Where am I?

Jose A Sanchez De Lucio

Abstract—Robot navigation is one of the most important topics in the field of robotics. Its importance comes from the importance of providing the ability to a robot to navigate in different environments. In this small article is presented a basic implementation of a robot navigation algorithm using ROS. The objective is to implement a basic robot, represented by a small two wheel vehicle, with the ability of navigate through a specific environment, having as an objective the reach of a specific location, within the environment. For this, an Adaptive Monte Carlo Localization algorithm was used. As obtained results, the implemented robot was able to reach the desired goal.

Index Terms—Monte Carlo Localization, AMCL, Robot Navigation, ROS.

1 Introduction

ROBOT navigation is a problem that can be divided in many different sub problems. For example, the use of sensors in the robot to interact with the environment and the use of Kalman filters or Monte Carlo algorithms to find the position of the robot. Furthermore, the definition of a global and local positions to estimate its position with respect to a previously learned map or keeping the track of the robots position over time, respectively [1].

The global localization problem involves a more difficult problem, where the robot does not know its initial position, meaning that the robot needs to estimate its position from scratch. On the other hand, in the local problem the robot knows its initial position and only needs to calibrate small errors around its odometry while moving within the environment [2]. Odometry is the use of data from motion sensors to estimate the change in position of a robot over time.

The most used algorithm to localize the position of a robot are the Kalman Filters and the Monte Carlo Localization algorithms. Depending of the problem each of them could give better results than the other [1]. In this work is used the Adaptive Monte Carlo Localization (AMCL) [3] to find the position of the robot.

2 Background

The Kalman filter is an estimation algorithm that is very prominent in control. It is used to estimate the value of variables in real time, where in some application the estimated variable is the position or the velocity of a robot. Therefore, Kalman filters can be used for robot navigation, or localization. Kalman filters have the property that they can give very accurate estimation of real values even in very noisy environments.

The Monte Carlo Localization (MCL) algorithm is also known as the bootstrap filter [4], the Monte-Carlo filter [5], the Condensation algorithm [6] or or the survival of the fittest algorithm [7]. All these methods are known as particle filters [8]. Particle filters are use to solve the global or local localization problems by randomly and uniformly distribute particles around the whole environment, particles that do not exist physically.

The particles are used to represent hypothesis of where the robot may be located. Furthermore, each particle has assigned a specific weight, defined as the different between the robot actual pose and the particle's predicted pose. Therefore, the importance of the particles depend on their weights, and then, the bigger the particle, or its weight, the more accurate the particle is. Then, after the re sampling process, some particles are discarded, where particles with larger weights are more likely to survive, given the process used to select the remaining particles.

The re sampling process discard some particles based on the process described in the Udacity Robotics Course. This process is used to find the particles that converge and can be used to estimate the robots true position. In addition, the Adaptive Monte Carlo Localization algorithm dynamically adjust the number of particles over a period of time, as the robot navigates around its environment, or map. As result, this adaptive process offers a significant computational advantage over MCL.

Nonetheless, in order to implement any of the previously mentioned algorithms, it was necessary to define parameters that can be used to adjust the performance of the algorithm, as mentioned in more detail in the next section.

3 Model Configuration

First, it was necessary to simulate the robot and the environment where it was tested, or localized. The robot was simulated as a two wheels robot, having a camera to get information from its environment and a hokuyo laser range finder, as presented in figure 1.

In the previous figure, the camara is represented by a red square, while the hokuyo laser is located at the top of the two wheels robot chassis. The chassis of the robot is represented in blue ahd the wheels in green. Additionally, the integration of Gazebo and Rviz for the simulated model is presented in figure 2.

The parameters used can be observed in table 1. The size of the robot and the location of the sensors parameters were set as proposed in the project instructions, given how they are represented in the map proposed for the problem. Additionally, satisfactory results were obtained using these parameters, as can be observed in the following section.

