轮 趣 科 技

ROS2 导航算法切换

推荐关注我们的公众号获取更新资料

版本说明：

版本	日期	内容说明
V1.0	2024/09/20	第一次发布

网址：www.wheeltec.net

序言

Nav2 提供了许多现成的局部规划器插件，包括 DWB(Dynamic Window Buffer)控制器、MPPI(Model Predictive Path Integral)控制器和 RPP(Regulated Pure Pursuit)控制器。本文主要讲述 ROS2 中常用的三种局部路径规划算法的切换方式，以及如何针对不同机器人模型选择最佳的规划算法。

目录

1. NAV2 中的规划器介绍

全局规划器的任务是根据特定的目标函数计算路径。根据所选的命名法和算法，路径也称为行程路线。有两个典型示例，一个是计算到目标位置的路径规划（例如从当前位置到目标姿势）或完全覆盖完整覆盖范围（例如覆盖所有自由空间的路径规划）。规划器可以访问全局环境表示和传感器的缓存数据。规划器可以具有计算最短路径，计算覆盖路径和沿稀疏或预定义路线计算路径的功能。Nav2 中规划器的一般任务是计算从当前位置到目标姿势的有效且可能最佳的路径。

Nav2 中常用的规划器如下所示。NavFn 规划器是一种使用 Dijkstra 或 A*算法的导航功能规划器。Smac 2D 规划器使用 4 或 8 个连通邻域实现 2D A* 算法，具有更平滑和多分辨率的特点。Theta Star 规划器（连续环境下的平滑任意角度路径规划，Theta Star 规划器是 Theta*实现的扩展，它沿着图的边缘传播信息，但不将路径约束在图的边缘（以“任意角度”寻找路径））是使用视线算法通过创建非离散方向路径段来实现的。

Plugin Name	Creator	Description	Drivetrain support
NavFn Planner	Eitan Marder-Eppstein & Kurt Konolige	A navigation function using A* or Dijkstras expansion, assumes 2D holonomic particle	Differential, Omnidirectional, Legged
SmacPlannerHybrid (formerly *SmacPlanner*)	Steve Macenski	A SE2 Hybrid-A* implementation using either Dubin or Reeds-shepp motion models with smoother and multi-resolution query. Cars, car-like, and ackermann vehicles. Kinematically feasible.	Ackermann, Differential, Omnidirectional, Legged
SmacPlanner2D	Steve Macenski	A 2D A* implementation Using either 4 or 8 connected neighborhoods with smoother and multi-resolution query	Differential, Omnidirectional, Legged
SmacPlannerLattice	Steve Macenski	An implementation of State Lattice Planner using pre-generated minimum control sets for kinematically feasible planning with any type of vehicle imaginable. Includes generator script for Ackermann, diff, omni, and legged robots.	Differential, Omnidirectional, Ackermann, Legged, Arbitrary / Custom
ThetaStarPlanner	Anshumaan Singh	An implementation of Theta* using either 4 or 8 connected neighborhoods, assumes the robot as a 2D holonomic particle	Differential, Omnidirectional

图 1-1-1 导航参数的 DWB 插件内容

针对以上常用的规划器更详细的介绍，可以在 NAV2 官网查阅。在 WHEELTEC_NAV2 源码中，常用的规划期是 NavfnPlanner 和 SmacPlannerHybrid。导航参数路径为：

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/param/wheeltec_param/
```

以 mini 麦轮的导航参数为例，使用 NavfnPlanner 规划器，插件的源码如图所示。

图 1-1-2 mec 导航参数规划器插件内容

以 mini 阿克曼的导航参数为例，使用 SmacPlannerHybrid 规划器，插件的源码如图所示。

图 1-1-3 akm 导航参数规划器插件内容

2. NAV2 中的控制器介绍

2.1 DWB 局部路径规划算法

DWB 控制器是 ROS1 导航局部路径规划算法 DWA(Dynamic Window Approach)控制器的优化版。DWB 是一种经典的局部路径规划算法，广泛应用于移动机器人避障任务中。其核心思想是通过机器人当前的运动状态（速度和加速度）来生成多个运动轨迹，并评估这些轨迹的可行性和优越性，从中选择最优的轨迹进行执行。DWB 的基本流程如下：

1、在给定的时间窗口内，考虑机器人的动态约束（如最大速度、加速度等）生成可能的运动轨迹。

2、使用启发函数对轨迹进行评分，启发函数通常考虑目标距离、障碍物距离和轨迹平滑度等因素。

3、选择评分最高的轨迹作为下一步的移动轨迹。

DWB 的优势在于能够快速反应并适应动态环境变化，同时考虑了机器人自身的动力学约束。NAV2 仓库源码链接：

https://github.com/ros-navigation/navigation2/tree/main/nav2_dwb_controller

在 WHEELTEC_NAV2 源码中，导航参数路径为：

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/param/wheeltec_param/
```

