Where Am I?

Danny Fisher

Abstract—This project implements the Adaptive Monte Carlo Localisation (AMCL) algorithm enabling a robot equipped with LIDAR and an RGB camera to navigate in a Gazebo environment. The project determines the key performance parameters and provides a robust solution for solving a navigation and localisation task. The trial and error selection process for the parameter values highlight the sensitivity of variables through the comparison of two robot designs. Lastly the limitations of this process are discussed

Index Terms—Gazebo, LIDAR

1 Introduction

Localisation and navigation are the two most fundamental problems for mobile robots to solve. [1] By accurately knowing where the robot is located it is able to make decisions to reach a desired location. [2] Within the scope of this paper, two methods of localisation are considered, namely Kalman Filter and Monte Carlo Localisation (MCL). Uncertainty in measurements and the environment are two of the most difficult challenges which robots face when localising. [3] [4] The following subsections provide two methods for solving the localisation problem with imperfect sensors.

1.1 Kalman Filter

The Kalman Filter (KF) aims to predict a state of the robot (for example position in Cartesian coordinates). It does this by having an initial guess at the robot's position which inevitably contains an error. In order to minimise the error, a measurement of the robots position is taken. As each measurement is received, the KF generates better and better guess' of the robot's state. Fig. 1 shows a simple representation of this cycle.

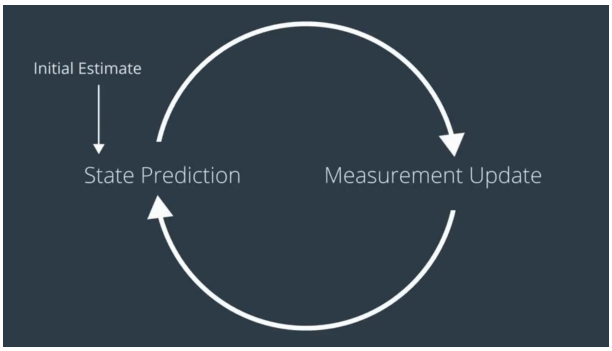

Fig. 1. Kalman Filter State-Measurement Update Cycle

The KF algorithm assumes the measurements are linear which allows the uncertainties to be estimated with a Gaussian (normal) distribution. The beneficial effect of combining uncertain guesses with uncertain measurements is that the next guess (current estimate) becomes more certain. The uncertainty within the measurement affects the current estimate; The more uncertain the measurement, the closer the current estimate is to the previous estimate (Fig. 3). The more certainty and trust the robot has in its measurement, the closer the current estimate is to the measurement. This is shown in the rightly skewed purple distribution in Fig. 2. The accurate estimate of a robot's state comes after multiple measurement- state prediction cycles.

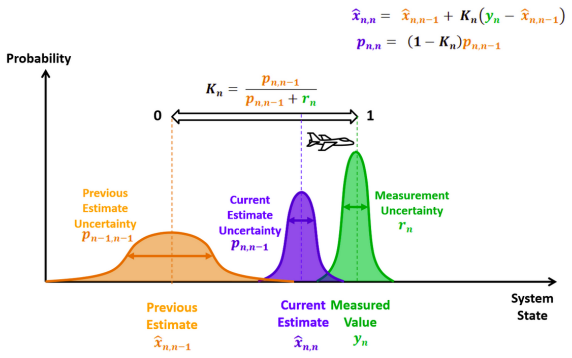

Fig. 2. Kalman Filter High Certainty of Measurement [1]

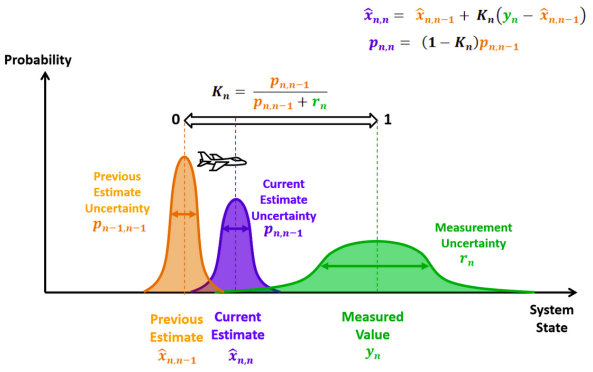

Fig. 3. Kalman Filter Low Certainty of Measurement [1]

