Where am I?

Gabriel Garcia

Abstract—Localization is a key problem for autonomous robotic systems. This work presents an application of Monte Carlo Localization into two different simulated robots. Results proving the efficiency of the proposed methods and the parameters used are presented.

Index Terms—Autonomous robots, localization, Monte Carlo, particle filter.

1 Introduction

Localization for mobile robots is a fundamental problem when we want to apply some level of autonomy into it. Therefore, to develop a robotic system that seeks to autonomously navigate a previously mapped environment it is necessary to use a robust and efficient method for localization.

This paper presents a system for autonomous navigation of two different mobile robots, both wheeled platforms restricted by non-holonomic constraints. Localization is performed through Monte Carlo Localization (MCL) method and the Trajectory Rollout algorithm is responsible for path planning.

2 Background

Localization can be defined as a version of on-line temporal state estimation, where a mobile platform should estimate its position with respect to a global coordinate frame, [1]. It has been referred as the most fundamental problem to provide a mobile robot with autonomous capabilities, [2].

There are different flavors for the localization problem. The most simple one is position tracking, where the initial robot pose in known and then, new poses are estimated based on robot's odometry compensated by some method. Another version of the localization problem is the global localization problem, in which the robot does not know its initial pose and should determine it from scratch. An even more complicated flavor of that problem is the kidnapped robot, where a well-localized robot is teleported to some other place without any information of that translation.

For this work, it is considered a robot that knows its initial pose and it should track the position based on its odometry and a Lidar. So, the problem is classified as position tracking. Different methods proposed in the literature, such as Markov Localization [3], Kalman Filtering [4] and Monte Carlo Localization [2], can be applied to solve that problem.

Thrun et al. [2] describe that the key idea of MCL is to represent the belief by a set of samples (also called: particles), drawn according to the posterior distribution over robot poses. In other words, rather than approximating posteriors in parametric form, as is the case for Kalman filter and Markov localization algorithms, MCL represents the posteriors by a random collection of weighted particles which approximates the desired distribution.

3 Model Configuration

Two robots were used to test the proposed method. The first one was provided by udacity and the other one was the SpeleoRobot. This section will describe each one and show the parameters used to tune both planner and localization methods.

3.1 UdacityBot

The UdacityBot is a two-wheeled robot that is embedded with a Hokuyo Lidar and an RGB camera. The robot is formed by a box (40cm x 20cm x 10cm), two wheels (radius of 10cm and 5 cm of length), and two spherical ball casters (radius of 5 cm). The Lidar is located between its middle and its front in a height of 5 cm from its top. That position maximizes Lidar performance. The camera is located in robot's front. Figure 2 shows a model of the UdacityBot. The robot is driven by a differential controller.

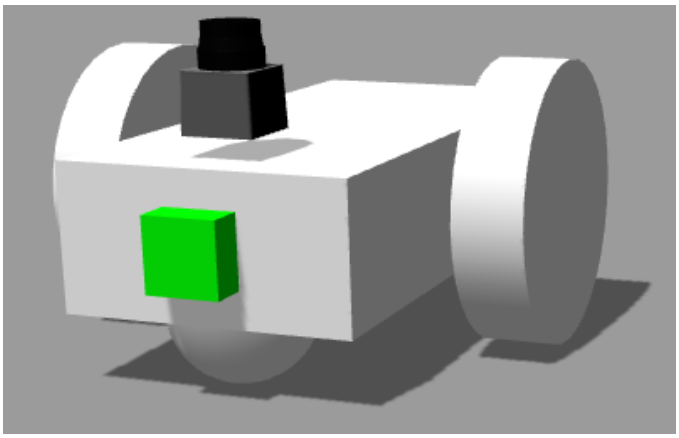

Fig. 1. UdacityBot model.

The parameters presented in Table 1 were used.

TABLE 1
My caption

Set	Parameter	Value	Reason for selecting
AMCL	min_particles	500	Based on turtle bot example.
	max_particles	2000	
	kld_err	0.05	
	kld_z	0.99	
	odom_alpha1 ... 4	0.05	Lower than default since we are in a simulated environment, then, the odometry is supposed to be accurate.
	laser_max_beams	40	Set to match laser scan frequency.
	laser_z_hit	0.5	Values for Hokuyo laser.
	laser_z_short	0.05	
	laser_z_max	0.05	
	laser_z_rand	0.5	
	laser_sigma_hit	0.2	
	laser_lambda_short	0.1	
	laser_model_type	likelihood_field	
	transform_tolerance	1.0	Set to avoid synchronization problems.
base_local_planner	max_vel_x	2.0	Set to increase robots velocity.
	robot_radius	0.33	An approximate size of the robot.
	holonomic_robot	false	Since the robot is non-holonomic.
local_costmap_params	width	7.0	The size of the considered map to plan the path. Bigger can slow down the system, and slower can difficult to find a path.
	height	7.0	
	update_frequency	20.0	Set to improve computational planning performance.
	publish_frequency	10.0	
global_costmap_params	update_frequency	10.0	
	publish_frequency	5.0	
costmap_common_params	transform_tolerance	0.1	Set to avoid synchronization problems.
	inflation_radius	0.5	Used to keep a safe distance from obstacles.
	robot_radius	0.33	An approximate size of the robot.

