2019 19th International Conference on Control, Automation and Systems (ICCAS 2019)
Oct. 15~18, 2019; ICC Jeju, Jeju, Korea

Particle Filter-based Localization of a Mobile Robot by Using a Single Lidar Sensor under SLAM in ROS Environment

Dhruv Talwar[1] and Seul Jung[2*]

[1] Department of Mechanical Engineering, IIT Delhi,
Delhi, India (Dhruv.Talwar.me116@mech.iitd.ac.in)
[2] Department of Mechatronics Engineering, Chungnam National University,
Daejeon, 34134, Korea (jungs@cnu.ac.kr) * Corresponding author

Abstract: One of the most popular issues in autonomous mobile robots is mapping, localizing and autonomous navigation. In this paper, Adaptive Monte Carlo Localization (AMCL) as particle filters method is presented to show how effectively it localizes the mobile robot in an indoor environment. During the simulations, the robot is localized and autonomously navigates in Gazebo and Rviz environment. The simulation results demonstrated that the particles in the filter quickly converge on the pose and the robot was successfully able to follow the path to reach its goal position. Simulations for a mobile robot to be localized in two different environments, static and dynamic, were carried out. The robot was successful in reaching its goal every time. Simulation results thus point out that AMCL performed effectively in these environments.

Keywords: ROS, localization, autonomous driving, AMCL, SLAM.

1. INTRODUCTION

With a rapid development in the field of mobile robots, it is now required of the upcoming mobile robots to have the capability of mapping and localization. The robot cannot take any decisions and actions on its own without knowing the current pose and orientation. Localization of the mobile robot in an environment plays a vital role to solve autonomous navigation.

SLAM is the main goal for a robot to acquire the map of the unknown environment while simultaneously the robot's pose in the map is localized. SLAM is a necessity to achieve obstacle avoidance in global and local path planning and ultimately autonomous navigation.

In the last years, a lot of progress has been made with SLAM and a lot of its challenging tasks have been tackled including computation complexity, data association and loop closure in a single robot.

Grisetti *et al.* presented many innovative methods to reduce the number of particles in the Rao-Blackwellized particle filter [1]. This algorithm takes in account the movement as well as the sensor updates to get the distribution of the robot in an environment. Imthiyas presented indoor localization by collecting large data sets from the indoor environment and producing a Gaussian Process(GP) model [2]. Dieter Fox proposed the Kullback-Leibler distance sampling(KLD sampling) which uses a small number of samples to represent the belief of the robot [3]. It generated samples until the number of particles is enough to guarantee that the KL distance between the robot's actual position and the estimate does not exceed a certain bound error. This technique uses an adaptive statistical method based on particle filters.

A new development in autonomous navigation and SLAM is the Fast-Slam where a particle filter is used and the paths for the robot and data associations are now represented as samples for the particle filter [4]. Much work has been done on this recent technique Fast-Slam. Montemerlo *et al.* proposed a FastSLAM2.0 algorithm which gives more weightage to the most recent measurement in the process of pose estimation [5]. Perez *et al.* compared the performance of localization for a mobile robot based on different sensors like vision systems and laser rangefinders [6]. They calculated the precision of the different localizations and compared them saying that a laser scanner is a more robust solution than vision systems.

Motivated by the AMCL algorithm, in this paper, a robot in a simulated environment is localized. Once an occupancy grid or map of the environment is formed the pose and orientation of the robot in the reference map frame are obtained. For this task, it is assumed that a single lidar sensor is used to implement SLAM in ROS environment. Simulation studies for static and dynamic environments are conducted.

2. ROS & ENVIRONMENT

2.1 ROS

Robot Operating System (ROS) is a Linux based software and has many prebuilt libraries and packages which can be modified and changed accordingly to build robot functions' framework, where it used nodes, packages, messages topic and services [7]. Any executable code which takes data from any of the robot's sensor and passes the same or the output value to another code is termed as a ROS Node. The ROS system is like a send and receive system of the data from the sensors, which are called messages are sent to nodes from ports called as topics. A node that sends messages on a topic is called a publisher node and the node which subscribe to the messages sent via another topic is termed as the subscriber node. All nodes are combined in a ROS package which can very easily be combined on any Linux software with ROS installed [8].

2.2 System environment

For the experiments conducted in Simulation we used Ubuntu 16.6 and ROS Kinetic version. For the real-life simulation, we used Ubuntu 14.04 and ROS Indigo.

