Project Report – Where am I?

Abstract:

This project is about localization of two mobile robots, one is given and other one should modified version of first or different mobile robot. In this project, localization is achieved by using amcl package in navigation stack of ROS. The project also gives detailed explanation of how localization is achieved in both the robots.

Introduction:

The project is about achieving the localization of the mobile robot in the known environment. With a few changes in parameters of the amcl packages, how localization is achieved in different types of mobile robots is measured. The second robot used in this project was a differential robot which was taken from a source mentioned in the reference and was modified by adding camera sensor to the robot beside laser sensor. By doing this I could know how to decide the coordinates of the robot in the urdf file.

Background:

This project deals with the localization problem. In each map robot should be able to know its position and orientation of it. Self-driving cars application is more useful application of localization. The robot should be able to locate itself to not to get lost from the initial position.

Robot localization is considered as 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. Monte Carlo localization(MCL) helps us achieve this.

MCL is also known as particle filter localization. Given a map of the environment the algorithm of MCL estimates the position and orientation of robot with the help of help of sensors attached to the robot. Because the robot may not always behave in a perfectly predictable way, it generates many random guesses of where it is going to be next. These guesses are known as particles. Each particle contains a full description of a possible future state. When the robot observes the environment, it discards particles inconsistent with this observation, and generates more particles close to those that appear consistent. In the end, hopefully most particles converge to where the robot is.

Model_configuration:

I configured the parameters based on the errors for the udacity_bot. Like for example, if the error says add meter_scoring then I added meter scoring and when the error says frequency error, I added frequency values and so on. Some other parameters like velocity and acceleration and their values are obtained by many iterations.

I found the optimum particles be min-500, max-2000 and the transform tolerance value in amcl is also an important factor for the robot movement. My robot moved after adding that parameter.

Also, I observed the importance of yaw_tolerance and XY_tolerance values in move_base affect the robot movement.

My Robot Model,

I added all the parameters I used for my robot model and added other parameters like amcl recommended parameters and tuned them to almost perfection.

The Robot model I used consists of a base_frame, two wheels and two sensors. It has castors to support the weight of the chasses. The chassis is circular in weight

For Udacity_bot

AMCL Parameters

Parameter	Value	Reason for Selecting
Update_min_d	0.5	The value mainly influences great localization of particles. More than 0.5 is bad localization for this kind of robot. Less than this value is making the robot stuck and abandon the path
Update_min_a	0.2	This value mainly influences great localization of particles. More than 0.2 is bad localization and less than this value is making the robot stuck for this Robot
XY goal tolerance	1.0	I tried putting more than 1.0 and it gave me errors. But less than 1.0 also works fine.

Base_local_parametes

Max,min vel x and max_vel_theta min_in_place_vel_theta	0.5, 0.01, 1.5, 0.01	These are the values are abstained by continuous iteration.
Acc_lim_x, acc_lim_y, acc_lim_theta	0.5, 0.5, 1.5	These are also obtained by iteration, but any reasonable values would work.
Meter_scoring	true	Added to avoid a warning

Global_costmap_params:

Update frequency, publish frequency	1.0, 1.0	The frequency of global should be less than local as it is not necessary to update cost map every time globally
Static map	true	The map is not moving as the robot moves around
Rolling window	false	As static map = true
Width and height	10, 10	Randomly taken but should be more than local cost_map values
resolution	0.05	Most cases default

Local cost map params:

Update frequency, publish frequency	2.0, 1.0	The frequency of global should be less than local as it is not necessary to update cost map every time globally
Static map	false	The map moves as the robot moves around
Rolling window	true	As static map = false
Width and height	6, 6	Randomly taken but should be less than local cost_map values
resolution	0.05	Most cases default

Cost map common params:

Obstacle range, raytrace_range	2.5,3.0	The ranges from sensor observation. I just took default values.
Robot radius	0.5	Distance a circular robot should be clear of the obstacle.
Inflation radius	3.0	Default value

For my bot configuration:

The robot I modified is a differential drive Robot with two actuators and two sensors. I have integrated the robot with the map and modified some parameters in amcl to get robot moving in the map and reach the goal point. The sensors used in this robot are 1. Camera and Hokuyo laser. As it is a differential drive robot, the wheels can move in separate velocity. As per my opinion the number of particles do not influence localization part of the robot. The most important params I found important are update_min_d and update_min_a in my opinion

AMCL parameters:

Parameter	Value	Reason for Selecting
Update_min_d	0.25	The value mainly influences great localization of particles. More than 0.25 is bad localization for this kind of robot. Less than this value is making the robot stuck and abandon the path
Update_min_a	0.2	This value mainly influences great localization of particles. More than 0.2 is bad localization and less than this value is making the robot stuck for this Robot
XY, yaw goal tolerance	0.3, 0.15	More or less than these values influenced robots end position at goal.

