Overview and Simulation of Adaptive Monte Carlo Localization using ROS on Two Different Mobile Robots

Karim Hassanieh

Abstract—In the following paper we present an overview of localization techniques that are widely used in the robotics community. Details and history of the localization algorithms are presented and described below. A comparison is done showing the advantages of Particle Filters over Kalman Filters. In particular we will describe the Adaptive Monte Carlo Localization Algorithm , a state of the art Particle filter localization algorithm. A simulation is done in ROS (Robot Operating System) for assessing the localization accuracy of the mobile robot using the Adaptive Monte Carlo Algorithm through the package in ROS. Two different mobile robot models are used for the simulations with results presented, compared and discussed below. In addition we will discuss the different input parameters which impact the accuracy of the robot for localization and navigation and the process of tuning those parameters.

Index Terms—Robot, Mobile Robotics, Kalman Filters, Particle Filters, Non Linear Filters, Localization.

1 Introduction

ROBOT localization is a widely studied problem due to its importance in navigation applications. Several sensors can be used to localize robots such as wheel encoders, global positioning systems, laser range finders, monocular and stereo cameras. Such sensors however have a major shortcoming due to physical limitation, which makes the sensors prone to error and noise. For example laser scans can deflect on glass and record inaccurate measurements, cameras have a limited field of view and are sensitive to lighting conditions, wheel encoders can carry error due to wheel slippage, floor roughness and discretized sampling of wheel increments.

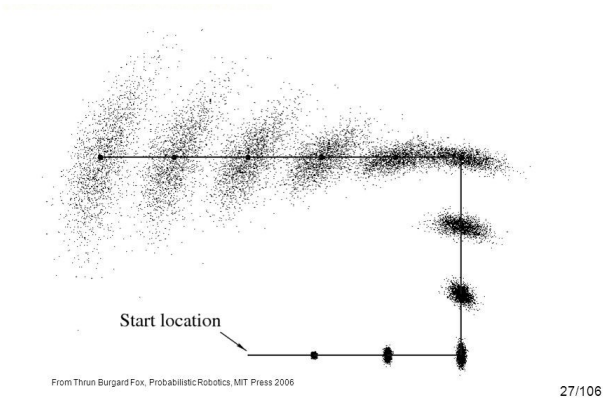

Fig. 1. Accumulation of Error and Uncertainty as a result of Robot Motion [1]

As a response to these shortcomings, several algorithms were developed to filter the noise and provide a more accurate pose estimate on where the robot could be located. The localization algorithms try to merge the different information from these sensors by modeling each sensors error and then predicting the most likely location of the robot. In this paper we present an overview of the most popular localization algorithms, the Kalman and Partcile Filters, while discussing the advantages and disadvantages of each. We will also preform localization in ROS and assess the algorithm under two different mobile robots configuration in a simulated environment.

2 Background

During the 1990's work in robotics has shifted greatly towards techniques that acknowledge the imperfections of collected sensor and robot models thus placing heavy focus on probabilistics. Probabilistic robotics tends to integrate imperfect sensors and models through probabilistic laws such as Bayes rule [2].

In the below sections we will describe the most common and popular state estimation techniques, which are the Kalman Filters and the Particle filters. We will also show the advantages, disadvantages of each family of state estimation algorithms.

2.1 Kalman Filters

The Kalman filter was invented by Swerling (1958) and Kalman (1960) as a technique for filtering and prediction in linear Gaussian systems [1].

Kalman filters work in two steps and is detailed in Figure 2.

Kalman Filters Algorithm:

1) Prediction Step : In the prediction step the robot state is updated based on the robot model assumption. In addition the error in the covariance matrix is projected.
2) Update Step : In the update step the Kalman Gain is calculated. After estimating the Kalman gain the

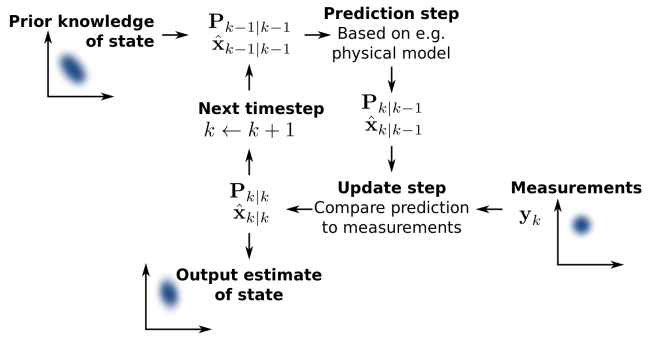

Fig. 2. Basic Steps of a Kalman Filter

measurement reading is used to update the pose estimate and error is covariance matrix.

