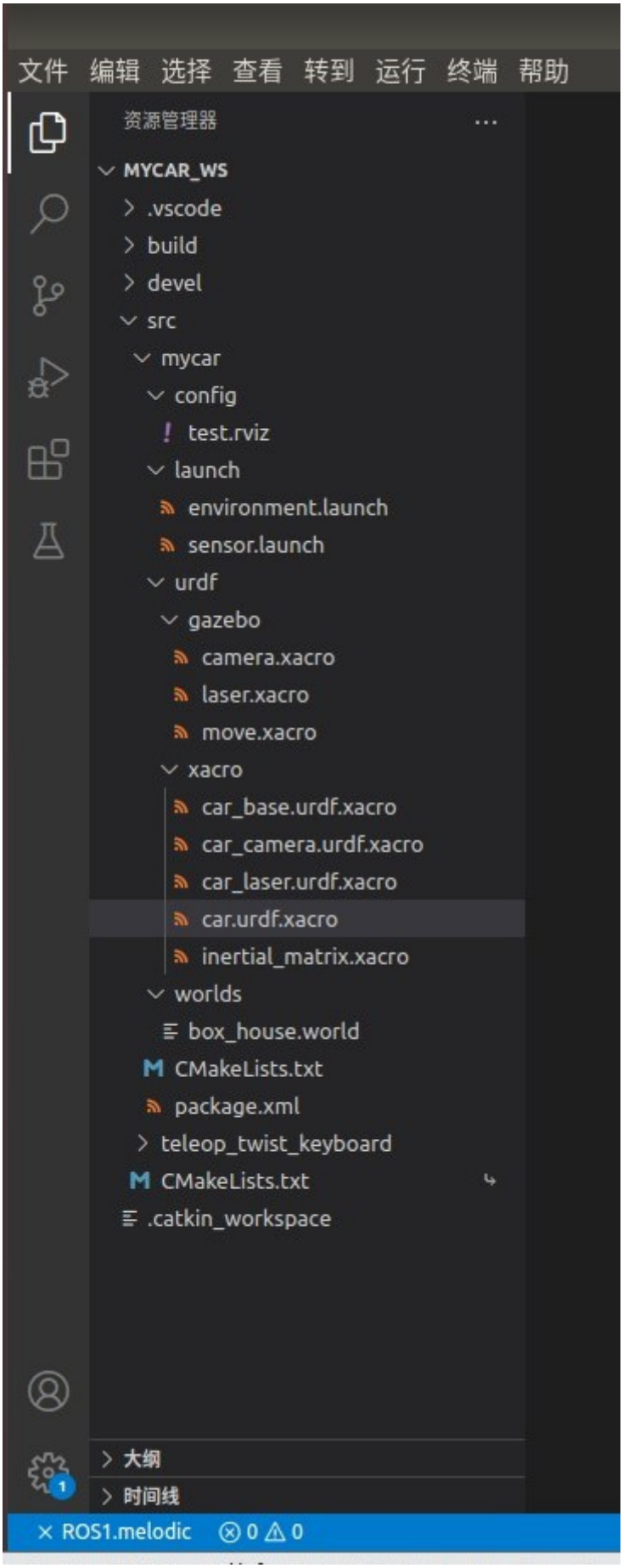
Gazebo+SLAM+rviz

地图建模仿真

1.前置模型准备

带有里程计/激光雷达/摄像头的小车和地图环境


文件 编辑 选择 查看 转到 运行 终端 帮助
资源管理器
MYCAR_WS
.vscode
build
devel
src
mycar
config
test.rviz
launch
environment.launch
sensor.launch
urdf
gazebo
camera.xacro
laser.xacro
move.xacro
xacro
car_base.urdf.xacro
car_camera.urdf.xacro
car_laser.urdf.xacro
car.urdf.xacro
inertial_matrix.xacro
worlds
box_house.world
CMakeLists.txt
package.xml
teleop_twist_keyboard
CMakeLists.txt
.catkin_workspace
大纲
时间线
ROS1.melodic
0
0

2.安装功能包

```
sudo apt install ros-melodic-map-server
sudo apt install ros-melodic-gmapping
sudo apt install ros-melodic-navigation
```

