Project: Where Am I?

Abstract

Localization is the task of determining the position of a robot in a world. Path planning is the task of determining movements of the robot that will get it to a desired location in the map. This experiment aims at achieving these two tasks in a pre-provided map of an environment. The first part involves building a model of a robot. In this project, a four wheeled robot with a skid steer drive has been built. Adaptive Monte Carlo Localization is used for localizing the robot in the map, and then the move base package is used to plan the path for the robot navigation using the base local planner.

Introduction

The first aim of the project is to create a model of a robot, and to localize the robot in a known map of the environment. Subsequently, the task is to make sure the robot is able to reach any assigned goal location and orientation, while maintaining an accurate localization throughout.

Background

Localization is the process by which the robot can figure out where it is in the current map, that is what is the position and the orientation of the robot with respect to it's environment. It plays a important role in helping the robot decide where and how to move next.

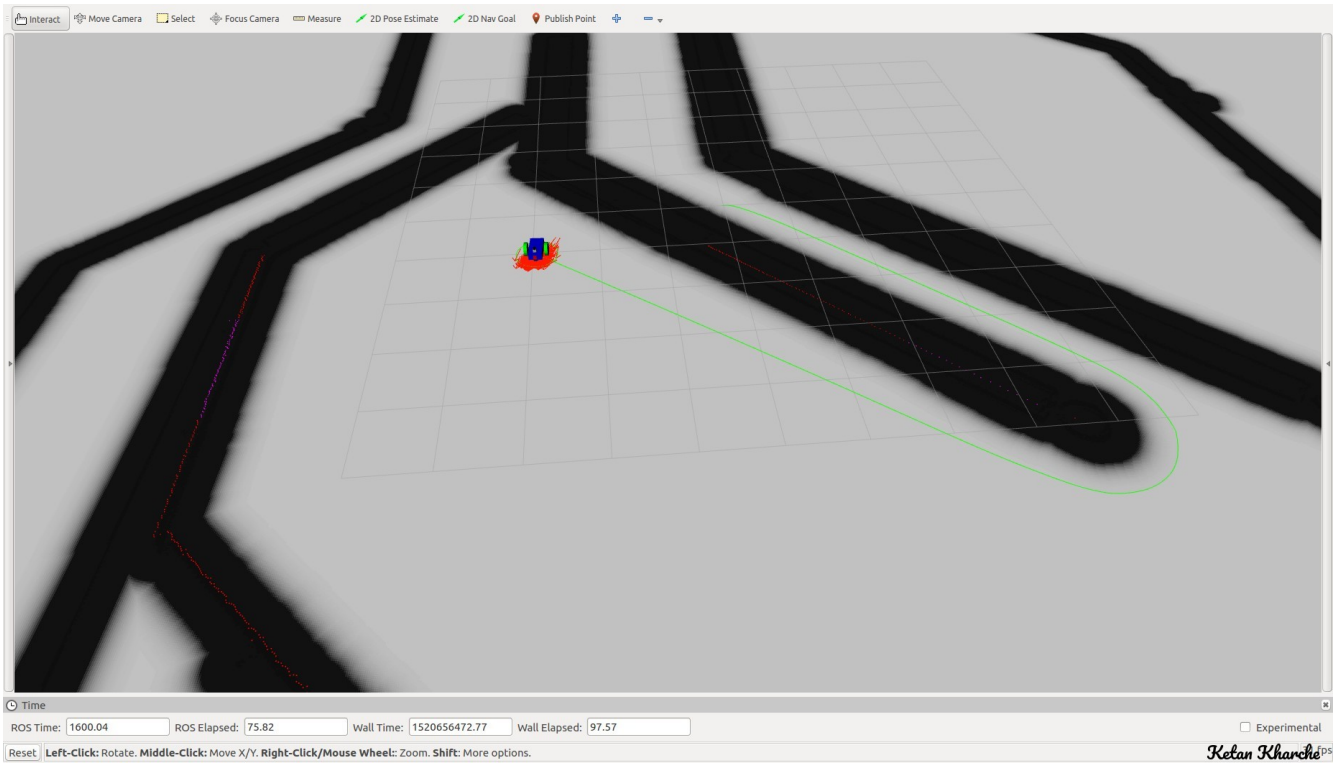
The challenges in robot localization depend on the scale of the localization. For position tracking when the intial pose is known, the main challenge is dealing with the uncertainty in both the motion of the robot and the measurements of the sensors. For global localization, the initial pose is unknown, which increases the uncertainty. Finally, the robot might miscalculate where it is in certain cases, and then it needs to have the ability to relocalize itself to the right pose, which is a difficult problem to solve.

