移动机器人开发技术（激光SLAM版）配套教学PPT

序　章

第1课 移动机器人的过去、现在及未来

机器人操作系统

第2课 初识ROS

第3课 ROS编程初步

第4课 机器人的坐标变换

第5课 机器人仿真环境

机器人硬件平台

第6课 TurtleBot3仿真环境实战

第7课 自主搭建机器人小车

机器人核心技术

感知	建图与定位	路径规划与导航
第08课 环境感知基础	第10课 机器人的移动控制	第13课 导航基础
第09课 感知数据融合	第11课 SLAM基础	第14课 ROS中的导航包
	第12课 SLAM实战	第15课 ROS导航实战

机器人应用实战

送餐	物流（专题讲座）	图书盘点（专题讲座）
1 送餐机器人结构设计	1 物流机器人结构设计	1 图书盘点机器人结构
2 送餐机器人环境搭建	2 物流机器人环境模拟	2 图书盘点机器人环境
3 送餐机器人建图	3 物流机器人关键技术	3 图书盘点机器人工作模式
4 送餐机器人导航	4 大规模多机器人调度	4 图书盘点中的视觉分析

移动机器人开发技术（激光SLAM版）配套教学PPT

第十五课 ROS导航实践

北京郵電大學
Beijing University of Posts and Telecommunications

移动机器人与智能技术实验室编

宋桂岭 明安龙 2021.9
expsong@qq.com

第15课 ROS导航实践

北邮移动机器人与智能技术实验室 编

1 ROS导航基础回顾

2 ROS导航文件参数

3 ROS导航实验

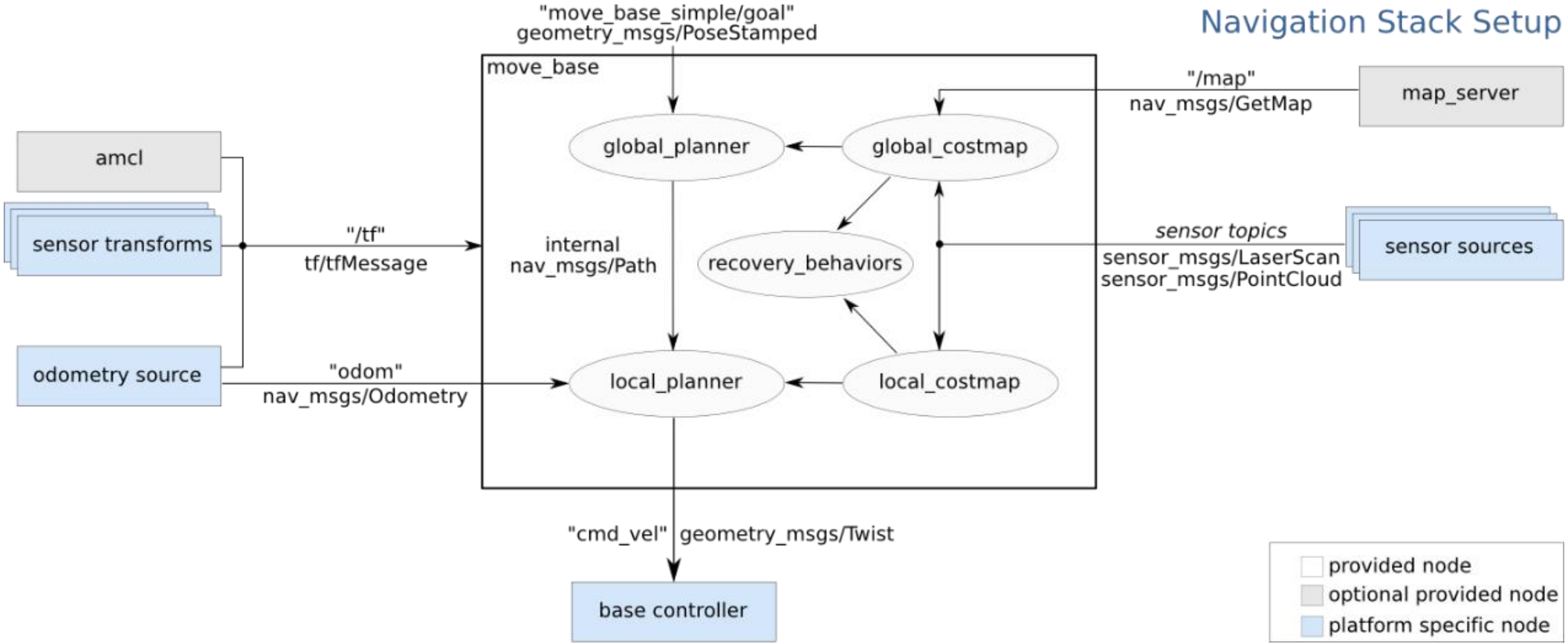
第15课 ROS导航实践

北邮移动机器人与智能技术实验室 编

1 ROS导航基础回顾

1.1 机器人导航的主要步骤

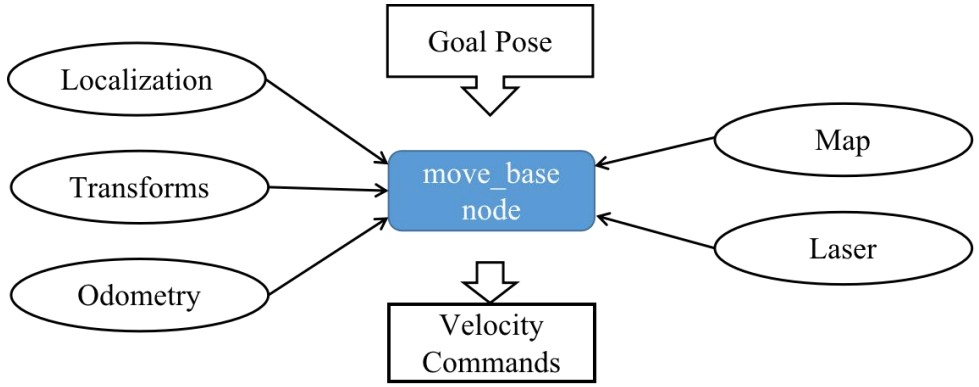
- 重新确定自己在地图中的位置 —— 重定位
- 计算出一条到达目标点的路径 —— 全局路径规划
- 驱动自身到达目标点 —— 局部路径规划

