Navigation with AMCL

Xueyang Kang

Abstract—To have an concrete insight into the whole work-flow about how to navigate a robot in an unknown environment, is the purpose of this paper. Navigation task normally is comprising of localization and path planning. An configurable simulation platform both for map world and robot model was established as a test platform. The tweaking work of localization and path planner parameters is also introduced in the paper as non-trival stuff. Two different robot models based on the same differential drive are considered for navigation performance evaluation.

Index Terms—Robot, Localization, Path Planner, Differential Drive, Evaluation.

1 Introduction

Navigation is a central issue in robotic field, to make the robot be aware of where it is. A bunch of localization methods can be used to solve this problem, mainly including Kalman Filters and MCL, Histogram Estimation, Bayes Filter. The adaptive Monte Carlo Localization is exploited in my work. The AMCL is dependent on both the laser and odometer measurement. And the pose generated by localization algorithm will be further passed to the path planner, to calculate the global path to the goal on a cost map. The tuning of parameters to put all the navigation packages together to work successfully is quite time consuming, if there is no clear understanding of the relationship between the nodes. To simplify this process, the gazebo based robot description model is utilized to analyze the relationship between algorithm parameters and robot model.

2 Background

The localization is try to estimate the robot pose along motion continuously. There are commonly three types of localization problems, the initial position is known, the initial position is unknown, the kidnapped problem, each kind of problem's complexity increases. Localization is the bottom task for robot navigation, assuming the robot is aware of where it is in an unknown world, it can find the path to reach the goal or to obtain the interest on its own. If the robot makes a fault guess of its current position, it will be in risk of losing its way and coming into collision with walls.

So the accuracy of localization matters a lot. In indoor environment, it is common to have many dynamic objects to affect the measurements, because the localization will use the local measurements to estimate its position, the temporary cluttered world may lead the robot to the fatal failure of localization. The work about how to integrate localization into navigation stack need some clear guide, as introduced in [1]

2.1 Kalman Filters

Kalman filters is comprising of motion update and measurement update, it at first produces the estimates of its current state, and along with the uncertainty of motion. Then when the measurement is available, the estimate is updated in a weighted sum over predict estimate and observation. The calculated uncertainty of the estimate will be reduced, compared to the uncertainty in motion and measurement alone. It is easily to be implemented in a recursive manner. But the motion model and observation model in reality are not linear often, like there is rotation in motion, so the linear approximation is infeasible for non-linear model. Extended Kalman Filter (EKF) will linearize the non-linear motion model at prior mean, non-linear measurement model at predicted mean through the first oder Taylor approximation, so this algorithm is still applicable in non-linear problem.

2.2 Particle Filters

Particle filter (PF) maintains a bunch of particles, which denotes a pose guess of the robot. It propagates the individual pose directly into motion model, then use measurement to calculate the weight of each particle. Lastly, the particles are drawn proportionally according to its corresponding weight, after that their weights are assigned same again to initiate next iteration. After several iterations the particles of pose will converge to the true value. PF is easily to be implemented in practice, additionally, it can be used even for non-Gaussian model, unlike the Kalman Filter, assuming the uncertainty of noise is Gaussian.

2.3 Comparison / Contrast

Kalman filter has some drawbacks.

- KF is not applicable in non-Gaussian model, sometimes the multi-model in measurement may occur, which may lead to the fatal error of localization.
- The inverse operation of covariance to compute the Kalman gain is also inefficient.

However, Particle Filter can solve all these problems, so we use the adaptive Particle Filter to help the robot complete localization task, in which the number of particle can vary according to the environment.

3 Simulations

Here the AMCL package is used to provide localization for robot, ”move_base” package is used to complete path planning and navigation. To be able to locate the robot properly, ”max_particles” and ”min_particles” is set to 100 and 30 individually, this setting is verified to attain a good trade-off between computational time and accuracy.

