Where Am I?

Panagiotis Angelakis

Abstract—This project goal is to solve the Robot Localization problem in a simulated environment (Gazebo/ Rviz) by applying the Adaptive Monte Carlo Localization Algorithm. A two wheeled robot was designed to traverse the map and be compared with a provided baseline robot. The robot's leverage the Navigation Stack move base and amcl packages to localize and move the robot. By tuning paramaters for the costmap, base local planner,amcl and controller the robot succesfully localized itself and reached the goal position.

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

1 Introduction

Localization in robotics means accurately estimating a robot pose and orientation in a map, despite noisy sensor measurements like camera and LIDAR and imperfect actuators that introduce noise in the robot motion. In this project two robot's were developed and tested in a simulated environment with a known map to solve the global localization challenge (initial pose of robot unknown). The robot's succesfully localize themselves with the help of the Adaptive Monte Carlo Localization algorithm and navigate to the end goal. The template robot was given and named Udacity bot (needed tuning) while the modified bot was designed and tuned with significant changes to the template robot.

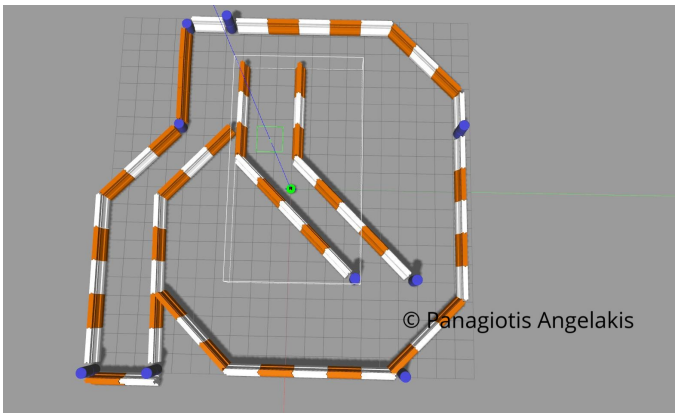

Fig. 1. Gazebo Map

TABLE 1
Table

One	Two
Three	Four

2 Background

There are several ways to approach Robot Localization. The (4) more important are:

- Markov Localization
- Grid Localization
- Kalman Filter Localization
- (Adaptive) Monte Carlo Localization

The two most popular methods will be further discussed

2.1 Kalman Filters

The Kalman Filter is very prevelant in Localization Systems. It excells at taking noisy measurements from multiple sensors in real time and providing accurate predictions of unknown variables like position or velocity. The Kalman Filter Algorithm is based on two assumptions

- Motion and measurement models are linear.
- State Space can be represented by a unimodal Gaussian Distribution

These limitations inhibit the use of the kalman filter in real robots since most of them do not meet these criteria. The Extended Kalman Filter EKF addresses this limitations by Linearizing the nonlinear motion and measurement models using multidimensional Taylor series. But this comes at a cost of higher complexity and compute resources.

2.2 Particle Filters

The Monte Carlo Localization (MCL) algorithm is the most popular in robotics. It assigns to the map a uniformal distribution of particles with different positions and orientations and then removes the particles that are deemed invalid or unlikely. The steps of this algorithm are:

1) Particles are initially distributed uniformly across the space/
2) Measurements are performed resulting in each particle gaining an importance weight
3) Motion is performed and a new particle set is resampled with uniform weights

After some steps the particles will converge in the true pose of the robot. The Adaptive Monte Carlo Algorithm can dynamically assign the number of particles.

2.3 Comparison / Contrast

- Kalman filters require a unimodal Gaussian probability distribution and linear models for measurement and actuation. The Extended KF overcomes these limitations but at a cost of higher complexity and compute resources, however EKF is a more suitable algorithm when the state space is high dimensional.
- Particle filters don't have this limitations are easy to implement and more efficient for our purpose and for that is the algorithm that we are going to use. Furthermore you can trade off accuracy for compute resources by changing the number of particles, so it is applicable even in low end systems.

3 Simulations

Simulations are commoly used to approximate the actual performance of a robot in the real world in a more cheaply, safer and quicker way. Gazebo is used for the physhics engine of the environment and Rviz for visualization.

3.1 Achievements

Both robots were able to localize themselves and reach the specified goal position, by tuning costmap paramaters, base local planner paramaters and AMCL paramaters. The tuning of these paramaters is usually a long iterative process though intuition here can help.

