7. AMCL adaptive Monte Carlo localization algorithm

7.1. Introduction

amcl stands for adaptive Monte Carlo localization, which is a probabilistic localization system for two-dimensional mobile robots. In fact, it is an upgraded version of the Monte Carlo localization method, using an adaptive KLD method to update particles and a particle filter to track the robot's posture based on a known map. According to the current implementation, this node is only applicable to laser scanning and laser maps. It can be extended to process other sensor data. amcl receives laser-based maps, laser scans, and transformation information, and outputs a posture estimate. At startup, amcl initializes its particle filter based on the provided parameters. Note that due to the default settings, if no parameters are set, the initial filter state will be a medium-sized particle cloud centered at (0,0,0).

Monte Carlo

- Monte Carlo method is also called statistical simulation method and statistical experimental method. It is a numerical simulation method that takes probabilistic phenomena as the research object. It is a calculation method that uses the sampling survey method to obtain statistical values to infer unknown characteristic quantities.
- For example: There is an irregular shape in a rectangle. How to calculate the area of the irregular shape? It is not easy to calculate. But we can approximate it. Take a bunch of beans, sprinkle them evenly on the rectangle, and then count the number of beans in the irregular shape and the number of beans in the remaining places. The area of the rectangle is known, so the area of the irregular shape is obtained by estimation. Take the robot positioning as an example. It is possible that it is in any position in the map. In this case, how do we express the confidence of a position? We also use particles. Where there are more particles, it means that the robot is likely to be there.

The biggest advantage of the Monte Carlo method

- The error of the method is independent of the dimension of the problem.
- For problems with statistical properties, it can be solved directly.
- There is no need to discretize continuous problems

Disadvantages of the Monte Carlo method

- Deterministic problems need to be converted into random problems.
- The error is a probabilistic error.
- Usually a large number of calculation steps N is required.

Particle filtering

- The number of particles represents the probability of something. Through a certain evaluation method (evaluating the probability of this thing), the distribution of particles is changed. For example, in robot positioning, a particle A, I think this particle is very likely to be at this coordinate (for example, this coordinate belongs to the "this thing" mentioned before), so I give it a high score. Next time when all the particles are rearranged, arrange more near this position. In this way, after a few more rounds, the particles will be concentrated in the position with high probability.

Adaptive Monte Carlo

- Solve the robot kidnapping problem. When it finds that the average score of particles suddenly decreases (which means that the correct particle is abandoned in a certain iteration), it will re-scatter some particles globally.
- Solved the problem of fixed number of particles, because sometimes when the robot positioning is almost achieved, for example, these particles are concentrated together, and it is unnecessary to maintain so many particles. At this time, the number of particles can be reduced.

7.2, Navigation positioning test

7.2.1, Startup

Turn off the automatic chassis service

```
sudo supervisorctl stop ChassisServer
```

Start the driver (robot side).

```
sudo supervisorctl restart LaserServer #start/stop Switch radar service (indoor version)
roslaunch scout_bringup scout_mini_robot_base.launch
```

- It is necessary to ensure that the radar starts normally. If the run `rostopic echo /scan`

prints empty and cannot obtain data, the startup is abnormal. Please restart the radar service command.