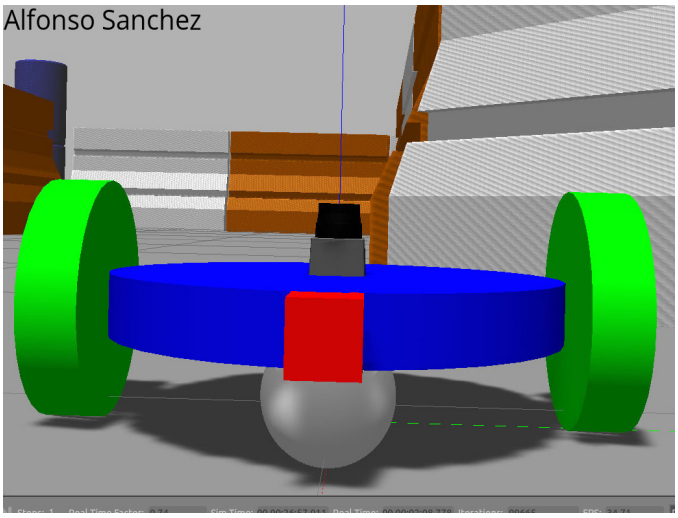


Fig. 1: Gazebo modified robot model.

Fig. 2: Rviz modified robot model.

3.0.1 *Localization*

The launch files were modified from the provided ones in order to include the required maps to test the robot navigation. To be more specific, the jackal map, the amcl and the moveb package files were considered. In figures and 3 are presented the provided map and the particles used for the amcl algorithm to find the location of the robot with respect to the map, in Gazebo and Rviz, respectively.

4 Results

Results can be observed in figure 4. In this figure are presented how two different robots navigate through a specific map using the AMCL algorithm, where the green arrows represents the AMCL particles and the red arrow the goal location. In 4a and 4b are presented the robot navigation in gazebo and Rviz for a robot model provided from Udacity as an example for the problem, while in figures 4c and 4d are presented, in gazebo and Rviz, the navigation of the map for a robot model generated specifically to solve this problem, which can be observed better in figure 1.

TABLE 1: Model parameters

Parameter	Value
chassis geometry	cylinder
chassis radius	0.18
chassis length	0.15
left_wheel origin	[0 .2 0]
right_wheel origin	[0 -.2 0]
min_particles	100
max_particles	800
laser_model_type	likelihood_field
likelihood_field_max_dist	2.0
initial_pose_x	0.0
initial_pose_y	0.0
initial_pose_a	0.0
odom_model_type	diff_corredted
odom_alpha1	0.02
odom_alpha2	0.02
odom_alpha3	0.02
odom_alpha4	0.02
obstacle_range	0.3
raytrace_range	0.3
transform_tolerance	1.0
robot_radius	0.42
inflation_radius	1.0
update_frequency	20.0
publish_frequency	20.0
local width	10.0
global heigth	10.0
local width	5.0
local heigth	5.0