There are however limitations to this particular solution. The main drawback of the standard Kalman filter are that

they are only useful in the case of linear systems. Real world systems are non-linear. This makes it necessary to 'Extend' the use of this filter by linearising the variance of the measurements using the Taylor series. [2] In addition the Extended Kalman Filter (EKF) has the property of being very computationally expensive when scaled up to multiple dimensions. [3]

1.2 Particle Filter

A particle filter solves the localisation problem in a very different way. It provides estimates of the robot's actual position by first positing numerous solutions or particles (Fig. 4). Each particle - already assigned a position and orientation - is assigned a weight based on the probability that it is the robot. [4]

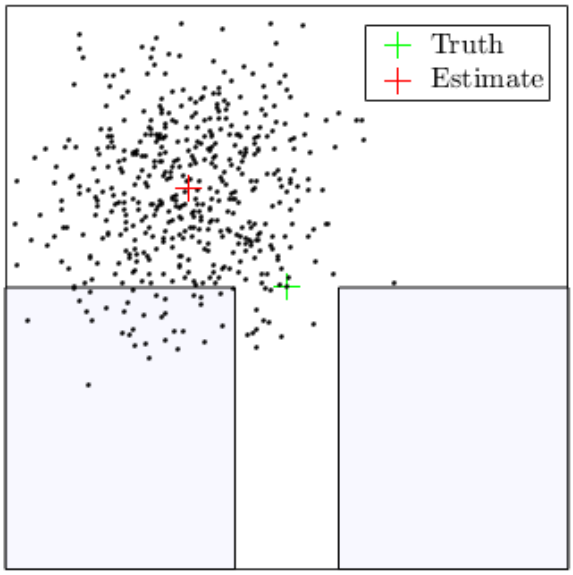

Fig. 4. Particle Filter Before Convergence [5]

The process starts with the particles simulating the motion the robot receives that iteration. Then a sensor measurement takes place with a corresponding probability being given to each particle. The higher the probability of the particle being the robot, the higher the weight.

The second stage of the algorithm involves re-sampling the particles; akin to the process of natural selection. The higher the weight of the particle the higher the probability of surviving the re-sampling process. This allows the particles to converge on the robot's actual position (Fig. 5) thus providing an accurate estimate.

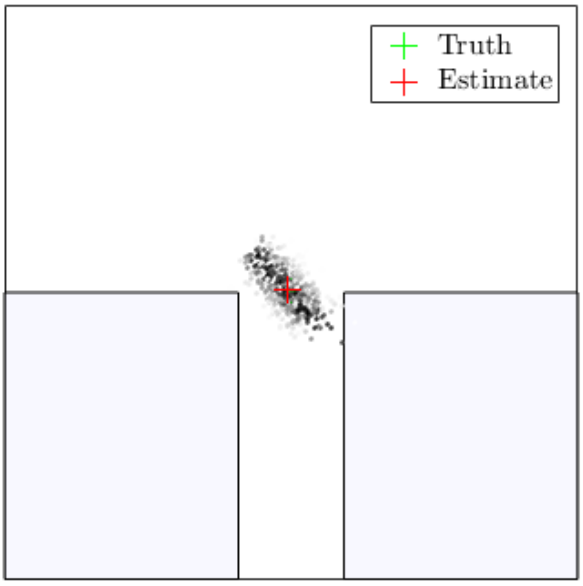

Fig. 5. Particle Filter After Convergence [5]

1.3 Comparison / Contrast

Kalman filters are not useful on non-linear measurement data and the assumption that the measurements are modelled on linear approximations inevitably limit the algorithm whereas particle filters do not have this limiting assumption. In addition the family of MCL algorithms provide scope to allow for a dynamic demand on computation and memory i.e. the particle number can be varied in the case of AMCL and thus reduce the computational burden, whereas EKF cannot. [6]

One of the technical challenges is The Kidnapped Robot Problem (KRP). The challenge is to re-localise in a random location. In its currently described form, the AMCL algorithm would not be able re-localise if it suffered signal loss from sensors, where the Kalman filter would as it is robust enough to converge from any initial guess.