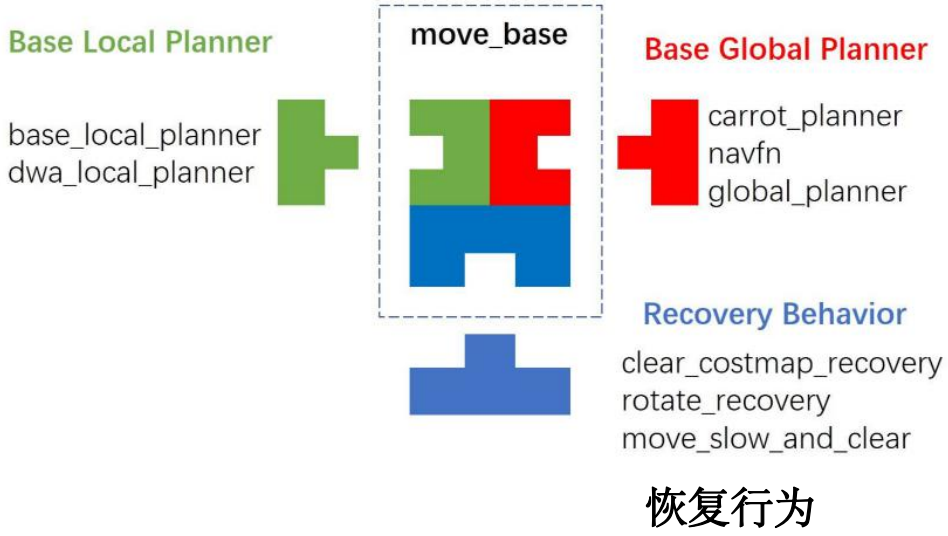


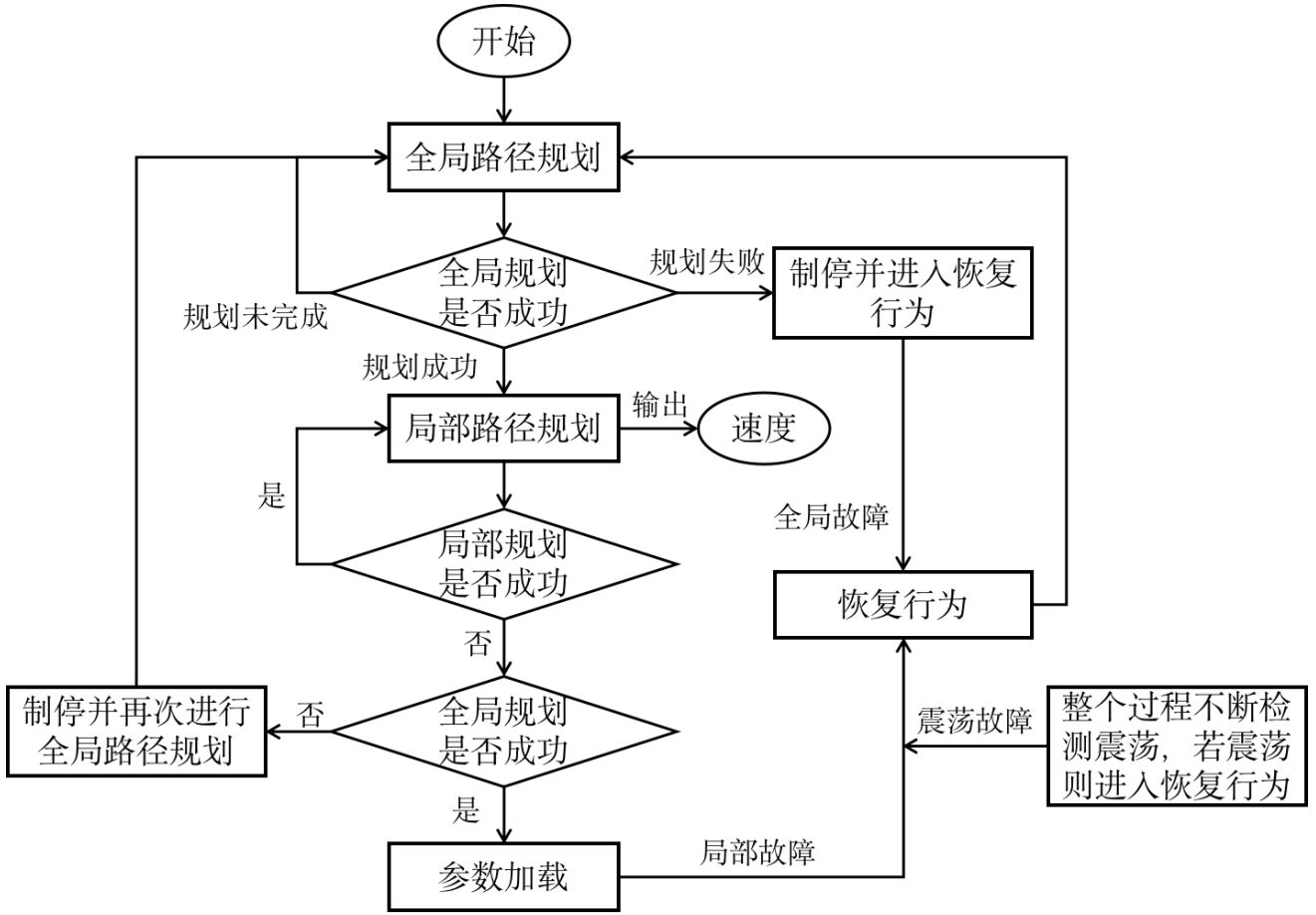
1.2 导航功能包

ROS程序包	功能
amcl	定位
fake_localization	定位
map_server	提供地图
move_base	路径规划节点
nav_core	路径规划接口类
base_local_planner	实现了Trajectory Rollout和Dynamic Window Approach(DWA)局部规划算法
dwa_local_planner	重新实现了DWA局部规划算法
parrot_planner	实现了较简单的全局规划算法
Navfn	实现了Dijkstra和A*全局规划算法
golbal_planner	重新实现了Dijkstra和A*全局规划算法
clear_costmap_recovery	实现了清除代价地图的恢复行为
rotate_recovery	实现了旋转的恢复行为
move_slow_and_clear	路径规划接口类
costmap_2d	2D代价地图
voxel_grid	三维小方块
robot_pose_ekf	机器人位姿的卡尔曼滤波

