Localization Project - Where Am I

Hanwei Zhang(WSYiDao)

Abstract—Project required develop a mobile robot model for Gazebo, and integrate the AMCL(Adaptive Monte Carlo Localization) and Navigation ROS packages for localizing the robot in the provided map. In order to reach the goal successful, it need to add and tune parameters for the running ROS packages to improve the test localization results.Two different robot models are considered for performance evaluation. After achieving the desired results for the robot model introduced in the lesson, a new robot model was made with significant changes , by tuning parameters it reach the goal successfully.

Index Terms—Robotics, Gazebo, AMCL, Udacity, Localization.

1 Introduction

Localization is the challenge of determining the robot's position in a mapped environment, by implementing a probabilistic algorithm to filter noisy sensor measurements and track the robot's position and orientation. In the project, we simulate a robot in real world with a known map to approximate the current position and move to the goal position, by using the localization algorithm and the ROS packages.

2 Background

Extended Kalman Fitel(EKF) and Monte Carlo Localization(MCL) are two popular localization algorithms. EKF is the most common Gaussian filter that helps in estimating the state of non-linear models, and MCL is a particle filter because is estimates the robot's pose using particles.In the real world, the environment which is dynamic where objects may shift over time, and static environments where it always matches the ground truth map. They all faced noisy sensor measurements and it's important to estimate the current pose of the robot with a acceptable computational complexity.

2.1 Kalman Filters

Kalman Filters is a very robust algorithm for filtering noisy sensor data. Unlike other algorithms that require a lot of data to make an estimate, the Kalman Filters is able to do so after juct a few sensor maesurements. It does so by using an initial guess and by taking into account the expected uncertainty of a sensor or movement. Sensor fusion is also a good way to localization by using the Kalman filter to calculate a more accurate estimate location using data from multiple sensors.

2.2 Particle Filters

Each particles represents the hypothesis of there the robot might be. They are each assigned a weight, the weight of the particles is the difference between the actual pose and the predicted pose. The importance and accurate of a particle depends on its weight, and the large weights particles are more likely to survive after a resampling process. Particle Filters will converge and estimate the robot's pose after several iterations and different stages of resampling.

2.3 Comparison / Contrast

Compare to the MCL, the Kalman Filters is often used to a Linear Gaussian state space assumption that can only solve the position tracking problem. Kalman Filter and EKF is a good algorithm in a sensor fusion system by estimate different maturement togather and in a fast motion and estimation system with small cost of compute resources. MCL can be applied to more systems with nonlinear and global lacalization, and by tuning parameters it can control the computation memory and resolution. In the model state space, MCl use muti-model discrete and EKF is unimodal Continuous.

3 Simulations

In the robot simulation, robot model designed by using Universal Robotic Description Format(URDF), include two wheels on the left and right side of the chassis, two wheels front and back below the chassis, cameras in front of the box and a Hokuyo laser on the front top of the box. In the project, I use ROS AMCL package, Move Base package and modified ROS packages udacity_bot for first robot test. The second develop robot in files develop_config and urdf_develop ,if substitute files config and urdf could run the develop robot test.

3.1 Achievements

The benchmark model and the own developed model both reach the goal state and passed the navigation_goal.cpp test.

3.2 Benchmark Model

see Fig. 1

3.2.1 Model design

In the benchmark model, chassis with the size of (.4 .2 .1), two cylinder wheels on the left and right side of the chassis with the size of radius "0.1" and length '0.05', two sphere wheel on the front and back size below the chassis.

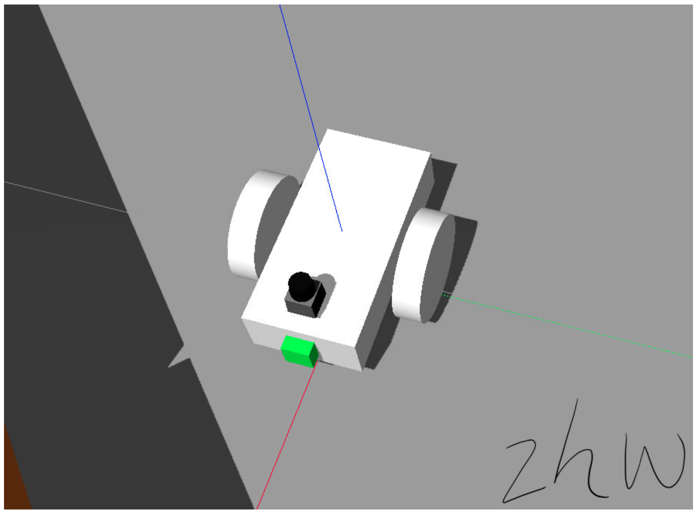


Fig. 1. Result 2:Develop robot has reached the goal position and pass the test.