In comparison the EKF and the AMCL both differ in computational effort scalability. The AMCL can adjust the number of particles processed, whereas the EKF will always have a constant processing speed.

For the purposes of localisation in this article, the AMCL algorithm and its implementation is considered.

2 Scene and Package Configuration

The following section details the configuration of the scene and the order it was carried out in:

1) A Gazebo world was generated in a launch file in the udacity_bot package, This loaded the custom world.
2) then a Unified Robot Description Format (URDF) file was completed, detailing the links & joints with their inertial, visual and collision properties.
3) sensors were then added to the URDF file, along with another '.gazebo' file detailing the plugins in order to operate the wheels and sensors.

4) the amcl package and move_base packages were added to a launch file.
5) parameter files were added to the config folder
6) the navigation_goal.cpp file was compiled in the src folder for testing.

3 Robot Configurations

3.1 Model Design

3.1.1 Classroom Robot

As can be seen in Fig. 6 the rectangular body[0.4m by 0.2m by 0.1m] has two cylindrical wheels that enable it to manoeuvre. The green box is an RGB camera capable of displaying what is in front of the robot as well as the black LIDAR sensor on top of the robot which can detect reflected laser light up to 5m away in front of the robot with a peripheral range of 120 degrees. The camera and LIDAR sensor are 0.1m and 0.2m off the ground. The total weight of the robot is 25.2kg.

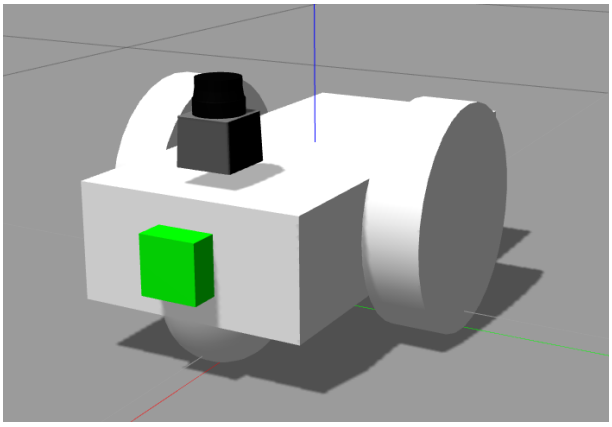

Fig. 6. Classroom Robot in Gazebo

3.1.2 My Robot

Fig. 7 shows the 'My Robot' design which has a circular body, with a laser scanner on top of a stem; 0.25m above the chassis body. The camera for the robot remains in a similar position. The robot weighs 22.2kg and it's footprint and therefore wheelbase is 0.1m larger.

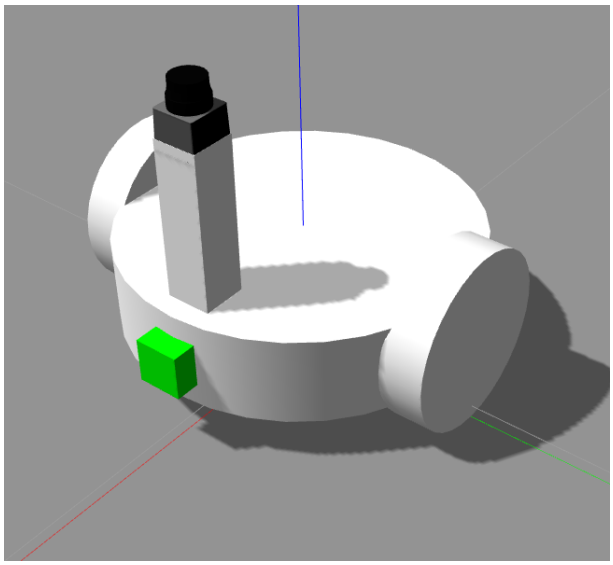

Fig. 7. My Robot in Gazebo

3.2 Packages Used

The main packages used are:

- move_base - this publishes the cmd_vel topic to move the base of the robot whilst clearing the cost map of old values.
- amcl - this package subscribes to laser_scan and tf topics. The tf topic contains the transforms of the robot (from the URDF file). The amcl node publishes the estimate of the robot's pose (amcl_pose) as well as the particle_cloud topic (posearray).
- costmap2d - this package subscribes to the footprint topic to determine the polygon representation of the robots outline. It also publishes the cost map.

