Elegant Racer-bot

Myoungki Jung

Abstract—A localisation of robots is a key requirments task management for autonomous robots. Its implementation in robotics operating system provides a good starter point entering this topic. The purpose of this report is to show the application of off the shelf localisation package inin a simulated condition, robotics operating system(ROS), its implementation, tunning parameters and results.

Index Terms—Robot, IEEEtran, Udacity, AMCL, ROS, Localization, Racer.

1 Introduction

THE localisation is a classic robotics topic with a long history and many variation. Localisation is a task with this question in mind continously. 'Where am I?' In addition, the result of localisation, space or position, is a major domain which compose of a task. Without knowing the position of an actor, the actor cannot perform any designated task in a place. To perform a task delegated to an actor, knowing the location is a essential step and the accuracy of the evaluation also influences the success of the task greatly. To carry out a sucessful localisation, kalman filters and a monte carlo localisation algorithm will be used from the robotics operating system packages, instead of implementing the algorithm from source code. Wihtout positioning the robot to a desired area no other subsuent action can take place in it. Therefore, correct parameters from an adequate tuning process, a high accuracy of pose estimation will be given and this allows other tasks robot performing as intended. However, derieving a set of optimised parameter requires much knowledge on the software package, observation skill, and scientific decisions.

2 Background

2.1 Kalman Filters

Kalman filters are used in evaluation of real value from sensors with noises. Extended Kalman Filter(EKF) is prferred when the filtered data is passed to a monte carlo localisation becuase normal linear kalman filter does not allow nonlinear transformation function, and monte carlo localisation allows computation of non linear functions. Filteration with an extended kalman filter is a crucial action before inserting the sensor data into a localisation process pipe line as the errors in raw sensor values can propagate and amplify its magnitude greater later in localisation processes. Therefore, filteration of noise and evaluation of near true value ensure the overall accuracy of the localisation.

2.2 Particle Filters

Particle filters is a common localisa tion tool Normal monte carlo localisation algorithm is shown in 1. Adaptive (or KLD-sampling) Monte Carlo localization (AMCL)is a probabilistic localization method for a robot being position on two dimensions. [2]. KLD sampling reduced computation load

Algorithm MCL(X_{t-1}, u_t, z_t):
$\bar{X}_t = X_t = \emptyset$
for $m = 1$ to M:
$x_t^{[m]} =$ **motion_update**$(u_t, x_{t-1}^{[m]})$
$w_t^{[m]} =$ **sensor_update**$(z_t, x_t^{[m]})$
$\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
endfor
for $m = 1$ to M:
draw $x_t^{[m]}$ from $\bar{X}_t$ with probability $\propto w_t^{[m]}$
$X_t = X_t + x_t^{[m]}$
endfor
return X_t

Fig. 1. Algorithm of Monte Carlo localization [1]

by probabilistic sampling in a error range. [2] This increases the performance of system performing MCL, allowing low end comuters acheive higher rate of MCL evaluation. Allowance of algorithms to low power machines is a important trend for mobile robots recently.

2.3 Comparison / Contrast

The Table 1 shows the comparision bewteen the extended kalman filter and monte carlo localisation in various features.

Kalman filters and particle filters can make synergy in performance and accuracy. For a mobile robot in 2D, kalman filter can be used in embedded sensors to output less noise sensor readings to the main processing unit, and particle filters in main processing unit can utilise the noiseless sensor values to evaluate the position of the system.

3 Simulations

This section should discuss the performance of robots in simulation. Items to include are the robot model design, packages used, and the parameters chosen for the robot to properly localize itself. The information provided here is critical if anyone would like to replicate your results. After all, the intent of reports such as these are to convey information and build upon ideas so you want to ensure

TABLE 1
EKF vs. MCL

	MCL	EKF
Measurement types	Raw	Landmark
Measurement Noise	Any	Gaussian
Posterior	Particles	Gaussian
Memory Efficiency	Normal	Good
Time Efficiency	Normal	Good
Code difficulty	Easy	Normal
Resolution	Normal	Good
Robustness	Good	No
Memory & resolution	Yes	No
Global localisation	Yes	No
State space	Multimodal discrete	Unimodal continous

others can validate your process. You should have at least two images here: one that shows your standard robot used in the first part of the project, and a second robot that you modified / built that is different from the first robot. Remember to watermark all of your images as well.

3.1 Achievements

You should describe what you achieved for localization in the project with the benchmark model and your own model. Includes charts and graphs show how parameters affect your performance.

3.2 Benchmark Model

3.2.1 Model design

the layout and used sensors are shown in Table 2.

TABLE 2
Model Design

Atribute	Value
Size	0.15 m by 0.10 m
Camera	Generic RGB
LIDAR	Hokyuo

3.2.2 Packages Used

The packages used in the project should be specified as well as the topics received and published; the services it used and provided should also be addressed.

3.2.3 Parameters

Localization parameters in the AMCL node should be described, as well as move_base parameters in the configuration file. You should be able to clearly demonstrate your understanding of the impact of these parameters.

