Where Am I project

Mohamed Ramzy

Abstract—Localization problem is a crucial problem in the field of Robotics. Many Robotics applications requires that the robot is capable of localizing its place in the environment. In this project a robot model is built and ACML algorithm is utilized to localize it in a world. The algorithm is applied on 2 models. The one that udacity provides and a new robotic model with a lot of changes. ACML parameters is tuned to give the best possible performance during the implementation.

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

1 Introduction

In the project we apply AMCL algorithm to solve the robot localization problem where a robot is in an environment and it is required to navigate to a particular navigation goal, This can be done by accurately localizing the robot using AMCL and navigation through the map. A robotic model is built and based on it the parameters of AMCL algorithm is set to achieve to less localization uncertainty.

2 Background

As mentioned above localization is crucial task for many robotics applications, As many tasks requires to accurately estimate the pose of the robot. Localization has many types, The local localization problem which the initial pose of the robot is known and it is required to estimate the pose while the robot moves. Another type of localization problems is the global localization problem where the initial pose of the robot is unknown unlike the local localization one. Also the kiddnapped robot problem that the robot's pose is suddenly changed and it is required to keep track of its location. Many localization algorithms exist. Kalman Filters and Particle Filters will be explained in the next sections.

2.1 Kalman Filters

Kalman filters is considered one of the most important localization algorithms. It works by modelling the movement and measurements with gaussian distributions. It consists of 2 steps, The first one is measurement update where the measurements' mean and standard deviation is updated from the current action, And then the state of the robot is updated. linear kalman filters has drawbacks as it works only with linear quantities, EKFs work in multiple dimensions and can model non linear functions so that won't be an obstacle.

2.2 Particle Filters

Monte carlo localization or Particle Filter is another localization algorithm that initializes the space with virtual particles and assigns a weight for everyone of them and then it updates the belief of the robot location, It is useful because it is easy to implement and can model distributions other than gaussian distributions

2.3 Comparison / Contrast

As shown above kalman filters and particle filters are so powerful but both of them has its strengths and weaknesses. kalman filters are more memory and time efficient than particle filters, So when we care about the running time and memory we can use kalman filters. But it has limitations as kalman filters works only with gaussian distributions unlike particle filters which work with any distribution, Also it doesn't support memory and control resolution, Also it is harder to implement than particle filters. The work in this project will be using particle filters

3 Model configuration

Here the model configuration is discussed, I changed the chassis from a box to cylinder with length 0.1 and radius 0.125, Also i changed changed the pose of the caster wheels to align with the new change to -0.1 0 -0.05 (xyz) and 0.1 0 -0.05 (xyz), Also i changed the position of the camera and laser sensor with -0.07 0 0.01 (xyz) and -0.05 0 -0.03 (xyz). Concerning the other parameters for the AMCL algorithm is as follows:

obstacle range	1.5
raytrace range	1.5
transform tolerance	0.5
inflation radius	0.5
update frequency	10
publish frequency	10

The other parameters for the localization algorithm is states in Figure 1

According to the explaination of parameters from the unit notes in udacity :

Concerning the obstacle range, raytrace range and transform tolerance it was set to give the best performance in reaching the goal. For example, if set to 0.1, that implies that if the obstacle detected by a laser sensor is within 0.1 meters from the base of the robot, that obstacle will be added to the costmap. Tuning this parameter can help with discarding noise, falsely detecting obstacles, and even with computational costs.
raytrace range This parameter is used to clear and update the free space in the costmap as the robot moves.

inflation radius This parameter determines the minimum distance between the robot geometry and the obstacles. Try setting a really high value for this parameter, and launch the project and select the global costmap selected. You will notice that the obstacles (the walls of the environment) seem to be ”inflated” as can be seen below. An appropriate value for this parameter can ensure that the robot smoothly navigates through the map, without bumping into the walls and getting stuck, and can even pass through any narrow pathways.

Overall Filter

min particles and max particles - As amcl dynamically adjusts its particles for every iteration, it expects a range of the number of particles as an input. Often, this range is tuned based on your system specifications. A larger range, with a high maximum might be too computationally extensive for a low-end system.
initial pose - For the project, you should set the position to [0, 0]. Feel free to play around with the mean yaw value.
update min* - amcl relies on incoming laser scans. Upon receiving a scan, it checks the values for update min a and update min d and compares to how far the robot has moved. Based on this comparison it decides whether or not to perform a filter update or to discard the scan data. Discarding data could result in poorer localization results, and too many frequent filter updates for a fast moving robot could also cause computational problems.

Concerning the laster There are two different types of models to consider under this - the likelihood field and the beam. Each of these models defines how the laser rangefinder sensor estimates the obstacles in relation to the robot.

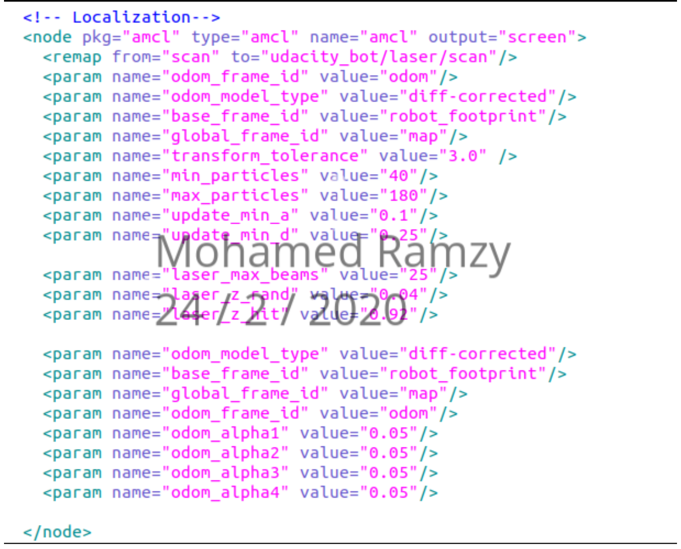
The likelihood field model is usually more computationally efficient and reliable for an environment such as the one you are working with. So you can focus on parameters for that particular model such as the -

laser * range
laser max beams
laser z hit and laser z rand

Tuning of these parameters will have to be experimental. While tuning them, observe the laser scan information in RViz and try to make sure that the laser scan matches or is aligned with the actual map, and how it gets updated as the robot moves. The better the estimation of where the obstacles are, the better the localization results.