The Lidar used here was the Hokuyo URG-04LX-UG01. The Laser beam diameter is less than 20mm at 2000mm with maximum divergence 40mm at 4000mm [9]. Table I lists the specifications of the lidar sensor.

Table I Lidar sensor specifications

Parameter	Value
Light Source	Infrared
Wavelength	785nm
Scan Area	240°
Maximum Radius	4000mm
Pitch Angle	0.36°

3. AMCL

The adaptive Monte Carlo approach uses particles to localize the robot. These particles have their own coordinate and orientation values just like the actual robot along with a given weight. The weight value (w_t) is defined as the absolute difference between the actual pose of the robot and the predicted pose by that specific particle. The bigger or larger the weight of the particle the more accurately it defines the pose of the robot.

Whenever the robot moves in the environment and gives new sensor data the particles are re-sampled. With each re-sample, particles that have low weights perish and the particles with high weights survive. After many iterations of the AMCL algorithm, the particles will converge and evaluate an approximate of the robot's pose. Therefore, this algorithm estimates the robot's orientation and position based on the sensor's input [10]. Table II shows the pseudocode for AMCL algorithm.

Table II AMCL Algorithm

1: **procedure** AMCL(x_{t-1}, v_t, s_t)
2: $X_t \leftarrow \varphi$
3: for n =1 to N loop:
4: $x_t^{[n]} \leftarrow$ Motion Update(v_t, $x_{t-1}^{[n]}$)
5: $w_t^{[n]} \leftarrow$ Sensor Update (s_t, $x_t^{[n]}$)
6: $X_t \leftarrow X_t + < x_t^{[n]}, w_t^{[n]} >$
7: end for
8: for n=1 to N loop:
9: draw $x_t^{[n]}$ with probability $\propto$ $w_t^{[n]}$
10: $X_t \leftarrow (X_t + x_t)$
11: end for
12: return X_t

The algorithm can be folded into 2 steps: motion movement and sensor update and resampling process. In this algorithm the robot's pose is represented by a belief factor (X_t). Initially, all the particles have equal weight (w_t) and the belief state are estimated by randomly generating N particles. The Algorithm takes into consideration of the previous belief state values X_{t-1}, the movement command V_t, and the sensor measurement S_t as the initial starting input.

In the first iteration of the AMCL algorithm, a hypothetical state is calculated as the initial position of the robot. As the robot's position is updated the laser scanner sends different and updated measurements to the ROS nodes. Using these measurements, the particles' weight is recalculated. The result of this iteration is added to the previous belief state and thus a new belief state which corresponds to the location of the robot is obtained.

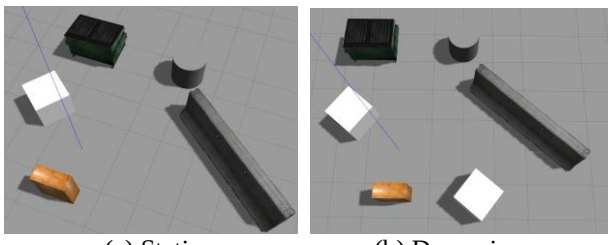
In the other part of the algorithm, resampling of the particles is done, as each belief state has a weight value associated with it, after each resampling iteration the particles that have a high weight value survive and are then used in the next iteration while those particles with a low weight value perish during the resampling. If this continues the number of particles in the filter will reduce drastically after a few iterations which can give inaccurate localization results. To avoid this, the particles with large weights are resampled into many particles with equal weights by restoring the number of particles in the filter. Finally, the algorithm estimates the position of the robot by calculating the belief state with the new sensor measurements.

4. SIMULATION STUDIES

4.1 Simulation environment

To check the functioning of the AMCL algorithm, the simulations using a differential drive mobile robot model were done in Gazebo and RVIZ. The navigation stack in ROS was used. To create a mobile robot, a Unified Robot Description Format (URDF) file is made. The URDF file defines the shape, color, origin, inertial and collision properties of the robot.

The big advantage of ROS is the prebuilt plugins for the sensors. A simple camera sensor and a Hokuyo laser finder can be added to the robot model. Two cases of simulations have been carried out in two different environments: static and dynamic environment.



(a) Static (b) Dynamic

Fig. 1. Gazebo environment

In the static environment, obstacles are fixed while the robot is moving as shown in Fig.1 (a). In the dynamic environment, meanwhile, one obstacle is added to the static environment while the robot is moving as shown in Fig.1 (b). This is aimed to see the dynamic path planning to avoid the collision with obstacles in real time.

