Where Am I

Mohamed Sayed Antar

Abstract—by building two mobile robots using Unified Robot Description Format which is an XML format for representing a robot model for Gazebo and RViz environment then by Creating a ROS package to launch these models in Gazebo and RViz and using packages like AMCL and the Navigation Stack for localizing the robot in a provided map, then by Exploring, adding, and tuning specific parameters corresponding to each package to achieve the best possible localization results.

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

1 Introduction

LOCALIZATION involves one question: Where is the robot now? Or, robo-centrically, where am I, keeping in mind that ”here” is relative to some landmark (usually the point of origin or the destination) and that you are never lost if you don’t care where you are. Although a simple question, answering it isn’t easy, as the answer is different depending on the characteristics of your robot. Localization techniques that work fine for one robot in one environment may not work well or at all in another environment. For example, localizations which work well in an outdoors environment may be useless indoors. All localization techniques generally provide two basic pieces of information:

- what is the current location of the robot in some environment?
- what is the robot’s current orientation in that same environment?

The first could be in the form of Cartesian or Polar coordinates or geographic latitude and longitude. The latter could be a combination of roll, pitch and yaw or a compass heading. Consider a robot with an internal map of its environment. When the robot moves around, it needs to know where it is within this map. Determining its location and rotation (more generally, the pose) by using its sensor observations is known as robot localization.

2 Background

localization is a core element for the mobile robots, several approaches to localize robots pose like Kalman Filter, Markov Localization, Grid Localization, Monte Carlo Localization.

2.1 Kalman Filters

also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe. The filter is named after Rudolf E. Klmn, one of the primary developers of its theory.

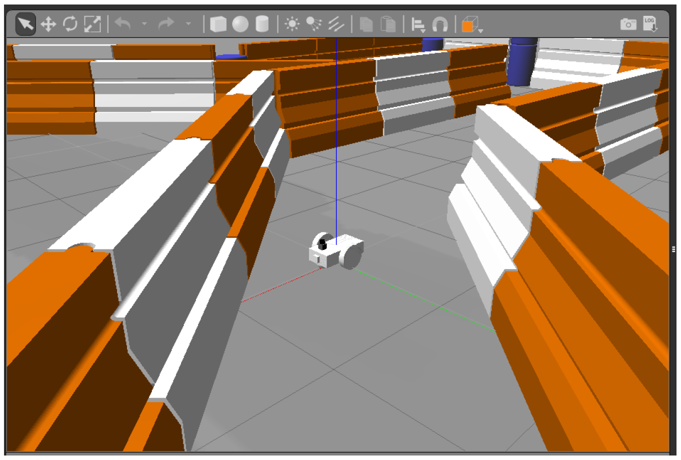


Fig. 1. the Robot in Gazebo

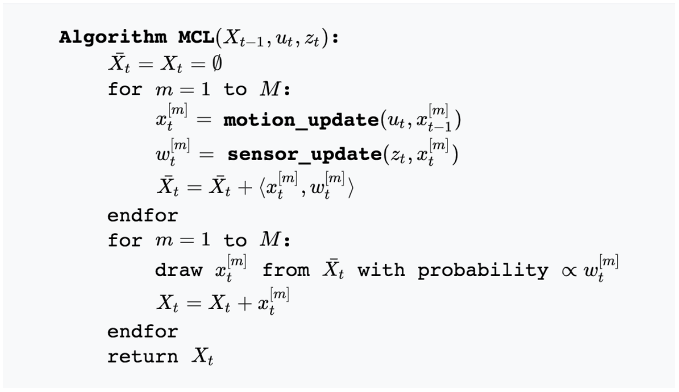
2.2 Particle Filters

Monte Carlo localization (MCL), also known as particle filter localization, is an algorithm for robots to localize using a particle filter. Given a map of the environment, the algorithm estimates the position and orientation of a robot as it moves and senses the environment. The algorithm uses a particle filter to represent the distribution of likely states, with each particle representing a possible state, a hypothesis of where the robot is. The algorithm typically starts with a uniform random distribution of particles over the configuration space, meaning the robot has no information about where it is and assumes it is equally likely to be at any point in space. Whenever the robot moves, it shifts the particles to predict its new state after the movement. Whenever the robot senses something, the particles are resampled based on recursive Bayesian estimation, how well the actual sensed data correlate with the predicted state. Ultimately, the particles should converge towards the actual position of the robot.

during this project the AMCL package will be used to localize the robot in the provided map.

