DWB parameter debugging

1. Car navigation speed adjustment related parameters

min_vel_x: The minimum linear velocity in the x direction, which is a negative value. After a negative value is given, the car can navigate backwards;

max_vel_x: the maximum linear velocity in the x direction, which is a positive value. If the forward speed of the car navigation is too slow, this parameter can be appropriately increased;

min_speed_theta: The minimum angular velocity, which is a negative value. After a negative value is given, the car can rotate counterclockwise to adjust its posture;

max_vel_theta: The maximum angular velocity, which is a positive value. If the car rotates too slowly, you can increase this parameter appropriately;

acc_lim_x: The linear acceleration limit in the x direction. If the car walks in a twisty way, you can adjust this value and acc_lim_theta appropriately;

acc_lim_theta: angular acceleration limit value. This value and acc_lim_x can be adjusted appropriately if the car walks in a twisty way;

2. Car target point related parameters

xy_goal_tolerance: The error tolerance of the target point of the car in the XY direction, which can be understood as the error range between the position of the car on the map and the position of the actual target point. The unit here is m;

yaw_goal_tolerance: The target point error tolerance of the car's front steering. It can be understood that it is the error range between the position of the car on the map and the position of the actual target point. The unit here is radians;

The car adjusts its posture back and forth near the target point, and the values of xy_goal_tolerance and yaw_goal_tolerance can be appropriately increased.

3. Car obstacle avoidance related parameters

Global Map Section

global_costmap.obstacle_max_range: The maximum range of obstacles measured by the global map. Adjust this parameter according to the actual distance between the car and the obstacle.

global_costmap.obstacle_min_range: The minimum range for measuring obstacles in the global map. Adjust this parameter according to the actual distance between the car and the obstacle.

global_costmap.obstacle_max_range: The maximum range of ray tracing, adjusted according to the car's perception ability and the distribution of obstacles in the environment. The car's sensor can accurately detect distant obstacles. This value can be set larger to better detect distant obstacles.

global_costmap.obstacle_min_range: The minimum range of ray tracing, adjusted according to the car's perception ability and the distribution of obstacles in the environment. The car's sensor range is limited, and this value may need to be set smaller to ensure that nearby obstacles can be accurately detected.

global_costmap.update_frequency: Global cost map release frequency. According to the performance of the board, this parameter can be adjusted appropriately so that obstacles can be detected, and then a path around the obstacles can be planned based on the obstacle information.

global_costmap.inflation_radius: The expansion coefficient of the global cost map. If the planned path cannot avoid obstacles, this value can be appropriately reduced.

global_costmap.robot_radius: The radius of the car. According to the actual size of the car and the value of inflation_radius, modify this value to be able to bypass obstacles.

Local map part

local_costmap.obstacle_max_range: The maximum range of obstacles measured in the local map. Adjust this parameter according to the actual distance between the car and the obstacle.

local_costmap.obstacle_min_range: The minimum range for detecting obstacles in the local map. Adjust this parameter according to the actual distance between the car and the obstacle.

local_costmap.obstacle_max_range: The maximum range of ray tracing, adjusted according to the car's perception ability and the distribution of obstacles in the environment. The car's sensor can accurately detect distant obstacles. This value can be set larger to better detect distant obstacles.

local_costmap.obstacle_min_range: The minimum range of ray tracing, adjusted according to the car's perception ability and the distribution of obstacles in the environment. The car's sensor range is limited, and this value may need to be set smaller to ensure that nearby obstacles can be accurately detected.

local_costmap.update_frequency: Local cost map publishing frequency. According to the performance of the board, this parameter can be adjusted appropriately so that obstacles can be detected, and then a path around the obstacles can be planned based on the obstacle information.

local_costmap.inflation_radius: The expansion coefficient of the local cost map. If the planned path cannot avoid obstacles, this value can be appropriately reduced.

local_costmap.robot_radius: The radius of the car. According to the actual size of the car and the value of inflation_radius, modify this value to be able to bypass obstacles.

4. Path related parameters

vx_samples: The number of velocity samples in the x direction during path planning affects the smoothness of the path and the speed change of the robot during movement. Increasing the value of this parameter can increase the exploration of the velocity space, making it possible to find better solutions. path. However, too many velocity samples also increase computational complexity and computational time.

vtheta_samples: The number of angular velocity samples performed in path planning affects the smoothness of the path and the angular velocity changes of the robot during movement. Increasing the value of this parameter can increase the exploration of the angular velocity space, and it is possible to find a better path. However, too many angular velocity samples will also increase computational complexity and calculation time.

controller_frequency: The frequency of the controller, that is, the frequency at which the controller performs control. According to the performance of the own board, appropriately adjusting this parameter will help to better plan real-time paths;

PathDist.scale: Adjust the scaling factor of path distance preference, which affects the robot's preference for straight paths and curved paths during the path planning process. Increasing the value of this parameter will make the robot more inclined to choose straight paths, while decreasing this parameter The value will make the robot more inclined to choose a curved path. When the space is small, this value can be increased appropriately to avoid detecting obstacles and re-planning the path.

GoalDist.scale: Adjust the scaling factor of target distance preference, which affects the robot's preference for distance from the target during path planning. Increasing the value of this parameter will make the robot more inclined to choose a path closer to the target, while decreasing it will The value of this parameter will make the robot more inclined to choose a path farther away from the goal. To reach the target point as soon as possible and there are not many obstacles in the environment, you can increase this value appropriately.

PathAlign.forward_point_distance: Determines the distance parameter of the forward point of the robot on the path. It is used to determine which direction of the path the robot should move in order to better follow the path. Larger values will cause the robot to look further towards the path. front, potentially making it more responsive to changes in path, but potentially increasing the risk of over-adjustment. Smaller values will move the robot closer to the waypoint at its current location, potentially resulting in more stable path tracking, but the robot may be slower to respond to path changes.

GoalAlign.forward_point_distance: Determines the distance of the forward point selected by the robot when heading toward the goal. It is used to determine in which direction the robot should move toward the goal in order to better move toward the goal and avoid deviating from the path. Larger values will cause the robot to look farther in front of the target point, possibly making it move towards the target faster, but may also increase the risk of over-adjustment. Smaller values bring the robot closer to the target point at its current location, potentially resulting in more stable target alignment, but the robot may move slower toward the target.