3.创建功能包

```
#使用vscode快捷创建
#包含gmapping map_server amcl move_base
```

4.运行仿真环境并保存地图

启动roscore

```
roscore
#下文省略roscore
```

environment.launch(打开Gazebo)

```
<launch>

    <param name="robot_description" command="$(find xacro)/xacro $(find
mycar)/urdf/xacro/car.urdf.xacro" />

    <include file="$(find gazebo_ros)/launch/empty_world.launch">
        <arg name="world_name" value="$(find mycar)/worlds/box_house.world" />
    </include>

    <node pkg="gazebo_ros" type="spawn_model" name="model" args="-urdf -model
mycar -param robot_description"  />
</launch>
```

```
#运行
source ./devel/setup.bash
roslaunch mycar environment.launch
```

nav01_slam.launch(运行仿真)

```
<launch>
    <!-- 仿真环境下，将该参数设置为true  -->
    <param name="use_sim_time" value="true"/>
    <!-- gmapping  -->
    <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping"
output="screen">
      <!-- 设置雷达话题  -->
      <remap from="scan" to="scan"/>

      <!-- 关键参数：坐标系 -->
      <param name="base_frame" value="base_footprint"/><!--底盘坐标系-->
      <param name="map_frame" value="map"/>
      <param name="odom_frame" value="odom"/> <!--里程计坐标系-->
```

```
      <param name="map_update_interval" value="5.0"/>
      <param name="maxUrange" value="16.0"/>
      <param name="sigma" value="0.05"/>
      <param name="kernelSize" value="1"/>
      <param name="lstep" value="0.05"/>
      <param name="astep" value="0.05"/>
      <param name="iterations" value="5"/>
      <param name="lsigma" value="0.075"/>
      <param name="ogain" value="3.0"/>
      <param name="lskip" value="0"/>
      <param name="srr" value="0.1"/>
      <param name="srt" value="0.2"/>
      <param name="str" value="0.1"/>
      <param name="stt" value="0.2"/>
      <param name="linearUpdate" value="1.0"/>
      <param name="angularUpdate" value="0.5"/>
      <param name="temporalUpdate" value="3.0"/>
      <param name="resampleThreshold" value="0.5"/>
      <param name="particles" value="30"/>
      <param name="xmin" value="-50.0"/>
      <param name="ymin" value="-50.0"/>
      <param name="xmax" value="50.0"/>
      <param name="ymax" value="50.0"/>
      <param name="delta" value="0.05"/>
      <param name="llsamplerange" value="0.01"/>
      <param name="llsamplestep" value="0.01"/>
      <param name="lasamplerange" value="0.005"/>
      <param name="lasamplestep" value="0.005"/>

    </node>

    <node pkg="joint_state_publisher" name="joint_state_publisher"
type="joint_state_publisher" />
    <node pkg="robot_state_publisher" name="robot_state_publisher"
type="robot_state_publisher" />

    <node pkg="rviz" type="rviz" name="rviz" />
    <!-- 可以保存 rviz 配置并后期直接使用-->
    <!--
    <node pkg="rviz" type="rviz" name="rviz" args="-d $(find
my_nav_sum)/rviz/gmapping.rviz"/>
    -->
</launch>
```

```
#运行
source ./devel/setup.bash
roslaunch mycar nav01_slam.launch
#设置Robort Model,map,TF,Laserscan并保存rviz设置
```

