Robotics Nanodegree Localization Project

Sebastian Castro — sebas.a.castro@gmail.com

Abstract—This paper outlines the design and implementation of the Adaptive Monte Carlo Localization (AMCL) algorithm. This is tested using two simulated mobile robots – a benchmark robot and a personal robot – navigating a known mapped environment from a start to a goal position. Some additional topics are discussed, such as simulated robot design, parameter choices and tradeoffs, and application and deployment of AMCL in the real world.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Localization.

1 Introduction

The introduction should provide some material regarding the history of the problem, why it is important and what is intended to be achieved. If there exists any previous attempts to solve this problem, this is a great place to note these while conveying the differences in your approach (if any). The intent is to provide enough information for the reader to understand why this problem is interesting and setting up the conversation for the solution you have provided Use this space to introduce your localization task and how you wish to accomplish it; save the details about the robot construction for later (simulation is a good point for this information). If you have any papers / sites / repositories you have referenced for your robot, please make sure to cite them.

References

2 Background

At this stage, you should begin diving into the technical details of your approach by explaining to the reader what are the characteristics of the filters, what localization method was chosen, and the reason that it was selected (i.e. particle filters). This should be factual and authoritative, meaning you should not use language such as "I think this will work" or "Maybe Monte Carlo Localization with these parameters is better...". Instead, focus on items similar to, "Adaptive Monte Carlo Localization was chosen because..." Provide a sufficient background into the scope of the problem technologically while also identifying some of the current challenges in robot localization and why the problem domain is an important piece of robotics.

2.1 Kalman Filters

Briefly describe Kalman filters. Explain how they work and why they are used for localization. Additionally, discuss the drawbacks of linear Kalman filters and how Extended Kalman Filters (EKFs) help resolve some of these issues.

[1] [2] [3]

2.2 Particle Filters

Briefly explain what a particle filter is, how it is used, and why it is useful.

[4] [5]

2.3 Comparison / Contrast

Explain the benefits and disadvantages of using a Kalman Filter / Particle Filter. Why would you use one over the other? Also inform the reader that the work presented here will be using only particle filters.

3 Approach

3.1 Model Design

To test localization and navigation capabilities, two robot simulation models were created – a benchmark robot and a personal robot. Both models were built using URDF files and plugins (camera, lidar, odometry, base controllers) for the Gazebo simulator.

3.1.1 Benchmark Model

The benchmark robot is a differential drive robot with a front-facing camera and lidar. Figure 1 shows the robot in RViz at its initial position and Table 1 lists some key robot parameters.

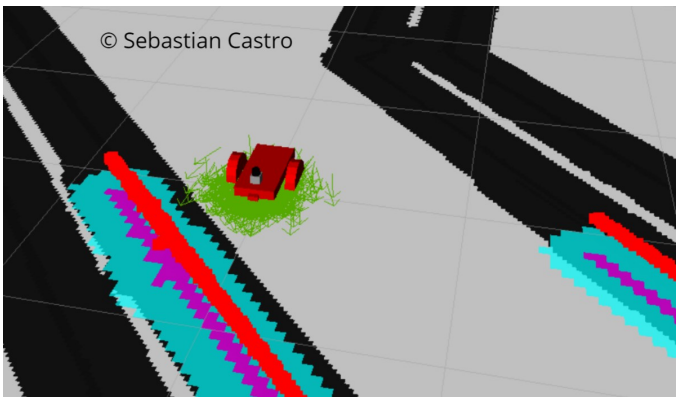

Fig. 1. Benchmark model at initial position

3.1.2 Personal Model

The personal model was developed as an extension to the benchmark model. The main difference is the redesign of the base to support a full 360 degree lidar scanner view. The list of changes is below:

TABLE 1
Benchmark Model Parameters

Total Mass	25.2 kg
Wheel radius	10 cm
Wheelbase	40 cm
Camera location [XYZ]	[0.2 0 0] m
Camera field of view	80 deg
Lidar location [XYZ]	[0.5 0 0.1] m
Lidar range	0.2 - 30 m
Lidar angles	720 samples, -90 to +90 deg

