大连理工大学
Dalian University Of Technology

智能机器人实验

——SLAM与自主导航仿真

李华龙

大连理工大学 Dalian University Of Technology

课程主要内容

1. 初识ROS

- ROS简介
- ROS的安装
- 第1个ROS例程
- ROS总体架构

2. ROS基础

- ROS工作空间
- ROS通信编程
- ROS分布式通信
- ROS关键组件

3. 建模与仿真

- 机器人系统简介
- 机器人URDF建模
- 机器人模型优化
- Gazebo物理仿真

4. SLAM与自主导航仿真

- 相关基础概念介绍
- SLAM功能包的应用
- ROS中的导航框架
- 综合仿真实现

5. SLAM与自主导航实践

- 机器人实验平台了解与使用
- 真实机器人SLAM测试与验证
- 真实机器人自主导航实现
- 综合测试与验证

大连理工大学
Dalian University Of Technology

内容概览

1. 相关基础概念
2. SLAM功能包的应用
3. ROS中的导航框架
4. 综合仿真实现

1 相关基础概念

相关基础概念

机器人定位与导航[1]

- 室内机器人需要解决的关键问题之一
- 主要应用场景：

  室内机器人在未知环境内，以**最优路径**完成从A点到B点的移动
- 三个主要方面：

  地图精确构建 -> **SLAM**（即时定位与地图构建，基础技术）

  类比：人在黑暗中探索未知环境

  机器人准确定位 -> **amcl**（自适应蒙特卡洛定位方法，功能包）

  路径实时规划 -> **move_base**（路径规划，功能包）

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

相关基础概念

机器人定位与导航

- 准备条件：

① 差速驱动的轮式机器人，可通过 Twist 类型消息控制运动

② 环境深度信息（激光雷达，RGB-D）

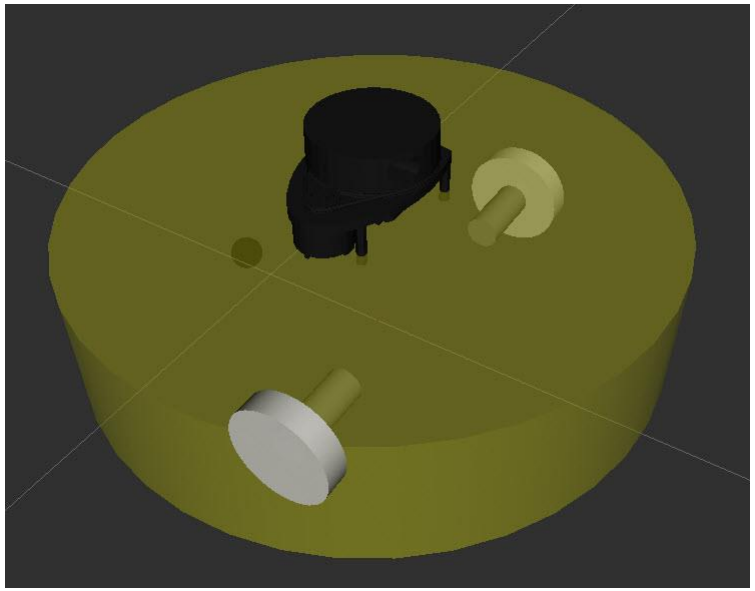
③ 里程计信息*

④ 外形是正方形或圆形的机器人为佳

⑤ 测试（仿真）环境

相关基础概念

差速轮式机器人



```
$ rosmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

相关基础概念

环境深度信息

```
$ rosmsg show sensor_msgs/LaserScan
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

相关基础概念

里程计信息

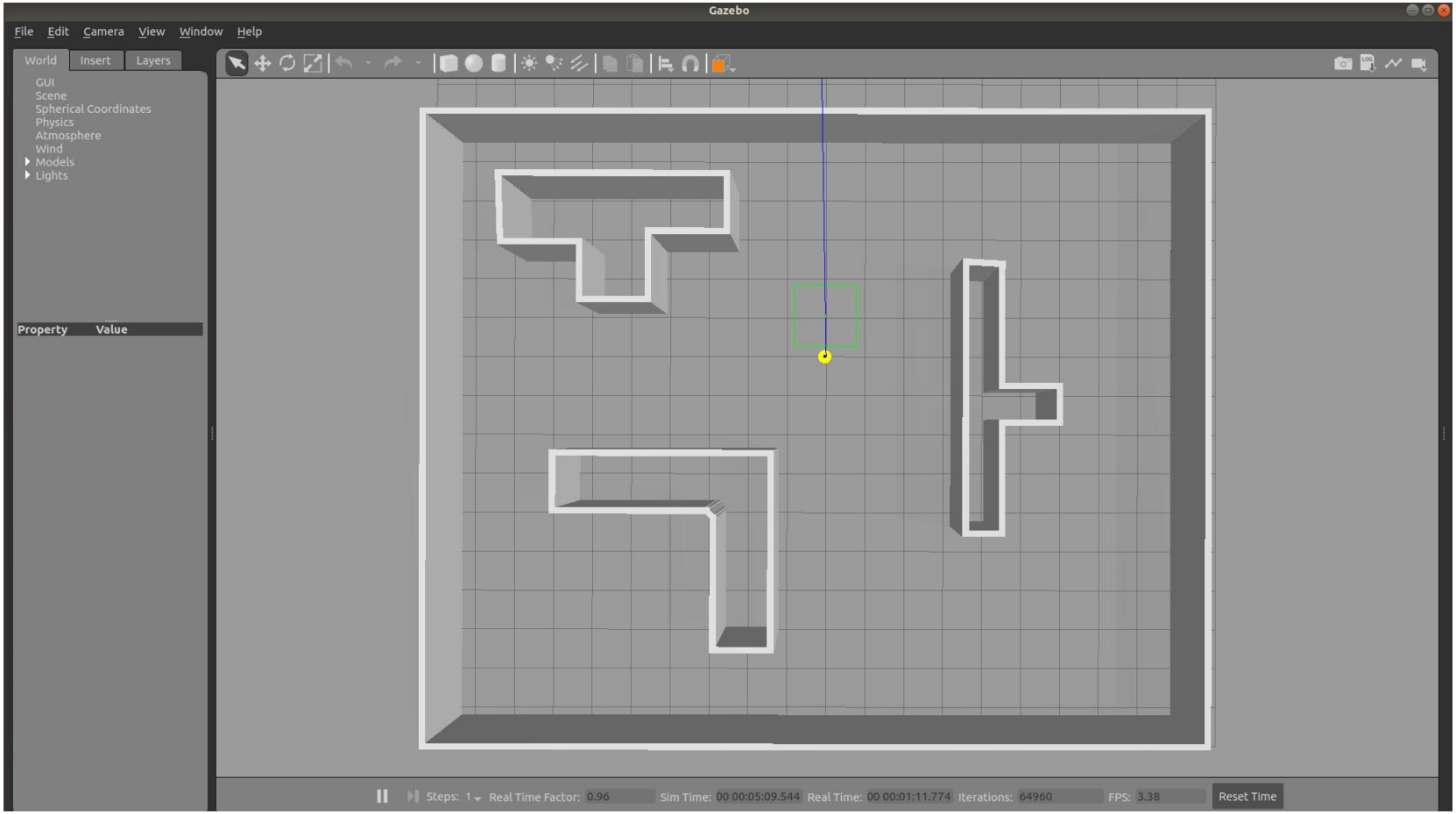
```
$ rosmsg show nav_msgs/Odometry
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
  float64[36] covariance
geometry_msgs/TwistWithCovariance twist
  geometry_msgs/Twist twist
    geometry_msgs/Vector3 linear
      float64 x
      float64 y
      float64 z
    geometry_msgs/Vector3 angular
      float64 x
      float64 y
      float64 z
  float64[36] covariance
```

