Mobile Robot Localization Using Adaptive Monte Carlo Algorithm

Mahmoud Selim

Abstract—The aim of this project is to establish parameter settings for ROS amcl and move base nodes, such that virtual robots of two designs could navigate robustly in a ROS-based, previously mapped environment. The robots were able to localize themselves, plan their motion, and maneuver obstacles on the road. The robots were also able to follow a set of points to the goal or roam freely to their destination. Additionally, they were able to change distinations and modify their planned path in case of changing goal points.

Index Terms—Where are I?, Adaptive Monte Carlo, Particle Filter, Localization, Robotics Nanodegree, Udacity.

1 Introduction

In 1991, Professor Ingemar J. Cox described the localization problem as "the most fundamental problem to providing a mobile robot with autonomous capabilities". Localization is the capacity of an autonomous agent to know its place and the places of objects in its environment, and is fundamentally a coordinate transformation problem: The robot's software seeks to define its own coordinates within the frame of its surroundings. It is the foundation of the robot's physical agency. The localization problem tries to determine the pose of a mobile robot in a known environment using sensor measurements. The challenges of localization lie in the uncertainties of the robot motions and noises of the sensor measurements. The performance and accuracy of the localization algorithms is of high importance. The higher accuracy, the more calculations you must perform; the more calculations, the more time any algorithm must take due to increasing complexity espesially that these algorithms run on a low computing resources or embedded systems. Localization algorithms, such as Extended Kalman Filter (EKF) and Monte Carlo Localization (MCL), aim at filtering out the noises and accurately identifying the position of the robot in the given environment. The Where Am I? project explores using the Adaptive Monte Carlo Localization (AMCL) algorithm to localize a mobile robot in a simulated environment with a provided map. Accurate localization results require tuning parameters of the algorithm to the specific robot sensors and the maze environment. Using the localization results of the algorithm, the robot can navigate through a maze to reach a designated goal pose.

2 Background

2.1 Localization Problem

As described in Probabilistic Robotics (Thrun et al. 2005), localization estimates the pose of a robot relative to an external reference frame. Localization is one of the most important problems in modern robotics. In order of difficulty, localization can be broken down into three main classes: First position tracking (also known as local localization), where the initial pose is known and then the robot moves. The goal of this problem is to keep track of the robot pose; Second is Global Localization, where the initial pose is unknown and all locations on the map have an equal probability. The goal of this one is to allow the robot to determine it's location and keep track of it.; And finally the Kidnapped Robot Problem where another diffculty is added. At any moment the robot can be kidnapped and moved to another location. That results in high uncertainty in this moment and require a more complex type of algorithms. Localization problem is an important part of modern robotics, since understanding of the environment and successful navigation are both paramount in performing many tasks, such as exploration of the environment, delivery of objects, and mapping the surroundings. There are multiple types of localization problems, and this project considers the local localization problem. The local localization problem has information about the robot's initial pose relative to the static map of the environment. While exploring the environment, the robot can localize itself through a combination of sensor measurements and movements. In the real world, many environmental and internal conditions can make the sensor measurements noisy, while motors and actuators of the robots are not flawless and can incur additional uncertainties. Thus, solutions to the localization problem require filtering out these noises from the sensor measurements.

2.2 Localization Algorithms

2.2.1 Extended Kalman Filter

he extended Kalman filter (EKF) is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance. In the case of well defined transition models, the EKF has been considered the standard in the theory of nonlinear state estimation.

2.2.1.1 Linearization in High Dimensional EKF: The Multivariate Gaussian Distribution is given by the equation in fig1.

And the equations for the EKF are shown in fig2.

2.2.2 Monte Carlo Algorithm

A Monte Carlo Localization filter, also known as a Particle Filter, uses sensor measurements to keep track of a robot's

$$p(x) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

Fig. 1. Multivariate Gaussian Equation.