3 Simulations

Building a mobile robot for localization task and Creating a ROS package that launches this robot model in a Gazebo



```
Algorithm MCL($X_{t-1}, u_t, z_t$):
    $\bar{X}_t = X_t = \emptyset$
    for $m = 1$ to $M$:
        $x_t^{[m]}$ = motion_update($u_t, x_{t-1}^{[m]}$)
        $w_t^{[m]}$ = sensor_update($z_t, x_t^{[m]}$)
        $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
    endfor
    for $m = 1$ to $M$:
        draw $x_t^{[m]}$ from $\bar{X}_t$ with probability $\propto w_t^{[m]}$
        $X_t = X_t + x_t^{[m]}$
    endfor
    return $X_t$
```

Fig. 2. MCL Algorithm

world and utilizes packages like AMCL and the Navigation Stack, trying to achieve the best possible localization results by Exploring, adding, and tuning specific parameters corresponding to each package.

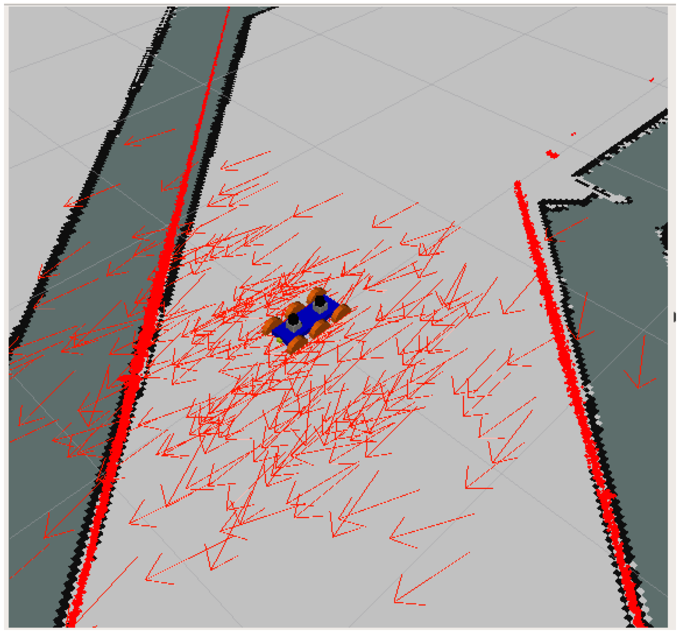


Fig. 3. my Robot in RViz

3.1 Achievements

several steps to achieve the requirements, the details for how these configurations done please visit my repository on github

- by navigating to the src directory and creating a new empty package.
- Next, create folders, launch and worlds, that will further define the structure of your package.
- then create udacity world, with no objects or models that will be launched later in Gazebo.
- we will create a launch file. Launch files in ROS allow us to execute more than one node simultaneously, which helps avoid a potentially tedious task of defining and launching several nodes in separate shells or terminals.
- create a new folder in your package directory and an empty xacro file for the robot's URDF description.
- Create a new launch file that will help load the URDF file.
- then Gazebo Plugins
- create the maps folder
- copy the two files needed for the maps inside the maps folder
- launch your robot model run the following