位

姿

相关基础概念

测试（仿真）环境



相关基础概念

测试（仿真）环境（Gazebo）

- Building Editor
- 直接插入模型（提前下载并放在安装目录指定文件夹内）
- 保存 world 文件时可能会遇到 bug：操作后界面不刷新

  解决办法：每操作1次，最小化窗口，再恢复窗口
- 修改启动 Gazebo 并加载机器人模型的 launch 文件

```
<include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="debug" value="$(arg debug)" />
    <arg name="gui" value="$(arg gui)" />
    <arg name="paused" value="$(arg paused)"/>
    <arg name="use_sim_time" value="$(arg use_sim_time)"/>
    <arg name="headless" value="$(arg headless)"/>
    <arg name="world_name" value="$(find ros_2dnav_learning)/world/room1/room1.world"/>
</include>
```

相关基础概念

- 可安装 teleop_twist_keyboard[1] 功能包，并在 launch 文件中添加如下内容，自动启动该速度控制节点：

```
$ sudo apt-get install ros-melodic-teleop-twist-keyboard
```

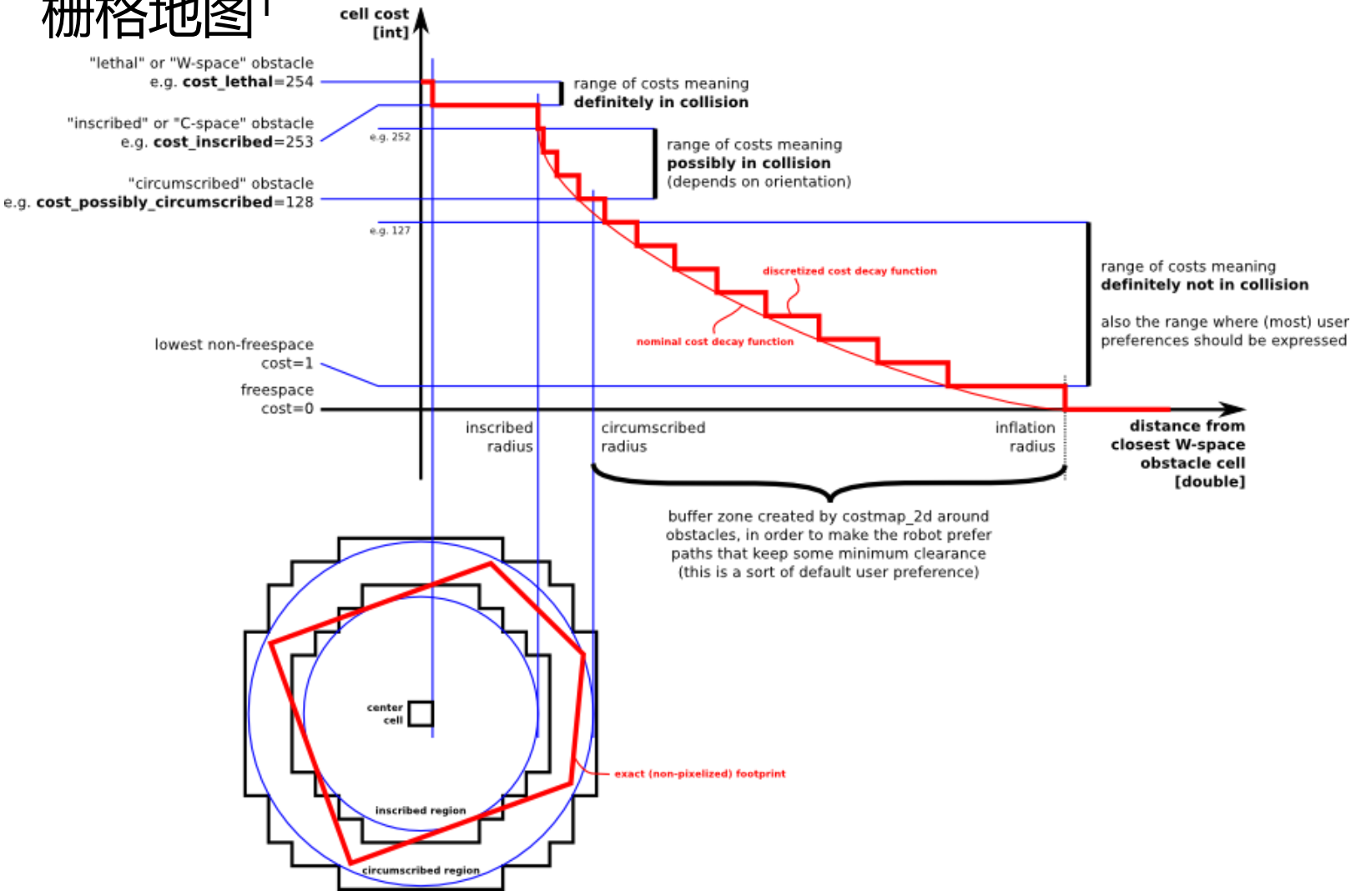
```
<node name="teleop_twist_keyboard" pkg="teleop_twist_keyboard"
      type="teleop_twist_keyboard.py"/>
```

```
$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py

Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
```

1. https://index.ros.org/p/teleop_twist_keyboard/github-ros-teleop-teleop_twist_keyboard/#melodic

相关基础概念

栅格地图[1]



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

准备工作

准备工作

- 安装相关功能包：

  gmapping: $ *sudo apt install ros-melodic-gmapping*

  map_server: $ *sudo apt install ros-melodic-map-server*

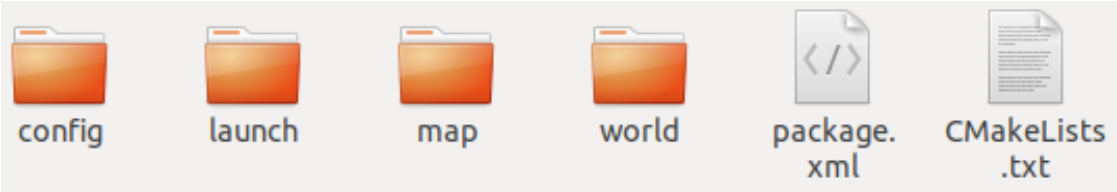
  navigation: $ *sudo apt install ros-melodic-navigation*

- 新建功能包：

  $ *catkin_create_pkg* ***ros_2dnav_learning***

  gmapping map_server amcl move_base

- 新功能包内创建文件夹：



2 SLAM功能包的应用

SLAM功能包的应用

SLAM：

- Simultaneous Localization and Mapping（即时定位与地图构建）
- ROS中实现，常用的功能包：

  gmapping[1]：激光雷达+里程计

  hector_slam[2]：激光雷达

  cartographer[3, 4]：激光雷达（里程计，IMU），2D或3D

1. http://wiki.ros.org/gmapping/
2. http://wiki.ros.org/hector_slam
3. http://wiki.ros.org/cartographer
4. https://google-cartographer-ros.readthedocs.io/en/latest/

