Robot Localization Using ROS AMCL Package

Muthanna A. Attyah

Abstract—Robot localization is the process of determining where a mobile robot is located with respect to its environment. Localization is one of the most fundamental competencies required by an autonomous robot as the knowledge of the robot's own location is an essential precursor to making decisions about future actions. [1] In this paper **Extended Kalman Filter (EKF)** localization algorithm will be compared to **Adaptive Monte Carlo localization (AMCL)** algorithm then AMCL performance will be tested in two different simulated robot configurations.

Index Terms—Robot, IEEEtran, Udacity, Localization, , AMCL, EKF, ROS

1 Introduction

NAVIGATION is one of the most challenging competencies required for a mobile robot; without it a mobile robot will not be able to achieve its tasks. Success in navigation requires four building blocks:

- Perception to detect environment.
- Localization to determine position.
- Cognition to decide on achieving goals.
- Motion control to achieve desired trajectory.

Of these four components, localization has received the greatest research attention and, as a result, significant advances have been made on this front. [2] Localization is basically the answer to the question of "Where am I ?" and there are many algorithms that can be used to answer this question; examples are Histogram Filter, Information Filter, Triangulation, Markov Filter, Kalman Filter (KF), Extended Kalman Filter (EKF), and Particles Filter or Monte Carlo Localization (MCL), and Adaptive Monte Carlo Localization (AMCL). [3] [4] Different algorithms has different approaches; linear or not, probabilistic or not, 1D, 2D, 3D, taking single sensor or more, high computing requirement or less, etc. In this paper focus will be on comparing EKF to AMCL then test the performance of AMCL on two different simulated robots.

2 Background

The focus of this project will be on probabilistic 2D filters that can handle non-linear sensor data with jerky errors. to better understand these filters, a high-level description of each type is given below:

2.1 Markov localization

Probabilistic localization algorithms are variants of the Bayes filter. The straightforward application of Bayes filters to the localization problem is called Markov localization. Markov localization, uses an explicitly specified probability distribution across all possible robot positions. [2]

2.2 Kalman Filters

A second method, Kalman filter (KF) localization, is a linear quadratic estimator that uses a Gaussian probability density representation of robot position and scan matching for localization. Unlike Markov localization, Kalman filter localization does not independently consider each possible pose in the robots configuration space. Interestingly, the Kalman filter localization process results from the Markov localization axioms if the robots position uncertainty is assumed to have a Gaussian form. [2] The disadvantage of KF is the assumption that motion and measurements are linear, and the assumption that state space can be represented by a unimodal Gaussian distribution. Both assumptions are not true in most of robots hence a non-linear algorithm will be a better fit for mobile robots. [3]

2.3 Extended Kalman Filters

The extended Kalman filter localization algorithm, or EKF localization, is a special case of Markov localization which will resolve the issues of KF by handling non-linear motion and measurements. EKF localization represent beliefs by their first and second moment, that is, the mean and the covariance. [3] [4]

2.4 Particle Filters

A Particle filter is another great choice for localization if a robot system doesn't fit nicely into a linear model, or a sensor uncertainty doesn't look very Gaussian. A Particle Filter make it possible to handle almost any kind of model, by discretizing the problem into individual "particles" – each one is basically one possible state of your model, and a collection of a sufficiently large number of particles lets you handle any kind of probability distribution, and any kind of evidence (sensor data).the estimation error in a particle filter does converge to zero as the number of particles (and hence the computational effort) approaches infinity. Particle filter is also called Monte Carlo Localization (MCL). . [5]

2.5 Comparison / Contrast

While Kalman filter can be used for linear or linearized processes and measurement system, the particle filter can

be used for nonlinear systems. The uncertainty of Kalman filter is restricted to Gaussian distribution, while the particle filter can deal with non-Gaussian noise distribution. In cases where abrupt sensor noise is rarely observed, both filters work fairly well. However, when sensor noise exhibits jerky error, Kalman filter results in location estimation with hopping while particle filter still produces robust localization. The paper also compares performance of these filters under various measurement uncertainty and process uncertainty. [6] Following table summarize the differences between Extended Kalman Filter (EKF) and Particle Filter (MCL).