3.3 Parameters

The parameters for the aforementioned packages were split into two groups and are elaborated below.

3.3.1 move_base

- transform_tolerance - The allowed latency between each robot transform message. If the delay is larger than this parameter, then an error is thrown.
- update & publish frequency (cost maps) - They both influence how responsive the robot is to the presence of an obstacle, preventing the robot footprint from intersecting with the wall.
- inflation radius - Determines the navigable terrain for the robot. It surrounds area around the obstacles with a high cost value, lessening the further from the obstacle.
- cost maps (resolution, width, height) - Very similar in effect to the update and publishing frequencies, these values affect the 'accuracy' within which the robot can navigate.
- obstacle & raytrace range - Ranges within which the cost map takes effect. It affects the localisation and therefore navigation.

3.3.2 amcl

- max & min particles - This range determines the how many particles are sampled through the AMCL algorithm at any one time. By limiting the maximum, it greatly reduces the computational cost. having a too low limit however reduces the confidence in the localisation measurement.
- update_min_d change - these parameter determine how often the particle filter updates itself when it moves a minimum distance or orientation respectively.
- laser_max_range - this value, when increased to 5m was able to include all particle cloud data points into its calculation of the posearray. Thus localising the robot effectively.
- lazer_z_hit/rand - proportion of laser scans that successfully 'hit' an obstacle and return. The lower the value, the less data points the robot has to localise with and therefore the less precise the posearray.
- noise in odometry (alpha) - reduced the noise of the odometry calculations, which increase the preciseness and accuracy of the pose array.

4 Results

The results section contains the results of running the navigational_goal.cpp node alongside the choice of parameters for both robots.

4.1 Localization Results

TABLE 1
Record of navigation goal attempts for Classroom and My Robot.

Robot	Attempt No.	Success?	ROS Time (s)
Classroom	1	Y	65
Classroom	2	Y	63
Classroom	3	Y	76
Classroom	4	Y	51
Classroom	5	Y	51
My Robot	1	Y	108
My Robot	2	N	N/A
My Robot	3	Y	105
My Robot	4	Y	108
My Robot	5	Y	106

TABLE 2
Summary of navigation goal attempts for Classroom and My Robot.

Robot	Success (%)	Average ROS Time (s)
Classroom	100	61.2
My Robot	80	107

4.2 Parameter Results

TABLE 3
Summary of move_base Parameters

Parameter	Value
holonomic_robot	false
meter_scoring	true
yaw_goal_tolerance	0.10
xy_goal_tolerance	0.10
obstacle_range	2.0
raytrace_range	2.5
transform_tolerance (Classroom)	0.1
transform_tolerance (My Robot)	0.2
inflation_radius	0.2
footprint (Classroom)	0.4m by 0.4m
footprint (My Robot)	0.5m by 0.5m
update_frequency (global & local)	10
publish_frequency (global & local)	10
global costmap width	40.0
local costmap height	40.0
resolution (global & local)	0.05
local costmap width	3.0
local costmap height	3.0

TABLE 4
Summary of amcl Parameters

min/max particles	50/700
update_min_d/a	0.05/0.1
transform_tolerance	0.2
laser_range_min/max	-1.0/5.0
laser_z_hit	0.95
laser_z_rand	0.05
laser_max_beams	50
laser_sigma_hit	0.2
odom_alpha1-4	0.02

5 Discussion

The discussion addresses the questions provided by the rubric.

- Which robot performed better?
  The data in Table 2 conclusively show that the 'Classroom Robot' completed all 5 of its goal attempts compared with 4 out of 5 successful attempts for 'My Robot'. The 'Classroom Robot' was also able to complete the goal attempt with an average of 61s compared with 107s for 'My Robot'.
- Why it performed better?
  When observing all goal attempts there were two main reasons for the longer time measured by the 'My Robot'. One cause was the apparent slower speed of 'My Robot' throughout the whole journey.
  The explanation for this was the transform_tolerance. During the parameter selection process 'My Robot' was not able to process the transform frames (tf) as quickly as the Classroom robot, leading to an acceptable latency of 0.2s (from 0.1s). The acceptable delay highlights that the actual delay between the transforms was larger. If the transforms between each robot frame are slightly delayed, this causes the corrective motion to the wheels to be delayed also. When navigating to a specified path through the

navigational goal node, this would mean any corrective action would come later, thus driving along the fastest path less often.