It is important to note that both the system and observation models are assumed to be linear which is not a realistic assumption. Additionally it is also assumed that the state belief is a linear assumption.

To resolve the issue of linearity Extended Kalman Filters were developed. The algorithm is very much similar to the traditional Kalman Filter algorithm with difference that a linear predictions (state and measurement predictions) in the Kalman Filter algorithm are replaced with non-linear generalizations in EKF's.

2.2 Particle Filters

Particle Filters are the alternative non-parametric implementation of the Bayes rule. The below describes the steps involved in the Particle filter algorithm.

Particle Filter Algorithm :

1) Particle Initialization : The algorithm generates hypotheses on the pose, the hypotheses are called particles and are distributed over the robot map, environment.
2) Particle Sampling : The particles are evaluated based on measurement/sensor reading and given weight. The most likely particles/hypotheses are kept and are kept when re sampling. The particles which are kept then are updated based on the robot motion model.

The main benefit behind particle filters is that instead of representing a distribution by a parametric form - which would have been the exponential function that defines the density of a normal distribution - particle filters represent the distribution by a set of samples drawn from this distribution [1]. This allows the representation of highly non-linear models.

2.3 Comparison / Contrast

For the sake of evaluation and in order to select the ideal filter for our simulation a comparison is done between Particle and Kalman Filters [1].

Factors	*Extended Kalman Filters*	*Particle Filters*
Measurements	Landmarks	Raw Reading
Measurement Noise	Gaussian	Any
Posterior	Gaussian	Particle
Efficiency - Memory	++	+
Efficiency - Time	++	+
Ease of Implementation	+	++
Resolution	++	+
Robustness	-	++
Global Localization	No	Yes

Particle filters would be best fit for the simulation to be preformed. This is due to the fact that particle filters are capable of the following :

1) Capability to do Global localization of the robot within its map.
2) Modeling of highly nonlinear models.
3) Localize without the need of landmarks (through the use of laser raw readings).

Such factors are highly critical for accurate localization. Accordingly the paper describes the simulation done based on particle filters.

3 SIMULATIONS

A simulation was done on both two different robot models with Particle Filter - *Monte Carlo Localization* using ROS with Gazeebo and Rviz for visualization.

1) Benchmark Model *("Udacity -bot")* : The first model is meant to be the standard reference mobile robot as provided in the Udacity *" Where am I ?"* Project Section.
2) Vacuum Sweeping Robot Model *("Karim-bot")* : The second model is meant to represent the typical sweeping robot with its circular chassis.

Both simulations were done in a simulation map which is filled with narrow barriers as shown in Figure 3 using Particle filters - *Monte Carlo Localization*.

The simulation hardware were done a DELL laptop with a proccessor of *Intel Core i7-7500U CPU @ 2.70GHz by 4*.

3.1 Achievements

Both robot models were able to successfully reach the destination location in simulation. Successful simulation results are shown in Figure 8 & Figure 10.

The below sections discuss details of the robot model and how the accurate localization was achieved.

3.2 Benchmark Model *"Udacity-bot"*

3.2.1 Model design

The below parameters in Table 1 further describe the parameters which were taken into consideration to develop the Benchmark Model *"Udacity-bot"*. The physical model of the robot visualized in Gazeebo is also shown in Figure 4. The robot basically consists of a rectangular chassis, left and right wheels , and back/front caster wheels.

