Where Am I

Ghee Chong Foo

Abstract—This write up is about creating ROS (Robot Operating System) package such as to develop mobile robot model and simulate under Gazebo environment. The robot models were then integrated with AMCL (Adaptive Monte Carlo Localization) and Navigation stack, in particular move_base ROS package to localize and direct the robot to a goal position in provided map. Parameters corresponding to each package are then added or tuned in order to achieve desired result. A benchmark robot model was first used for base line performance evaluation, and student robot model was then developed for comparison. Both benchmark robot and student robot model were able to reach the goal, with some performance lag for student robot model.

Index Terms—Robotics, Udacity, LaTeX, Localization.

1 Introduction

Localization is an important topic for mobile robots to determine its position based on sensor observations such as camera and range sensor. However noisy sensor input and uncertainty in the environment poses tremendous challenges for the mobile robots. Multiple algorithm has been developed to filter measurement and process noise, including Kalman Filters and Particle Filters. For this project, particle filter was chosen for its ease of implementation and robustness, but with the cost of being more resource intensive.

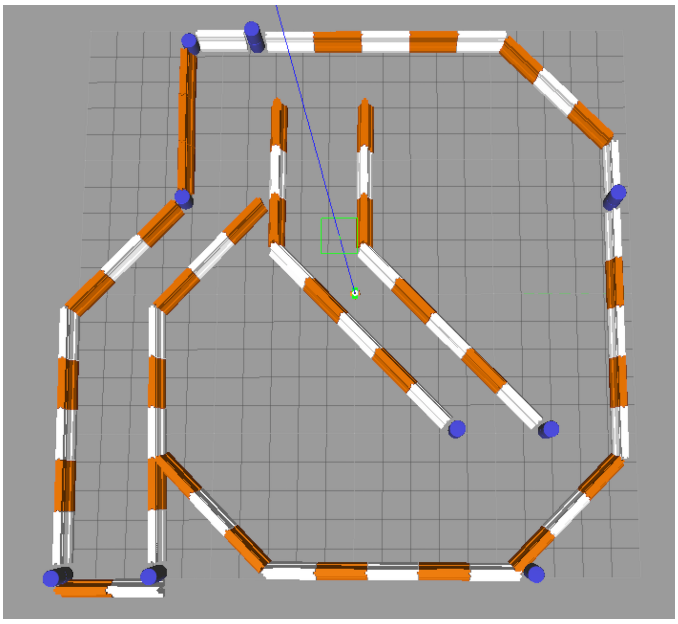

Fig. 1. Jackal Race Map

2 Simulations

Two robots models was developed under ROS environment. Both robots are able to navigate and reach the goal position through the maze within 5 minutes. Result varies depending on running on own machine (MacBook Pro) or Udacity provided workspace, the later has GPU enabled and had much stable performance, and hence all simulation results were based on Udacity workspace.

2.1 Benchmark Model

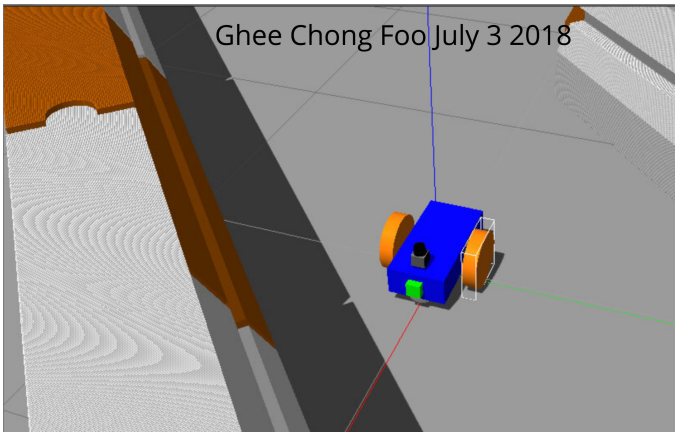

Fig. 2. udacity_bot

2.1.1 Model design

A benchmark model was provided by Udacity, which was a 2-wheel robot on a rectangular chassis with an RGB sensor at the front, and Hokuyo laser range finder mount at the front top of the chassis, as depicted in Fig. 2

2.1.2 Packages Used

The following ROS package were utilized:

1) AMCL
2) move_base

2.1.3 Topics received and published

Below is the list of non-default topics that is received and published based on "rostopic list" output:

- /amcl
- /initialpose

- /joint_states
- /map
- /odom
- /particlecloud
- /tf
- /udacity_bot/camera1
- /udacity_bot/laser/scan

2.1.4 Parameters

Critical parameters that had significant influence on robot model's performance are listed and discussed as below. Parameters definition was refer from [1].