base_local_planner params:

Max,min vel x and max_vel_theta min_in_place_vel_theta	0.3, 0.1, 1.0, 0.6	These are the values are obtained by continuous iteration.
Acc_lim_x, acc_lim_theta	0.5, 0.5, 1.0	These are also obtained by iteration, but any reasonable values would work.
Meter_scoring	true	Added to avoid a warning

Global Costmap params:

Update frequency, publish frequency	1.0, 0.5	The frequencies of global should be less than local as it is not necessary to update cost map every time globally
Static map	true	The map is not moving as the robot moves around
Rolling window	false	As static map = true
Width and height	10, 10	Randomly taken but should be more than local cost_map values
resolution	0.05	Most cases default

Local costmap params:

Update frequency, publish frequency	5.0, 2.0	The frequency of global should be less than local as it is not necessary to update cost map every time globally
Static map	false	The map is not moving as the robot moves around
Rolling window	true	As static map = false
Width and height	10, 10	Randomly taken but should be more than local cost_map values
resolution	0.05	Most cases default

Cost map common params:

Obstacle range, raytrace_range	2.5,3.0	The ranges from sensor observation. I just took default values.
Robot radius	0.4509	Distance a circular robot should be clear of the obstacle.
Inflation radius	0.5	Default value

Results:

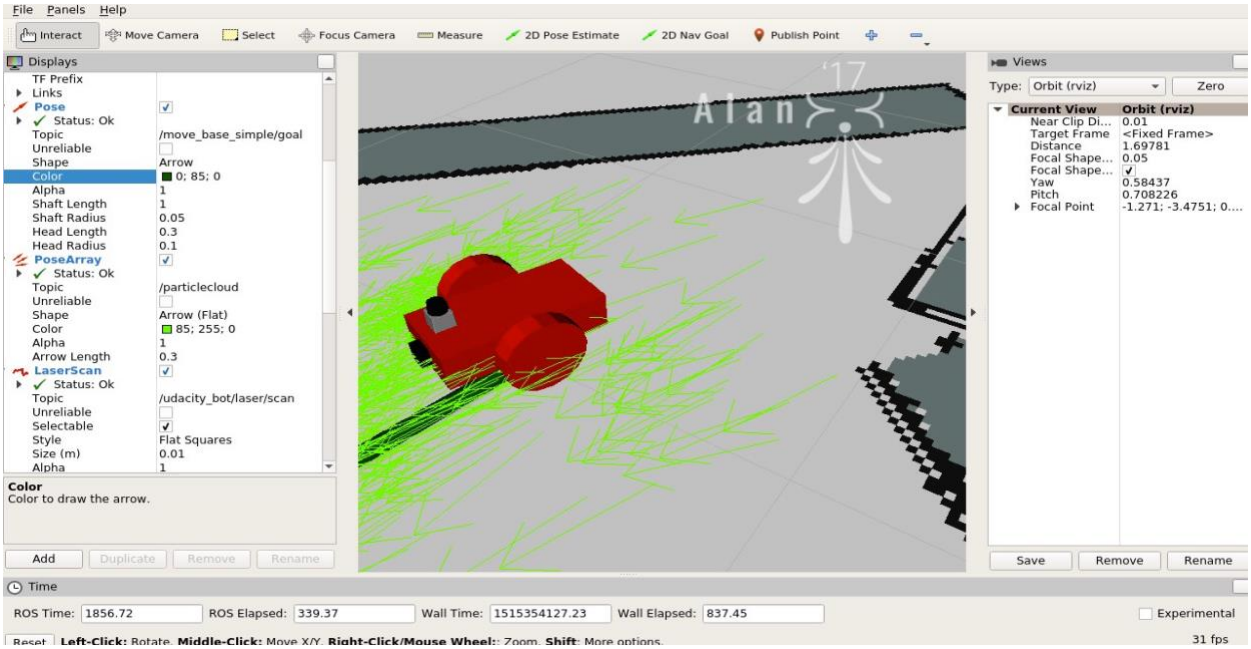

Figure 1: Showing the udacity_bot reaching goal position

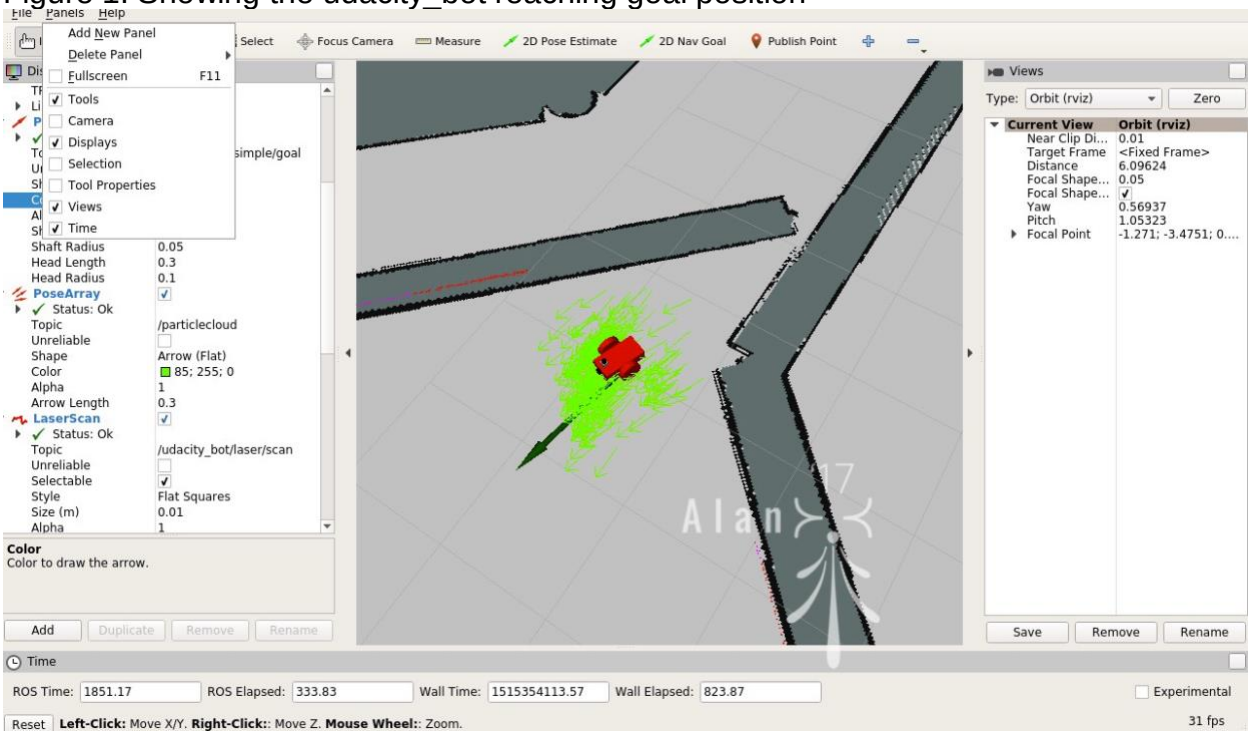

Figure 2: Udacity_bot reaching the goal position

Figure 3: figure showing my_bot reaching goal position

From the above figure the robot did not reach its goal position and orientation but is very close to desired poition.

Discussion:

Everything started going wrong when I first started the project. I was very confused at first and started adding all the parameters I found randomly. Also, I did not understand how to calculate the goal position. I did not find the arrow mark in my rviz as I thought the robot must come to the position of the arrow mark that is given in the problem and I saw in the instructions or in the slack to first make small goal positions. So, I used to draw the 2D nav gal arrow very near to the robot position and I was thinking robot will come to the head of the arrow. As starting point, the tail of the arrow is very near to the robot, the robot just moves little bit and stops. I did not realize my perception was wrong and was thinking my values were wrong. But after numerous trails (meanwhile I might have lost the correct parameters), I came to know about it and finally drawn the arrow near the goal position.

I thought the second part would be easy but I felt the difficulty when I started working on it. As a first trail, I thought of adapting and building a robot from already existing robots like husky, jackal etc, but after numerous trails that did not work out. Eventually I started working on differential drive robot.

The results are good for Udacity bot after tuning some parameters. But can be improved more like the localization of particles after reaching the goal position.

For My bot, the localization part is good but the only problem is, it is taking too long to reach the target and moving through the walls of the map without taking turn at the map to reach the goal. I thought inflation radius would influence its behavior but that's not the problem. I am in the process of figuring this out.

Future Work:

Kidnapped robot problem can be addressed using amcl. For the kidnapped robot problem, the robot should be able to detect its own position and it can be achieved using particle filter techniques in amcl. The robot should be able to detect obstacles and if stuck it should be able to retrieve itself from the obstacle and move on to the goal position which in this case my robot can do. So amcl can address kidnaped robot problem. Also, the robot Here we are not considering the initial position or subscribing to the topic of the initial pose. The initial position of the robot by default is taken as (0,0,0, theta) and the final goal position is calculated automatically and robot can move to the goal position by localizing itself.

For further improvement of mobile robot, a robot with four wheel and a carrying box and an arm on it like mars rover which collects objects would be a great application. I thought of working on it but now just started with small robot. In my opinion the two parameters in amcl update_min_a and update_min_d are important factors in increasing the accuracy of the robot localization and descreasing the processing time. So, for the differential drive robot, adjusting those parameters would help us in achieving that.

References:
https://books.google.com/books/about/Mastering_ROS_for_Robotics_Programming.html?id=2jjlCwAAQBAJ