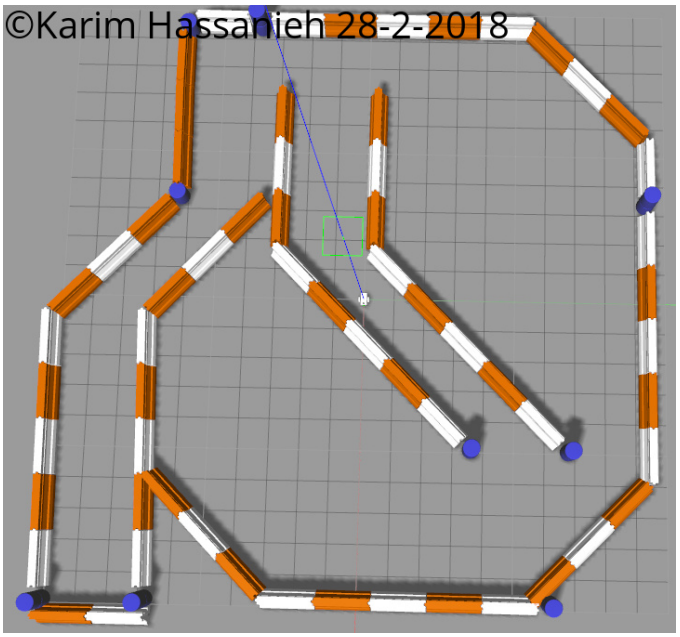

Fig. 3. Simulation Map

Udacity Bot Physical Parameters		
Component	Shape	Size
Chassis	Rectangular	0.4x0.2x0.1
Left Wheel	Circular	0.1
Right Wheel	Circular	0.1
Back Caster Wheel	Circular	0.05
Front Caster Wheel	Circular	0.05

TABLE 1
Udacity Bot Parameters

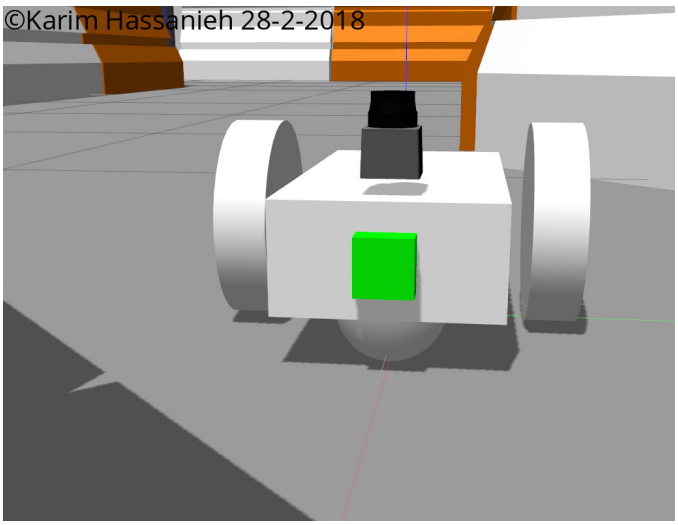

Fig. 4. Benchmark Model

3.2.2 Packages Used

Simulation were preformed on ROS (Robot Operating System) [3]. The main Packages used from ROS include the following

1) AMCL Package : The AMCL Package is an adaptation of the Adaptive Monte Carlo Algorithm in ROS. AMCL tries to match the scan to the given environment map along with odometer readings to give accurate localization results.
2) Move base Package : Given a goal in the world, the robot will attempt to reach it with a mobile base. The move base node links together a global and local planner to accomplish its global navigation task. The move base node also maintains two costmaps, one for the global planner, and one for a local planner that are used to accomplish navigation tasks.

Full details/relation of the topics, services and packages used are displayed in the rqt-graph (Figure 5)

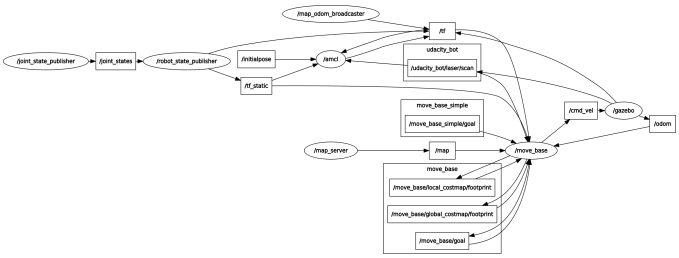

Fig. 5. ROS Graph for Simulation on Udacity Bot

3.2.3 Parameters

Tuning the parameters was the main challenge in completing the project for successful localization. We will only discuss the main key parameters.The remaining parameters are part of the github source code and will be too much to detail in this paper.