通过键盘控制小车移动

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py  _speed:=0.5 _turn:=0.5
```

nav02_map_save.launch（保存地图）

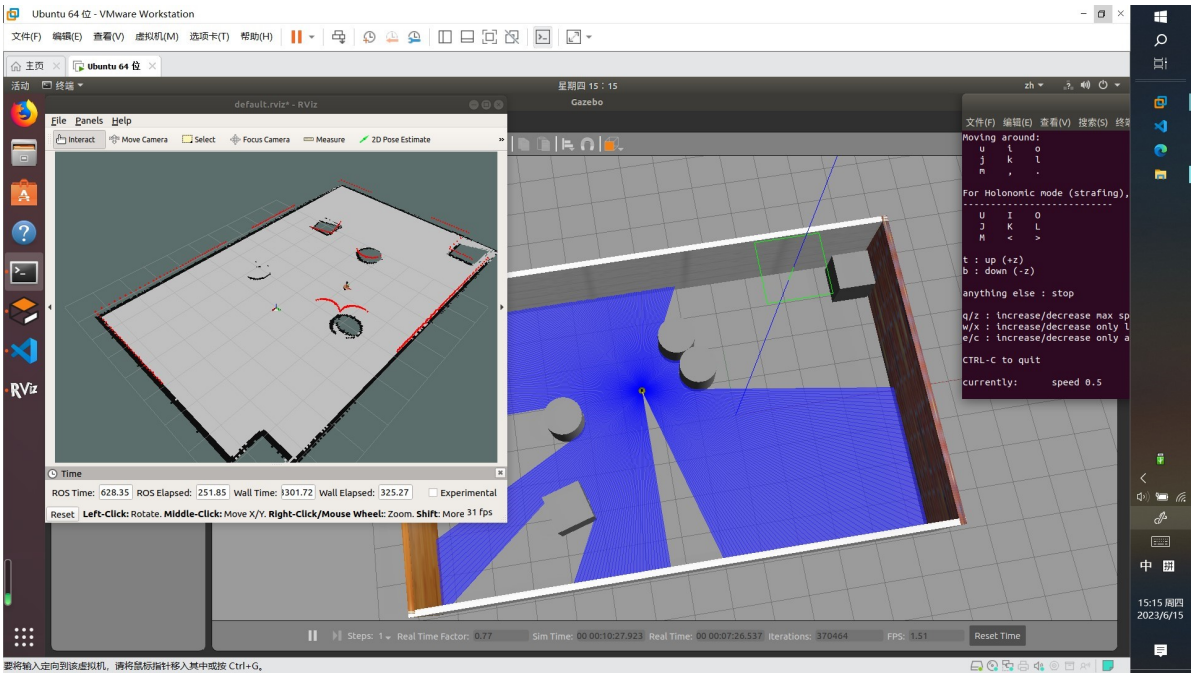
```
<launch>
    <arg name="filename" value="$(find nav_demo)/map/nav" />
    <node name="map_save" pkg="map_server" type="map_saver" args="-f $(arg
filename)" />
</launch>
```