Concerning the odometry model type - Since you are working with a differential drive mobile robot, it’s best to use the diff-corrected type. There are additional parameters that are specific to this type - the odom alphas (1 through 4). These parameters define how much noise is expected from the robot’s movements/motions as it navigates inside the map.

These parameters are set like in the table and figure to give the best stability.



```
<!-- Localization-->
<node pkg="amcl" type="amcl" name="amcl" output="screen">
  <remap from="scan" to="udacity_bot/laser/scan"/>
  <param name="odom_frame_id" value="odom"/>
  <param name="odom_model_type" value="diff-corrected"/>
  <param name="base_frame_id" value="robot_footprint"/>
  <param name="global_frame_id" value="map"/>
  <param name="transform_tolerance" value="3.0" />
  <param name="min_particles" value="40"/>
  <param name="max_particles" value="180"/>
  <param name="update_min_a" value="0.1"/>
  <param name="update_min_d" value="0.25"/>

  <param name="laser_max_beams" value="25"/>
  <param name="laser_z_rand" value="0.04"/>
  <param name="laser_z_hit" value="0.92"/>

  <param name="odom_model_type" value="diff-corrected"/>
  <param name="base_frame_id" value="robot_footprint"/>
  <param name="global_frame_id" value="map"/>
  <param name="odom_frame_id" value="odom"/>
  <param name="odom_alpha1" value="0.05"/>
  <param name="odom_alpha2" value="0.05"/>
  <param name="odom_alpha3" value="0.05"/>
  <param name="odom_alpha4" value="0.05"/>

</node>
```

Fig. 1. Parameters for AMCL

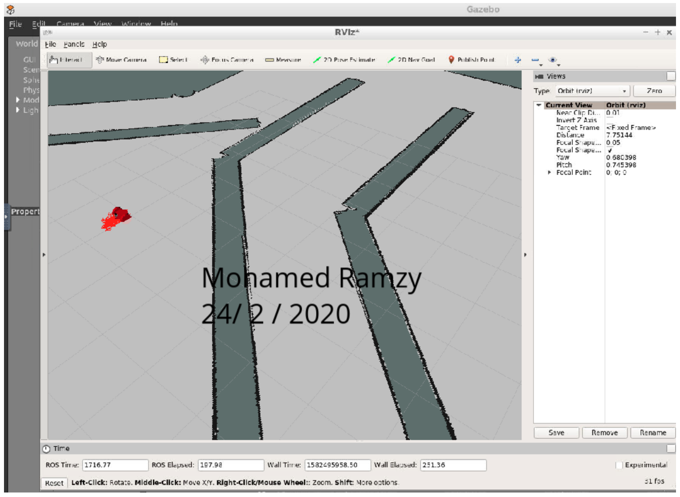


Fig. 2. My robot reaches his goal

4 Results

Here the results of both robots are shown, The udacity bot and ramzy bot -my created robot model- in Figure 2 and Figure 3.

5 Discussion

5.1 Topics

- Which robot performed better? The udacity robot performed better as it reached the goal faster
- Why it performed better? (opinion) The parameters of the AMCL fits it more than the other model as i tried to tweak them both but the first robot achieved better.
- How would you approach the ’Kidnapped Robot’ problem? We can solve it using MCL Algorithm by giving the robot the signal that its position has changed so he can account for it.
- What types of scenario could localization be performed? When we have a robot with uknown initial

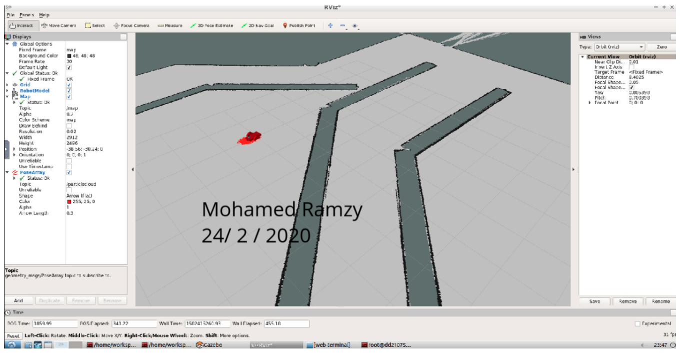


Fig. 3. Udacity bot reaches his goal

position and we want to track it, And when we have a robot and don't know the initial position, And when we have a robot and its position was changed suddenly.

- Where would you use MCL/AMCL in an industry domain? For applications that require localization such as self driving cars and shipping robots and cleaner robots

6 Conclusion / Future work

To summarize, The AMCL algorithm works really well in practice but there is a tradeoff between accuracy and time, As accurate results requires more computations as the input is for example high dimensional, So in the application that we use the algorithm it is needed to count for this.

6.1 Modifications for Improvement

Examples:

- Adding multiple cameras
- Adding new kinds of sensors such as sonar
- Using more computational power to achieve much better results using higher resolutions