The *odom alpha parameters* represent the noise in the odometer measurements. If the values were high the particle filters would take a longer time to converge therefore the parameters had to be tuned in such a way to take into account the highest level of noise while guarantying the convergence of the algorithm. Thurefore the values were decreased to ensure stability and convergence of the system.

The number of *particles* was chosen to be between 25-200. Having too many particles added on the computational cost and made the processing extremely slow. On the other hand the lower the number of particles lowers the accuracy in localization.

The *transform tolerance* was also important parameter to be tuned. The transform tolerance is the tolerable delay in the transform data in seconds. The tolerance was selected at 0.2 after many experimentation. The parameter is basically a trade-off between accuracy and robustness of the system at work.

Inflation radius was also important as it relates to the maximum distance at which obstacles will have a cost. The radius was set to be at 0.45.

The *obstacle range* described the distance at which the laser sensor would insert a detected obstacle in the cost map. The value was set at 5 meters. The ray trace range is the maximum range in meters to ray trace obstacles in the map. The optimal value was found to be at 8 after various iterations.

Robot Radius describes the robot of the radius. It was found that if the value was too small it would get stuck and

Sweeping Bot Physical Parameters		
Component	Shape	Size
Chassis	Circular	0.2
Left Wheel	Circular	0.03
Right Wheel	Circular	0.03
Back Caster Wheel	Circular	0.03

TABLE 2
Sweeping "KarimBot" Parameters

hit the walls. Having the value too high the robot would stop due to the fact that the inputs would indicate that the robot is too big to pass through the passage.

Yaw Goal tolerance describes the rotation tolerance around the z-axis which is acceptable between goal and current robot position. The value was decreased to reach a more accurate pose with respect to the goal state.

3.3 Sweeping Robot Model *"Karim-bot"*

The sweeping robot is meant for indoor vacuum cleaning applications. Below parameters in Table 2 further describe the parameters which were taken into consideration to develop the Sweeping robot *"Karim-bot"*. The physical model of the robot visualized in Gazeebo is also shown in Figure 6. The robot basically consists of a circular chassis, left and right wheels, and back caster wheels.

Fig. 6. Sweeping Robot Model

3.3.1 Model design

Table 2 parameters further describe the parameters which were taken into consideration to develop the Sweeping Robot Model. The urdf file describing the full details is part of the github code under karimbot package. The robot is also visualized in Gazeebo in Figure 6

3.3.2 Packages Used

Same Packages were used as described in the Udacity-Bot refer to Section 3.2.2

3.3.3 Parameters

Parameters similair to those for udacity-bot were used. The performance was acceptable so not much further alterations were done to what was done to the benchmark udacity-bot model.Refer to Section 3.2.3

4 Results

Both robots were able to reach the destination successfully. In Section 4.1 obtained results are presented.

4.1 Localization Results

4.1.1 Benchmark Model "Udacity-bot"

Figure 7 shows the robot at the beginning of the simulation. As shown around 200 particles are shown and distributed. Showing a large uncertainty in the robot position.

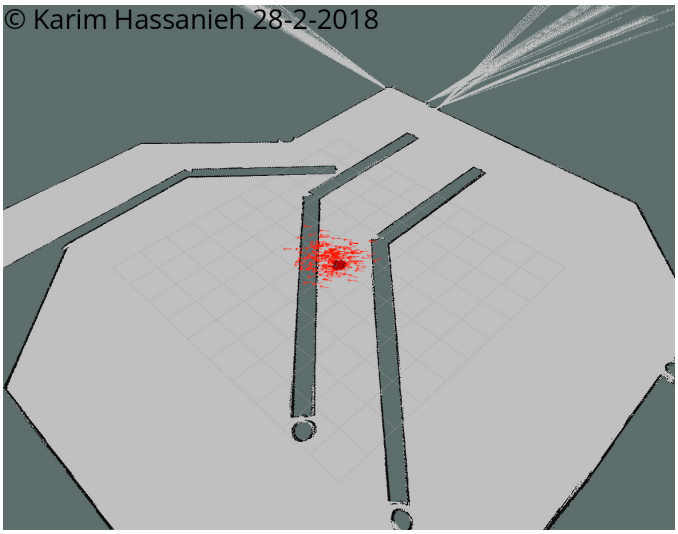

Fig. 7. Initial Udacity-bot Location

Figure 8 shows the robot at his final position with the goal reached and limited number of particles.