TABLE 2
Personal Model Parameters

Total Mass	25.2 kg
Wheel radius	75 cm
Wheelbase	45 cm
Camera location [XYZ]	[0.1 0 0.05] m
Camera field of view	75 deg
Lidar location [XYZ]	[0 0 0.15] m
Lidar range	0.2 - 20 m
Lidar angles	720 samples, -180 to +180 deg

- Lidar angular range doubled to 360 degrees, but with the same number of scans, so angular resolution was halved.
- Wheel radius decreased by 25%, wheels lowered, and lidar raised to prevent lidar collisions
- Lidar was moved to be directly above the center of the robot chassis to make coordinate transformations simpler
- Camera was moved to the top of the robot chassis instead of in front of the robot with safety against frontal collisions in mind. Since the camera was moved further towards the center of the robot, field of view was decreased by 5 degrees.
- Cosmetic changes to URDF materials: body was colored black, wheels gray, and camera red

Figure 2 shows the personal robot in RViz at its initial position and Table 2 lists some key robot parameters.

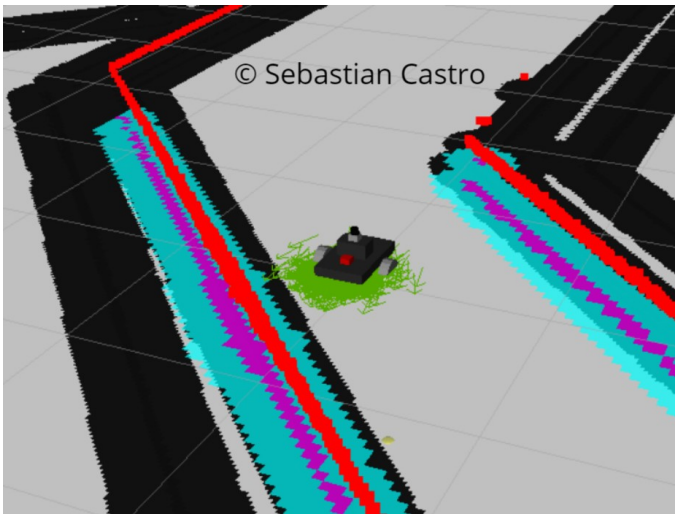

Fig. 2. Personal model at initial position

3.2 Packages Used

The package used for AMCL and navigation of both robots is named `udacity_bot`, which can be downloaded from this GitHub repository.

For both robots, the list of ROS nodes and topic names can be found in Table 3 and in the `rqt_graph` snapshot in Figure 3.

TABLE 3
ROS Topic Names

Odometry	`\odom`
Laser scan	`\udacity_bot\laser\scan`
Velocity command	`\cmd_vel`

3.2.1 *Benchmark Model*

Bringing up the experiment for the benchmark robot requires the following commands:

- `roslaunch udacity_bot udacity_world.launch`
- `roslaunch udacity_bot amcl.launch`
- `rosrun udacity_bot navigation_goal`

3.2.2 *Personal Model*

Bringing up the experiment for the personal robot requires the following commands:

- `roslaunch udacity_bot udacity_world_sebastian.launch`
- `roslaunch udacity_bot amcl_sebastian.launch`
- `rosrun udacity_bot navigation_goal`

3.3 Parameters

The navigation parameters for AMCL and mobile base control were selected with the intent of being general enough to localize and control both benchmark and personal robots so they effectively reach the goal position within the provided map. A few parameter values were different for both robots, which will be discussed below.

The AMCL parameters were chosen as shown in Table 4.

- The range of particle numbers was reduced from the default 100-5000 to 100-1000 to reduce computation and memory usage while still localizing the robot in the known map.

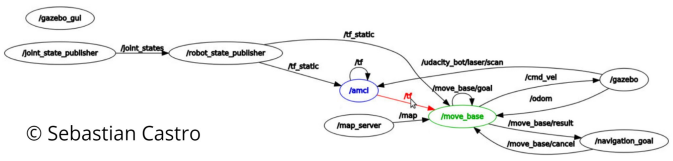

Fig. 3. ROS graph of simulated robot, AMCL, `move_base`, and `navigation_goal` nodes

- Initial X and Y pose covariance was reduced significantly to $0.1 * 0.1 = 0.01m^2$ since quick convergence was useful to more reliably navigate around the wall directly in front of the robot starting position, while the estimated pose is still converging. However, localization results still converges with larger initial covariance.
- Update distances – both linear and angular – were reduced from their default values so the filter updated more often to avoid changes in localization results.
- Lidar range and hit standard deviation were chosen to match the simulated lidar parameters, and therefore differ between the benchmark and personal robots.

