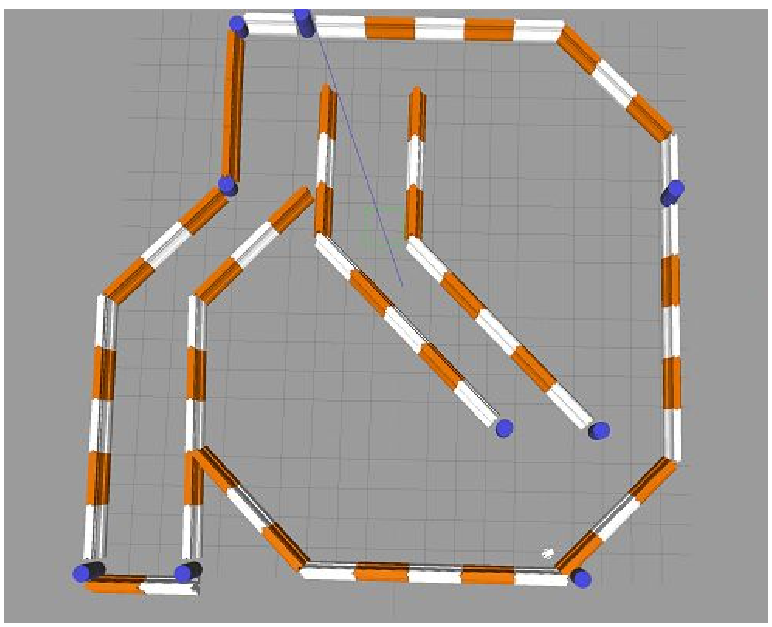
Udacity RoboND Project Report: Where am I ?

Done by: Tan Kai Xiong



Abstract

The goal of the project is to make use of ROS (Robotics Operating System) packages to accurately localize a mobile robot inside a provide map in the Gazebo and RViz simulation environments. A custom differential wheel mobile robot model will be created in a Gazebo world. ROS navigation stack is applied in this project, packages such as AMCL (Adaptive Monte Carlos Localization), Map_Server and Move_Base. Depend on compute platform and sensor accuracy, performance tuning is required to achieve the best localization results.

Introduction

The objective of this project is to use ROS packages to localize a mobile robot inside a provided map in the Gazebo and RViz simulation environments. This article will present an overview of a navigation stack in ROS.

- Create a custom ROS package
- Write a custom robot model in a Gazebo world
- Write launch script to bring up the custom robot in the Gazebo world
- Write launch script to launch AMCL and MOVE_BASE packages
- Tune the AMCL parameter to achieve the best possible localization results

Robot Model

A custom differential wheel mobile robot is built in URDF file. For this type of wheel drive, it requires two independent motor driven wheels to enable this type of driver to work. It can have either one or two caster wheels to support the robot. It consists of a hokuyo laser scanner, camera and differential wheel sensor.

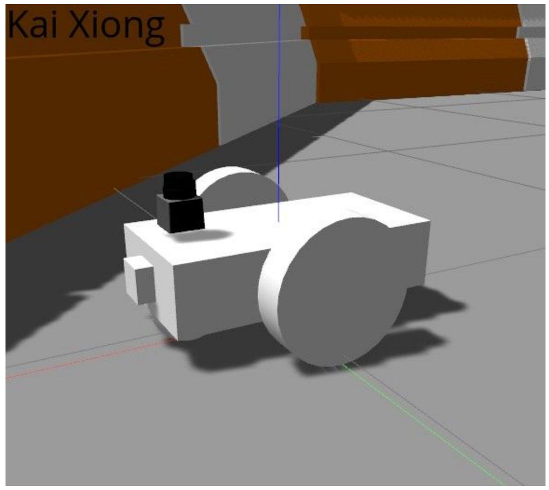


Figure 1. Robot Model

Localization

Localization is the basic function of a mobile robot. Before the robot can execute any movement, it need to determine its own position with respect to the map. With the ability of localization, the robot can navigate from point A to B. Mobile robot also must be able to avoid dangerous situations such as collisions and unsafe environment.

EKF vs MCL

The main two types of localization methods are “Extended Kalman Filter” and “Monte Carlo Localization”. MCL is a type of particle filter algorithm. Compare EFK with MCL, MCL is more robust and it capable of global localization.

