Adaptive Monte Carlo Localization Project

Pritesh Gudge

Abstract—The project includes the application of the AMCL Package in ROS for localization and using sensor fusion of an Odometer and a Hokuyo Laser Range Finder in a Gazebo Simulation Environment. Two Robots with different configurations are used simulated in Gazebo for the project.The simulated robot is supposed to navigate in a controlled obstacle simulation by continuously localizing its pose on the global map and building a cost map to calculate further trajectory in order to be able to reach the goal location without getting stuck and by using the shortest route possible.

Index Terms—Robot, IEEEtran, Udacity, Localization, Monte Carlo Localization.

1 Introduction

The real world mobile robots always have to perform certain actions based on the tasks they are supposed to perform. The next action of the robot is based on the position on the robot in the operating environment and the task to be performed. The project attempts to implement Adaptive Monte Carlo Localization [1] for a simulated environment. The base environment provided by Udacity is used here. ROS AMCL package is used to for the localization. The decisions about the actuation robot are based on the sensor fusion data fed into the AMCL algorithm and the corresponding outputs.

2 Background

For a robot to operate in the real world, it needs to keep track of its position and orientation, also called pose. The robots needs to continuously calculate its pose to be able to perform its movement and tasks. This is called the localization problem.

Robot Localization problems are classified on the complexity of the localization task and the environment. Position Tracking, Global Localization and Kidnapped Robot Localization. Due to the uncertainties in measurement and real world operation scenarios the robot's position and the related actuation required also has uncertainty. Filters and localization algorithms are used to solve some of the problems. Robots operating in the real world have to adapt the terrain, location and displacement from predefined paths. The robots should be able to recover and continue functioning effectively.

2.1 Kalman Filters

Kalman filters work by performing state prediction, followed by measurement update from multiple sensors(also called sensor fusion). The measurement update with the previous state provides an estimate of the next state of the robot. The Kalman filters take into account the uncertainty in measurement of the sensors and performs state prediction by using the previous state and measurement update.

The assumptions of Kalman filters are: motion and measurement models are linear, state space can be represented by a unimodal Gaussian distribution. These assumptions cannot be applied to all the real world robots where motion and measurement models are complex non-linear curves and states cannot be modelled as unimodal Gaussian distribution.

The Extended Kalman Filters can be used to when the motion or measurement functions are non-linear. The non-linear functions can be used to update the mean but not the variance. Over a short interval of time, non-linear functions can be approximated to locally linear functions by using the Taylor series equation. Thus, Taylor series [2] can be used to provide approximate state and measurement updates for the variance.

Briefly describe Kalman filters. Explain how they work and why they are used for localization. Additionally, discuss the drawbacks of linear Kalman filters and how Extended Kalman Filters (EKFs) help resolve some of these issues.

2.2 Particle Filters

In robotics, a particle is a virtual element that represents a robot's pose. The particles are sampled at each movement of the robot. Monte Carlo Localization Algorithm is called Particle Filter Algorithm. This works only for Global Localization problems. 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

The differences between the EKF and MCL(Particle Filter) approaches are described below.

2.3.1 EKF

- EKF is memory and computationally efficient algorithm.
- EKF can effectively model Unimodal Continuous State Spaces.

- EKF can be used in Position Tracking Localization Problems
- EKF cannot be used to solve Global Localization Problem
- EKF is relatively difficult to implement due to the complex equations involved to code.
- For larger problems the memory and resolution cannot be controlled.

2.3.2 MCL

- MCL is memory and computationally heavy algorithm.
- MCL can effectively model Multimodal Discrete State Spaces.
- MCL can be used to solve Global Localization Problem
- MCL is relatively easy to implement.
- For larger problems the memory and resolution can be controlled.
- MCL is robust and can recover from errors and wrong estimations.

3 SIMULATIONS

The simulation is performed in the Gazebo Physics simulator environment. The simulation is done with the provided benchmark robot and a custom robot. The simulation is aimed at reaching a pre-defined target by performing simultaneous localization and motion until the robot reaches the final goal pose.

3.1 Achievements

3.2 Benchmark Model

3.2.1 Model design

The project has a base model provided for benchmarking and parameter setting for evaluation.

