7、导航避障

amcl：http://wiki.ros.org/amcl

navigation：http://wiki.ros.org/navigation/

navigation/Tutorials：http://wiki.ros.org/navigation/Tutorials/RobotSetup

costmap_2d：http://wiki.ros.org/costmap_2d

nav_core：http://wiki.ros.org/nav_core

global_planner：http://wiki.ros.org/global_planner

dwa_local_planner：http://wiki.ros.org/dwa_local_planner

teb_local_planner：http://wiki.ros.org/teb_local_planner

move_base：http://wiki.ros.org/move_base

depthimage_to_laserscan：http://wiki.ros.org/depthimage_to_laserscan

功能包：~/rplidar_ws/src/transbot_nav

7.1、操作使用

注意：遥控手柄的【R2】具备取消目标点的功能。

7.1.1、启动

启动驱动，根据需求启动（注意：纯深度建图导航效果不佳，不推荐使用）。

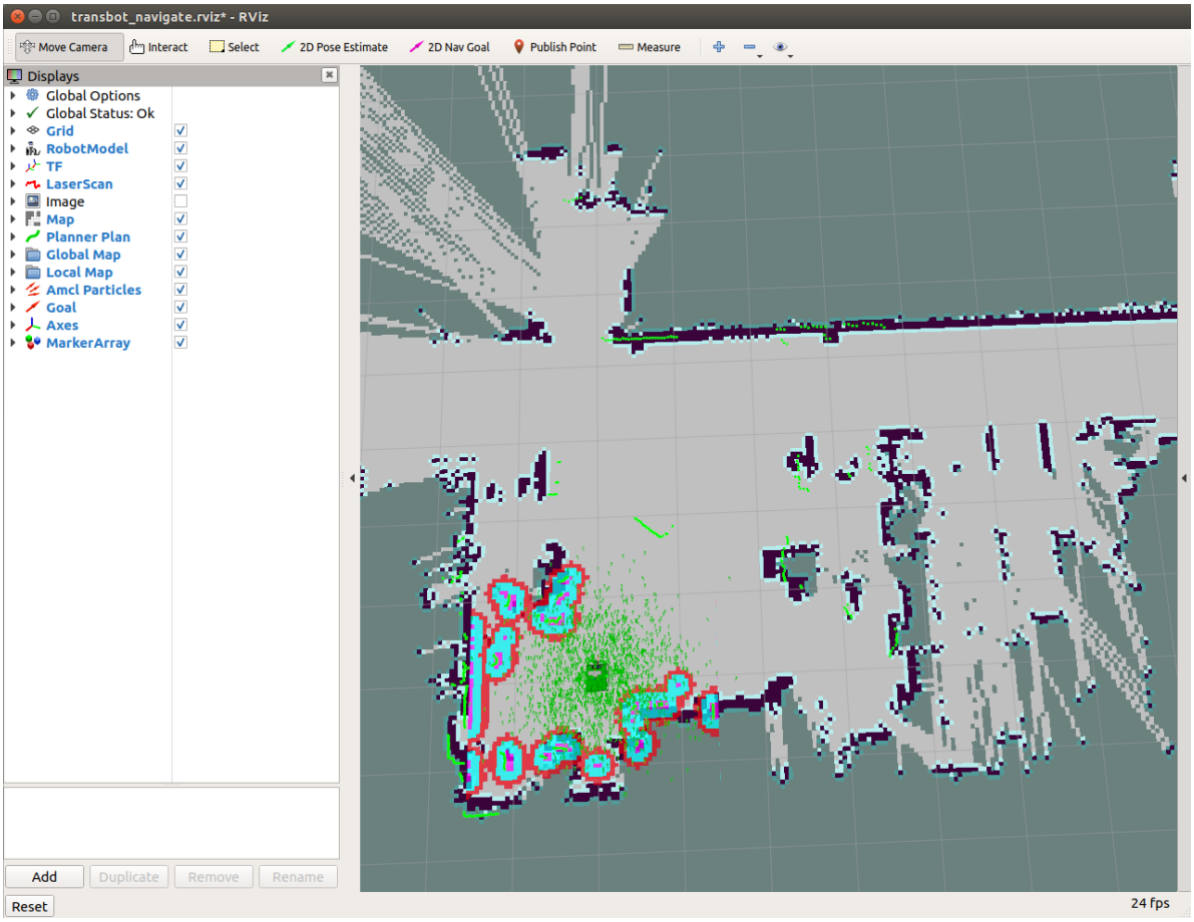
```
roslaunch transbot_nav usbcam_bringup.launch lidar_type:=a1   # mono + laser +
Transbot
roslaunch transbot_nav astra_bringup.launch                   # Astra + Transbot
roslaunch transbot_nav laser_bringup.launch lidar_type:=a1    # laser + Transbot
roslaunch transbot_nav transbot_bringup.launch lidar_type:=a1 # Astra + laser +
Transbot
```

lidar_type参数：使用激光雷达的型号：[a1,a2,a3,s1,s2]。

启动导航避障功能，可根据需求设置参数，也可修改launch文件。

```
roslaunch transbot_nav transbot_navigation.launch open_rviz:=true map:=house
```

- open_rviz参数：是否打开rviz。
- map：地图名，要加载的地图。

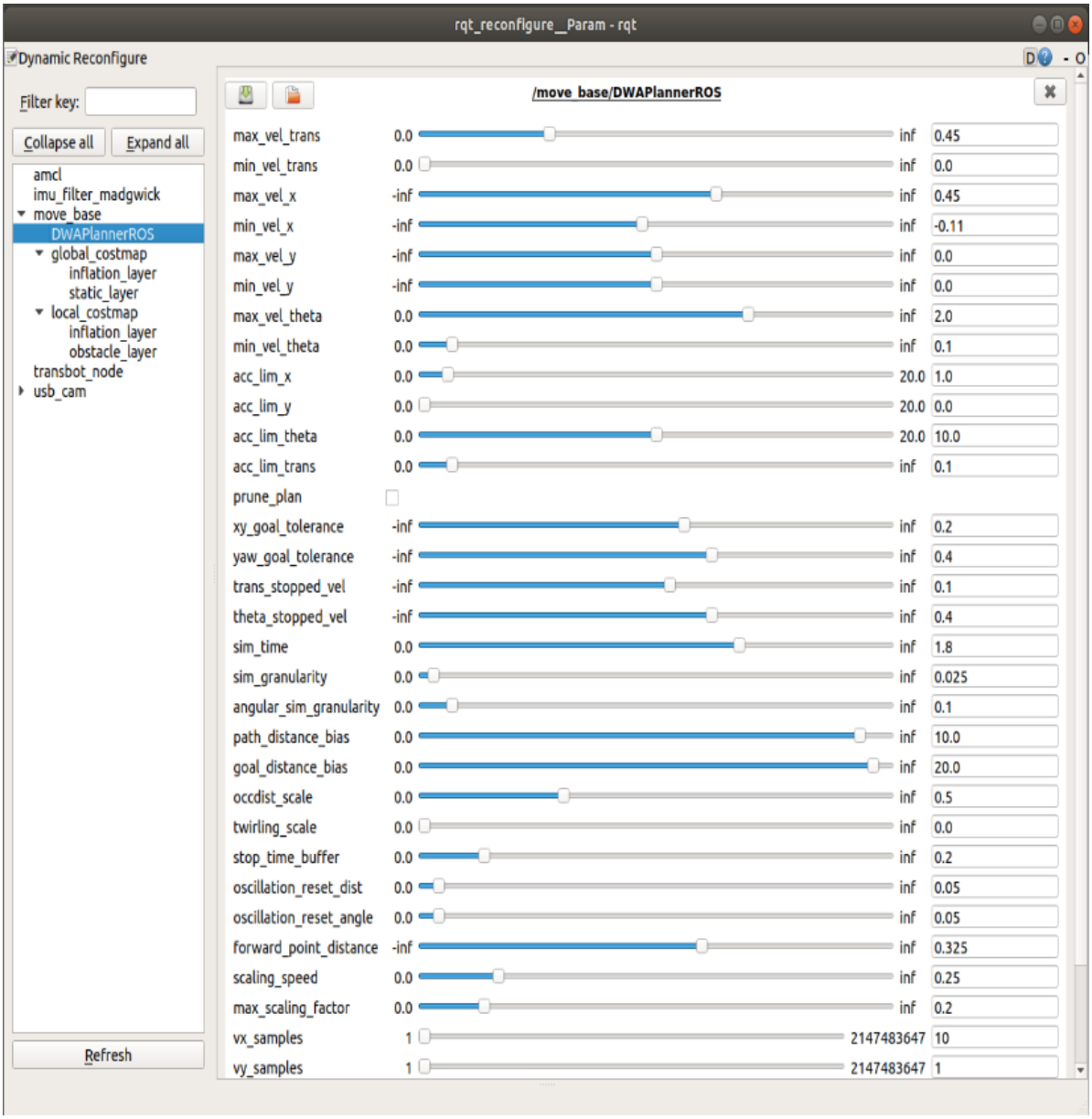


7.1.2、使用

- 将机器人放在原点处，如果雷达扫描边线和地图没有重合，需要使用【rviz】工具的【2D Pose Estimate】设置初始位姿，如果机器人找不到在地图中的位姿时，也需要设置初始位姿。
- 点击【rviz】工具的【2D Nav Goal】，然后再地图上在没有障碍物的地方选择目标点，松开鼠标即开始导航，只能选择一个目标点，到达即停止。
- 多点导航：点击【rviz】工具的【Publish Point】，然后再地图上在没有障碍物的地方选择目标点，松开鼠标即开始导航，可再次点击【Publish Point】，然后选点，机器人就会点与点之间巡航。

7.1.3、动态参数调整

```
rosrun rqt_reconfigure rqt_reconfigure
```



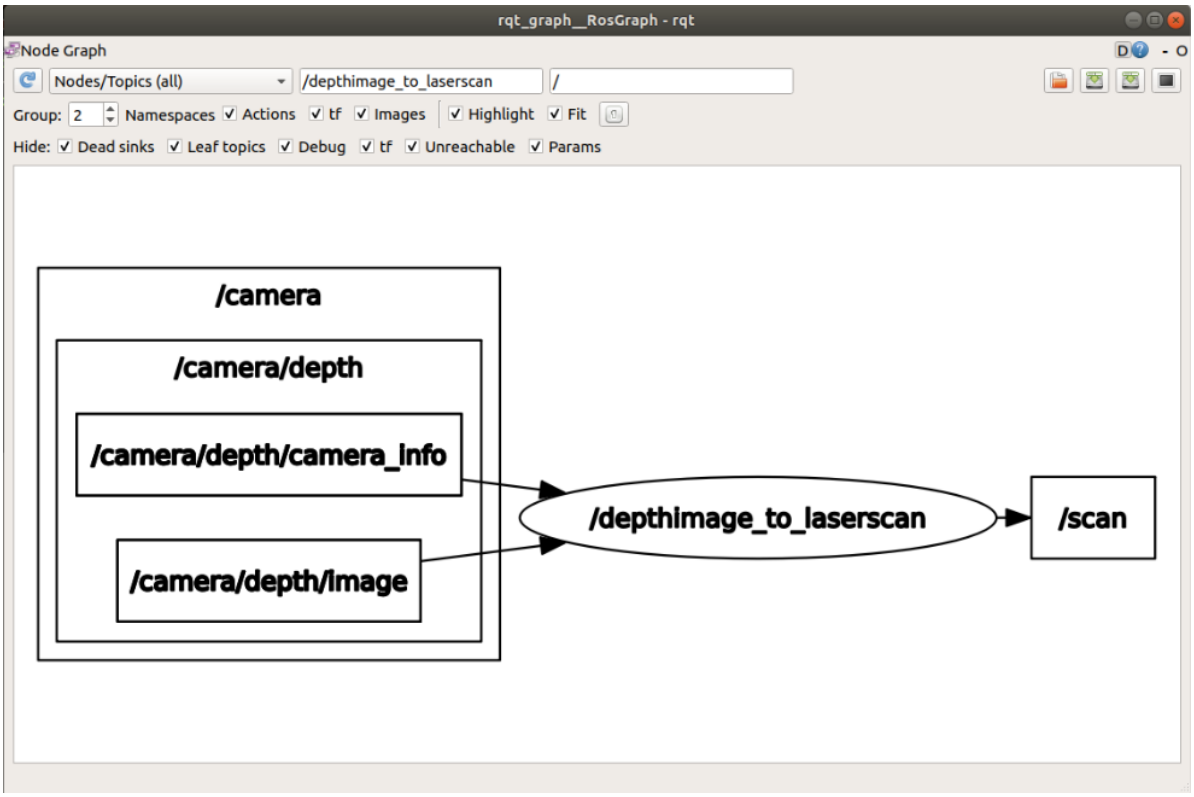
可以查看一下节点和tf树的关系

```
rqt_graph
rosrun rqt_tf_tree rqt_tf_tree
```