SLAM功能包的应用

gmapping[1]：

- ROS 中较常用且比较成熟的 SLAM 算法之一
- 可根据移动机器人的**激光雷达**和**里程计**数据来绘制二维的**栅格地图**
- 订阅的话题：

  tf (tf/tfMessage)：关联雷达、底盘与里程计之间的坐标变换

  scan(sensor_msgs/LaserScan)：构建地图所需的雷达信息

- 发布的话题：

  map_metadata(nav_msgs/MapMetaData)：地图元数据，包括地图的宽度、高度、分辨率，周期性更新，可 RViz 显示

  map(nav_msgs/OccupancyGrid)：地图栅格数据，周期性更新

- 提供的服务：

  dynamic_map(nav_msgs/GetMap)：用于获取地图数据

1. http://wiki.ros.org/gmapping/

SLAM功能包的应用

gmapping[1]：

- 主要参数：

  ~base_frame(string, default:“base_link”)：机器人基坐标系

  ~map_frame(string, default:“map”)：地图坐标系

  ~odom_frame(string, default:“odom”)：里程计坐标系

  ~map_update_interval(float, default: 5.0)：地图更新频率

  ~maxUrange(float, default: 80.0)：激光探测的最大可用范围

  ~maxRange(float)：激光探测的最大范围

  其余更多参数参阅 ROS Wiki

- 坐标变换：

  需要的：雷达坐标系 -> 基坐标系，基坐标系 -> 里程计坐标系

  发布的：地图坐标系 -> 里程计坐标系

1. http://wiki.ros.org/gmapping/

SLAM功能包的应用

gmapping 使用：

- 创建相关 launch 文件[1]
- 启动之前创建的 Gazebo 仿真环境（含机器人模型）
- 运行刚创建的 gmapping.launch 文件
- 运行键盘控制节点*：

  rosrun teleop_twist_keyboard teleop_twist_keyboard.py
- 可将上述过程写成 1 个 launch 文件

1. https://github.com/ros-perception/slam_gmapping/blob/melodic-devel/gmapping/launch/slam_gmapping_pr2.launch

SLAM功能包的应用

```
<launch>
    <arg name="scan_topic" default="/scan" />

    <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen" clear_params="true">
        <param name="base_frame" value="base_footprint"/>
        <param name="odom_frame" value="odom"/>
        <param name="map_update_interval" value="5.0"/>
        <param name="maxRange" value="20.0"/>
        <param name="maxUrange" value="8.0"/>
        <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="0.01"/>
        <param name="srt" value="0.02"/>
        <param name="str" value="0.01"/>
        <param name="stt" value="0.02"/>
        <param name="linearUpdate" value="0.5"/>
        <param name="angularUpdate" value="0.436"/>
        <param name="temporalUpdate" value="-1.0"/>
        <param name="resampleThreshold" value="0.5"/>
        <param name="particles" value="80"/>
        <param name="xmin" value="-1.0"/>
        <param name="ymin" value="-1.0"/>
        <param name="xmax" value="1.0"/>
        <param name="ymax" value="1.0"/>
        <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"/>
        <remap from="/scan" to="$(arg scan_topic)"/>
    </node>
</launch>
```

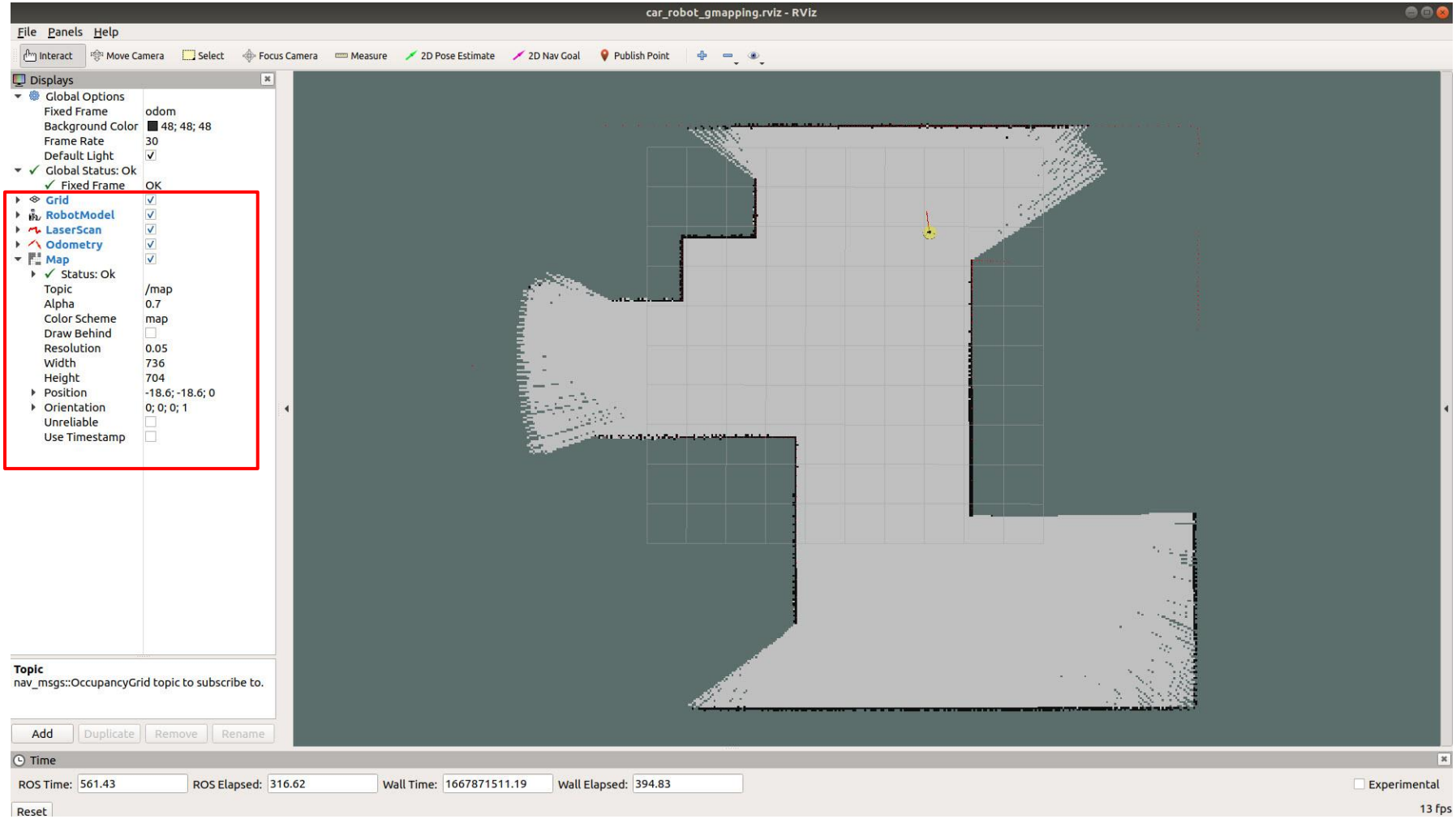
理解各参数含义的前提下，对其进行适当调整

gmapping.launch

1. https://github.com/ros-perception/slam_gmapping/blob/melodic-devel/gmapping/launch/slam_gmapping_pr2.launch