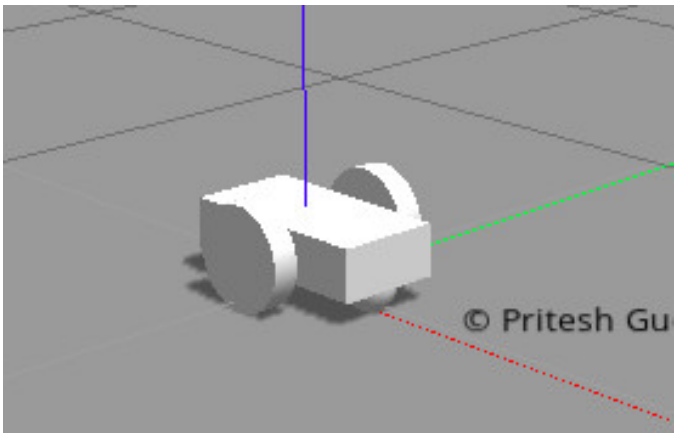

Fig. 1. Benchmark Model

TABLE 1
Benchmark Model

Property	Values
Length	0.4m
Width	0.2m
Heigth	0.1m
Mass	15 kg
Wheel Radius	0.1m
Castor Radius	0.05m
Camera	Front of Chassis (x=0.2m)
Hokuyo Depth Sensor	Top of Chassis (x=0.15m & z=0.1m)

3.2.2 Packages Used

The following Packages were used:

i Camera: Topic- `"/udacity_bot/camera1/image_raw"`

ii Depth Sensor(Hokuyo): Topic- `"/udacity_bot/laser/scan"`

iii Move Base Package Reads the config parameters and controls the cmd_vel topic. It reads from the odom and depth sensor topics.

iv AMCL Package Reads the Config Parameters. Reads the Odom and Depth Sensor Topics and writes out the robots estimated state.

v Robot State Publisher Publishes the robots pose

vi Joint State Publisher Publishes the robots joint states

3.2.3 Parameters

- AMCL
  - min_particles: 10
  - max_particles: 50
  - transform_tolerance: 0.3
  - controller_frequency: 15
  - odom_alpha(all): 0.01
- local_costmap
  - global_frame: odom
  - robot_base_frame: robot_footprint
  - update_frequency: 20.0
  - publish_frequency: 15.0
  - width: 5.0
  - height: 5.0
  - resolution: 0.1
  - static_map: false
  - rolling_window: true
- global_costmap
  - robot_base_frame: robot_footprint
  - update_frequency: 20.0
  - publish_frequency: 15.0
  - width: 80.0
  - height: 80.0
  - resolution: 0.1
  - static_map: true
  - rolling_window: false
- costmap_common
  - obstacle_range: 2.5

  - raytrace_range: 3.0
  - transform_tolerance: 0.3
  - robot_radius: 0.2
  - inflation_radius: 0.1
  - observation_sources: laser_scan_sensor
  - pdist_scale: 2.0
  - gdist_scale: 3.3
  - occdist_scale: 1.5
  - meter_scoring: true
  - sim_time: 2.0
  - xy_goal_tolerance: 0.05
  - yaw_goal_tolerance: 0.05
- TrajectoryPlannerROS
  - holonomic_robot: false
  - acc_lim_x: 2.0
  - acc_lim_y: 2.0
  - acc_lim_theta: 1.5
  - min_vel_theta: -0.4
  - max_vel_theta: 0.4
  - max_vel_x: 0.4
  - min_vel_x: -0.4
  - escape_vel: -0.3
  - xy_goal_tolerance: 0.05
  - yaw_goal_tolerance: 0.05

3.3 Custom Model

3.3.1 Model design

The model is designed using a chassis with two wheels in the rear connected by a differential drive controller. On the front a smooth caster is placed for support and easy turn manoeuvring. Refer figure 2