7.1.4、纯视觉2D导航

启动驱动参考【7.1.1】# Astra + Transbot；启动导航命令和【7.1.1】一样。主要使用功能包depthimage_to_laserscan，将深度图像转化为激光雷达数据，其导航功能和激光雷达相同。注意：深度相机的扫描范围不是360°。

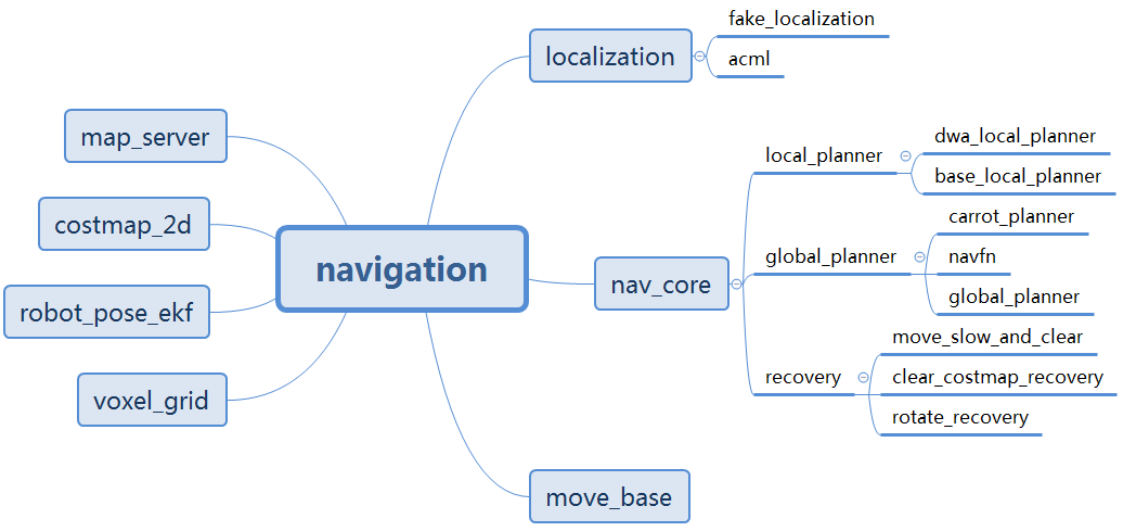
```
rqt_graph
```



7.2、navigation

7.2.1、简介

navigation是ROS的二维导航避障功能包，简单来说，就是根据输入的里程计等传感器的信息流和机器人的全局位置，通过导航算法，计算得出安全可靠的机器人速度控制指令。

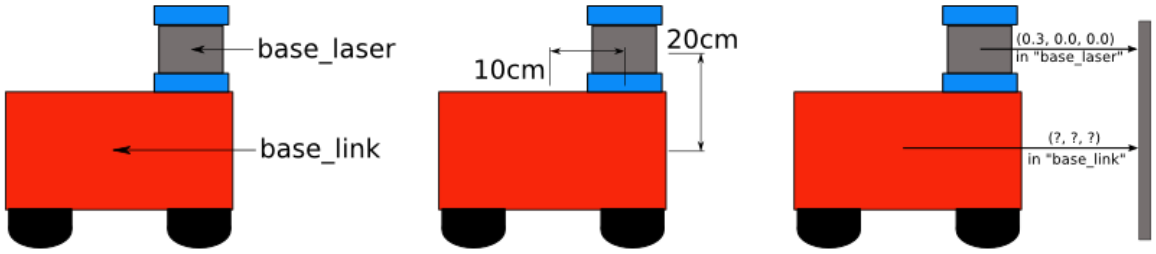


navigation主要节点及配置

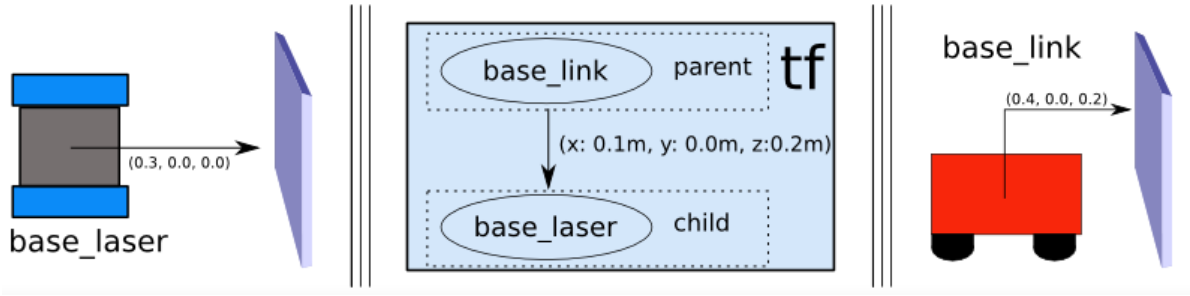
- move_base：导航避障控制的最终执行机构，move_base订阅导航目标move_base_simple/goal，并实时发布运动控制信号cmd_vel，move_base中的各种导航算法模块都是以插件的形式进行调用的。
- global_planner：用于全局路径规划。
- local_planner：用于局部路径规划。
- global_costmap：全局代价地图用于描述全局环境信息。
- local_costmap：局部代价地图用于描述局部环境信息。
- recovery_behaviors：恢复策略用于机器人碰到障碍后自动进行逃离恢复。
- amcl：利用粒子滤波算法实现机器人的全局定位，为机器人导航提供全局位置信息。
- map_server：通过调用SLAM建图得到的地图为导航提供环境地图信息。
- costmap_2d：可以生产代价地图，以及提供各种相关的函数。
- robot_pose_ekf：扩展卡尔曼滤波器，输入是里程计、IMU、VO中的任意两个或者三个，输出是一个融合之后的pose。
- fake_localization：一般是仿真用的。
- nav_core：这里面只有三个文件，对应的是全局路径规划、局部路径规划、recovery_action的通用接口定义，具体功能实现则是在各个对应的规划器功能包里。
- 还需要提供机器人模型相关的tf信息、里程计odom信息、激光雷达信息scan。

7.2.2、设置tf

导航功能要求机器人使用tf发布有关坐标系之间关系的信息。例如：激光雷达



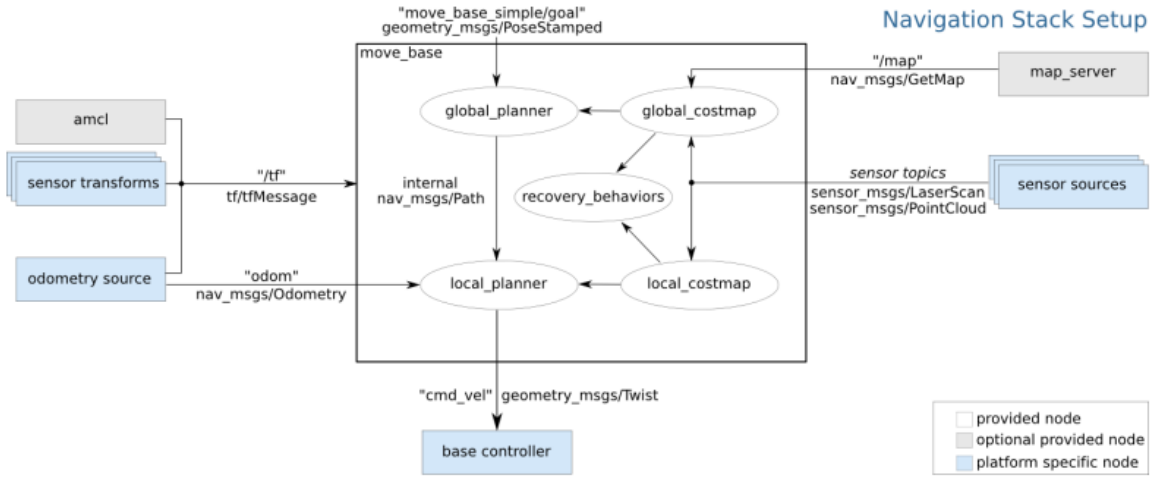
假设我们知道激光雷达安装在移动底座中心点上方 10 厘米和 20 厘米处。这为我们提供了将“base_link”框架与“base_laser”框架相关联的平移偏移量。具体来说，我们知道要从“base_link”坐标系获取数据到“base_laser”坐标系，我们必须应用 (x: 0.1m, y: 0.0m, z: 0.2m) 的平移，并从“ base_laser”框架到“base_link”框架，我们必须应用相反的平移（x：-0.1m，y：0.0m，z：-0.20m）。



7.3、move_base

7.3.1、介绍

move_base提供了ROS导航的配置、运行、交互接口。



实现机器人导航功能必须以特定方式配置，如上图：

- 白色组件是已经实现的必需组件，
- 灰色组件是已经实现的可选组件，
- 蓝色组件必须为每个机器人平台创建。

7.3.2、move_base通讯机制

1）Action

move_base节点提供了SimpleActionServer的一个实现，它接收包含geometry_msgs/PoseStamped消息的目标。您可以通过ROS直接与move_base节点通信，但是如果您关心跟踪目标的状态，建议使用SimpleActionClient将目标发送到move_base。（更多信息请参阅actionlib documentation）

名称	类型	说明
move_base/goal	move_base_msgs/MoveBaseActionGoal	move_base订阅将要到达的目标点。
move_base/cancel	actionlib_msgs/GoalID	move_base订阅取消特定目标的请求。
move_base/feedback	move_base_msgs/MoveBaseActionFeedback	发布包含底盘的当前位置。
move_base/status	actionlib_msgs/GoalStatusArray	发布移动到目标点过程的状态信息。
move_base/result	move_base_msgs/MoveBaseActionResult	发布移动的最终结果。

2）topic

名称	类型	说明
move_base_simple/goal	geometry_msgs/PoseStamped	为不关注跟踪目标的执行状态提供一个非action的接口。move_base订阅将要到达的目标点。
cmd_vel	geometry_msgs/Twist	发布小车移动速度。

3）servies

名称	类型	说明
make_plan	nav_msgs/GetPlan	允许外部用户从move_base请求给定姿势的计划，而不会导致move_base执行该计划。
clear_unknown_space	std_srvs/Empty	允许外部用户通知move_base清除机器人周围区域的未知空间。当move_base的costmaps停止很长一段时间，然后在环境中的新位置重新启动时，这一点非常有用。
clear_costmaps	std_srvs/Empty	允许外部用户告知move_base清除move_base使用的代价地图中的障碍。这可能会导致机器人撞到东西，使用时应谨慎

4）参数配置

move_base_params.yaml

```
# 设置move_base的全局路径规划器的插件名称
#base_global_planner: "navfn/NavfnROS"
base_global_planner: "global_planner/GlobalPlanner"
#base_global_planner: "carrot_planner/CarrotPlanner"

# 设置move_base的局部路径规划器的插件名称
#base_local_planner: "teb_local_planner/TebLocalPlannerROS" # 实现DWA（动态窗口法）局部规划算法
base_local_planner: "dwa_local_planner/DWAPlannerROS"       # 实现一个在线优化的本地轨迹规划器
# 恢复行为。
recovery_behaviors:
  - name: 'conservative_reset'
    type: 'clear_costmap_recovery/ClearCostmapRecovery'
  #- name: 'aggressive_reset'
  #  type: 'clear_costmap_recovery/ClearCostmapRecovery'
  #- name: 'super_reset'
  #  type: 'clear_costmap_recovery/ClearCostmapRecovery'
  - name: 'clearing_rotation'
    type: 'rotate_recovery/RotateRecovery'
    #- name: 'move_slow_and_clear'
    #type: 'move_slow_and_clear/MoveSlowAndClear'

# 向机器人底盘cmd_vel发送命令的频率
controller_frequency: 10.0 #default:20.0
# 空间清理操作执行前，路径规划器等待有效控制命令的时间
planner_patience: 5.0 #default:5.0
# 空间清理操作执行前，控制器等待有效控制命令的时间
controller_patience: 15.0 #default:15.0
# 仅当默认恢复行为用于 move_base 时才使用此参数。
conservative_reset_dist: 3.0 #3.0,
# 是否启用move_base恢复行为以尝试清除空间。
recovery_behavior_enabled: true
# 机器人是否采用原地旋转的运动方式清理空间,此参数仅在使用默认恢复行为时使用。
clearing_rotation_allowed: true
# 当move_base进入inactive状态时，是否停用节点的costmap
shutdown_costmaps: false #false
```

```
# 执行恢复操作之前允许震荡的时间，0代表永不超时
oscillation_timeout: 10.0 #0.0
# 机器人需要移动该距离才可当做没有震荡。移动完毕后重置定时器参数
oscillation_distance: 0.2 #0.5
# 全局路径规划器循环速率。
planner_frequency: 5.0 #0.0
#在执行恢复行为之前允许计划重试的次数。值-1.0对应于无限次重试。
max_planning_retries: -1.0
```

