Where am I

Siddharth Kothiyal

Abstract—A robot's tasks can be broken down into Sensing, Decision making and Actuation. An important criteria for decision making is for the robot to know with a certain level of confidence, where it is in the environment. The aim of this paper is to explore Monte Carlo approximation and use it to perform localisation. This paper explores this by using gazebo simulator along with ROS to implement AMCL and find the robot's way through a track to reach the goal.

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

1 Introduction

Localisation is an important function for a robot to perform. It helps the robot determine whether it's reached its goal or if it's near the object it needs to actuate upon etc, and thus effecting the decision making of the robot. There are three types of localisation problems:

1) Local localisation: robots initial pose is known and it keeps track of the changes to preform localisation.
2) Global localisation: the robot does not know it's original pose in the environment, it tries to identify the features in it's surrounding to map it to the ground truth map, and hence its pose.
3) Kidnapped robot localisation: it is an extension of the global localisation problem, but during the localisation steps, the robot can be kidnapped and placed anywhere else randomly. This form of localisation helps the robot recover in case of failures or if it has made a mistake in localisation.

Robots use a variety of sensors varying from Depth sensing cameras to IMU sensors depending on the problem they need to solve. This paper looks into global localisation and uses a hokuyo lidar to help it in localisation.

2 Background

Localisation can be performed in a variety of ways, these include using sensors like cameras and lidars, or location tracking sensors like IMU and GPS. All sensors suffer from some noise and error in measurement, and even after combining data from all these sensors, some error remains in the estimation part. Our aim is to minimize this error; different techniques work best depending on the environment. Some of the main techniques used for localisation are Grid Localisation, Monte Carlo Localisation, Kalman Filters and Markov Localisation.

2.1 Kalman Filters

Kalman filter is a method of state estimation, where the algorithm takes an initial assumption about the state of the object, and with every noisy measurement, the state estimation changes and the degree of confidence on the estimation increases. As Kalman filter assumes state transition and measurement vary in a linear fashion, it is only able to model the state of the object according to a gaussian distribution. Kalman filter is known to provide accurate estimates at a very high speed and with low computation. It is a preferred method, when the computation power available is limited and real time estimation is very important.

Since most real life scenarios exhibit non-linear behavior, an extension of Kalman Filter, Extended Kalman Filter (EKF) technique is used to model the state of an object when state transition and measurements are not varying in a linear fashion.

2.2 Particle Filters

Particle Filters or the Monte Carlo Localization (MCL) are another way used to state estimation. In this technique, the algorithm initially takes a lot of samples randomly and then tries to compute the likelihood of the particle being the same state as the object. According to the likelihood, some particles are kept and some less likely particles are dropped. The algorithm then continues to re-sample the particles, the region from which more particles survived are preferred for during sampling of new particles, and those particles have a higher weight/likelihood of being in the same state as the object. There are no limits to the number of particles sampled at one time, but depending on the environment, a perfect trade-off needs to be found between the computation power needed and the gain in accuracy due to increasing the number of particles being sampled in the system.

2.3 Comparison / Contrast

As mentioned earlier, all methods of estimations have their own set of advantages, and the best method depends on the environment of the object. Kalman filter turns out to be computationally less expensive and works very well in situations where the state of the object is less uncertain. But it is only able to model the state of objects according to the gaussian distribution, whereas, the MCL can model any distribution. MCL also performs very well in global and kidnapped robot problems, whereas the Kalman Filter is not robust enough to perform well in these situations.

MCL is able to represent non-gaussian distribution and is able to represent any other distribution, hence it is more practical in real world, as we are able to represent the world according to gaussian distribution only in very limited

scenarios. On top of that, we can change the memory and computation requirement in MCL by changing the number of particles being used for approximation.

	MCL	EKF
Measurements	Raw Measurements	Landmarks
Measurement Noise	Any	Gaussian
Posterior	Particles	Gaussian
Efficiency(memory)	✓	✓✓
Efficiency(time)	✓	✓✓
Ease of Implementation	✓✓	✓
Resolution	✓	✓✓
Robustness	✓✓	x
Memory & Resolution Control	Yes	No
Global Localization	Yes	No
State Space	Multimodel Discrete	Unimodal Continuous

Fig. 1. Comparison between MCL and EKF

3 Simulations

The aim of the project was to make the benchmark model of a robot pass through an already mapped maze. The world was created and simulated using Gazebo simulator, and the communication with the world was handled by ROS framework, Rviz was used for various visualisation purposes, e.g the data coming through the sensors, the cost map created by the robot etc. The ROS navigation stack along with the AMCL package was used to provide localisation capabilities to the robot.