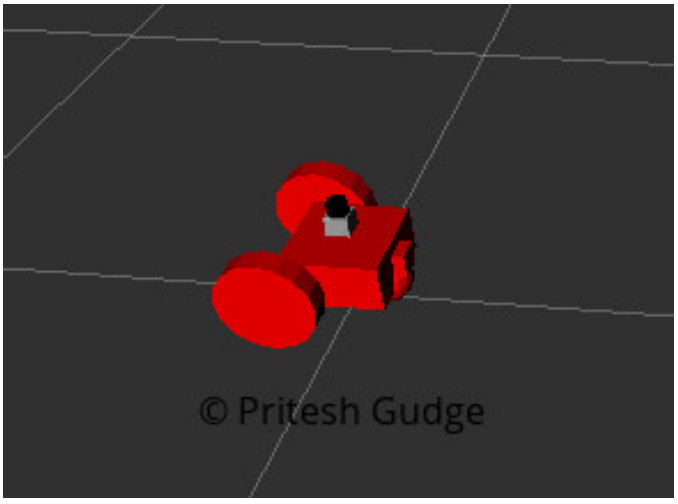

Fig. 2. Custom Model

3.3.2 Packages Used

The following Packages were used:

i Camera: Topic-
`"/udacity_bot/camera1/image_raw"`

ii Depth Sensor(Hokuyo): Topic-
`"/udacity_bot/laser/scan"`

iii Move Base Package Reads the config parameters and controls the cmd_vel topic. It reads from the odom and depth sensor topics.

iv AMCL Package Reads the Config Parameters. Reads the Odom and Depth Sensor Topics and writes out the robots estimated state.

v Robot State Publisher Publishes the robots pose

vi Joint State Publisher Publishes the robots joint states

TABLE 2
Custom Model

Property	Values
Length	0.2m
Width	0.2m
Heigth	0.1m
Mass	10 kg
Wheel Radius	0.1m
Castor Radius	0.05m
Camera	Front of Chassis (x=0.1m)
Hokuyo Depth Sensor	Top of Chassis (z=0.1m)

3.3.3 Parameters

- AMCL
  - min_particles: 10
  - max_particles: 50
  - transform_tolerance: 0.3
  - controller_frequency: 15
  - odom_alpha(all): 0.1
- local_costmap
  - global_frame: odom
  - robot_base_frame: robot_footprint
  - update_frequency: 20.0
  - publish_frequency: 15.0
  - width: 5.0
  - height: 5.0
  - resolution: 0.1
  - static_map: false
  - rolling_window: true
- global_costmap
  - robot_base_frame: robot_footprint
  - update_frequency: 20.0
  - publish_frequency: 15.0
  - width: 90.0
  - height: 90.0
  - resolution: 0.1
  - static_map: true
  - rolling_window: false
- costmap_common
  - obstacle_range: 2.5
  - raytrace_range: 3.0
  - transform_tolerance: 0.3
  - robot_radius: 0.2
  - inflation_radius: 0.1
  - observation_sources: laser_scan_sensor
  - pdist_scale: 2.5

  - gdist_scale: 1.5
  - occdist_scale: 1.6
  - meter_scoring: true
  - sim_time: 1.5
  - xy_goal_tolerance: 0.05
  - yaw_goal_tolerance: 0.05
  - oscillation_reset_dist: 0.01
- TrajectoryPlannerROS
  - holonomic_robot: false
  - acc_lim_x: 2.0
  - acc_lim_y: 2.0
  - acc_lim_theta: 1.5
  - min_vel_theta: -0.4
  - max_vel_theta: 0.4
  - max_vel_x: 0.4
  - min_vel_x: -0.4
  - escape_vel: -0.3
  - xy_goal_tolerance: 0.05
  - yaw_goal_tolerance: 0.05

4 RESULTS

4.1 Localization Results

The below sections describe the results of the AMCL run on the benchmark and the custom models.

4.1.1 Benchmark

The benchmark model reaches the destination with a position tolerance set at 0.05 metres and orientation tolerance set at 0.05 radians. The path taken is show in figure and figure 3, figure 4 and figure 5. The model smoothly reaches the goal. The output image is described in the figure 6.