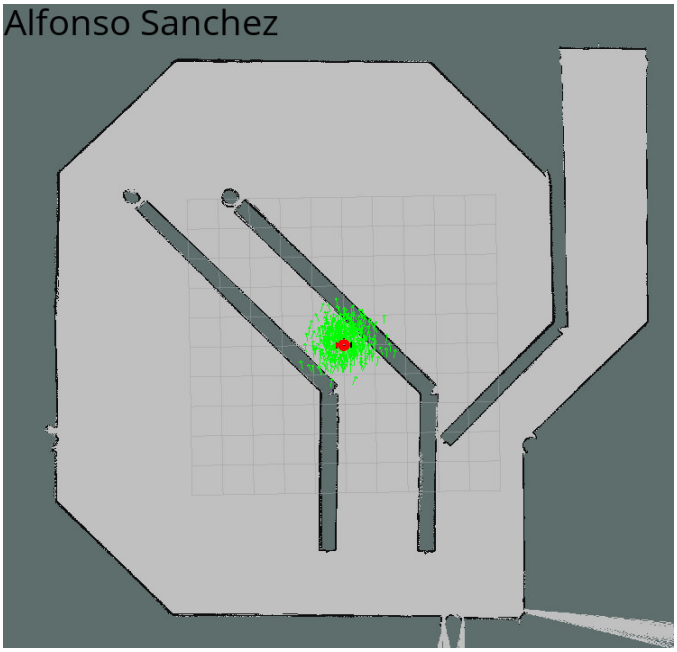
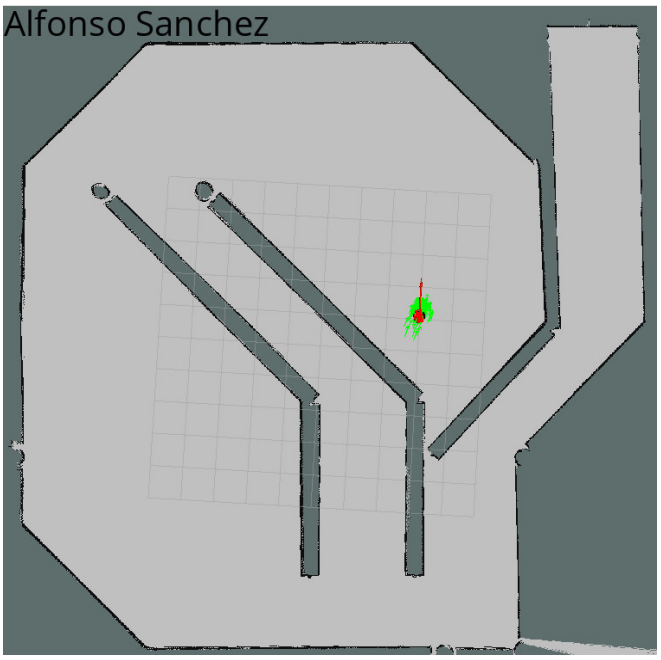


Fig. 3: Initial position for the robot localization problem.

The different between results obtained from the Udacity robot model and the generated for the solution of this problem are not only related with the shape of the robot, but which different parameters which were tunned in order for the robot to have a better idea of its environment and to constraint their movements.

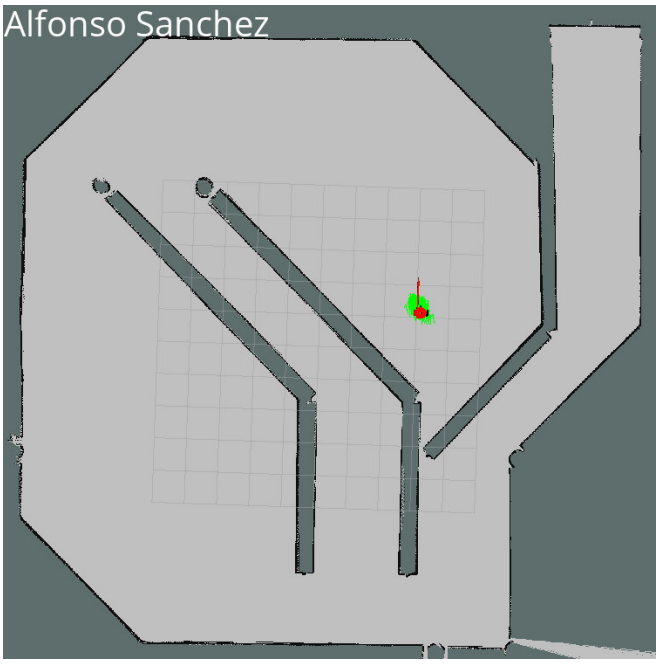
Example of the previously mentioned parameters can be observed better in table 1. The number of particles helps to decrease uncertainties about the robot location, then,

(a) Gazebo provided Robot model navigation



(b) Rviz provided Robot model navigation

(c) Gazebo modified Robot model navigation



(d) Rviz modified Robot model navigation.