The benchmark robot model is base on differential drive with two caster wheels at bottom, and two active wheels as drive, additionally, it is outfitted with camera and laser. The personal robot model is also base on differential drive

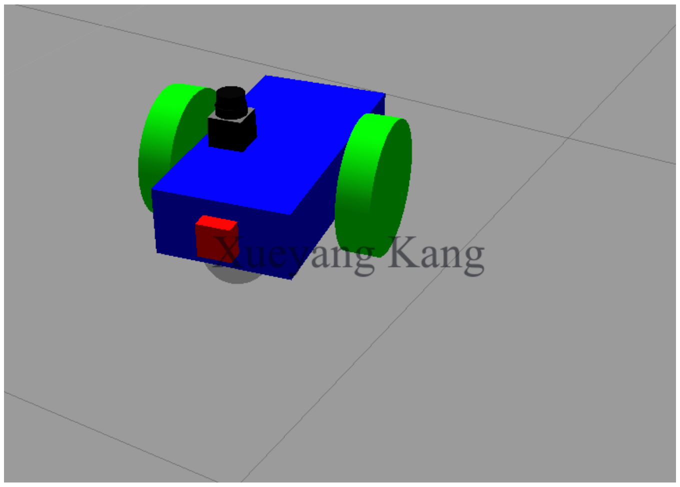

Fig. 1: Benchmark model

with two big caster wheels at bottom, and two big active wheels as drive, but is has three layers on top, and with four poles as support of the top whole structure. Each top layer is circular, and the shape of the whole robot resembles a classical ”turtle robot”.

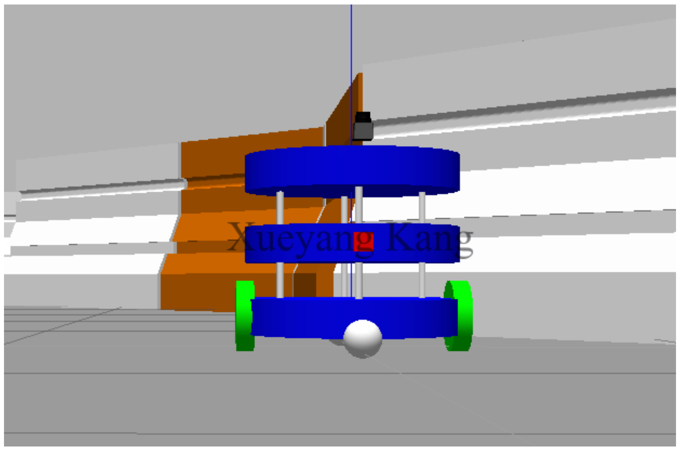

Fig. 2: Personal model

3.1 Achievements

My localization for benchmark model and my own model can work well, after the robot model is out of narrow corridor, the particles can converge to the true value, and are distributed closely around the model, the orientation of the particles are almost consistent. The table below presents the main parameters which have effect on the localization, the tuning work of AMCL is mainly focused on them.

As shown below, the tuning work is tested initially along

TABLE 1: Localization

max_particles	Upper limit of particle number
min_particles	Lower limit of particle number
laser_model_type	The probability model of measurement

the corridor, the particles are distributed along the length of corridor, so the uncertainty is distributed mainly in one dimension. The number of particles should be set properly to make the particle distribution consistent, instead of scattering too much.

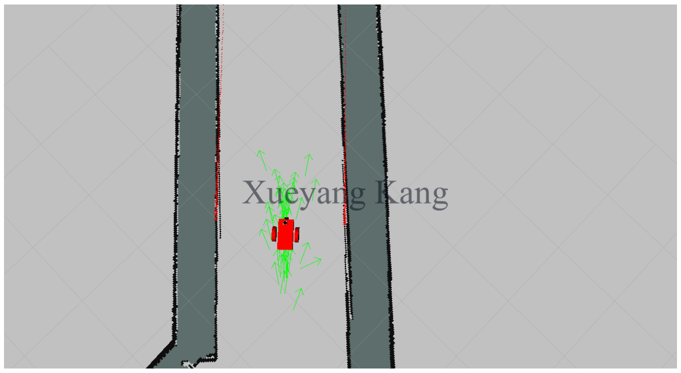

Fig. 3: Particle distribution

3.2 Benchmark Model

3.2.1 Model design

Both robot models use same differential drive mode. The main difference between the two models is the chassis design, here the benchmark model’s chassis is a simple box.

TABLE 2: Benchmark model design vs Personal model design

	Benchmark model	Personal model
Chassis	Box 0.4m×0.2m×0.1m	cylinder1 length 0.1m×radius 0.3m cylinder2 length 0.5m×radius 0.01m
Laser	Top middle	Top front
Camera	Front	Front

3.2.2 Packages Used

”AMCL” receives ”scan” topic from laser and ”map” topic from service ”static_map”, and publishes the ”odom” topic. ”move_base” package subscribes to it, along with ”scan” topic, and then publishes ”cmd_vel” topic to drive the robot model. ”map_server” node will load the settings file for the gazebo world.