State Prediction:

$$x' = f(x)$$
$$P' = FPF^T + Q$$

Measurement Update:

$$y = z - h(x')$$
$$S = HP'H^T + R$$

Calculation of Kalman Gain:

$$K = P'H^TS^{-1}$$

Calculation of Posterior State and Covariance:

$$x = x' + Ky$$
$$P = (I - KH)P'$$

Fig. 2. EKF Equations.

pose. The algorithm takes as inputs the previous belief about the robot's pose, the actuation command and the sensor measurements. The belief is given by the Bayes Filter as written in the equation below

$$Bel(X_t) = P(X_t|Z_{1..t}) \quad (1)$$

where Xt is the state at time t, and Z1...t are the measurements from time 1 up to t.

the whole algorithm for the particle filter is shown below

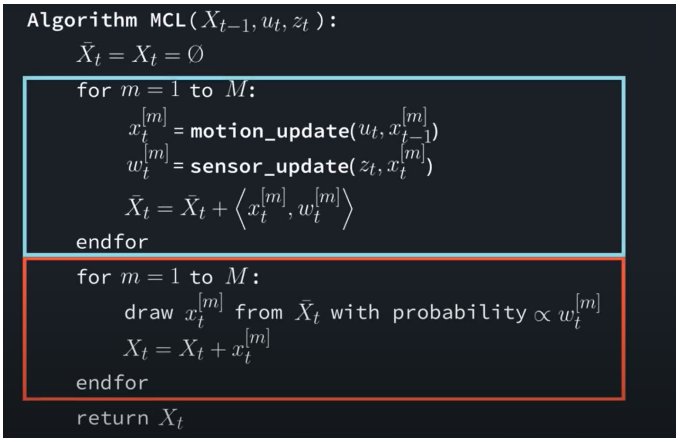

Fig. 3. Monte Carlo Algorithm.

2.2.3 Comparison

The Monte Carlo Localization algorithm, also known as particle filtering, uses particles of varying poses to represent the probability distribution of the robot's pose, and resamples the set of particles by the likelihood of each particle based on measurements of landmarks each time the robot moves. However, the EKF and MCL algorithms differ in many aspects (Table 1). The MCL algorithm can model noise of any distribution, is capable of performing global localization, and is generally more robust in noisy environments. In addition, the MCL algorithm can adjust its memory and time efficiency by changing the number of particles representing the pose's probability distribution. The advantage of being able to adjust resource consumption is further refined with the Adaptive Monte Carlo Localization (AMCL) algorithm. Probabilistic Robotics (Thrun et al. 2005) explained that the AMCL algorithm differs from the MCL with the amount of resamples it takes at every iteration, by accounting for the likelihood of the sensor measurements. This allows AMCL to save significant computational resources compared to the MCL.

TABLE 1
Comparison between Localization Algorithms

Property	EKF	MCL	AMCL
Measurement Noise	Gaussian Distribution	Any	Any
Time Efficiency	Great	Adjustable	Adjustable
Resolution	Great	Adjustable	Adjustable
Memory Efficiency	Great	Adjustable	Adjustable
Robustness	Average	Great	Great
Global Localization	Incapable	Capable	Capable
Kidnapped robot	Incapable	Incapable	Capable

3 Results

3.1 Robot Models

Two robot models were used. The first one is the udacity robot model. The other one is a robot created a race car by adding front portion, a driver seat and a rear wing. The front wheels are simplified while the back wheels are preserved. The laser sensor was placed on top of the driver seat, while the camera is moved to the front of the car.

3.2 Reaching the Goal Position

Both robots are tested with the navigation and amcl packages. They both reach their goals and navigate the environment while localizing themselves very well. Some screenshots on how my custom robot is moving towards its goal in both rviz and gazeboin fig4-11.

4 Model Configuration

4.1 Robot Model

Two robot models were used. They normal udacity bot, and a race car robot. The race car robot has two differences compared to the original udacity robot. The race car has the same sensors as the udacity bot but with a different mechanical design. Both models use the same amcl launch file. They also use the same parameters for the algorithm. However, each one has it's own launch file. A different launch file was made for each of them for ease of use. Both

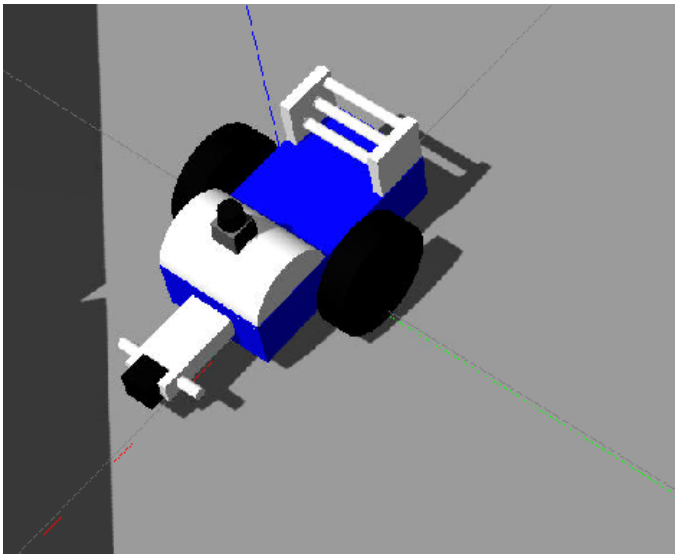

Fig. 4. race car.

