Overview and Simulation of Adaptive Monte Carlo Localization & Navigation Techniques in ROS

Stephan Becker

Abstract—Two robot models are implemented as Gazebo models and tested on a simulated obstacle navigation course in ROS (Robot Operating System). The robots are able to perceive their environment using a LIDAR module and odometer measurements. Using AMCL for localization within a given map and a trajectory planner plug-in, they are required to navigate to a predefined goal position. Both robots are eventually able to reach their goal, although the calculated trajectory is not always optimal. The tuned parameters for the AMCL plug-in as well as the trajectory planner plug-in are presented and discussed.

Index Terms—Robot, IEEEtran, Mobile Robotics,Kalman Filters, Particle Filters, Localization.

1 Introduction

In a real world environment, localization is a difficult task for a robot. The measurements the robot makes of the environment are inherently noisy and movements are not always executed exactly, so some techniques are necessary to approximate the current position of the robot.

2 Background

For a robot to be able to successfully navigate within a simulated or real environment, an accurate estimate of the robots current pose is necessary. But measurements taken in the real world are almost always noisy, so it is not sufficient to only consider the latest measurement when estimating the current position. Furthermore the movements the robot actually makes might deviate for various reasons from the movement the robot expected it was doing (e.g. due to slippage etc.). To handle these problems with uncertain pose estimations, algorithms which not only consider the current measurements, but also the most likely estimate given the previous measurements and movements by the robot, are necessary. In practice two approaches have been established to work well in this domain: Kalman Filters and Particle Filters.

2.1 Kalman Filters

Kalman Filters are a method to model a belief state about noisy measurements using a uni-modal Gaussian distribution. The Gaussian indicates how likely it is to be in a certain state. A higher variance signifies uncertainty about the current state. Because linear Kalman Filters can only model linear behaviors, they are often unsuitable for real world applications. The Extended Kalman Filter is a way of dealing with nonlinear behaviors, although it is only approximate (approximating the non-linearity by a Taylor Expansion) and computationally expensive, due to the necessity of calculating Jacobians.

2.2 Particle Filters

Particle Filters use a Monte Carlo Method to model the current belief state. They are multi-modal and easier scalable than EKFs (the number of particles can easily adapted to the computational capabilities of the hardware). Particle Filters work by randomly guessing possible states and the resampling the states according how well they match the measurements - unlikely states are sampled with low probability.

2.3 Comparison / Contrast

The limitations of EKFs (uni-modality, not good at modeling non-linear behaviors) compared to Particle Filters make Particle Filters often the better choice in complex environments. For this project we will consider only Particle Filters.

3 Simulations

Two robot models were build and run in a RViz/Gazebo environment.

3.1 Achievements

Both robots were able to reach the goal state, as shown by the following images:

3.2 Benchmark Model

3.2.1 Model design

The benchmark model consists of a 0.4x0.2x0.1 rectangular chassis, to which two side wheels of radius 0.1 are attached. For stability the model also has a spherical front and a back caster. Measurements are made by a front-facing camera and Hokuyo LIDAR, attached to the top of the chassis.

3.2.2 Packages Used

For localization the AMCL package was used; the environment was ”jackal_race” by Clearpath Robotics.

