Where Am I? Project Writeup

Gang Wang

Abstract—In this work, we set out to build a mobile robot in simulation and try to understand how to apply a proper localization algorithm to it, such that our mobile robot navigates to its desired goal destination successfully.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Localization.

1 Introduction

LOCALIZATION is the most fundamental task for a mobile robot to navigate through its running environment to a goal destination autonomously. For it expresses the robot's local pose with respect to the global map of the environment. The more certain its location is, the more confident for it to navigate through its running environment.

In this project, we first created a mobile robot running in Gazebo and Rviz with a laser rangefinder. We then feed the laser scanned map along with the reading odometer data into AMCL package. The main objective of this project is to tune AMCL algorithm, so that it updates the estimated poses of our mobile robot as accurate as possible. In the end, our mobile robot had the ability to navigate to a predefined goal destination successfully.

2 Background

Depending on the observed sensor data and robot's running environment, the localization problems fall into three categories with an increasing order of difficulty. First is called ***position tracking*** problem, which deals with local uncertainty stems from small sensor noises only since the initial pose is known. Second and the more general one is ***global localization*** problem. It localizes the robot from scratch without knowing its initial pose and does not require the pose errors in unimodal probability distribution. The last one is ***Kidnapped robot problem***, which is a variant of the global localization problem. In practice, whenever a well-localized robot lost its current poses, either by teleported secretly to some other location or the localization algorithm failed, has to deal with this Kidnapped problem.

In this project, we chose Adaptive Monte Carlo Localization (AMCL) algorithm to localize our robot. To reach this conclusion, we first turned to another localization algorithm – the Kalman Filter Localization.

2.1 Kalman Filters

Kalman filter (KF) is a Bayes filter for filtering and prediction in continuous linear Gaussian systems. It's easy to implement and usually runs efficiently, which makes it suitable for localization. However, because KF by its nature assumes input being a collection of linear features in Gaussian distribution, a Kalman filter localization algorithm is more suitable for a mobile robot running in a landmark-based environment.

2.2 Particle Filters

Like KF, Particle filter (PF) is also an implementation of the Bayes filter. The key idea of PF is it is nonparametric, meaning it represents belief by a set of random state samples drawn from posterior. Therefore PF can represent a much broader space of distributions, which makes it works for most localization problems.

2.3 Comparison / Contrast

In this project, our mobile robot is running in a maze-like environment. We do not have any unique identifiable landmarks in it. So a nonparametric algorithm such as Particle filter localization tends to work well.

3 Simulations

3.1 Model Configuration

3.1.1 Model design

The following Table 1 and Table 2 list how we defined our mobile robot in simulation that are not clearly given in the lecture.

Name	Origin	Geometry
chassis	xyz="0 0 0" rpy="0 0 0"	box size="0.4 0.2 0.1"
left_wheel	xyz="0 0 0" rpy="0 1.5707 1.5707"	cylinder radius="0.1" length="0.05"
right_wheel	xyz="0 0 0" rpy="0 1.5707 1.5707"	cylinder radius="0.1" length="0.05"
camera	xyz="0 0 0" rpy="0 0 0"	box size="0.05 0.05 0.05"
hokuyo	xyz="0 0 0" rpy="0 0 0"	box size="0.1 0.1 0.1"

TABLE 1
Gazebo model links

Name	Origin	Parent link	Child link
left_whieel_hinge	xyz="0 0.15 0" rpy="0 0 0"	chassis	left_wheel
right_whieel_hinge	xyz="0 -0.15 0" rpy="0 0 0"	chassis	right_wheel
camera_joint	xyz="0.2 0 0" rpy="0 0 0"	chassis	camera
hokuyo_joint	xyz="0.15 0 0.1" rpy="0 0 0"	chassis	hokuyo

TABLE 2
Bazebo model joints

3.1.2 Packages Used

We used *amcl* and *move_base* packages from ROS Navigation Stack (http://wiki.ros.org/navigation) in this work. Launching this project creates two ROS nodes out of these two package. The amcl node receives the rangefinder sensor topic in order to localize the robot. The resulting estimated pose then feeds into move_base node, along with odometer, navigation goal and map topic, to get a trajectory towards the goal smoothly.

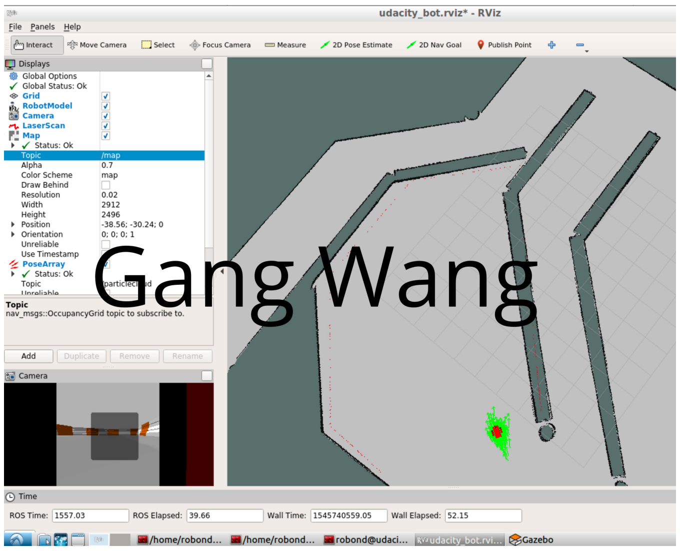
3.1.3 Parameters

Three amcl node parameters are tuned in this project. The minimum allowed number of particles (*min_particles*) is 50, and the maximum (*max_particles*) is 300. The maximum laser scan range ($laser_max_range is set to 4.3 to cause amcl discards datas that lie bey$