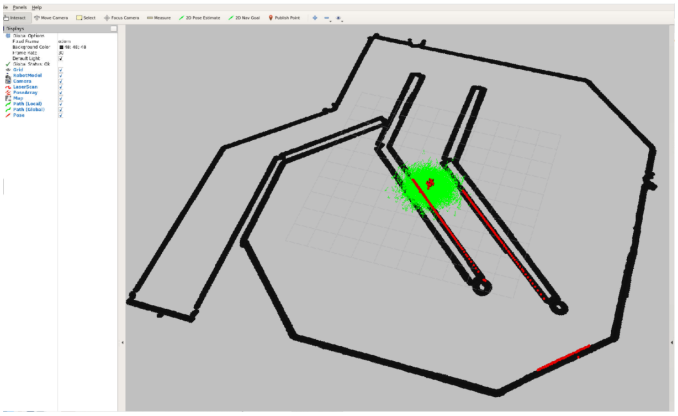

Fig. 5. Race Car Spawn From Rviz.

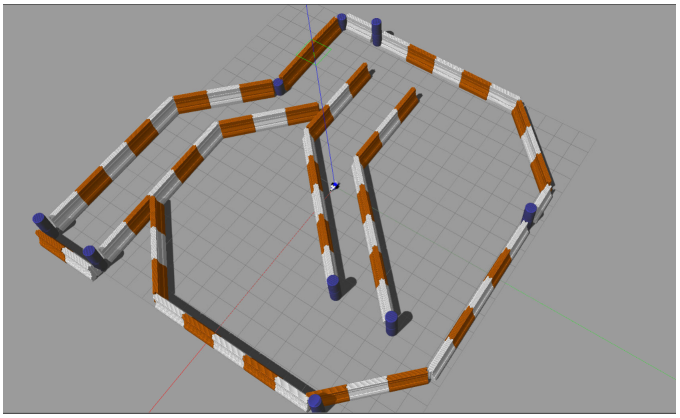

Fig. 6. Race Car Spawn From Gazebo.

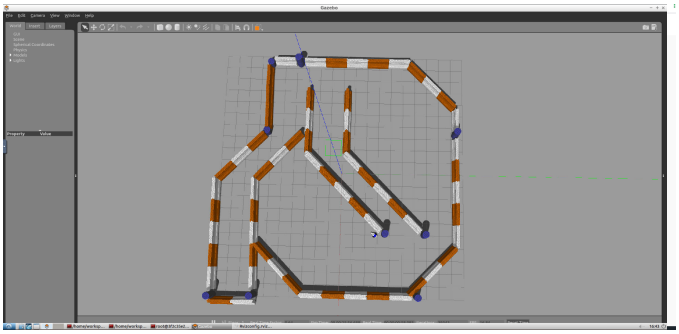

Fig. 7. Race Car Moving in Gazebo.

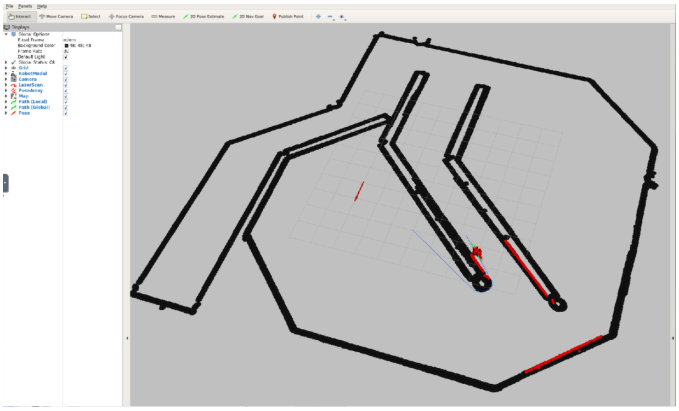

Fig. 8. Race Car Moving in Rviz.