Fig. 4: Top, results obtained using the provided robot model. Bottom, results obtained using the modified robot model.

by modifying their numbers it is easy to observe how well the robot is moving on its environment. Furthermore, robot_radius and inflation_radius help the robot to know its limits and the limits of its environments, in certain way. Additionally, the type of the odometry and the laser helps by allowing the users to tune how the sensors reacts within the robot environment.

5 Discussion

From the results obtained it can be conclude that the AMCL algorithm performed really well helping the robot to know its location with respect to its environment. However, a comparison with results obtained using the Kalman filter to know the robot location will be required in order to know how well the AMCL algorithm performs. The implementation of the problem was really straight forward, however, defining the best solution and changing the robot model were more complicated than expected.

Changing the robot model affected the performance of the solution obtained using the provided robot model, because the limits for the collisions and the radius of the robot changed, and therefore, the way the robot interacted with its environments caused the robot to stop while navigating through the provided map.

As result, many parameters were adjusted until the robot navigation performed well, using the parameters presented in table 1. The key parameters for the modified robot model to work correctly were the size of the global and local maps. Further investigation is needed in order to get a conclusion about this problem, or behavior.

Nonetheless, even when the AMCL algorithm is more robust than MCL; handling the kidnapped problem can be problematic. Sometimes, while running the AMCL the robot moved to a different place, like in the kidnapped problem. In those cases the robot was not able to find its current position and it was impossible for the robot to find directions in order to reach the desired location. However, using the AMCL in combinations with other techniques to solve the kidnapped problem is possible, but further investigations is necessary for specific problem.

The use of MCL and AMCL could be used in the industry for different applications, even for the design and development of commercial products such as the roonba products. Additionally, they can be used to determine the position of robotics arms with respect to their environment to develop specific actions at specific places, or locations.

6 Conclusion / Future work

The proposed future work for this problem is the comparison between results obtained using a kalman filter with the results obtained based on the AMCL algorithm, specifically, using the Extender Kalman filter algorithm. Additionally, it is necessary to find ways to improve the use of resources when solving similar problems. Parameters such as the update and published frequencies will help us to have a better flow of the data, but it may not be necessary, especially given that a better flow of the data also increases the use of resources.

Furthermore, measuring the performance of the robot while navigating its environment with different amount of particles is something that needs to be explored, especially since the amount of particles used is related directly with the amount of resources that will be required. Moreover, different analysis can be implemented to determine if a higher number of sensors, or sensors with better performances, could increase the accuracy of the robot location estimation, or at least maintain the current performance while reducing the use of computational resources.

Nonetheless, the implementation of the AMCL algorithm with a real robot can be considered as an additional future work. However, given the complexity related with the use of specialized sensors reduce the possibility for the implementation to be possible. Nevertheless, in case the sensors are available, a cheap remote control vehicle can be used as the robot_model and a nvidia jetson as the processor of the data and the control for the device, or a raspberry pi or an arduino module with a bluetooth receiver can be used in combination with a computer running ROS as as transmitter for the processing of the data and for the implementation of the control system.

References

[1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1322–1328, IEEE, 1999.

[2] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.

[3] D. Fox, "Kld-sampling: Adaptive particle filters," in *Advances in neural information processing systems*, pp. 713–720, 2002.

[4] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, pp. 107–113, IET, 1993.

[5] G. Kitagawa, "Monte carlo filter and smoother for non-gaussian nonlinear state space models," *Journal of computational and graphical statistics*, vol. 5, no. 1, pp. 1–25, 1996.

[6] R. Poppe, "Condensation-conditional density propagation for visual tracking," *Comput. Vis. Image Underst*, vol. 108, pp. 4–18, 2007.

[7] K. Kanazawa, D. Koller, and S. Russell, "Stochastic simulation algorithms for dynamic probabilistic networks," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 346–351, Morgan Kaufmann Publishers Inc., 1995.

[8] A. Doucet, "On sequential simulation-based methods for bayesian filtering," 1998.