3.2 SpeleoRobot

The SpeleoRobot, shown in Figure 2, is capable of inspecting underground locations adjacent to existing mining operations. The robot is currently in test phase and can detect the threat of falling rocks or the presence of animals, for example. The robot is equipped with a camera and lighting system that affords a real-time view of the underground environment, enabling the operator to conduct simultaneous inspections or surveys for future access, thereby reducing the risks inherent to speleological activities.

Fig. 2. Robot in field test.

For this project a simulated version of the SpeleoRobot was used. It was added with a Hokuyo Lidar located in such a way that the wheels will not interfere in measurements. The robot's body has a box format (54 cm x 25 cm x 12 cm), and it has six wheels of 14 cm radius. Figure 3 presents the model used to represent the SpeleoRobot (Christmas Version). The robot uses a six wheels Skid Steer controller.

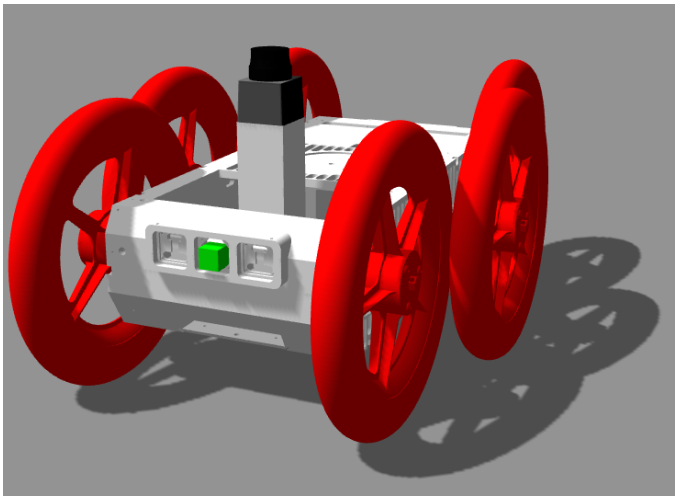

Fig. 3. SpeleoRobot model.

Since the UdacityBot and the SpeleoRobot have similar sensors and formats, the parameters from Table 1 were used for both.

4 Results

The proposed method was implemented in the well know Robot Operating System (ROS) and the Gazebo was used to simulate a simple maze. The robot should be able to move itself to an established target marked in the figures as an arrow. Figures 4 and 5 shows the UdacityBot moving to target position. And Figures 6 and 7 presents the results for the SpeleoRobot.

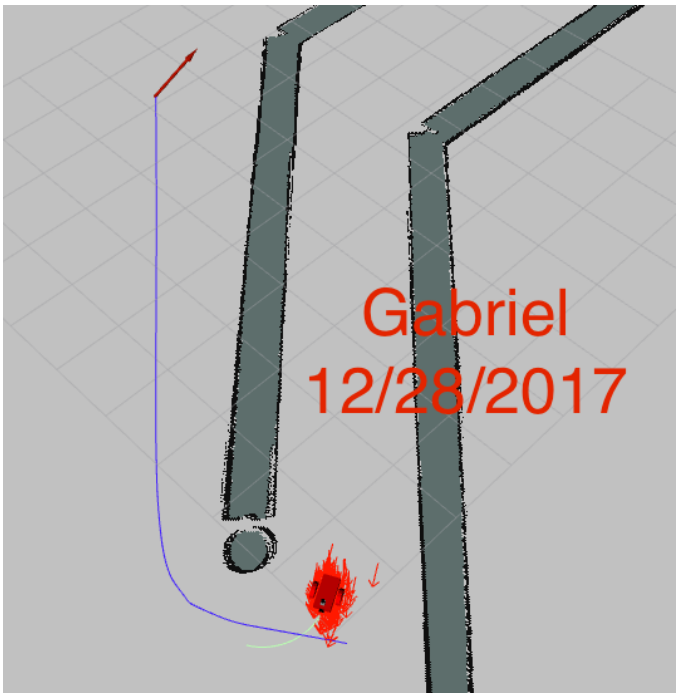

Fig. 4. UdacityBot moving to target.