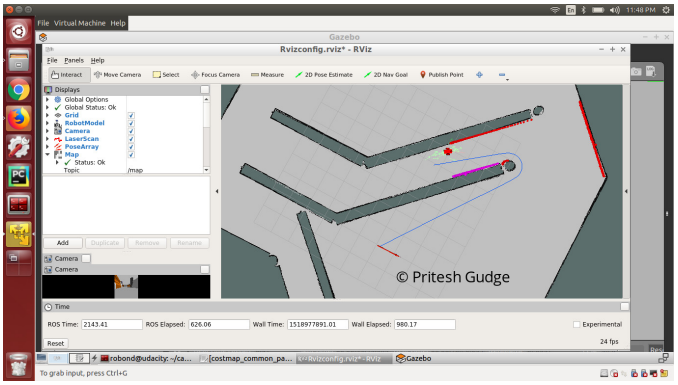

Fig. 3. Benchmark model Intermediate Position 1

4.1.2 Custom

The custom robot takes the path as shown in figure 8,9 and figure 10. The custom robot reaches near the location as shown in figure 11

4.2 Technical Comparison

4.2.1 Layout

The layout of the robot models are compared in the table 3

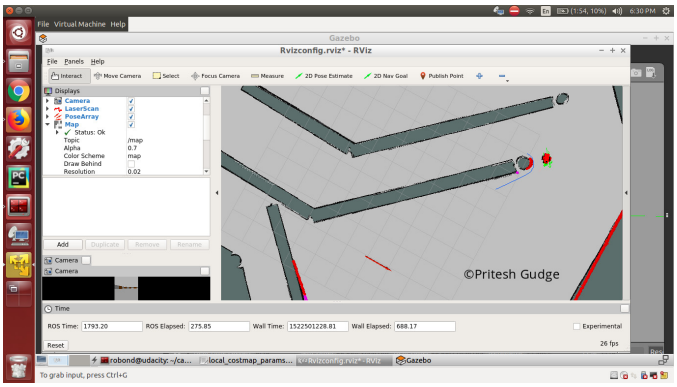

Fig. 4. Benchmark model Intermediate Position 2

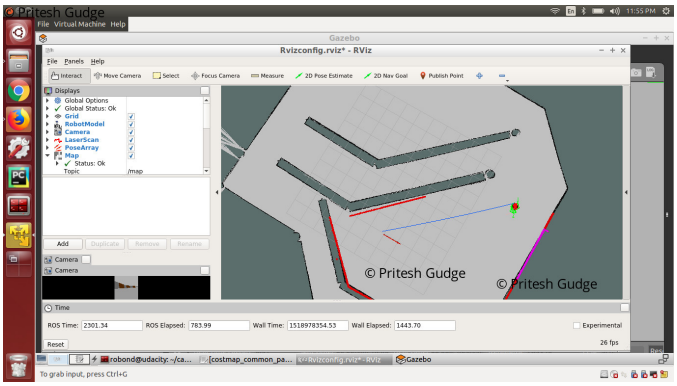

Fig. 5. Benchmark model Position Intermediate Position 3

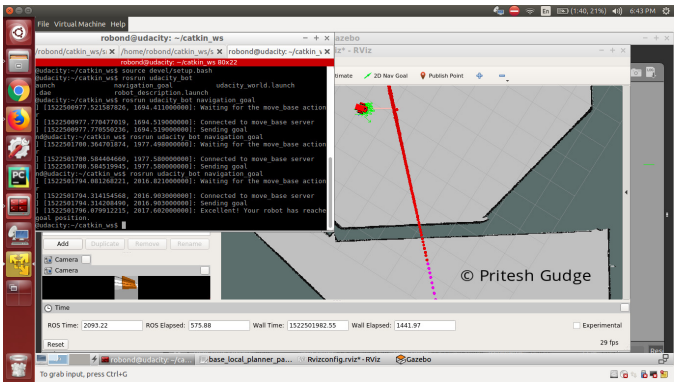

Fig. 6. Benchmark model Final Position

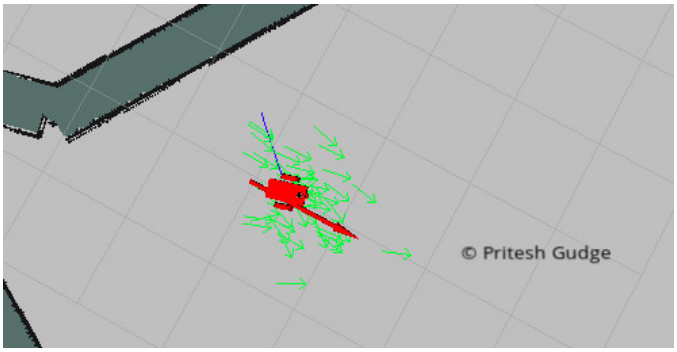

