Where Am I?

Mohamed Nour Mejdoub

Abstract—This paper presents a ROS based approach solution for solving the robot localization problem by using the Monte Carlo Algorithm within a Gazebo/Rviz simulated environment The ROS framework offers several configurable plugins that help optimizing how the robot traverse through a provided map. A two-wheeled robot was designed and developed on the basis of a provided baseline robot design, they both apply the navigation stack move base and amcl packages to perform the localization of the robot. By tuning the different costmap and base local planner parameters, the robot can accurately localized itself and managed to successfully plan a trajectory and navigate to the designated goal.

Index Terms—Robotics, Localization, ROS, Monte Carlo Algorithm, Udacity, IEEEtran.

1 Introduction

The increasing need for mobile autonomous robots nowadays, has made the localization problem at the center core of interest and challenges of the robotics field. This capability allows the robot to identify its location within a known map based on its surrounding environment perception and then plan accordingly a path to move to a certain goal with precision. The 3 major localization challenges are The local localization, in which the robot is aware of its initial pose and have to keep tracking its pose as it moves. However in the global localization the robots have no insights of its initial pose but it tries to gain confidence of its position relative to the global map as it visit its corners. In the kidnapped robot problem, the robots initial pose is also unknown, however the robot maybe transported unexpectedly at anytime to another location of the map, which make it more complicated to solve, since the robot would have lost the track of its pose and need to recover from getting totally lost. In this project, the global localization challenge is visited, using ROS packages like **acml** and **move base** to help the two robots localise their poses in a know map (see Fig. 1) and traverse freely with no collisions trough it to target goal.

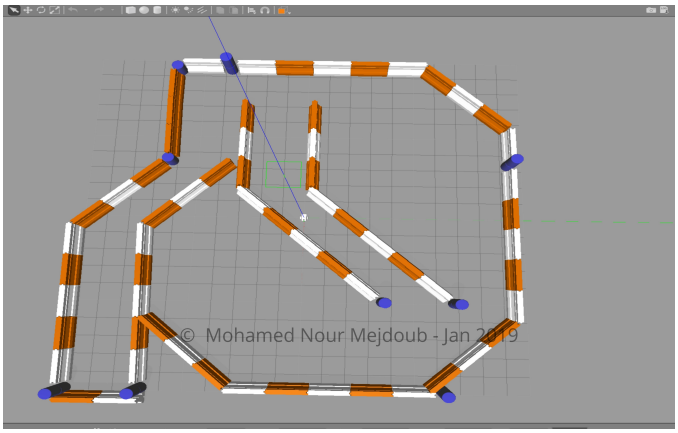


Fig. 1. Project provided map

2 Background / Formulation

Two of the main approaches to implement localization are: Kalman Filter and the Monte Carlo Localisation (Particle filter). In this section both techniques are presented and a short comparison between them is discussed.

2.1 Kalman Filters

The Kalman filter is an estimation algorithm used to track the value of a variable in real time. By collecting data that can be uncertain or containing noise in the measurements it can efficiently and precisely provide an accurate estimate of the real value within few update cycles. This algorithm make assumption that motion and measurement models are linear and that the state can be represented by a unimodal Gaussian distribution to be able include uncertainty in state estimation. However, sine most mobile robots are not bound to linear motions, another variant of the algorithm the Extended Kalman Filter (EKF) linearizes non-linear problems, in order to convert the result into a Gaussian distribution. In addition the Kalman Filter is also used to perform sensor fusion, by combining data from multiple sensors to obtain an accurate estimate of the measured value, for example collecting and fusing RGB-D camera and Radar data to estimate a mobile robot pose.