3.2.2 *Packages Used*

In the project ,ROS AMCL package and Move Base package are used to localize the position and navigate the robot to the goal position.

3.2.3 *Parameters*

```
”base_local_planner_params.yaml”
TrajectoryPlannerROS:
  holonomic_robot: false
  yaw_goal_tolerance: 0.1
  xy_goal_tolerance: 0.1
  sim_time: 1.0
  meter_scoring: true
  controller_frequency: 15.0
```

```
”costmap_common_params.yaml”
  map_type: costmap
  obstacle_range: 3.0
  raytrace_range: 5.0
  transform_tolerance: 0.3
  inflation_radius: 0.3
  observation_sources: laser_scan_sensor
  laser_scan_sensor: {sensor_frame: hokuyo,
  data_type: LaserScan,
  topic: /udacity_bot/laser/scan,
  marking: true, clearing: true}
```

3.3 Personal Model

3.3.1 *Model design*

In the develop model, I try three kinds of model and find a best one (Personal model 3). The best personal model Robot included: chassis with the cylinder size of length ”0.1” radius”0.15”, four sphere wheel below the chassis size of radius”0.0299”, two cylinder wheels on the left and right side of the chassis with the size of radius ”0.1” and length ’0.05’,

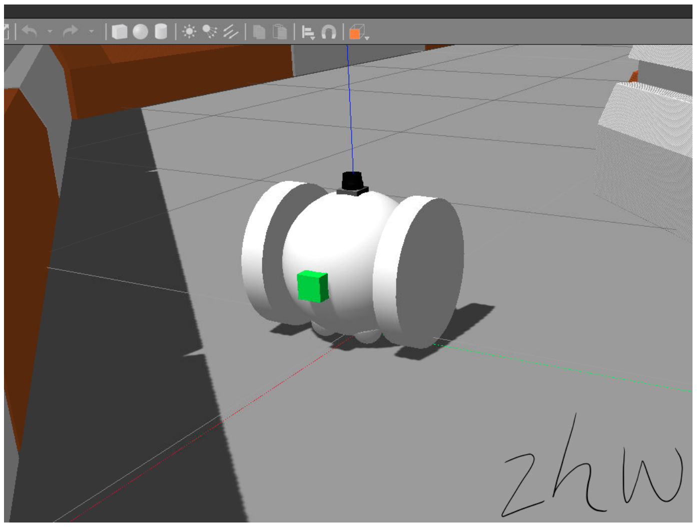


Fig. 2. Personal model 1.

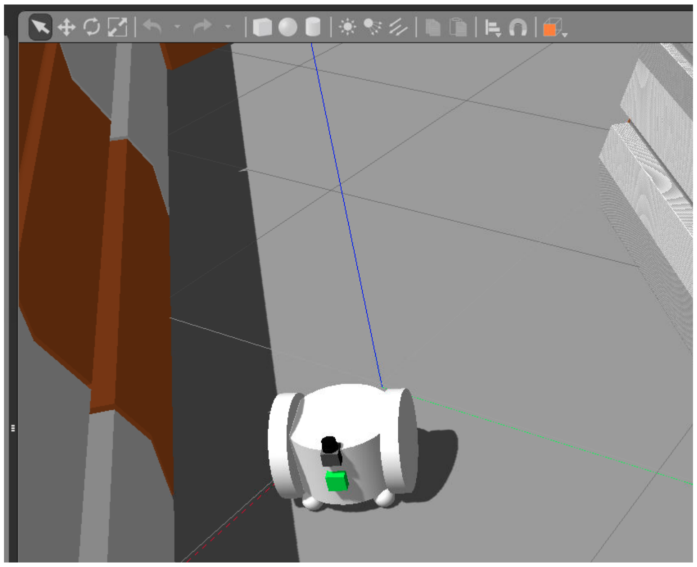


Fig. 3. Personal model 2.