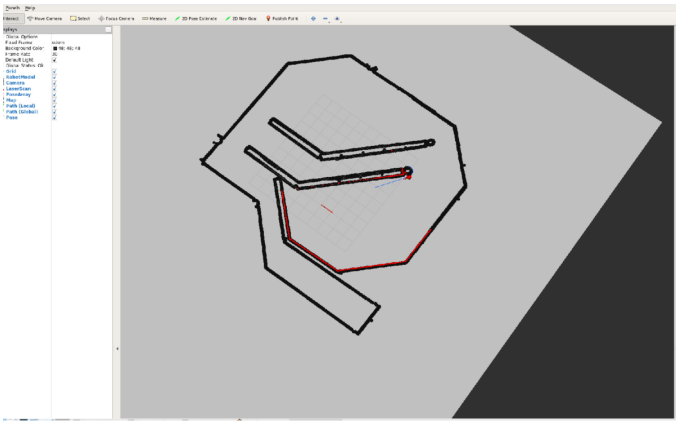

Fig. 9. Race Car Moving in Rviz.

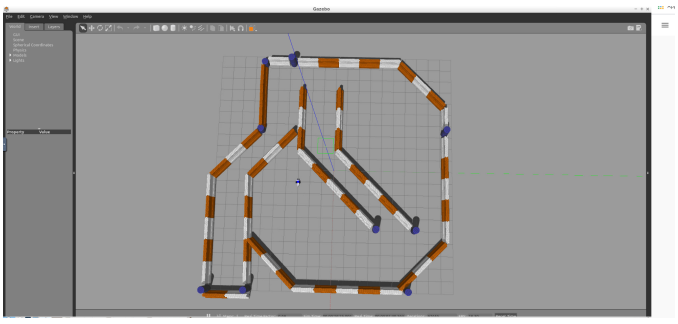

Fig. 10. Race Car at it's goal in Gazebo.

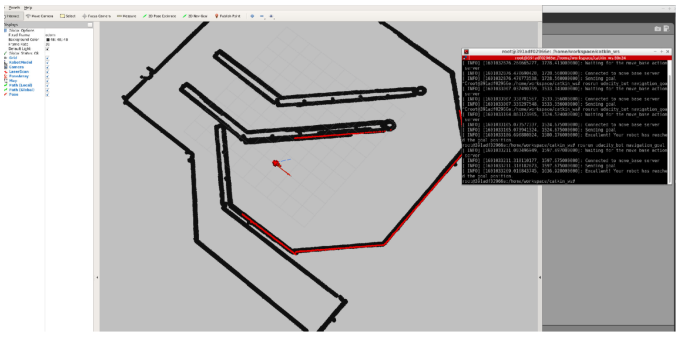

Fig. 11. Race Car at it's goal in Rviz.

cars have the same sensors which are lidar and a camera. They also use the same algorithm (Adaptive monte carlo) and the same parameter configuration.

4.2 Parameter Configuration

Many parameters were tested to suit both robot models. The final parameters used for both robots are provided in tables2-4.

TABLE 2
AMCL Parameters

Param.	Value
Min Number of Particles	200
Max Number of Particles	5000
update_min_d	0.1
update_min_a	0.1
resample_interval	2
laser_max_beams	30

TABLE 3
Local Cost Map Parameters

Param.	Value
update_frequency	4.0
publish_frequency	4.0
width	6.0
height	6.0

TABLE 4
Global Cost Map Parameters

Param.	Value
update_frequency	4.0
publish_frequency	4.0
width	40
height	40

5 Discussion

a) Which robot performed better and why?. The performance of the two robots was very similar. In terms of path planning and following, both robots performed extremely well due to well selected parameters for the move base node, in particular the cost scaling factor which kept the robots close to walls but with enough clearance to not get stuck. not converge. b) 'Kidnapped Robot' problem. Kidnapped robot problem is the problem combining the global localization with the following challenge. At any moment the robot can be kidnapped and moved to another location. That results in high uncertainty in this moment and require a more complex type of algorithms. Both robots were tested for the kidnapped robot and they both were able to recover and go straight to the goal.

d) AMCL in industry domains. AMCL and MCL would be of greatest use in mostly static environments such as warehouses, or perhaps as a patrol robot and specially unoccupied buildings.

6 Conclusion / Future work

Given more computation resources, it would be better to run the algorithm in many situations and with more computations. Also, the navigation script isn't the best thing out there. I think it could be much better and many things can be added to it. Also, a real deployment on a physical system would be great. Simulations are great, but real life scanrios can't be all encapsulated with simulations.