3.2 Benchmark Model

3.2.1 Udacity bot Model design

The provided benchmark Udacity bot was designed in a URDF file. This file describes the size and geometry links and joints of our robot along with physical properties like inertia and collision properties. After that Gazebo can take this model and perform measurements and actions. Specifications of the Udacity bot:

TABLE 2
Udacity Bot specs

Part	Size
Chassis	0.4x0.2x0.1
Wheels	0.1(radius), 0.05 (length)
Caster wheels	0.0499 (radius)
Camera sensor	0.05x0.05x0.05
Hakuyo Sensor	0.1x0.1x0.1

The exact details can be found in the urdf file

3.2.2 Packages Used

For the purpose of launching the mobile in the simulatios a ros package was created called Udacity bot The structure of this package is as follows:

- config
- maps
- launch
- meshes
- rviz
- src

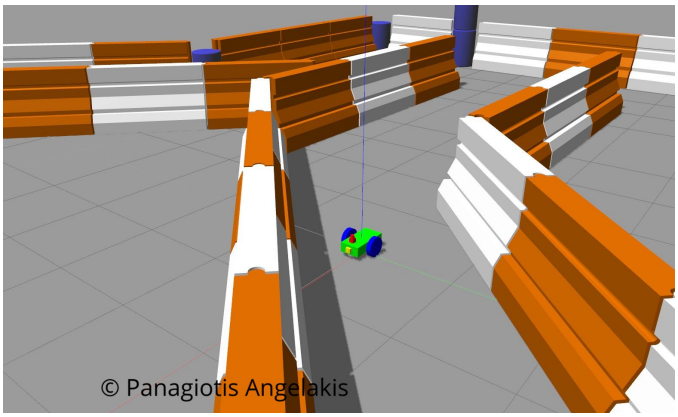

Fig. 2. Udacity benchmark bot

- urdf
- worlds

Navigation's stavk Move base package was deployed for controlling the rover autonomously, along with a teleop package if we want to drive it manually, at last AMCL packages were used for localization purposes. The most important topics are:

- cmd vel
- amcl pose
- /move base/NavfnROS/plan
- /udacity bot/laser/scan
- /udacity bot/camera/image raw

And services:

- /amcl/set parameters
- /map server/get loggers
- /move base/get loggers

3.2.3 Parameters

TABLE 3
Amcl paramaters

min particles	10
max particles	200
odom model type	diff-corrected
initial pose	origin
odom alpha 1-4	0.01

TABLE 4
Costmap and base local planner paramaters

obstacle range	2.5
raytrace range	3
robot radius	0.25
inflation radius	0.8
update frequency	15
max vel x	0.6
min vel x	0.1
occdist scale	0.02
transform tolerance	0.3

- The min and max particles: refer to the amcl algorithm, and depends on the tradeof of accuracy versus speed for the localization algorithm.
- Odom model type: Here we use diff-corrected to correctly model our differential robot.
- transform tolerance: is the maximum allowed delay or latency allowed between transforms
- inflation radius: determines the minimum distance bewteen the robot geometry and the obstacles. A high value can help avoid obstacles but may have trouble navigating in narrow passages.

3.3 Personal Model

3.3.1 Model design

The modified robot main difference from the base line robot is it's shape and size being cylindrical and bigger. Furthermore a sensor base was introduced in the middle of the robot where the camera and laser were put.

TABLE 5
Udacity Bot specs

Part	Size
Chassis	(radius) 0.25 (length) 0.1
Wheels	0.1(radius), 0.04 (length)
Caster wheels	0.0499 (radius)
Sensor base 0.05x0.05x0.05	0.15x0.15x0.15 heightCamera sensor
Hakuyo Sensor	0.1x0.1x0.1

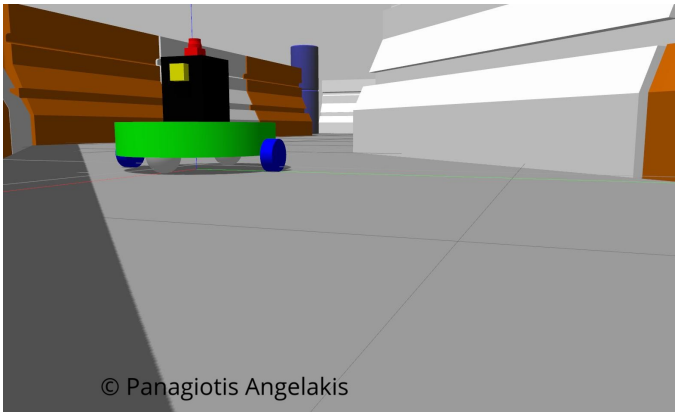

Fig. 3. Modified Robot

3.3.2 Packages Used

The modified robot uses the same packages for localization, navigation and control as the Udacity bot, however it's description is in a different package similar to Udacity$_bot$