参数解析

- base_global_planner：全局路径规划器。
- base_local_planner：局部路径规划器。
- recovery_behaviors：恢复行为。
- controller_frequency：运行控制回路并向机器人发送速度命令的速率（单位：Hz）。
- planner_patience：在执行空间清理操作之前，规划器尝试查找有效计划的等待时间（单位：s）。
- controller_patience：在执行空间清理操作之前，控制器在没有收到有效控制的情况下将等待多长时间（单位：s）。
- conservative_reset_dist：距离机器人的距离，当试图清除地图中的空间时，超过该距离，障碍物将从costmap中清除。注意，此参数仅在默认恢复行为用于move_base时使用。
- recovery_behavior_enabled：是否启用move_base恢复行为以尝试清除空间。
- clearing_rotation_allowed：确定在尝试清理空间时，机器人是否将尝试原地旋转。注意：此参数仅在使用默认恢复行为时使用，这意味着用户未将恢复行为参数设置为任何自定义。
- shutdown_costmaps：是否在move_base处于非活动状态时关闭代价地图。
- oscillation_timeout：在执行恢复行为之前允许振荡的时间（单位：s）。值0.0对应于无限超时。
- oscillation_distance：机器人必须移动多远才能被视为不摆动（单位：m）。
- planner_frequency：运行全局规划循环的速率（单位：Hz）。如果频率设置为0.0，则仅当收到新目标或本地规划器报告其路径被阻止时，全局规划器才会运行。
- max_planning_retries：在执行恢复行为之前允许计划重试的次数。值-1.0对应于无限次重试。

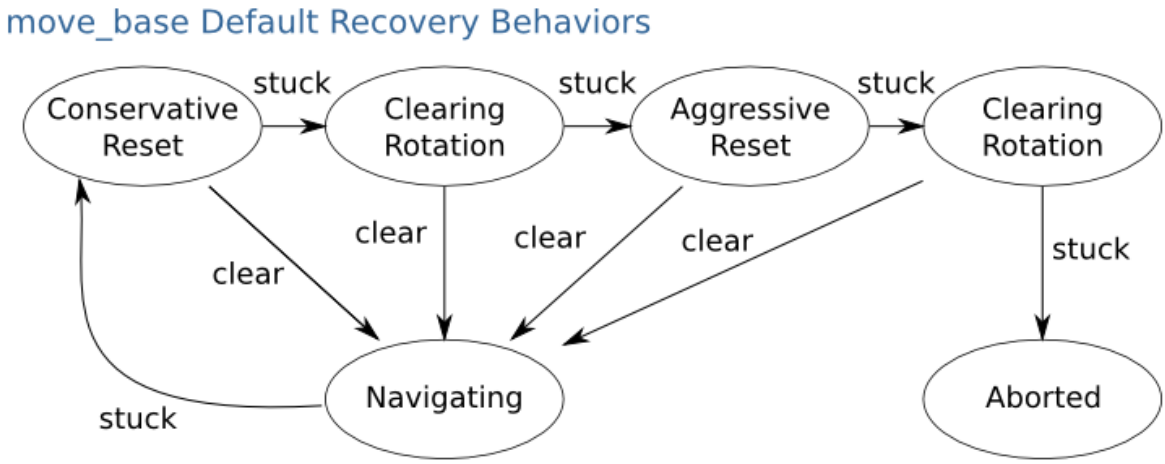
7.3.3、Recovery Behavior

1）简介

当①全局规划失败、②机器人震荡、③局部规划失败时会进入到恢复行为。可以使用recovery_behaviour参数配置这些恢复行为，并使用recovery_behavior_enabled参数禁用这些恢复行为。

期望的机器人行为

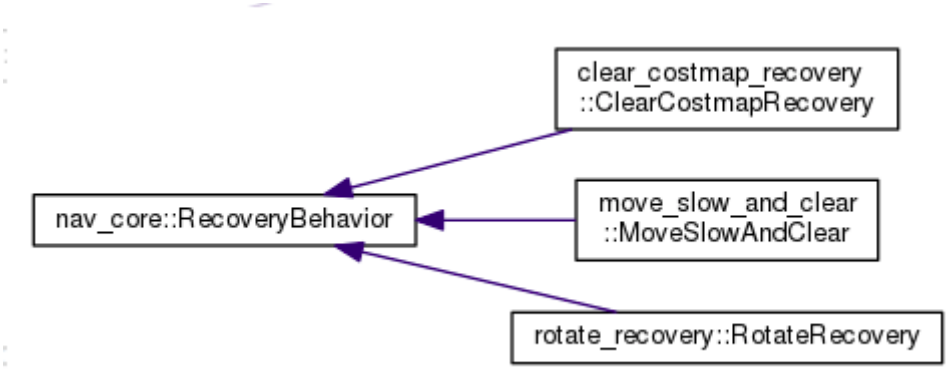
首先，用户指定区域之外的障碍物将从机器人的地图上清除。接下来，如果可能的话，机器人将执行原地旋转以清理空间。如果这也失败了，机器人将更积极地清除地图，清除矩形区域之外的所有障碍物，在该区域内机器人可以原地旋转。随后将进行另一次就地旋转。如果所有这些都失败，机器人会认为它的目标是不可行的，并通知用户它已经中止。



- conservative reset：保守恢复。
- clearing rotation：旋转清除。
- aggressive reset：积极恢复。
- aborted：中止。

2）相关功能包

在导航功能包集合中，有3个包与恢复机制有关。分别为：clear_costmap_recovery、move_slow_and_clear、rotate_recovery。这3个包中分别定义了3个类，都继承了nav_core中的接口规范。



- move_slow_and_clear：是一种简单的恢复行为，它清除代价图中的信息，然后限制机器人的速度。注意，这种恢复行为并不是真正安全的，机器人可能会击中物体，它只会以用户指定的速度发生。

参数	类型	默认值	解析
clearing_distance	double	0.5	在机器人清除距离内的障碍物会被清除（单位：m）。
limited_trans_speed	double	0.25	执行此恢复行为时，机器人的平移速度将受到限制（单位：m/s）。
limited_rot_speed	double	0.25	执行此恢复行为时，机器人的转速将受到限制（单位：rad/s）。
limited_distance	double	0.3	解除速度限制之前，机器人必须移动的距离（单位：m）。
planner_namespace	string	"DWAPlannerROS"	要重新配置参数的规划器的名称。

- rotate_recovery：旋转恢复，通过机器人360度旋转来清除空间。

参数	类型	默认值	解析
sim_granularity	double	0.017	检查原地旋转是否安全时，检查障碍物之间的距离默认为1度（单位：rad）。
frequency	double	20.0	向移动机器人发送速度命令的频率（单位：HZ）。
TrajectoryPlannerROS/yaw_goal_tolerance	double	0.05	实现其目标时控制器在偏航/旋转中的弧度公差。
TrajectoryPlannerROS/acc_lim_th	double	3.2	机器人的旋转加速度极限（单位：rad/s^2）。
TrajectoryPlannerROS/max_rotational_vel	double	1.0	底座允许的最大旋转速度（单位：rad/s）。
TrajectoryPlannerROS/min_in_place_rotational_vel	double	0.4	执行在位旋转时底座允许的最小旋转速度（单位：rad/s）。

注意：TrajectoryPlannerROS参数在使用base_local_planner::TrajectoryPlannerROS规划器时才被设置；一般无需设置。

- clear_costmap_recovery：一种恢复行为，将move_base使用的代价地图还原为用户指定范围之外的静态地图。

参数	类型	默认值	解析
clearing_distance	double	0.5	以机器人为中心的长度，当障碍物恢复为静态地图时，障碍物将从代价地图中移除。

7.4、costmap_params

导航功能使用两个代价地图来存储障碍物的信息。一个代价地图用于全局规划，这意味着在整个环境中创建全局规划路线，另一个用于局部规划和避障。这个两个代价地图有一些共同的配置，也有一些单独的配置。因此，代价地图配置有以下三个部分：通用配置、全局配置和局部配置。

7.4.1、costmap_common

代价地图公有参数配置costmap_common_params.yaml

```
obstacle_range: 2.5
raytrace_range: 3.0
footprint: [[x0, y0], [x1, y1], ... [xn, yn]]
# robot_radius: ir_of_robot
inflation_radius: 0.55

observation_sources: laser_scan_sensor point_cloud_sensor

laser_scan_sensor: {sensor_frame: frame_name, data_type: LaserScan, topic:
topic_name, marking: true, clearing: true}

point_cloud_sensor: {sensor_frame: frame_name, data_type: PointCloud, topic:
topic_name, marking: true, clearing: true}
```

参数解析

- obstacle_range：默认值为2.5米，意味着机器人将只更新2.5米范围内障碍物的信息。
- raytrace_range：默认值为3.0米，意味着机器人将尝试清理其前方3.0米以外的空间。
- footprint：设置机器人的占用面积，按机器人坐标设置填入“footprint” 指定的footprint时，机器人的中心被认为是在(0.0,0.0)，顺时针和逆时针设置都可以。
- robot_radius：是机器人的占用面积为圆形，直接设置半径即可。与footprint不可共用。
- inflation_radius：设置代价地图膨胀半径。默认0.55。意味着机器人会将距离障碍物0.55米的范围内视为障碍物。
- observation_sources：定义了一个传感器列表，将信息传递到由空格分隔的代价地图。
- laser_scan_sensor：定义了每个传感器。
    - sensor_frame：传感器坐标系的名称。
    - data_type：参数设置为LaserScan或PointCloud，这取决于话题使用的消息。
    - topic：传感器发布数据的话题名。
    - marking：是否用于将障碍物信息添加到代价地图。
    - clearing：是否用于将障碍物信息在代价地图中清除。

7.4.2、global_costmap

全局代价地图参数配置global_costmap_params.yaml

```
global_costmap:
  global_frame: /map
  robot_base_frame: base_link
  update_frequency: 5.0
  static_map: true
```

- global_frame：全局代价地图在那个坐标系下运行。
- robot_base_frame：全局代价地图参考的机器人基座坐标系。
- update_frequency：全局代价地图循环更新的频率，单位为Hz。
- static_map：全局代价地图是否应根据map_server提供的地图进行初始化。如果未使用现有地图或地图服务器，请将static_map参数设置为false。

7.4.3、local_costmap

局部代价地图参数配置local_costmap_params.yaml

```
local_costmap:
  global_frame: odom
  robot_base_frame: base_link
  update_frequency: 5.0
  publish_frequency: 2.0
  static_map: false
  rolling_window: true
  width: 6.0
  height: 6.0
  resolution: 0.05
```

参数解析

- global_frame：局部代价地图在那个坐标系下运行。
- robot_base_frame：局部代价地图参考的机器人基座坐标系。
- update_frequency：局部代价地图循环更新的频率，单位为Hz。
- publish_frequency：局部代价地图发布可视化信息的速率，单位为Hz。
- static_map：局部代价地图是否应根据map_server提供的地图进行初始化。如果未使用现有地图或地图服务器，请将static_map参数设置为false。
- rolling_window：（滚动窗口）参数设置为true意味着当机器人在移动时，局部代价地图将保持以机器人的中心。
- width：局部代价地图宽度（米）。
- height：局部代价地图高度（米）。
- resolution：局部代价地图分辨率（米/单元）。

7.4.4、costmap_2D

1）简介

costmap_2d这个包提供了一种2D代价地图的实现方案，该方案利用输入的传感器数据，构建数据2D或者3D代价地图（取决于是否使用基于voxel的实现），并根据占用网格和用户定义的膨胀半径计算2D代价地图的代价。

- 红色代表代价地图中的障碍物。
- 蓝色表示机器人内接半径膨胀的障碍物，
- 红色多边形表示机器人的足迹。
- 为了使机器人避免碰撞，机器人的外壳绝对不允许与红色单元格相交，机器人的中心点绝对不允许与蓝色单元格相交。

2）topic

名称	类型	说明
footprint	geometry_msgs/Polygon	机器人外壳规范。这将替换封装外形的先前参数规范。
costmap	nav_msgs/OccupancyGrid	代价地图
costmap_updates	map_msgs/OccupancyGridUpdate	代价地图的更新区域
voxel_grid	costmap_2d/VoxelGrid	体素网格

3）参数配置

如果您不提供plugins参数，那么初始化代码将假定您的配置是pre-Hydro，默认名称空间是静态层（static_layer）、障碍层（obstacle_layer）和膨胀层（inflation_layer）。