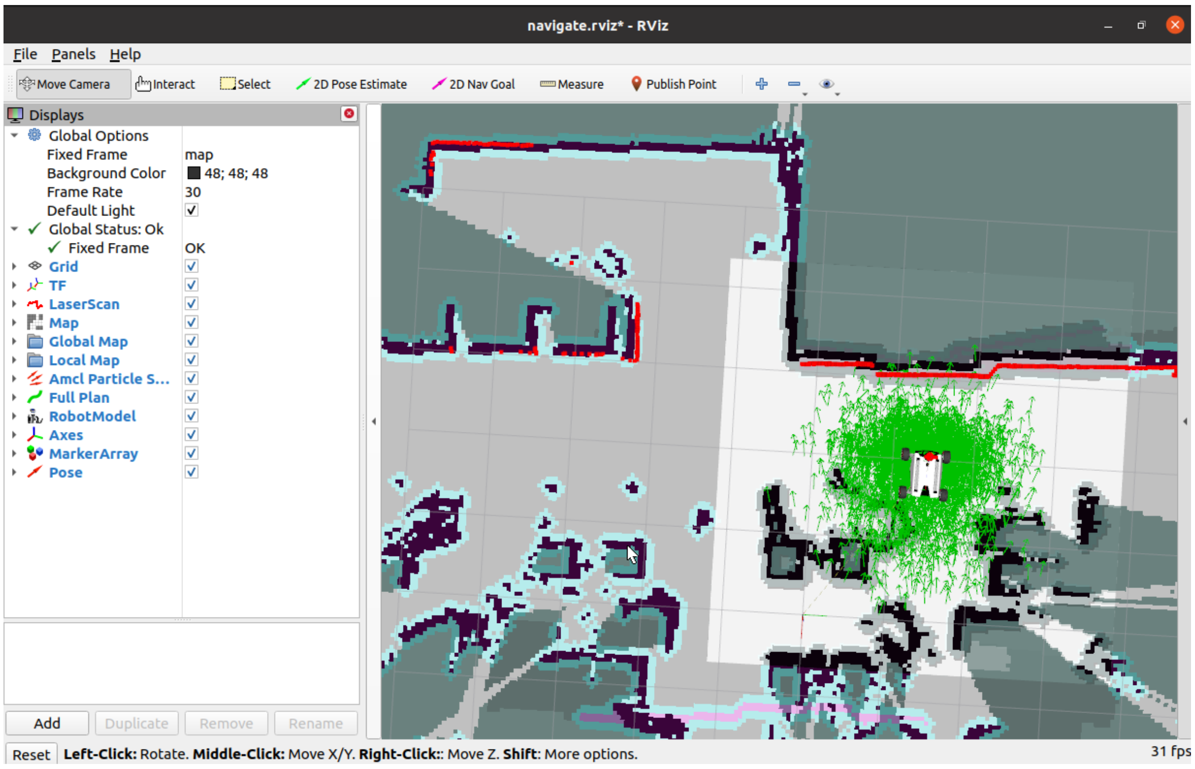
Start the navigation obstacle avoidance function (robot side). You can set parameters according to your needs and modify the launch file.

```
roslaunch yahboom_navrobo_nav yahboom_navrobo_navigation.launch use_rviz:=false map:=my_map
```

- [use_rviz] parameter: whether to open rviz.
- [map] parameter: map name, map to load.

Open the visualization interface

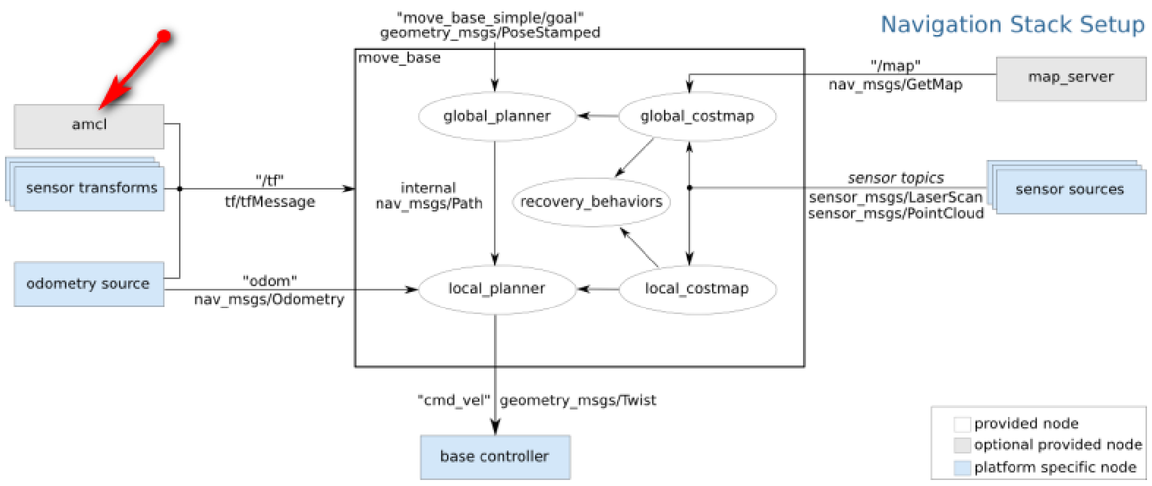
```
roslaunch yahboom_navrobo_nav view_navigate.launch
```



Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Default Light ✓
✓ Global Status: Ok
✓ Fixed Frame OK
Grid ✓
TF ✓
LaserScan ✓
Map ✓
Global Map ✓
Local Map ✓
Amcl Particle S... ✓
Full Plan ✓
RobotModel ✓
Axes ✓
MarkerArray ✓
Pose ✓

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click::** Move Z. **Shift**: More options.