1.3 move_base


Goal Pose
Localization
Transforms
Odometry
move_base
node
Map
Laser
Velocity
Commands


局部路径规划
Base Local Planner
base_local_planner
dwa_local_planner
move_base
全局路径规划
Base Global Planner
carrot_planner
navfn
global_planner
Recovery Behavior
clear_costmap_recovery
rotate_recovery
move_slow_and_clear
恢复行为


开始
全局路径规划
全局规划
是否成功
规划失败
制停并进入恢复
行为
规划未完成
规划成功
局部路径规划
输出
速度
是
局部规划
是否成功
全局故障
恢复行为
否
制停并再次进行
全局路径规划
否
全局规划
是否成功
震荡故障
整个过程不断检
测震荡，若震荡
则进入恢复行为
是
参数加载
局部故障

第15课 ROS导航实践

北邮移动机器人与智能技术实验室 编

2 ROS导航文件参数

2.1 launch文件

1、引入map文件

第六章中保存的地图路径及文件名

```
<arg name="map_file" default="$(find mrobotit)/map/map.yaml"/>
<node name="map_server_for_test" pkg="map_server" type="map_server" args="$(arg map_file)">
</node>
```

2、引入AMCL定位节点

```
<arg name="use_map_topic" default="false"/>
<arg name="scan_topic" default="scan"/>
<node pkg="amcl" type="amcl" name="amcl" clear_params="true">
    <param name="use_map_topic" value="$(arg use_map_topic)"/>
    <!-- Publish scans from best pose at a max of 10 Hz -->
    <param name="odom_model_type" value="diff"/> <!--omni -->
    <param name="odom_alpha5" value="0.1"/>
    <param name="gui_publish_rate" value="10.0"/>
    <param name="laser_max_beams" value="60"/>
    <param name="laser_max_range" value="12.0"/>
    <param name="min_particles" value="500"/>
    <param name="max_particles" value="2000"/>
    <param name="kld_err" value="0.05"/>
    <param name="kld_z" value="0.99"/>
    <param name="odom_alpha1" value="0.2"/>
    <param name="odom_alpha2" value="0.2"/>
    <!-- translation std dev, m -->
    <param name="odom_alpha3" value="0.2"/>
    <param name="odom_alpha4" value="0.2"/>
    <param name="laser_z_hit" value="0.5"/>
    <param name="laser_z_short" value="0.05"/>
    <param name="laser_z_max" value="0.05"/>
    <param name="laser_z_rand" value="0.5"/>
    <param name="laser_sigma_hit" value="0.2"/>
    <param name="laser_lambda_short" value="0.1"/>
    <param name="laser_model_type" value="likelihood_field"/>
    <!-- <param name="laser_model_type" value="beam"/> -->
    <param name="laser_likelihood_max_dist" value="2.0"/>
     <param name="update_min_d" value="0.25"/>
    <param name="update_min_a" value="0.2"/>
    <param name="odom_frame_id" value="odom_combined"/>
    <param name="resample_interval" value="1"/>
    <!-- Increase tolerance because the computer can get quite busy -->
    <param name="transform_tolerance" value="1.0"/>
    <param name="recovery_alpha_slow" value="0.0"/>
    <param name="recovery_alpha_fast" value="0.0"/>
    <remap from="scan" to="$(arg scan_topic)"/>
</node>
```