Item	MCL	EKF
Measurement	Raw Measurement	Landmarks
Measurement Noise	Any	Gaussian
Posterior	Particles	Gaussian
Efficiency - Memory	✔	✔✔
Efficiency - Time	✔	✔✔
Easy of Implementation	✔✔	✔
Resolution	✔	✔✔
Robustness	✔✔	N.A.
Memory/Resolution Control	Yes	No
Global Localization	Yes	No
State Space	Multimodal Discrete	Unimodal Continuous

TABLE 1: Comparing MCL with EKF

When good computing capability is available for the mobile robot, MCL filter will be a good choice for localization. Two types of simulated robots will be used to test the performance of an Adaptive MCL filter (AMCL) in ROS environment.

3 Simulations

Next sections will describe the design details of the two simulated robots used to test performance of AMCL package in ROS/Gazebo/Rviz environment.

3.1 Achievements

You should describe what you achieved for localization in the project with the benchmark model and your own model. Includes charts and graphs show how parameters affect your performance.

3.2 World Map

Jackal Race world map will be used in simulation; this map was originally created by Clearpath Robotics.

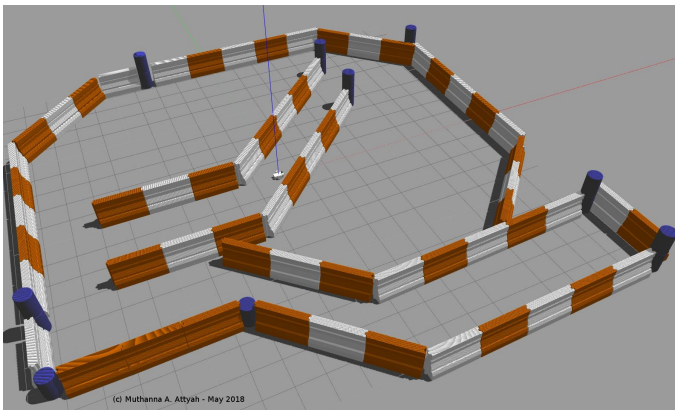

Fig. 1: Jackal Race Map

3.3 Benchmark Model

1st simulated robot (**udacity_bot**) is the benchmark model that was provided as part of udacity project.

3.3.1 Model design

Benchmark robot **udacity_bot** design includes the following components:

Item	Type	Size
robot_footprint	Link	N.A.
robot_footprint_joint	joint - fixed	N.A.
chassis	Link	box: 0.4 x 0.2 x 0.1
back_caster_visual	visual	sphere: r=0.05
front_caster_visual	visual	sphere: r=0.05
left_wheel	Link	cylinder: r=0.1 l=0.05
left_wheel_hinge	joint - continuous	N.A.
right_wheel	Link	cylinder: r=0.1 l=0.05
right_wheel_hinge	joint - continuous	N.A.
camera	Link	box : 0.05 x 0.05 x 0.05
camera_joint	joint - fixed	N.A.
hokuyo	Link	mesh : 0.1 x 0.1 x 0.1
hokuyo_joint	joint - fixed	N.A.

TABLE 2: udacity_bot design

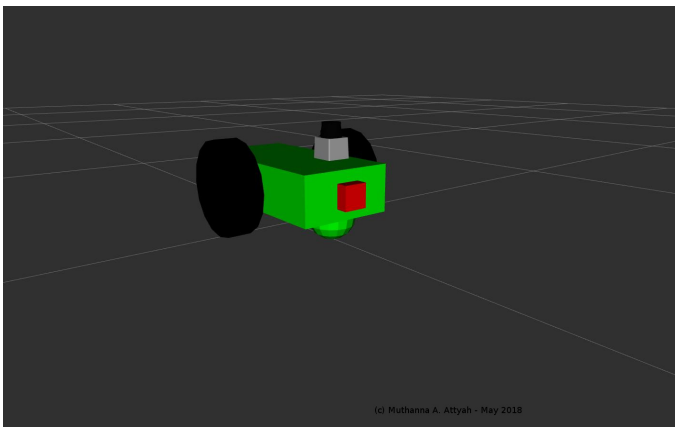

Fig. 2: udacity_bot in RViz

3.3.2 Packages Used

Following Packages were used in ROS simulated robots and environment:

