Where Am I? Localization in a Mapped Environment

Harrison Seung

Abstract—This report provides an application case for developing a ROS package, creating a mobile robot model in Gazebo, and integrating the Advanced Monte Carlo Localization (AMCL) and Navigation ROS packages for localizing a robot in a provided mapped environment. The process is first implemented using a standard mobile robot designed by Udacity and then on a customized mobile robot. Rationale for the selection of each implementation's ROS package parameters is also discussed in order to reach the designated goal.

Index Terms—Robot, Udacity, AMCL, EKF, ROS, Particle Filters, Localization.

1 Introduction

A key component in mobile robot navigation is determining it's position and orientation for use in path planning. For a given mapped environment, the challenge of determining a robot's position is called Localization. This is achieved using a probabilistic algorithm to filter noisy sensor data to track position and orientation. To demonstrate the usage of the Advanced Monte Carlo Localization (AMCL) algorithm, the following report will detail the process of building a mobile robot in simulation (Gazebo) and developing a ROS package with AMCL and Navigation ROS packages to localize a robot given a mapped environment.

2 Background / Formulation

Localization is critical for a robot's navigation. Without accurate data of where a robot's pose is within an environment, the robot will be unable to navigate around the environment to perform desired tasks.

2.1 Localization Problems

When addressing localization problems, they are typically described in three types of categories: Position Tracking, Global Localization, and Kidnapped Robot.

2.1.1 Position Tracking

Position Tracking or Local Localization is considered the simplest of the three. In this problem, the robot's initial pose is known with respect to the map and the uncertainty is limited to regions around the robot.

2.1.2 Global Localization

In Global Localization, a robot's initial pose with respect to the map is unknown. In this problem, a robot must determine it's position relative to the ground truth map. The uncertainty is much greater than position tracking.

2.1.3 Kidnapped Robot

The Kidnapped Robot problem is considered the worst case scenario. This problem considers the instance where the robot could be picked up and re-positioned anywhere else in the map. The robot must be able to recover it's position and orientation with respect to the map.

2.2 Localization Algorithms

To tackle these localization problems, two of the most common localization algorithms used are the Extended Kalman Filter (EKF) and the Monte Carlo Localization (MCL) particle filters.

2.2.1 Extended Kalman Filter

The Extended Kalman Filter is a nonlinearized version of the Kalman Filter (KF). The first step of localization using a Kalman Filter is making an initial estimate of the robot's pose, taking into account the uncertainty of the robot's sensors and movements. This uncertainty is modeled as an uni-modal Gaussian distribution. Next, the robot's pose is measured using available sensors and another pose estimate, based on the sensor measurements, is generated. Lastly, using the initial pose estimate (prior) and the sensor pose estimate, a new pose estimate is calculated. This new estimate is called the posterior and will be located within the intersection of the prior and sensor estimate with a more accurate Gaussian distribution. For nonlinear systems, the mean of the Gaussian can be calculated using a nonlinear function, f(x), but the variance cannot. To solve this, the EKF uses a 2nd order Taylor Series function to linearize f(x) over a small section centered around the mean. Overall, the EKF is useful as it can quickly estimate a robot's pose using only a few measurements by fusing data from several sensors as long as the data can be modeled as a Gaussian distribution.

2.2.2 Monte Carlo Localization

In MCL, localization occurs in two phases. First, particles are randomly placed throughout a mapped environment with each pose represented by an x, y, and theta coordinate. These particles each represent a hypothesis of where the robot might be. Along with the pose coordinates, each particle is assigned a weight factor which correlates to the particles distance from the robot's predicted pose, determined by sensor measurements. The larger the weight, the closer the particle is to the robot's predicted pose. In the second phase, the particles are re-sampled using a recursive Bayes Filter approach to create a new particle set with particles of greater weight more likely to survive into the new set than those with smaller weights. As this process is iterated, particles more accurately representing the robot's pose will survive and the robot's pose is determined.

2.2.3 *EKF vs MCL*

In Robotics, MCL is preferred over the EKF algorithm as it presents several advantages. In terms of dealing with measurement noise, The Extended Kalman Filter requires the input measurement to be a uni-modal Gaussian distribution which may not correlate to all real world applications. MCL can take any type of distribution as inputs providing greater versatility. MCL is also simpler to implement in terms of programming. Additionally, MCL allows greater control of computational memory and resolution by changing the number of particles distributed uniformly and randomly across the map.

3 Results

3.1 Udacity Bot

A simple mobile robot was developed using the instructions provided in the lesson. This robot consisted of a rectangular base with front and back casters, two wheels, a camera, and hokuyo laser range finder sensor.