As for the move_base node parameters, we ended up with $inflation_radius is 0.35, obstacle_range is 2.0 and raytrace_range is 2.5. In the trial and error parameter tuning process, we found these costmap$

After tuning the AMCL parameters and other system configuration, our robot steer to the goal position immediately. In the beginning, it ran around trying to explore surrounding and as well as localizing itself. Soon it moved smoothly towards the goal and the particles converged as shown in Fig.1.

4.1 Localization Results



udacity_bot.rviz* - RViz

Displays: Global Options; Global Status: Ok; Grid ✓; RobotModel ✓; Camera ✓; LaserScan ✓; Map ✓; Status: Ok

Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	
Resolution	0.02
Width	2912
Height	2496
Position	-38.56; -30.24; 0
Orientation	0; 0; 0; 1
Unreliable	
Use Timestamp	

PoseArray ✓; Status: Ok; Topic /particlecloud

Topic
nav_msgs::OccupancyGrid topic to subscribe to.

ROS Time: 1557.03 ROS Elapsed: 39.66 Wall Time: 1545740559.05 Wall Elapsed: 52.15

Fig. 1. Half-way on the road.

5 Discussion

Although our mobile robot can navigate to the destination successfully, it moves blindly around in the beginning. One might believe this is due to the lack of confidence at that stage. This also suggests if there exists some ways to eliminate as much uncertainty as possible during the first handful of pose estimations.

Since it's easy to emulate a scenario that teleports our robot in Gazebo, we tried kidnapping it to see how the localization recovers. We observed the robot could not always succeed in recovering and reaching destination. It depends on where it was teleported. For example, the robot had a higher chance to recover if it was

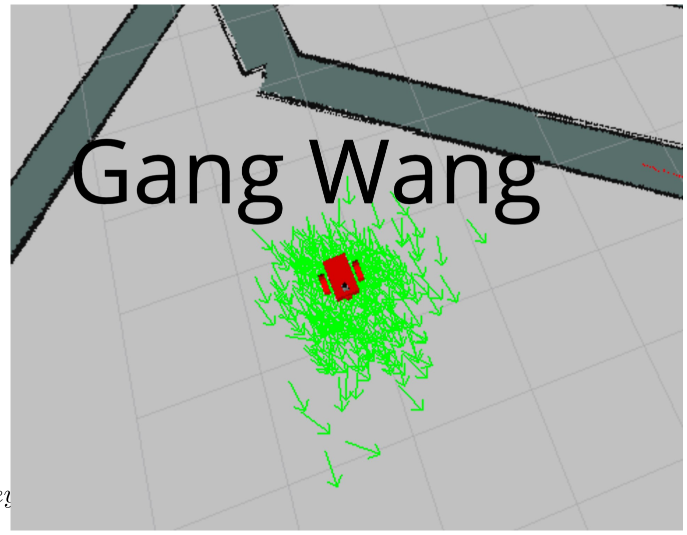


Fig. 2. Reached goal destination.

placed secretly in another corridor. However, being placed in a spacious area where fewer obstacles were detected, it failed to navigate to goal most of the time. Increasing maximum allowed particles improved it a bit, but we did not try it out further. One surmise this kidnapped problem could be improved by increasing maximum laser range as well as allowed particle samples.

5.0.1 Apply MCL in an industry domain

Aside from the mobile robot like the one in this project, in many robotics applications where estimation of a moving object's pose with respect to an arbitrary frame is involved, MCL could find its place in solving the problem. For example, an application that a robot arm is commanded to pick up objects from a conveyor belt. It's essential to localize the object as accurately as possible in order to grab hold it.

6 Conclusion / Future work

In this work, we built a robot that equipped with a laser rangefinder on its head. Meaning it senses obstacles ahead in an angled viewing area. One might think to replace it with a wider angled rangefinder, or even better, a 360-degree one. Because from the observed local costmap updates by tuning minimum and maximum laser range, more measures from both sides helped in generating trajectory, which in turn improves the localization result.

It's one thing our mobile robot accomplishes moving to goal in simulation, it's quite another if this whole project were to be deployed in a real robot. For we have a nearly perfect laser rangefinder in simulation, and the static environment surely is static. Given the opportunity, it's tempting to further pursuit this challenge that deploys amcl on a real application.

References