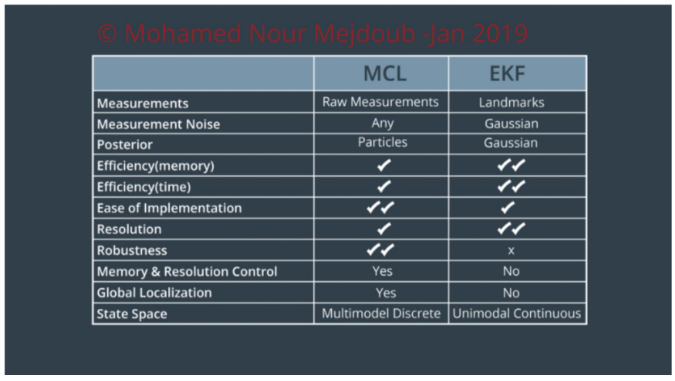
2.2 Particle Filter

The Particle Filter algorithm also known as the Monte Carlo Localization (MCL) is one of the robust and easy to implement localization algorithm, which makes its use very attractive and common for localization application in robotics. The essence idea behind this algorithm is the use of randomly generated particles on the map, whose position and orientation represent a guess of where the robot could be positioned. These particles are re-sampled every time the robot moves and senses the surrounding environment through range-finder sensors like an RGB-D camera. The re-sampling policy is straightforward, the more confident a particle of its pose based on the robot motion and the sensed environment, the more likely it would survive to the next re-sampling cycle. So Eventually, after few iterations, these

re-sampled particles converge with the actual robots pose. The MCL algorithm can be used for both Local and Global Localization problems, and is not limited to linear models.

2.3 Comparison

Although the Extended Kalman Filter can in general localize a robot faster, more precisely and requires fewer memory, when compered the MCL performance, it can't be used for the problem at hand, the global localization. For this purpose the Monte Carlo Localization (MCL) algorithm overcomes the EKF algorithm. The MCL algorithm is easier to implement than the EKF algorithm, which requires a tedious linearization of the state space model, whereas MCL can be used to represent any model. Furthermore, the computation memory and resolution of the MCL algorithm solution can tuned as a hyper-parameters that control its accuracy and speed. (See Fig. 2).



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

Fig. 2. Comparison between MCL and EKF

3 Gazebo/Rviz Simulation

The simulation was developed in Udacitys VM workspace and the first brick in mobile robot localization simulation, is to define the robot's shape and description by specifying the characteristics of its links and joints in an URDF file. This file is then used in ROS framework, where it's launched and displayed on Gazebo and Rviz. A ROS package was created to launch the robot description and its world configuration, which contains the map description. Additionally, the Navigation Stack and AMCL packages were installed on the catkin workspace, to provide the robots with localization capabilities.

3.1 Localization result

Both the baseline and the custom robots were able to localize themselves and reach their goal by tuning costmap and base local planner parameters to cope with the lag and computation limitation on Udacitys VM workspace. In order to meet requirements the tuning was a long and iterative try and error process, but ultimately it enabled the robots to localize itself.

3.2 Benchmark Model: udacity bot

3.2.1 Model design

To develop the provided benchmark mobile robot description an Unified Robot Description Format file (URDF) was created, in which the shape, the size and geometry of the robot is designated and set to have some physical configuration such as inertial and collision parameters. The chassis of udacity bot has a rectangular cube shape with a size of [0.4, 0.2, 0.1], it has also two casters to provide balance with spherical shape with a radius of 0.0499, and two wheels with a cylindrical shape of 0.1 radius and a length of 0.05 that are represented as individual links. The two wheels are connected to the chassis via continuous joints, meaning they are free to rotate around the joint axis. The udacity bot is also equipped with two on-board sensors, a camera and a laser range-finder. (See Fig. 3 for udacity bot visual. See TABLE 1 for detailed specifications)