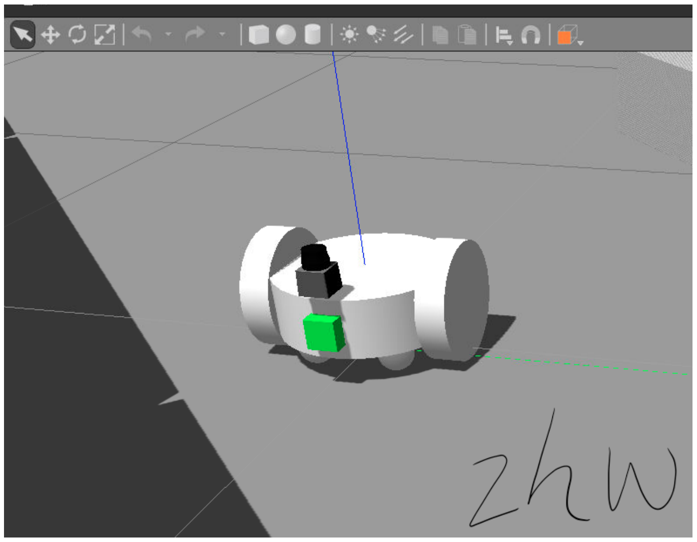
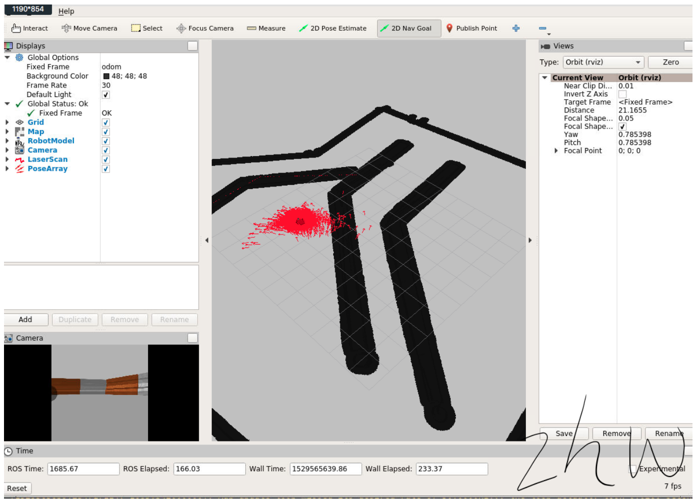
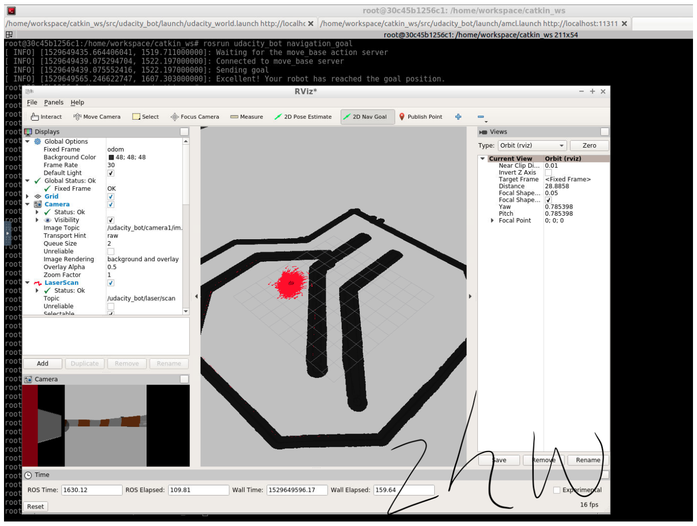


Fig. 4. Personal model 3.



```
ROS Time: 1685.67   ROS Elapsed: 166.03   Wall Time: 1529565639.86   Wall Elapsed: 233.37
```

Fig. 5. Result 1:Benchmark robot has reached the goal position and pass the test.



```
root@30c45b1256c1:/home/workspace/catkin_ws# rosrun udacity_bot navigation_goal
[ INFO] [1529649435.664406041, 1519.711000000]: Waiting for the move_base action server
[ INFO] [1529649439.075294704, 1522.197000000]: Connected to move_base server
[ INFO] [1529649439.075552416, 1522.197000000]: Sending goal
[ INFO] [1529649565.246622747, 1607.303000000]: Excellent! Your robot has reached the goal position.
```

```
ROS Time: 1630.12   ROS Elapsed: 109.81   Wall Time: 1529649596.17   Wall Elapsed: 159.64
```

Fig. 6. Result 1:Benchmark robot has reached the goal position and pass the test.

3.3.2 Packages Used

In this part, ROS AMCL package and Move Base package are used to localize the position and navigate the robot to the goal position.