TABLE 4
AMCL Parameters

Particle numbers	100 - 1000
Update minimum distance	5 cm
Update minimum angle	0.2 rad
Initial std. dev. [X, Y, theta]	[0.1 0.1 $\frac{\pi}{12}$]
Laser range (benchmark)	0.2 - 30 m
Laser range (personal)	0.2 - 20 m
Lidar location [XYZ]	[0 0 0.15] m
Lidar range	0.2 - 20 m
Max laser beams	720
Laser hit std. dev.	0.1

The mobile base control parameters were chosen as shown in Table 5.

- Different inflation radius was used for both robots. Specifically, a smaller radius was chosen for the personal robot because of its wider wheelbase requiring more free space for the robot to navigate, and because the lidar sensor was farther towards the back of the robot.
- Obstacle range was chosen to be 5 meters in accordance with the local costmap size of 5 meters. This was sufficient for the robot to navigate obstacles without using too large a local costmap.
- Both local and global costmaps used a resolution of 0.05 meters (5 cm)
- To ensure the algorithms ran on the development machine without significant timing issues, the map was updated at 5 Hz, the base was controlled at 10 Hz, and a transform tolerance of 0.2 was selected.

TABLE 5
Move_Base Parameters

Inflation radius (benchmark)	0.25 m
Inflation radius (personal)	0.2 m
Obstacle range	5 m
Ray trace range	10 m
Costmap resolution	0.05 m
Global costmap size	40 m
Local costmap size	5 m
Costmap frequency	5 Hz
Controller frequency	10 Hz
Transform tolerance	0.2

4 RESULTS

With the design choices described in the previous section, both robots were able to successfully localize and navigate from the starting position to the goal specified in the `navigation_goal` program.

4.1 Localization Results

4.1.1 Benchmark Robot

Figure 4 shows the in-progress and final navigation results for the benchmark robot. As can be seen in the figure, the AMCL particles converge around the true robot pose. In fact, they remain close to the robot but diverge again when the robot changes orientation around the goal position.

Figure 5 shows the time history of the ground truth and localized X and Y positions of the benchmark robot, and that they are consistently close to each other during the navigation task. Navigation for the benchmark robot took approximately 47 seconds.