7.2.2, Positioning Analysis



amcl is located in the upper left corner of the navigation frame. Its function is to tell the robot where it is.

After opening it, we can see a handful of green particles evenly scattered around the robot. When we move the robot randomly (keyboard or mouse control), all the particles move with it. Use the position of each particle to simulate a sensor information and compare it with the observed sensor information (usually laser), so as to assign a probability to each particle. Then regenerate the particles according to the generated probability. The higher the probability, the greater the probability of generation. After such iterations, all the particles will slowly converge together, and the exact position of the robot will be calculated.

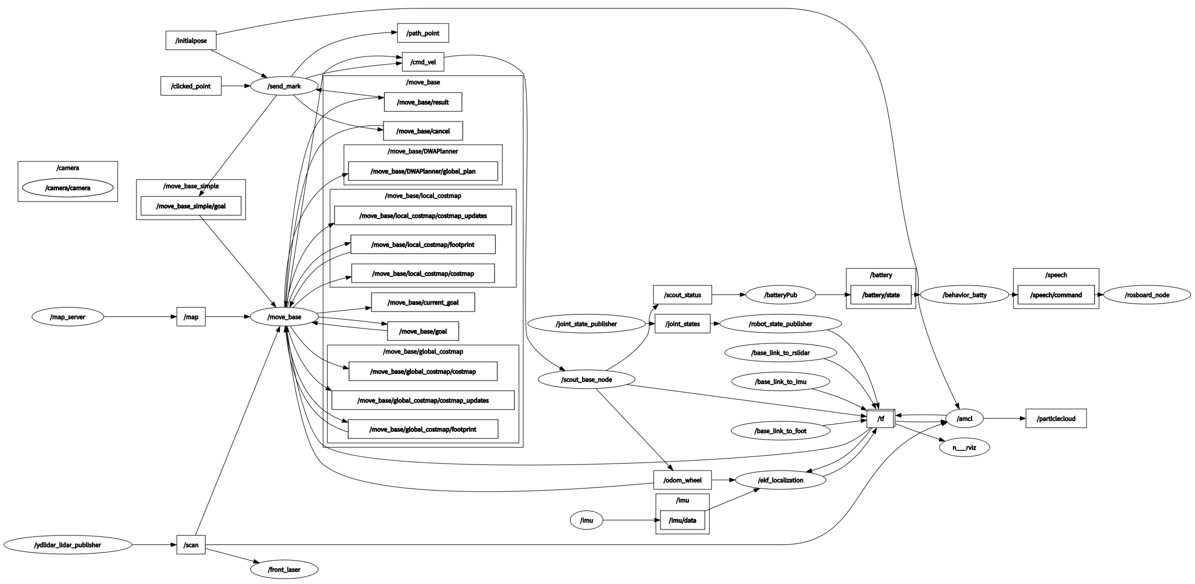
7.2, Topics and Services

Subscribe to Topics	Type	Description
scan	sensor_msgs/LaserScan	LiDAR data
tf	tf/tfMessage	Coordinate transformation information
initialpose	geometry_msgs/PoseWithCovarianceStamped	Mean and covariance used to (re)initialize particle filters.
map	nav_msgs/OccupancyGrid	After setting the use_map_topic parameter, AMCL subscribes to this topic to retrieve maps for laser-based positioning. New in navigation 1.4.2.
Publish topic	**Type**	**Description**
amcl_pose	geometry_msgs/PoseWithCovarianceStatmped	Pose estimate of the robot in the map, with covariance information
particlecloud	geometry_msgs/PoseArray	Pose estimate collection maintained by the particle filter
tf	tf/tfMessage	Publish the conversion from odom to map
Server	**Type**	**Description**
global_localization	std_stvs/Empty	Initialize global localization, all particles are randomly scattered in the free area of the map
request_normotion_update	std_stvs/Empty	Manually perform updates and publish updated particles
set_map	nav_msgs/SetMap	Service for manually setting a new map and pose.
Client	**Type**	**Description**

Subscribe to Topics	Type	Description
static_map	nav_msgs/GetMap	amcl calls this service to retrieve a map for laser positioning; start blocks getting maps from this service.