以 mini 麦轮的导航参数为例，控制器插件的源码如图所示。

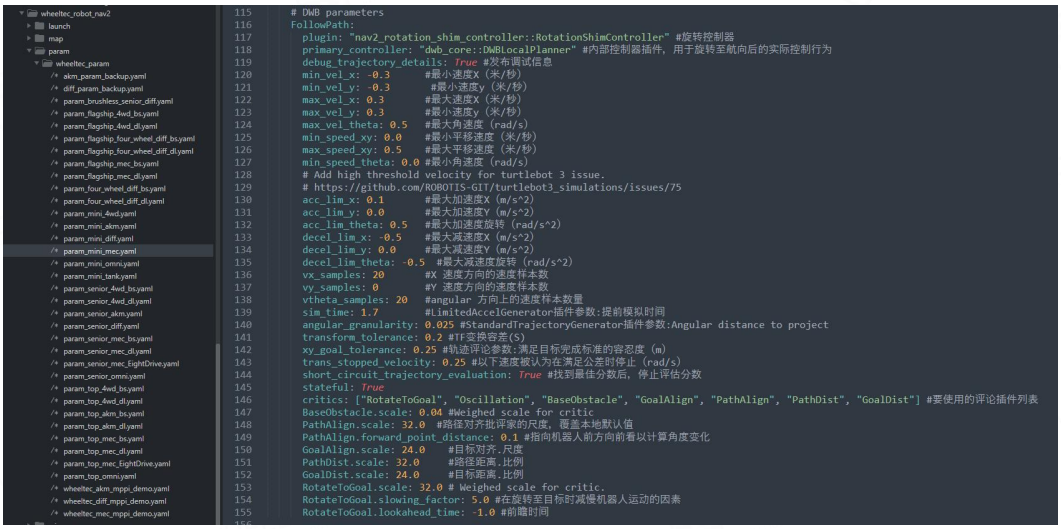


图 2-1-1 导航参数的 DWB 插件内容

2.2 MPPI 局部路径规划算法

MPPI(Model Predictive Path Integral）模型预测路径积分控制是一种基于采样的最优控制方法，常用于路径规划和轨迹跟踪问题。它结合了模型预测控制（MPC）的思想和路径积分的优化原理，能够在不确定和复杂的环境下进行实时路径规划。MPPI 算法的核心思想是通过在每个控制周期内预测未来的路径，并基于随机采样来优化控制输入。这种方法不仅考虑了当前状态，还预测了未来状态，从而提高了路径规划的效果。MPPI 的基本工作流程如下：

1、通过随机采样生成一组可行的控制输入序列，并使用系统动力学模型预测其未来轨迹。

2、根据目标函数（通常包括目标距离、避障、控制代价等）对每条轨迹进行评

分。

3、使用路径积分方法加权这些轨迹，计算最优的控制输入序列。

MPPI 的特点是能够处理非线性系统，且能够在存在不确定性和噪声的环境下工作。相比传统的 MPC，MPPI 更加适合复杂的非线性场景和实时路径规划任务，但计算量相对较大。NAV2 仓库源码链接：

https://github.com/ros-navigation/navigation2/tree/main/nav2_mppi_controller

在 WHEELTEC_NAV2 源码中，导航参数路径为：

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/param/wheeltec_param/
```

以 mini 麦轮的导航参数为例，控制器插件的源码如图所示。

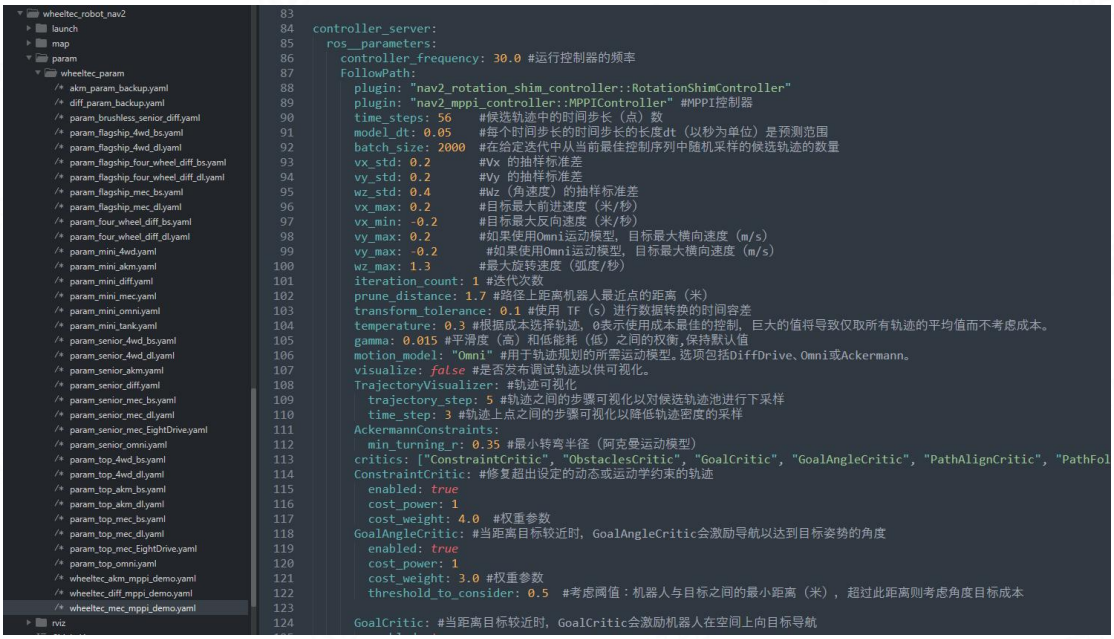


图 2-2-1 导航参数的 MPPI 插件内容

2.3 RPP 局部路径规划算法

RPP 快速随机路径规划算法（Regulated Pure Pursuit）是一种局部路径规划算法，主要用于移动机器人在动态环境中进行路径跟踪。RPP 控制器实现了 Pure Pursuit 控制器的扩展，专门针对服务/工业机器人的需求。通过路径曲率调节线性速度，可以机器人帮助减少盲角高速时的超调，使操作更加安全。该算法可以在接近其他障碍物时减速，机器人可以在附近发生潜在碰撞时自动减慢移动速度，它还实现了自适应前瞻点功能，可以通过速度进行缩放，从而在更大范围的平移速度下实现更稳定的行为。主要通过设定一个目标点，使机器人按照预定轨迹行

驶。该算法简洁易于实现，适用于直线和曲线路径。RPP 的基本工作流程如下：

1、获取当前机器人位置和路径点，定义机器人需要跟踪的全局路径。

2、在离散路径中，RPP 会根据机器人的当前位置选择一个目标点。

3、一旦确定了目标点，RPP 需要计算从当前机器人位置到目标点所需的转向角。

4、速度条件，RPP 算法除了控制转向，还会对机器人的速度进行调节，以确保平稳行驶和精确跟踪。

RPP 算法的优势在于通过几何关系进行目标点的追踪，逻辑清晰，易于实现和调试。在直线和轻微弯曲的路径上，RPP 具有较高的跟踪精度，尤其是在环境较简单的情况下表现良好。NAV2 仓库源码链接：

https://github.com/ros-navigation/navigation2/tree/main/nav2_regulated_pure_pursuit_controller

在 WHEELTEC_NAV2 源码中，导航参数路径为：

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/param/wheeltec_param/
```

