Localization - Where Am I

Jason, Yu-Chieh Huang

Abstract—Localization is an important task for a robot to understand its position and information for path planning. This task is performed base on previous stats and sensor measurements, but they are noisy and have a certain amount of uncertainty. To accurately localize robots, AMCL(Adaptive Monte Carol Localization) is adopted to estimate possible positions. Two robots are constructed in a Gazebo environment to analysis the localization performance. After tunning parameters, both robots show good localization results.

Index Terms—Robot, Localization, ROS, AMCL, particle filters, Kalman filters

1 Introduction

Localization is one of the most important tasks of mobile robots. This task provide a location of a robot's position, and the robot can do path planning base on this information. If an accurate location can not be retrieved, the robot will deviate from a designed route, and it is likely that the robot will walk into a catastrophe, like bumping into a wall, going to wrong directions, falling into traps, etc.

To solve a localization problem, we integrate map, control data, measurements and previous state to infer where the robot is. However, these sources are not perfect, and contain different kind of noise and uncertainties. For example, a CCD camera is a popular sensor for robots, but it resolution is limited by the size of pixels, and different environments may introduce different light noise. To overcome issues, Kalman filters or particle filters are candidates to increase accuracy.

In this project, two robot models inside a simulation environment with pre-defined map are built. A ROS AMCL package is adopted to solve the localization problem. In the end, AMCL related parameters are fine-tuned to achieve a good performance for localization.

2 Background

The robot location at time T, there are data, like previous location, velocity, acceleration, control signal and sensor measurements which can be used to get the robot position. However all data contains some level of noise and uncertainties which lead to calculation error. To deal with the noise and the uncertainties of a localization problem, to extract more information from data is a way to achieve higher accuracy. Kalman filters or Particle filters are explained, and one of them will be selected for simulations.

2.1 Kalman Filters

Kalman Filters are an data fusion algorithm, and it is a recursive calculation to smooth noise and optimize estimations for linear systems. It requires small memory resource, since the computer only needs previous state, not the entire history.

The basic idea of Kalman Filters is estimations have uncertainty σ_e, while measurements have uncertainty σ_m, and by combining estimations and measurements, the fused estimation have a lower uncertainty comparing to sources:

$$\sigma_{fuse}^2 = \frac{\sigma_e^2 \cdot \sigma_m^2}{\sigma_e^2 + \sigma_m^2}$$

$$\sigma_{fuse}^2 < \sigma_e^2 = \frac{\sigma_e^4 + \sigma_e^2 \cdot \sigma_m^2}{\sigma_e^2 + \sigma_m^2}$$

$$\sigma_{fuse}^2 < \sigma_m^2 = \frac{\sigma_m^4 + \sigma_e^2 \cdot \sigma_m^2}{\sigma_e^2 + \sigma_m^2}$$

However, Kalman filters deal with linear systems. To optimize estimations of nonlinear systems, Extended Kalman filters(EKF) is a better candidate, which adopts Taylor series expansion to approximate nonlinear equations to linear equations.

Besides, all uncertainties are assumed to be Gaussian distributions in Kalman filters. In real world cases, this is not always true. Particle filters solve the problem without these limitations.

2.2 Particle Filters

The first step of particle filters is to generate hypothesis points over the map, and these points can be any distributions, including random distributions. By comparing the map and measurement signal, some points are likely to exist, and they are assigned as high probabilities; Points which are unlikely to exist are assigned as low probabilities. As time goes by, we continue to update probabilities of hypothesis points, and the optimized location should be represented by points with high probabilities.

2.3 Comparison / Contrast

Particle filters has no limitations for noise distributions compared fo Kalman filters. The drawback of particle filters is high computational resource needed, but it is not a crucial problem in nowadays computer systems. Therefore, the work presented here integrates algorithms base on particle filters.

3 SIMULATIONS

3.1 Benchmark Model

3.1.1 Model design

The benchmark model has a 0.4 x 0.2 x 0.1 chassis, and there are two wheels with two spherical slippery casters under the chassis. A camera is attached to the front face of the chassis, while a Hokuyo laser range finder is mounted on the top of the chassis.

