Where Am I? (Robot Localization)

Esraa Magdy

Abstract—The localization problem is highly encountered in most of the robotics applications.In this project, Two differential mobile robots models were built and became able to localize themselves and navigate in a known environment.

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

1 Introduction

The localization problem is one of the fundamental problems that needs to be solved, in order to build an autonomous robot. To localize a robot using the data from its sensors is an operation that entails high uncertainty due to sensors' imperfections. There are several probabilistic filters aiming to solve the localization problem. The Extended Kalman Filter (EKF) and Particles filter are two of the most famous filters to solve the localization problem. In this paper, a comparison between EKF and particle filter will be discussed. Then the localization problem will be solved for two different mobile robots in simulation using AMCL ROS package.

2 Background

There are three types of the localization problem namely the local localization problem, the global localization problem and the kidnapped robot problem. In the local localization problem also known as position tracking, the robot knows its initial position and the challenge is to estimate its position as it moves through the environment. The global localization problem is more complicated than the position tracking as the robot initial pose is now unknown and the robot needs to determine its pose relative to the predefined map. The kidnapped robot problem resembles the global localization but it entails the possibility of the robot being kidnapped at any time and appear at a random position. Probabilistic filters are trying to better estimate the robot pose through multiple iterations of estimation and correction with the help of the sensors' data. In the subsequent sections, Two probabilistic filters will be presented.

2.1 Kalman Filters

Kalman Filter is an estimation algorithm that tries to estimate a certain state using series of measurements update. Kalman filter is a continuous iterations of two steps :1-Measurements update in which sensors readings are obtained and used along with the priori estimation to reduces the uncertainty in state estimation. 2-State prediction in which control actions is used to predict the state of the robot, this step usually increases the uncertainty. There are three types of Kalman filter:1- The simplest Kalman filter (KF) used for linear systems. 2-Extended Kalman filter(EKF) which works for non-linear systems. 3- Unscented Kalman filter (UKF) that is used for highly non-linear systems. The EKF and UKF extended the range of problems that Kalman filter could solve as they are not restricted to linear systems only. However, Kalman filter still posses a major drawback which is it could only be used to estimate states that could be represented by unimodal Gaussian distribution.

2.2 Particle Filters

Particles are virtual elements that resembles a robot. Each particle has a position & orientation and represents a guess of where your robot might be. Monte-Carlo localization algorithm uses the robot's sensor measurements and particles to keep track of the robot position. First, the particles are spread uniformly and randomly throughout the map. Every particle has a weight which is an indication of how well the particle represents the robot states. After several iterations of MCL and re-sampling the particles, particles will converge and better estimate the robot pose.

2.3 Comparison / Contrast

While Kalman filter is restricted to states represented by unimodal Gaussian distribution, MCL could be used to estimate states that follows any probability distribution function. MCL is also able to estimate the robot states in local/global localization problem where KF could only work for the local localization problem. An improvement to MCL known as Adaptive Monte-Carlo localization algorithm could be used to solve the kidnapped robot problem. On the other hand, the KF filter is more efficient in terms of memory needed and the time used to complete state estimation. In this paper AMCL was used to solve the localization problem for two differential drive robots.

3 Simulations

In this section, Two mobile robots models will be presented along with their respective results in the localization problem. The two mobile robots are equipped with a laser scanner and RGB Camera.However, they have different models and AMCL parameters.

3.1 Achievements

The two robots were able to reach the target location and the particles were able to converge to the actual robot pose in a short time.

3.2 Benchmark Model

3.2.1 Model design

The udacity_bot was a simple cuboid that is [0.4 0.2 0.1] in size with two driving wheels,two caster wheel, a hokuyo laser scanner and a camera.

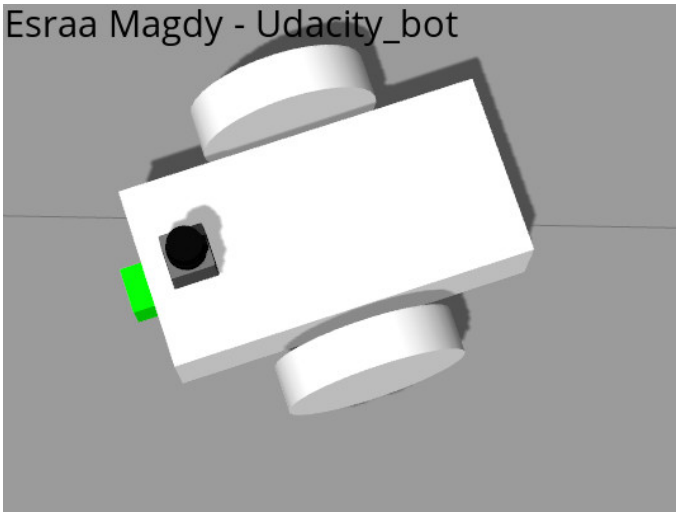

Fig. 1. udacity_bot

3.2.2 Packages Used

Two main packages were used :1-AMCL for the localization task. 2-Move_base for navigation and path planning. the AMCL package subscribed to /udacity_bot/laser/scan , tf_messages and /map topics. the move_base subscribed to /udacity_bot/laser/scan, /move_base/goal and publishes to /cmd_vel [1][2].

