

# Localization Project

Reinaldo Ossuna

**Abstract**—In this project we implemented the Monte Carlo algorigtm in two diferents robots created in a Gazebo/Rviz simulated environments, each equipped with laser range sensors and camera.Both robots successfully navigated in the provided map (Jackal Race car).

**Index Terms**—Robot, Udacity, Localization, Monte Carlos, AMCL, Kalman Filter, EKF

---

## 1 INTRODUCTION

LOCALIZATION is the problem of the estimation of robot's pose relative to a map. Cox [1] considers localization the most fundamental problem to providing a mobile robot with autonomous capabilities. There is many localization problems. The simplest one, is *local localization*. Where the initial pose is known and the principal task is to correct the errors in the robot's sensors. The *global localization*, the robot's initial pose is unknown and the robot has to determine it. We can cite more difficult problems, the *kidnapped robot* problem, in which the robot's pose can change any time. And the *multi-robot localization* problem, where multiple robots need to localize themselves. In this project we'll focus on the global localization problem using ROS packages, this use a Augmented Monte Carlo Localization algorithm, wich uses a partivle filter to track the pose of a robot against a know map.

## 2 BACKGROUND

In this section we discuss the two principal approaches to the localization problem, Kalman filter and MCL.

### 2.1 Kalman Filters

The Kalman filter is a set of mathematics equations that is extensive used in area of autonomous or assisted navigation. The filter supports estimation of past, presents and even future states, and it can do so even when the precise nature of the modeled system is unknown [2].

Unfortunately, Once into the nonlinear, non-Gaussian problems, most of the real system are nonlinear and no-Gausian, the Kalman filter don't work very well. With this in mind was created the Extended Kalman filter, which on adapted techniques from calculus, Taylor Series expansions, to linearize a model.

### 2.2 Particle Filters

The Monte Carlo Localization (MCL) represent the belief by a set of weighted vector distributed in the global map. In each period of time the vector are re-sampled and eventually converge to the robot's pose.

"MCL has been at the core of our robot navigation software. It's more efficient and accurate than any of our precious algorithms" This citation from Sarkar prove the importance of MCL in the robotic field. [3].

## 2.3 Comparison / Contrast

MCL has many advantages over EKF, the Table 1 show a full comparison between the both algorithms, but 3 points can be exalted. MCL has a easier implementation, it's represent non-Gaussian distribution and it can approximate any kind of distribution and the more important we can control the memory use.

TABLE 1  
MCL × EKF

	MCL	EKF
Measurements	Raw Measurements	Landmarks
Measurements Noise	Any	Gaussian
Posterior	Particles	Gaussian
Efficiency (memory)	A-	A+
Efficiency (time)	A-	A+
Ease of Implementation	A+	A-
Resolution	A-	A+
Robustness	A+	F
Memory & Resolution Control	Yes	No
Global Localization	Yes	No
State Space	Multimodel Discrete	Unimodal Continuous

## 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 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

Both robots reached the end goal in the map. Beside that more 4 goals was added to the navigation stack and both robots reached all goals.

### 3.2 Benchmark Model

### 3.2.1 Model design

The provided robot has a rectangular prism chassis with a size of 0.4, 0.2, 0.1. It's has two casters in spherical shape with radius of 0.05 and two wheels with radius of 0.1.

### 3.2.2 Packages Used

- AMCL  
A probabilistic localization system for a robot moving in 2D. it implements the Adaptive Monte Carlo Localization (AMCL), which uses a particle filter to track the pose of a robot against a known map.
  - Navigation Stack  
A 2D navigation stack that takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base.
  - Turtlebot Teleop  
Provides teleoperation using keyboard
  - Drive bot  
Define 5 goals to the Navigation Stack

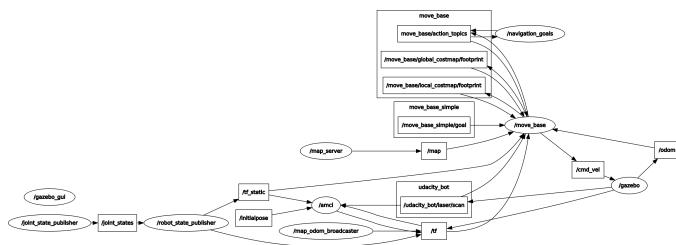


Fig. 1. RQT graph

### 3.2.3 Parameters

### 3.2.3.1 AMCL:

All parameter below are necessary to increase the performance of the algorithms. This parameters can be higher depending on the hardware. The quantity of particles is quite low, but with a small number of particles they converge much faster and much preciser.