```
#运行
source ./devel/setup.bash
roslaunch mycar nav02_map_save.launch
```

结果

与下文验证地图时使用的不是一个地图

（龙卷小车摧毁圆柱地图）

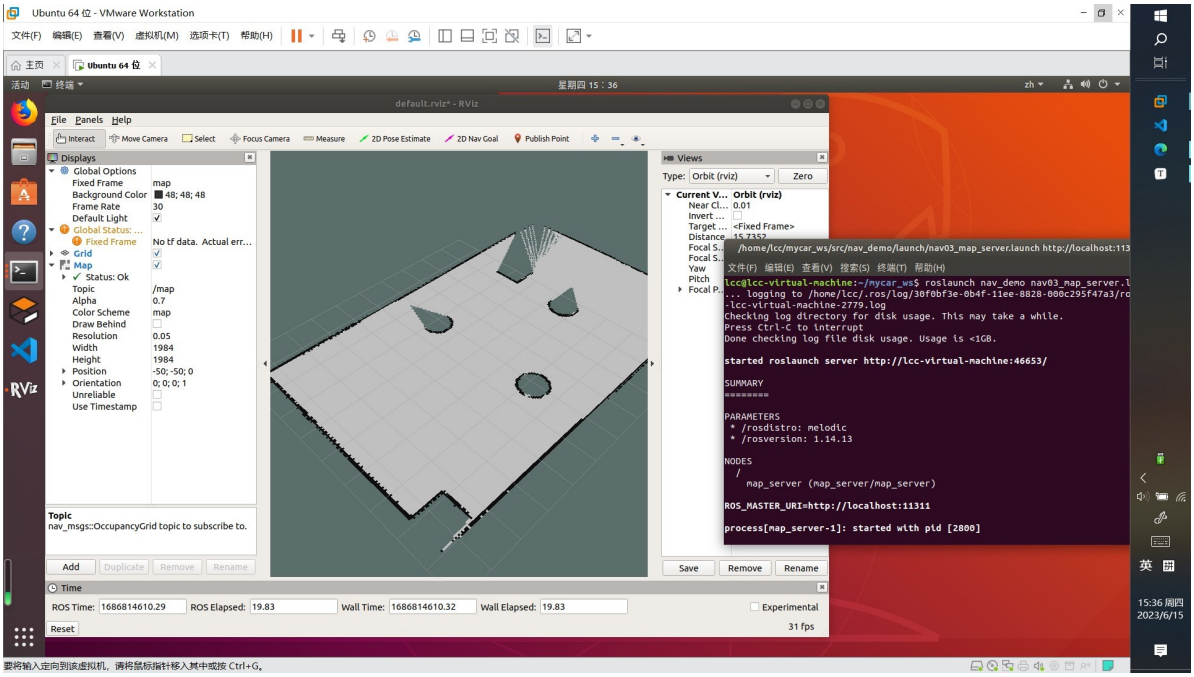


5.读取地图进行验证

nav03_map_server.launch（读取地图）

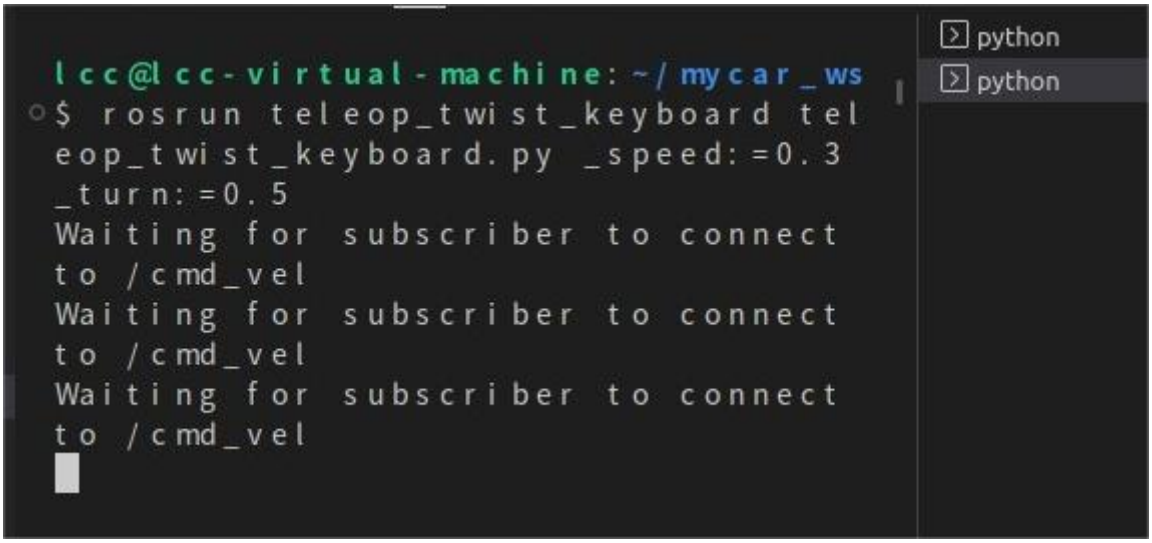
```
<launch>

    <!-- 设置地图的配置文件 -->
    <arg name="map" default="nav.yaml" />

    <!-- 运行地图服务器，并且加载设置的地图-->
    <node name="map_server" pkg="map_server" type="map_server" args="$(find
nav_demo)/map/$(arg map)"/>