3.3.3 Parameters

```
"base_local_planner_params.yaml"
TrajectoryPlannerROS:
  holonomic_robot: false
  yaw_goal_tolerance: 0.1
  xy_goal_tolerance: 0.1
  sim_time: 1.0
  meter_scoring: true
  max_vel_x: 0.4
  min_vel_x: 0.1
```

```
"costmap_common_params.yaml"
  map_type: costmap
  obstacle_range: 3.0
  raytrace_range: 5.0
  transform_tolerance: 1.0
  inflation_radius: 0.6
  observation_sources: laser_scan_sensor
  laser_scan_sensor: {sensor_frame: hokuyo,
  data_type: LaserScan,
  topic: /udacity_bot/laser/scan,
  marking: true, clearing: true}
```

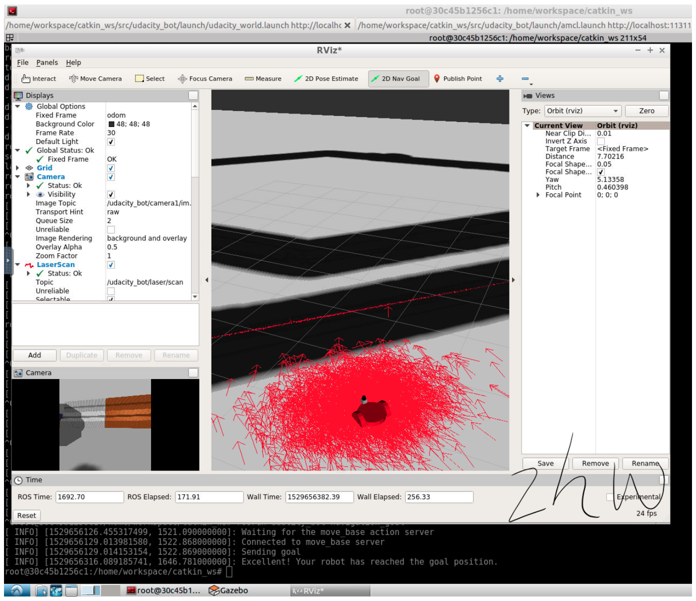
4 Results

The localization results look reasonable, it takes about 3min for the Benchmark robot to reach the goal and the develop model less than 3 min. It is not a smooth path to the goal and the robot get wrong direction at the beginning.

4.1 Localization Results

4.1.1 Benchmark

See the Fig. 5 and Fig. 6



```
ROS Time: 1692.70   ROS Elapsed: 171.91   Wall Time: 1529656382.39   Wall Elapsed: 256.33
```

```
[ INFO] [1529656126.455317499, 1521.090000000]: Waiting for the move_base action server
[ INFO] [1529656129.013981580, 1522.868000000]: Connected to move_base server
[ INFO] [1529656129.014153154, 1522.869000000]: Sending goal
[ INFO] [1529656316.089185741, 1646.781000000]: Excellent! Your robot has reached the goal position.
root@30c45b1256c1:/home/workspace/catkin_ws# 
```

Fig. 7. Result 2:Develop robot has reached the goal position and pass the test.

4.1.2 Student

See the Fig. 7

4.2 Technical Comparison

At first, the parameters used in Benchmark robot is also used to run the own robot model, the performance are all worse than the Benchmark. But some of them are more flexible and faster. After a long time tuning parameters, when improve the inflation_radius "0.3" to "0.6" and add parameters max_vel_x"0.4" and min_vel_x"0.1", it get a good result for the custom robot to reach the goal faster than the Benchmark robot. Inflation_radius parameters help the robot get away from the obstacle and quickly find the best orientation. Parameters max_vel_x and min_vel_x"0.1" help the robot to control the running speed and avoid hit the obstacle.

5 Discussion

There are many different develop models are test, in the project the develop robot performed better then the based model. The influences in conclusion, the control wheels size, the length between wheels, the position of the camera and Hokuyo and the stable of the chassis it's important to contribute the sensor layout and performance. The robot performed better with low center of gravity, smaller size of wheel size and a more stable chassis by adding four bottom wheel. And the shape of the chassis with circle in XY axis is greater than others to avoid hit the obstacle when change the orientation.

6 Conclusion / Future work

In the project, there are lots of work to utilize ROS packages to accurately localize a mobile robot inside a provided map in the Gazebo and RViz simulation environments. Building a good mobile robot is also a hard tasks for simulated and there are many more parameters in each package did not explore, with more parameters tuned the robot would achieve a better localization and perform result.

6.1 Modifications for Improvement

- Base Dimension
- Sensor Location
- Sensor Layout
- Sensor Amount
- Center of gravity
- Wheels size
- Wheels Amount
- chassis shape

6.2 Hardware Deployment

1) The Jetson TX2 board with ROS to explore.
2) Computation time/resource improvement.
3) Add more sensor with sensor fusion