- transform\_tolerance  
Time with which to post-date the transform that is published, to indicate that this is valid into the future  
value= 0.3
  - min\_particles  
Minimum allowed number of particles.  
value= 5
  - max\_particles  
Maximum allowed number of particles  
value= 25
  - controller\_frequency  
The frequency at which this controller will be called in Hz  
value= 12

### 3.2.3.2 Costmap Parameters:

- **obstacle\_range**  
The default maximum distance from the robot at which an obstacle will be inserted into the cost map in meters.  
Value= 2.5
  - **raytrace\_range**  
This parameter is used to clear and update the free space in the costmap as the robot moves  
Value= 3.0
  - **inflation\_radius**  
This parameter determines the minimum distance between the robot geometry and the obstacles. See Fig:2  
Value= 0.55
  - **width and height**  
The width and height of the map in meters.  
Global= 10,10  
Local= 5.0,5.0
  - **resolution**  
The resolution of the map in meters/cell.  
Global= 0.05  
Local= 0.02
  - **rolling\_windows**  
Whether or not to use a rolling window version of the costmap  
Global= False  
Local= True
  - **pdist\_scale**  
The weighting for how much the controller should stay close to the path it was given.  
Value= 0.6
  - **gdist\_scale**  
The weighting for how much the controller should attempt to reach its local goal, also controls speed.  
Value= 0.8

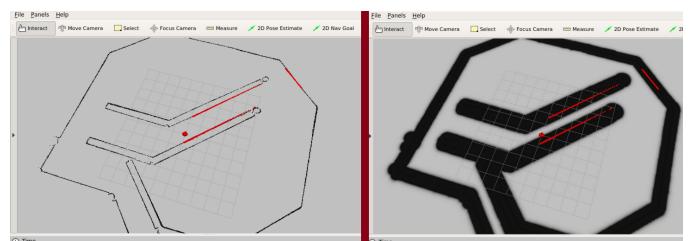


Fig. 2. Low  $\times$  High inflation radius

### 3.3 MeineBot Model

### 3.3.1 Model design

This model has design based in a real robot, Fig:3. All the robot was created using macros and constants values to easy resize the robot if necessary.

The robot is a rectangle with size of 0.1, 0.2 with just one caster and a battery and the raspberry pi in the back used to counterbalance. Like the Udacitybot, it has one camera and a ranger finder.

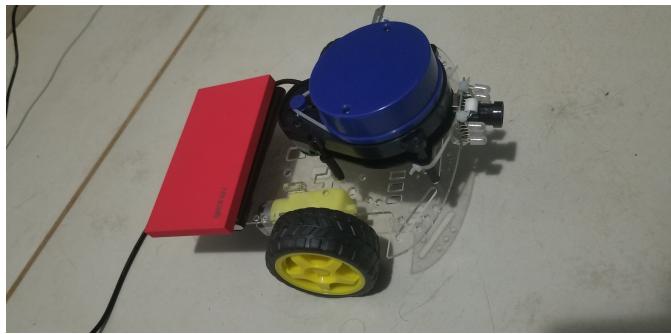


Fig. 3. Real Model

### 3.3.2 Packages Used

The same packages used in the Udacity bot

### 3.3.3 Parameters

Almost all values used in the udacity bot were reused in MeineBot.

- `inflatius_radius`  
Size my robot is quite small, the controller tried to pass between the barrel and the wall. Value= 0.9
- `controller_frequency` The MeineBot is much faster than the udacityBot and the terminal start to print missed controller.  
value= 10

## 4 RESULTS

Both models navigated very well, size MeineBot is smaller, it was much faster and this could bring some problems in the navigation stack.