以 mini 阿克曼的导航参数为例，控制器插件的源码如图所示。

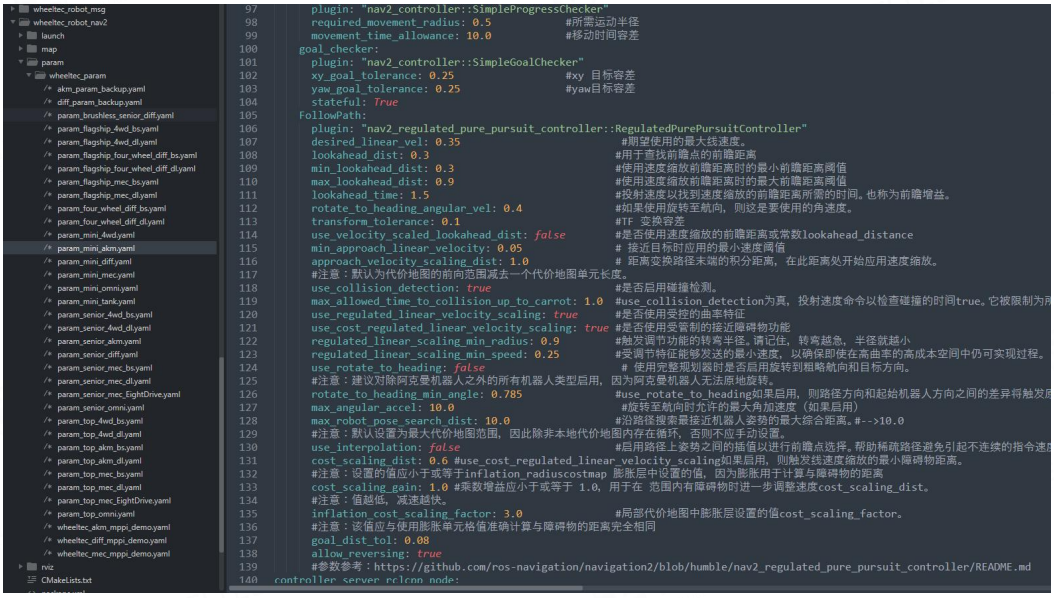


图 2-3-1 导航参数的 RPP 插件内容

3. 针对不同车型选择控制器插件

在为全向机器人、差速机器人和阿克曼机器人选择局部路径规划算法时，需要考虑各自的运动特性和动力学约束。每种机器人都有不同的运动模型和控制要求，因此适合的路径规划算法也不同。

3.1 全向机器人

全向机器人能够在任何方向上自由移动，可以独立控制前进、后退、侧向移动和旋转。因此，它的运动模型相对简单，运动自由度高。这类机器人在局部路径规划中需要快速灵活的避障能力。

适合全向机器人的局部规划算法包括：

- DWB 控制器：全向机器人能够利用 DWB 的动态窗口法很好地在复杂环境中实时避障。DWB 能够考虑机器人的动力学限制，通过评估多个可能轨迹，找到适合的运动路径。由于全向机器人没有传统的转向限制，DWB 生成的轨迹更加灵活，能够充分利用其运动自由度。
- MPPI 控制器：MPPI 可以基于全向机器人的动力学模型，实时生成多个轨迹并优化选择最优的运动路径。这对于需要高效避障且保持轨迹平滑的全向机器人非常有用。由于全向机器人没有复杂的运动限制，MPPI 的非线性优化方法能够提供高度灵活和高效的控制效果。

在 WHEELTEC_ROS2 机器人中，全向机器人通常使用 DWB 控制器来做局部路径规划。DWB 和 MPPI 都是理想选择。全向机器人运动灵活，可以充分利用 DWB 实时避障的能力，而 MPPI 能提供更平滑和优化的运动控制。

3.2 差速机器人

差速机器人通过两个独立的轮子控制前进和转向，但没有侧向移动的能力。虽然只能通过调整左右轮的速度来改变行驶方向，因此转弯半径受到限制，在允许原地自转的麦轮和四驱机器人中，转弯半径为 0。差速机器人的运动模型相对简单，但其运动自由度较低，需要考虑转向和轨迹平滑性。

适合全向机器人的局部规划算法包括：

- DWB 控制器：DWB 是差速机器人最常用的局部路径规划算法之一。DWB 考虑了机器人左右轮的速度限制以及加速度限制，能够在这种运动约束下生成可行的轨迹。DWB 通过对不同轨迹进行评分，选择既避障又能平稳转向的路径，非常适合差速机器人这种需要平稳转弯的场景。
- MPPI 控制器：MPPI 可以基于差速机器人的动力学模型，实时生成多个轨迹并优化选择最优的运动路径。这对于需要高效避障且保持轨迹平滑的差速机器人非常有用。由于差速机器人没有复杂的运动限制，MPPI 的非线性优化方法能够提供高度灵活和高效的控制效果。
- RPP 控制器：RPP 是路径跟踪算法，也适合差速驱动机器人，特别是在路径已知的情况下。由于 RPP 直接根据几何关系计算转向角，并且差速驱动通过速度差进行转向，RPP 的转向控制相对简单。