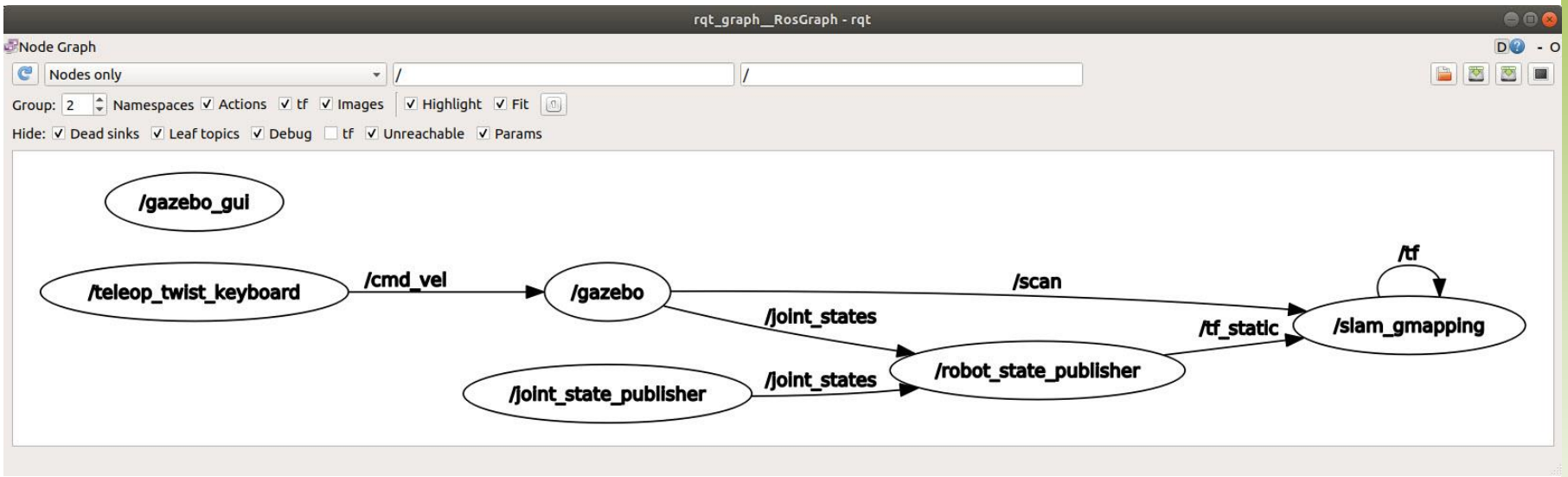
大连理工大学 Dalian University Of Technology

SLAM功能包的应用



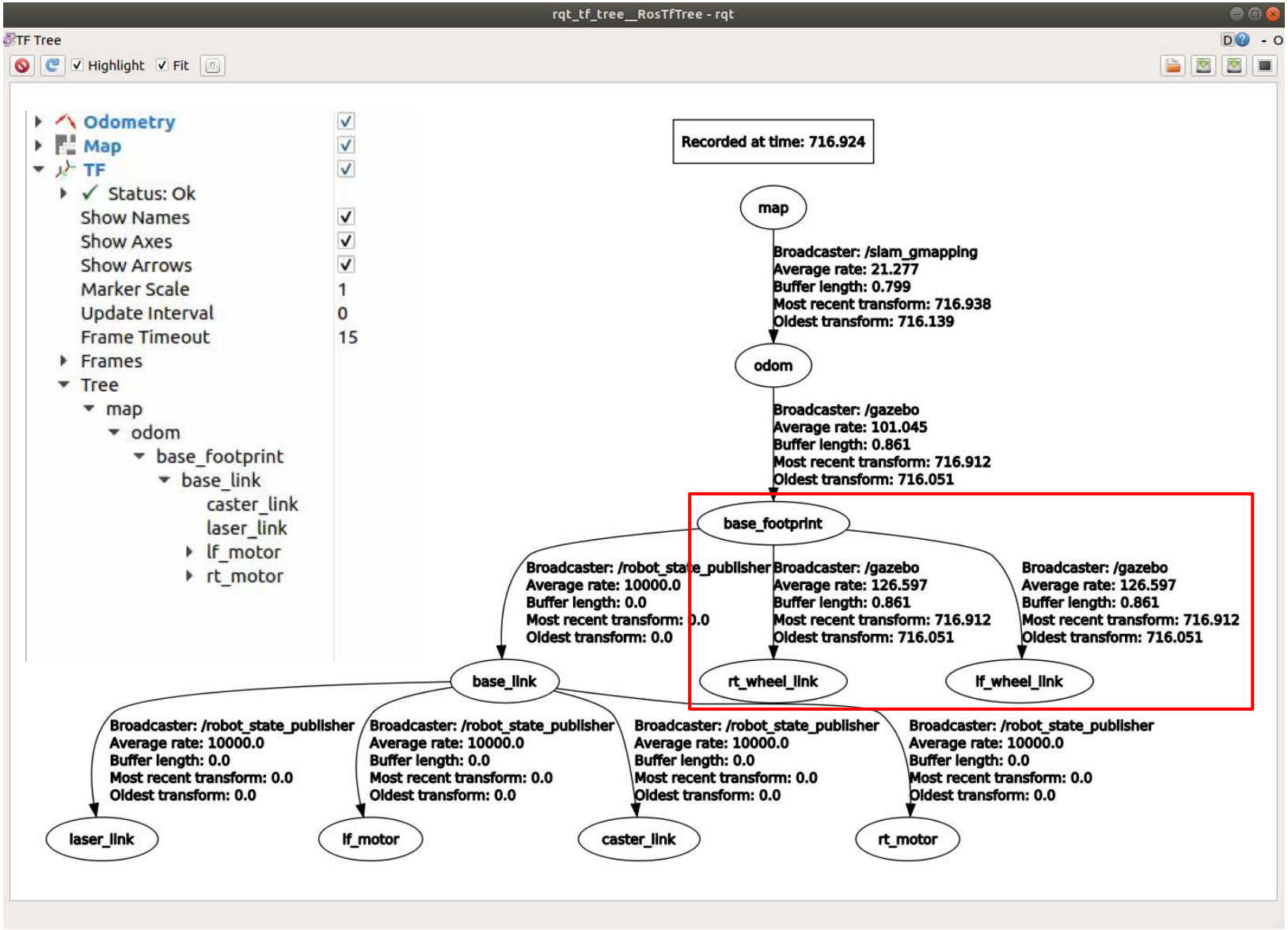
SLAM功能包的应用

```
$ rqt_graph
$ rosrun rqt_tf_tree rqt_tf_tree
```



