Robot Localization Using ROS

Vijayasri Iyer

Abstract—This paper presents an attempt to solve the task of robot localization in a simulated ROS environment. The robot performs this task using the Adaptive Monte Carlo Localization (AMCL) algorithm, which is readily available as a ROS package. The robot is successfully able to localize itself, in the given environment with proper tuning of a few important parameters.

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

1 Introduction

In robotic systems the navigation problem consists of three important sub-problems; localization, mapping and path-planning. Localization is the process of a robot estimating it's location in an environment, based on a combination of it's prior knowledge of the environment and sensor measurements. This paper describes the process of a custom made robot that localizes itself in a ROS (Robot Operating System) environment, using the Adaptive Monte Carlo Localization or AMCL algorithm. This algorithm along which is available as a readily available package, is successful in localizing the robot by tuning the parameters efficiently.

2 Background

As mentioned above, localization is the challenge of estimating a robot's pose in an environment given a map of the environment, and is one of the main areas of focus in robotics research. This is because, unlike a simulation which is an extremely controlled environment, real-life situations have result in much more noisy sensor measurements. The localization problem is solved using a combination of two methods 1)leveraging sensor data 2)using an approximation algorithm to filter out all of the improbable locations of the robot. There are various approximation algorithms available to solve this problem, the most common ones being the Kalman and Particle Filters. These algorithms are discussed in detail in the sections below.

2.1 Kalman Filters

The Kalman filter is a type of approximation algorithm, named after it's creator Rudolf E. Kalman. This algorithm has been used widely in a variety of navigation and control systems, including the Apollo Program in 1960. It is used to measure the state of the system when the measurements are noisy. This filter uses two main functions in an infinite loop namely state prediction and measurement update, which is also known the predictor-corrector logic. These two steps are used to recursively update the mean and variance of the initial gaussian distribution until it arrives at a distribution with a very high mean and low variance. The Kalman filter exclusively operates when all it's variables are Gaussian Distributions. There are many variants of the Kalman filter namely the 1)Multidimensional Kalman filter (linear) 2)Extended Kalman filter (non-linear) and 3)Unscented Kalman filter (highly non-linear). For real-world tasks, since they are non-linear in nature, the extended and unscented variants of the filter are more likely to provide better results.

2.2 Particle Filters

A particle filter basically uses the principle of a probability distribution to eliminate the unlikely poses (position + orientation) of a robot. Here, the current robot pose is represented using three parameters the x-coordinate, y-coordinate, and an orientation theta with respect to the global frame. In a particle filter, particles are initially spread randomly throughout the entire map. these particles are not physical, they exist only in simulation. Even the particles have x,y and theta coordinates. Hence, each particle represents a hypothesis of the robot being in that particular location. In addition, each particle has a weight which represents the difference between the location of the particle i.e predicted pose and the actual location of the robot. Importance of a particle depends on its weight, which indicates its likelihood to survive a resampling process. After several iterations of MCL and several stages of resampling the particles will converge into a probable robot location. Particle filters are said to be better than Kalman filters in situations where the belief is said to be multimodal ie we have to deal with non-gaussian distributions. The powerful Monte Carlo localization algorithm estimates the posterior distribution of a robots position and orientation based on sensory information. This process is known as a recursive Bayes filter.

2.3 Comparison / Contrast

A comprehensive explanation of the difference between the above approximation algorithms is given below in the form of a table.

Kalman Filter	Particle Filter
Linear/Non-Linear	Non-Linear
Less flexible	More flexible
Lower computation cost	Higher computation cost
Solution is stable	Solution is relatively unstable
Only Gaussian distribution	Any arbitrary distribution

As mentioned in the above table, even though particle filters are computationally expensive, they are better suited to noisy and non-linear environments and work with non-gaussian distributions. Hence, for this task the AMCL algorithm is used consisting of a particle filter. AMCL or Adaptive Monte Carlo Localization is a variant of the MCL algorithm which performs localization on samples of particles rather than all of the particles together. This leads to a more stable solution.

3 SIMULATIONS

This project used the Jackal simulator created by Clearpath Robotics for testing their mobile base robot Jackal. It consists of a simple maze-like environment.

3.1 Benchmark Model

3.1.1 Model design

The benchmark model is a very simple differential drive mobile robot with four wheels; two for driving and two castor wheels for support. The size of the robot is quite small and it has two sensors attached; a 2D stereo camera and a Hokuyo Lidar. An image of the benchmark model is provided below in fig 1.

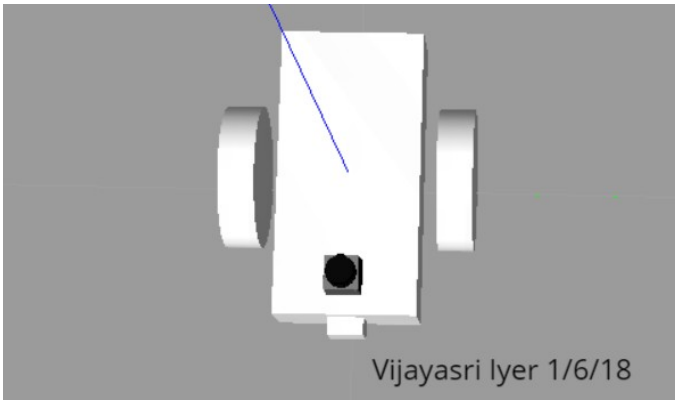

Fig. 1. Benchmark Model.