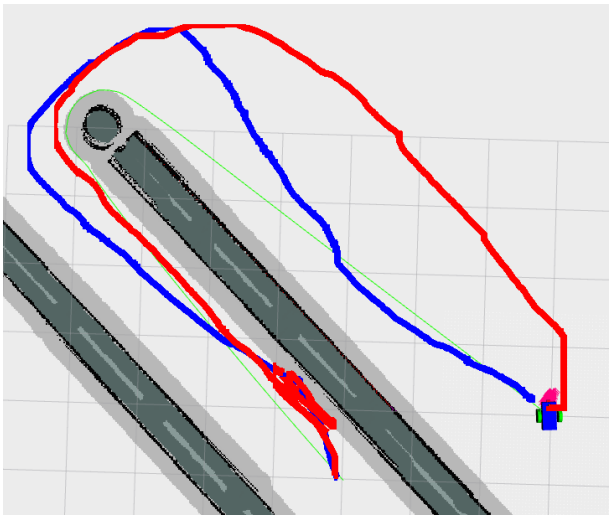

Fig. 8. Routes of Both Robots

Further evidence on the robot's actual path (Fig. 8) helps support this. It shows an artistic impression of the routes taken with the actual screen shots in folders Classroom-Nav-Goal and My-Robot-Nav-Goal. In blue is the Classroom robot, which shows a 'corrective' path on both straights with a tighter corner. However for 'My Robot' it had to reverse as it collided with the wall, costing time. In addition to this the path taken around the corner was much further from the optimal path in green which also would have contributed further to the results that were observed in Table 1.
The other cause was that 'My Robot' appeared to get stuck for longer on the initial part of the course.
One cause of the getting stuck near the barrier is the positioning of the laser sensor. With the laser sensor being placed higher (10cm more) there is a unique part of the barriers within the map which are indented further where the laser scanner happens to be placed. As both robots have identical parameters for object detection, this could be the difference between one robot getting too close to the cost map values near barrier and the other narrowly missing it. Another explanation for 'My Robot' getting 'stuck' is that the robot is not able to localise accurately at the beginning of the goal attempt. By not knowing precisely where the robot is due to longer processing times 'My Robot' the corrective actions for the robot aren't as effective, causing the robot to accidentally enter the cost map barrier and require it to stop.

- The tradeoff's in accuracy and processing time.
  The tradeoff's are normally found when an error occurs. Typically the tradeoff's between accuracy and processing time come in two forms:
  costmap - The cost map provides a pivot for a tradeoff. The cost map provides multiple data points for every laser scanned point. Thus increase in the cost map size greatly impacts the number of processes. These can be things such as increasing resolution and map dimensions, increasing the update and publishing of these cost maps. Lastly obstacle range and inflation radius both increase the number of data points which the cost map has to process. All of these factors bear a large weight on the processing time and accuracy.
  amcl - In the amcl package, the number of particles considered has the largest impact on localisation accuracy and therefore processing time. The frequency with which these are processed is also significant, grounded in the update_min_d and update_min_a values. The smaller these values, typically the more frequent the localisation cycles, increasing the processing time and also accuracy.
- How would you approach the 'Kidnapped Robot' problem?
  The Kidnapped Robot Problem (KRP) details the scenario whereby a robot is picked up from a known location (i.e. where it has localised) and dropped into an unknown location. The particle filter (AMCL) algorithm in its current form would not be able to solve the locallisation problem of readjusting to the map. When the robot initially localises itself, the particles or guesses of where the robot is are spread out randomly in the environment. Through a process of resampling the particles which are most likely to be where the robot is converge until all the particles are where the robot is located. Once the robot is moved to another location, the particles are so precisely placed, that the resampling process is not able to converge again. The solution to this is to set up the initial condition of placing the particles randomly throughout the environment again when the KRP has occured and the robot is placed on the ground again. Solutions are available including the Sensor Resetting Localization (SRL) and Received SignalStrength (RSS) which uses wifi signals to relocalise. [7] [8].
- What types of scenario could localization be performed?
  AMCL would be beneficial in environments where there isn't much change in the environment and the terrain is level as the localisation is heavily dependant on stable features in the environment. Fig. 9 shows how the global costmap fails to accurately determine the robot's location when a random object is placed into the Gazebo environment.
- Where would you use MCL/AMCL in an industry domain?
  Therefore the localisation algorithms which are present in this report would be most useful in a factory item retrieval robot or a mobile robot that is used in an indoor environment.