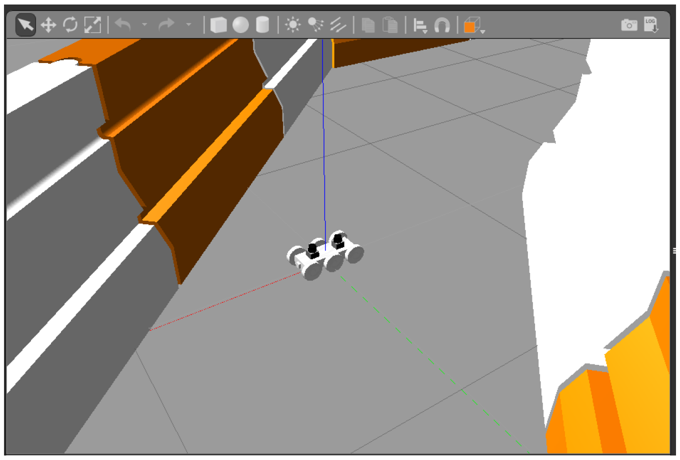


Fig. 4. my Robot in Gazebo

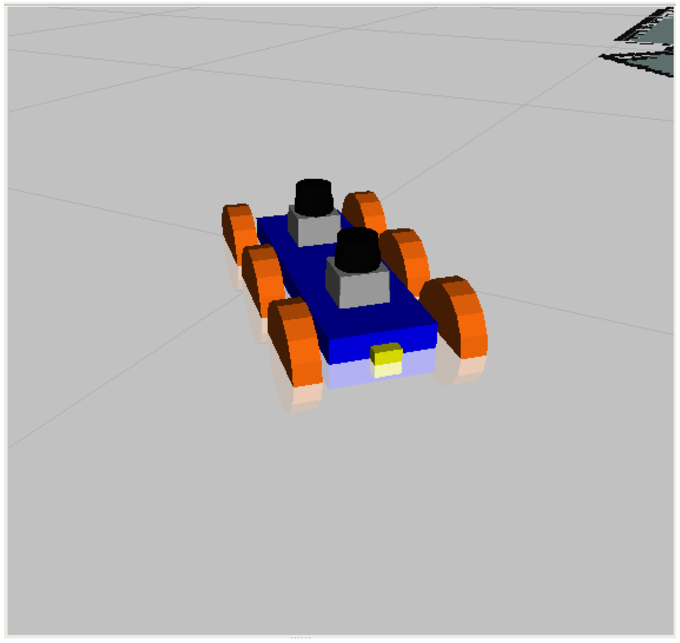


Fig. 5. my Robot

3.2 AMCL

Adaptive Monte Carlo Localization (AMCL) dynamically adjusts the number of particles over a period of time, as the robot navigates around in a map. This adaptive process offers a significant computational advantage over MCL.

The ROS amcl package implements this variant and you will integrate this package with your robot to localize it inside the provided map. do the following

- creating a new launch file for amcl
- create the config folder for the configuration files.
- copy the four files inside the config folder from the github repo to your config folder.

3.3 Packages Used

- AMCL
- move base

3.4 Parameters

TABLE 1
parameters table

update$_{f}requencyforlocal$	20.0
publish$_{f}requencyforlocal$	25.0
width for local	3.5
height for local	3.5
resolution for local	0.05
update$_{f}requencyforglobal$	15.0
publish$_{f}requencyforglobal$	20.0
width for global	500
height for global	500
resolution for global	0.05
sim$_{t}ime$	7.5
obstacle$_{r}ange$	1
raytrace$_{r}ange$	5
transform$_{t}olerance$	3.5
inflation$_{r}adius$	0.7
min$_{p}articles$	100
max$_{p}articles$	200
update$_{m}in_{d}$	0.005
update$_{m}in_{a}$	0.01

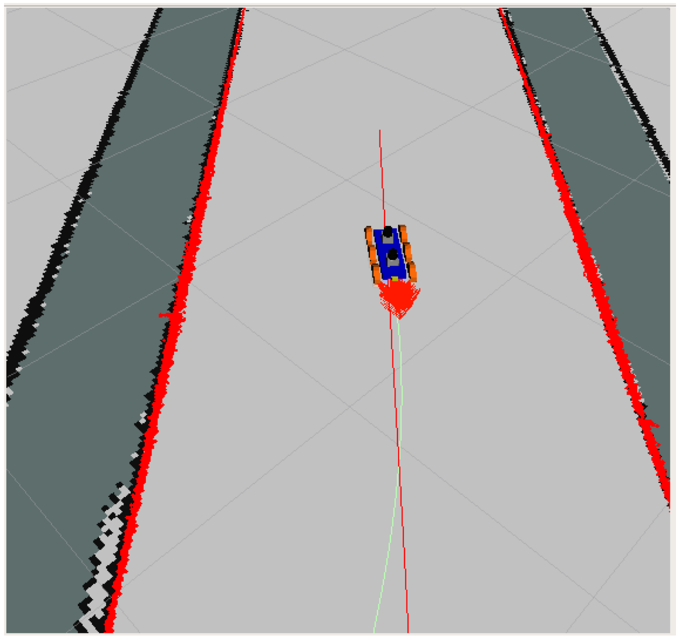


Fig. 6. my Robot

4 Results

the two robots reach to their goal successfully without any collision with any of the barriers, both of the robots almost