2.1 launch文件

3、引入move_base导航节点

```
<node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
    <rosparam file="$(find mrobotit)/param/costmap_common_params.yaml" command="load" ns="global_costmap" />
    <rosparam file="$(find mrobotit)/param/costmap_common_params.yaml" command="load" ns="local_costmap" />
    <rosparam file="$(find mrobotit)/param/local_costmap_params.yaml" command="load" />
    <rosparam file="$(find mrobotit)/param/global_costmap_params.yaml" command="load" />
    <rosparam file="$(find mrobotit)/param/move_base_params.yaml" command="load" />
    <rosparam file="$(find mrobotit)/param/dwa_local_planner_params.yaml" command="load" />
</node>
```

4、引入rviz节点

```
<node name="rviz" pkg="rviz" type="rviz" args="-d $(find mrobot)/rviz/navigation.rviz" required="true">
</node>
```

2.2 costmap_common_params.yaml文件

```
robot_radius: 0.15
obstacle_layer:
  enabled: true
  max_obstacle_height: 2.0
  min_obstacle_height: 0.0
  combination_method: 1
  track_unknown_space:  true
  obstacle_range: 2.0
  raytrace_range: 5.0
  publish_voxel_map: false
  observation_sources:  scan
  scan:
    data_type: LaserScan
    topic: "/scan"
    marking: true
    clearing: true
    expected_update_rate: 0
inflation_layer:
    enabled: true
    cost_scaling_factor:  10.0
    inflation_radius: 0.15
static_layer:
  enabled: true
  map_topic: "/map"
```

设置机器人半径，单位米

obstacle_layer：配置障碍物图层；
enabled：是否启用该层；
obstacle_range：设置机器人检测障碍物的最大范围；
raytrace_range:在机器人移动过程中，实时清除代价地图上的障碍物的最大范围，更新可自由移动的空间数据；
observation_sources：设置导航中所使用的传感器；
scan：添加的激光雷达传感器；
data_type：激光雷达数据类型；
topic：该激光雷达发布的话题名；
marking:：是否可以使用该传感器来标记障碍物；
clearing：是否可以使用该传感器来清除障碍物标记为自由空间

inflation_layer：膨胀层，用于在障碍物外标记一层危险区域，在路径规划时需要避开该危险区域；
enabled：是否启用该层；
cost_scaling_factor：膨胀过程中应用到代价值的比例因子；
inflation_radius：膨胀半径

static_ layer：静态地图层，即SLAM中构建的地图层；
enabled：是否启用该地图层

2.3 global_costmap_params.yaml文件

```
global_costmap:
  global_frame: map
  robot_base_frame: base_footprint
  update_frequency: 1.0
  publish_frequency: 0.5
  static_map: true
  transform_tolerance: 0.5
  plugins:
    - {name: static_layer,          type: "costmap_2d::StaticLayer"}
    - {name: obstacle_layer,        type: "costmap_2d::VoxelLayer"}
    - {name: inflation_layer,       type: "costmap_2d::InflationLayer"}
```

global_frame：全局代价地图需要在哪个坐标系下运行；

robot_base_frame：在全局代价地图中机器人本体的基坐标系，就是机器人上的根坐标系。通过global frame和robot_ base_ frame就可以计算两个坐标系之间的变换，得知机器人在全局坐标系中的坐标了；

update_ frequency：全局代价地图更新频率，一般全局代价地图更新频率设置的比较小；

static_ map：配置是否使用map_server提供的地图来初始化，一般全局地图都是静态的，需要设置为true；

rolling_ window：是否在机器人移动过程中需要滚动窗口，始终保持机器人在当前窗口中心位置；

transform_tolerance：坐标系间转换可以忍受的最大延时；

plugins：在global costmap中使用下面三个插件来融合三个不同图层，分别是static_layer、obstacle_layer和inflation_layer，合成一个master_layer来进行全局路径规划。