View Node

```
rosrun rqt_graph rqt_graph
```



7.3, Parameter Configuration

There are three categories of ROS parameters that can be used to configure amcl nodes: overall filter, laser model, and odometry model.

- Overall filter parameters

Parameter	Type	Default	Description
~min_particles	int	100	Minimum number of particles allowed
~max_particles	int	5000	Maximum number of particles allowed
~kld_err	double	0.1	Maximum error between the true distribution and the estimated distribution
~kld_z	double	0.99	Upper standard normal quantile of (1-p), where p is the probability that the error of the estimated distribution is less than kld_err
~update_min_d	double	0.2(m)	Translational distance required to perform one filter update
~update_min_a	double	pi/6.0(rad)	Rotational movement required to perform one filter update
~reseample_interval	int	2	Number of filter updates before resampling
~transform_tolerance	double	0.1(s)	Time to publish the transform, indicating that this transform is valid in the future
~recovery_alpha_slow	double	0.0	Exponential decay rate of the slow average weight filter, used to decide when to recover by adding random poses, 0.0 means disabled
~recovery_alpha_fast	double	0.0	Exponential decay rate of the fast average weight filter, used to decide when to recover by adding random poses, 0.0 means disabled
~initial_pose_x	double	0.0(m)	Initial pose average value (x), used to initialize the Gaussian distribution filter
~initial_pose_y	double	0.0(m)	Initial pose average value (y), used to initialize the Gaussian distribution filter
~initial_pose_a	double	0.0(m)	Initial pose average value (yaw), used to initialize the Gaussian distribution filter
~initial_cov_xx	double	0.5*0.5(m)	Average initial pose (x*x), used to initialize Gaussian distribution filter
~initial_cov_yy	double	0.5*0.5(m)	Average initial pose (y*y), used to initialize Gaussian distribution filter
~initial_cov_aa	double	(pi/12)*(pi/12)(rad)	Average initial pose (yaw*yaw), used to initialize Gaussian distribution filter

Parameter	Type	Default	Description
~gui_publish_rate	double	-1.0(Hz)	Maximum rate of publishing information during visualization, -1.0 means disabled
~save_pose_rate	double	0.5(Hz)	Maximum rate of storing pose estimates initial_pose_ and covariance initial_cov_ in the parameter server, used for subsequent filter initialization. -1.0 means disabled
~use_map_topic	bool	false	When set to true, amcl will subscribe to the map topic instead of receiving the map via a service call
~first_map_only	bool	false	When set to true, amcl will only use the first map it subscribes to instead of updating the received map every time
~selective_resampling	bool	false	When set to true, will reduce the resampling rate when not needed and help avoid particle starvation. Resampling will only occur when the number of effective particles (N_eff=1/(sum(k_i^2)) is less than half of the current number of particles.

