Where Am I?

Andrew Truong

Abstract—The project goal was to implement the adaptive monte carlo localization (amcl) algorithm in simulation programs, Gazebo and RViz. The amcl algorithm uses particles to predict the robot's pose and orientation. Adaptive monte carlo algorthm differs from the basic monte carlo algorithm by adapting the number of particle prediction for each forecast. The project also reviewed URDF code for the setup and simulation of a robot. The robot used in this project has a chassis with two cylindrical wheels and two caster wheels for stability. The robot is placed in a maze and the goal of the project is to use amcl to accurately navigate the maze to the goal destination. Several amcl parameters were tuned in the algorithm for precise particle forecasting. The project is complete when the robot reaches the goal destination with an accurate pose array. A new robot designed with new amcl parameters was created to compare the amcl parameters with the initial robot design.

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

1 Introduction

THE adaptive monte carlo algorithm predicts the robots pose by using predictive particles. Each particle has a position, direction, and a weight associated with it. The particle weight is used to determine the likelihood that the particle correctly predicts the robot's pose. The monte carlo particles are implemented in ROS by activating the amcl node. As the parameters of the amcl node are tuned, the particle's are distributed closer to the actual robot's pose. The robot has two sensors a laser sensor and a camera sensor. The laser sensor detects obstacles in the environment and maps the localize environment in RViz. Laser sensor parameters can also be tuned to avoid obstacles in the map. The monte carlo particles can be displayed in RViz using the pose array function. In the image below, the initial pose array is represented by green arrows near the robot. Since the algorithm is initially using default parameters, the monte carlo particles are scattered and inaccurate. Parameter tuning for the adaptive monte carlo algorithm as well as the laser sensors were done to improve the accuracy of the simulation.

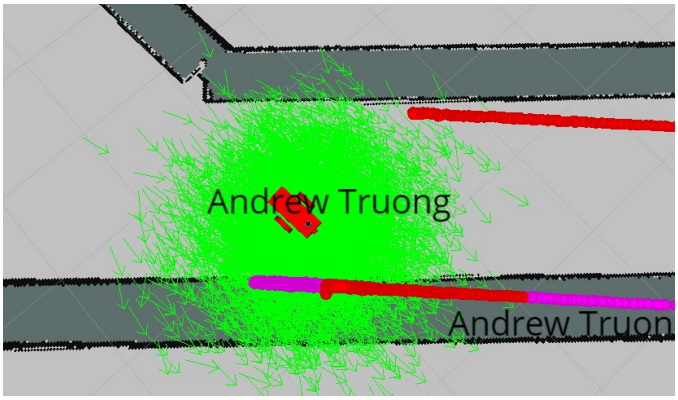

Fig. 1. Initial Pose Array

2 Background

The adaptive monte carlo algorithm was used instead of the extended kalman filter, because amcl has the ability to adjust memory and resolution and map the robot in non gaussian environments. Amcl is widely used in robotics, so the ease of implementation in C++ is greater than EKF. The amcl package in ROS is divided into three filters: overall, laser, and odometry. The overall filter adjusts parameters such as the number of particles and the update frequency. The laser filter parameters adjusts the laser sensor being used. Odometry adjusts the expected noise from actuation. Since the robot is simulated in the gazebo program, there is no expected noise and the odometry paramters do not need to be adjusted. A full list of parameters can be found at: http://wiki.ros.org/amcl

2.1 Kalman Filters

The Kalman filter is an algorithm that can make state predictions given data not limited to robot pose. The Kalmann filter is used in robot localization, because of its ability to filter out noise. The linear Kalmann filter can only approximate linear data, so robot motion that is nonlinear cannot be estimated. The Extended Kalmann filter (EKF) is used to estimate non linear data by approximating non linear functions with linear functions at the mean of the non linear function. The covariance and the mean of the function is used to estimate non linear functions.

2.2 Particle Filters

A particle filter is used to determine to accuracy of each predicted particle. Adjusting the number of particles and the frequency of updates can adjust the memory and resolution of the amcl algorithm. Since, the particle filter uses a pose array that is scattered throughout the map, nonlinear robotic motion can be estimated.

2.3 Comparison / Contrast

Particle filters are widely used in the robotics industry and have a robust algorithm for implementation. The particle

filter is able to characterize non linear data and adjust memory and resolution of the algorithm. The Kalmann Filter is an algorithm that relies on the gaussian function to handle non linear applications.

- EKF characterizes noise as a Gaussian
- EKF cannot localize globally
- MCL has memory and resolution control
- MCL uses particles to filter noise and estimate the robot's pose

3 Simulations

Parameters were adjusted to improve the accuracy of the robot localization. Map parameters were adjusted to tune to mapped environment locally and globally. Parameters that affected the results of the robot localization were the inflation radius, update and publish frequency, and the local map width and height. The amcl parameters were adjusted to tune the particles and laser sensor. The parameters that affected the robot results were laser z hit, laser z rand, laser max beams, update min d, update min a, and max particles.

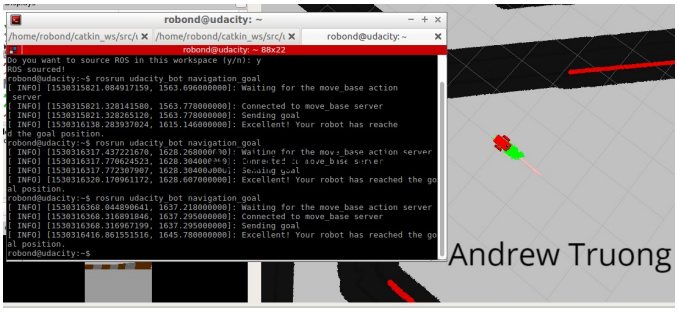

Fig. 2. Pose Array at Navigation Goal

3.1 Achievements

Given the default parameter setting for the simulation, the robot model was not able to correctly move towards the goal. Changing the parameters in the local costmap yaml file improved the robotic motion and the robot was then able to move towards the goal. The local costmap represents the robotic vision. Decreasing the width and height parameter on the local costmap improve the distribution of the pose array for the robot simulation.