插件

- plugins：一般使用默认即可。

坐标系与tf参数

- global_frame：代价地图运行的全局坐标系。

- robot_base_frame：机器人base_link的坐标系名称。
- transform_tolerance：指定可容忍的转换（tf）数据延迟（单位：s）。

速率参数

- update_frequency：更新地图的频率（单位：Hz）。
- publish_frequency：发布显示信息的地图的频率（单位：Hz）。

地图管理参数

- rolling_window：是否使用代价地图的滚动窗口版本。如果static_map参数设置为true，则此参数必须设置为false。
- always_send_full_costmap：如果为true，则每次更新都会将完整的costmap发布到"/costmap"。如果为false，则仅在“/costmap_updates”主题上发布已更改的costmap部分。

静态图层

- width：地图的宽度（单位：m）。
- height：地图的高度（单位：m）。
- resolution：地图分辨率（单位：m/cell）。
- origin_x：全局框架中地图的x原点（单位：m）。
- origin_y：全局框架中地图的y原点（单位：m）。

tf变换

global_fram——>robot_base_frame

4）图层规格

- 静态层 static map layer：静态图层在代价地图中基本不变。

  订阅主题

  - map：代价地图将向map_server发出服务调用以获取此映射。

  参数

  - unknown_cost_value：从map server提供地图中读到这个值，其代价将被按unknown看待。值为零也会导致此参数未使用。
  - lethal_cost_threshold：从地图服务器读取地图时考虑致命的阈值。
  - map_topic：指定代价地图用来订阅静态地图的主题。
  - first_map_only：仅订阅地图话题上的第一条消息，忽略所有后续消息。
  - subscribe_to_updates：除map_topic外，还订阅map_topic + "_updates"。
  - track_unknown_space：如果为true，则地图消息中的未知值将直接转换到图层。否则，地图消息中的未知值将转换为层中的空闲空间。
  - use_maximum：只有当静态层不是底层时才重要。如果为true，则仅将最大值写入主成本图。
  - trinary_costmap：如果为true，则将所有地图消息值转换为NO_INFORMATION/FREE_SPACE/LETHAL_OBSTACLE（三个值）。如果为false，则可能出现完整的中间值范围。
- 障碍物层obstacle layer：障碍层跟踪传感器数据读取的障碍物。碰撞代价地图插件在二维中标记和光线跟踪障碍物，而VoxelCostmapPlugin 在三维中标记和光线跟踪障碍物。
- 膨胀层inflation layer：在致命障碍物周围添加新值（即膨胀障碍物），以使代价地图表示机器人的配置空间。
- 其他层：其他层可以通过pluginlib在代价地图中实现和使用。
  - Social Costmap Layer
  - Range Sensor Layer

5）obstacle layer

障碍物层和体素层以点云或激光扫描的形式包含来自传感器的信息。障碍层在二维中跟踪，而体素层在三维中跟踪。

代价地图会自动订阅传感器主题并相应地更新。每个传感器用于标记（将障碍物信息插入代价地图）、清除（从代价地图中删除障碍物信息）。每次观察数据，清除操作从传感器原点向外通过网格执行光线跟踪。在体素层中，将每列中的障碍物信息向下投影到二维地图中。

订阅话题

话题名	类型	解析
point_cloud_topic	sensor_msgs/PointCloud	将PointCloud信息更新到代价地图。
point_cloud2_topic	sensor_msgs/PointCloud2	将PointCloud2信息更新到代价地图
laser_scan_topic	sensor_msgs/LaserScan	将LaserScan信息更新到代价地图
map	nav_msgs/OccupancyGrid	代价地图具有从用户生成的静态地图初始化的选项

传感器管理参数

- observation_sources：观测源名称列表

  观测源中的每个源名称定义了一个命名空间，可以在其中设置参数：

  - <source_name>/topic：传感器数据所涉及的话题。
  - <source_name>/sensor_frame：传感器。可以是sensor_msgs/LaserScan、sensor_msgs/PointCloud和sensor_msgs/PointCloud2。
  - <source_name>/observation_persistence：保持每个传感器读数的时间（单位：s）。值为0.0将仅保留最近的读数。
  - <source_name>/expected_update_rate：传感器读数的频率（单位：s）。值为0.0将允许读数之间有无限的时间间隔。
  - <source_name>/data_type：与话题关联的数据类型，目前仅支持“PointCloud”、“PointCloud2”和“LaserScan”。
  - <source_name>/clearing：该观察值是否应该用来清除自由空间。
  - <source_name>/marking：该观察值是否应用于标记障碍物。
  - <source_name>/max_obstacle_height：被认为有效的传感器读数的最大高度（单位：m）。这通常设置为略高于机器人的高度。
  - <source_name>/min_obstacle_height：被认为有效的传感器读数的最小高度（单位：m）。这通常设置为地面高度，但可以根据传感器的噪声模型设置得更高或更低。
  - <source_name>/obstacle_range：使用传感器数据将障碍物插入成本图的最大范围（单位：m）。
  - <source_name>/raytrace_range：使用传感器数据从地图中光线追踪障碍物的最大范围（单位：m）。
  - <source_name>/inf_is_valid：允许在“激光扫描”观测信息中输入Inf值。Inf值转换为激光最大范围

全局滤波参数：这些参数适用于所有传感器。

- max_obstacle_height：要插入代价地图的任何障碍物的最大高度（单位：m）。此参数应设置为略高于机器人的高度。
- obstacle_range：将障碍物插入代价地图时与机器人的默认最大距离（单位：m）。这可能会在每个传感器的基础上过度使用。

- raytrace_range：使用传感器数据从地图中光线追踪障碍物的默认范围（单位：m）。这可能会在每个传感器的基础上过度使用。

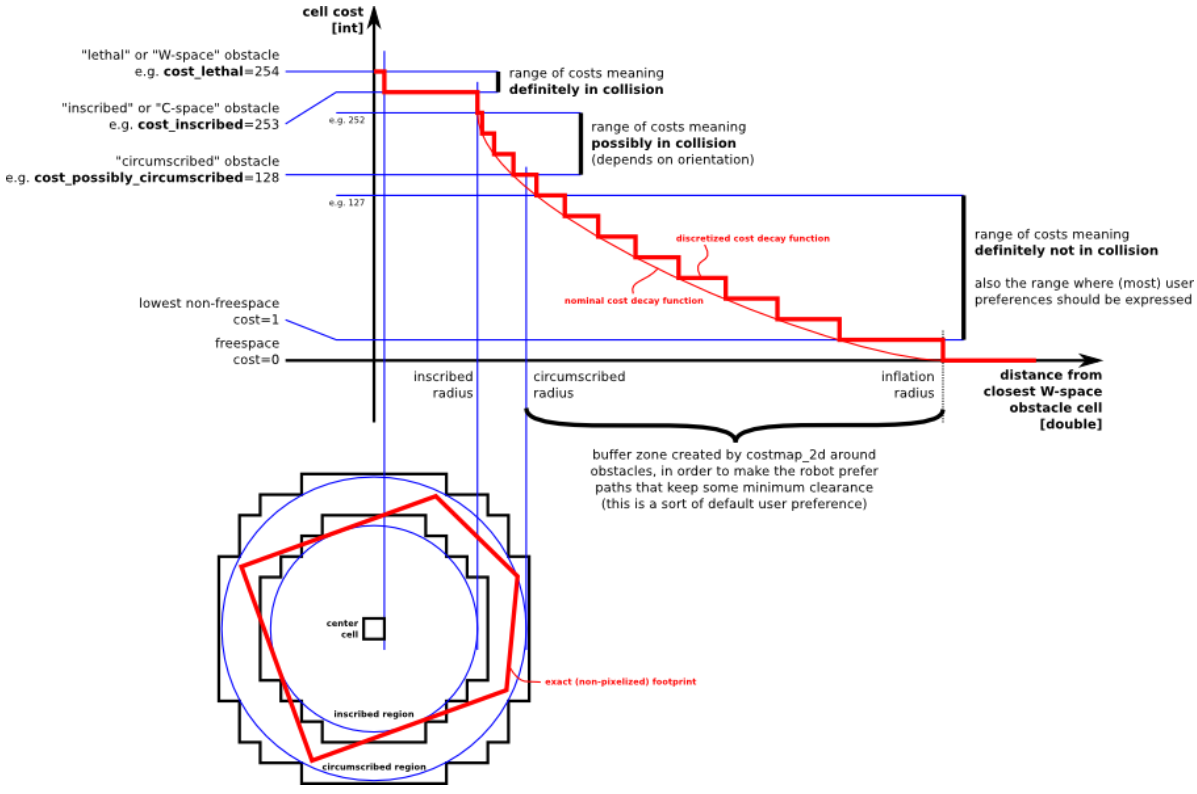
ObstacleCostmapPlugin

- track_unknown_space：如果为false，则每个像素具有两种状态之一：致命障碍或自由。如果为true，则每个像素具有三种状态之一：致命障碍、自由或未知。
- footprint_clearing_enabled：如果为true，则机器人足迹将清除（标记为自由）其移动的空间。
- combination_method：更改障碍层处理来自其以外层的传入数据的方式。可能的值有“覆盖”（0）、“最大值”（1）和“无”（99）。

VoxelCostmapPlugin

- origin_z：地图的z原点（单位：m）。
- z_resolution：地图的z分辨率（单位：m/cell）。
- z_voxels：每个垂直列中的体素数量，栅格高度为z_resolution * z_voxels
- unknown_threshold：列中被认为“已知”的未知单元格数
- mark_threshold：被视为“空闲”的列中允许的最大标记单元格数。
- publish_voxel_map：是否为可视化目的发布基础体素栅格。
- footprint_clearing_enabled：如果为true，则机器人足迹将清除（标记为自由）其移动的空间。

6）Inflation layer



膨胀代价值随着机器人离障碍物距离增加而减少。为代价地图的代价值定义5个与机器人有关的特定符号。

- 致命的（"Lethal" cost）：说明该单元格中有真实的障碍。若机器人的中心在该单元格中，机器人必然会跟障碍物相撞。
- 内切（"Inscribed" cost）：说明该单元格离障碍物的距离小于机器人内切圆半径。若机器人的中心位于等于或高于"Inscribed" cost的单元格，机器人必然会跟障碍物相撞。
- 可能外切（"Possibly circumscribed" cost ）：说明一个单元格离障碍物的距离小于机器人外切圆半径，但是大于内切圆半径。若机器人的中心位于等于或高于"Possibly circumscribed" cost 的单元格，机器不一定会跟障碍物相撞，其取决于机器人的方位。
- 自由空间（"Freespace" ）：没有任何东西可以阻止机器人去那里。
- 未知（"Unknown"）： 未知空间。

参数

- inflation_radius：地图将障碍物代价值膨胀到的半径（单位：m）。
- cost_scaling_factor：膨胀期间应用于代价值的比例因子。

7.5、planner_params

7.5.1、global_planner

nav_core::BaseGlobalPlanner为导航中使用的全局规划器提供接口。所有作为move_base节点插件编写的全局规划器都必须遵守此接口。关于NavaCys::BaseGoLBalPrimeNe：C++文档的文档可以在这里找到：BaseGlobalPlanner documentation。

全局路径规划插件

- navfn：基于栅格地图的全局规划器，使用导航功能时计算机器人的路径。实现了dijkstra和A*全局规划算法。(插件名："navfn/NavfnROS")
- global_planner：重新实现了Dijkstra和A*全局路径规划算法,可以看作navfn的改进版。(插件名："global_planner/GlobalPlanner")
- carrot_planner：一个简单的全局路径规划器，它获取用户指定的目标点，并尝试将机器人移动到尽可能靠近它的位置，即使目标点位于障碍物中。(插件名："carrot_planner/CarrotPlanner")

全局路径规划global_planner_params.yaml

```yaml
GlobalPlanner:
  allow_unknown: false
  default_tolerance: 0.2
  visualize_potential: false
  use_dijkstra: true
  use_quadratic: true
  use_grid_path: false
  old_navfn_behavior: false
  lethal_cost: 253
  neutral_cost: 50
  cost_factor: 3.0
  publish_potential: true
  orientation_mode: 0
  orientation_window_size: 1
```