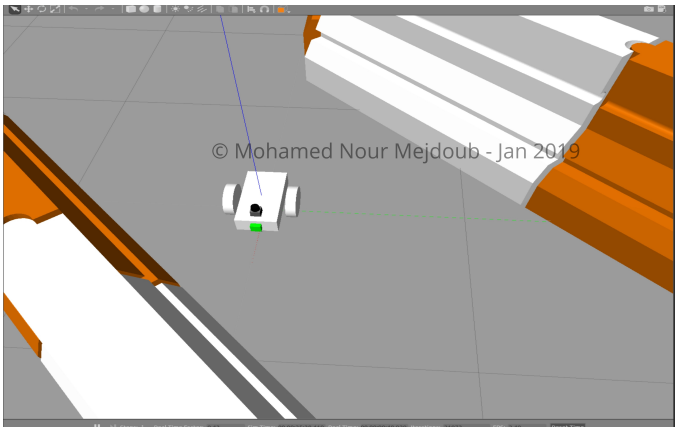


Fig. 3. Udacity Bot Visual

TABLE 1
Udacity Bot Setup Specification

Udacity Bot Body		
Part	Geometry	Size
Chassis	Cube	0.4 x 0.2 x 0.1
back and Front Casters	Sphere	0.499 (radius)
Left and right wheels	Cylinders	0.1(radius), 0.05(length)
Udacity Bot Equipment		
Camera Sensor	Link origin	[0,0,0,0,0,0]
	Shape-Size	Box - 0.05 x 0.05 x 0.05
	Joint Origin	[0.2,0,0,0,0,0]
	Parent Link	chassis
	Child Link	camera
Hokuyo Sensor	Link origin	[0,0,0,0,0,0]
	Shape-Size	Box - 0.1 x 0.1 x 0.1
	Joint Origin	[0.15,0,0.1,0,0,0]
	Parent Link	chassis
	Child Link	hokuyo

3.2.2 Model parameters

The ROS acml localization packages have several parameters (see Table 2 and 3) that can be tuned to optimize the localization accuracy, this is an iterative try and error process to find out a combination of those parameters that worked the best on the simulation environment (VM Workstation). In the AMCL node, the most important parameters

were the min and max particles which were set to 40 and 120, respectively. The choice of these parameters values for instance shall be wise to guarantee simultaneously the mapping accuracy and efficiency on the target simulation environment. Several other parameters were tuned in the different config files. The inflation radius, robot radius, the transform tolerance and obstacle range were obtained after several iterations. Increasing the inflation radius would have an impact on the costmap, as it makes the obstacles look bigger while detecting them and hence adding a buffering distance separating the robot from the real obstacle. Whereas robot radius represents the radius of the robot as it relates to its environment.

TABLE 2
GLobal and Local Costmap parameters: Udacity Bot

Costmap Parameters		
Parameter	Global	Local
global frame	map	odom
robot base frame	robot footprint	robot footprint
update frequency	0.25	7.5
publish frequency	2.0	10
width	20	5
height	20	5
resolution	0.05	0.05
static map	true	false
rolling window	false	true

3.3 Personal Model

3.3.1 Model Design

The personal designed mobile robot, named mnm bot, was developed on the basis of the benchmark provided Udacity bot description by cloning its Unified Robot Description Format file (URDF) all within the same packages with remapped subscribers and publishers. In this file, the size and geometry of the robot is modified and set to have physical configuration such as inertial and collision parameters. The mnm bot has a cylindrical shape chassis with a radius of 0.2 and length of 0.1, two casters to provide balance with spherical shape with a radius of 0.0499, and two wheels with a cylindrical shape of 0.1 radius and a length of 0.05. The two wheels are connected to the chassis via continuous joints. The mnm bot is also equipped with two on-board sensors similar to the Udacity bot setup, a camera and a laser range-finder. (See Fig. 4 for mnm bot visual. See TABLE 4 for detailed specifications)

3.3.2 Used Packages

Similar to udacity bot, the mnm bot uses the same packages for simulation and to perform successful localization. For this end the udacity bot localization simulation launch files were cloned and update accordingly to create an instance of the mnm bot into the simulation environment instead.