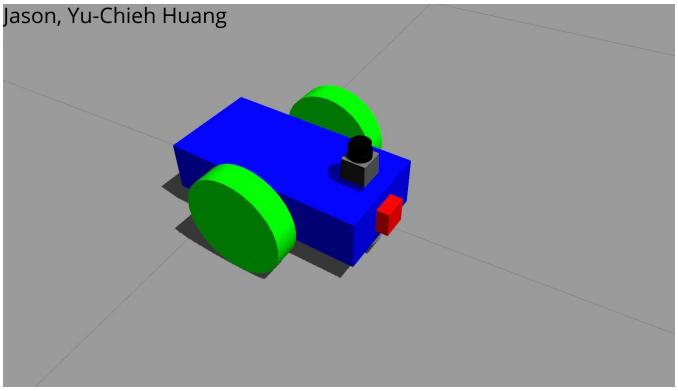

Fig. 1. The benchmark robot.

ROS AMCL [1] and move_base [2] packages are integrated in models. The move_base package controls the robot trajectory using information from the map, odom, the range finder and the tf package. The AMCL package generates PoseArrays base on the range finder and the tf package.

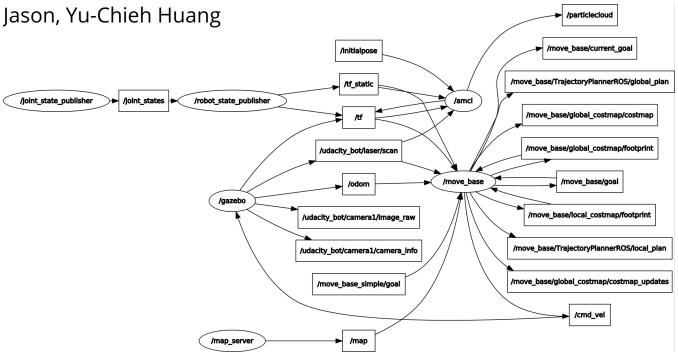

Fig. 2. rosgraph.

3.1.2 Parameters

base_local_planner.yaml from the base_local_planner package provides a controller to connect a path planner to the robot. It highly affects the robot's trajectory. When parameters are not proper tuned, the robot will oscillate in the map or collide obstacles. If the robot follows a route designed by the path planner, it can avoid diversion and hitting obstacles. Therefore, meter_scoring is set to *true*, and a path weighting, *pdist_scale*, is increased to keep the robot prefer to stay on the planned path [3].

The navigation stack stored obstacles information in costmap, *costmap_common_params.yaml* parameters are set base on the ROS wiki [4]. Obstacle_range and ray-trace_range can not be 0, and the robot can identify whether there is obstacles or free space in front of it.

The global costmap is related to a long-term path planning, and the local costmap is for a short term planning. The default parameters are kept, except update_frequency and publish_frequency. They are decreased to avoid warning messages in the console.

To increase particle filters accuracy, update_min_d and update_min_a are decreased to increase filters update and speed up convergence.

3.2 Personal Model

3.2.1 Model design

The personal model has a 0.4 x 0.4 x 0.1 chassis, and there are two wheels in the front with single rear wheel. The camera and the Hokuyo laser range finder are connected to a rod with a higher z-position.

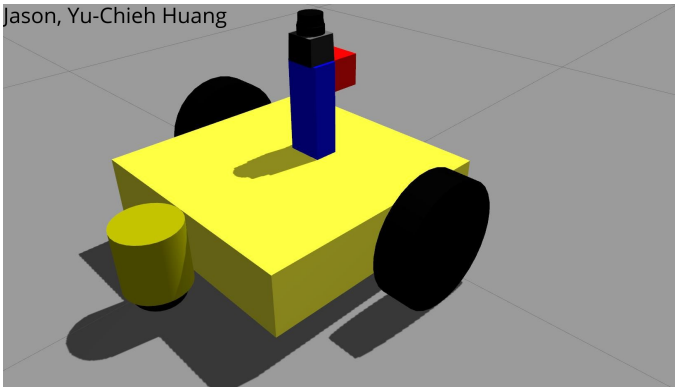

Fig. 3. The personal robot.

4 RESULTS

4.1 Localization Results

4.1.1 Benchmark

Figure 4 and 5 are images containing PostArrays for the benchmark model with different parameters.

By comparing different update_min_d and update_min_a values, figure 4 shows that improving PostArrays accuracy with smaller update_min_d and update_min_a against larger values lower accuracy shown in figure 5.

Figure 4 is the best result I have, and PostArrays are converged to a small range with high accuracy.

4.1.2 Student

Figure 6 is the best result I have for the personal model, and PostArrays are converged to a small range with high accuracy.

5 DISCUSSION

Two models demonstrate similar performance.

When tunning the personal model, it is crucial to adjust architecture of wheels. If they are not balanced, the robot will oscillate along z-axis, and the laser range finder detects on different altitude during oscillations. Sometimes it heads to for-ground, and the robot reads obstacles in front of it, but in reality, it is a free space, and accuracy decreases.