5.1 Modifications for Improvement on 'My Robot'

There are a few suggestions for how 'My Robot' would benefit from modification to its design:

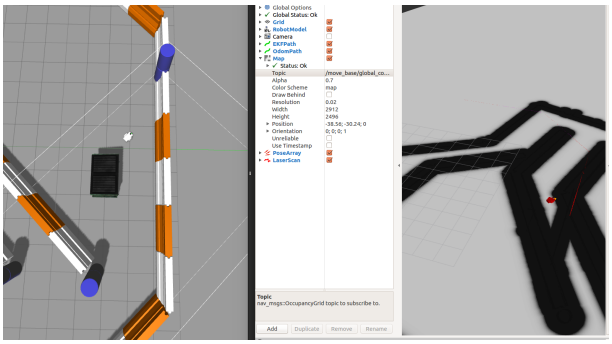

Fig. 9. Unknown Object In Environment

- weight: reduce weight of robot; this would increase the sensitivity of torque input from the wheels to the angle of rotation of the chassis
- increase the maximum torque in the plugin files; again this would increase the sensitivity of torque input to rotational angle
- Base Dimension: reduce wheelbase; this would reduce rotational inertia in the robot
- Sensor Location: move laser sensor onto chassis; by reducing the height of the camera, this would remove any ambivalence on what gets the robot stuck.
- Sensor Layout: keep sensors at front; assumed ideal position.
- Sensor Amount: keep the same; classroom robot works sufficiently.

6 Conclusion

This report covered the implementation of a localisation problem in the ROS framework. By optimising parameters in the ROS packages: amcl, costmap2d and move_base, the adaptive monte carlo localisation method was successfully implemented in two robots. The robot which was most successful (classroom robot) has parameters that suited its design best. The exact reason why it performed better was put down to the layout of the laser sensor and the chassis design of the 'My Robot'. The laser sensor placement highlighted a weakness in robustness of the parameter choice where it appeared to take multiple readings. The mechanical factors of 'My Robot' also meant that the driving behaviour is sub optimal causing the robot to follow a far less optimal path.

Out of the 5 tests the Classroom Robot had a success rate of 100% with an average time of 61s. 'My Robot' has a success rate of 80% with an average time of 107s.

7 Future Work

The work to be carried out would be to implement the changes in the modifications section.

References

[1] Alex Becker, "Kalman filter in one dimension." https://www.kalmanfilter.net/kalman1d.html , 2018. Accessed: 20/08/19.

[2] R. A.UmaMageswari, J.Joseph Ignatious, "A Comparitive Study Of Kalman Filter, Extended Kalman Filter And Unscented Kalman Filter For Harmonic Analysis Of The Non-Stationary Signals," *International Journal of Scientific Engineering Research*, vol. 3, July 2012.

[3] "Taylor series." https://en.wikipedia.org/wiki/Taylor$_series$, 2019. $Accessed$: 20/08/19.

[4] D. Sabinasz, "Robot localization iv: The particle filter." http://www.deepideas.net/robot-localization-particle-filter/, 2017. Accessed: 20/08/19.

[5] P. Owan, "Limited access manufacturing." https://depts.washington.edu/barc/projects/limited-access-manufacturing, 2019. Accessed: 22/08/19.

[6] *Localization and Navigation*, pp. 241–269. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[7] S. Lenser and M. Veloso, "Sensor resetting localization for poorly-modelled mobile robots," in *Robotics and Automation (ICRA)*, vol. 2, (2000IEEE International Conference on), pp. 1225–1232, IEEE, 2000.

[8] D. H. B. Ferris and D. Fox, "Gaussian processes for signal strength-based location estimation," *Robotics Science and Systems*, 2006.