大连理工大学 Dalian University Of Technology

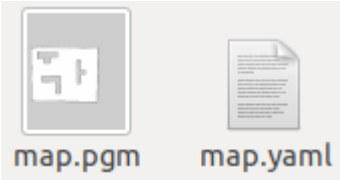
SLAM功能包的应用

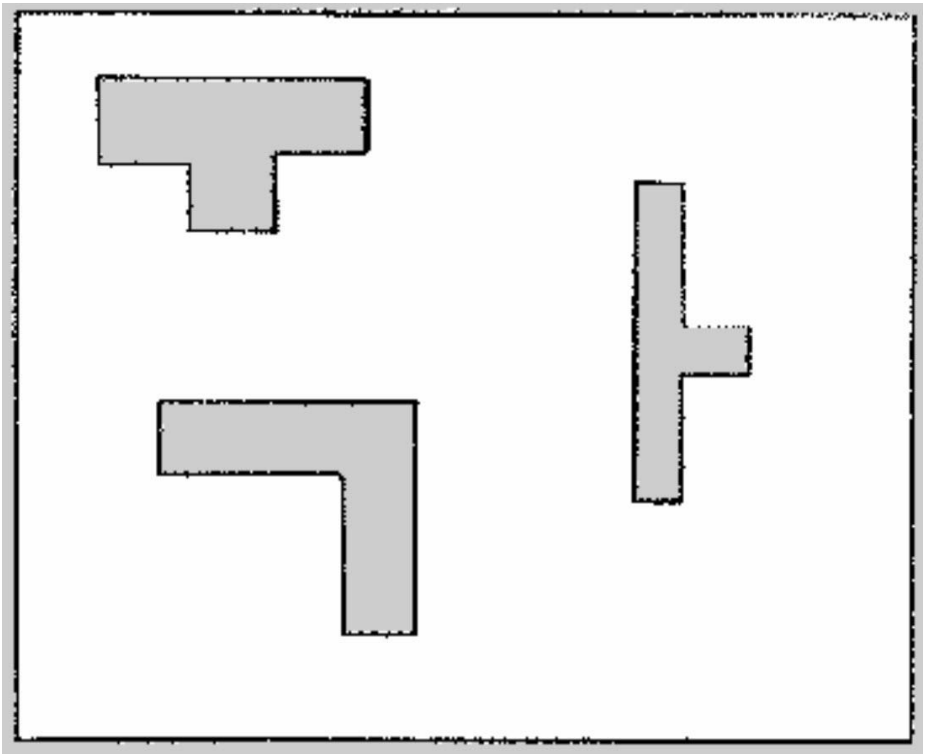


SLAM功能包的应用

- 保存地图[1]：

```
$ rosrun map_server map_saver
[ INFO] [1667872577.600411091]: Waiting for the map
[ INFO] [1667872577.830813607, 1388.101000000]: Received a 736 X 704 map @ 0.050 m/pix
[ INFO] [1667872577.830867829, 1388.101000000]: Writing map occupancy data to map.pgm
[ INFO] [1667872577.864128146, 1388.137000000]: Writing map occupancy data to map.yaml
[ INFO] [1667872577.866484023, 1388.139000000]: Done
```





1. http://wiki.ros.org/map_server

SLAM功能包的应用

- **map.yaml** 文件

  image：图片路径，绝对路径或相对路径均可

  resolution：图片分片率(单位: m/像素)