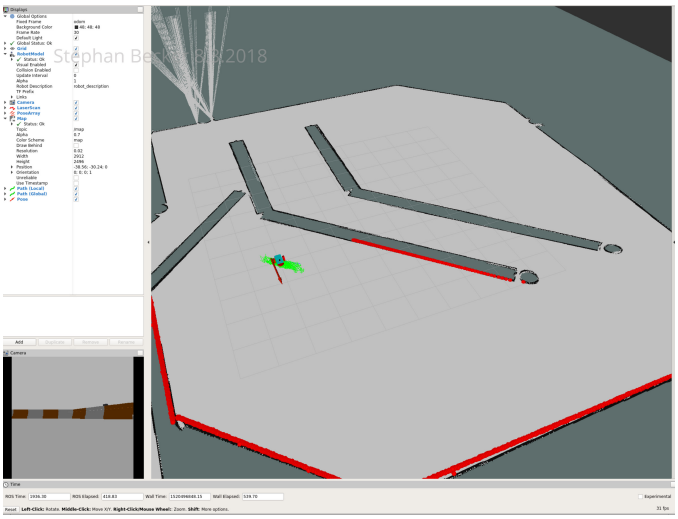

Fig. 1. Final pose benchmark model

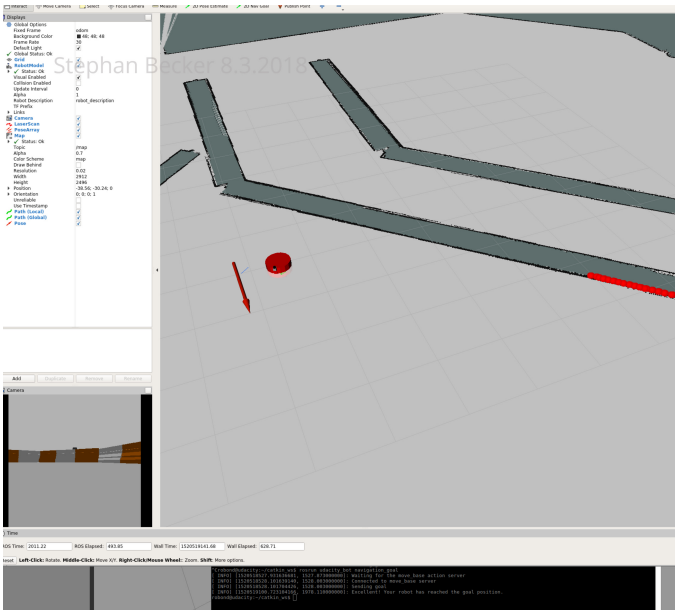

Fig. 2. Final pose personal model

3.2.3 Parameters

For the AMCL the min and max particles were set to 20 and 200 respectively; a higher number of particles could potentially be beneficial, but comes with a higher computational cost. The transform_tolerance was set to 0.2 seconds, meaning that published transforms are valid for this time interval; setting the time interval too short would lead to a lot of dropped transforms, a too long interval would result in the system processing stale data.

The following parameters were chosen for move_base: The obstacle and ray-trace ranges were increased to 5 and 8 to give the robot a further observation range. The inflation radius was set to 0.55 - indicating that obstacles are avoided at this distance.

The update and publish frequencies were set to 10 Hz. The xy and yaw goal tolerances were set to 0.05, allowing the robot to converge on the goal - using too low values would result in the robot circling around the goal, never quite reaching the exact position.

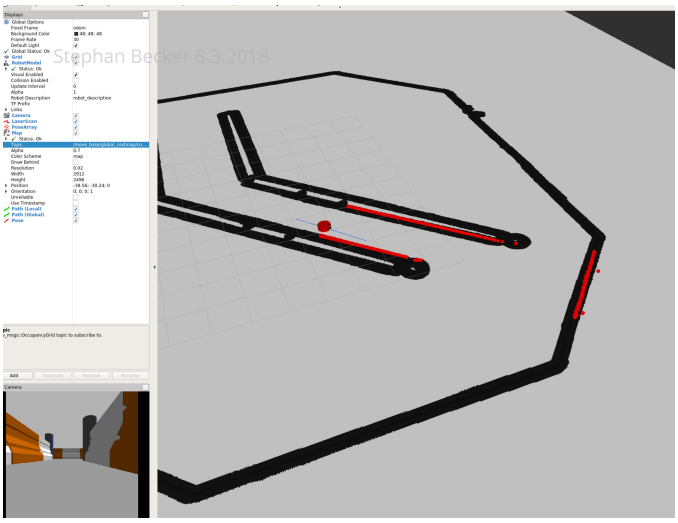

Fig. 3. Global cost map - increasing the inflation radius would make the black lines thicker, leaving the robot less space to navigate

3.3 Personal Model

3.3.1 Model design

The personal model consists of a horizontal cylindrical chassis with radius 0.2 and hight 0.2. Below the chassis are 3 wheels: to front wheels used for steering and a back wheel for stability. The camera sits at the front of the chassis and the LIDAR sits on top and at the front of the chassis.

3.3.2 Packages Used

For localization the AMCL package was used; the environment was ”jackal_race” by Clearpath Robotics.

3.3.3 Parameters

The parameters for the AMCL package were the same as for the benchmark model. As the personal model was smaller than the benchmark, some parameters were changed accordingly: The inflation radius was set to 0.3. The xy and yaw goal tolerances were set to 0.5.

4 Results

Although both robot models were able to eventually reach the goal, it took them a long time to navigate to the goal and they often took unnecessary detours. Sometimes the robots would get stuck and drive in circles or just oscillate in place. The Particle Filters were giving accurate predictions and converged rather quickly to a small area.

4.1 Technical Comparison

The personal robot was able to reach the goal quicker, but was not able to reach the position as exact as the benchmark; the xy and yaw goal tolerances had to be increased for the personal model to prevent it from circling around the goal position, being unable to eventually reach it.

5 Discussion

Overall the performance of both robots was unsatisfactory. The Particle Filters were giving accurate predictions, so this seems likely to be a problem with the trajectory planner. Comparing the two tested models, the personal model was achieving better results. This was presumably because of the compacter size of the model, giving it better stability, a smaller turn radius and more distance to navigate between the obstacles.

In a kidnapped robot problem, the real world position of the robot changes significantly (e.g. if the robot would be picked up and placed in a different location); AMCL would be unsuitable for this kind of problem, because there wouldn't be sufficient variability in the particle states once the model had converged on a small area - there wouldn't be a suitable particle which accurately described the new position.

MCL can be used in an industry environment where the robot can be fairly certain in its position and obstacles are clearly visible.

6 Future work

To improve the performance of the robot models, further tuning of the trajectory planner is certainly required. In general adding more particles should improve the accuracy of the state estimation at the cost of higher compute time, although the current number of particles seems to already be sufficient for localization.