map_server: provides the map_server ROS Node, which offers map data as a ROS Service. [7]

move_base: provides an implementation of an action that, given a goal in the world, will attempt to reach it with a mobile base. The move_base node links together a global and local planner to accomplish its global navigation task. The move_base node also maintains two costmaps, one for the global planner, and one for a local planner that are used to accomplish navigation tasks. [7]

joint_state_publisher: contains a tool for setting and publishing joint state values for a given URDF. [7]

robot_state_publisher: allows to publish the state of a robot to tf. Once the state gets published, it is available to all components in the system that also use tf. The package takes the joint angles of the robot as input and publishes the 3D poses of the robot links, using a kinematic tree model of the robot. [7]

amcl: a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map. [7]

gazebo_ros: provides ROS plug-ins that offer message and service publishers for interfacing with Gazebo through ROS. [7]

rviz: 3D visualization tool for ROS. [7]

Other than what is described in next subsection, all topics published, topics subscribed, services, and parameters of above packages are as described on ROS wiki [7], there are no customized service, topics, messages, or parameters used in this package.

3.3.3 Gazebo plug-ins

libgazebo_ros_diff_drive.so: provides a basic controller for differential drive robots in Gazebo.

libgazebo_ros_camera.so: provides ROS interface for simulating cameras.

libgazebo_ros_laser.so: simulates laser range sensor by broadcasting LaserScan message as described in sensor_msgs.

3.3.4 Parameters

Parameters used to fine tune localization performance are divided into 5 yaml formatted configuration files:

1) amcl.yaml
2) base_local_planner_params.yaml
3) costmap_common_params.yaml
4) global_costmap_params.yaml
5) local_costmap_params.yaml

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. [8]

3.4 Personal Model

2nd simulated robot (**muth_bot**) is designed with different configuration than the benchmark robot to allow testing of related AMCL parameters.

3.4.1 Model design

Personal robot **muth_bot** includes the following components:

Item	Type	Size
robot_footprint	Link	N.A.
robot_footprint_joint	joint - fixed	N.A.
chassis	Link	box: 0.4 x 0.2 x 0.15
front_left_wheel	Link	cylinder: r=0.1 l=0.05
front_left_wheel_ hinge	joint - continuous	N.A.
front_right_wheel	Link	cylinder: r=0.1 l=0.05
front_right_wheel_ hinge	joint - continuous	N.A.
back_left_wheel	Link	cylinder: r=0.1 l=0.05
back_left_wheel_ hinge	joint - continuous	N.A.
back_right_wheel	Link	cylinder: r=0.1 l=0.05
back_right_wheel_ hinge	joint - continuous	N.A.
camera	Link	box : 0.05 x 0.05 x 0.05
camera_joint	joint - fixed	N.A.
hokuyo	Link	mesh : 0.1 x 0.1 x 0.1
hokuyo_joint	joint - fixed	N.A.

TABLE 3: muth_bot design

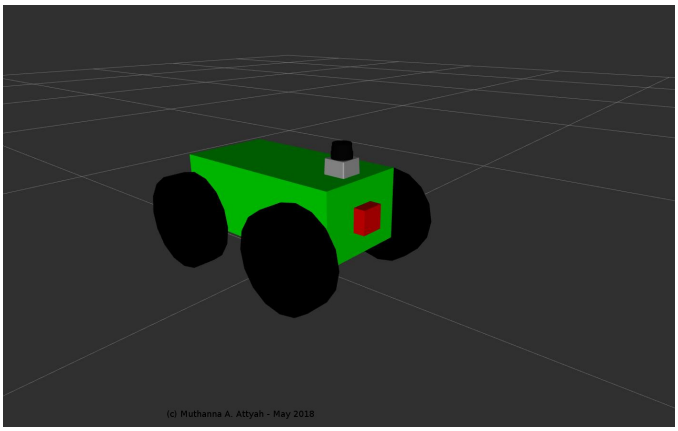

Fig. 3: muth_bot in RViz

3.4.2 Packages Used

Packages used in the 2nd robot is the same as listed in the 1st robot.