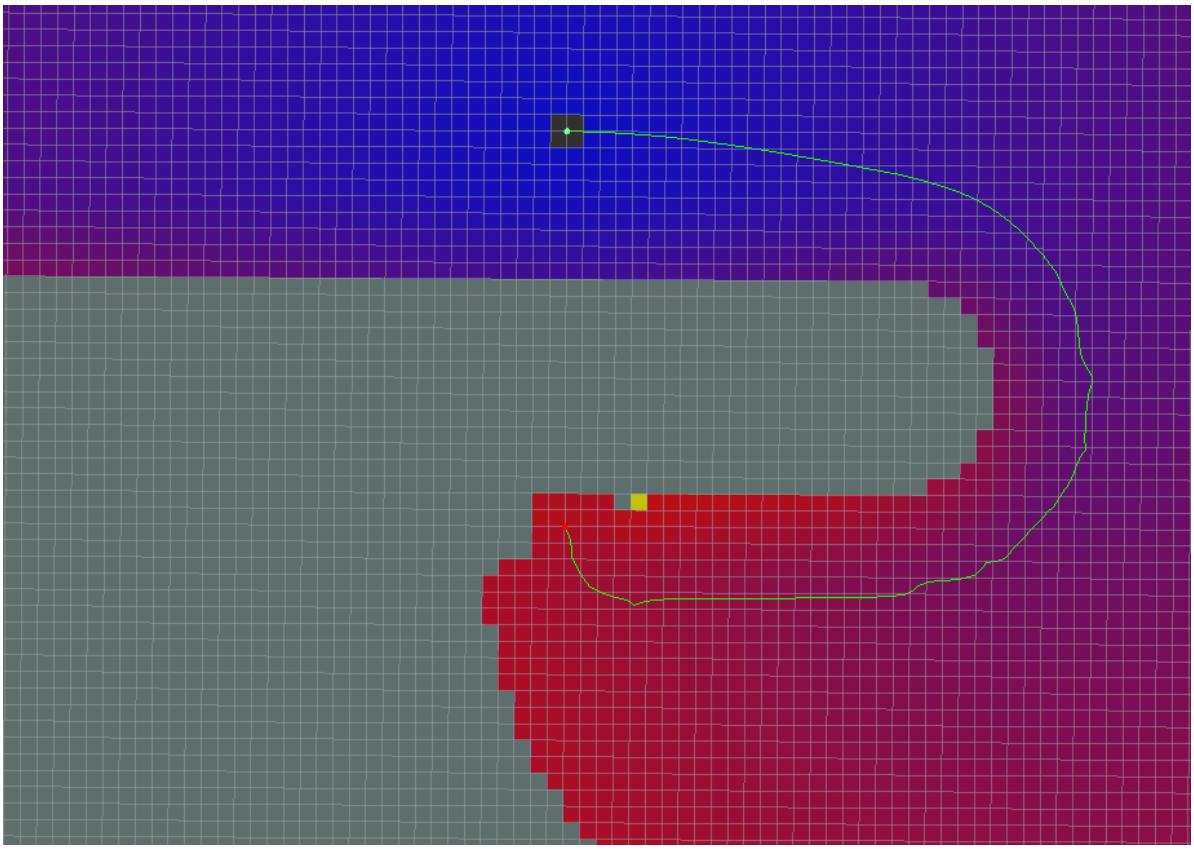
参数解析

- allow_unknown：是否选择探索未知区域。只设计该参数为true还不行，还要在costmap_commons_params.yaml中设置`track_unknown_space`必须同样设置为`true`。
- default_tolerance：当设置的目的地被障碍物占据时,需要以该参数为半径寻找到最近的点作为新目的地点。
- visualize_potential：是否显示从PointCloud2计算得到的可能的区域。
- use_dijkstra：如果为true，则使用dijkstra算法。否则，A*。
- use_quadratic：设置为true，将使用二次函数近似函数，否则使用更加简单的计算方式，这样节省硬件计算资源。
- use_grid_path：如果为true，则创建沿栅格边界的路径。否则，使用梯度下降法，路径更为光滑点。
- old_navfn_behavior：如果你想要让global_planner跟之前的navfn版本效果一样，就设true，所以不建议设置为true。
- lethal_cost：障碍物致命区域的代价数值（可动态配置）。
- neutral_cost：障碍物中等代价值（可动态配置）。
- cost_factor：代价地图与每个代价值相乘的系数（可动态配置）。

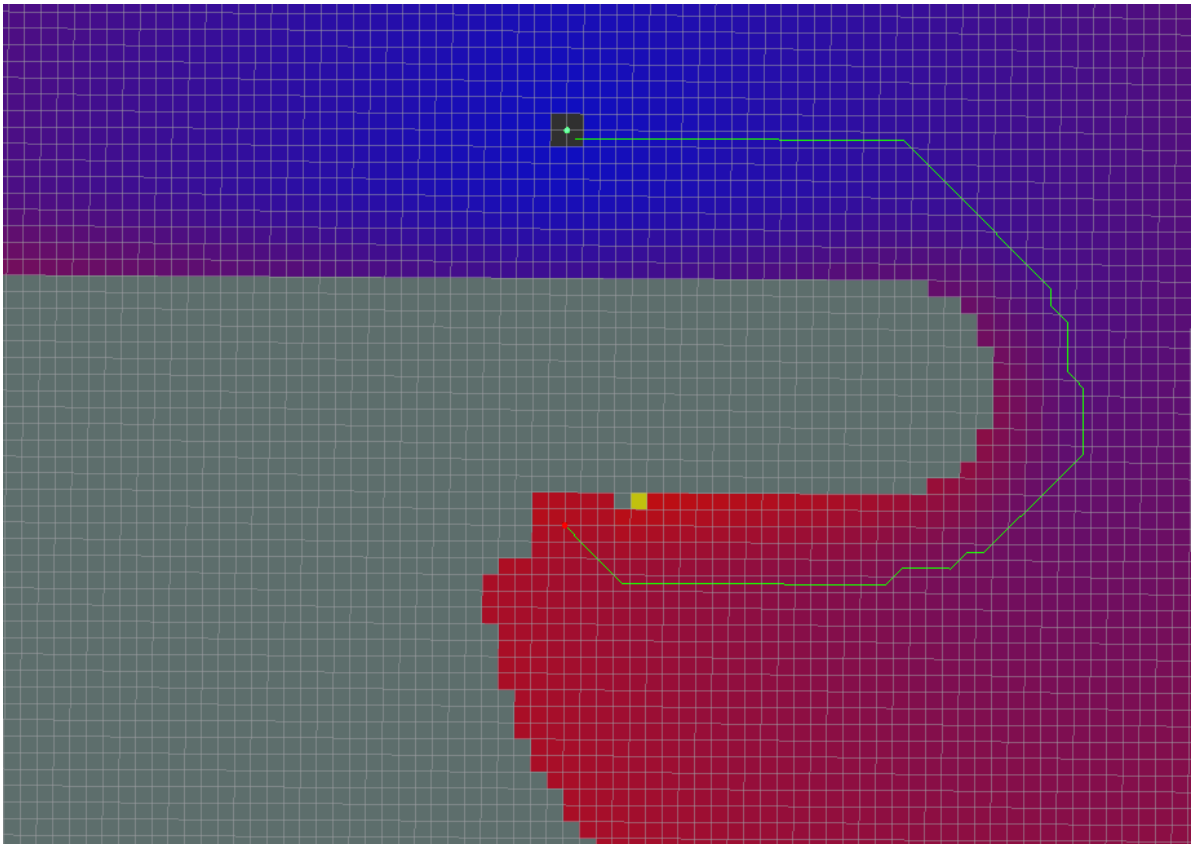
- publish_potential：是否发布可能的代价地图（可动态配置）。
- orientation_mode：设置每个点的朝向。(None=0, Forward=1, Interpolate=2, ForwardThenInterpolate=3, Backward=4, Leftward=5, Rightward=6)（可动态配置）。
- orientation_window_size：根据定向方式指定的位置积分来得到使用窗口的方向；默认值1（可动态配置）。
- outline_map：用致命的障碍勾勒出全局代价地图。对于非静态（滚动窗口）全局代价地图的使用，需要将其设置为false