3.2 Benchmark Model

3.2.1 Model design

The robot design was developed with a xacro file in URDF. The robot links include: robot footprint, chassis, left wheel, right wheel, camera, and hokuyo. The chassis has two caster wheels for stability. The left and right are centered on the chassis and enable differentiation wheel motion.

3.2.2 Packages Used

The ROS packages that were used in the amcl launch file were map server, amcl, and move base. The map server creates the map in Gazebo and RViz. The amcl package links the map and odometry frames and continuously updates the map with the remap and laser scan ROS topics.

3.2.3 Parameters

The amcl parameters that were tuned were laser z hit and laser z rand. These parameters adjust the particle weight. The particle adjusts the probability that the particle accurately estimates the robot's pose. The laser max beams parameter adjusts the laser sensor for accurate laser detection of obstacles. Update min d and update min a are parameters for the update frequency based on the minimum distance and angle movements for an update particle array to occur. The max particle parameter is maximum amount of particle that are generated in the array. The adaptive monte carlo algorithm is the able to adjust the number of particle with each update to the pose array.

3.3 Personal Model

3.3.1 Model design

Several model designs were implemented for new robot designs. The addition of wheels and configurations require the package update to not only the xacro file but also the gazebo file. A two wheel differentiation system is the only system that is able to run with the current gazebo file. A three spherical wheel robot was modeled in the xacro file, but the motion controllers were not compatibility with the differential drive controller. Easy to implement configurations to the robot model were the chassis weight and the wheel friction factors.

3.3.2 Packages Used

The packages used for navigation were move base and the differential wheel controller. These packages were able to implement the navigation of the robot to be able to achieve non linear motion.

3.3.3 Parameters

Changes to the chassis weight and the friction factors did not alter the parameters for reaching the navigation goal. However, The time it took for the robot to reach the navigation goal increased.

4 Results

The results of the project were the correct robot motion and localization with the pose array. With the default parameter settings for the robot model, the robot's local cost map caused the robot to move opposite of the navigation goal. The updates to the local cost map improved the robot's motion by putting the robot on a direct path to the navigation goal. This was the main factor in reaching the navigation goal. The other main factor in rounding the corner to reach the navigation goal was the map inflation radius to avoid collisions to the obstacles. Other parameters that were added to the amcl package tuned the time it took to reach the navigation goal.

4.1 Localization Results

4.1.1 Benchmark

The benchmark results were able to reach the navigation goal with a direct path avoiding the corner obstacle. The pose array was accurately distributed under the robot's pose. The pose array accurately adapted to the robot's motion and increasingly improved the distribution of the monte carlo particles.

4.1.2 Student

The adjustments to the robot model were the robot's chassis weight and the wheel friction factors. The robot took longer to reach the navigation goal. The parameters remained equal to the benchmark robot model.

4.2 Technical Comparison

The robot parameters for the adjusted robot model and the benchmark did not differ. Since the monte carlo algortihm is based on a probabilistic function, each run resulted in varying results towards the navigation goal. The parameters were adjusted for both models to accurately move towards the navigation goal. Increasing the chassis weight and wheel friction factors increased the time it took for the robot to reach the navigation goal.

5 Discussion

The algorithm was adjusted to increase the robustness of the simulation. The robot simulation used the monte carlo particle algorithm to predict the robot's pose. The pose is based on a probabilistic function, so the robot had varying results for each simulation. To decrease the variation between simulations, parameters such as the inflation radius and the local cost map size were adjust. These parameters were adjusted so that the robot always moves towards the navigation goal. The amcl parameter were tuned to decrease the uncertainty in the robot pose array and reduced the time it took for the robot to move towards to navigation goal.

5.1 Topics

- Which robot performed better? The benchmark robot performed better
- Why it performed better? (opinion) The benchmark robot had was mechanical better.
- How would you approach the 'Kidnapped Robot' problem? A local sensor that is placed on a control point can measure the distance from the robot to the control point.
- What types of scenario could localization be performed? Construction equipment automation.
- Where would you use MCL/AMCL in an industry domain? Robotics and sensor industries to filter noise and create accurate measurements.

6 Conclusion / Future work

The project developed robot simulation skills working with XML format and URDF with added C++ programs. Localization problems were introduced in this project and the importance of localizing robot's for real time decision making and actuation. Several filters such as the EKF and MCL algorithms were also introduced for taking in sensor data and filtering out noise in the sensor data.

6.1 Modifications for Improvement

Examples:

- Base Dimension- 4 wheel drive on non flat terrain
- Sensor Location- A raise laser sensor for less robot chassis interference with laser sensor
- Sensor Layout- A local control point sensor or GPS to communicate with robot
- Sensor Amount- Inertia sensors for accurate robot motion

6.2 Hardware Deployment

1) What would need to be done? GPU and circuit board for the sensor collection, decision making and actuation
2) Computation time/resource considerations? Memory size and resolution for efficiency computation.

https://ieeexplore.ieee.org/document/6720407/
http://robots.stanford.edu/papers/fox.aaai99.pdf
http://wiki.ros.org/amcl