3.3.3 Parameters

Some paramaters needed some tuning in order to accomodate the modified robot these were

- obstacle range to 3 (distance that obstacles are added to the costmap)
- occdist scale was set to 0.02 helps for avoiding obstacles
- robot radius was set to 0.5 (allow for leeway)
- inflation radius was set to 0.65 in order for the rover to maintain an accptable distance to the obstacles

4 Results

Both the Udacity bot and the modified one managed to reach the goal in a smooth path and without many hiccups(sometimes an unnecessary rotation may occur). The particle filter converged after a few iteration's and both robot's reach the goal under a minute

4.1 Localization Results

4.1.1 Benchmark

At the start the spread of particles is quite large and the robot is uncertain of it's position

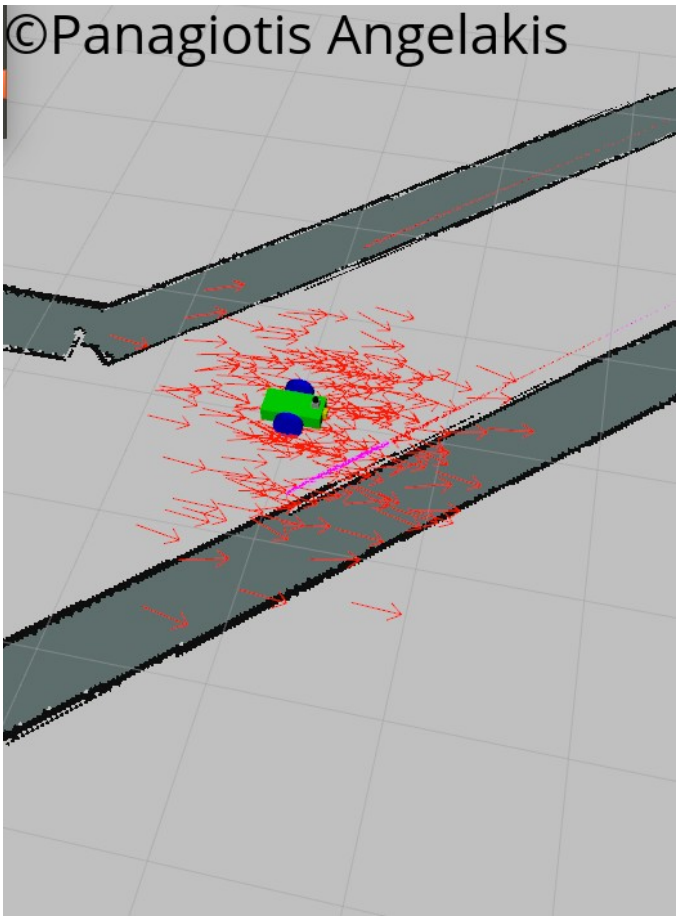

Fig. 4. Initial uncertainty

4.1.2 Student

After a few iterations we see that the particles are almost converging And finally we have reached the goal and the particles are converged

4.1.3 Personal Model

Similar result were achieved for the modified robot after paramater tuning.