  negate：是否应该反色

  origin：地图中左下像素的二维姿态

  occupied_thresh：占用概率大于此阈值的像素被视为完全占用

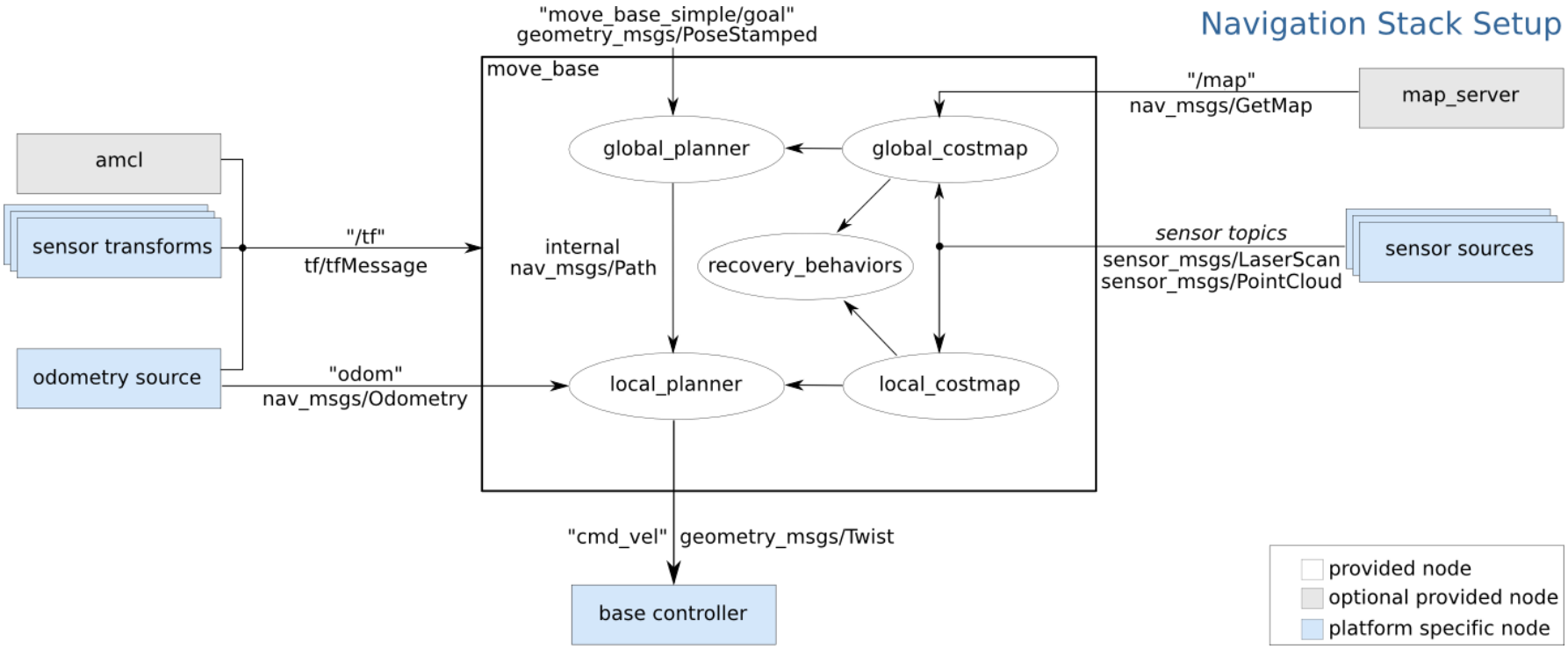
  free_thresh：占用率小于此阈值的像素被视为完全空闲

```
image: map.pgm
resolution: 0.050000
origin: [-18.600000, -18.600000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

3 ROS中的导航框架

ROS中的导航框架[1]

基于 **move_base** 的导航框架

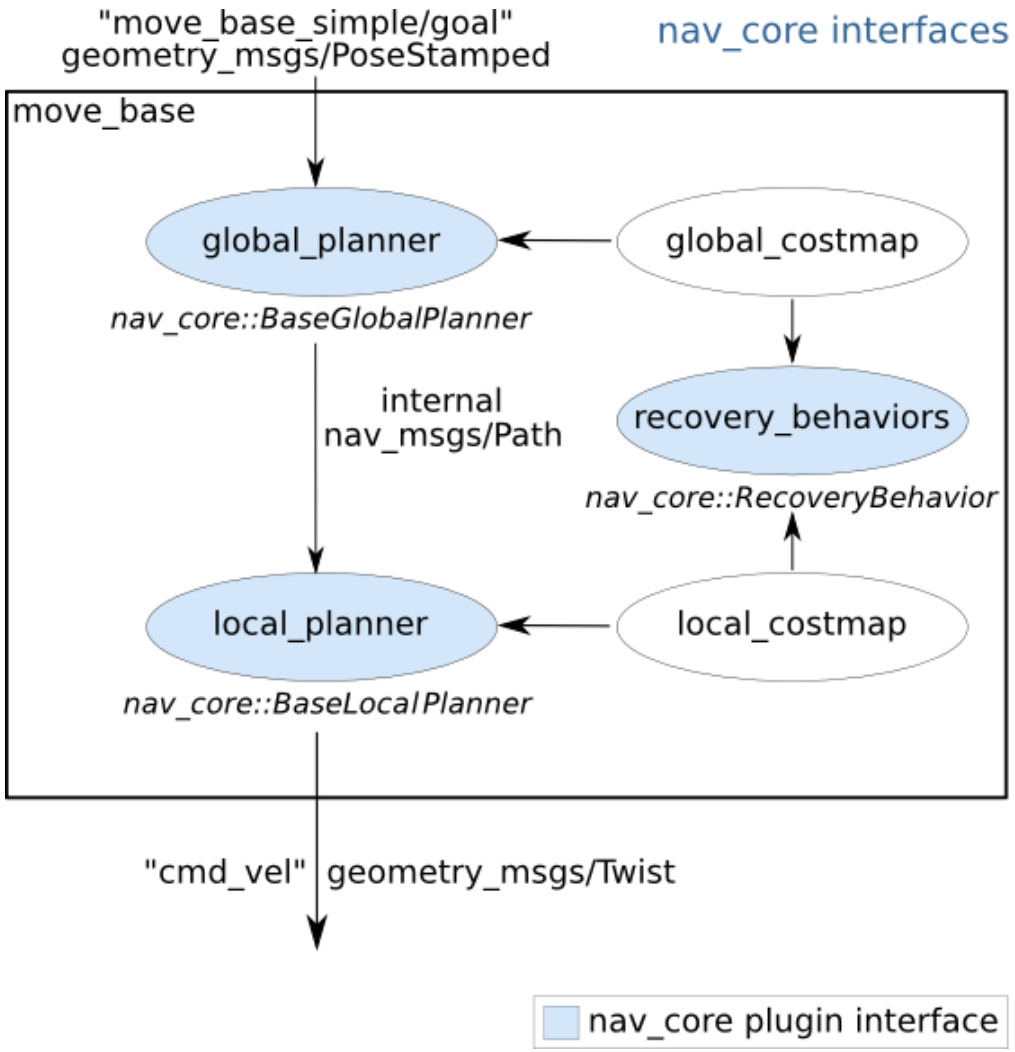


1. http://wiki.ros.org/move_base?distro=melodic

ROS中的导航框架

move_base[1]

- 全局路径规划
  - global_planner
  - navfn
  - carrot_planner
- 本地实时规划
  - 对机器人每个控制周期内的线速度、角速度进行规划，使运动路径尽量符合全局最优路径
  - 实时避障
  - base_local_planner
  - dwa_local_planner
  - teb_local_planner



1. http://wiki.ros.org/nav_core?distro=melodic

ROS中的导航框架

move_base[1]

- 提供了基于动作(action)的路径规划实现
- 在机器人运动过程中连续反馈机器人的姿态和目标点的状态信息
- 动作(话题)订阅：

  move_base/goal(move_base_msgs/MoveBaseActionGoal)：
  move_base 的运动规划目标。

  move_base/cancel(actionlib_msgs/GoalID)：取消目标
- 动作(话题)发布：

  move_base/feedback(move_base_msgs/MoveBaseActionFeedback)：
  包含机器人底盘坐标的连续反馈的信息

  move_base/status(actionlib_msgs/GoalStatusArray)：
  发送到 move_base 的目标状态信息

  move_base/result(move_base_msgs/MoveBaseActionResult)：
  操作结果(此处为空)

1. http://wiki.ros.org/move_base?distro=melodic

ROS中的导航框架

move_base[1]

- 订阅的话题

  move_base_simple/goal(geometry_msgs/PoseStamped)：

  运动规划目标(没有连续反馈，无法跟踪执行状态)

- 发布的话题

  cmd_vel(geometry_msgs/Twist)：发送到机器人底盘的运动控制消息

- 服务

  ~make_plan(nav_msgs/GetPlan)：

  获取给定目标的规划路径，但并不执行该路径规划

  ~clear_unknown_space(std_srvs/Empty)：

  允许用户直接清除机器人周围的未知空间

  ~clear_costmaps(std_srvs/Empty)：

  允许清除代价地图中的障碍物，可能会导致碰撞，慎用

- 参数：参阅 ROS Wiki

1. http://wiki.ros.org/move_base?distro=melodic

ROS中的导航框架

amcl[1]

- 自适应蒙特卡洛定位（Adaptive Monte Carlo Localization）
- 用于2D移动机器人的概率定位系统，可根据已有地图使用粒子滤波器推算机器人位姿，再结合里程计提高定位精度
- 已经被集成到了 navigation 包
- 订阅的话题：

  scan(sensor_msgs/LaserScan)：激光雷达数据
  tf(tf/tfMessage)：坐标变换消息
  initialpose(geometry_msgs/PoseWithCovarianceStamped)：
  用来初始化粒子滤波器的均值和协方差
  map(nav_msgs/OccupancyGrid)：获取地图数据

1. http://wiki.ros.org/amcl?distro=melodic

ROS中的导航框架

amcl[1]

- 发布的话题：
  - amcl_pose(geometry_msgs/PoseWithCovarianceStamped)：
    机器人在地图中的位姿估计
  - particlecloud(geometry_msgs/PoseArray)：
    位姿估计集合，RViz 中可以被 PoseArray 订阅并图形化显示
  - tf(tf/tfMessage)：发布从 odom 到 map 的转换
- 提供的服务：
  - global_localization(std_srvs/Empty)：初始化全局定位
  - request_nomotion_update(std_srvs/Empty)：
    手动执行更新和发布更新的粒子
  - set_map(nav_msgs/SetMap)：手动设置新地图和姿态
- 调用的服务：
  - static_map(nav_msgs/GetMap)：获取地图数据

1. http://wiki.ros.org/amcl?distro=melodic

ROS中的导航框架

amcl[1]

- 主要参数：

  ~odom_model_type(string, default: “diff”)：里程计模型选择

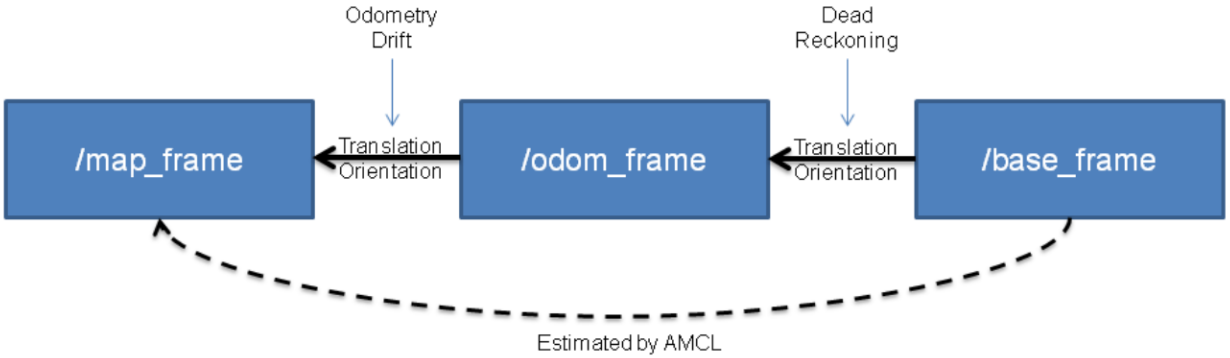
  ~odom_frame_id(string, default: “odom”)：里程计坐标系

  ~base_frame_id(string, default: “base_link”)：机器人基坐标系

  ~global_frame_id(string, default: “map”)：地图坐标系

  其余更多参数参阅 ROS Wiki

- 坐标变换：



1. http://wiki.ros.org/amcl?distro=melodic

4 综合仿真实现

综合仿真实现

amcl 使用

- 编写相关 launch 文件

  参考 amcl 功能包下的 example 目录内的示例：amcl_diff.launch

  基本参数：