</launch>
```

```
#下文起省略启动过程（roslaunch）
```



注意事项

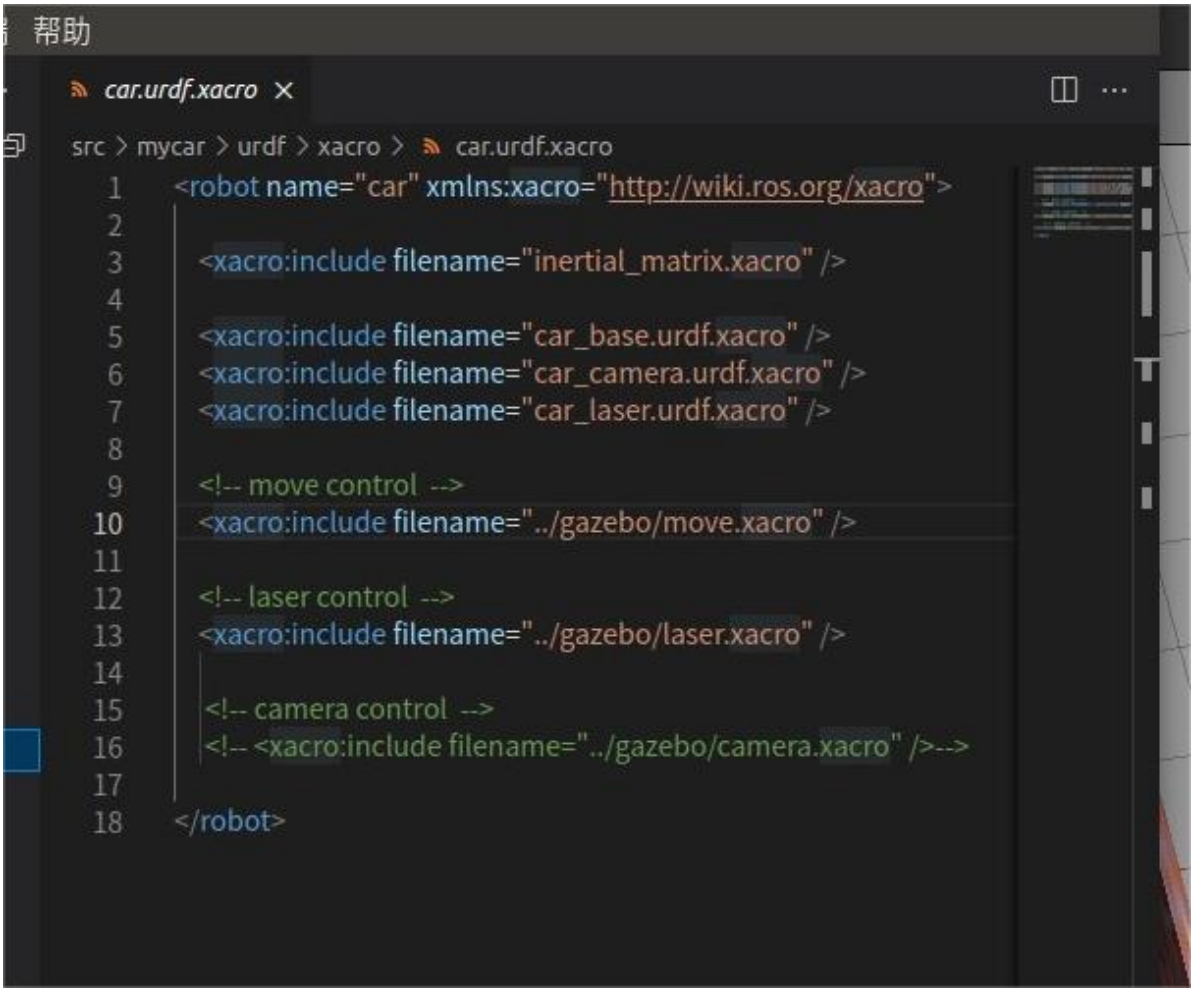
问题A



原因：小车的urdf/xacro文件中未启动里程计使move_base发送消息的节点出错

解决方式：

修改小车的urdf/xacro文件



定位仿真