3.4.3 Gazebo plug-ins

skid_steer_drive_controller: plug-in was used to test skid steering.

object_controller: plug-in was used to test planner move (omni directional move).

libgazebo_ros_camera.so: provides ROS interface for simulating cameras.

libgazebo_ros_laser.so: simulates laser range sensor by broadcasting LaserScan message as described in sensor_msgs.

3.4.4 Parameters

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 Robot udacity_bot

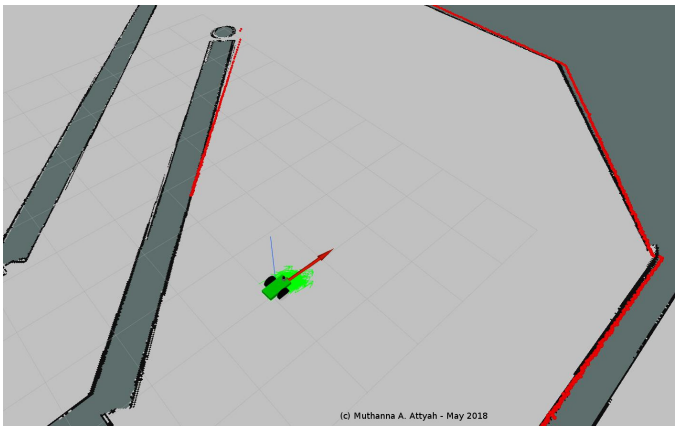

Fig. 4: udacity_bot reached target

4.1.2 Student Robot muth_bot

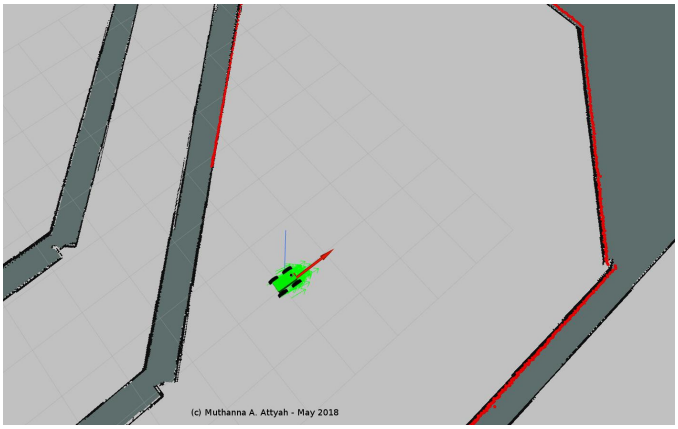

Fig. 5: Muth_bot reached target

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.

6.1 Modifications for Improvement

Examples:

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

6.2 Hardware Deployment

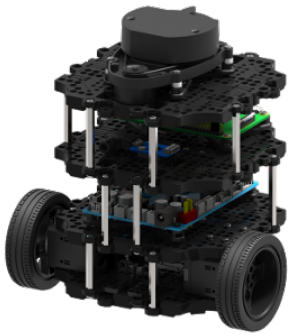

Fig. 6: TurtleBot 3

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

REFERENCES

[1] S. H. G. Dissanayake, "Robot localization: An introduction," *Wiley Encyclopedia of Electrical and Electronics Engineering*, 15 August 2016.
[2] I. R. N. Roland Siegwart, *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.
[3] K. M.-M. Ling Chen, Huosheng Hu, "Ekf based mobile robot localization," 2012.
[4] D. F. Sebastian Thrun, Wolfram Burgard, *Probabilistic Robotics*. MIT Press, 2005.
[5] J. Reich, "What is the difference between a particle filter and a kalman filter?."
[6] T. G. K. Nak Yong Ko, "Comparison of kalman filter and particle filter used for localization of an underwater vehicle," 19 February 2013.
[7] "Ros wiki web site."
[8] Gilbert, "Ros wiki: Amcl package summary."