在 WHEELTEC_ROS2 机器人中，差速机器人通常使用 DWB 控制器来做局部路径规划，因为它能高效地处理差速机器人的运动约束，生成平滑可行的轨迹。

3.3阿克曼机器人

阿克曼机器人模拟汽车的运动模式，使用前轮转向、后轮驱动的方式前进。它只能沿某个转向角移动，无法像全向或差速机器人那样自由控制转向，因此其运动学更加复杂。阿克曼机器人通常具有较大的最小转弯半径，需要规划更加平滑和符合转向约束的路径。

适合的局部路径规划算法：

- MPPI 控制器：阿克曼机器人由于其运动学限制，适合使用能够处理复杂非线性系统的 MPPI 算法。MPPI 能够基于阿克曼运动模型生成未来的轨迹并优化控制输入，确保转弯角度和速度符合阿克曼运动学要求。这种方法不仅能够处理阿克曼机器人的转向约束，还可以在动态环境中实时进行路径优化，确保安全避障。
- RPP 控制器：RPP 非常适合阿克曼转向的车辆，因为它根据目标点计算转向角度，这与阿克曼车辆的转向控制模型高度匹配。该算法能很好地处理直线和弯道路径，尤其适合有明确路径的车辆导航场景。

在 WHEELTEC_ROS2 机器人中，差速机器人通常使用 RPP 控制器来做局部路径规划。

3. 4WHEELTEC NAV2 控制器插件切换

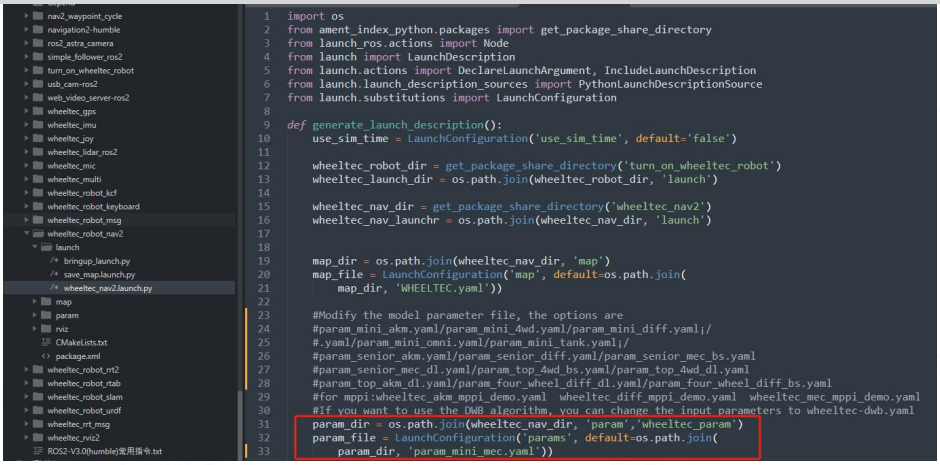
在 WHEELTEC_NAV2 源码中，导航参数路径为：

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/param/wheeltec_param/
```

WHEELTEC NAV2 导航文件夹中，包含了全向机器人（omni.yaml）、差速机器人（diff.yaml）、四驱机器人（4wd.yaml）、履带机器人（tank.yaml）、麦轮机器人（mec.yaml）、阿克曼机器人（akm.yaml）这些机器人的运动模型对应的参数文件。其中全向机器人、差速机器人、四驱机器人、履带机器人和麦轮机器人默认使用 DWB 控制器，阿克曼机器人默认使用 RPP 控制器。

参数文件在 wheeltec_nav2.launch.py 中被调用，具体路径如下：

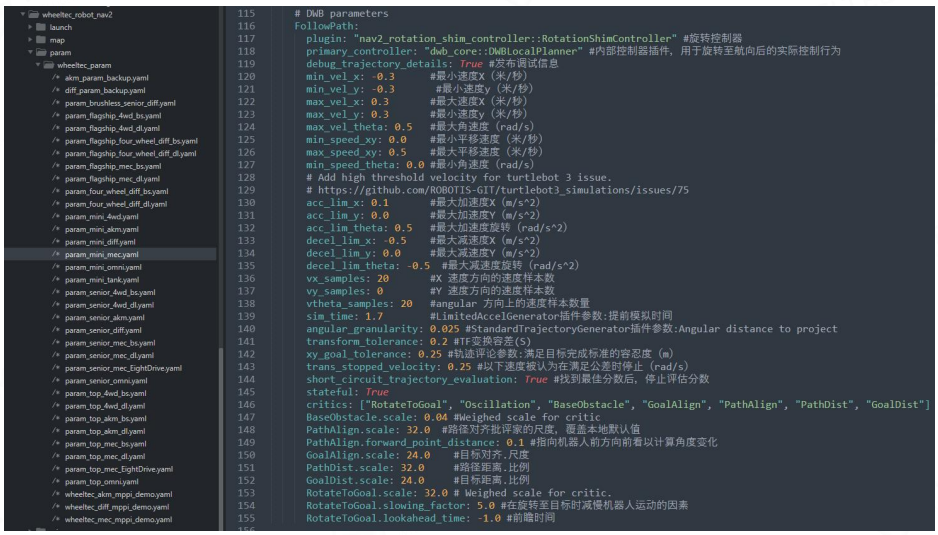
```
~/wheeltec_ros2/src/wheeltec_robot_nav2/launch/wheeltec_nav2.launch.py
```