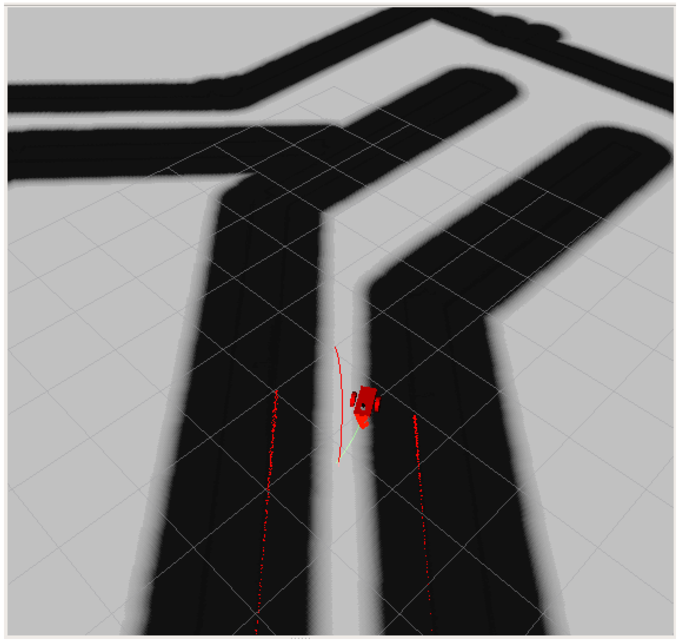


Fig. 7. my Robot

have the same performance and reach the goal almost with the same time.

below some photos showing the moment of arrival for each robot and during moving to the goal.

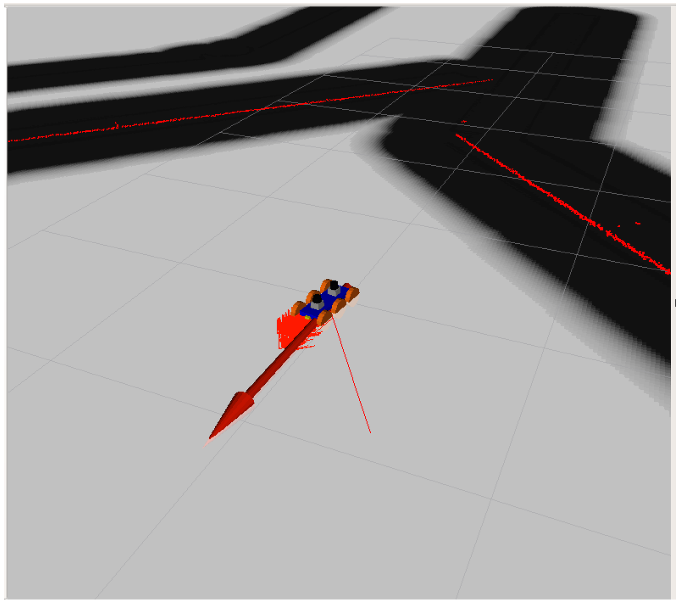


Fig. 8. the Robot in Gazebo

5 Discussion

the two robots reached their goal and successfully determined their location and orientation, although the two robots have different number of joints and links and one of them has two laser sensors and the other has only one they eventually reach thier goal at almost the same time, this happens due to same parameters for both robots wich

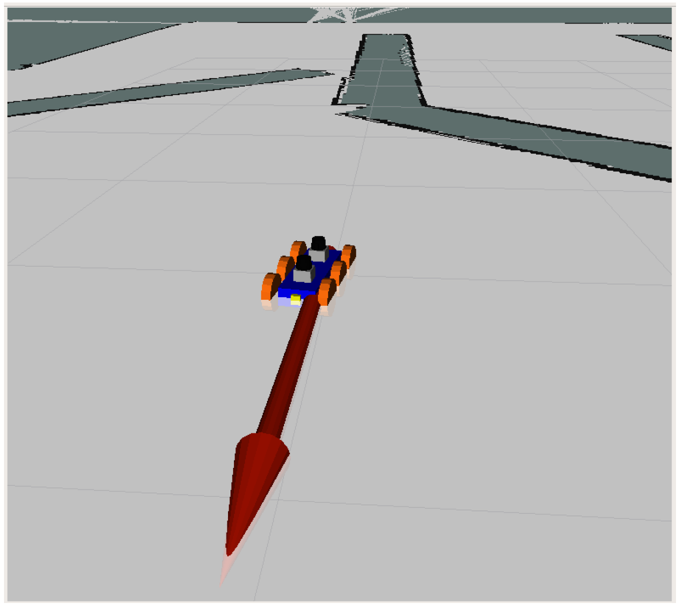
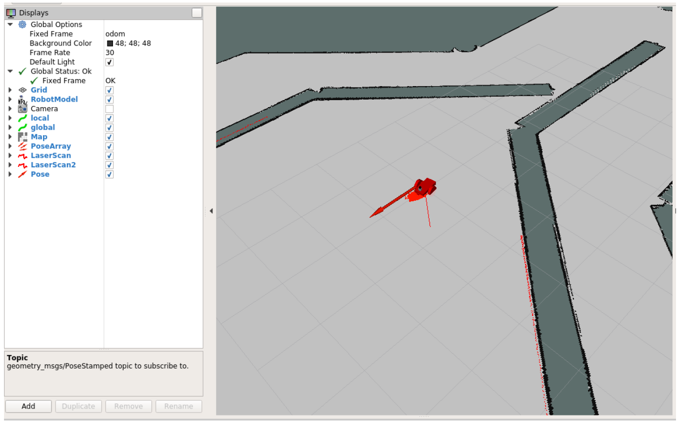