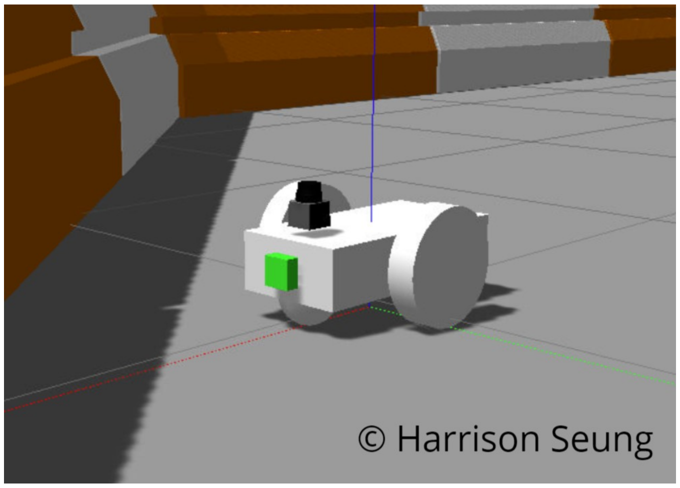

Fig. 1. Model - Udacity Bot

After some parameter tuning the robot was able to successfully reach the goal using the provided navigation.cpp script.

3.2 Custom Bot

A custom robot was developed consisting of a circular base, two wheels, a camera, hokuyo laser range finder sensor, and robotic arms atop the base to simulate a humanoid helper robot.

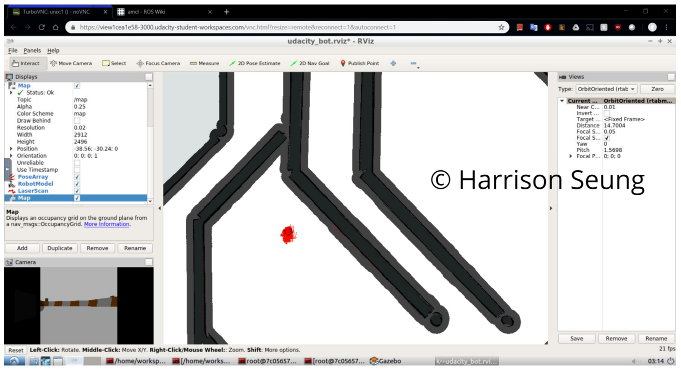

Fig. 2. RViz Goal - Udacity Bot

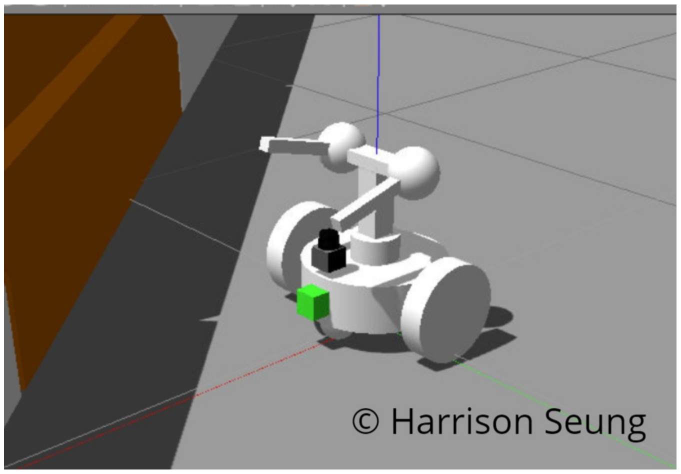

Fig. 3. Model - Custom Udacity Bot

After some modifications to the simple robot parameters to accommodate for the geometry, the custom robot was also able to successfully reach the designated goal.

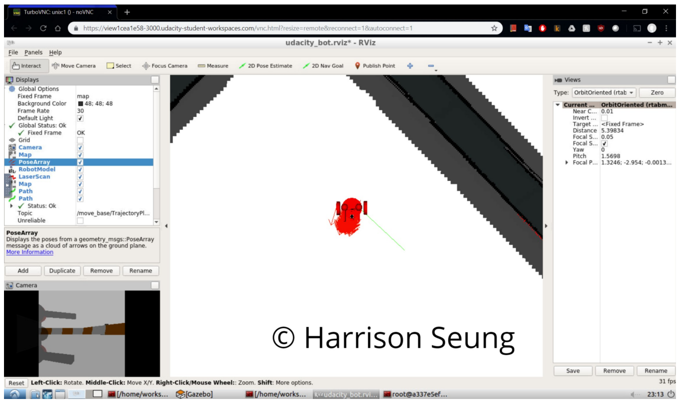

Fig. 4. RViz Goal - Custom Udacity Bot

4 Model Configuration

The robot uses the "move base" package to navigate to the goal position and the "amcl" package to localize the robots pose. Each of these packages has a set of parameters that can be tuned for the robot to improve the performance and accuracy of movement. The effect of each parameter is evaluated using the ”PoseArray” option in RVIZ which represents each particle of the AMCL algorithm as arrows around the robot.

4.1 Udacity Bot Parameters

Before tuning the robot for movement or localization, a few parameters were added for stability issues. In addition to the ”transform tolerance” per the lesson, the ”controller frequency=10” and ”meter scoring=true” were added to eliminate error messages when launching the simulation. For the global and local costmaps, the update frequency was set to 5.0 Hz and the publish frequency to 2.0 Hz, ensuring visualization of the map is always updated before the map. For navigation purposes, an inflation radius of 0.3 was chosen as this enclosed half of the width of the Udacity Bot to prevent collisions with walls. An obstacle and raytrace range to allow sufficient reaction time of the robot before colliding with an obstacle. In terms of localization, the laser range was set to the minimum distance for the range finder to the designated obstacle range. Additionally the laser beam count was increased from 30 to 60 to improve the fidelity of the range finder. The number of particles was set to 5000 as lower particles caused less accurate adherence to global trajectory.

