Navigation Technique using Adaptive Monti Carlo Localization in Simulation

Shahnawaz Khokhar

Abstract—Localization techniques have been deployed for autonomous navigation with remarkable success in recent years. Advances in computing power have led to efficient navigation techniques that can be combined with recent advances in machine vision algorithms. Here, Adoptive Monti Carlo Localization is used to demonstrate an autonomous navigation task in a wheeled robot. The

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

1 Introduction

Localization is the fundamental problem in robotics autonomous navigation. One of the first use cases emerged in the 1970s with the race to the moon. NASA engineers were tackling the problem of outer space telemetry that relied on noisy measurements from sensor data. They were looking for a way to estimate the trajectory of a space craft as the measurments were being collected. They also needed a system to be able to quickly recover the pose in case of computer failure.

Rudolp Kalman devised a solution that calculated the joint probability distribution of the measurements. Though Kalman Filters are useful to calculating speed and direction for linear systems they are not applicable in non linear motions like moving in a circle. Extended Kalman filters work on non-linear systems and use the same Gaussian of the posterior. They are not robust and limited to local state. Adoptive Monti Carlo localization (AMCL) uses particle filters and can represent a global state and is more accurate in predicting pose estimates.

introduction should provide some material regarding the history of the problem, why it is important and what is intended to be achieved. If there exists any previous attempts to solve this problem, this is a great place to note these while conveying the differences in your approach (if any). The intent is to provide enough information for the reader to understand why this problem is interesting and setting up the conversation for the solution you have provided Use this space to introduce your localization task and how you wish to accomplish it; save the details about the robot construction for later (simulation is a good point for this information). If you have any papers / sites / repositories you have referenced for your robot, please make sure to cite them.

2 Background

Here an AMCL package is used to demonstate a localization task for a two wheel robot. . AMCL package is implemented in a ROS Gazebo simulation. for the exersieIt was chosen over extended Kalman Filters because of its ability to do global localization. The

[?]

2.1 Kalman Filters

Extended Kalman filters are an acurate way to estimate robot pose in uncertain. Conventional Kalman Filters can be conditioned on sensor readings to update velocity or direction over time. Extended Kalman filters can be applied to non linear systems to predict the value of a control over time as the data is being collecting. In both cases, they are running averages of the state of trajectory.

2.2 Particle Filters

Particles are a hypothesis of where the robot might be on a map. This is represented pose estimates in cartesean plane with x and y coordinates associated with each particle. In AMCL these particles are spread out randomly on the map. The particles where the robot is least likely to be are eliminated as sample are collected. Particle filters are particularly useful because they are easy to implement. Each particle is a sample of a robots location and orientation.

2.3 Extended Kalman Filters vs Particle Filters

Particle filters provide estimates of robots pose in a global setting. This means that a robot can be localized even if the initial pose is not known. Both methods use Gaussian of the posteriors. Extended Kalman Filters do not work in global localization tasks.

3 Simulations

A two wheeled robot with actuators on each wheel is used in this demonstration. A ROS package was created to launch the robot world configuration. The robot is equipped with a laser range sensor and a front facing camera. The simulation was run in the provided environment complete with barriers and obstacles that the robot had to avoid. Rviz was utilized as a visualization tool. Navigation Stack and AMCL packages were installed to the catkin workspace locally in the Jetson Tx2 development kit. The goal was to tune the parameters of the local and global costmap as well as the number of particles.

3.1 Achievements

Both robots successfully localized themselves and performed the requested navigation task.

3.2 Benchmark Model

The Udacity bot is the benchmark robot model used here.

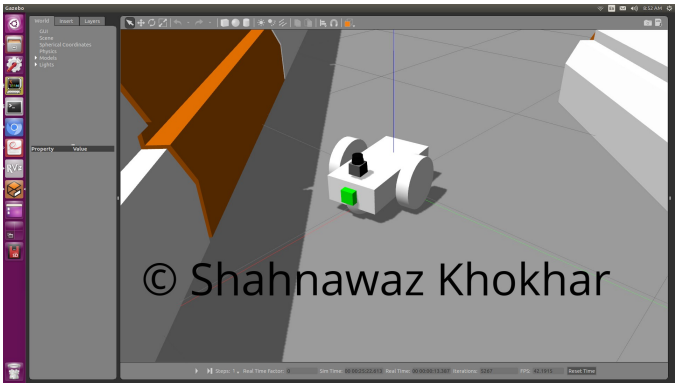

Fig. 1. Udacity bot model

3.2.1 Model design

The Robot's design consists of a box chassis, two wheels with actuators, a Hoyuku range finder and a forward facing camera. The robot was developed using Unified Robot Description Format file also known as URDF. The file describes the configurations such as base geometry, inertial measurements and collision parameters. A detailed set of description of the URDF is tabulated below.