```python
import os
from ament_index_python.packages import get_package_share_directory
from launch_ros.actions import Node
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument, IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch.substitutions import LaunchConfiguration

def generate_launch_description():
    use_sim_time = LaunchConfiguration('use_sim_time', default='false')

    wheeltec_robot_dir = get_package_share_directory('turn_on_wheeltec_robot')
    wheeltec_launch_dir = os.path.join(wheeltec_robot_dir, 'launch')

    wheeltec_nav_dir = get_package_share_directory('wheeltec_nav2')
    wheeltec_nav_launchr = os.path.join(wheeltec_nav_dir, 'launch')


    map_dir = os.path.join(wheeltec_nav_dir, 'map')
    map_file = LaunchConfiguration('map', default=os.path.join(
        map_dir, 'WHEELTEC.yaml'))

    #Modify the model parameter file, the options are
    #param_mini_akm.yaml/param_mini_4wd.yaml/param_mini_diff.yaml¡/
    #.yaml/param_mini_omni.yaml/param_mini_tank.yaml¡/
    #param_senior_akm.yaml/param_senior_diff.yaml/param_senior_mec_bs.yaml
    #param_senior_mec_dl.yaml/param_top_4wd_bs.yaml/param_top_4wd_dl.yaml
    #param_top_akm_dl.yaml/param_four_wheel_diff_dl.yaml/param_four_wheel_diff_bs.yaml
    #for mppi:wheeltec_akm_mppi_demo.yaml  wheeltec_diff_mppi_demo.yaml  wheeltec_mec_mppi_demo.yaml
    #If you want to use the DWB algorithm, you can change the input parameters to wheeltec-dwb.yaml
    param_dir = os.path.join(wheeltec_nav_dir, 'param','wheeltec_param')
    param_file = LaunchConfiguration('params', default=os.path.join(
        param_dir, 'param_mini_mec.yaml'))
```

图 3-4-1 wheeltec_nav2.launch.py 文件内容

以 mini 麦轮的导航参数为例，控制器插件的源码如图所示。默认为 DWB 控制器。



```
# DWB parameters
FollowPath:
  plugin: "nav2_rotation_shim_controller::RotationShimController" #旋转控制器
  primary_controller: "dwb_core::DWBLocalPlanner" #内部控制器插件，用于旋转至航向后的实际控制行为
  debug_trajectory_details: True #发布调试信息
  min_vel_x: -0.3      #最小速度X（米/秒）
  min_vel_y: -0.3       #最小速度y（米/秒）
  max_vel_x: 0.3       #最大速度X（米/秒）
  max_vel_y: 0.3       #最小速度y（米/秒）
  max_vel_theta: 0.5   #最大角速度（rad/s）
  min_speed_xy: 0.0    #最小平移速度（米/秒）
  max_speed_xy: 0.5    #最大平移速度（米/秒）
  min_speed_theta: 0.0 #最小角速度（rad/s）
  # Add high threshold velocity for turtlebot 3 issue.
  # https://github.com/ROBOTIS-GIT/turtlebot3_simulations/issues/75
  acc_lim_x: 0.1       #最大加速度X（m/s^2）
  acc_lim_y: 0.0       #最大加速度Y（m/s^2）
  acc_lim_theta: 0.5   #最大加速度旋转（rad/s^2）
  decel_lim_x: -0.5    #最大减速度X（m/s^2）
  decel_lim_y: 0.0     #最大减速度Y（m/s^2）
  decel_lim_theta: -0.5  #最大减速度旋转（rad/s^2）
  vx_samples: 20       #X 速度方向的速度样本数
  vy_samples: 0        #Y 速度方向的速度样本数
  vtheta_samples: 20   #angular 方向上的速度样本数量
  sim_time: 1.7        #LimitedAccelGenerator插件参数:提前模拟时间
  angular_granularity: 0.025 #StandardTrajectoryGenerator插件参数:Angular distance to project
  transform_tolerance: 0.2 #TF变换容差(S)
  xy_goal_tolerance: 0.25 #轨迹评论参数:满足目标完成标准的容忍度（m）
  trans_stopped_velocity: 0.25 #以下速度被认为在满足公差时停止（rad/s）
  short_circuit_trajectory_evaluation: True #找到最佳分数后，停止评估分数
  stateful: True
  critics: ["RotateToGoal", "Oscillation", "BaseObstacle", "GoalAlign", "PathAlign", "PathDist", "GoalDist"]
  BaseObstacle.scale: 0.04 #Weighed scale for critic
  PathAlign.scale: 32.0  #路径对齐批评家的尺度，覆盖本地默认值
  PathAlign.forward_point_distance: 0.1 #指向机器人前方向前看以计算角度变化
  GoalAlign.scale: 24.0    #目标对齐.尺度
  PathDist.scale: 32.0     #路径距离.比例
  GoalDist.scale: 24.0     #目标距离.比例
  RotateToGoal.scale: 32.0 # Weighed scale for critic.
  RotateToGoal.slowing_factor: 5.0 #在旋转至目标时减慢机器人运动的因素
  RotateToGoal.lookahead_time: -1.0 #前瞻时间
```

图 3-4-2 参数文件内容

① 切换为 MPPI 局部路径规划算法

修改导航启动文件，麦轮机器人的 mppi 参数文件名为‘wheeltec_mec_mppi_demo.yaml’，全向机器人和差速机器人的 mppi 文件对应‘wheeltec_diff_mppi_demo.yaml’，阿克曼机器人的 mppi 参数文件名为‘wheeltec_akm_mppi_demo’。以下以迷你麦轮机器人为例，修改需要调用的导航参数文件名，改为‘wheeltec_mec_mppi_demo.yaml’，如图所示：

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/launch/wheeltec_nav2.launch.py
```