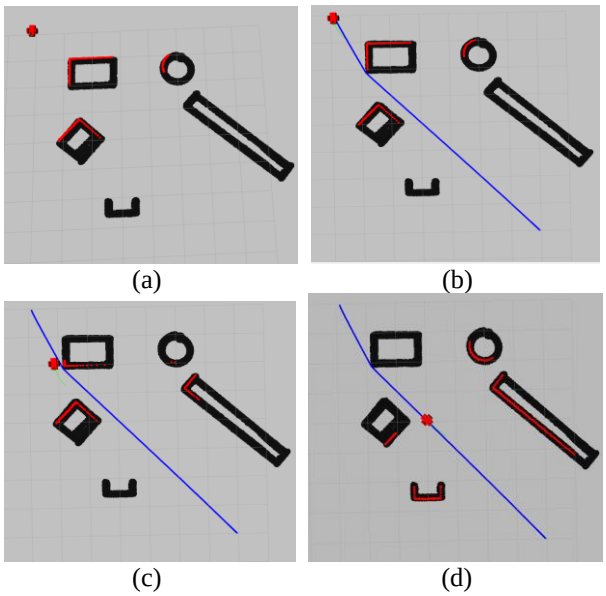
4.2 Simulation results

The ROS navigation stack generates the global and local path for the robot to follow. The robot will be able to follow and reach the goal only if its localization in the map is accurate. We can simulate two different cases of the path generated into two different scenarios.

CASE I: Static environment

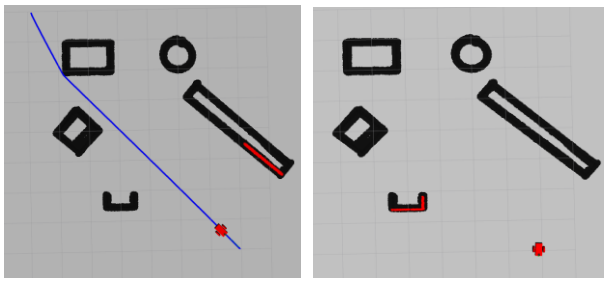
When the whole map is stationary, there is no movement of obstacles. In this case the global path that is formed will not change and the local path will follow the global path quite closely.

Fig. 2 shows the navigation in the static environment. Fig. 2 (a) shows the initial position, (b) goal point given to the bot. The robot now follows the global path. Fig. 2 (c), (d), and (e) show that the global planner forms the shortest trajectory from the initial point to the goal using the static map. While cornering sometimes it comes close to the obstacle, then the local planner deviates from the global path to avoid it. After avoiding the obstacle, the robot again returns to follow the global path, as now the local and global path coincide. The robot managed to reach the goal point every time as shown in Fig. 2 (f).



(a) (b)

(c) (d)

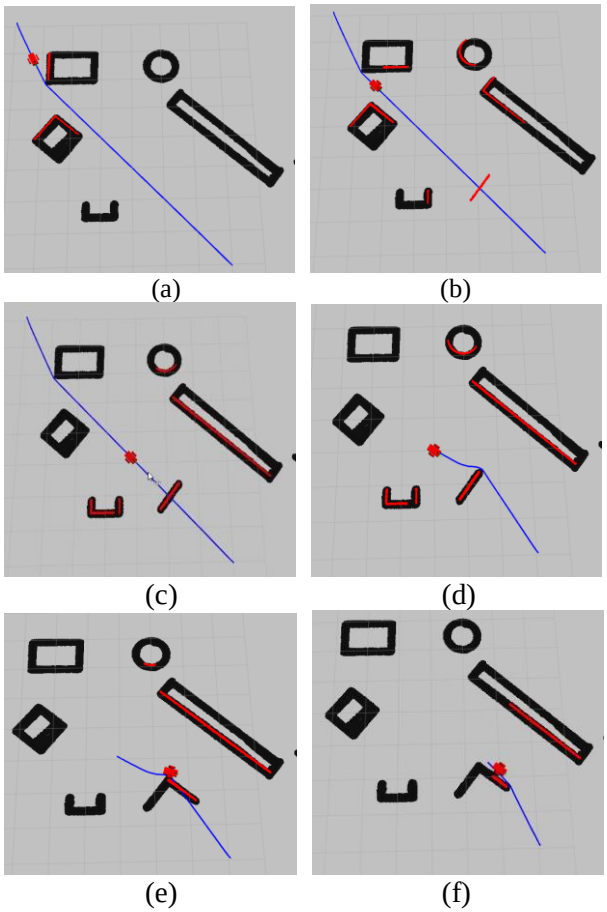


(e) (f)

