move_base_params.yaml

- **controller_frequency**: 以Hz为单位的速率运行控制循环并向基座发送速度cmd_vel命令
- **controller_patience**: 在空间清理操作执行前，控制器会等待多长时间（秒）用来找出一个有效控制
- **planner_patience**: 在空间清理操作执行前，路径规划器等待多长时间（秒）用来找出一个有效规划
- **planner_frequency**: 全局规划操作的执行频率
- **oscillation_timeout**: 执行修复操作之前，允许的震荡时间是几秒
- **oscillation_distance**: 机器人需要移动多少距离才算作没有震荡
- **conservative_reset_dist**: 当在地图中清理出空间时候，距离机器人几米远的障碍将会从costmap清除

costmap_common_params.yaml

- **obstacle_range**: 更新以机器人为中心，按此数值为半径内的障碍物信息
- **raytrace_range**: 更新以机器人为中心，按此数值为半径内的自由空间
- **robot_radius**: 机器人的占用面积（半径）
- **inflation_radius**: 设置代价地图膨胀半径，C-space
- **observation_sources**: 定义了一系列传递空间信息给代价地图的传感器 *（原先wpb里wpb_bringup下的wpb_home_lidar_filter.cpp发布了一个/scan_filtered，所以之前用的scan_filter）*
- **sensor_frame**: 传感器参考坐标系名
- **data_type**: LaserScan或PointCloud，取决于主题使用的信息
- **topic_name**: 发布传感器数据的话题名
- **marking**: 是否用于向代价地图添加障碍物信息
- **clearing**: 是否从代价地图清除障碍信息

global_planner_params.yaml

- **allow_unknown**: 是否允许规划器规划穿过未知区域的路径，只设计该参数为true还不行,还要在costmap_commons_params.yaml中设置track_unknown_space参数也为true才行
- **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计算得到的势区域
- **use_dijkstra**: 设置为true,将使用dijkstra算法，否则使用A*算法
- **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**: 如果设置为true，则会规划一条沿着网格边界的路径，偏向于直线穿越网格，否则将使用梯度下降算法，路径更为光滑点
- **old_navfn_behavior**: 若在某些情况下，想让global_planner完全复制navfn的功能，那就设置为true，但是需要注意navfn是非常旧的ROS系统中使用的，现在已经都用global_planner代替navfn了，所以不建议设置为true
- **lethal_cost**: 致命代价值，默认是设置为253，可以动态来配置该参数
- **neutral_cost**: 中等代价值，默认设置是50，可以动态配置该参数
- **cost_factor**: 代价地图与每个代价值相乘的因子
- **publish_potential**: 是否发布costmap的势函数

- **orientation_mode**: 如何设置每个点的方向（None = 0, Forward = 1, Interpolate = 2, ForwardThenInterpolate = 3, Backward = 4, Leftward = 5, Rightward = 6）
- **orientation_window_size**: 根据orientation_mode指定的位置积分来得到使用窗口的方向，默认值1，可以动态重新配置

dwa_local_planner_params.yaml

- **acc_lim_x**: x方向的加速度绝对值
- **acc_lim_y**: y方向的加速度绝对值,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
- **max_trans_vel**: 平移速度最大值绝对值
- **min_trans_vel**: 平移速度最小值的绝对值
- **max_vel_x**: x方向最大速度的绝对值
- **min_vel_x**: x方向最小值绝对值,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
- **min_vel_y**: y方向最小速度的绝对值
- **max_rot_vel**: 最大旋转速度的绝对值
- **min_rot_vel**: 最小旋转速度的绝对值
- **yaw_goal_tolerance**: 到达目标点时偏行角允许的误差，单位弧度
- **xy_goal_tolerance**: 到达目标点时,在xy平面内与目标点的距离误差
- **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,即使转动是会跑出容错距离外
- **sim_time**: 向前仿真轨迹的时间
- **sim_granularity**: 步长,轨迹上采样点之间的距离,轨迹上点的密集程度
- **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**: 发送给底盘控制移动指令的频率
- **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程度
- **stop_time_buffer**: 为防止碰撞,机器人必须提前停止的时间长度
- **scaling_speed**: 启动机器人底盘的速度
- **max_scaling_factor**: 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
- **oscillation_reset_dist**: 机器人运动多远距离才会重置振荡标记
- **prune_plan**: 机器人前进是是否清除身后1m外的轨迹