm to assume the barriers to be slimmer than they actual were from the bottom.

The AMCL algorithm can be used for global localization problems and it can even recover from a situation where the robot is suddenly teleported to a new location. Using the random and dynamic sampling process of the particles AMCL can relocate itself in most of the cases. An ideal real word environment for the Udacity_bot_v2 programmed with AMCL algorithm would a warehouse with a flat surface. In the warehouse the robot could manipulate packages using the robotic arm and navigate autonomously inside the building.

6 Conclusion / Future work

In this work two different robot models were created and in the experiments both robots were able to reach the invoked goal position in a simulated environment. Using the onboard sensors such as the odometry and the laser scanner the robots were able to localize themselves and avoid collisions with the obstacles. During the experiments it was noted that the path planning algorithm was not working optimally and the robots always started the experiment by moving to the opposite direction where the actual goal position was. This be might a parameter issue and the parameter tuning to optimize the path planning will be left as a future work. In addition two different robot models were successfully designed. The second version of the mobile robot was equipped with a robotic arm and in future work a controller will be created for the arm's actuators.

References

[1] G. Welch and G. Bishop, "An introduction to the kalman filter," tech. rep., Chapel Hill, NC, USA, 1995.

[2] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.