	EKF	**MCL**
Observation	Landmarks	Raw measurement
Observation noise	Gaussian	Any
Posterior	Gaussian	Particles
Efficiency (Memory)	☆☆	☆
Efficiency (time)	☆☆	☆
Ease of Implementation	☆	☆☆

Resolution	☆	☆☆
Robustness	╳	☆☆
Memory & Resolution Control	No	Yes
Global Localization	No	Yes
State Space	Unimodel Continuous	Multimodel Discrete

Table 1. EKF vs MCL comparison table

AMCL

AMCL (Adaptive Monte Carlo Localization) is a probabilistic localization package for 2D navigation. It implements the adaptive (or KLD-Sampling) Monte Carlo Localization approach which use a particle filter to track the pose of a robot with a known map.

The AMCL pose output frequency is depended on the frequency rate of the laser scans. If the laser scan frequency is 10 Hz, the maximum output of the AMCL_pose will be 10 Hz. In order to make the particle converge quickier, by reducing the “update_min_d” and “update_min_a”.

Parameters	**Value**
odom_frame_id	odom
odom_model_type	diff-corrected
base_frame_id	robot_footprint
global_frame_id	map
min_particles	10
max_particles	500
transform_tolerance	0.1
resample_interval	1
odom_alpha1	0.05
odom_alpha2	0.05
odom_alpha3	0.05

odom_alpha4	0.05
update_min_a	0.01
update_min_d	0.01

Table 2. AMCL parameters

Move_base

The move_base package provides an implementation of an action (actionlib package) that, given a goal in the world, it will attempt to reach it with a mobile base. The move_base node links together a global and local planner to accomplish its global navigation task. It supports any global planner adhering to the nav_core::BaseGlobalPlanner interface specified in the nav_core package and any local planner adhering to the nav_core::BaseLocalPlanner interface specified in the nav_core package.

Parameters	Value
base_global_planner	navfnNavfnROS
base_local_planner	TrajectoryPlannerROS
max_vel_x	0.5
min_vel_x	0.1
xy_goal_tolerance	0.05
yaw_goal_tolerance	0.05

Table 3. Move_base parameters

Map_server

Map_server provide the map_server ROS Node, which offers map data as a ROS Service. It also provides the map_saver command-line utility, which allows dynamically generated maps to be saved to file. It is used for loading static map.

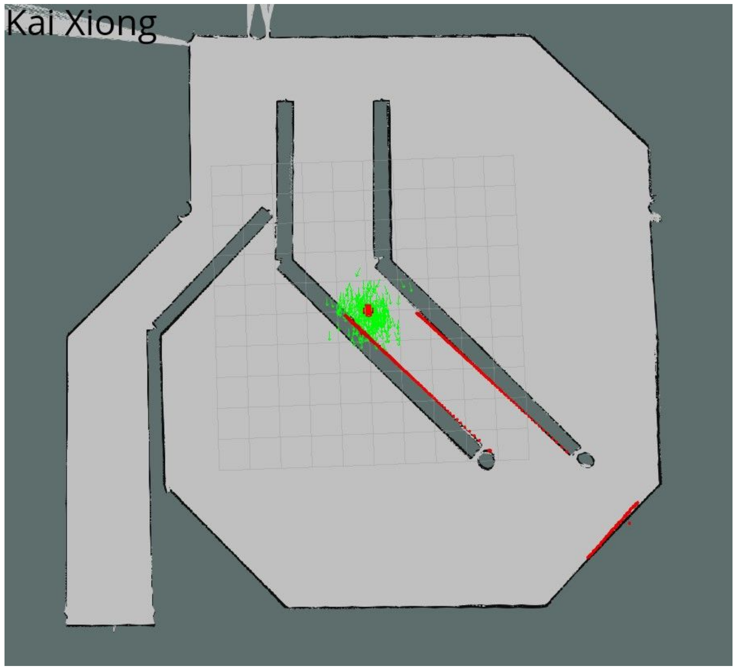


Figure 2. Jackal race Grid Map

Localization Accuracy

The AMCL output AMCL_pose message. The message type is geometry_msgs/PoseWithCovarianceStamped. The localization accuracy is represented in the pose covariance. The 6x6 covariance matrix represent in x, y, z rotation about X axis, rotation about Z axis. The higher the value, it represent higher uncertainty.