3.1.2 Packages Used

This project requires mainly the use of three ROS packages among others. The **amcl** package provides a readymade implementation of AMCL to localize the robot. It takes in a laser-based map, laser scans, and transform messages, and provides pose estimates as output. It subscribes to the scan, tf, initiapose and map topics and publishes to the amcl pose, particlecloud and tf topics. The amcl package provides global_localization and request_nomotion_update services which are further explained in the ROS wiki pages.
The **move_base** node in the move_base package, a ROS interface for configuring, running, and interacting with the navigation stack on a robot. It combines together a global and local planner to accomplish its global navigation task. The move_base node subscribes to the topics move_base_simple/goal, geometry_msgs/PoseStamped and publishes to the cmd_vel topic.
The **map_server** package provides the map_server node which allows ROS to use map data as a service.
The role of each package in the navigation stack is shown in the image provided below.

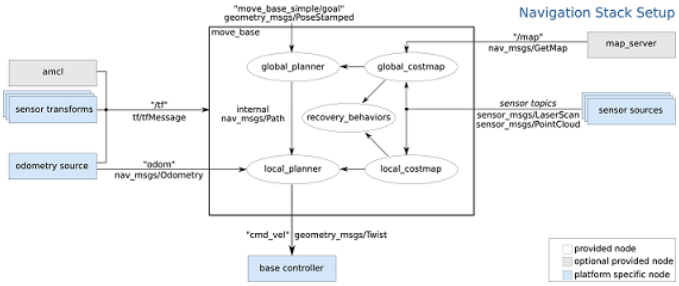

Fig. 2. Benchmark Model.

3.1.3 Parameters

Localization parameters in the AMCL node should be described, as well as move_base parameters in the configuration file. You should be able to clearly demonstrate your understanding of the impact of these parameters.

3.2 Personal Model

3.2.1 Model design

The custom model was desgined based on the benchmark model with a few additional changes. This model had an additional platform on top of the base platform, with the camera sensor placed on top of this new platform. An image of this model is provided below fig 2.

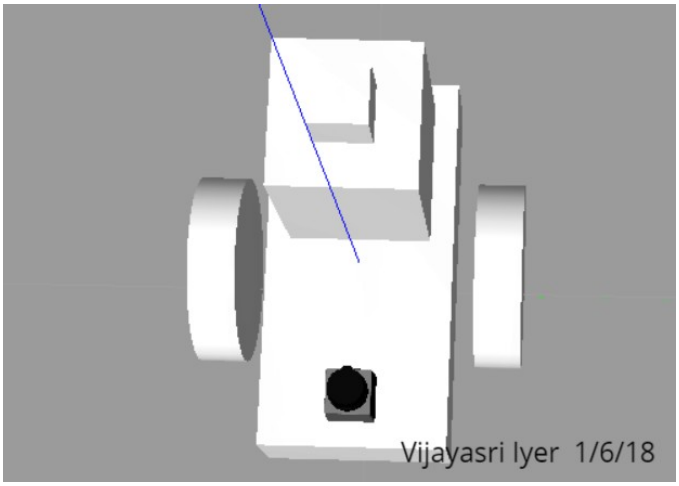

Fig. 3. Personal Model.

3.2.2 Packages Used

The packages used for this model are identical to the packages used for the benchmark model.

3.2.3 Parameters

The parameter values of the config files and the packages also remain the same for this model.

3.3 Achievements

4 Results

Present an unbiased view of your robot's performance and justify your stance with facts. Do the localization results look reasonable? What is the duration for the particle filters to converge? How long does it take for the robot to reach the goal? Does it follow a smooth path to the goal? Does it have unexpected behavior in the process?
For demonstrating your results, it is incredibly useful to have some watermarked charts, tables, and/or graphs for the reader to review. This makes ingesting the information quicker and easier.

4.1 Localization Results

4.1.1 Benchmark

Fig. 4. Benchmark Model.

4.1.2 Student

Fig. 5. Benchmark Model.

4.2 Technical Comparison

Discuss the difference of the layout, parameters, performance etc. between the benchmark robot and your robot. It is acceptable for your custom robot to perform worse than the provided robot. The focus is on learning and understanding, not performance.

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.

5.1 Topics

- Which robot performed better?
- Why it performed better? (opinion)
- How would you approach the 'Kidnapped Robot' problem?
- What types of scenario could localization be performed?
- Where would you use MCL/AMCL in an industry domain?

6 Conclusion / Future work

This section is intended to summarize your report. Your summary should include a recap of the results, did this project achieve what you attempted, how would you deploy it on hardware and how could this project be applied to commercial products? For Future Work, address areas of work that you may not have addressed in your report as possible next steps. This could be due to time constraints, lack of currently developed methods / technology, and areas of application outside of your current implementation. Again, avoid the use of the first-person.

- example 1
- example 2

1) example 1
2) example 2

6.1 Subsection Heading Here

Subsection text here.

Fig. 6. Robot Revolution.

6.1.1 Subsubsection Heading Here

Subsubsection text here.

6.2 Modifications for Improvement

Examples:

- Base Dimension
- Sensor Location
- Sensor Layout
- Sensor Amount

6.3 Hardware Deployment

1) What would need to be done?
2) Computation time/resource considerations?