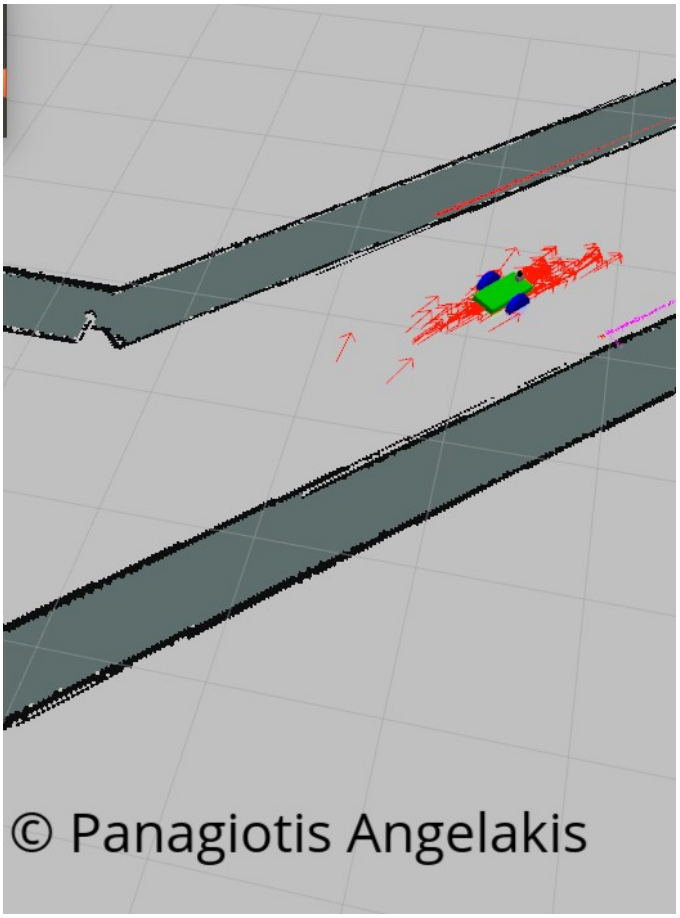

Fig. 5. Semi convergence

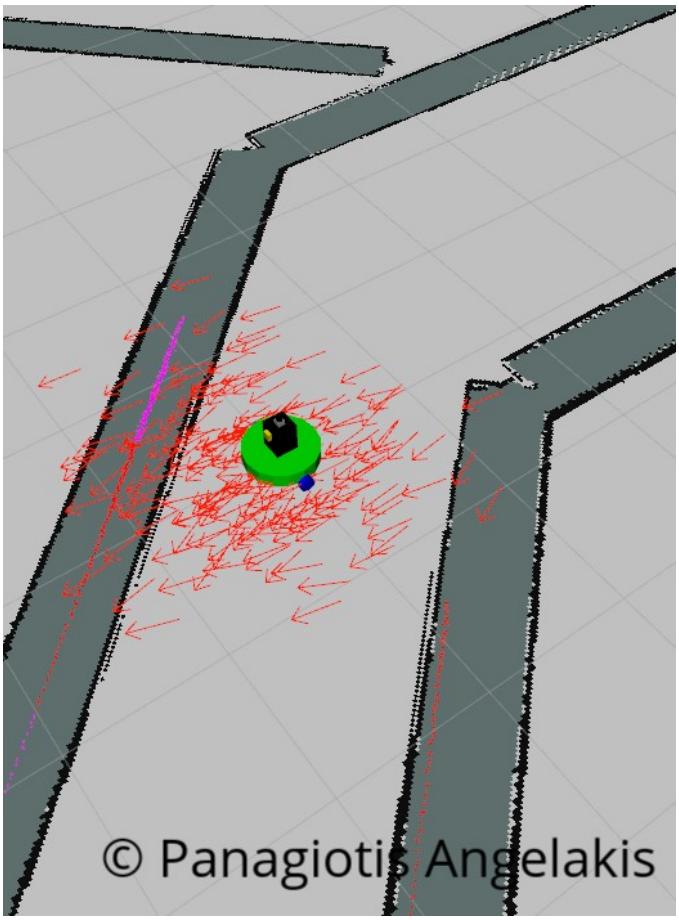

Fig. 7. Initial spread of modified robot

4.2 Technical Comparison

Due to the modified robot being bigger and round was a little slower and it's turning power was diminished however both models were adequate for the task at hand

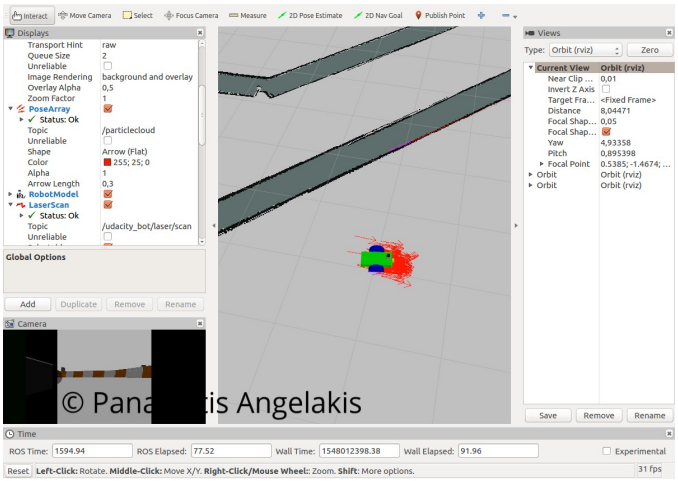

Fig. 6. Reached goal

5 Discussion

5.1 Topics

- Both robots performed equally well, the modified robots bigger size didn't seem to be a drawback in this map beacause the passages were quite large, additional fine tuning of the paramaters may be needed for the robots to navigate narrow passages. The extra height of the sensors in the modified robot didn't impact localization performance in any way since the walls are vertical and uniform.
- The robots should be able to recover if suddenly they lost their bearing also known as robot kidnapped problem since the amcl can adjust the number of particles used based on the uncertainty,however they would need time to recover.
- Localization is important in many different domains where the map is not changing dramatically
- MCL/AMCL algorithm is better suited for close industrial environments that are low dimensional in order for the particles to converge fast and accurately