3.3 Personal Model

3.3.1 Model design

The design of the model is the same as the udacity model. This was intended as offline vehicle for this simulation is sparkfun rebot(Figure 2). The realsense camera D435 was intended to use in the model the package in githubs were obsolete and did not work with the project well.

Fig. 2. Sparkfun Redbot

3.3.2 Packages Used

In the simulation the same sensor was used, however, in offline model will have two proximity sensors or a generic sweeping LIDAR instead of hokyuo 1D LIDAR.

3.3.3 Parameters

Listing 1. Fully convolutional network code

```
def fcn_model(inputs, num_classes):

    filter= 32

    return filter
```

- example 1
- example 2

1) example 1
2) example 2

4 Results

Present an unbiased view of your robot's performance and justify your stance with facts. Do the localization results look reasonable? What is the duration for the particle filters to converge? How long does it take for the robot to reach the goal? Does it follow a smooth path to the goal? Does it have unexpected behavior in the process?
For demonstrating your results, it is incredibly useful to have some watermarked charts, tables, and/or graphs for the reader to review. This makes ingesting the information quicker and easier.

4.1 Localization Results

The Figure 3 shows the robot can reach the goal by itselft. The pose array is densely populated right infront of the

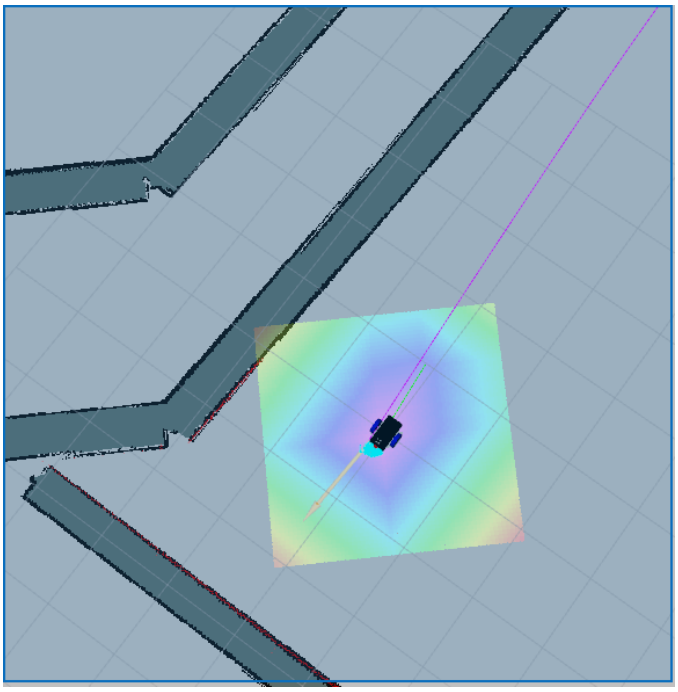

Fig. 3. Arrival to the goal

robot, not on the centre of inertia. However, this does not cause any trouble in navigation and collision avoidance. The orange arrow represents the current goal pose, the violet line is the NavFPath of ROS, a short green line is the local path, and the rainbow box shows the coloured heat map of total cost. The Figure 4 shows that the robot arrived to the In the figure, local cost map is relatively flat because

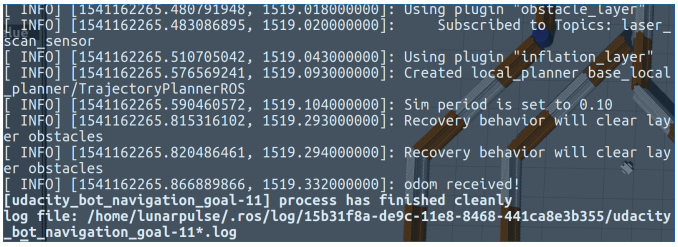

Fig. 4. Arrival to the goal

no obstacles inside the local cost map because The total cost function is dependent on the sensored obstacles and the certainty of mcl localisation and the result cost map becomes indifferent. Figure 5 shows the local cost map with an array of obstacle in the scene, the contour of cost map is more steep and shows the prefered heading of the robot as bright purple, red is the opposite direction. In addition, there is a slight descripency between local path and global navigation path. the robot is following in between two path, and sometimes runs off the route before and after a narrow curve. The Figure 5 shows a recovery of its heading from a U-turn.