3.2.3 Parameters

”AMCL” package contains ”max_particles” and ”min_particles” to determine the accuracy and computational complexity of localization. Here after test 100 and 30 are utilized individually. As for navigation, the parameters below play an important role.

1) ”max_vel_x” and ”min_vel_x” determine the moving speed of the robot, here 2.2 m/s and -0.5m/s are adopted.
2) ”yaw_goal_tolerance” , ”xy_goal_tolerance” set to 0.1 rad, 0.15m, to leave a tolerance for the robot pose at goal.
3) ”pdist_scale” and ”gdist_scale” are the weights on global plan distance error and local plan distance error, set to value 0.8 and 0.3. ”occdist_scale” determines the distance to the occupied objects, set to 0.4 here.
4) ”obstacle_range” 2.5m, ”raytrace_range”, 3.0m.
5) ”robot_radius” corresponding to robot model size, 0.2m used here, and ”inflation_radius” is set to 0.2m.
6) ”update_frequency” and ”publish_frequency” of local costmap configuration is 5.0Hz and 2.5Hz, ”controller_frequency” for publish rate of move base is set to 1.5Hz.
7) As for global cost map size, local cost map is with 6m Height × 6m Width, while global cost map is with 12m Height × 12m Width.

3.3 Personal Model

3.3.1 Model design

As in table 2 the chassis size of my personal model is with three circular plates, the size is denoted as cylinder1, the size of pole denoted as cylinder2 to provide support for the top part.

3.3.2 Packages Used

The previous used packages hold for the personal model.

3.3.3 Parameters

Parameter configuration for ”AMCL” package is unchanged. As for navigation,

1) ”max_vel_x” and ”min_vel_x” determine the moving speed of the robot, here speed upper limit is increased to 3.2m/s to compensate bigger mass of robot model.
2) ”yaw_goal_tolerance” , ”xy_goal_tolerance” are also increased to 0.15 rad, 0.2m, taking into account the robot size.
3) ”pdist_scale” and ”gdist_scale” , weights of global plan distance error and local plan distance error, are changed to 0.6 and 0.4. ”occdist_scale” set to 0.6 here, because the size of robot model is bigger and the agility decreases somewhat, so the local planner distance error should have more weights and collision avoidance also matters.
4) ”robot_radius” is corresponding to robot size 0.3m, and ”inflation_radius” is set to 0.3m accordingly.

4 Results

After the tuning work, the pose particles can converge relatively fast along motion, and the path trajectory generated by the planner can provide a reasonable plan for benchmark model, and the robot follows the plan and moves smoothly, with some exception that sometimes position near the corner at the end of corridor may lead to the ”clear path movement”, the robot hunts for the navigable area. For personal model, the move speed is relatively slow, but the robot can still move smoothly to reach the desired goal, additionally it consumes more time on adjustment of the pose at goal position, which is caused by its big size and mass.

