Where am I? Robot Localization

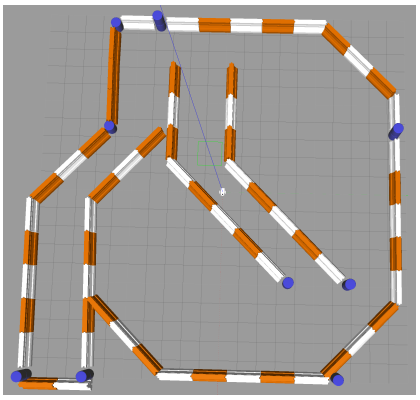
Hassan Sirelkahtim

Abstract—We build a basic robot with two actuators (wheels), a rectangular base, and a camera and laser sensor (hokuyo). Using the amcl package and differential drive controller, we are able to localize the robot and move the robot to the desired location. We then modify the robot base by using a cylindrical base instead of a rectangular one. We then repeat the exercise of localizing the robot and move it to the direction desired and achieve success. Although, it was more difficult for the modified robot to reach the destination - It bumped into obstacles and rotated a few times to get a bearing.

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

1 Introduction

It is a pretty trivial exercise for a human being to reach a desired destination within a given short range environment. We might need to go back to the time when we were infants to get an idea of how difficult the task becomes when we want a robot to do the same. In this paper, we attempt to localize a robot in a relatively simple environment, which you can see below.



We use a Adaptive Monte Carlo Localization (AMCL) package that helps us with the localization of the robot. In order to test how effective the localization is we use a C++ script to move the robot to a destination. We then try to replicate our results in modified robot, that has a change in the base structure (cylindrical, instead of rectangular) and attempt the same exercise of moving the robot to a certain location. If the localization is successful, the robot should know where it is and how far it is from obstacles and be able to navigate a certain terrain and thus be able to move to the right destination. Moreover, the robot should therefore also plan it's trajectory. We use the move_base package for this.

2 Background

In this project we use particle filters to localize the robot. Each particle around the robot has a x and y coordinate as well as an angle (θ) to give the orientation of the robot. Each particle is assigned a weight, which changes according to the sensor measurement. More details will be explained below. We end up using adaptive monte carlo localization (amcl) since it more computationally efficient and achieves convergence in fewer steps than the traditional monte carlo localization problem

2.1 Kalman Filters

A kalman filter uses linear quadratic estimators that assume gaussian noise. The main difference between a kalman filter and a particle filter is in the assumptions it makes. Because kalman filters do not use particles, the method generally has more stringent assumptions (gaussian distribution)

2.2 Particle Filters

As we explained above particle filters use particles around the robot to estimate the pose of the robot. Each particle has an x and y coordinate, as well an angle θ to indicate the pose of the particle. Particle filters have two steps: update measurement and re-sampling. Each particle is assigned a weight and every time new sensor measurements are received the particles are re-sampled and assigned new weights. The particles with lower weights do not survive, and thus after a few iterations the particles are able to estimate the pose of the robot.

2.3 Comparison / Contrast

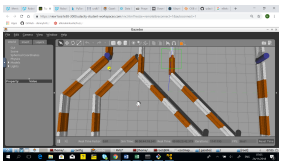
The benefits of using kalman filters is that it is generally computationally faster. However, this speed comes with the drawback of being inflexible. Particular filters are slower but doesn't assume a gaussian distribution.

3 Simulations

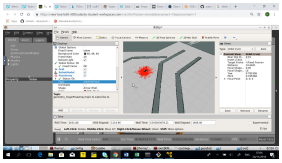
In order to simulate the robot we used Ros alongside gazebo and rviz.

3.1 Achievements

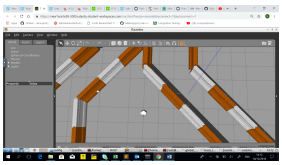
We were able to get the robot to reach the goal that we had set for it on its own traversing the obstacles of the map. Below you see the chart of the robot reaching its destination in both gazebo and rviz



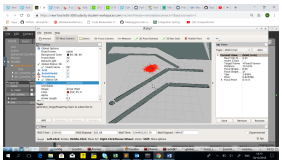
And here you can see the udacity bot in the rviz environment



After customizing my robot we were also able to make this robot reach the intended destination. Here you can see the customized robot reach its destination in the gazebo environment



Here you can see the customized robot reach its destination in the rviz environment



In this project, we created two robot designs. The difference between them lay in the base shape of the robot. The Udacity bot had a rectangular base, whilst the customized robot had a cylindrical shape with a radius of 0.2 and a length of 0.1. Both models had two wheels with a radius of 0.1 and a length of 0.05. Both had hokuyo sensors (laser sensors) hooked up on top of the cylindrical/rectangular base. Both models were equipped with camers as well.

3.1.1 Packages Used

In order to run this simulation many packages were used. For moving the robot we used the differential drive controller from gazebo. We also used a package for localizing the robot: the amcl package. We also used the move base package to help with navigation to the intended goal.

3.1.2 Parameters

Parameter	myrobot	udacity_file	File
map_type	costmap		costmap_common_
obstacle_range	2.5	5.0	costmap_common_
raytrace_range	3	8.0	costmap_common_
transform_tolerance	0.3	0.3	costmap_common_
inflation_radius	0.1	0.7	costmap_common_
robot_radius	0.2		
observation_sources:	laser_scan_sensor		costmap_common_
global_frame	odom	odom	local_costmap_par
robot_base_frame	robot_footprint	robot_footprint	local_costmap_par
update_frequency	5.0	5.0	local_costmap_par
publish_frequency	5.0	5.0	local_costmap_par
width:	5.0	5.0	local_costmap_par
height	5.0	5.0	local_costmap_par
resolution	0.05	0.05	local_costmap_par
static_map	false	false	local_costmap_par
rolling_window	true	true	local_costmap_par
global_frame	odom	odom	global_costmap_pa
robot_base_frame	robot_footprint	robot_footprint	global_costmap_pa
update_frequency	5.0	5.0	global_costmap_pa
publish_frequency	5.0	5.0	global_costmap_pa
width:	40	40	global_costmap_pa
height	40	40	global_costmap_pa
resolution	0.05	0.05	global_costmap_pa
static_map	false	false	global_costmap_pa

4 Results

Both robots managed to reach the desired destination. However the customized robot took longer time to reach the destination and bumped into more obstacles, whereas the Udacity bot, in general, was able to get to the destination in less time and bumping into less obstacles.

5 Discussion

This is the only section of the report where you may include your opinion. However, make sure your opinion is based on facts. If your robot performed poorly, make mention of what may be the underlying issues. If the robot runs well, which aspects contribute to that? Again, avoid writing in the first person (i.e. Do not use words like ”I” or ”me”). If you really find yourself struggling to avoid the word ”I” or ”me”; sometimes, this can be avoid with the use of the word one. As an example: instead of : ”I think the robot cannot localize itself because the sensor does not provide enough information for localization” try: ”one may believe the localization performance is poor because the sensor layout is not able to provide enough information for localization”. They say the same thing, but the second avoids the first person.

6 Conclusion / Future work

In general the localization of the robot was a success since both robots reached their intended destination. We might want to increase the complexity of the environment and see how well our robots would do there. We could also try to benchmark the robot’s performance against the Kalman filter algorithm. Another interesting addition to the models, would be if we increased the number of wheels to 4 and 6. How would that change the performance of the robot. We could also experiment with camera positions and different types of sensors.