2.4 local_costmap_params.yaml文件

```
local_costmap:
  global_frame: odom_combined
  robot_base_frame: base_footprint
  update_frequency: 3.0
  publish_frequency: 1.0
  static_map: false
  rolling_window: true
  width: 2.0
  height: 2.0
  resolution: 0.05
  transform_tolerance: 0.5
  plugins:
   - {name: obstacle_layer,      type: "costmap_2d::ObstacleLayer"}
   - {name: inflation_layer,     type: "costmap_2d::InflationLayer"}
```

global_frame：在局部代价地图中的全局坐标系，一般需要设置为odom_frame；
robot_base_frame：机器人本体基坐标系；
update_frequency：局部代价地图更新频率；
publish_frequency：局部代价地图发布频率；
static_map：局部代价地图一般不设置为静态地图，因为需要检测是否在机器人附近有新增的动态障碍物；
rolling_window：使用滚动窗口，始终保持机器人在当前局部地图的中心位置；
width：滚动窗口宽度，单位是米；
height：滚动窗口高度，单位是米；
resolution：地图分辨率，该分辨率可以从加载的地图相对应的配置文件中获取到；
transform_tolerance：局部代价地图中的坐标系之间转换的最大可忍受延时；
plugins：在局部代价地图中，不需要静态地图层，因为我们使用滚动窗口来不断的扫描障碍物，所以只需融合两层地图(inflation_ layer和obstacle_ layer) 即可，融合后的地图用于进行局部路径规划。

2.5 move_base_params.yaml文件

```
shutdown_costmaps: false
controller_frequency: 3.0
controller_patience: 3.0
planner_frequency: 1.0
planner_patience: 5.0
oscillation_timeout: 10.0
oscillation_distance: 0.2
base_local_planner: "dwa_local_planner/DWAPlannerROS"
base_global_planner: "navfn/NavfnROS"
```

shutdown_costmaps：当move_base在不活动状态时，是否关掉costmap；

controller_frequency：向底盘控制移动话题cmd_vel发送命令的频率；

controller_patience：在空间清理操作执行前，控制器花多长时间等有效控制下发；

planner_frequency：全局规划操作的执行频率如果设置为0.0，则全局规划器仅在接收到新的目标点或者局部规划器报告路径堵塞时才会重新执行规划操作；

planner_patience：在空间清理操作执行前，留给规划器多长时间来找出一条有效规划；

oscillation_timeout：执行修复机制前，允许振荡的时长；

oscillation_distance：来回运动在多大距离以上不会被认为是振荡；

base_local_planner：指定用于move_base的局部规划器名称；

base_global_planner：指定用于move_base的全局规划器插件名称

2.6 dwa_local_planner_params.yaml文件

```
DWAPlannerROS:
  max_vel_x: 0.25
  min_vel_x: 0.0
  max_vel_y: 0.0
  min_vel_y: 0.0
  max_trans_vel: 0.5
  min_trans_vel: 0.1
  trans_stopped_vel: 0.1
  max_rot_vel: 0.7
  min_rot_vel: 0.3
  rot_stopped_vel: 0.4
  acc_lim_x: 0.8
  acc_lim_theta: 3.5
  acc_lim_y: 0.0
# Goal Tolerance Parameters
  yaw_goal_tolerance: 0.1
  xy_goal_tolerance: 0.1
```

acc_lim_x：x方向加速度的绝对值；
acc_lim_y：y方向加速度的绝对值，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：到达目标点时偏航角允许的误差，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；

2.6 dwa_local_planner_params.yaml文件

```
# Forward Simulation Parameters
  sim_time: 1.8
  vx_samples: 6
  vy_samples: 1
  vtheta_samples: 20
# Trajectory Scoring Parameters
  path_distance_bias: 64.0
  goal_distance_bias: 24.0
  occdist_scale: 0.5
  forward_point_distance: 0.325
  stop_time_buffer: 0.2
  scaling_speed: 0.25
  max_scaling_factor: 0.2
# Oscillation Prevention Parameters
  oscillation_reset_dist: 0.05
# Debugging
  publish_traj_pc : true
  publish_cost_grid_pc: true
  global_frame_id: odom_combined
```

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；
v_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；
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；
oscillation_reset_dist：机器人运动多远距离才会重置振荡标记；
prune_plan：机器人前进时是否清除身后1m外的轨迹