4.1 Localization Results

4.1.1 Benchmark

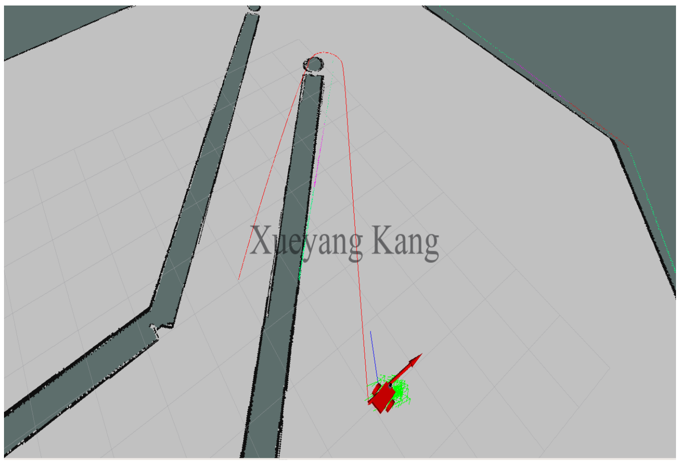

Fig. 4: Benchmark model result

4.1.2 Personal model

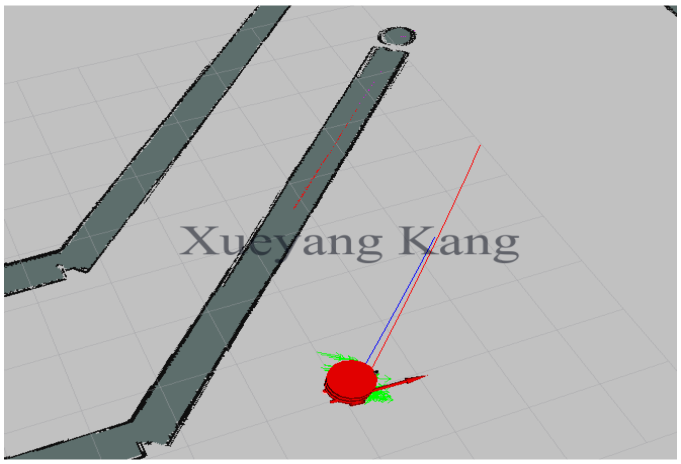

Fig. 5: Personal model result

As the results shown for both models, the particles can orient them with the arrow direction , and they are centered around the goal position. The elapsed time is the total time consumed by the navigation process of robot model from start to the goal.

TABLE 3: Results Comparison

	Benchmark model	Personal model
Particle Converging rate	Fast	Fast
Elapsed time	52.3s	160.2s

4.2 Technical Comparison

Technically speaking, the general motion structure is almost same, but the shapes of the chassis of both models are different, my personal model is similar to some real robot model structure, The parameter settings for both robot model are almost same, except that the distance weight parameters and ”inflation_radius” are different according to model size. The localization performance is almost same for both models, on the other side, the consuming time of movement of my own model is longer but without interruption. As for the benchmark mode, the robot shoot shortly to the destination but with possibility of running into wall. The same applies to their adjustment at goal, the agility feature of smaller one makes it fast to orient itself with goal pose, while the clumsy one needs more adjustment at destination.

5 DISCUSSION

My robot model performs quite well according to the test, the main contribution towards this is the localization part, so that the robot can locate itself quite fast, therefore, the navigation based on it can be realized conveniently. Additionally, the weights of the distance error calculated on cost map was also tuned quite well, a good solution between local and global plan is found. The distance of the inflation radius also assures the robot won’t collide with walls.

5.1 Topics

- According to 3 the time elapsed of navigation for benchmark model is shorter, according this criteria, the default model works better.
- The mass and the size of default model is smaller, so it is agile enough to make some sharp turn, and the overall global trajectory to reach the goal in this case is shorter, it can even make some short-cut.
- The ’Kidnapped Robot’ means the robot may be transported to an unknown environment abruptly or temporary blind, so the localization algorithm must maintain some robust landmark features in a prior map, so that the robot can find the same features and locate itself again.
- Localization algorithm can be used to track the vehicles, drones, or any other moving targets, so that the higher task like navigation or human operation can be better carried on based on the known position of robot.
- MCL/AMCL can be applied onto the mobile robot in warehouse, like some autonomous ground vehicles, to help people manage inventory.

6 CONCLUSION / FUTURE WORK

This project has achieved what I expected, both the robot models can reach its desired goal successfully. And the time consumed by my personal model for the task is a little longer, but the model can move along the trajectory almost without stop.

6.1 Modifications for Improvement

Some improvement work can carried on in the following items.

- The Gazebo Plugin adopted in this project is based on the differential drive mode, so it can not rotate in place, this has increased the complexity of kinematic control, even result in some inevitable error to motion. So some omni drive can be introduced in the future to make the model more flexible.
- The laser for benchmark model can be put higher to assure some wrong measurements like the consideration of wheel as wall never occur.
- Additional laser sensor can be mounted onto the model, but at a same plane with the original one, in oder to have a 360 degrees’ view.

6.2 Hardware Deployment

To deploy the whole navigation stack onto the real hardware, the real laser sensor and odometer driver should be configured properly to publish the corresponding topics, and laser-based map should be constructed prior. Then some base controller setting work should be completed based on real mechanical structure of the robot drive, in order to transform the velocity command to the real speed of the motor properly. The processor performance on which the navigation algorithm runs, along with the memory size matters a lot, so the whole map size and cost map size should not be too big. The resolution of the map, the number of particles are also important, need to be configured based on the processing speed, furthermore, the fine granularity may lead to computational inefficiency.

REFERENCES

[1] Z. Kaiyu, “Latex: Ros navigation tuning guide,” 2017.