3.3.3 Tuned parameters

The same cloning approach continues with the parameters tuning, since the mnm bot was derived from the Udacity bot, they share a lot of common physical, visual and perceptual characteristics, therefore the same parameters configuration was used. (See TABLE 3 to for the different parameters)

TABLE 3
AMCL and Other Parameters: Udacity Bot

AMCL Node Parameters	
min particles	40
max particles	120
initial pose x	0
initial pose y	0
initial pose a	0
odom model type	diff-corrected
odom alpha 1	0.20
odom alpha 2	0.20
odom alpha 3	0.20
odom alpha 4	0.20
update min d	0.05
update min a	0.017
laser min range	0.20
laser max range	-1
laser max beams	50
laser z hit4	0.80
laser z rand	0.20
laser sigma hit	0.25
laserlikelihood max dist	3.00
Costmap Common Parameters	
obstacle range	2.5
raytrace range	3.0
transform tolerance	0.3
robot radius	0.5
inflation radius	0.5
Base Local Planner Parameters	
holonomic robot	false
yaw goal tolerance	0.05
xy goal tolerance	0.1
sim time	1.0
meter scoring	true
pdist scale	0.5
gdist scale	1.0
max vel x	0.5
max vel y	0.1
max vel theta	2.0
acc lim theta	5.0
acc lim x	2.0
acc lim y	5.0
controller frequency	15.0

4 RESULTS

After a long process of try and error involving simulating, testing, and tuning parameters, the results reached at the end were valuable for both robots, Udacity and mnm bots, were able to navigate trough the map with relative ease and reach the designated goal. The particle filters converged quite fast after launching and both robots traversed adequately with minimal interruptions or collisions.

4.1 Localization result

4.1.1 Udacity Robot

At the start of the simulation for the acml algorithm is not confident of the robot position, therefore the particles spread out around of the robot (See Fig. 5).

Once the target position is set on the map, the robot begins moving towards it by flowing a planed trajectory. After a few iterative steps, the particles update their pose and stat to to converge on the robot (See Fig. 6).

While traversing trough the map heading toward the target pose, the robot gains more confident of its pose since the acml algorithm is converging. Eventually the robot

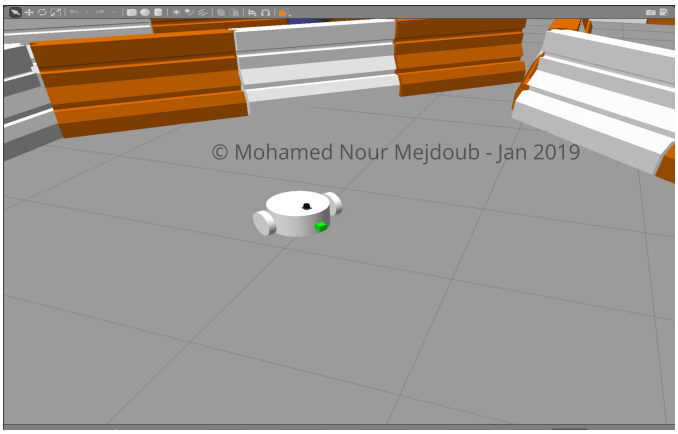


Fig. 4. Udacity Bot Visual

TABLE 4
Udacity Bot Setup Specification

Mnm Bot Body		
Part	Geometry	Size
Chassis	Cylinder	0.2(radius), 0.1(length)
back and Front Casters	Sphere	0.499 (radius)
Left and right wheels	Cylinders	0.1(radius), 0.05(length)
Mnm Bot Equipment		
Camera Sensor	Link origin	[0,0,0,0,0,0]
	Shape-Size	Box - 0.05 x 0.05 x 0.05
	Joint Origin	[0.25,0,0,0,0,0]
	Parent Link	chassis
	Child Link	camera
Hokuyo Sensor	Link origin	[0,0,0,0,0,0]
	Shape-Size	Box - 0.1 x 0.1 x 0.1
	Joint Origin	[0.1,0,0.1,0,0,0]
	Parent Link	chassis
	Child Link	hokuyo

reaches the goal and the particles are aligned with the robot pose. (See Fig. 7).

4.1.2 MNM Robot

In the following figure (fig. 8) one can see the mnm robot at the start of simulation.