3.1 Achievements

The benchmark model was used during the simulation to tune the parameters over several iterations. Rviz was used to visualize the costmaps created by the bot in real time and was used for debugging. The benchmark model was then edited to create a modified model, and the parameter tuning was updated for the model accordingly, to minimize the time it takes for the model to reach the destination.

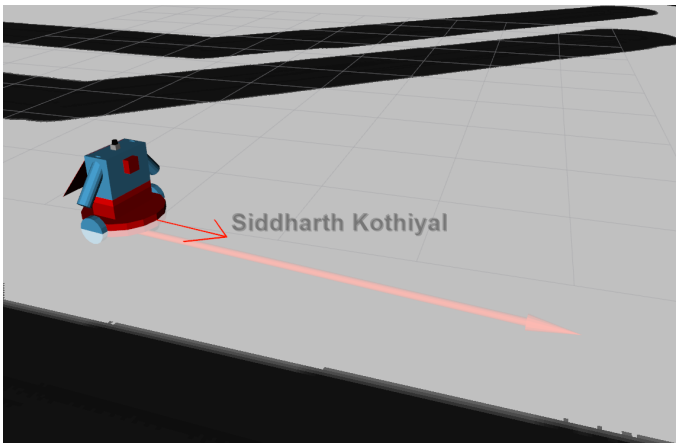

Fig. 2. Robot after reaching final location

from 2019-01-29 16-40-42.png

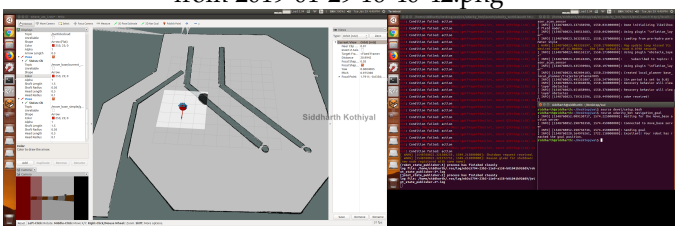

Fig. 3. Workspace after robot reached the goal

3.2 Benchmark Model

3.2.1 Model design

The benchmark model was equipped with a rectangular chassis of size 0.4, 0.2, 0.1. The chassis had two cylindrical wheels attached on either sides, in the middle of the longest side of the chassis, with radius of 0.1 and thickness of 0.05. These wheels were joined using a continuous joint. The chassis had two hemispherical casters attached on the bottom to provide balance to the chassis.

The benchmark model was also equipped with a camera sensor on the front side and a lidar on top of the chassis.

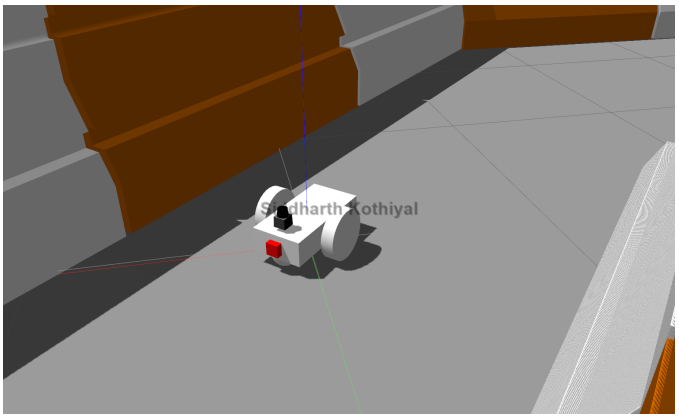

Fig. 4. Benchmark model

3.2.2 Packages Used

To control the bot ROS framework was used and Gazebo simulator and Rviz were used for visualization. ROS navigation stack was used to control the movement of the bot and AMCL package was used for localisation and path planning to the goal.

/udacity_bot/laser/scan topic was used to publish the data from the lidar and was subscribed by the AMCL package.

The /move_base/ topics were used by navigation stack and AMCL to control the bot.

3.2.3 Parameters

There are 4 files available for parameter adjustment for AMCL package and navigation stack, base_local_planner is used for configuring movements for the robot, costmap_common_params is used during generating costmap and local and global costmap param files used for generating costmap on a local and global level respectively.

