report.md

where am I

Abstract

In this project ROS packages AMCL and the Navigation Stack will be utilized to accurately localize a mobile robot inside a provided map in the Gazebo and RViz simulation environments. Two mobile robot equipped with laser ranger and odometry wheel is designed for navigation ,while specific parameters tunned for the best possible localization results.

Introduction

Knowing your location, and being able to navigate to other locations, is extremely important for autonomous robots with an internal map of its environment.

Using a Bayes filtering approach, roboticists can estimate the state of a dynamical system from sensor measurements.

- State: The robot's pose, including its position and orientation.
- Measurements: Perception data(e.g. laser scanners) and odometry data(e.g. rotary encoders)

This project is to accurately localize a mobile robot with a provided map in the Gazebo and RViz simulation environments, using ROS navigation stack,robot will at last navigate to specified goal position. Two different size bots are designed for this task

Background

Localization involves one question of Navigation: Where is the robot now relative to Map or some landmark which provide input to path planning--another question of navigation Combined with noisy sensor readings and some form of closed-loop motion feedback, the localization algorithm need to filter out the uncertainty, which estimate the robot state/pose (simply the robot's X and Y coordinates, and the heading/orientation)

Compare two Localization method as following , Kalman and Particle filters :

	MCL	EKF
Measurements	Raw Measurements	Landmarks
Measurement Noise	Any	Gaussian
Posterior	Particles	Gaussian
Efficiency(memory)	✓	✓✓
Efficiency(time)	✓	✓✓
Ease of Implementation	✓✓	✓
Resolution	✓	✓✓
Robustness	✓✓	x
Memory & Resolution Control	Yes	No
Global Localization	Yes	No
State Space	Multimodel Discrete	Unimodal Continuous

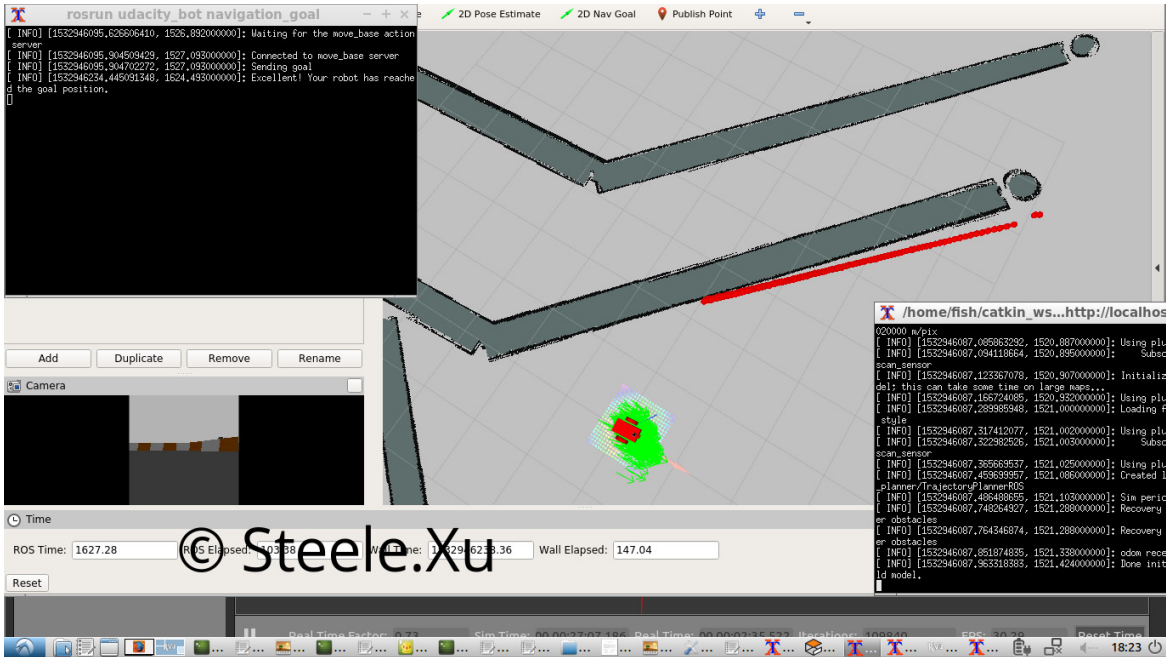
from the table,we know MCL is simple , robust and flexible, but need more computation cost and low resolution than KF; Kalman filter is accurate and efficient with limited situation,though EKF can deal with non-linear problem