```
<param name="odom_frame_id" value="odom"/>

<param name="base_frame_id" value="base_footprint"/>
<param name="global_frame_id" value="map"/>
```

- 编写测试 launch 文件（amlc_test.launch）

  加载全局地图（前面 gmapping 生成的）

  启动之前创建的 Gazebo 仿真环境（含机器人模型）

  启动刚编写的 amcl_diff.launch

综合仿真实现

```
<launch>
    <arg name="map" default="$(find ros_2dnav_learning)/map/map.yaml"/>

    <node name="map_server" pkg="map_server" type="map_server" args="$(arg map)"/>

    <include file="$(find ros_2dnav_learning)/launch/car_robot_rplidar.launch"/>
    <include file="$(find ros_2dnav_learning)/launch/amcl_diff.launch"/>
</launch>
```

amcl_test.launch

大连理工大学 Dalian University Of Technology

综合仿真实现

综合仿真实现

move_base 使用

- 编写相关 launch 文件（move_base.launch）
- 编写相关配置文件[1]

  costmap_common_params.yaml -> 通用配置文件
  global_costmap_params.yaml -> 全局规划配置文件
  local_costmap_params.yaml -> 本地规划配置文件
  base_local_planner_params.yaml -> 本地规划器配置

- 编写测试 launch 文件（nav_test.launch）

  加载全局地图（前面 gmapping 生成的）
  启动之前创建的 gazebo 仿真环境（含机器人模型）
  运行前面编写的 amcl_diff.launch
  运行刚编写的 move_base.launch

1. https://github.com/ROBOTIS-GIT/turtlebot3/tree/master/turtlebot3_navigation/param

综合仿真实现

```
<launch>
    <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen" clear_params="true">
        <rosparam file="$(find ros_2dnav_learning)/config/costmap_common_params.yaml" command="load" ns="global_costmap"/>
        <rosparam file="$(find ros_2dnav_learning)/config/costmap_common_params.yaml" command="load" ns="local_costmap"/>
        <rosparam file="$(find ros_2dnav_learning)/config/local_costmap_params.yaml" command="load"/>
        <rosparam file="$(find ros_2dnav_learning)/config/global_costmap_params.yaml" command="load"/>
        <rosparam file="$(find ros_2dnav_learning)/config/base_local_planner_params.yaml" command="load"/>
    </node>
</launch>
```

move_base.launch

综合仿真实现

```
obstacle_range: 3.0
raytrace_range: 3.5

#footprint: [[0.25, 0.3], [0.25, -0.3], [-0.25, -0.3], [-0.25, 0.3]]
robot_radius: 0.17

inflation_radius: 0.8
cost_scaling_factor: 5.0

map_type: costmap
observation_sources: scan
scan: {sensor_frame: laser_link, data_type: LaserScan, topic: /scan, marking: true, clearing: true}
```

理解各参数含义的前提下，对其进行适当调整

costmap_common_params.yaml

综合仿真实现

```
global_costmap:
  global_frame: map
  robot_base_frame: base_footprint

  update_frequency: 10.0
  publish_frequency: 10.0
  transform_tolerance: 0.5

  static_map: true
```

global_costmap_params.yaml

综合仿真实现

```
local_costmap:
  global_frame: odom
  robot_base_frame: base_footprint

  update_frequency: 10.0
  publish_frequency: 10.0
  transform_tolerance: 0.5

  static_map: false
  rolling_window: true
  width: 3
  height: 3
  resolution: 0.05
```

local_costmap_params.yaml

综合仿真实现

```
TrajectoryPlannerROS:
# Robot Configuration Parameters
  max_vel_x: 0.5
  min_vel_x: 0.1
  max_vel_theta:  1.0
  min_vel_theta: -1.0
  min_in_place_vel_theta: 1.0
  acc_lim_x: 1.0
  acc_lim_y: 0.0
  acc_lim_theta: 0.6
# Goal Tolerance Parameters
  xy_goal_tolerance: 0.10
  yaw_goal_tolerance: 0.05
# Differential-drive robot configuration
  holonomic_robot: false
# Forward Simulation Parameters
  sim_time: 0.8
  vx_samples: 18
  vtheta_samples: 20
  sim_granularity: 0.05
```

base_local_planner_params.yaml

理解各参数含义的前提下，对其进行适当调整

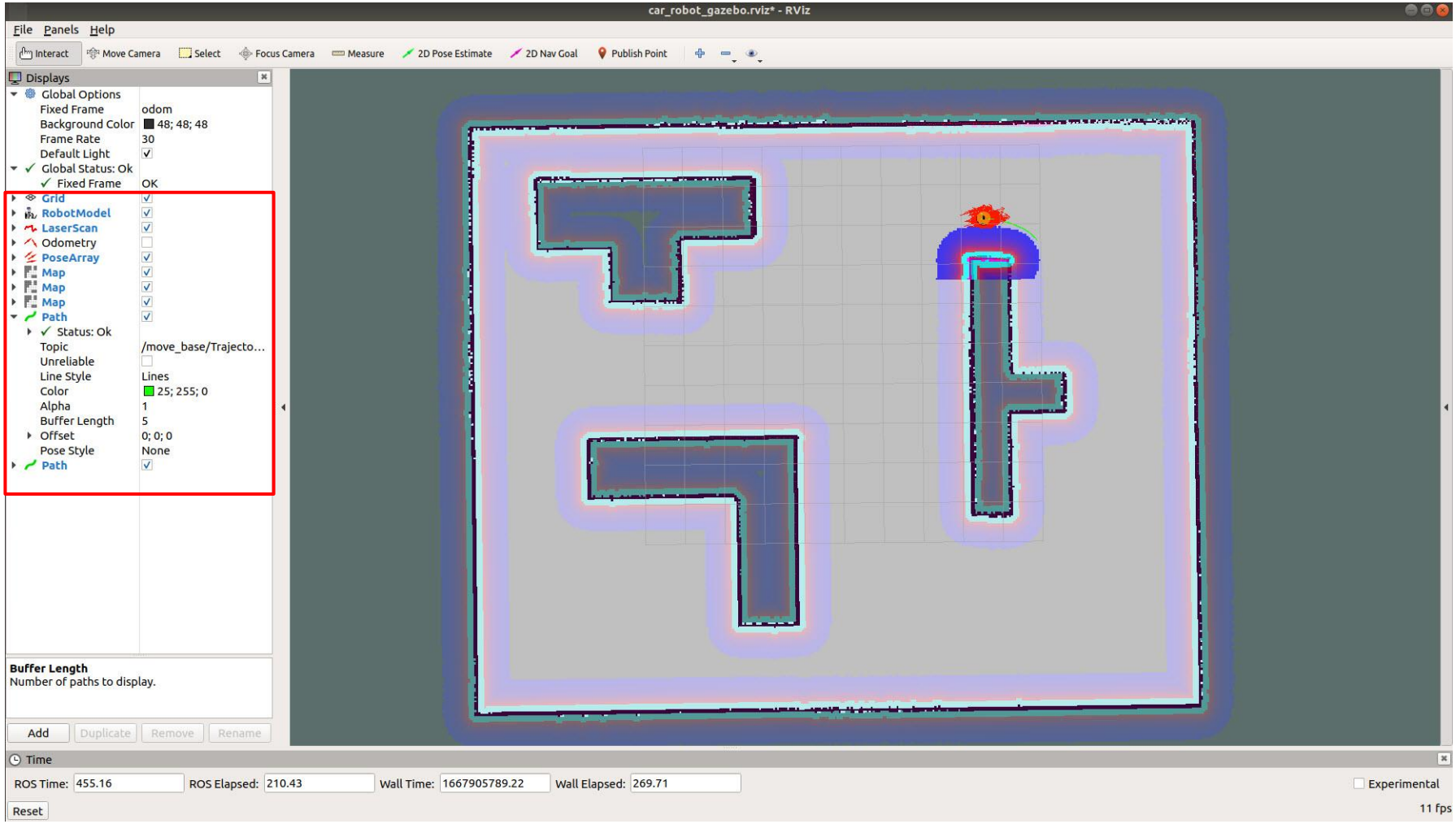
综合仿真实现