Table 1:

TABLE 1
Costmap Common Parameters

Parameter	Value
obstacle_range	2.0
raytrace_range	3.0
transform_tolerance	0.2
footprint	[[-0.22, -0.20], [-0.22, 0.20], [0.22, 0.20], [0.22, -0.20]]
inflation_radius	0.4

Referring to Table 1, the parameters were tuned to match the size of the robots, also for proper visualization of world map, global costmap and local costmap.

Table 2:

TABLE 2
Base Local Planner Parameters

Parameter	Value
controller_frequency	10.0

Referring to Table 2, controller_frequency was reduced to prevent "Control Loop Missed" error message.

Table 3:

TABLE 3
Global Costmap Parameters

Parameter	Value
update_frequency	1.0
publish_frequency	1.0

Referring to Table 3, update_frequency and publish_frequency were lowered to 1.0 so as not to over-stress the system and churning out "Map update loop" missed error message.

Table 4:

Referring to Table 4, as in Global Costmap, a lower update_frequency and publish_frequency of 1.0 was set so only lower processing power was required. Also width and height were reduced to 5.0 so that robot can be more certain of its local costmap and thus its whereabout.

Table 5:

Referring to Table 5, min and max particles was reduced to meet hardware constraint of the system. odom_alpha value 1 through 4 which define noise level expected from the robot's movements/motions during navigation inside the map, is reduced to 0.002. Initial pose of the robot is at the center of the map, and was setup accordingly.

TABLE 4
Local Costmap Parameters

Parameter	Value
update_frequency	1.0
publish_frequency	1.0
width	5.0
height	5.0

TABLE 5
AMCL Parameters

Parameter	Value
min_particles	50
max_particles	2000
odom_alpha1-4	0.002
initial_pose_x	0.0
initial_pose_y	0.0
initial_pose_a	0.0

2.2 Personal Model

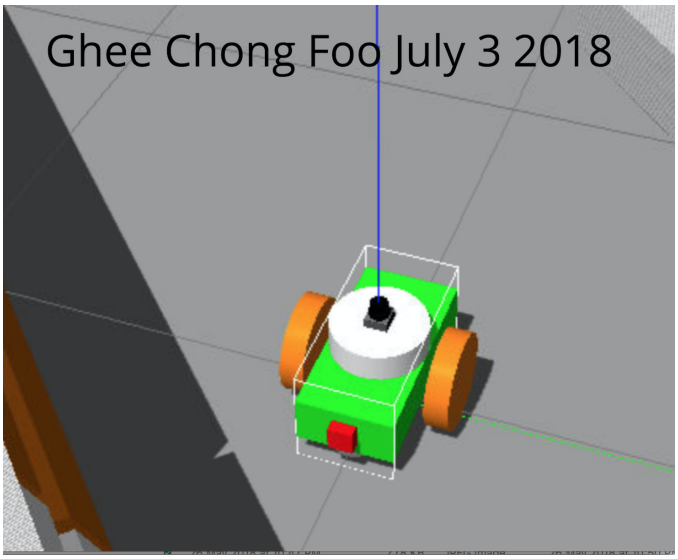

Fig. 3. GC_bot

The personal model is named GC_bot, after the author, as depicted in Figure 3

2.2.1 Model design

GC_bot is based on the benchmark model, with additional cylinder placed on top of the chassis, and moved the Hokuyo sensor on top of the cylinder so that the sensor has a better view of the environment, this is depicted in 3.

2.2.2 Packages Used

Same package was used as in benchmark model.

2.2.3 Parameters

Since similar form factor between benchmark and student model, same parameters was used.