Fig. 7. Benchmark model Final Position

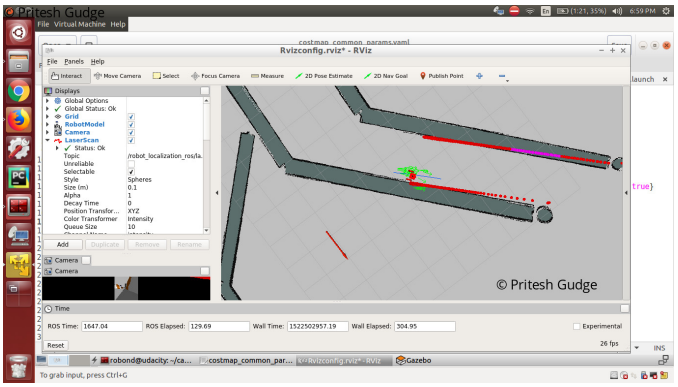

Fig. 8. Custom model Intermediate Position 1

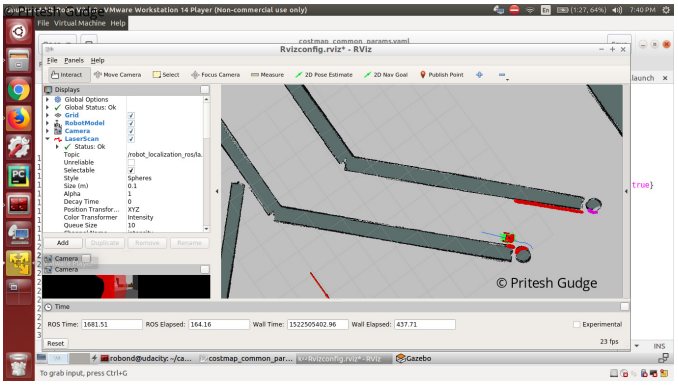

Fig. 9. Custom model Intermediate Position 2

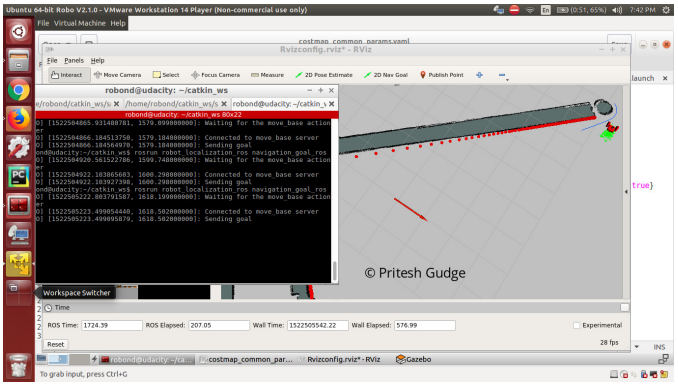

Fig. 10. Custom model Intermediate Position 3

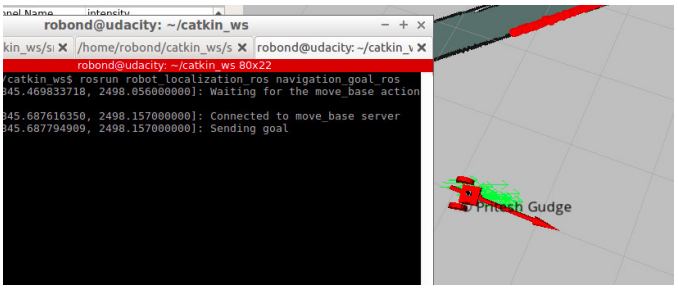

Fig. 11. Custom model Final Position

TABLE 3
Layout Comparison

Benchmark	Custom
2 wheels lateral,2 casters longitudinal	2 wheels rear,1 caster front
Camera in front	Camera in Front
Laser Range Finder on Top Front	Laser Range Finder on Top Middle
Torque 20	Torque 20