```
#利用amcl在rviz中完成定位仿真
```

1.添加amcl

nav04_amcl.launch

```
<launch>
    <node pkg="amcl" type="amcl" name="amcl" output="screen">
        <!-- Publish scans from best pose at a max of 10 Hz -->
        <param name="odom_model_type" value="diff"/>
        <param name="odom_alpha5" value="0.1"/>
        <param name="transform_tolerance" value="0.2" />
        <param name="gui_publish_rate" value="10.0"/>
        <param name="laser_max_beams" value="30"/>
        <param name="min_particles" value="500"/>
        <param name="max_particles" value="5000"/>
        <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="0.8"/>
        <param name="odom_alpha4" value="0.2"/>
```

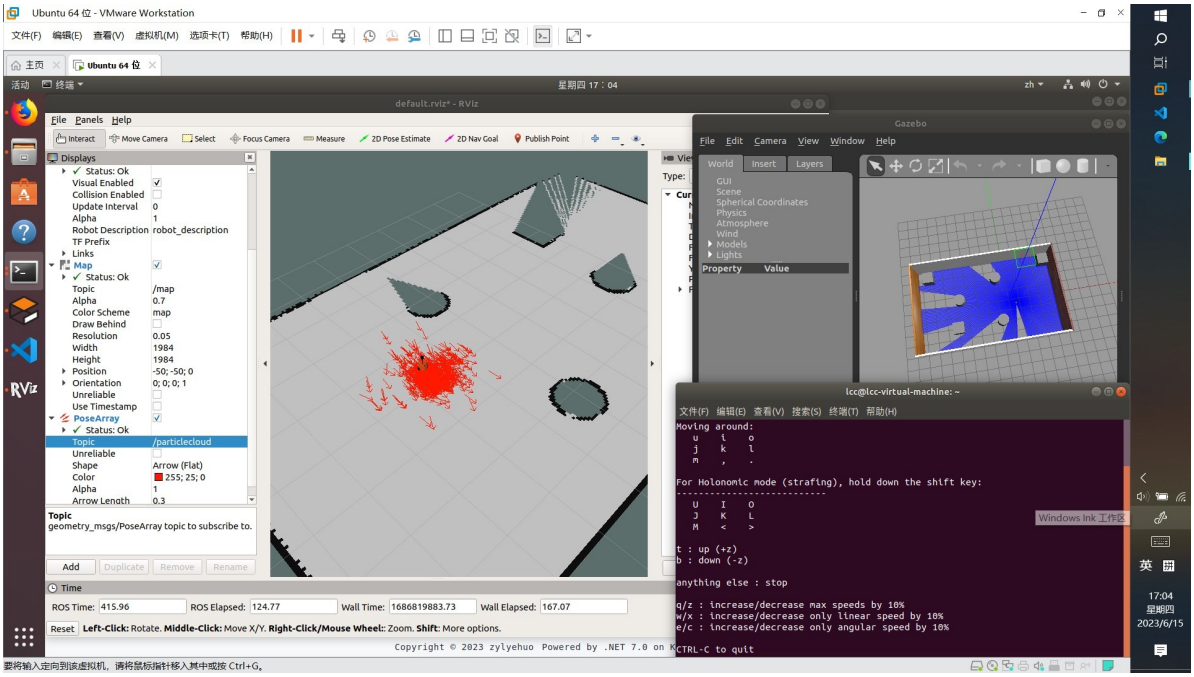
```
        <param name="laser_z_hit" value="0.5"/>
        <param name="laser_z_short" value="0.05"/>
        <param name="laser_z_max" value="0.05"/>
        <param name="laser_z_rand" value="0.5"/>
        <param name="laser_sigma_hit" value="0.2"/>
        <param name="laser_lambda_short" value="0.1"/>
        <param name="laser_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="0.2"/>
        <param name="update_min_a" value="0.5"/>

        <!--  set coordinate system: odom、map、base_link  -->
        <param name="odom_frame_id" value="odom"/>
        <param name="base_frame_id" value="base_footprint"/>

        <param name="resample_interval" value="1"/>
        <param name="transform_tolerance" value="0.1"/>
        <param name="recovery_alpha_slow" value="0.0"/>
        <param name="recovery_alpha_fast" value="0.0"/>
    </node>
</launch>
```

2.编写launch启动测试

test_amcl.launch(启动rviz调用amcl)