3.2.3 Parameters

TABLE 1
AMCL Parameters

min_particles = 10	minimum number of particles
max_particles = 100	maximum number of particles
initial_pose_x = 0	robot initial pose in x direction
initial_pose_y = 0	robot initial pose in y direction
initial_pose_a = 0	robot initial yaw angle
transform_tolerance	0.9
laser_model_type	”likelihood_field”
odom_model_type	diff-corrected

TABLE 2
move base Parameters

controller_frequency	5 Hz
robot_radius	0.1
inflation_radius	0.5
obstacle_range	0.7
raytrace_range	1.0

To tune the parameters, initial values were used as suggested by the ros_wiki page[1] and observation were made to better tune the parameters. At first the computation time was too large so the number of the particles were reduced, the global cost map width/height were reduced to 30 meters and the local cost map width/height were reduced to 5.0 meters. A persistent warning was noticed to reoccur regarding the Control loop frequency. From the warning it seemed that the control loop frequency is less than 20 Hz which is the default value, to resolve this issue the controller_frequency was set to 5 Hz.Other parameters like the inflation_radius , obstacle_range and raytrace_range were tuned to make sure the robot does not hit the wall and better notice the obstacles.

3.3 Personal Model

3.3.1 Model design

my_bot model was a simple differential mobile robot with cylindrical base that is 0.15 m in radius and 0.02 m length. it is also equipped with a hokuyo laser scanner and RGB Camera.

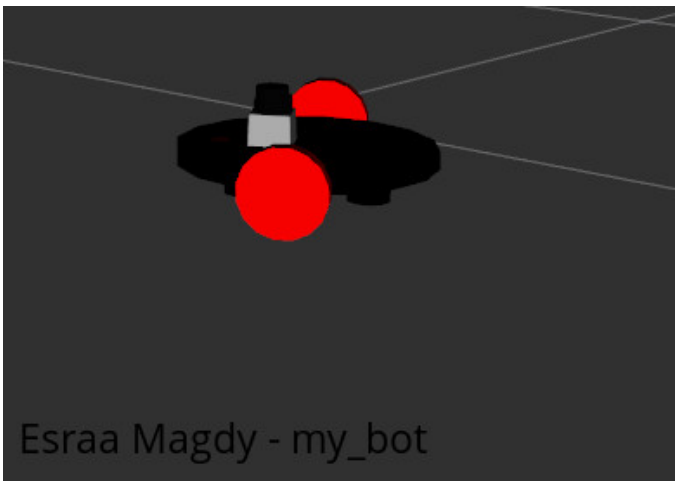

Fig. 2. my_bot

3.3.2 Packages Used

The same packages used for udacity_bot was used again for my_bot model but a slight change in the topics' names were made. for example, /udacity_bot/laser/scan changed to /my_bot/laser/scan.

3.3.3 Parameters

At first the parameters used were the same as that for udacity_bot but the behavior was not the same.For example, the bot would spin around itself or hit the walls. So, the parameters was tweaked a bit to better accommodate for the model change.

TABLE 3
move base Parameters

controller_frequency	4 Hz
robot_radius	0.15
inflation_radius	2.0
obstacle_range	0.7
raytrace_range	1.0

TABLE 4
AMCL Parameters

min_particles = 10	minimum number of particles
max_particles = 100	maximum number of particles
initial_pose_x = 0	robot initial pose in x direction
initial_pose_y = 0	robot initial pose in y direction
initial_pose_a = 0	robot initial yaw angle
laser_model_type	”likelihood_field”
odom_model_type	diff-corrected
transform_tolerance	0.5

4 Results

The udacity_bot would take a moderate amount of time to reach the goal position. Sometimes, it stops by the wall and it takes a minute or two to recover from that position and continue to reach the target. On the other hand, the particles converges smoothly to the actual robot pose in a short time after the robot starts moving.
For my_bot, it reaches the target position faster than udacity_bot.However, sometime it oscillates around itself while moving. As for AMCL, the particles converges to the robot pose but the number of particles at the goal position is larger than that of udacity_bot which indicate higher uncertainty.