Fig. 2. Static environment

CASE II: Dynamic environment

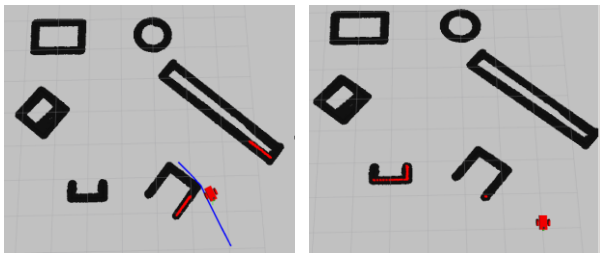
This is the case when we suddenly introduce an obstacle in the path of the robot while the robot is moving. The aim of this experiment is to localize the robot in the environment with the changes. Simulations were carried out to see if the localization of the robot was accurate enough to guide itself from the obstacle which is not present on the global map. The result shows that the path planning algorithm now calculates an alternative global path and again the local path follows the global path as shown in Fig. 3



(a) (b)

(c) (d)

(e) (f)



(g) (h)

Fig. 3. Dynamic environment

In Fig.3, each plot is described as follows:

(a) Goal point given to the bot. As the global planner uses the static map to form a trajectory to the goal position.

(b) The lidar sensor detects an obstacle at some distance. Map update even as the robot moves is taking place as the area that was free for the robot will now be changed to occupied area where the robot cannot go.

(c) The new obstacle's cost has been updated in the map. The global path will now receive messages simultaneously as the map is being updated using the current local map. The global planner will now again form the shortest path to the goal position by avoiding all the obstacles. The map and the path are simultaneously being updated

(d) After receiving messages about the map update, the global path recalculates the shortest path to the goal from the current position of the robot avoiding all obstacles. The global Path has now changed from the original one.

(e) The robot now follows the new path generated. But again, the new path goes through an obstacle. So, the map and the global path are updated again to account for this change.

(f) A new Global path is generated to avoid the obstacle and simultaneously the map is also updated. The robot now follows this path.

(g,h) The bot now easily follows the path to reach its goal.

5. CONCLUSIONS

Adaptive Monte Carlo Localization is a very effective solution for localizing the robot in a given environment. The robot was successfully localized in the ROS Gazebo and Rviz environment. The results clearly show that the particle filter were successfully able to converge quickly and were able to give the exact pose and orientation of the mobile robot.

Because of the effective implementation of AMCL the robot was able to reach a goal in the map within a specific time interval. Moreover, the robot was able to navigate in the map while performing AMCL algorithm in real time, thus the robot was able to localize itself and find a way around the obstacle even with changes in the static map. This algorithm along with all the ROS packages demonstrates that it is very much viable to realize autonomous navigation on mobile robots using the available sources.

ACKNOWLEDGMENT

This work was partially supported by Indo-Korea JNC program of National Research Foundation of Korea (NRF-2017K1A3A1A68072072).

REFERENCES

[1] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Trans. Robotics and Automation*, vol. 23, no. 1, pp. 34-46, 2007.

[2] M. P. Imthiyas, "Indoor Environment Mobile Robot Localization", *International Journal on Computer Science and Engineering*, vol. 2, no.3, pp.714–719, 2010.

[3] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *The international Journal of robotics research*, vol. 22, no. 12, pp.985-1003, 2003.

[4] M. Montemerlo and S. Thrun, "FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics (Springer Tracts in Advanced Robotics)", *Springer-Verlag*, 2007.

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," *International Joint Conference on Artificial Intelligence*. pp. 1151-1156, 2003

[6] Pérez, J.A., Castellanos, J.A., Montiel, J.M.M., Neira, J. and Tardos, J.D., "Continuous mobile robot localization: Vision vs. laser", *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2917-2923, 1999

[7] 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", *ICRA Workshop on Open Source Software*, vol.3, no.2 pp.5, 2009.

[8] Zaman, S., Slany, W. and Steinbauer, G., "ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues", *Saudi International Electronics, Communications and Photonics Conference*, pp. 1-5, 2011

[9] https://www.hokuyo-aut.jp/search/single.php?serial=166.

[10] Das, S., "Robot localization in a mapped environment using Adaptive Monte Carlo algorithm", *https://drive. google. com/open*.