The possible approaches for localization include a Kalman filter based and a particle filter based approach. The main advantage of the particle filter approach is that it is more robust and less retricttive than the Kalman filters. Here is a comparison table for both:

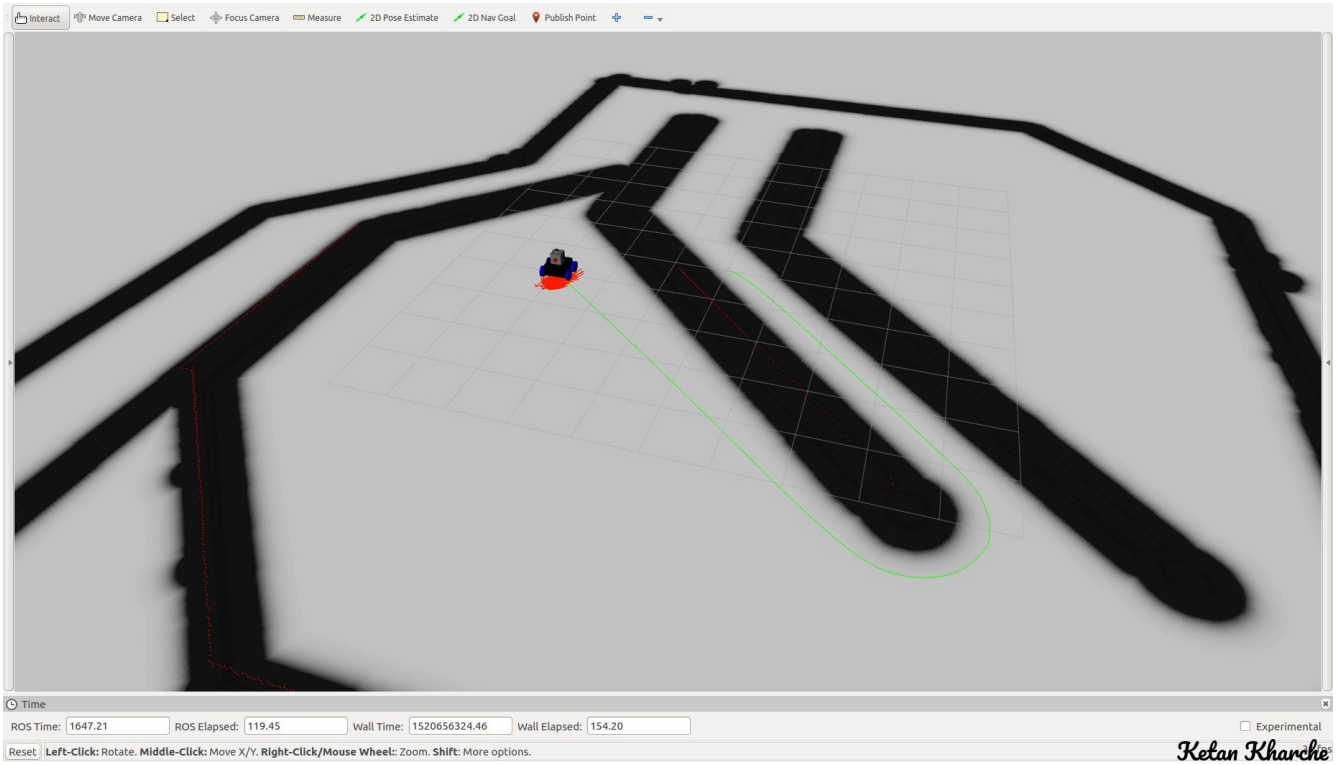
Kalman Filter	Particle Filter
Assumes Gaussian measurement noise	Can be any measurement noise
The state space is unimodal and continuous	The state space is multimodal and disccrete
Faster and uses less memory	Slower and uses more memory
Can be used only for local localization or positional tracking	Can be used for both global and local localization
Posterior is a Gaussian	Posterior is represented by particles
Fixed memory and details in information for all regions	Can adaptively change memory and amount of information
Higher resolution	Lower resolution