4.1.2 Sweeping Robot Model "Karimbot"

Figure 9 shows the robot at the beginning of the simulation. As shown around 200 particles are shown and distributed. .

Figure 10 shows the robot at the end of the simulation with little number of particles.

4.2 Technical Comparison

Both robots were able to reach position under 14 minutes. It was noticed that even though the physical configuration (mass and components) of the robots was different this did not impact much the simulation time and the objective to reach under 14 minutes the goal position.

5 Discussion

- Both robots achieved good results the fact that the sweeping robot is lighter did not impact the performance much.
- It should be noted that the robot does work in the kidnapped case scenario and global localization. The random initialization of particles in the map environment and the dynamic particle re-sampling would ensure that the particles can reallocate accurately. As described each particle will be assessed based on the sensor reading input and location of the particle which ensures that the algorithm will succeed in such scenarios.

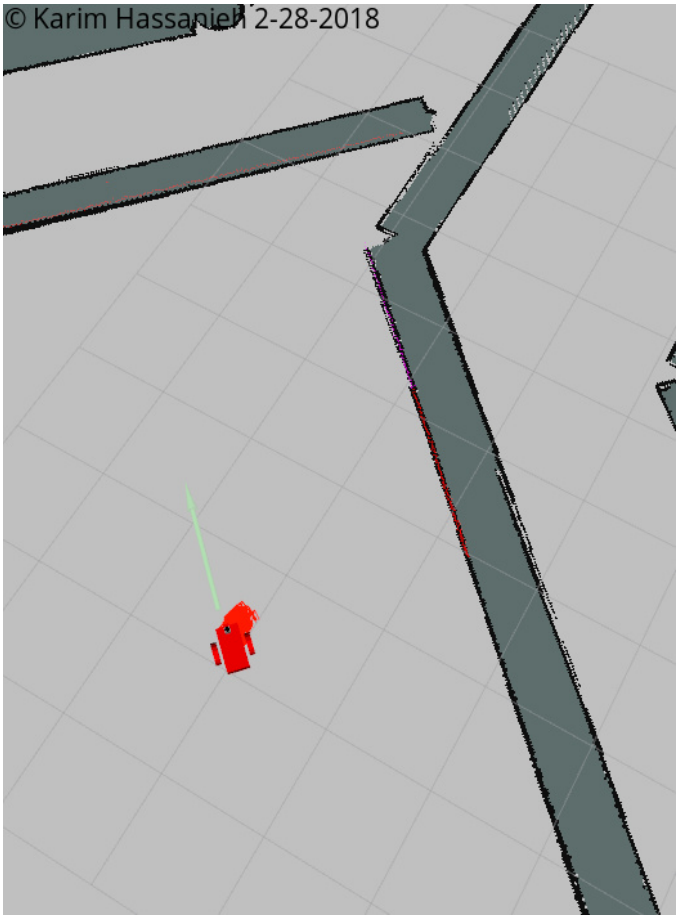

Fig. 8. Final Udacity-bot Location

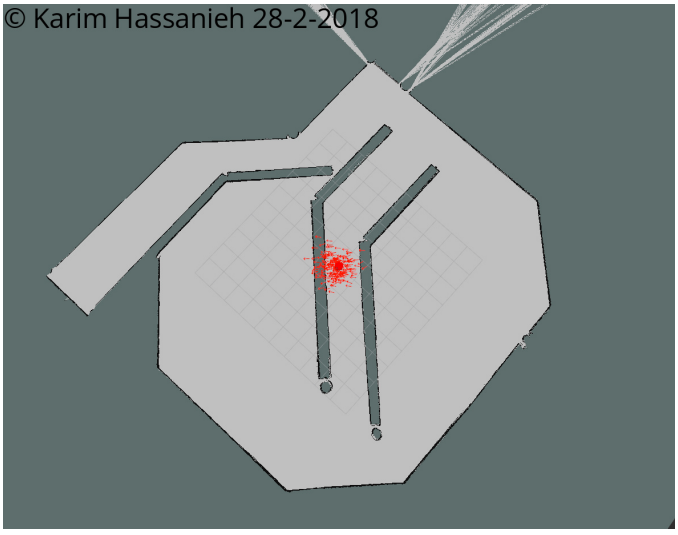

Fig. 9. Initial Sweeping-bot Location

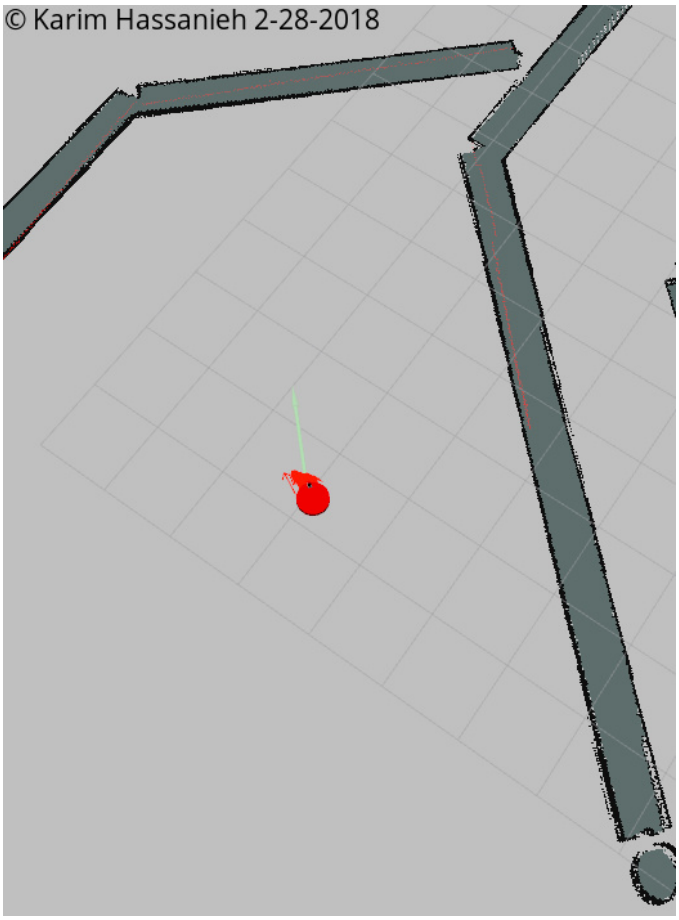

Fig. 10. Final Sweeping-bot Location

- MCL/AMCL would work perfectly in known indoor environments where the scene is static. Examples would include self serving robots were the robot has to localize and reach out to specific locations at specific service points. Robots in factories who are also transporting packages and materials need to accurately localize to be able to reach the correct shelf/container.

6 Conclusion / Future work

Through experimentation it was clear that the particle filters successful localization is highly dependent on the number of particles introduced. The greater the number of particles the more accurate the localization. This however comes at the cost of processing and computational time. At times the robot may need to reach the goal position immediately (consider a mobile firefighting rescue robot). Such scenarios require immediate action with low processing by the robot along with an acceptable localizing error.

While the results have proven to be successful there are still various ways to improve on the results obtained. The camera installed on the robot can be used to localize with the visual odometer or structure from motion algorithm. The camera sensor feedback can then be used as input to correct the odometer readings and have better localization results. In addition we can also include 3D Laser sensor to better

localize in a 3D environment. The given Hokuyo sensor is utilized only for 2D readings.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[2] S. Thrun, "Particle filters in robotics," in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, (San Francisco, CA, USA), pp. 511–518, Morgan Kaufmann Publishers Inc., 2002.

[3] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.