3 Results

3.1 Localization Results

3.1.1 Benchmark

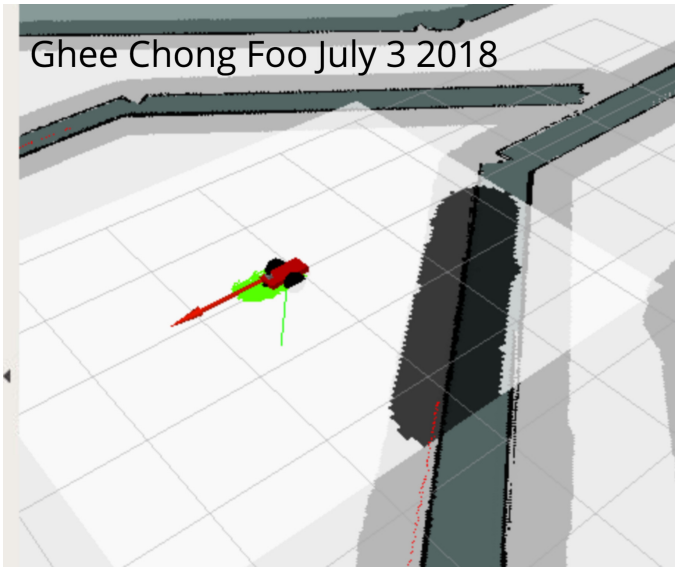

Fig. 4. udacity_bot reached goal Rviz view

Fig. 5. udacity_bot reached goal Gazebo view

The benchmark model, namely udacity_bot, was able to reach the goal within 2 minutes. The particles start to converge around 30 seconds. In some run robot was uncertain of its position especially during turning of the corners, but still able to navigate to the final goal.

Video link showing udacity_bot moving from initial position to goal: https://youtu.be/A2lqlh1FHoc

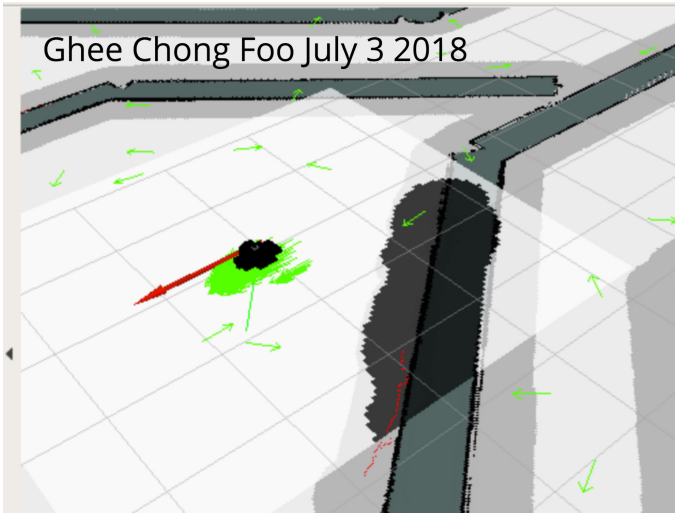

Fig. 6. GC_bot reached goal Rviz view

Fig. 7. GC_bot reached goal Gazebo view

3.1.2 Student

The student model, i.e. GC_bot, was able to reach the goal within 4 minutes, with particles start to converge after 90 seconds. The robot is less certain of its position and making unnecessary turns during navigation. This can be concluded by observing the particles around the robot. The particles was more scattered even after it reached the goal, as depicted in Fig. 6.

Video link showing GC_bot moving from initial position to goal: https://youtu.be/3mdWm8pfrZw

4 Discussion

4.1 Topics

- The benchmark model perform better than student model
- The benchmark model performed better because the sensor range was better tuned

5 Conclusion / Future work

The robot model achieved the project goal of moving from initial position to the goal position and direction as specified. There are wide range of usage for this kind of mobile

robot application, e.g. transporting items from one area to another in factory/hospital/farm, which are often crowded with obstacles, and constant changes in environment. The model can be further deployed in real hardware, e.g. mounting the Nvidia TX2 on a robot with real camera and sensors. As this model was done in simulation mode, more fine tuning will be needed as real-world noise will inevitably creep in and make the robot moving more uncertain, if not fail to reach the goal.

REFERENCES

[1] E. Marder-Eppstein, "Costmaps in ros," 2018. last edited 2018-01-10.