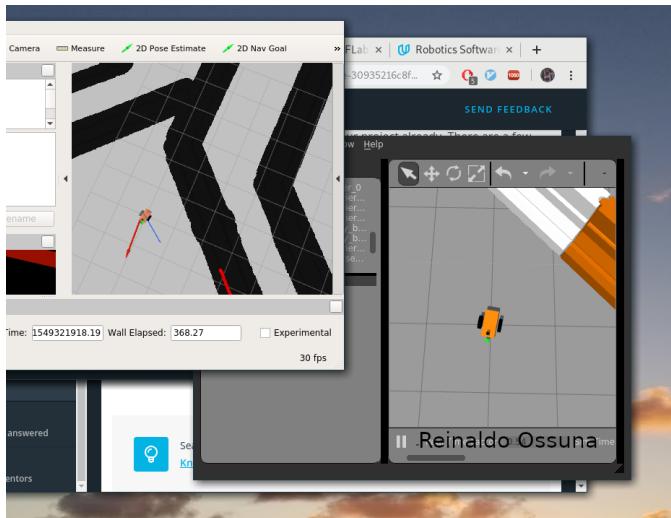


Fig. 4. Udacity bot Goal

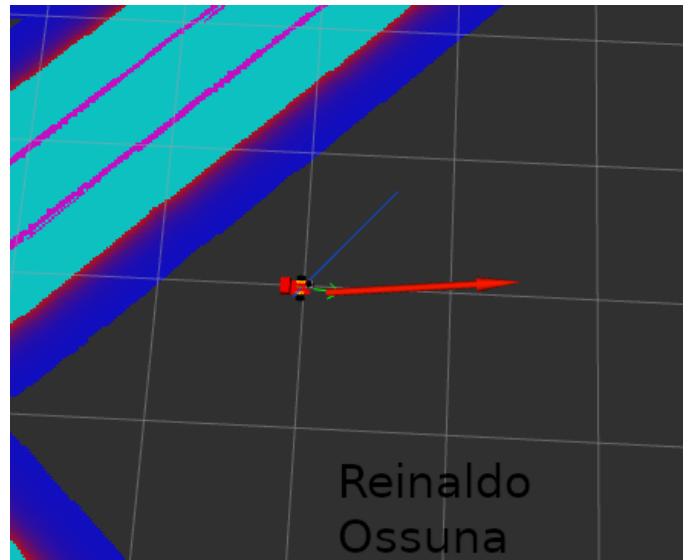


Fig. 5. Meinebot Goal

## 4.1 Localization Results

### 4.1.1 Benchmark

### 4.1.2 Student

## 4.2 Technical Comparison

The MeineBot has a bigger problem, size it's has just one caster and depends the of counterbalance from the baterry. This could induce some type of accident. Another aspect to observe is the ranger finder from the MeineBot is not parallel to the ground.

## 5 DISCUSSION

- Which robot performed better?  
In the localization aspect both robots worked very well. In the velocity aspect MeineBot performed much better.
- How would you approach the 'Kidnapped Robot' problem?  
The AMCL can solve the 'Kidnapped robot problem', size it's add random particles in each re-sampling.
- What types of scenario could localization be performed?  
In know and closed places, without people walking around.
- Where would you use MCL/AMCL in an industry domain?  
The MCL/AMCL would work very well in warehouse or any industry domain.
- Localization accuracy and processing time.  
The trade between accuracy and processing depends on the number of particles used. With a increase of the number of particles the processing time increase but a good accuracy can be archivied with a small number of particles, this was showed in this project.

## 6 CONCLUSION / FUTURE WORK

The AMCL is a powerful tool in the robotic. The fact the all of the algorith is ready to use as a ROS package is facinating.

With small adjust in parameter we are able to deploy this powerful tool.

In the future, deploy the model in a real world and perform similar task performed here. And the use of SLAM algorithm to map a unknown environment.

## REFERENCES

- [1] I. J. Cox, "An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle," no. 9, p. 12, 1991.
- [2] G. Welch, G. Bishop, and C. Hill, "An Introduction to the Kalman Filter by," pp. 1-16, 1997.
- [3] P. Sarkar, *Sequential Monte Carlo Methods in Practice*, vol. 45. 2009.
- [4] P. Pfaff, W. Burgard, and D. Fox, "Robust Monte-Carlo localization using adaptive likelihood models," *Springer Tracts in Advanced Robotics*, vol. 22, no. Mcl, pp. 181-194, 2006.
- [5] V. Celan, I. Stancic, and J. Music, "Cleaning up smart cities-Localization of semi-autonomous floor scrubber," *2016 International Multidisciplinary Conference on Computer and Energy Science, SpliTech 2016*, 2016.
- [6] T. Zhang, Z. J. Chong, B. Qin, J. G. Fu, S. Pendleton, and M. H. Ang, "Sensor fusion for localization, mapping and navigation in an indoor environment," *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2014 - 7th HNICEM 2014 Joint with 6th International Symposium on Computational Intelligence and Intelligent In*, no. November, pp. 1-6, 2014.
- [7] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots Dieter Fox, Wolfram Burgard," *AAAI-99 Proceedings*, no. Handschin 1970, p. 7, 1999.
- [8] B. Ristic and S. A. N. Gordon, "Beyond the Kalman Filter: Particle Filters for Tracking Application," no. 1, pp. 1-5, 2004.