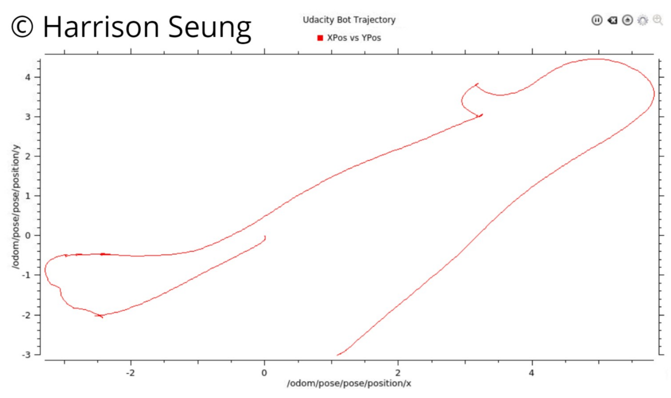

Fig. 5. Udacity Bot Trajectory 5000 particles

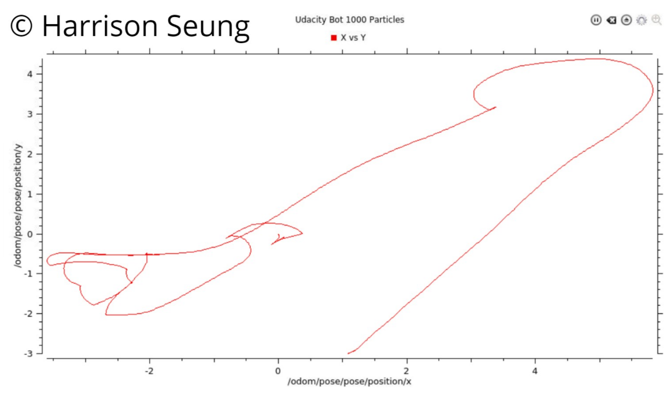

Fig. 6. Udacity Bot Trajectory 1000 particles

4.2 Custom Bot Parameters

For the custom robot, ”pdist scale=0.4” and ”gdist scale=0.8” were added to the base local planner. ”pdist scale” applies weighting for how much the robot controller should stay close to the given path while ”gdist scale” applies weighting for how much the controller should stay towards the local goal. The higher the weight, the higher the cost of a calculated path and the least likely the robot will move towards that direction. A higher ”gdist scale” will push the robot towards staying on the global trajectory. Additionally, the inflation radius to 0.1 to encompass half the width of the robots circular base. All other parameters were kept the same as the Udacity Bot and was sufficient for the custom robot to reach the designated goal.

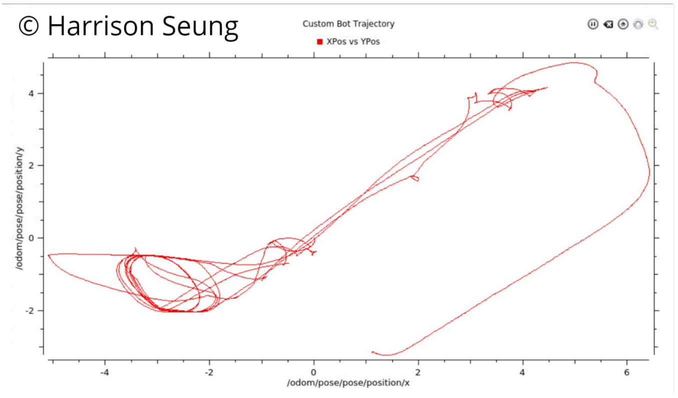

Fig. 7. Custom Bot Trajectory

5 Discussion

The performance of the Udacity Bot was as expected. The robot was able to successfully navigate its way to the designated goal. For the Custom Bot, although the robot was able to reach the designated goal, the duration to complete the task was about 5x longer than the Udacity Bot. This could be due to the larger and asymmetrical geometry of the custom bot causing difficulty of the laser to determine it's location. Both robots would have difficulty with the kidnapped robot problem as AMCL requires measurement sensor data to correlate with particle data. If robot were moved outside the max range of the robot's laser sensor into a position with no available particles, the robot would not be able to recover. With this in mind, possible industrial applications for using a MCL/AMCL based localization method would be in static enclosed environments such as storage warehouses or offices.

6 Future Work

In future work, the custom robot could be improved by properly assigning moments of inertia based on the geometry configuration. This would improve the overall movement of the robot. To improve accuracy of localization, additional sensors could be placed behind the robot to determine a better picture of its surroundings. A majority of the time, the robot appeared to be circling in position to determine the best path forward often rebuilding the local map around it.