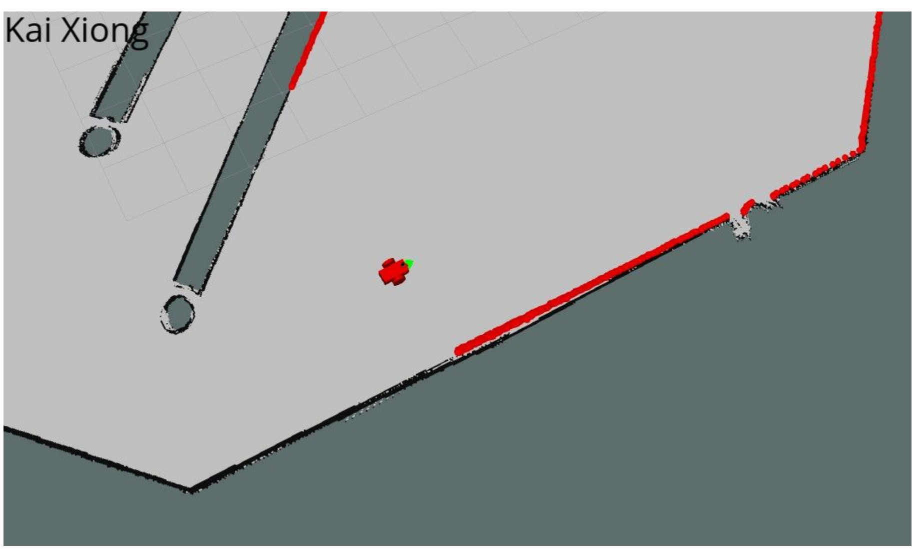


Figure 3. Localized map

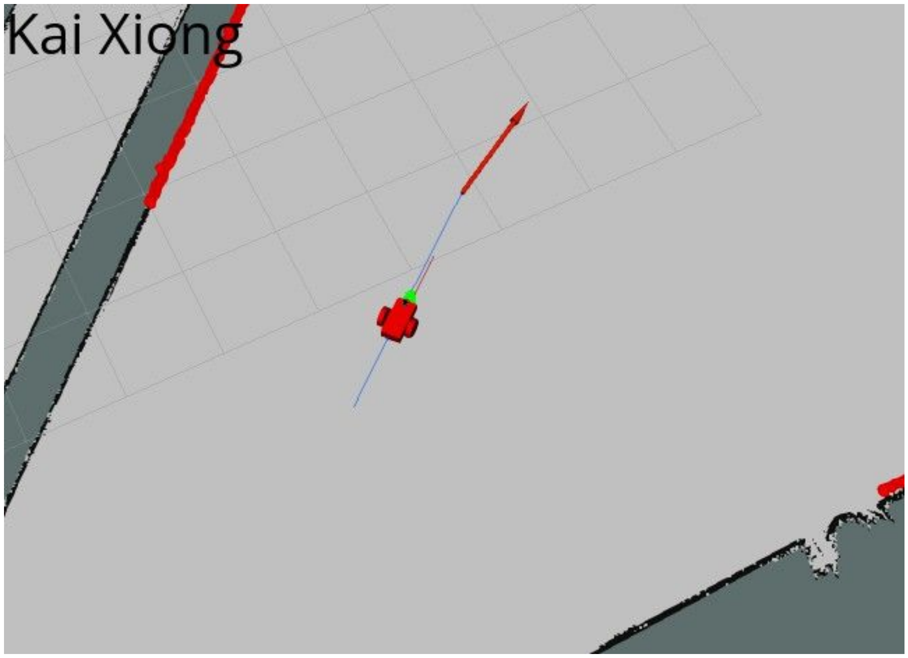


Figure 4. Robot move to goal

Discussion

The AMCL performance depended on the CPU performance, laser scan accuracy and AMCL parameters. AMCL may perform badly in location such as long corridor, where there is less distinct feature in the map.

Conclusion / Future Work

In conclusion, the custom mobile robot capable of accurately localized in a known map. Sensor fusion can help to improve the accuracy. Sensors such as IMU, RGBD camera and GPS. Robot_pose_ekf packages can perform sensor fusion by taking in sensors output. It can help to output more accurate position, even if one of the sensors is failed.