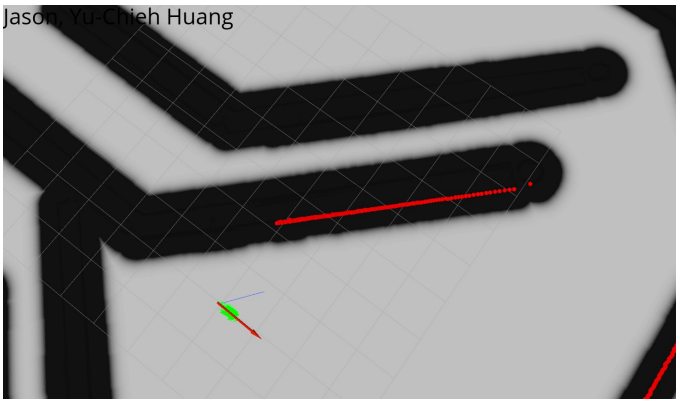

Fig. 4. A good PostArrays when update_min_d and update_min_a are 0.01.

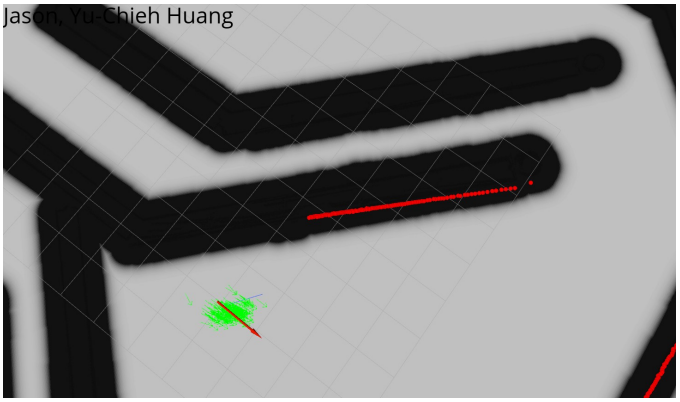

Fig. 5. A bad PostArrays when update_min_d and update_min_a are 0.2.

On the other hand, the benchmark has front and rear slippery casters which provide a good balance. After fine-tuning the balance of the personal model, it achieves similar performance as the benchmark robot.

In the kidnapped robot problem, a robot is placed in a new position without a map. Therefore, the localization system described in this report is not suitable for it. To solve the kidnapped robot problem, the robot has to construct a map and localize itself at the same time, and SLAM(Simultaneous localization and mapping) is the solution for this scenario.

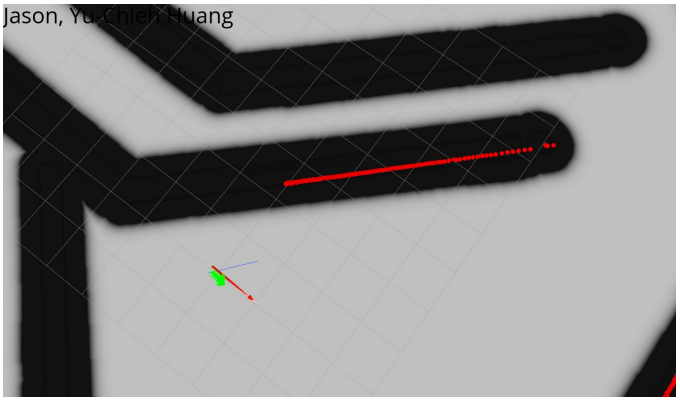

Fig. 6. A good PostArrays from the personal model.

The localization system described in this report is suitable for scenario with predefined maps. In the industry domain, it can be applied in the case like a warehouse transporters, since maps are predefined, and the environment can be well-controlled. Additional robots or objects in the warehouse have to be integrated with maps.

6 Conclusion / Future work

In this report, two robots with good localization accuracy are constructed from scratch. The navigation stack is integrated to drive robots under Gazebo environments. To ensure a smooth trajectory, parameters in *base_local_planner.yaml* are set. Different parameters in *amcl_params.yaml* show great improvements of convergence of PostArrays.

A Hokuyo laser range finder is integrated, and different devices providing laser scans can replace it.

Moreover, in the personal model, the motion control is a differential_drive_controller, and the rear wheel is only a passive support without any active driver. It will be interesting to implement a four wheels robot, like a car, which rear wheels has forward drivers and front wheels control the heading.

References

[1] *ROS wiki - AMCL package.*
[2] *ROS wiki - Move Base package.*
[3] *ROS wiki - base_local_planner.*
[4] *ROS wiki - Navigation Stack.*