TABLE 1
Udacity Bot URDF Specifications

Udacity Bot body

Part	Geometry	Size
Chassis	Cube	0.4 x0.2 x 0.1
Back and front Casters	Sphere	0.0499(radius)
Left and Right wheels	Cylinders	0.1(radius), 0.05(length)

3.2.2 Packages Used

A ROS package was created in order to launch the robots. The directory structure for the packages is outlined below.

- AMCL
- RVIZ
- Meshes
- Maps
- launch
- worlds
- URDF
- config

3.2.3 Parameters

In order to archive the results required, parameters for local cost map and global cost-map had to be tuned along with robot speed. The parameters were tuned in a brute force approach followed by an iterative adjustment. See Table 3 for parameters used to obtain results for this demonstration.

TABLE 2

AMCL Node Parameters for Benchmark	
holonomic robot	false
yaw goal tolerance	0.05
xy goal tolerance	0.1
sim time	1.0
meter scoring	true
pdist scale	0.5
gdist scale	1.0
max vel x	0.5
min vel x	0.1
max vel theta	2.0
acc lim theta	5.0
acc lim x	2.0
acc lim y	5.0
controller frequency	15.0
map type	costmap
obstacle range	2.5
raytrace range	3.0
transform tolerance	0.3
inflation radius	0.5

3.3 Personal Model

The personal model shared the same features the benchmark except a cylindrical chassis for better weight distribution. This repository was referred to for design considerations. A cylindrical design may have better cornering capabilities and less likely to get stuck in tough corners.

Fig. 2. Custom Bot model

3.3.1 Model design

TABLE 3
Custom Bot URDF Specifications

Custom Bot body

Part	Geometry	Size
Chassis	Cylinder	Radius 0.2 x Length 0.1
Back and front Casters	Sphere	0.0499(radius)
Left and Right wheels	Cylinders	0.1(radius), 0.05(length)

3.3.2 Packages Used

All the same packages were used as the benchmark.

- AMCL
- RVIZ

- Meshes
- Maps
- launch
- worlds
- URDF
- config

3.3.3 Parameters

TABLE 4

AMCL Node Parameters for Benchmark	
holonomic robot	false
yaw goal tolerance	0.05
xy goal tolerance	0.1
sim time	1.0
meter scoring	true
pdist scale	0.5
gdist scale	1.0
max vel x	0.5
min vel x	0.1
max vel theta	2.0
acc lim theta	5.0
acc lim x	2.0
acc lim y	5.0
controller frequency	15.0
map type	costmap
obstacle range	2.5
raytrace range	3.0
transform tolerance	0.3
inflation radius	0.5

4 Results

The pose matrix for both the benchmark and the custom robot converged to the ground truth quickly. The cost-map hyper-parameters were tuned first. With initial tuning the benchmark was able to find a correct path but would get stuck on turns. The local cost map values and global cost-map values had to be added. I used a value close to half of the Hoyuku maximum range to determine local cost-map configurations and then adjusted it with iterations. The robot radius,obstacle range, inflation radius were the parameters that allowed better navigation around walls and tight corners. This was important to avoid Collision with the wall and situations where the robot can get stuck making sharp turns around corners.

The benchmark and the custom robot were able to take the similar paths to the goal because of marked similarities in weights and inertia. The number of particles, which is tuned for accuracy of the localization process, was set to 10 minimum and 200 maximum. This number worked for the simulation. More particles would increase the accuracy of localization but constrain more computational resources.

The custom model fared well with the benchmark overall but seemed to be a little less responsive in turns.

4.1 Technical Comparison

The custom robot shared many features with the benchmark apart from the shape of the body. All wheels and actuators are similar to the original robot. The minor changes in robot's body shape did not seem affect performance.

5 Discussion

Two wheel robots are very simple, hence don't have as many kinematic challenges as 4 wheel or legged robots. One may think that a round chassis has better weight distribution and better turning angles. However, this is not the case.

5.1 Topics

- Both robots performed the same task with high degree of robustness.
- Particle filters can partially solve the kidnapping robot problem for small closed environments having different features. A better solution for KRP is in the SLAM problem. Data association is the main issue with particle filters.
- MCL/AMCL can be used in closed environments with many features. It can be used for resource constrained systems that need to get the task done.

6 Conclusion / Future work

In the future, a 4 wheel robot can be tested. Further considerations can take into account a turning module. Hardware deployment can also be tested on a Jetson TX2.

6.1 Modifications for Improvement

Robot design considertions

- Longer length of cylindrical chasiss
- A range finder can be located on a bracket in front of the robot.
- a second range sensor located behind the
- Sensor Amount

6.2 Hardware Deployment

1) Jetson TX2 GPIO controllers can designate cmd vel commands to motors of a two wheeled robot.
2) The hyperparamaters will need to be tuned.