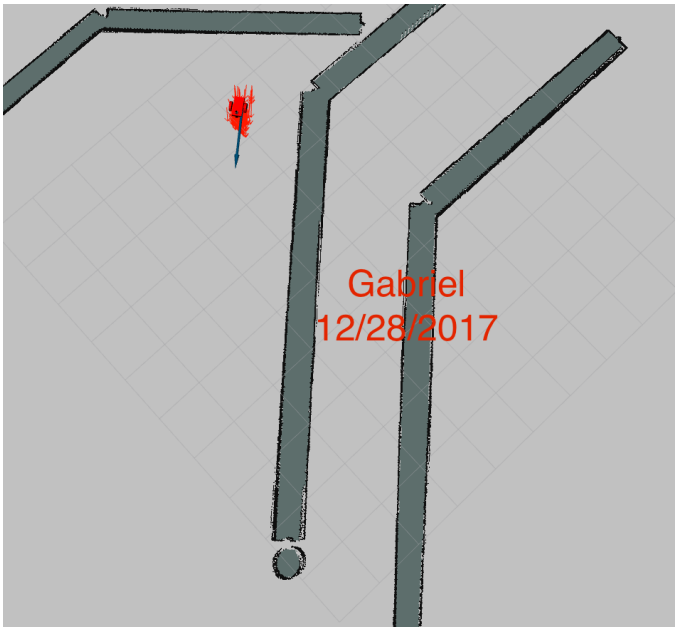

Fig. 5. UdacityBot reaching target.

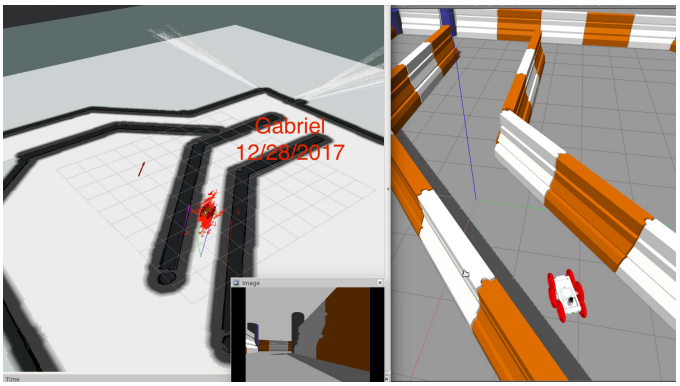

Fig. 6. SpeleoRobot moving to target.

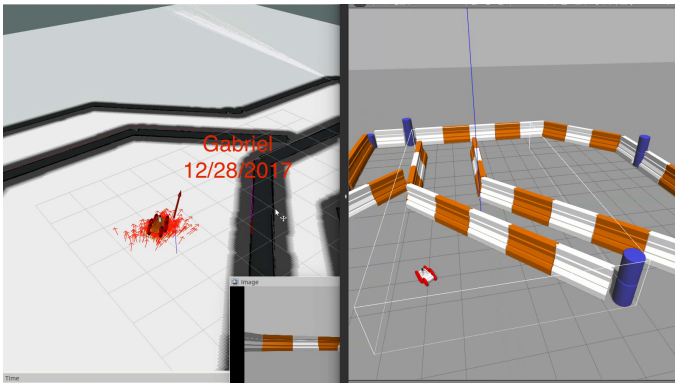

Fig. 7. SpeleoRobot reaching target.

5 Discussion

The main task of this project was to use Monte Carlo localization and trajectory planner with feasible parameters to allow a robotics system to navigate through an environment avoiding beating obstacles. The results presented in Section 4 shows both robots successfully completing the task. To achieve that, a lot of different parameters were tested. UdacityBot takes too much time (2 minutes) to reach desired target and it does a lot of turns. That can be related with its controller. EspeleoRobot performs a better path following, achieving target in approximately 20 seconds. The images show a bunch of red arrows around the robot that represents the particles from MCL. In both tests we can see that the arrows are near to the robot, proving the efficiency of the proposed method.

6 Future work

Future work should be focused on improving the controller for UdacityBot and embedding the method into the robot. To implement the code into the robots the Jetson TX2 could be used.

The kidnapped robot problem is not covered by the proposed method, since it does not add new random poses eventually. This can be implemented by modifying two parameters from amcl package (recovery-alpha-slow and recovery-alpha-fast).

References

[1] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.

[2] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[3] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.

[4] S. I. Roumeliotis and G. A. Bekey, "Bayesian estimation and kalman filtering: A unified framework for mobile robot localization," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 3, pp. 2985–2992, IEEE, 2000.