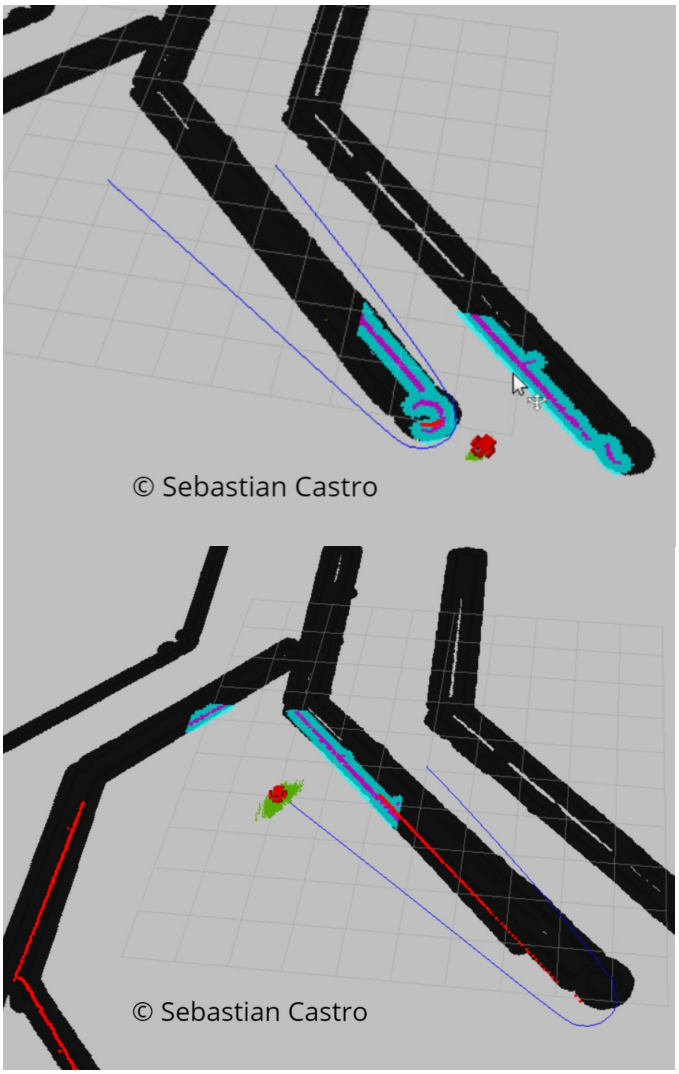

Fig. 4. Benchmark robot during navigation [top] and at the goal [bottom]

4.1.2 Personal Robot

Figure 6 shows the in-progress and final navigation results for the personal robot. As can be seen in the figure, the

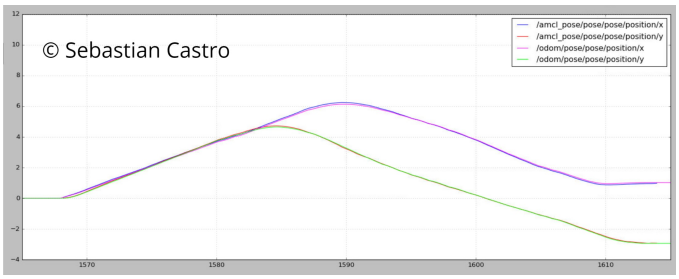

Fig. 5. Benchmark robot localization plot

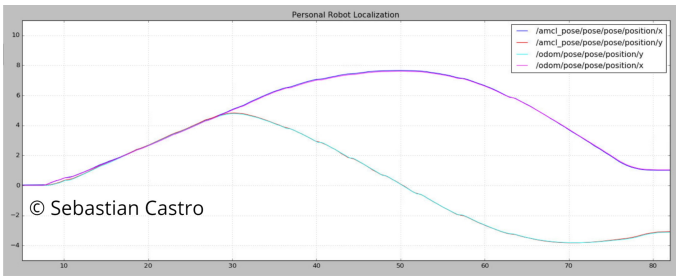

Fig. 7. Personal robot localization plot

AMCL particles converge around the true robot pose. In fact, they remain close to the robot but diverge again when the robot changes orientation around the goal position.

Figure 7 shows the time history of the ground truth and localized X and Y positions of the custom robot, and that they are consistently close to each other during the navigation task. Navigation for the benchmark robot took approximately 75 seconds.