During the parameter setting, first the update frequency of both local and global costmap were set, so that the machine is able to computationally handle the update frequency of the costmap generation. Then the raytrace_range and obstacle_range of the robot were changed so that the robot does not collide or get too close to obstacles while moving and get stuck.

After this point the bot was able to move around, without colliding into the walls but was not able to perform path planning properly. The height and width of the local and global costmaps were then adjusted, and it was finally able to move towards the goal. The robot would sometimes gain too much momentum and the localisation algorithm would lose confidence in its location, so the max_vel and max_acc of the robot was lowered. This increased the time the robot takes to reach the goal, but decreases the possibility of going astray randomly by accidentally confusing its location.

Finally, some tolerance was added so that after reaching the final location the bot does not take a lot of time in finding the exact orientation.

3.3 Personal Model

3.3.1 Model design

The modified bot was redesigned from scratch, the bot was given more human-like features. The base_chassis was made a thin cylinder with length 0.1 with a large radius of .3, and the same wheels were attached to two opposite ends of it, and same casters were attached to the base to provide balance to the bot. On top of the base_chassis resided the body of the bot, the whole body has a size of 0.2, 0.4, 0.45, and has been divided to give it color. There are two cylinders attached on each end of the body, to give it the feel of hands, with size 0.05 radius and 0.35 as the length of the hand. A rectangular cape is attached to the body with size 0.002, 0.35, 0.5. The camera sensor is placed in the position of the heart, whereas the lidar sensor acts as the head.

The color scheme of modified model was based on the colors of superman, making the bot a superbot. :P

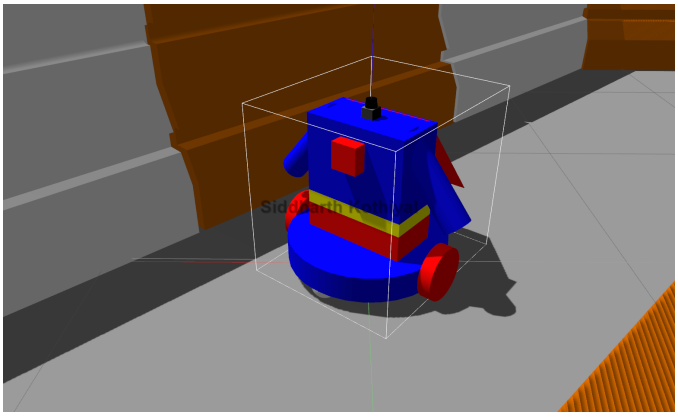

Fig. 5. Personalised model, Superbot

3.3.2 Packages Used

The packages used for the modified bot were the same as the original bot.

3.3.3 Parameters

The parameters for the benchmark bot and the modified bot remained almost the same. Only the costmap_common_params.yaml file had some changes in the obstacle and raytrace range, as the shape and size of the robots different and the location of the lidar sensor also varied a little. Since the base of the modified bot was much bigger, the obstacle_range parameter had to be increased so that the bot doesn't collide with the wall.

4 Results

The AMCL package took data coming from the lidar sensor and was successfully able to perform 2D localization on an already mapped environment. Both the robots were successfully able to reach the goal.

Right from the start the approximate location of the robot was pretty accurate as the particles were densely packed at the actual location of the robot. The particles started converging as soon as the robot started moving, and as soon as that happened the robot got on the shortest path to the goal. The overall time taken by the two robots on an average was around 1.5 minutes. The robot sometimes does get stuck or misidentifies its location, but then is able to take corrective measures in no time.

4.1 Localization Results

Both the robots showed very similar results in terms of localization, as the parameters for both the robots were very similar.

As the robot was originally positioned at the centre of the map, where there were a lot of features, the original number of particles is very high and the spread of particles was already quite certain, than if it would have been positioned in an open area. As the robot started moving towards the goal, the robot became more and more certain, and the number of particles would decrease significantly.

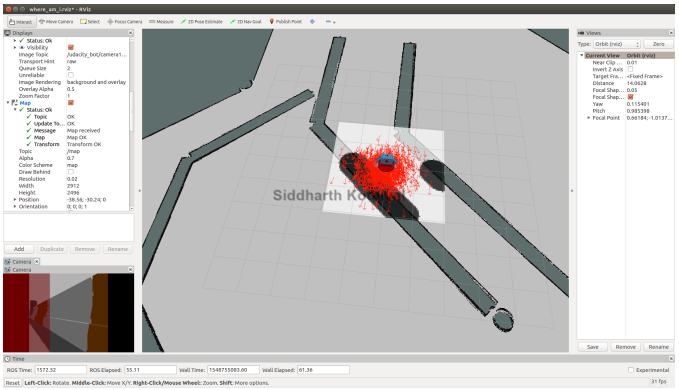

Fig. 6. Particles at the start of simulation

The number of particles is very low and most are aligned with the nav_goal arrow by the time the bot reaches the goal.