Results

Classroom robot



Project robot



Model Configuration

The project robot is a four wheel robot with a skid steer drive. It is comparatively larger in size than the classroom robot, with a dimension of 0.4 (length) by 0.6 (breadth) by 0.45 (height). The camera is located at around 0.2 m above the ground and it is slightly set in compared to the outermost point of the robot. The lidar is located on the top of the robot, to ensure there's no obstruction from the robot body to it's field of view. The skid steer drive is implemented in the ground_robot.gazebo file.

For the base local planner, I am using an xy goal tolerance of 0.1 and yaw goal tolerancce of 0.01 to ensure the robot gets close to the goal without being too restrictive about the exact pose. I am using a sim time of 1.5, so that the robot knows where it needs to head in the near future. I observed that having a low sim time, the robot sometimes got stuck on long turns such as a complete U turn. Further, I am using a pdist_scale of 1.2 and gdist_scale of 1.0, to ensure that the robot stays close to the path with some leeway about straying a little from the path and the local goal.

In costmap common params, I use an obstacle range of 9.0 and raytrace range of 10.0. I also have a transform tolerance of 1.0, to allow some latency in the transforms, and a robot radius of 0.35 to include the whole robot. The inflation radius was set to 0.7 so that the global plan gets planned at a safe distance from the obstacles.

The global costmap params have the same transform tolerance of 1.0. The update and publish frequency are set to 10.0 Hz for my laptop's hardware constraints. The above properties were kept the same for the local costmap. However, I had to decrease the width and height for the local costmap to ensure the robot did not keep rotating in the same place continuously.

In the amcl launch, I set the min number of particles as 200 and the max number of particles as 1000. For the given map, 1000 seems to be a high enough number of particles to avoid the situation that we end up with no particles left in the actual robot position. If the particles are too less, we can end up with no particles near the robot position, and in that case amcl can localize the robot somewhere else. Further, I have turned off transform broadcast from amcl, to avoid a conflict with the transform published by gazebo and the static transform publisher. I am using a laser min range of 0.1 m and a laser max range of 30 m (which is the max range of Hokuyo 30lx 2D Lidar).

The move base controller frequency is set to 10 Hz to meet the laptop hardware constraints.

Discussion

The robot performs well in the given map. Initially, for the new robot, getting the skid steer working required some work. Further, initially I had some trouble with the navigation to goal. The robot would start rotating in place if the goal was set far away, typically beyond an obstacle. After trying out multiple iterations with varying parameters, the fix for the problem was decreasing the width and height of the local costmap. After fixing that, the robot is able to navigate it's way to the goal location consistently, and pretty quickly (close to 1 minute).

Adaptive Monte Carlo Localization(AMCL) would not work too well fro the kidnapped robot problem, because once the robot is localized, almost all the particles will be collected around the robot. So, the algorithm would not be able to account for an unexpected change in the position of the robot, and there would be no particles present in the new changed location of the robot. One solution for this could be

adding new particles to the map at random locations, either at some regular intervals, or when the confidence of the localization decreases. This might lead to some particles being present close to the new robot location, and it might be able to relocalize.

The MCL algorithm can be used whenever we need to localize a robot in a an environment with a pre-existing map. In an industrial environment, MCL can be used to localize robots operating in a factory setting. For example, these can be cleaning robots, or inventory robots, or delivery robots. The adaptive MCL further provides a good option for localization in computationally constrained systems, for similar applications.

Future Work

The future work involves improving the ability of the robot to stick closer to the path, so that it can maneuver more complex paths and tighter spaces properly. This will however increase the time needed by the robot to get to the goal point, since it constrains the orientation and motion of the robot further. Further, increasing the robot size would also affect the robot performance similarly and will require dealing with the same constraints. Thus, we will get higher accuracy but it would also require more time. Another target would be making the robot move in an environment with more complex obstacles, i.e. maybe obstacles with overhangs or extensions. This would likely require a different sensor that can build a 3D map, such as a depth camera or a 3D Lidar, so that apoint cloud of the environment can be built.