Extra Questions

Results for classroom robot and the custom robot

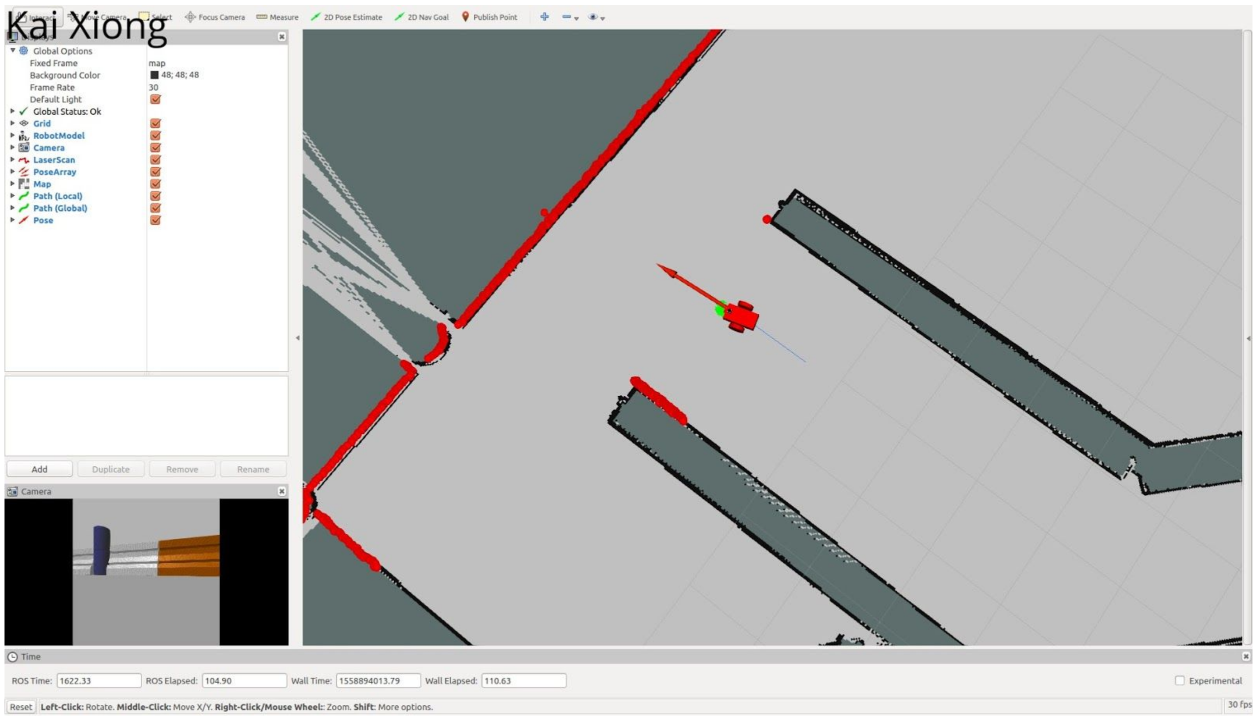


Figure 4. Robot at goal position for classroom robot and custom robot

Discuss about whether AMCL would work well for the kidnapped robot problem and what kind of scenarios would need to be accounted for it.

- AMCL would work well for the kidnapped robot problem as it can perform global localization. It would require a known map. Global localization is being used if the robot do not know its initial position. There is a ROS service in AMCL which can trigger the global localization. The particles are dispersed randomly through the free space in the map. When the robot move, the particles are being updated and resampled. After a few iterations, the particle will converge.

Also provide examples with very brief discussions on where they would use MCL/AMCL in an industry domain

- MCL/AMCL can be used in application such as cleaning robot, service robot, transportation robot and AGV (Autonomous Guided Vehicle).

Explain the trade-offs in accuracy and processing time.

- The number of particles being used in AMCL
- Update distance and angular rate

Number of particles used in AMCL is depended on the map area and whether global localization is needed. Increase the number of particles will affect the processing time. If there is too few particle, the accuracy of the AMCL pose will decrease.

Reduce the update distance and angular will increase the update frequency, thus increase the processing time. It allow the particle to converge faster and the accuracy of the AMCL pose will increase.

The performance of AMCL also depend on the compute platform. If the compute platform CPU performance is slow, the user need to reduce the number of particle or increase the update distance and angular rate.

Reference

- http://wiki.ros.org/map_server
- http://wiki.ros.org/amcl
- http://wiki.ros.org/move_base
- http://wiki.ros.org/costmap_2dhttp://wiki.ros.org/costmap_2d
- http://wiki.ros.org/navigation