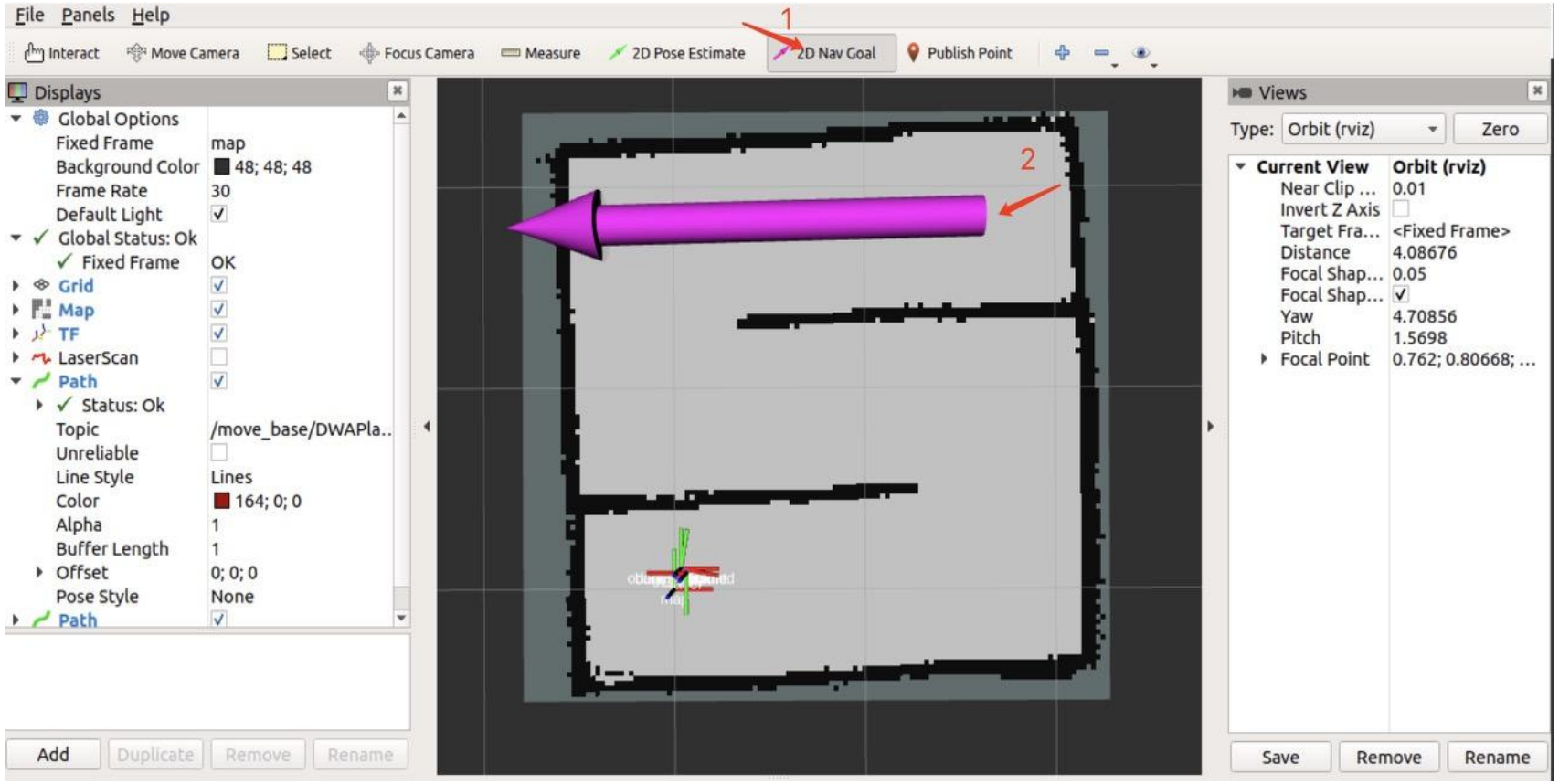
第15课 ROS导航实践

北邮移动机器人与智能技术实验室 编

3 ROS导航实验

3.1 实验过程

roslacunh mrobotit robot_start.launch //启动mRobot小车
roslaunch mrobotit navigation.launch //启动导航节点



具体参考视频：https://gitee.com/mrobotit/mrobot_book/tree/master/ch7