4.2.2 *Parameters*

The main parameters of the models are compared in the table 4

TABLE 4
Parameters Comparison

Parameter	Benchmark	Custom
sim_time	2.0	1.5
p_dist	2.0	2.5
g_dist	3.3	1.5
occ_dist	1.5	1.6

4.2.3 *Performance*

The performances of the models are compared in the table 5

TABLE 5
Performance Comparison

Parameter	Benchmark	Custom
Goal Position Time	12 min	26 min
Goal Path	Smooth	Oscillating

5 Discussion

- **Performance:** The custom robot performance is poor compared to the bench mark robot.
- **Performance Reason:** The design of the custom robot with two wheels in the rear and one caster in the front makes it harder for the custom robot to rotate in position. The center of the custom robot has to go around in a circle for it to be able to change its direction by 180 degrees. There are many reasons for this. This reduced flexibility especially around bends and in positions where the robot gets stuck causes problems as the localization algorithm does not consider the layout of the robot while estimating the measurements and pose.
- **Kidnapped Robot Situation**: The detection of kidnapped robot is one of the most difficult problems in Monte Carlo localiza- tion (MCL).This is due to the nature of particle filter used in MCL itself, where the convergence process of hypotheses (called particles) causes the absence of particles in some areas, leading to a localization failure if the robot is kidnapped

to that area [3]. In augmented MCL(AMCL) [4], random particles are injected in each iteration so that the possibility of particles absence in kidnapping destination area is reduced. AMCL does not clearly draw a line between detection and recovery of kidnapping. This creates a problem when the concern is not only in the re-localization but also the needs to know when the kidnapping really happens, such as in fault detection. This is where AMCL fails [3].

- **Kidnapped Robot Situation Alternate Solution:** The 'Kidnapped Robot' problem could attempted to be solved by storing images, distances and locations of the landmarks eg. Corners, bends, poles and the pose of the robot from which the image is captured. A neural network can be trained on this and the trained model can be used to predict the location of the robot based on the image captured by the camera.
- **Industry Use:** MCL and AMCL could be used in robots which performing picking and putaway operations in a warehouse as in a typical warehouse the layout is known precisely and does not change frequently.

6 Conclusion / Future work

6.1 Modifications for Improvement

The following modifications can be attempted to improve the performance of the custom robot.

- **Base Dimension:** The base dimension can be decreased to allow longer radius of turn and preventing the differential from locking.
- **Sensor Location:**The Laser range finder can be placed in the front of the chassis for better measurement precision without the chassis blocking part of the surrounding.
- **Additional Sensors:** Additional range finders and cameras can be added to the rear and the sides of the chassis to provide for better estimation in location.

6.2 Hardware Deployment

1) The custom chassis can be built by using a block of wood and the range finder and RGB camera can be bought off the shelf. An off the shelf microcontroller can be bought and used to run the localization on board the robot. Standard Wheels and Differential drive controller can be bought off the shelf. The robot can be assembled using the above components.
2) The sim_time parameter and the resolution parameters for the local and global costmaps can be controlled based on the precision of the final pose of the robot. For the robot to have precise final pose, it is necessary to deploy a computer with higher configuration of CPU and RAM.

References

[1] B. P. Gerkey, "Adaptive monte carlo localization." http://wiki.ros.org/amcl#Parameters.
[2] E. W. Weisstein, "Taylor series.." http://mathworld.wolfram.com/TaylorSeries.html.
[3] I. Bukhori and Z. H. Ismail, "Detection of kidnapped robot problem in monte carlo localization based on the natural displacement of the robot," *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, p. 1729881417717469, 2017.
[4] S. Thrun, D. Fox, W. Burgard, and F. Dallaert, "Robust monte carlo localization for mobile robots," *Artif. Intell.*, vol. 128, pp. 99–141, May 2001.