```
<!-- 测试文件 -->
<launch>

    <!-- 启动 rviz -->
    <node pkg="joint_state_publisher" name="joint_state_publisher"
type="joint_state_publisher" />
    <node pkg="robot_state_publisher" name="robot_state_publisher"
type="robot_state_publisher" />
    <node pkg="rviz" type="rviz" name="rviz"/>

    <!-- 加载地图服务 -->
    <include file="$(find nav_demo)/launch/nav03_map_server.launch"  />

    <!-- 启动 amcl 节点 -->
    <include file="$(find nav_demo)/launch/nav04_amcl.launch"  />

</launch>
```

```
#rviz中配置PoseArray，Map，Robort Model
```



运动规划和仿真

1.配置Costmap（yaml）

base_local_planner_params

```
TrajectoryPlannerROS:

# Robot Configuration Parameters
  max_vel_x: 0.5 # X 方向最大速度
  min_vel_x: 0.1 # X 方向最小速速

  max_vel_theta:  1.0 #
  min_vel_theta: -1.0
  min_in_place_vel_theta: 1.0

  acc_lim_x: 1.0 # X 加速限制
  acc_lim_y: 0.0 # Y 加速限制
  acc_lim_theta: 0.6 # 角速度加速限制

# Goal Tolerance Parameters，目标公差
  xy_goal_tolerance: 0.10
  yaw_goal_tolerance: 0.05

# Differential-drive robot configuration
# 是否是全向移动机器人
  holonomic_robot: false

# Forward Simulation Parameters，前进模拟参数
  sim_time: 0.8
  vx_samples: 18
  vtheta_samples: 20
  sim_granularity: 0.05
```

costmap_common_params

```
#机器人几何参，如果机器人是圆形，设置 robot_radius,如果是其他形状设置 footprint
robot_radius: 0.12 #圆形
# footprint: [[-0.12, -0.12], [-0.12, 0.12], [0.12, 0.12], [0.12, -0.12]] #其他形状

obstacle_range: 3.0 # 用于障碍物探测，比如: 值为 3.0，意味着检测到距离小于 3 米的障碍物时，就会引入代价地图
raytrace_range: 3.5 # 用于清除障碍物，比如：值为 3.5，意味着清除代价地图中 3.5 米以外的障碍物


#膨胀半径，扩展在碰撞区域以外的代价区域，使得机器人规划路径避开障碍物
inflation_radius: 0.2
#代价比例系数，越大则代价值越小
cost_scaling_factor: 3.0

#地图类型
map_type: costmap
#导航包所需要的传感器
observation_sources: scan
#对传感器的坐标系和数据进行配置。这个也会用于代价地图添加和清除障碍物。例如，你可以用激光雷达传感器用于在代价地图添加障碍物，再添加kinect用于导航和清除障碍物。
scan: {sensor_frame: laser, data_type: LaserScan, topic: scan, marking: true, clearing: true}
```

global_costmap_params

```
global_costmap:
  global_frame: map #地图坐标系
  robot_base_frame: base_footprint #机器人坐标系
  # 以此实现坐标变换

  update_frequency: 1.0 #代价地图更新频率
  publish_frequency: 1.0 #代价地图的发布频率
  transform_tolerance: 0.5 #等待坐标变换发布信息的超时时间

  static_map: true # 是否使用一个地图或者地图服务器来初始化全局代价地图，如果不使用静态地图，这个参数为false.
```

local_costmap_params

```
local_costmap:
  global_frame: odom #里程计坐标系
  robot_base_frame: base_footprint #机器人坐标系

  update_frequency: 10.0 #代价地图更新频率
  publish_frequency: 10.0 #代价地图的发布频率
  transform_tolerance: 0.5 #等待坐标变换发布信息的超时时间