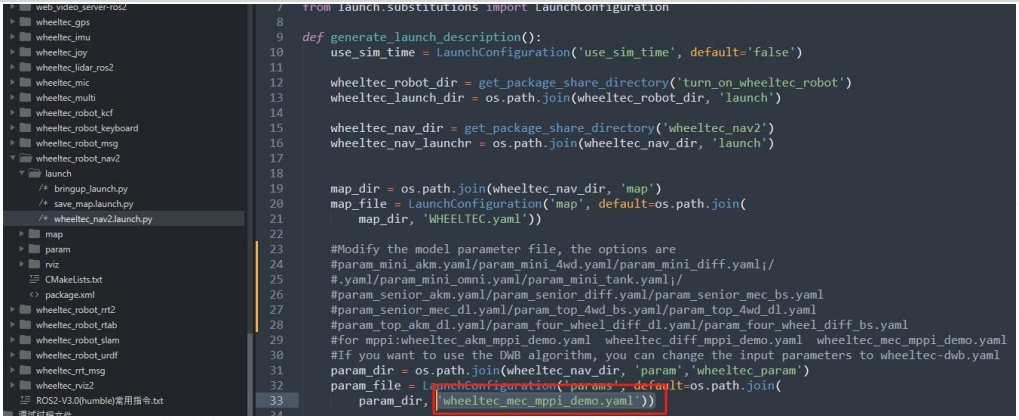


图 3-4-3 wheeltec_nav2.launch.py 文件内容

修改 wheeltec_mec_mppi_demo.yaml 文件中的车型参数，文件路径为

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/param/wheeltec_param/wheeltec_mec_mppi_demo.yaml
```

修改图示位置，将 local_costmap 节点 footprint 和 global_costmap 节点的 local_costmap 节点 footprint 改为迷你麦轮机器人的外形参数，参考 param_mini_mec.yaml 参数文件中的[footprint]值。

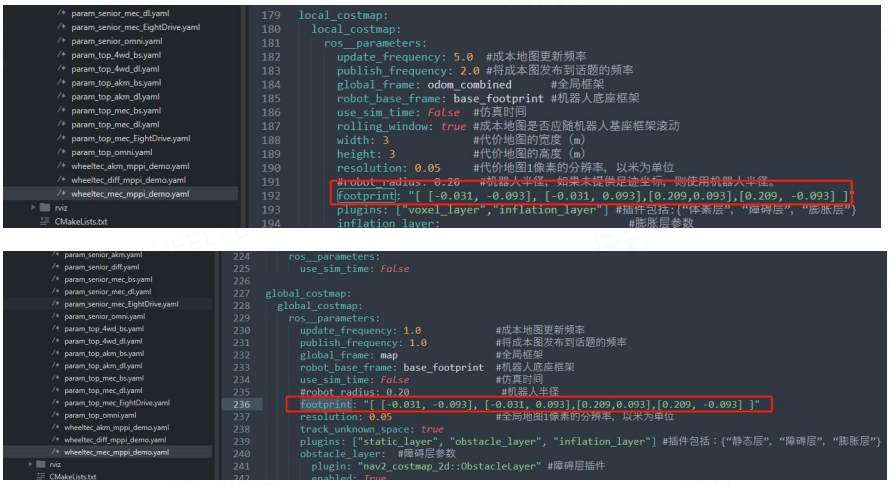


图 3-4-4 参数文件内容

保存文件后退出，打开终端进入 wheeltec_ros2 工作空间中，编译

wheeltec_nav2 功能包：

```
colcon build --packages-select wheeltec_nav2
```

② 切换为 RPP 局部路径规划算法

注意：DWB 算法是最匹配差速机器人的路径规划算法，RPP 算法精度要求较高，切换为 RPP 算法在实车上效果可能不如 DWB 规划算法，实际上用于导航还是建议使用 DWB 算法。

以下演示差速机器人切换为 RPP 局部路径规划算法的具体步骤。以迷你麦轮机器人为例，需要先确认修改需要调用的导航参数文件，如图所示，导航启动的参数文件为 param_mini_mec.yaml。

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/launch/wheeltec_nav2.launch.py
```

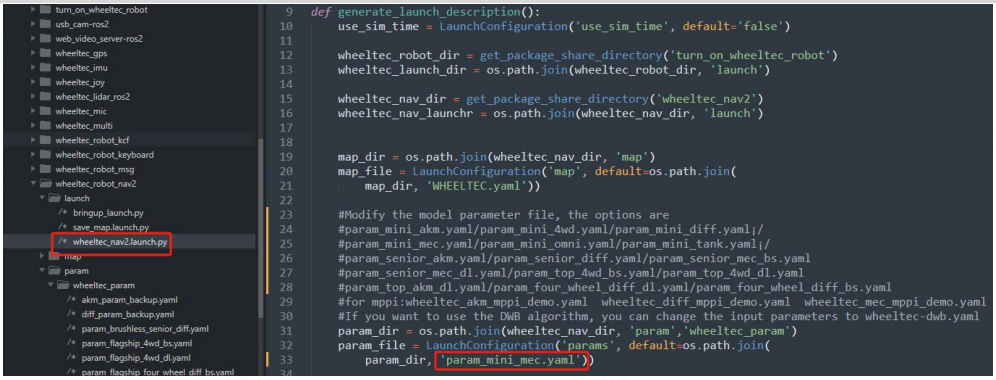


图 3-4-5 wheeltec_nav2.launch.py 文件内容

修改param_mini_mec.yaml文件中的controller_server控制器服务器部分参数，文件路径为：

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/param/wheeltec_param/param_mini_mec.yaml
```

修改 FollowPath 的插件内容，可以参考 param_mini_akm.yaml 参数文件，如下所示：

图 3-4-6 参数文件内容

同时，因为 Smac_planner 规划器(全局路径算法)在生成的路径更平滑、更易于跟踪，所以规划服务器参数节点也要修改，如下所示。