MCL

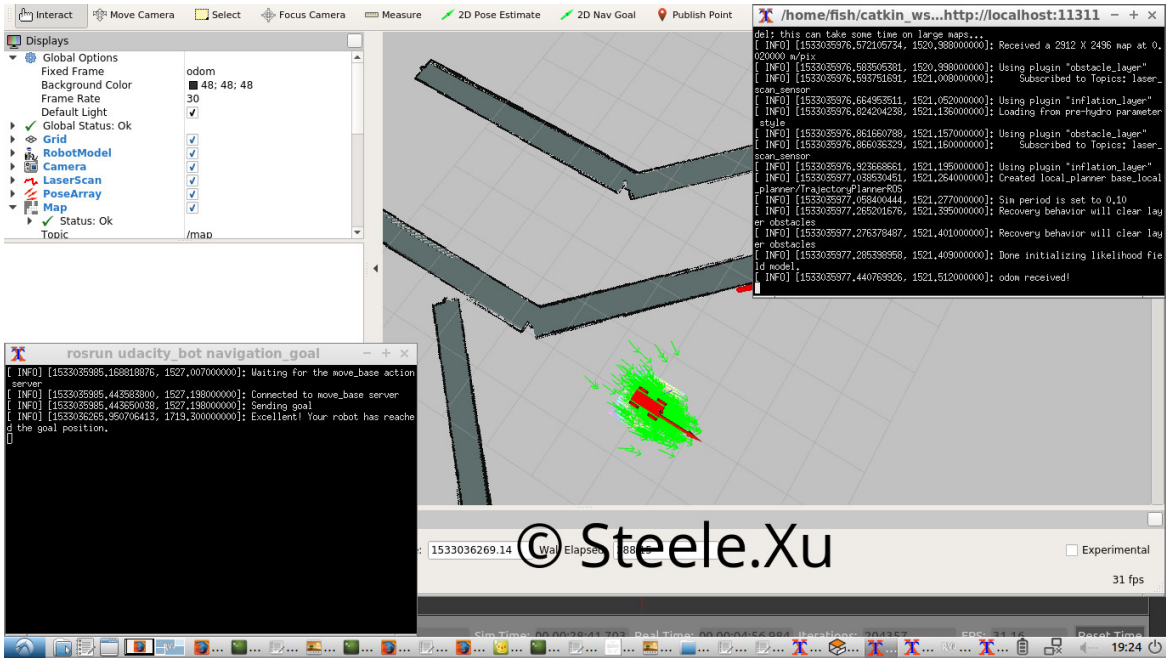
In this project we use MCL ,in which Each "particle" is a guess of the robot's pose, Initialy, particles are uniformly distributed on the map. when the bot moves, the particles with the lowest probability of correctness are replaced by new guesses, eventualy converge very closely to our real position over many iterations. MCL run under bayes filter framework, involving steps: motion update,sensor udate and resampling

Results

- classroom robot at the goal, robot_radius:0.2 inflation_radius: 0.2, chasis box size=.4 .2 .1



- my robot at the goal, robot_radius:0.3 inflation_radius: 0.4, chasis box size=.6 .2 .1



Model Configuration

robot creation, environment setup

- prepare world launch file(jackal_race)