In the following figure (fig. 9) one can see the mnm robot after few cycle towards the goal position.

In the following figure (fig. 10) one can see the mnm robot at the target pose.

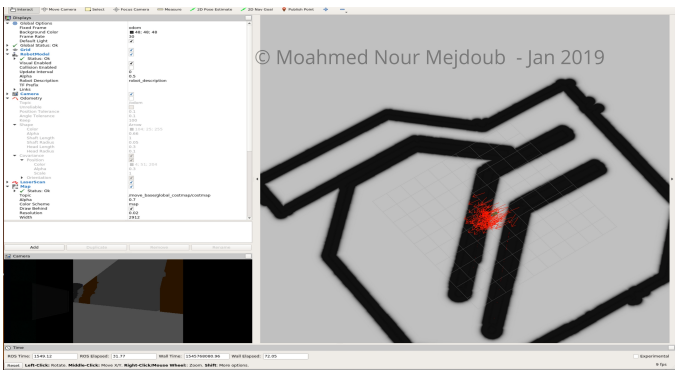


Fig. 5. uncertain localization at the simulation start

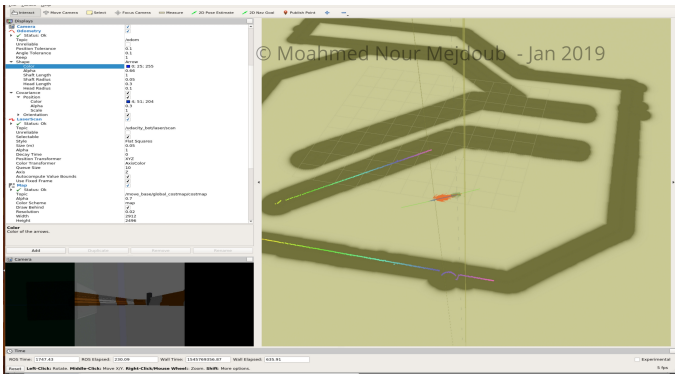


Fig. 6. Localization status after few update cycles

target.jpg

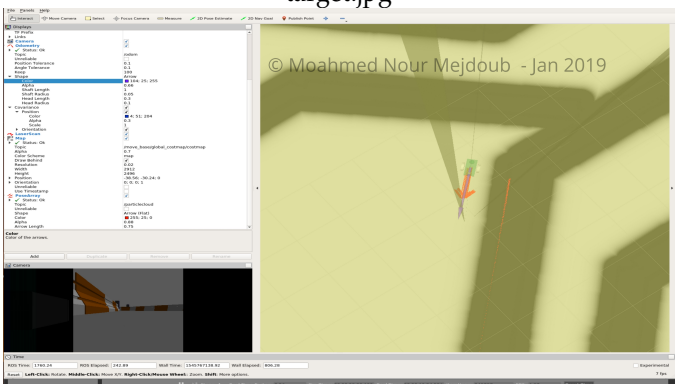


Fig. 7. Localization status at the target pose

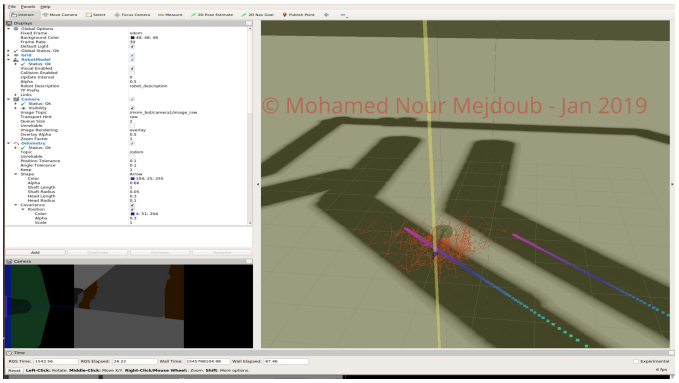


Fig. 8. Mnm bot uncertain localization at the simulation start

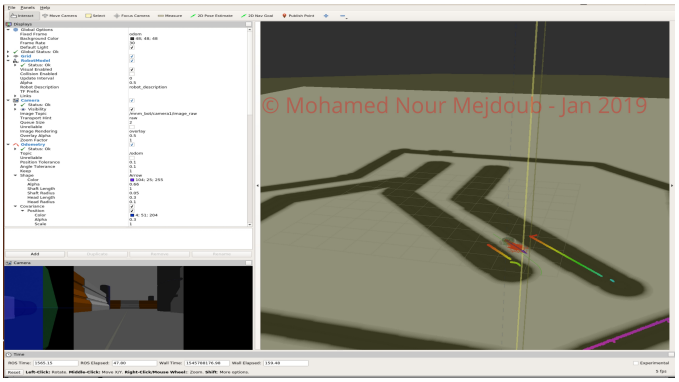


Fig. 9. Mnm bot localization status after few update cycles

target.jpg

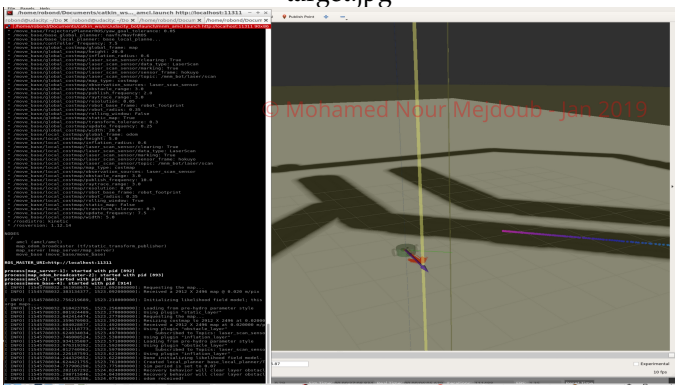


Fig. 10. Mnm bot localization status at the target pose

5 Discussion

Both robot designs, udacity and mnm bots, were able to solve the localization problem. Although they have traversed through the map to the target goal easily, the benchmark model was showing a superior smoother motion and responsiveness. This behaviour lead to the assumption that further adjustment and calibration are required to the mnm URDF physical and inertial configuration to achive similar results.

The ACML algorithm can inherently solve the kidnapped robot problem, since at the starting stage, a higher uncertainty of the robot pose is caused by the random distribution of the particles. Then from cycle to cycle the resampled particles converge to the actual robot pose. Therefore the kidnapped robot problem represents the scenario where the ACML is reinitialized.