全局路径规划算法效果图

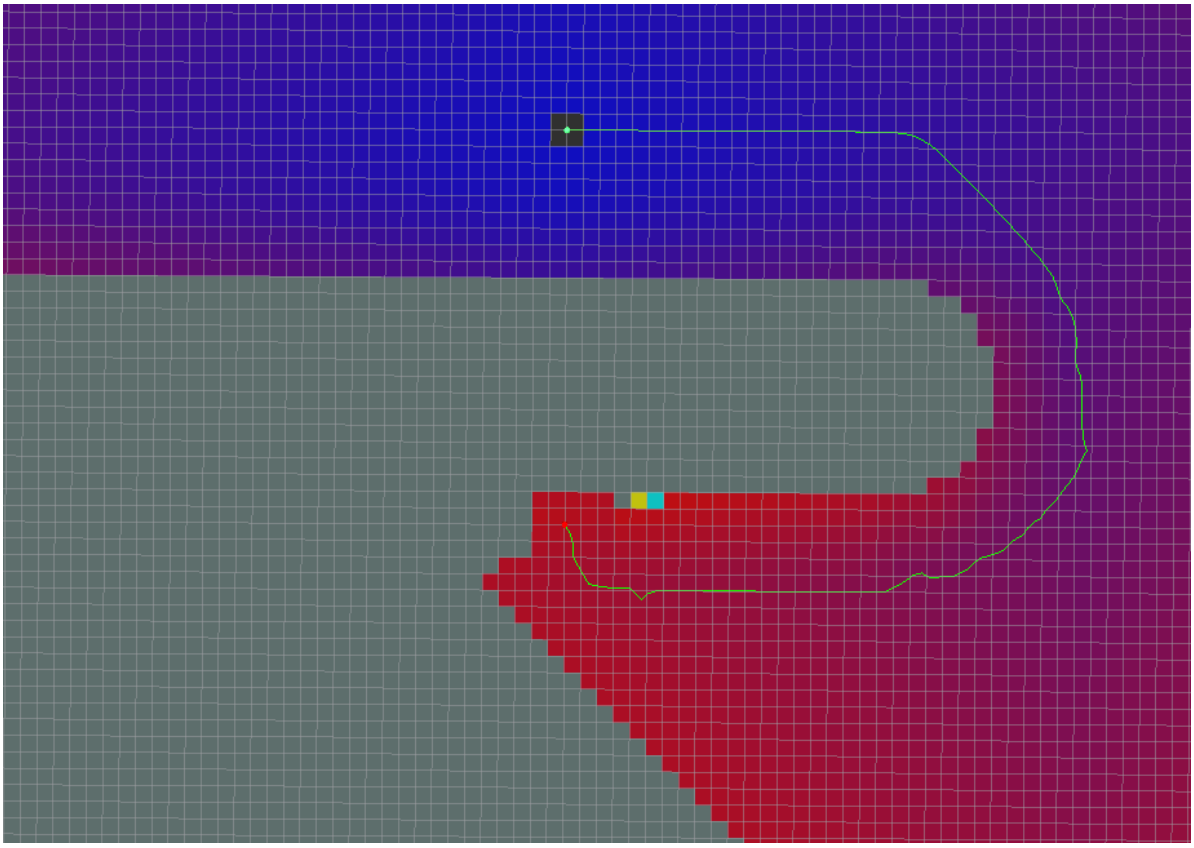
- 所有参数都是默认值



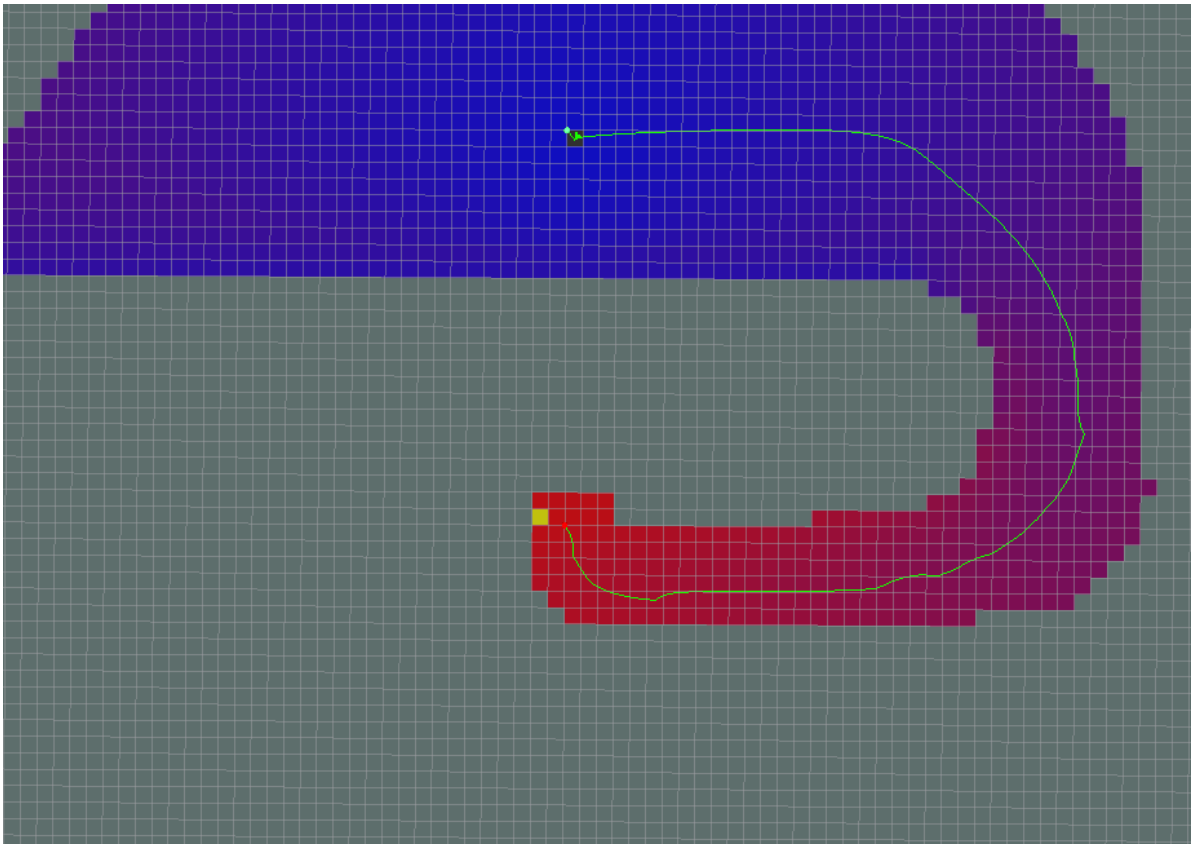
- use_grid_path=True



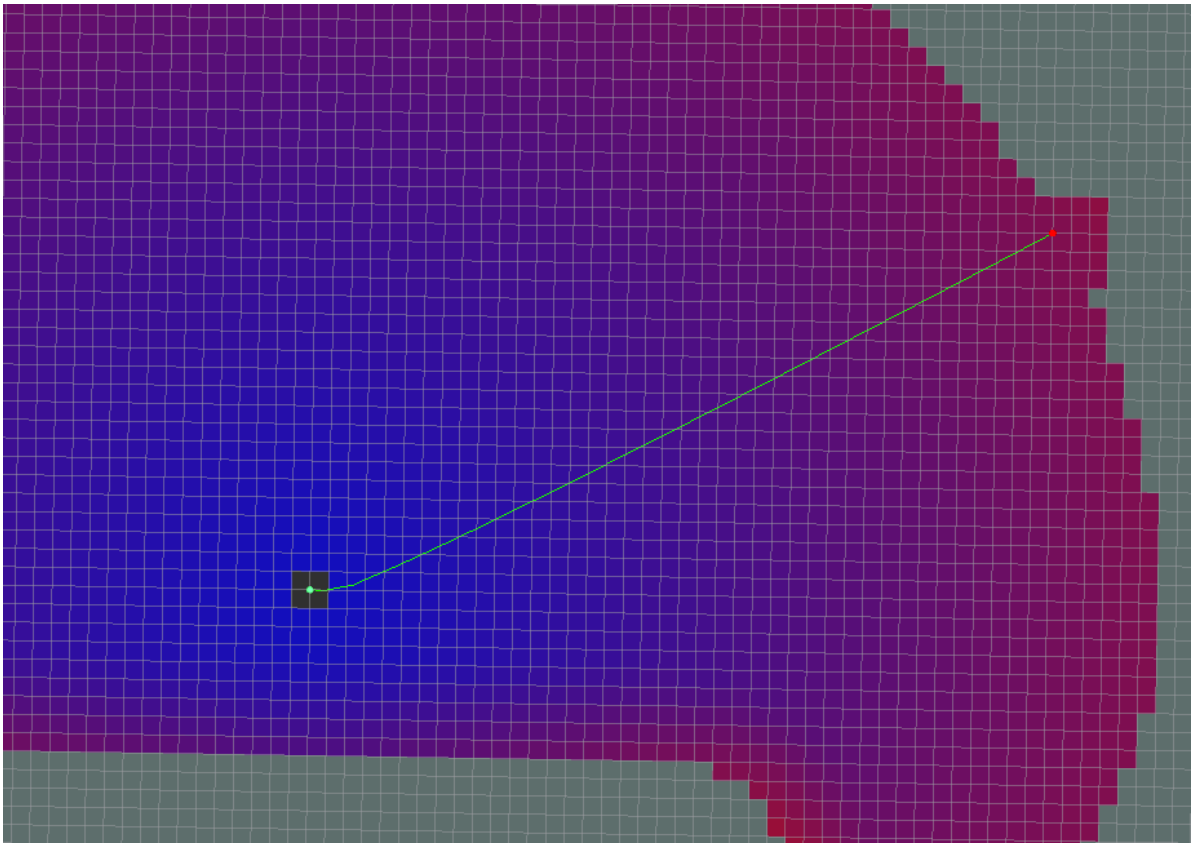
- use_quadratic=False



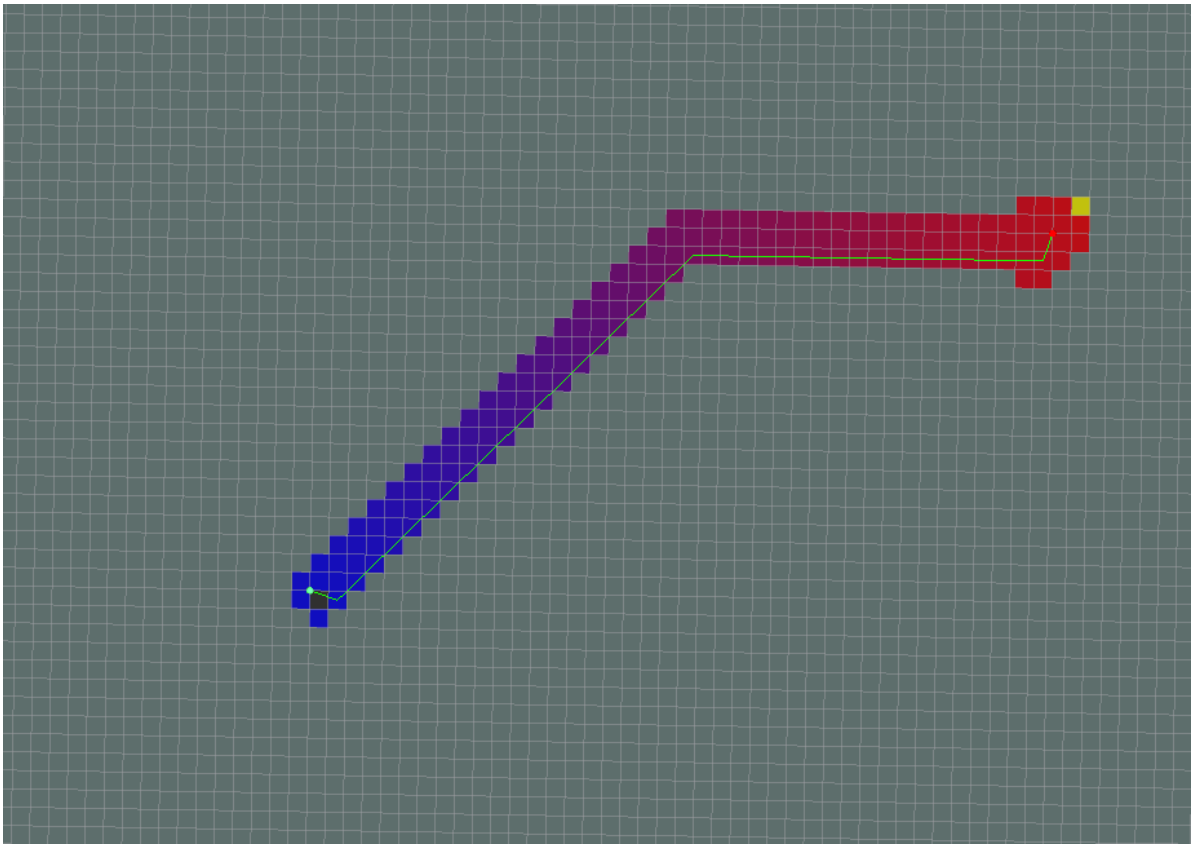
- use_dijkstra=False



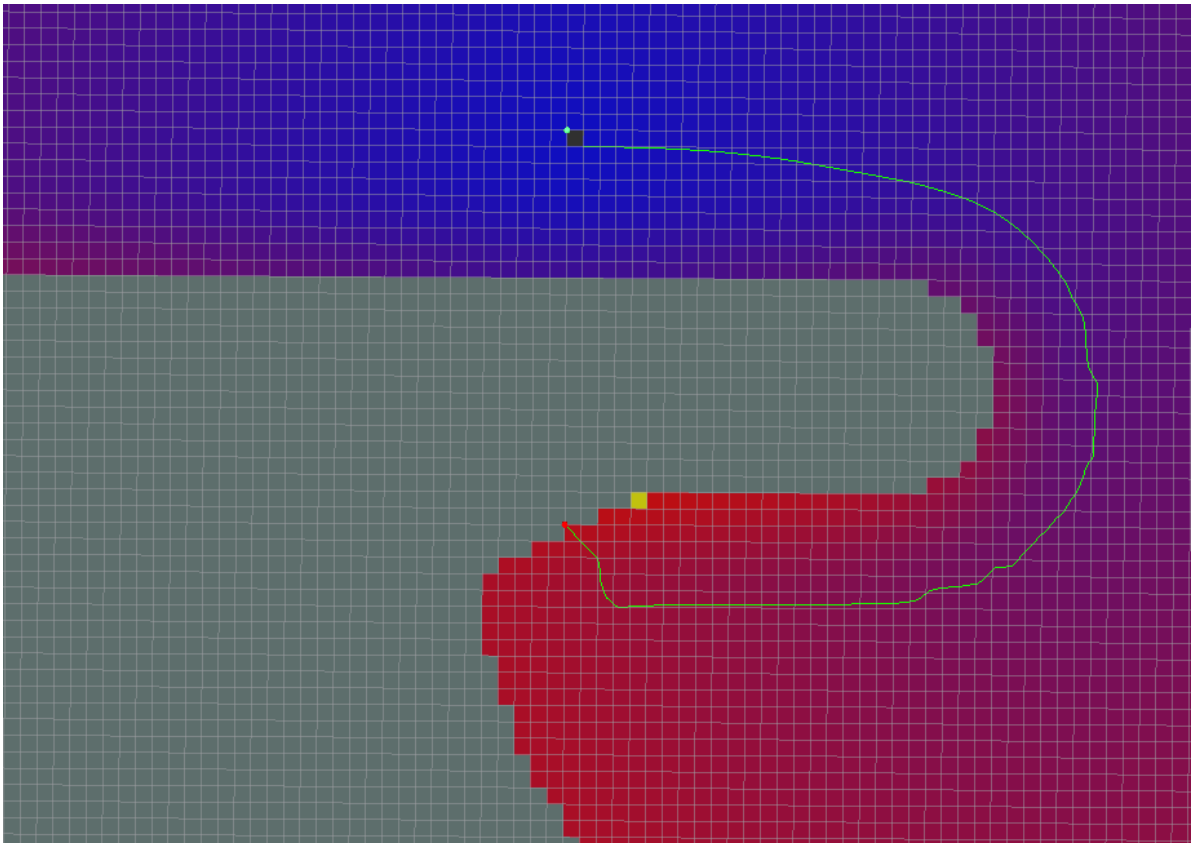
- Dijkstra



- A*



- old_navfn_behavior=True



如果在最开始的阶段会出现：

```
[ERROR] [1611670223.557818434, 295.312000000]: NO PATH!
[ERROR] [1611670223.557951973, 295.312000000]: Failed to get a plan from
potential when a legal potential was found. This shouldn't happen.
```

这种情况跟跟机器人设定的方向有关，建议全局路径规划试试默认的【navfn】插件。

7.5.2、local_planner

nav_core::BaseLocalPlanner 为导航中使用的局部路径规划人员提供接口。所有作为move_base节点插件编写的局部路径规划器都必须遵守此接口。关于NavaCys::BaseLoCalPrnor的C++ API的文档可以在这里找到：BaseLocalPlanner documentation。

局部路径规划插件

- base_local_planner：实现了Trajectory Rollout和DWA两种局部规划算法。
- dwa_local_planner ：与base_local_planner的DWA相比，模块化DWA实现具有更清晰、更易于理解的界面和更灵活的y轴变量的优点。
- teb_local_planner：实现用于在线轨迹优化的Timed-Elastic-Band方法。
- eband_local_planner：Implements the Elastic Band method on the SE2 manifold仅适用于圆形、差速驱动、向前驱动（而不是向后）、全向的机器人。
- mpc_local_planner：Provides several model predictive control approaches embedded in the SE2 manifold