Fig. 9. the Robot in Gazebo



```
Displays
▾ Global Options
    Fixed Frame          odom
    Background Color     ■ 48; 48; 48
    Frame Rate           30
    Default Light        ✓
▾ ✓ Global Status: Ok
    ✓ Fixed Frame        OK
▸ Grid                   ✓
▸ RobotModel             ✓
▸ Camera                 ☐
▸ local                  ✓
▸ global                 ✓
▸ Map                    ✓
▸ PoseArray              ✓
▸ LaserScan              ✓
▸ LaserScan2             ✓
▸ Pose                   ✓

Topic
geometry_msgs/PoseStamped topic to subscribe to.
```

Fig. 10. the Robot in Gazebo

lead to almost no change in their performance, maybe their is a slight change but this can be overcome by modifying the config parameters.

for the kidnapped robot problem the mcl may give a good results and may give bad results according to the number of nearby particles, may we can solve this problem by increasing the number of spread particles on the map or by refreshing the particles pose periodically, many other approaches to solve this problem may lead to better results, also these robots will work fine for static maps in industry like cleaning robots in home or factories.

6 Conclusion / Future work

tuning the parameters in the four files inside config folder may lead to better performance specially the running time and the speed of robot movement on the right path, but for parameters in the amcl file tunning it may lead to small change in the performane because the pose already estimated with good accuracy.

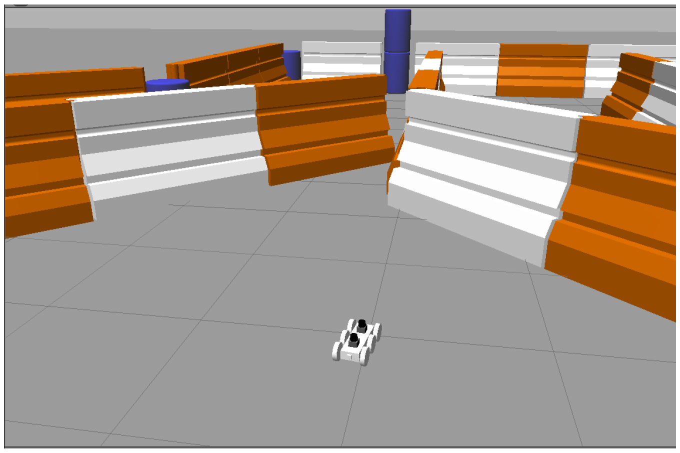


Fig. 11. the Robot in Gazebo

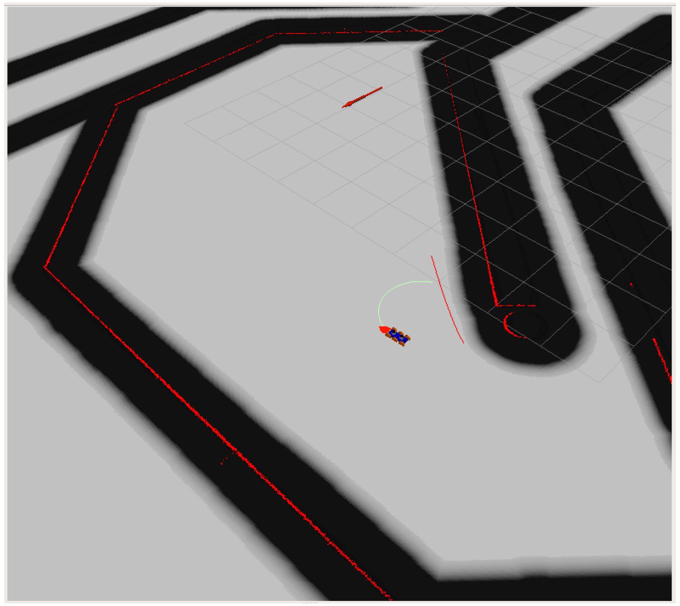


Fig. 12. the Robot in Gazebo