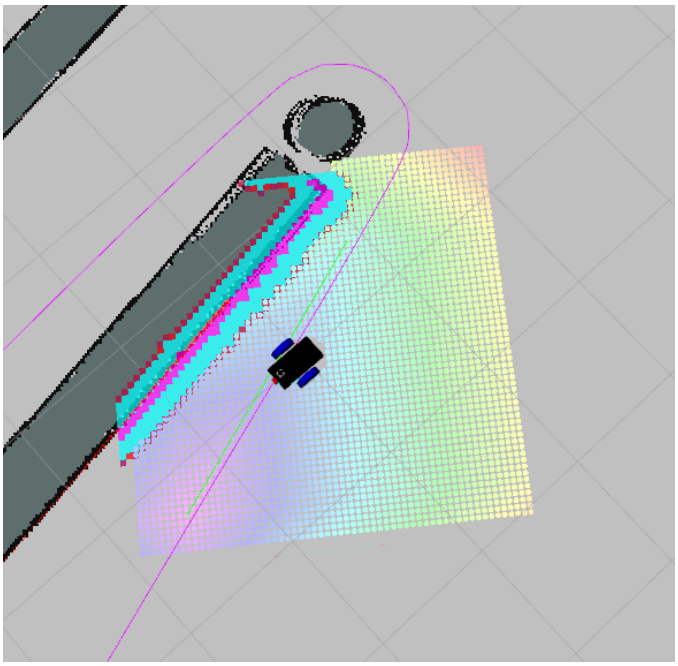

Fig. 5. Local cost map with obstacle existing on one side

4.1.1 Benchmark

4.1.2 Student

4.2 Technical Comparison

Discuss the difference of the layout, parameters, performance etc. between the benchmark robot and your robot. It is acceptable for your custom robot to perform worse than the provided robot. The focus is on learning and understanding, not performance.

5 Discussion

The localisation provided by the AMCL was intuitive and provided a successful localisation with small variance was achieved with the default values for its parameters. It is an easy use of the adaptive monte carlo localisation. The scattered arrows of estimated poses were gathered to the model of the robot in 5 seconds while navigating between the obstacles due to the observed obstacles in a continous manner, if it was in a place without any obstacles in a detectable range of the sensor, it takes a longer time to pin the pose of the robot. The cost cloud map also is related to the existance of local obstacles too.

The local cost map becomes indifference when the robot enters in a area without any obstacle in its sernsoring area. The contour of the cost map become flatter and the navigation stack started to make oscilation becuase the local goal is spreaded and not obvious to take. In some areas robot roams randomly or make arcs instead of following the planned path. This issue was solved by increasing the pDist twice more than gDist(default value 0.8). With pDist (3.5) and and gDist (1.5) robot tends to follow the global goal more than being lost in an uncertain local plan. This also was tweaked by the maximum speed of the robot in later section.

With robot radious 0.0 or not defined, the robot tends to hit to the corner, as the path planner provides very tight

corners which is surely the optimised or the shortest path. However, with out the robot radious parameter tuned robot hits to the pillar and slows down by drawing arcs around it. In addition, top speed of the robot set to 0.5, this was entirely solved and shows a easy cornering without touching the corners.

The resolution of the local cost map was set to 0.05. A higher value, more than 0.1 gave a coarse representation of action and cost anaylysis and rendered the robot roaming indecisively. However, the lower value resulted a flat, less gradient between contours of cost in the grid, and this also affected the behaivour of the robot in a similar manner with the higher value.

5.1 Topics

- Which robot performed better?
- Why it performed better? (opinion)
- How would you approach the 'Kidnapped Robot' problem?
- What types of scenario could localization be performed?
- Where would you use MCL/AMCL in an industry domain?

6 Conclusion / Future work

This section is intended to summarize your report. Your summary should include a recap of the results, did this project achieve what you attempted, how would you deploy it on hardware and how could this project be applied to commercial products?

The set of technology used in this project can apply to many fields operating on the ground with two wheels. Robot vaccum cleaners have the same sets of sensors and actuators except, the vaccum device and continers. This two wheeled mobile platform with accurate localisation can be used as the base vehicle or carrier for many different research too. The callenges in implementation of this platfrom from a simulation to the offline is to select actuators, and its control logic, to program the interface firmware to the control logic and to build and tune according to the hardware specification iteratively.

Porting the simulation to a real robot has challenges. The calculation of inertia of the real physical body of the robot, instead of assumming all axis of inertia is 0.1. This will influence the parameters of costmap as the acceleration and deceleration of the offline robot will be different from the simulation, not only the calculation of the inertia, the noise level from the sensor will vary depending on the operational condition, EKF needs to be taking it accounted.

Overall, processes of tuning consumed majority of time and the amount of effort to understand the dynamics among navigation, localisation, and physics of robot was considerable in developing this even tough refering to the guide. With growing popularity of machine learning, the reinforcement learning is considered to be the very topic which can substitute human tunner's effort with probablistic analysis of parameters and acheive higher performance in a short time. This was shown by alpha go zero, mastering condensed human knowledge of Go for 2000 years practice in a week. With new or continously changing systems, assumming no prior knowledge or short time expiry knowledge, such model free reinforcement leanring should be more approperiate to apply than relying human effort, as human tends to make mistakes in those situation.

6.1 Modifications for Improvement

Examples:

- Base Dimension
- Sensor Location
- Sensor Layout
- Sensor Amount

6.2 Hardware Deployment

1) What would need to be done?
2) Computation time/resource considerations?

References

[1] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, 2002.
[2] D. Fox, "KLD-sampling: Adaptive particle filters and mobile robot localization," *Advances in Neural Information Processing Systems*, 2001.