```
<launch>
    <arg name="map" default="$(find ros_2dnav_learning)/map/map.yaml"/>

    <node name="map_server" pkg="map_server" type="map_server" args="$(arg map)"/>

    <include file="$(find ros_2dnav_learning)/launch/car_robot_rplidar.launch"/>
    <include file="$(find ros_2dnav_learning)/launch/amcl_diff.launch"/>
    <include file="$(find ros_2dnav_learning)/launch/move_base.launch"/>
</launch>
```

nav_test.launch

大连理工大学 Dalian University Of Technology

综合仿真实现



综合仿真实现

SLAM+导航

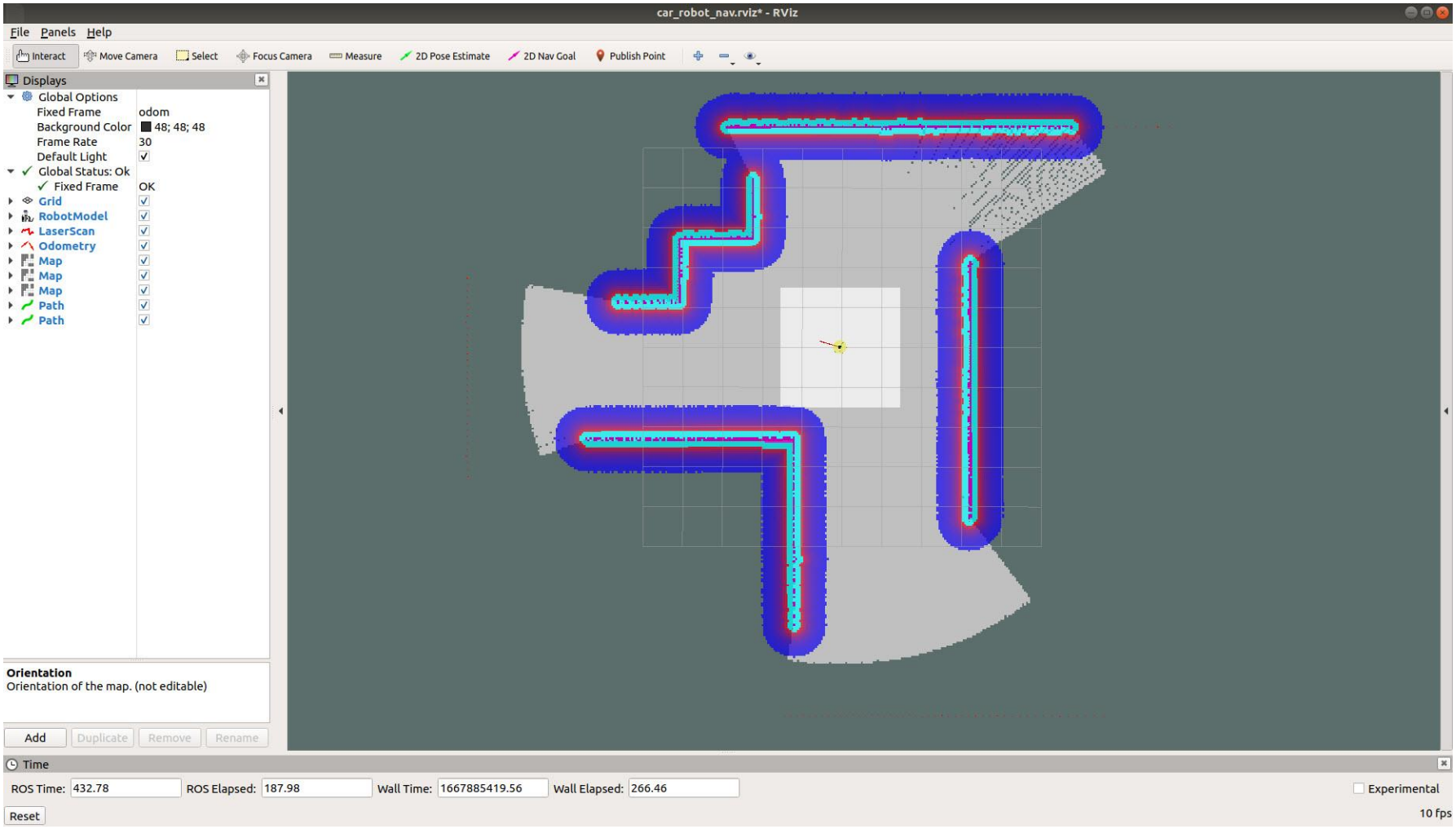
- 编写相关 launch 文件（nav_slam_test.launch）
  启动之前创建的 Gazebo 仿真环境（含机器人模型）
  运行前面编写的 gmapping.launch
  运行前面编写的 move_base.launch

```
<launch>
    <include file="$(find ros_2dnav_learning)/launch/car_robot_rplidar_nav.launch"/>
    <include file="$(find ros_2dnav_learning)/launch/gmapping.launch"/>
    <include file="$(find ros_2dnav_learning)/launch/move_base.launch"/>
</launch>
```

nav_slam_test.launch

大连理工大学 Dalian University Of Technology

综合仿真实现



综合仿真实现

建图+自主导航？

小结

小结

1. 相关基础概念

- 机器人定位与导航，栅格地图(代价地图)，准备工作

2. SLAM功能包的应用

- ROS 中相关常用功能包，gmapping 功能包的使用

3. ROS 中的导航框架

- move_base、amcl

4. 综合仿真实现

- amcl 的使用，move_base 的使用，SLAM+导航
- 建图+自主导航仿真?

注意：使用虚拟机仿真可能会有一些奇怪的错误

谢谢！