TEB与与DWA对比：

teb在运动过程中会调整自己的位姿朝向，当到达目标点时，通常机器人的朝向也是目标朝向而不需要旋转。dwa则是先到达目标坐标点，然后原地旋转到目标朝向。对于两轮差速底盘，teb在运动中调节朝向会使运动路径不流畅，在启动和将到达目标点时出现不必要的后退。这在某些应用场景里是不允许的。因为后退可能会碰到障碍物。而原地旋转到合适的朝向再径直走开是更为合适的运动策略。这也是teb需要根据场景要优化的地方。

1）dwa_local_planner

dwa_local_planner软件包支持任何机器人，其底盘可以表示为凸多边形或圆形。该软件包提供了一个控制器，在平面驱动机器人移动。该控制器将路径规划器连接到机器人。规划器使用地图为机器人创建从起点到目标位置的运动轨迹，发送给机器人的dx、dy、dtheta速度。

DWA算法的基本思想

- 机器人控制空间中的离散采样（dx、dy、dtheta）
- 对于每个采样速度，从机器人的当前状态执行正向模拟，以预测如果采样速度应用一段（短）时间会发生什么。
- 评估（评分）前方模拟产生的每条轨迹，使用包含以下特征的指标：接近障碍物、接近目标、接近全局路径和速度。丢弃非法轨迹（与障碍物碰撞的轨迹）。
- 选择得分最高的轨迹，并将相关速度发送到移动机器人。
- 清洗数据并重复。

可以设置大量ROS参数来自定义dwa_local_planner::DWAPlannerROS的行为。这些参数分为几个类别：机器人配置、目标容差、正向模拟、轨迹评分、振荡预防和全局规划。可以使用dynamic_reconfigure工具调试这些参数，以便于在运行的系统中调整局部路径规划器。

```
DWAPlannerROS:
# Robot Configuration Parameters
  acc_lim_x: 2.5
  acc_lim_y: 2.5
  acc_lim_th: 3.2
  max_vel_trans: 0.55
  min_vel_trans: 0.1
  max_vel_x: 0.55
  min_vel_x: 0.0
  max_vel_y: 0.1
  min_vel_y: -0.1
  max_rot_vel: 1.0
```

```
  min_rot_vel: 0.4
# Goal Tolerance Parameters
  yaw_goal_tolerance: 0.05
  xy_goal_tolerance: 0.10
  latch_xy_goal_tolerance: false
# Forward Simulation Parameters
  sim_time: 2.0
  sim_granularity: 0.025
  vx_samples: 6
  vy_samples: 1
  vth_samples: 20
  controller_frequency: 5.0
# Trajectory Scoring Parameters
  path_distance_bias: 90.0      # 32.0
  goal_distance_bias: 24.0      # 24.0
  occdist_scale: 0.3            # 0.01
  forward_point_distance: 0.325 # 0.325
  stop_time_buffer: 0.2         # 0.2
  scaling_speed: 0.20           # 0.25
  max_scaling_factor: 0.2       # 0.2
  publish_cost_grid: false
# Oscillation Prevention Parameters
  oscillation_reset_dist: 0.05  # default 0.05
# Global Plan Parameters
  prune_plan: false
```

机器人配置参数

- acc_lim_x：机器人的x加速度极限（单位：m/s^2）。
- acc_lim_y：y方向的加速度绝对值（单位：m/s^2），注意：该值只有全向移动的机器人才需配置。
- acc_lim_th：旋转加速度的绝对值（单位：rad/s^2）。
- max_vel_trans：平移速度最大值绝对值（单位：m/s）。
- min_vel_trans：平移速度最小值的绝对值（单位：m/s）。
- max_vel_x：x方向最大速度的绝对值（单位：m/s）。
- min_vel_x：x方向最小值绝对值（单位：m/s），如果为负值表示可以后退。
- max_vel_y：y方向最大速度的绝对值（单位：m/s）。
- min_vel_y：y方向最小速度的绝对值（单位：m/s）。
- max_rot_vel：最大旋转速度的绝对值（单位：rad/s）。
- min_rot_vel：最小旋转速度的绝对值（单位：rad/s）。

目标容差参数

- yaw_goal_tolerance：到达目标点时，偏行角允许的误差（单位：rad）。
- xy_goal_tolerance：到达目标点时，在x&y距离内允许的公差（单位：m）。
- latch_xy_goal_tolerance：设置为true，如果到达容错距离内，机器人就会原地旋转，即使转动跑出容错距离外。

正向模拟参数

- sim_time：向前仿真轨迹的时间（单位：s）。
- sim_granularity：给定轨迹上各点之间的步长（单位：m）。
- vx_samples：x方向速度空间的采样点数。
- vy_samples：y方向速度空间采样点数。
- vth_samples：旋转方向的速度空间采样点数。
- controller_frequency：调用此控制器的频率（单位：Hz）。

轨迹评分参数

- path_distance_bias：定义控制器与给定路径接近程度的权重。
- goal_distance_bias：定义控制器与局部目标点的接近程度的权重。
- occdist_scale：定义控制器躲避障碍物的权重。
- forward_point_distance：从机器人中心点到放置额外计分点的距离（单位：m）。
- stop_time_buffer：碰撞前机器人必须提前停止的时间长度（单位：s）。
- scaling_speed：启动缩放机器人底盘的速度（单位：m/s）。
- max_scaling_factor：机器人底盘的最大缩放参数。
- publish_cost_grid：是否发布规划器在规划路径时的代价网格。如果设置为true，那么就会在~/cost_cloud话题上发布sensor_msgs/PointCloud2类型消息。

防振参数

- oscillation_reset_dist：机器人运动多远距离才会重置振荡标记（单位：m）。

全局规划参数

- prune_plan：机器人前进时，是否清除身后1m外的轨迹。

2）teb_local_planner

teb_local_planner是一个基于优化的局部轨迹规划器。支持差分模型，car-like模型。该软件包实现了一个用于移动机器人导航和控制的在线最优局部轨迹规划器，通过求解稀疏标量化多目标优化问题，有效地获得了最优轨迹。用户可以为优化问题提供权重，以便在目标冲突的情况下指定行为。

teb_local_planner软件包允许用户设置参数以自定义行为。这些参数分为几类：机器人配置、目标公差、轨迹配置、障碍物、优化、独特拓扑中的规划和其他参数。其中一些被选为符合基本的本地规划者。许多（但不是全部）参数可以在运行时使用rqt_重新配置进行修改。

局部路径规划器teb_local_planner_params.yaml

```
TebLocalPlannerROS:
# Miscellaneous Parameters
 map_frame: odom
 odom_topic: odom
# Robot
 acc_lim_x: 0.5
 acc_lim_theta: 0.5
 max_vel_x: 0.4
 max_vel_x_backwards: 0.2
 max_vel_theta: 0.3
 min_turning_radius: 0.0
 footprint_model:
   type: "point"
# GoalTolerance
 xy_goal_tolerance: 0.2
 yaw_goal_tolerance: 0.1
 free_goal_vel: False
# Trajectory
 dt_ref: 0.3
 dt_hysteresis: 0.1
 min_samples: 3
 global_plan_overwrite_orientation: True
 allow_init_with_backwards_motion: False
 max_global_plan_lookahead_dist: 3.0
 global_plan_viapoint_sep: -1
 global_plan_prune_distance: 1
```

```
  exact_arc_length: False
  feasibility_check_no_poses: 5
  publish_feedback: False
# Obstacles
  min_obstacle_dist: 0.25
  inflation_dist: 0.6
  include_costmap_obstacles: True
  costmap_obstacles_behind_robot_dist: 1.5
  obstacle_poses_affected: 15
  dynamic_obstacle_inflation_dist: 0.6
  include_dynamic_obstacles: True
  costmap_converter_plugin: ""
  costmap_converter_spin_thread: True
  costmap_converter_rate: 5
# Optimization
  no_inner_iterations: 5
  no_outer_iterations: 4
  optimization_activate: True
  optimization_verbose: False
  penalty_epsilon: 0.1
  obstacle_cost_exponent: 4
  weight_max_vel_x: 2
  weight_max_vel_theta: 1
  weight_acc_lim_x: 1
  weight_acc_lim_theta: 1
  weight_kinematics_nh: 1000
  weight_kinematics_forward_drive: 1
  weight_kinematics_turning_radius: 1
  weight_optimaltime: 1 # must be > 0
  weight_shortest_path: 0
  weight_obstacle: 100
  weight_inflation: 0.2
  weight_dynamic_obstacle: 10
  weight_dynamic_obstacle_inflation: 0.2
  weight_viapoint: 1
  weight_adapt_factor: 2
# Parallel Planning
  enable_homotopy_class_planning: True
  enable_multithreading: True
  max_number_classes: 4
  selection_cost_hysteresis: 1.0
  selection_prefer_initial_plan: 0.9
  selection_obst_cost_scale: 100.0
  selection_alternative_time_cost: False
  roadmap_graph_no_samples: 15
  roadmap_graph_area_width: 5
  roadmap_graph_area_length_scale: 1.0
  h_signature_prescaler: 0.5
  h_signature_threshold: 0.1
  obstacle_heading_threshold: 0.45
  switching_blocking_period: 0.0
  viapoints_all_candidates: True
  delete_detours_backwards: True
  max_ratio_detours_duration_best_duration: 3.0
  visualize_hc_graph: False
  visualize_with_time_as_z_axis_scale: False
# Recovery
  shrink_horizon_backup: True
```

```
shrink_horizon_min_duration: 10
oscillation_recovery: True
oscillation_v_eps: 0.1
oscillation_omega_eps: 0.1
oscillation_recovery_min_duration: 10
oscillation_filter_duration: 10
```

机器人配置

- acc_lim_x：机器人的最大平移加速度（单位：m/s^2）。
- acc_lim_theta：机器人的最大角加速度（单位：rad/s^2）。
- max_vel_x：机器人的最大平移速度（单位：m/s）。
- max_vel_x_backwards：向后行驶时机器人的最大绝对平移速度（单位：m/s）。
- max_vel_theta：机器人的最大角速度（单位：rad/s）。

  以下参数仅与carlike机器人相关
- min_turning_radius：carlike机器人的最小转弯半径（对于差速机器人，设置为零）。
- wheelbase：后轴和前轴之间的距离。对于后轮机器人，该值可能为负值（仅当/cmd_angle_instead_rotvel设置为true时才需要）。
- cmd_angle_instead_rotvel：用相应的转向角[-pi/2，pi/2]代替指令速度信息中的转速。

  以下参数仅与完整机器人相关：ROS动力学中的新参数
- max_vel_y：机器人的最大扫射速度（非完整机器人应为零！）。
- acc_lim_y：机器人的最大扫射加速度。

  以下参数与用于优化的底盘模型相关
- footprint_model:

  type: "point":

  参数【footprint_model】

  指定用于优化的机器人示意图模型类型。不同的类型有"point", "circular", "line", "two_circles", "polygon"。模型的类型显著影响所需的计算时间。
- footprint_model/radius：此参数仅与“circular”类型相关。它包含圆的半径。圆心位于机器人的旋转轴上。
- footprint_model/line_start：此参数仅与“line”类型相关。它包含线段的起始坐标。
- footprint_model/line_end：此参数仅与“line”类型相关。它包含线段的端点坐标。
- footprint_model/front_offset：此参数仅与“two_circles”类型相关。它描述了前圆的中心沿机器人的x轴移动的程度。假设机器人的旋转轴位于[0,0]。
- footprint_model/front_radius：此参数仅与“two_circles”类型相关。。它包含前圆的半径。
- footprint_model/rear_offset：此参数仅与“two_circles”类型相关。它描述了后圆的中心沿机器人的负x轴移动的程度。假设机器人的旋转轴位于[0,0]。
- footprint_model/rear_radius：此参数仅与“two_circles”类型相关。它包含后圆的半径。
- footprint_model/vertices：此参数仅与“polygon”类型相关。它包含多边形顶点列表（每个顶点的二维坐标）。多边形始终是闭合的：不要重复末端的第一个顶点。
- is_footprint_dynamic：如果为true，则在检查轨迹可行性之前更新足迹。

目标公差

- yaw_goal_tolerance：到达目标点时，偏行角允许的误差（单位：rad）。
- xy_goal_tolerance：到达目标点时，在x&y距离内允许的公差（单位：m）。