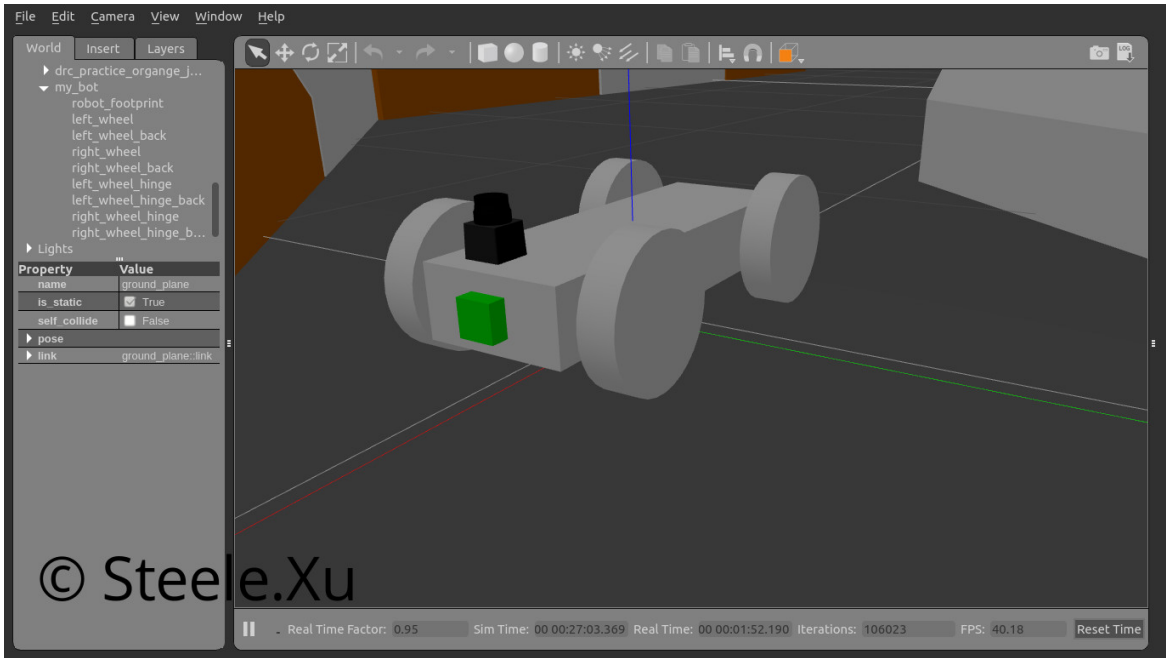
- bot with chasis ,wheel ,sensor, caster sphere radius decrease 0.499 to move smoothly
  - using differential_drive_controller with setting: commandTopic:cmd_vel, odometryTopic:odom, odometryFrame:odom
  - increase the minimum of hokuyo or wheel treat as obstacle
- test bot with : rostopic pub /cmd_vel geometry_msgs/Twist

localization setup

- Modify the appropriate launch files to include the provided map, integrate the AMCL and Navigation ROS/ move_base packages.
- rviz config file, add poseArray, costmap, path, etc
- navigate it to the goal position by provided cpp file

my_bot setup

- enlarge its size, with a longer chasis ,length from 0.4 to 0.6, making space for additional wheel, using skid_steer_drive_controller in my_bot.gazebo; accordingly move the sensor to the head of bot
- test navigation_goal on my_bot



parameters from config files for the amcl and move_base nodes.

- Ensured that the robot and the map are responsive by tuning the transform tolerances 8 , and the map update and publishing frequencies. 10hz
- a little bigger obstacle_range raytrace_range will better
- to lowering the computational load, max_particles = 2000; at the same time insrease the frequent of filter updates by half default update_min_d and update_min_a, then I got a better tight poseArray
- set inflation_radius=0.3; for different size of bot, robot_radius should be different
- set size of local_costmap to 1 by 1
- set publish_cost_grid_pc: true , so I can add pointcloud2 of local planner in rviz
- 4-wheel bot need latch_xy_goal_tolerance be true,do rotation when reach goal ;
- when set heading_scoring: true, the heading of car will oscillation, so close this switch

Discussion

In my_bot setup, the 4-wheel seems slower and not flexible as udacity_bot, so set the latch_xy_goal_tolerance for a rotation when reach the goal; another problem is my_bot often stuck in the turning point, increase inflation_radius may mitigate this.

The poseArray degrade when bot turn around or rotate, need further investigate .

Future Work

To make bot run more smoothly, need to fine tuning more :

- find more powerful computer to do experiment, so more particles update gain more accuracy
- laser related parameters not tunned in this sessiom ,e.g. increase laser_max_beams.
- Trajectory Scoring including pdist/gdist/occdist