```
planner_server: #规划器服务器参数
  ros__parameters:
    planner_plugins: ["GridBased"]
    use_sim_time: False
    expected_planner_frequency: 20.0     #预期的规划器频率。如果当前频率小于预期频率，则显示警告消息。
    GridBased:
      plugin: "nav2_smac_planner/SmacPlannerHybrid" #Smac Hybrid-A* 规划器
      tolerance: 0.5                      # 如果无法达到精确的姿势，则规划的容差(单位m)
      downsample_costmap: false           # 是否将代价图下采样至另一个分辨率以进行搜索
      downsampling_factor: 1              # 对代价图进行下采样的乘数因子
      allow_unknown: false                # 允许在未知空间搜索
      max_iterations: 1000000             # 失败前搜索的最大迭代次数（以防无法到达），设置为 -1 以禁用
      max_on_approach_iterations: 1000    # 在公差范围内尝试达到目标的最大迭代次数，仅限 2D
      max_planning_time: 3.5              # 规划器进行规划、平滑和上采样的最大时间（以秒为单位）。
      motion_model_for_search: "REEDS_SHEPP"    # 2D Moore, Von Neumann; Hybrid Dubin, Redds-Shepp; State Lattice set internally
      cost_travel_multiplier: 2.0         #应用于搜索以避开高成本区域的成本乘数。
      # For 2D: Cost multiplier to apply to search to steer away from high cost areas. Larger values will place in the center of ais
      angle_quantization_bins: 64         #用于 SE2 搜索的角度箱数量。这可以是任何偶数，但好的基线是 64 或 72（以 5 度为增量）
      # For Hybrid/Lattice nodes: Number of angle bins for search, must be 1 for 2D node (no angle search)
      analytic_expansion_ratio: 3.5       #规划器将尝试以与该值和最小启发式方法成比例的频率完成分析扩展。
      # For Hybrid/Lattice nodes: The ratio to attempt analytic expansions during search for final approach.
      minimum_turning_radius: 0.770        #最小转弯半径
      # For Hybrid/Lattice nodes: minimum turning radius in m of path / vehicle
      reverse_penalty: 2.1                #如果反向搜索，则对 SE2 节点应用启发式惩罚。
      # For Reeds-Shepp model: penalty to apply if motion is reversing, must be => 1
      change_penalty: 0.15                #如果在搜索中改变方向（例如从左到右），则对 SE2 节点应用启发式惩罚。
      # For Hybrid nodes: penalty to apply if motion is changing directions, must be >= 0
      non_straight_penalty: 1.50          #如果在非直线方向上搜索，则对 SE2 节点应用启发式惩罚。
      # For Hybrid nodes: penalty to apply if motion is non-straight, must be => 1
      cost_penalty: 1.7                   #应用于 SE2 节点的姿势成本的启发式惩罚。
      # For Hybrid nodes: penalty to apply to higher cost areas when adding into the obstacle map dynamic programming distance expan
      lookup_table_size: 20.0              #距离窗口的缓存大小
      # For Hybrid nodes: Size of the dubin/reeds-sheep distance window to cache, in meters.
      cache_obstacle_heuristic: True       #缓存障碍启发式，默认False
      # For Hybrid nodes: Cache the obstacle map dynamic programming distance expansion heuristic between subsiquent replannings of
      smoother:
        max_iterations: 1000 #平滑器平滑路径的最大迭代次数，以限制潜在的计算。
        w_smooth: 0.3 #应用于平滑数据点的平滑权重
        w_data: 0.2 #应用权重平滑以保留原始数据信息
        tolerance: 1.0e-10 #终止平滑会话的参数容差变化量
```

图 3-4-7Smac_planne 规划器参数内容

然后修改机器人外形参数。修改图示位置，将 local_costmap 节点 footprint 和 global_costmap 节点的 local_costmap 节点 footprint 改为迷你麦轮机器人的外形参数，参考 param_mini_mec.yaml 参数文件中的[footprint]值。

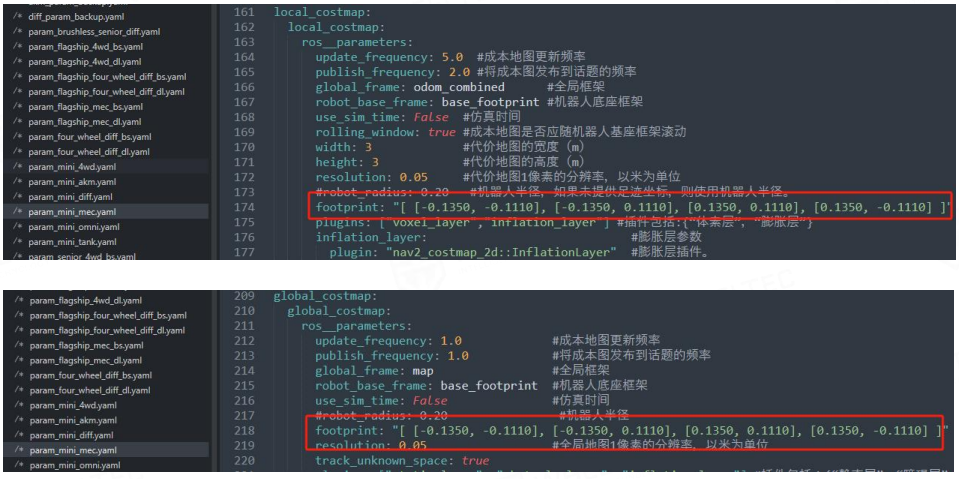


图 3-4-8 机器人外形参数内容

保存文件后退出，打开终端进入 wheeltec_ros2 工作空间中，编译 wheeltec_nav2 功能包：

```
colcon build --packages-select wheeltec_nav2
```

③ 切换为 DWB 局部路径规划算法

注意：RPP/MPPI 算法是最匹配阿克曼机器人的路径规划算法，DWB 规划器生成的路径点较少，且不平滑，切换为 DWB 算法在实车上效果可能不如 RPP/MPPI 规划算法，实际上用于导航还是建议使用 RPP 或 MPPI 算法。以下演示阿克曼机器人切换为 DWB 局部路径规划算法的具体步骤。以迷你阿克曼机器人为例，需要先确认修改需要调用的导航参数文件，如图所示，导航启动的参数文件为 param_mini_akm.yaml。

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/launch/wheeltec_nav2.launch.py
```