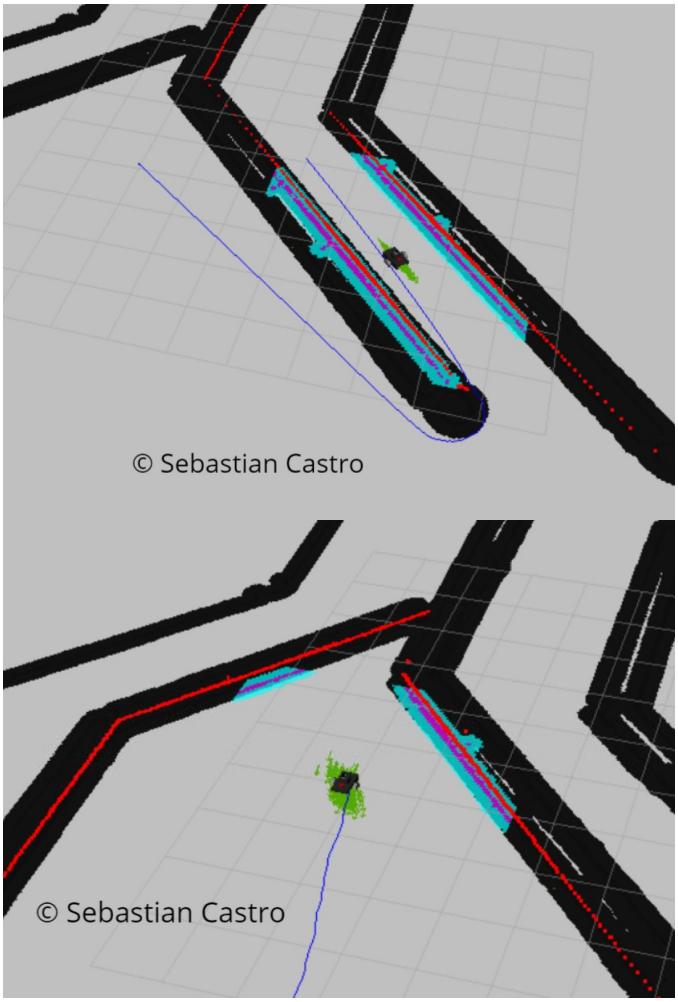

Fig. 6. Personal robot during navigation [top] and at the goal [bottom]

4.2 Technical Comparison

Although both robots were able to successfully localize in the environment using AMCL, the benchmark robot completed the navigation task approximately 63% faster than the personal robot (47 vs. 75 seconds).

Based on the results, this was primarily due to the personal robot path tracing a significantly wider arc towards the goal after clearing the initial corridor in the map. This can be seen by comparing the plots in Figures 4 and 6, or by seeing that the path leading the goal position in Figure 4 is much closer to the wall than the one in Figure 6.

5 Discussion

5.1 Performance Comparison

In terms of computational load, both robots should have been equal. While the personal robot has a 360 degree lidar scan view, the angular resolution was halved, so both robots have the same number of total beams. Also, all the localization and navigation parameters (number of particles, update thresholds, frequencies, and costmap size/resolution) were selected to the be same. Therefore, this should not be a factor in justifying the differences in performance.

It is believed that the robot layout contributed significantly to the discrepancy in performance. These are similar reasons for why the map inflation radius had to be changed for the personal robot.

Mechanical parameters – The personal robot has a larger wheelbase and smaller wheel radius, which made it less responsive to sharper turns. This caused obstacle avoidance in the initial corridor to have a few more oscillations, but also is believed to be a main factor for why the path to the goal after the initial corridor was much wider and continuously needed to be adjusted.

Sensor positioning and controller tuning – The lidar sensor in the benchmark robot was located near the front of the robot, which helped with clearing the wall near, and directly in front of the robot, right around the starting position. More importantly, the `navigation_goal` program was likely tuned for the benchmark robot configuration – that is, with obstacle avoidance thresholds that assumed a lidar near the front of the chassis, and with a control strategy that mostly drove the robot forward unless an obstacle was directly in front of the robot.

With this largely forward-driving controller, having a full 360 degree lidar range is not necessarily an advantage, though having half the angular resolution and a sensor farther from the front of the chassis seemed to be a disadvantage for obstacle avoidance and navigation.

5.2 Other Topics

- How would you approach the 'Kidnapped Robot' problem? – covariance blow-up
- What types of scenario could localization be performed? – known maps, constrained environment
- Where would you use MCL/AMCL in an industry domain? – in industrial settings, home/workplace service robots, other things like museums/tours

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? For Future Work, address areas of work that you may not have addressed in your report as possible next steps. This could be due to time constraints, lack of currently developed methods / technology, and areas of application outside of your current implementation. Again, avoid the use of the first-person.

6.1 Modifications for Improvement

Examples:

- 4 wheel or omnidirectional drive, though odom is challenging
- 360 lidar for localization, high res front facing lidar for obstacle avoidance
- retuning controller for obstacle avoidance with a more back-facing lidar

6.2 Hardware Deployment

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

References

[1]
[2] S. Julier and J. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defense Sensing, Simulations and Controls*, 1997.
[3] E. A. Wan and R. V. D. Merwe, "The unscented kalman filter for nonlinear estimation," pp. 153–158, 2000.
[4] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Monte carlo localization for mobile robots," in *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, 1999.
[5] D. Fox, "Kld-sampling: Adaptive particle filters and mobile robot localization," in *In Advances in Neural Information Processing Systems (NIPS*, 2001.