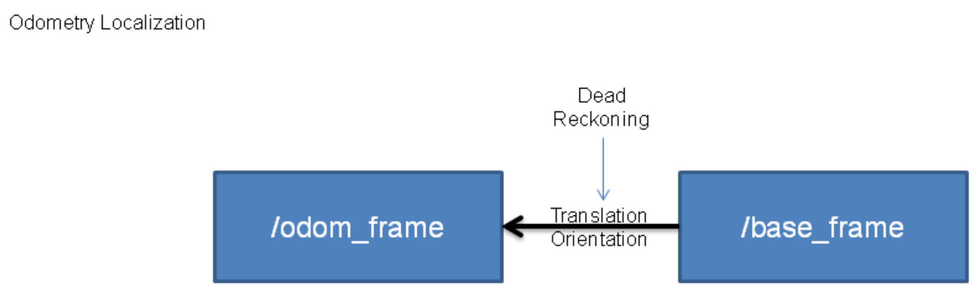
- Laser Model Parameters

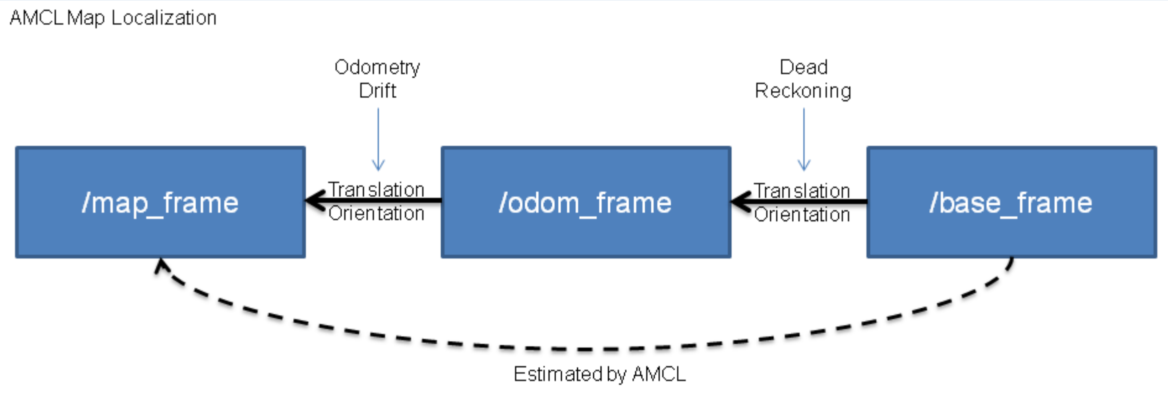
Parameter	Type	Default	Description
~laser_min_range	double	-1.0	Minimum scan range, set to -1, the minimum use range reported by the laser radar.
~laser_max_range	double	-1.0	Maximum scan range, set to -1, the maximum use range reported by the laser radar.
~laser_max_beams	int	30	How many evenly spaced beams to use in each scan when updating the filter
~laser_z_bit	double	0.95	Blending weight for the z_bit portion of the model
~laser_z_short	double	0.1	Blending weight for the z_short portion of the model
~laser_z_max	double	0.05	Blending weight for the z_max portion of the model
~laser_z_rand	double	0.05	Blending weight for the z_rand portion of the model
~laser_sigma_hit	double	0.2(m)	Standard deviation of the Gaussian model used in the z_hit portion of the model
~laser_lambda_short	double	0.1	Exponential decay parameter for the z_short portion of the model
~laser_likelihood_max_dist	double	2.0(m)	Maximum distance on the map at which to measure obstacle inflation
~laser_model_type	string	"likelihood_field"	Model choice, bean, likelihood_field, or likelihood_field_prob

- Odometry model parameters

Parameter	Type	Default	Description
~odom_model_type	string	"diff"	Model choice, diff, omni, diff-corrected, or omni-corrected
~odom_alpha1	double	0.2	Specifies the expected noise in the odometry rotation estimate, based on the rotation component of the robot's motion
~odom_alpha2	double	0.2	Specifies the expected noise in the odometry rotation estimate, based on the translation component of the robot's motion
~odom_alpha3	double	0.2	Specifies the expected noise in the odometry translation estimate, based on the translation component of the robot's motion
~odom_alpha4	double	0.2	Specifies the expected noise in the odometry translation estimate based on the rotational component of the robot motion
~odom_alpha5	double	0.2	Translation-related noise parameter (used only in model omni)
~odom_frame_id	string	"odom"	The coordinate frame of the odometry
~base_frame_id	string	"base_link"	The coordinate frame of the robot chassis
~global_frame_id	string	"map"	The coordinate frame published by the positioning system
~tf_broadcast	bool	true	When set to false, amcl will not publish the coordinate transformation between map and odom

7.4 Coordinate transformation





Comparison of two methods Odometry Localization and AMCL Map Localization:

Odometry Localization: The rotation speed of the wheel can be obtained through the feedback of the motor encoder, the current wheel speed can be obtained according to the radius of the wheel, the overall moving speed of the robot can be obtained according to the forward kinematics model, and the total mileage can be obtained by integrating according to time.