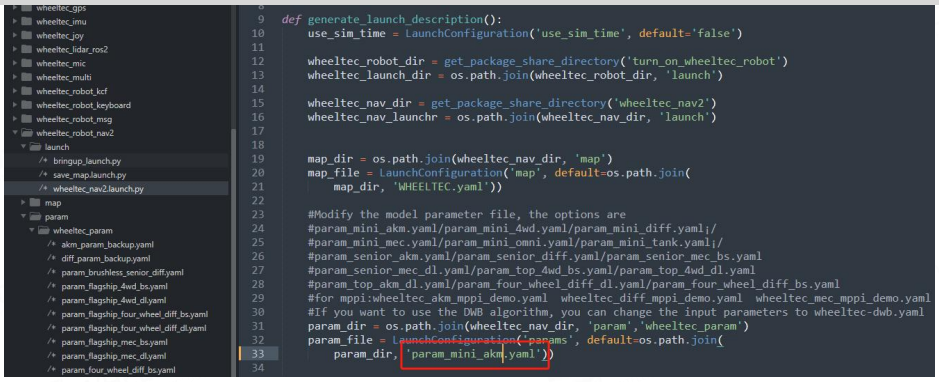


图 3-4-9wheeltec_nav2.launch.py 文件内容

修改 param_mini_akm.yaml 文件中的 controller_server 控制器服务器部分参数，文件路径为：

```
~/wheeltec_ros2/src/wheeltec_robot_nav2/param/wheeltec_param/param_mini_akm.yaml
```

修改 FollowPath 的插件内容，可以参考 param_mini_mec.yaml 参数文件，如下所示：

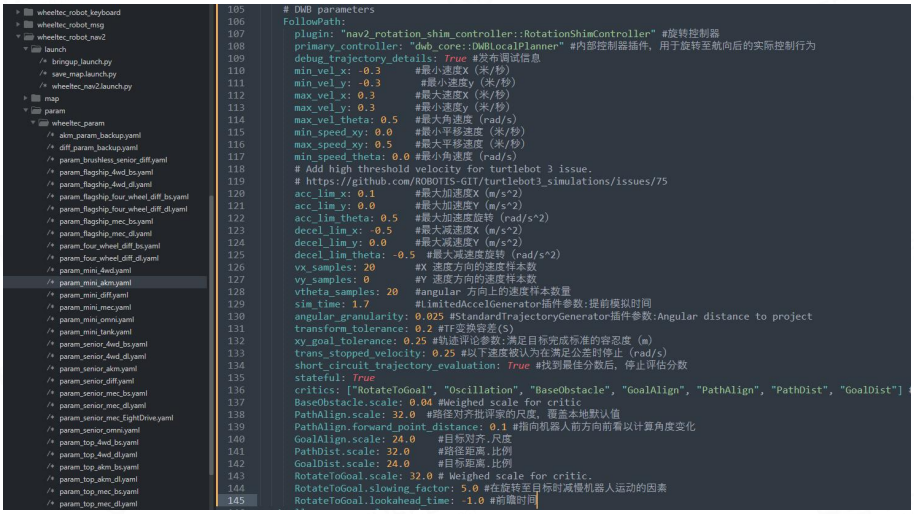


图 3-4-10 参数文件内容

因为 NavfnPlanner 是基于经典的栅格地图导航算法，它使用 Dijkstra 或 A* 算法在离散的栅格地图上寻找最短路径，所以使用 DWB 局部规划器算法时，全局规划器需要改为 NavfnPlanner，如下所示。

图 3-4-11 修改全局规划器

然后修改机器人外形参数。修改图示位置，将 local_costmap 节点 footprint 和 global_costmap 节点的 local_costmap 节点 footprint 改为迷你麦轮机器人的外形参数，参考 param_mini_akm.yaml 参数文件中的[footprint]值。

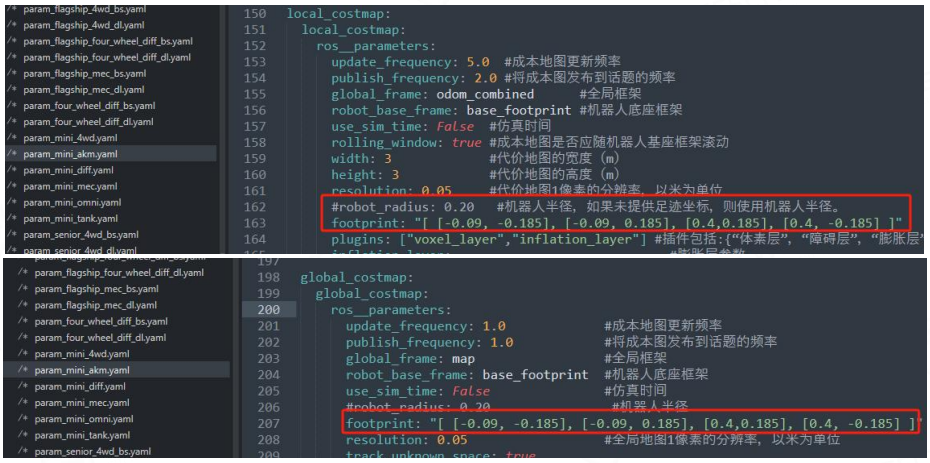


图 3-4-12 修改机器人外形参数内容

保存文件后退出，打开终端进入 wheeltec_ros2 工作空间中，编译 wheeltec_nav2 功能包：

```
colcon build --packages-select wheeltec_nav2
```