- free_goal_vel：移除目标速度约束，使机器人能够以最大速度到达目标。

轨迹配置

- dt_ref：期望的轨迹时间分辨率。
- dt_hysteresis：根据当前时间分辨率自动调整大小的滞后，通常建议约为dt_ref的10%。
- max_samples：最小样本数（应始终大于2）。
- global_plan_overwrite_orientation：覆盖全局规划器提供的局部子目标的方向（因为它们通常只提供二维路径）
- global_plan_viapoint_sep：如果为正，则从全局平面中提取过孔点（路径跟随模式）。该值确定参考路径的分辨率（沿全局平面的每个连续过孔点之间的最小间隔，如果为负值：禁用）。
- max_global_plan_lookahead_dist：指定优化时考虑的全局计划子集的最大长度（累积欧氏距离）。实际长度由局部代价地图的大小和该最大界限的逻辑连接决定。设置为零或负以停用此限制。
- force_reinit_new_goal_dist：如果前一个目标的更新间隔超过指定的米数（跳过热启动），则重新初始化轨迹。
- feasibility_check_no_poses：指定每个采样间隔应检查预测计划中哪个姿势的可行性。
- publish_feedback：发布包含完整轨迹和活动障碍物列表的规划器反馈（应仅在评估或调试时启用）
- shrink_horizon_backup：允许规划器在自动检测到问题（例如不可行）时临时缩小范围（50%）。
- allow_init_with_backwards_motion：如果为true，则如果目标在本地成本图中的起点之后，则可能会使用向后运动初始化基本轨迹（仅当机器人配备有后传感器时才建议这样做）。
- exact_arc_length：如果为true，则规划器在速度、加速度和转向率计算中使用精确的弧长（->增加的cpu时间），否则使用欧几里德近似值。
- shrink_horizon_min_duration：在检测到不可行轨迹的情况下，指定缩小地平线的最短持续时间。

障碍物

- min_obstacle_dist：与障碍物的最小期望间距（单位：m）。
- include_costmap_obstacles：指定是否应考虑局部代价地图的障碍。
- costmap_obstacles_behind_robot_dist：限制机器人后面规划时考虑的占用的局部代价地图障碍物（单位：m）。
- obstacle_poses_affected：每个障碍物位置都附加到轨迹上最近的姿势以保持距离。
- inflation_dist：非零惩罚成本障碍物周围的缓冲区（应大于最小障碍物距离，以便生效）。
- include_dynamic_obstacles：如果此参数设置为true，则在优化过程中，通过恒定速度模型预测和考虑速度非零的障碍物的运动。
- legacy_obstacle_association：修改了将轨迹姿态与障碍物连接以进行优化的策略。

以下参数仅在需costmap_converter插件时相关：

- costmap_converter_plugin: ""

  定义插件名称，以便将代价地图单元转换为点/线/多边形。设置空字符串以禁用转换，以便将所有单元格视为点障碍物。

- costmap_converter_spin_thread：如果设置为true，costmap_converter将在不同的线程中调用其回调队列

- costmap_converter_rate：定义costmap_converter插件处理当前代价地图的频率的速率（单位：Hz）。

优化

- no_inner_iterations：在每个外层循环迭代中调用的实际解算器迭代次数。
- no_outer_iterations：每个外层循环迭代都会根据所需的时间分辨率dt_ref自动调整轨迹大小，并调用内部优化器（不执行内部迭代）。

- penalty_epsilon：为硬约束近似的惩罚函数添加一个小的安全裕度。
- weight_max_vel_x：满足最大允许平移速度的优化权重。
- weight_max_vel_theta：满足最大允许角速度的优化权重。
- weight_acc_lim_x：满足最大允许平移加速度的优化权重。
- weight_acc_lim_theta：满足最大允许角加速度的优化权重。
- weight_kinematics_nh：满足非完整运动学的优化权重。
- weight_kinematics_forward_drive：用于强制机器人仅选择前进方向（正平移速度）的优化重量。
- weight_kinematics_turning_radius：实施最小转弯半径的优化重量（仅适用于carlike机器人）。
- weight_optimaltime：缩短轨迹关于过渡/执行时间的优化权重
- weight_obstacle：与障碍物保持最小距离的优化权重。
- weight_viapoint：用于最小化到过孔点的距离（分别为参考路径）的优化权重。
- weight_inflation：膨胀惩罚的优化权重（应较小）。
- weight_adapt_factor：某些特殊权重（当前权重）在每次外部TEB迭代中都会通过该因子重复缩放。

规划

- enable_homotopy_class_planning：在独特的拓扑中激活并行规划。
- enable_multithreading：激活多线程以规划不同线程中的每条轨迹。
- max_number_classes：指定所考虑的不同轨迹的最大数量。
- selection_cost_hysteresis：指定一个新的候选轨迹必须有多少轨迹成本才能被选择。
- selection_obst_cost_scale：额外扩展障碍成本条件。
- selection_viapoint_cost_scale：额外扩展过孔点成本条件。
- selection_alternative_time_cost：如果为true，则时间成本（时间差平方和）将替换为总过渡时间。
- roadmap_graph_no_samples：指定为创建路线图而生成的样本数。
- roadmap_graph_area_width：随机关键点/航路点在起点和目标之间的矩形区域中采样（单位：m）。
- obstacle_heading_threshold：指定障碍物标题和目标标题之间的标量乘积的值，以便在探索时考虑它们（障碍物）。
- visualize_hc_graph：可视化为探索独特轨迹而创建的图形。
- viapoints_all_candidates：如果为true，则不同拓扑的所有轨迹都将连接到过孔点集，否则，只有与初始/全局计划共享相同拓扑的轨迹才会与其连接。
- switching_blocking_period：指定允许切换到新等价类之前需要过期的持续时间（单位：s）。

7.6、AMCL

7.6.1、简介

amcl的英文全称是adaptive Monte Carlo localization，是一种二维移动机器人的概率定位系统。其实就是蒙特卡洛定位方法的一种升级版，使用自适应的KLD方法来更新粒子，使用粒子过滤器根据已知地图跟踪机器人的姿势。按照目前的实施，该节点仅适用于激光扫描和激光地图。它可以扩展到处理其他传感器数据。amcl接收基于激光的地图、激光扫描和变换信息，并输出姿势估计。启动时，amcl根据提供的参数初始化其粒子过滤器。请注意，由于默认设置，如果未设置任何参数，则初始过滤器状态将是以（0,0,0）为中心的中等大小的粒子云。

mcl（蒙特卡洛定位）法使用的是粒子滤波的方法来进行定位的。举例说明粒子滤波：首先在平面地图上均匀的撒一把粒子，然后向前移动机器人移动了一米，所有的粒子也跟随机器人向前移动一米。使用每个粒子所处位置模拟一个传感器信息跟观察到的传感器信息（一般是激光）作对比，从而赋给每个粒子一个概率。之后根据生成的概率来重新生成粒子，概率越高的生成的概率越大。这样的迭代之后，所有的粒子会慢慢地收敛到一起，机器人的确切位置也就被推算出来了。

7.6.2、话题与服务

订阅话题	类型	描述
scan	sensor_msgs/LaserScan	激光雷达数据
tf	tf/tfMessage	坐标变换信息
initialpose	geometry_msgs/PoseWithCovarianceStamped	用于（重新）初始化粒子过滤器的平均值和协方差。
map	nav_msgs/OccupancyGrid	设置use_map_topic参数后，AMCL订阅此主题以检索用于基于激光的定位的地图。导航1.4.2中的新增功能。
发布话题	**类型**	**描述**
amcl_pose	geometry_msgs/PoseWithCovarianceStatmped	机器人在地图中的位姿估计，带有协方差信息
particlecloud	geometry_msgs/PoseArray	粒子滤波器维护的位姿估计集合
tf	tf/tfMessage	发布从odom到map的转换
服务端	**类型**	**描述**
global_localization	std_stvs/Empty	初始化全局定位，所有粒子被随机撒在地图上的空闲区域
request_normotion_update	std_stvs/Empty	手动执行更新并发布更新的粒子
set_map	nav_msgs/SetMap	手动设置新地图和姿势的服务。
客户端	**类型**	**描述**
static_map	nav_msgs/GetMap	amcl调用此服务来检索用于激光定位的地图；启动阻止从此服务获取地图。

7.6.3、参数配置

有三类ROS参数可用于配置amcl节点：整体过滤器、激光模型和里程计模型。

- 整体过滤器参数

参数	类型	默认值	描述
~min_particles	int	100	允许的最少粒子数
~max_particles	int	5000	允许的最多粒子数
~kld_err	double	0.1	真实分布与估计分布之间的最大误差
~kld_z	double	0.99	（1-p）的上标准正常分位数，其中p是估计分布误差小于kld_err的概率
~update_min_d	double	0.2(m)	执行一次滤波器更新所需的平移距离
~update_min_a	double	pi/6.0(rad)	执行一次滤波器更新所需的旋转移动
~reseample_interval	int	2	重采样之前滤波器的更新次数
~transform_tolerance	double	0.1(s)	发布变换的时间，以指示此变换在未来有效
~recovery_alpha_slow	double	0.0	慢速平均权重滤波器的指数衰减率，用于决定何时通过添加随机姿态进行恢复操作，0.0表示禁用
~recovery_alpha_fast	double	0.0	快速平均权重滤波器的指数衰减率，用于决定何时通过添加随机姿态进行恢复操作，0.0表示禁用
~initial_pose_x	double	0.0(m)	初始姿态平均值（x），用于初始化高斯分布滤波器
~initial_pose_y	double	0.0(m)	初始姿态平均值（y），用于初始化高斯分布滤波器
~initial_pose_a	double	0.0(m)	初始姿态平均值（yaw），用于初始化高斯分布滤波器
~initial_cov_xx	double	0.5*0.5(m)	初始姿态平均值（x*x），用于初始化高斯分布滤波器
~initial_cov_yy	double	0.5*0.5(m)	初始姿态平均值（y*y），用于初始化高斯分布滤波器
~initial_cov_aa	double	(pi/12)*(pi/12)(rad)	初始姿态平均值（yaw*yaw），用于初始化高斯分布滤波器
~gui_publish_rate	double	-1.0(Hz)	可视化时，发布信息的最大速率，-1.0表示禁用
~save_pose_rate	double	0.5(Hz)	参数服务器中的存储姿态估计initial_pose和协方差*initial_cov*的最大速率，用于后续初始化过滤器。-1.0表示禁用
~use_map_topic	bool	false	当设置为true时，amcl将订阅地图话题，而不是通过服务调用接收地图

参数	类型	默认值	描述
~first_map_only	bool	false	当设置为true时，amcl将只使用它订阅的第一个地图，而不是每次更新接收到的地图
~selective_resampling	bool	false	设置为true时，将在不需要时降低重采样率，并有助于避免粒子剥夺。仅当有效粒子数（N_eff=1/（sum（k_i^2））小于当前粒子数的一半时，才会发生重采样。

- 激光模型参数

参数	类型	默认值	描述
~laser_min_range	double	-1.0	最小扫描范围，设置-1，激光雷达报告的最小使用范围。
~laser_max_range	double	-1.0	最大扫描范围，设置-1，激光雷达报告的最大使用范围。
~laser_max_beams	int	30	更新过滤器时要在每次扫描中使用多少均匀间隔的光束
~laser_z_bit	double	0.95	模型z_bit部分的混合权重
~laser_z_short	double	0.1	模型z_short部分的混合权重
~laser_z_max	double	0.05	模型z_max部分的混合权重
~laser_z_rand	double	0.05	模型z_rand部分的混合权重
~laser_sigma_hit	double	0.2(m)	模型z_hit部分中使用的高斯模型的标准偏差
~laser_lambda_short	double	0.1	模型z_short部分的指数衰减参数
~laser_likelihood_max_dist	double	2.0(m)	地图上测量障碍物膨胀的最大距离
~laser_model_type	string	"likelihood_field"	模型选择，bean、likelihood_field或likelihood_field_prob

- 里程计模型参数

参数	类型	默认值	描述
~odom_model_type	string	"diff"	模型选择，diff、omni、diff-corrected或omni-corrected
~odom_alpha1	double	0.2	根据机器人运动的旋转分量，指定里程计旋转估计中的预期噪声
~odom_alpha2	double	0.2	根据机器人运动的平移分量，指定里程计旋转估计中的预期噪声
~odom_alpha3	double	0.2	根据机器人运动的平移分量，指定里程计平移估计中的预期噪声
~odom_alpha4	double	0.2	根据机器人运动的旋转分量，指定里程计平移估计中的预期噪声
~odom_alpha5	double	0.2	平移相关的噪声参数（仅在模型omni中使用）
~odom_frame_id	string	"odom"	里程计的坐标系
~base_frame_id	string	"base_link"	机器人底盘的坐标系
~global_frame_id	string	"map"	定位系统发布的坐标系
~tf_broadcast	bool	true	设置为false时，amcl不会发布map与odom之间的坐标系变换