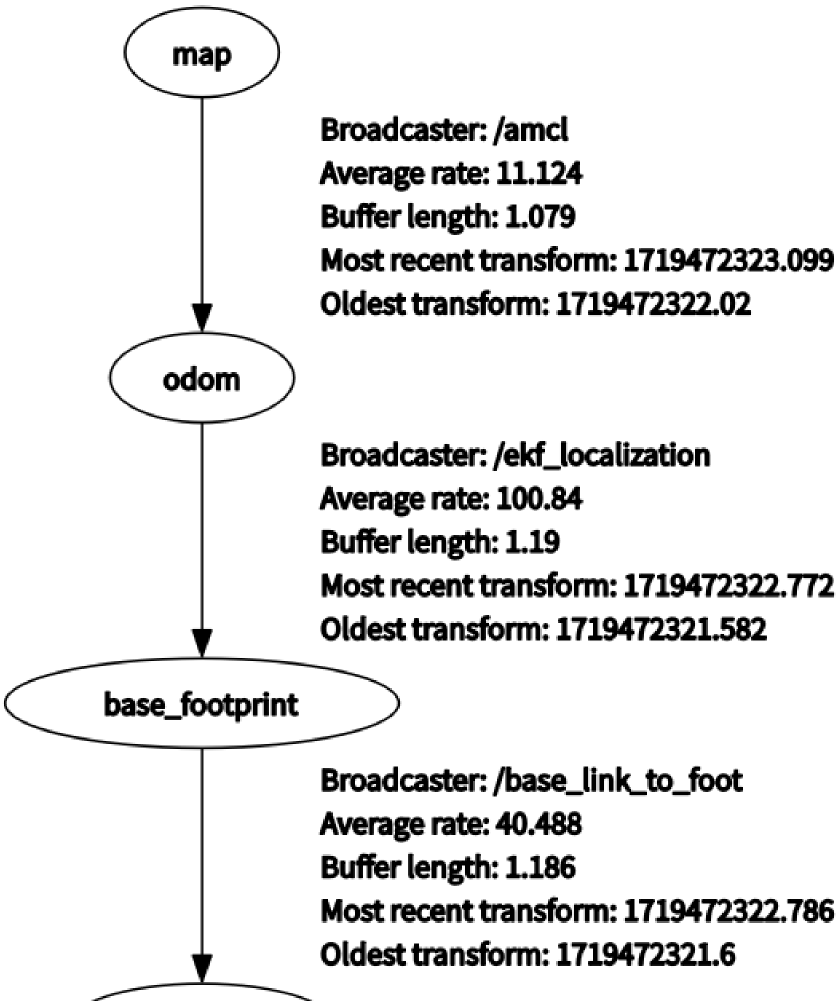
AMCL Map Localization: In amcl, the position information of the base_frame corresponding to the map_frame has been obtained through particle filtering positioning, but in order to ensure that the base_frame has only one parent node, this conversion is not directly published, because the parent node of the base_frame is already odom_frame, then amcl can only publish the conversion from map_frame to odom_frame, which also has the advantage that we can get the cumulative error of odom.

We have been saying that the distance information obtained by the wheel odometer will have cumulative errors, which is similar to the car skidding. If the moving distance is calculated only by the wheel odometer, it will be wrong:

For example: the robot moves forward in a straight line, and the result obtained by the wheel odometer should be 1 meter, so the distance from odom_frame should be 1 meter, but because of the slipping in place, the robot actually finds that it has not moved at all through amcl positioning, so amcl needs to publish the conversion from map_frame to odom_frame to remove the 1 meter error, so as to maintain the normality of the entire tf tree.

The conversion relationship between coordinates.

```
rosrun rqt_tf_tree rqt_tf_tree
```


map
Broadcaster: /amcl
Average rate: 11.124
Buffer length: 1.079
Most recent transform: 1719472323.099
Oldest transform: 1719472322.02
odom
Broadcaster: /ekf_localization
Average rate: 100.84
Buffer length: 1.19
Most recent transform: 1719472322.772
Oldest transform: 1719472321.582
base_footprint
Broadcaster: /base_link_to_foot
Average rate: 40.488
Buffer length: 1.186
Most recent transform: 1719472322.786
Oldest transform: 1719472321.6

As can be seen from the above figure, the coordinate conversion from map->odom is completed by the amcl node.