  static_map: false  #不需要静态地图，可以提升导航效果
  rolling_window: true #是否使用动态窗口，默认为false，在静态的全局地图中，地图不会变化
  width: 3 # 局部地图宽度 单位是 m
  height: 3 # 局部地图高度 单位是 m
  resolution: 0.05 # 局部地图分辨率 单位是 m，一般与静态地图分辨率保持一致
```

2.编写launch文件启动rviz

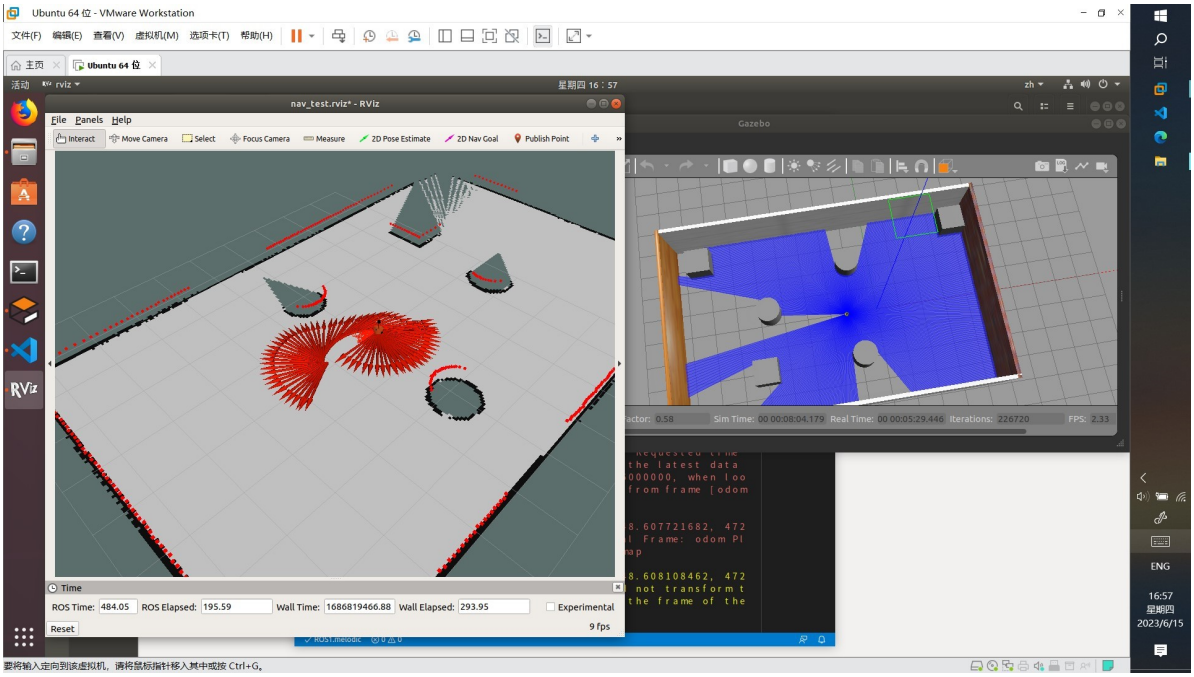
test.launch

```
<!-- 集成导航相关的 launch 文件 -->
<launch>

    <!-- 地图服务 -->
    <include file="$(find nav_demo)/launch/nav03_map_server.launch" />

    <!-- 启动AMCL节点 -->
    <include file="$(find nav_demo)/launch/nav04_amcl.launch" />

    <!-- 运行move_base节点 -->
    <include file="$(find nav_demo)/launch/nav05_path.launch" />

    <!-- 运行rviz -->
    <node pkg="joint_state_publisher" name="joint_state_publisher"
type="joint_state_publisher" />
    <node pkg="robot_state_publisher" name="robot_state_publisher"
type="robot_state_publisher" />
    <node pkg="rviz" type="rviz" name="rviz" />

</launch>
```

3.启动软件

```
#运行roscore，environment.launch，test.launch

#在rviz中加入Robort Model，Map，LaserScan，Odometry参数保存
```

4.运行仿真


Ubuntu 64 位 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
主页
Ubuntu 64 位
活动
rviz
星期四 16：57
nav_test.rviz* - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Time
ROS Time: 484.05 ROS Elapsed: 195.59 Wall Time: 1686819466.88 Wall Elapsed: 293.95 Experimental
Reset
9 fps
Gazebo
Sim Time: 00 00:08:04.179 Real Time: 00 00:05:29.446 Iterations: 226720 FPS: 2.33
ENG
16:57
星期四
2023/6/15
要将输入定向到该虚拟机，请将鼠标指针移入其中或按 Ctrl+G。