4.2 Technical Comparison

As the parameters for both the bots were very similar, both the bots were able to complete the course and reach the end

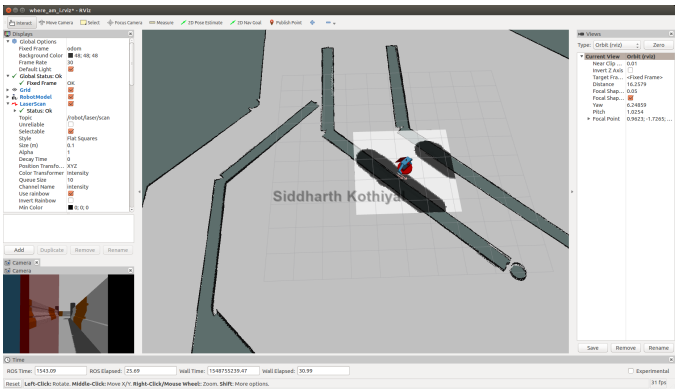

Fig. 7. Particles sometimes after the simulation started

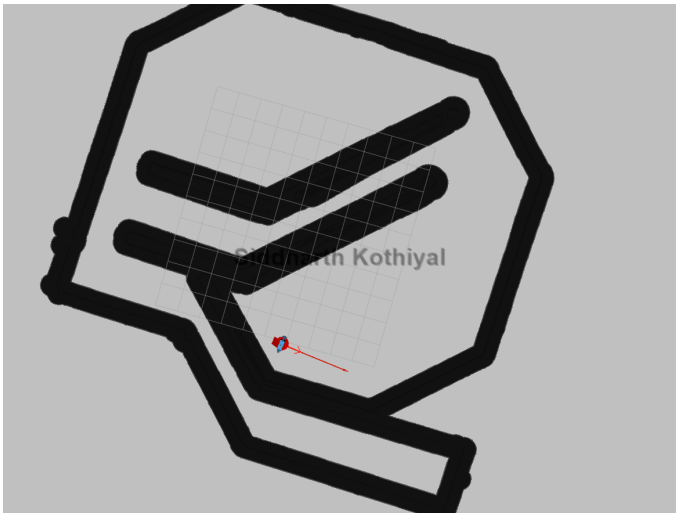

Fig. 8. Top view of robot after reaching final location

location in around 1.5 minutes. The benchmark robot was able to perform slightly better though, with on average the benchmark robot taking 10 seconds less to cover the whole course. This was because the robot was able to get closer to the walls and also had a smaller turning radius because of its smaller base size.

5 Discussion

Both the robots were able to perform localization successfully and there was not any significant difference in the localization performance of the two robots. The benchmark robot was able to complete the course a little bit faster, because of the smaller size of its base, hence it was able to get closer to the obstacles and had a smaller turning radius.

During some of the iteration of simulations, the robot would get stuck or would misidentify its current location, but AMCL was able to handle it and was able to take corrective measure. Hence proving that AMCL algorithm is able to even solve the kidnapped localisation problem. AMCL performs localization very well when it is near obstacles or features it is able to identify, but fails when it is in open environments.

Localization is almost certainly required for any autonomous system, and AMCL is versatile way of performing localization, hence it will be very useful in such scenarios.

6 Conclusion / Future work

Localisation is an important part for any autonomous system. This project shows that MCL is a very powerful technique for localization in a mapped environment. It is able to perform very well, and allows one to adjust the number of particles used for approximation in the environment hence allowing one to change the amount of memory and computation power required. This makes this algorithm suitable for robots with high computational power as well as robots which have limited resources like drones. During the course of this project, the bot would often get lost and misidentify it's location, and MCL was also able recover from that problem, hence showing that it is able to perform Kidnapped robot localisation as well.

This project can be tweaked to take data from multiple lidar sensors placed at different heights and locations, so that obstacles at different heights can also be mapped by the robot and hence avoided. Thus providing the robot ability to percieve 3D environment better.