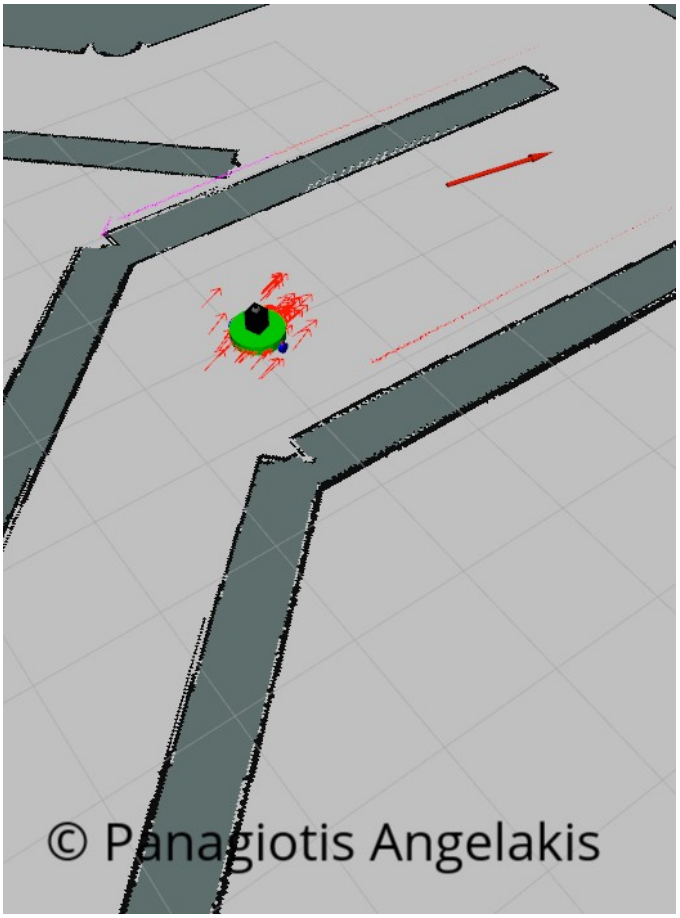

Fig. 8. convergance

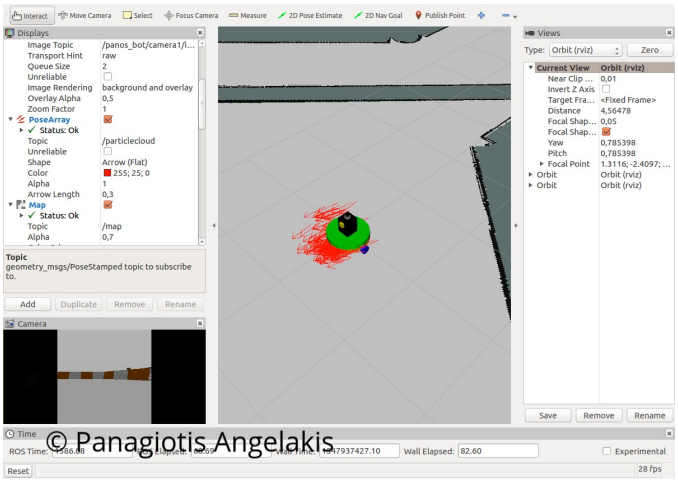

Fig. 9. Reached goal

6 Conclusion / Future work

Both robots were able to perform adequately localizing themselves and navigating to the goal with ease. This was only achieved after excessive iteration and tuning of the paramaters for the amcl and move base package. Future work may include additional tuning of the paramaters in order to improve efficiency in narrow passages and a more complex motion model like the bicycle model.Also different sensors like an RGBD camera might be tested.

6.1 Hardware Deployment

1) In order to build the actual robot one will need a ″brain″ to perform the computations. The TX2 board has adequate compute power for this purpose
2) We will need to have drivers for the camera laser and motors in order to communicate with the different hardware parts of the robot
3) GIO connections would be needed in order to actuate the motors

References