The MCL/AMCL algorithms are informed localization approaches where the map is static and already known. It's a best fit for known indoor and closed space mapping, such as indoor autonomous vacuum cleaner. Due to the fact that the particles have to be spread out throughout the environment while applying MCL/ACML mapping, it seems very computationally intensive to cover an immense open space with many particles spread out allover an unbounded area.

The ACML accuracy would be improved if its perception function uses more accurate and precise sensors. In addition, the use of a greater number of particles randomly distributed in the environment will increase the accuracy of the algorithm, since the probability of greater number of particles contributing in precisely estimating the robot pose increases and thus gaining more confident in the actual pose. However, more particles to re-sample means more time is needed for computation, this would affect the efficiency of the localization process. So the chosen number of particles shall be investigated and tuned according the targeted localization precision and the available computational resources.

6 Conclusion / Future work

The ACML represents an attractive solution for solving indoor mapping challenges. It's an easy to implement algorithm, a robust and a powerful tool to use when trying to solve the localization problem for robots. For each giving mapping scenario a different set of tuned variables values should be found out trough a laborious process of setting, testing in simulation and variable tuning. One of the future thoughts is to implement AMCL for 3-Dimensional localization problems, Such as indoors flying drones or indoors warehouse autonomous delivery robots (like amazon is using). But for these more advanced application, the perception shall be more accurate and the application of sophisticated sensor, for instance 3D-Lidar and radars, is a prerequisite.