4.1 Localization Results

4.1.1 Benchmark

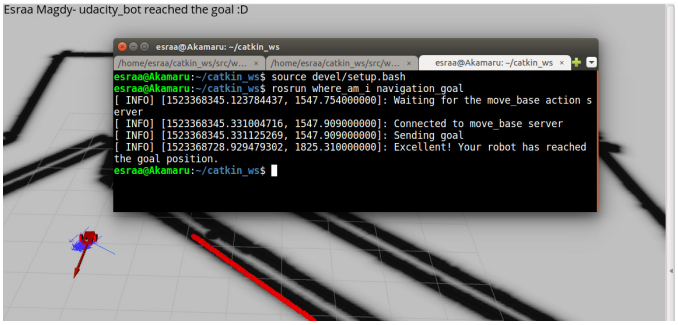

Fig. 3. udacity_bot a the goal

4.1.2 Student

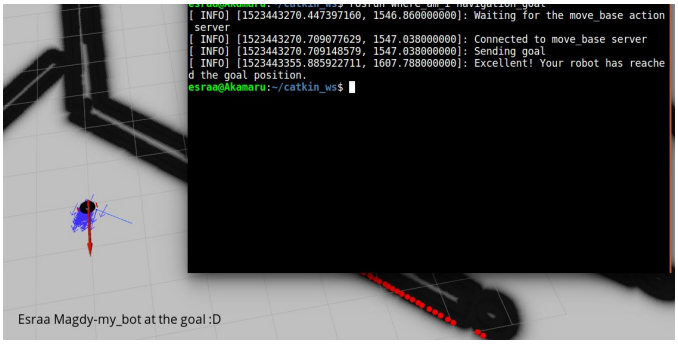

Fig. 4. my_bot a the goal

4.2 Technical Comparison

Both the udacity_bot and my_bot have the same trajectory planned for them. my_bot experienced oscillatory behavior and its movement was a bit unstable but it could follow the path and reach the goal. The results for AMCL were almost the same.

5 Discussion

The two robots performed their required tasks in a decent amount of time. They were able to follow the path planned by move_base, reach the target goal while localizing themselves in the map. the my_bot movement could be characterized as unstable which might be a result for its low inertia so a small disturbance leads to oscillations. the udacity_bot performs better but it takes a long time to reach the goal.

5.1 Topics

- Which robot performed better? Both robots did their required tasks in following the planned path and localizing themselves.
- Why it performed better? Different parameters tuning lead them to have similar responses.
- How would you approach the ’Kidnapped Robot’ problem? To approach the kidnapped robot, One might think of adding more on-board sensors to reduce the uncertainty in the sensors readings.Also increasing the number of particles while computationally expensive but could help reduce the uncertainty.
- What types of scenario could localization be performed? It could be used for indoor environment where the actual map doesn’t change from the one provided to the robot.
- Where would you use MCL/AMCL in an industry domain? In indoor environment, maybe to deliver something from one place to another one inside a factory.

6 Conclusion / Future work

In this project, a comparative analysis between two probabilistic filters namely KF and particle filter were discussed. An attempt to solve the localization problem for two different mobile robots were made and the two robots were able to successfully reach a specified target location while localizing themselves in a predefined map. To deploy this project on a hardware, a map of the environment in which the robot will work in needs to be provided, also the specifications of the actual robot needs to be known to tune the parameters for it.

For Future Work, the parameters might be tuned further to reach better performance. the my_bot robot’s dimensions and inertia might change a bit to withstand disturbances. The robots might be tested in different environments to experiment how that will affect the parameters.

6.1 Modifications for Improvement

6.2 Hardware Deployment

1) What would need to be done? Survey about the working environment, its friction coefficient for example to build a robot model that withstand it. Get the map of the environment and the robot model to be tested first in a simulation environment and tune the AMCL and move_base parameters.
2) Computation time/resource considerations? The project could be run on 2GB RAM board as was the case on the VM.

REFERENCES

[1] [5]"navigation/Tutorials/RobotSetup - ROS Wiki", Wiki.ros.org, 2018. [Online]. Available: http://wiki.ros.org/navigation/Tutorials/RobotSetup#Costmap_Configuration_.28local_costmap.29_.26_.28global_costmap.29. [Accessed: Apr- 2018].
[2] "amcl - ROS Wiki", Wiki.ros.org, 2018. [Online]. Available: http://wiki.ros.org/amcl#Parameters. [Accessed: Apr- 2018].