

华中科技大学计算机学院 《系统能力培养》课程报告

项目名称： 系统能力培养—智能协同系统

团队成员：

班 级	姓 名	学 号	贡献百分比	个人得分
CS2004	王博涵	U202015387	0.35	
CS2004	李学森	U202015409	0.35	
CS2006	齐奕然	U202015459	0.2	
CS2008	曾睿孜	U202015534	0.1	

注：团队成员贡献百分比之和为1

教师评语：

小组总得分：

教师签名：

给分日期：

华中科技大学课程设计报告

目 录

《系统能力培养》课程报告.....	1
1 概述	3
2 工作目标和系统模块设计	5
2.1 工作目标.....	5
2.2 系统模块设计	6
3 关键技术及其解决思路	8
3.1 实验一.....	8
3.2 实验二.....	10
3.3 实验三.....	13
4 系统实现	20
4.1 实验一.....	20
4.2 实验二.....	21
4.3 实验三基础.....	21
4.4 实验三进阶	22
5 系统测试及结果分析	23
5.1 实验一.....	23
5.2 实验二.....	23
5.3 实验三基础.....	24
5.4 实验三进阶	25
6 心得体会	26
参考文献.....	30

1 概述

当今社会，人工智能、控制技术和通信技术的不断革新推动无人系统（Unmanned System-UMS）高速发展，其类型和功能不断丰富，续航感知荷载能力不断增强。智能协同系统（Smart Cooperation System-SCS）源自 UMS，通过提升其多机通信和控制决策能力，实现智能协同控制。目前智能协同系统在工业生产、安全巡查、科技研究等众多领域大放异彩。

ROS 2 (Robot Operating System 2) 是一种用于机器人和自动化系统开发的开源框架。它是 ROS (Robot Operating System) 的下一代版本，旨在解决 ROS 1 中存在的一些限制和挑战。ROS 2 的设计目标是提供更好的可扩展性、可靠性、安全性和实时性，以适应日益复杂的机器人应用和系统需求。

Rviz (ROS Visualization) 是 ROS (Robot Operating System) 中的一个强大可视化工具，用于实时显示和调试机器人系统的各个方面，包括传感器数据、机器人模型、导航路径等。它能够以三维形式显示机器人的模型，传感器数据、点云、激光雷达数据等，并支持用户交互，允许用户实时查看和修改机器人的状态和任务执行情况。

Gazebo 是一个多机器人仿真工具，用于在虚拟环境中模拟机器人的运动、传感器反馈和环境交互。它提供了一个真实感强、高度可定制的仿真平台。它支持多机器人协同工作，可以模拟多种传感器（如激光雷达、摄像头）的输出，并允许用户在仿真环境中测试和验证机器人算法，从而减少在物理机器人上进行试验的成本。

URDF (Unified Robot Description Format) 是一种 XML 格式的文件，用于描述机器人的三维模型和其相关的运动学、动力学信息。URDF 通常用于 ROS 中，以定义机器人的结构、关节、传感器和其他相关属性，使得机器人系统能够在 ROS 中进行仿真、控制和可视化。URDF 文件的创建和编辑通常由机器人开发者完成，它为机器人系统提供了一个统一的描述格式，使得机器人能够在 ROS 中无缝集成，同时也为仿真和可视化工具提供了必要的信息。URDF 的使用促进了机器人系统的模块化和可扩展性，使得开发者能够更方便地进行机器人设计、仿真和控制。

华中科技大学课程设计报告

实验中，我团队将基于上述的技术点，设计并实现一种具有多机通信和控制决策能力并能实现智能协同控制的智能小车，我们将从基础设计出发，搭建一个完整的机器人仿真环境，包括机器人模型、传感器模型和场景模型。设计仿真和寻路机制，利用 Cartographer 进行机器人地图构建与定位，并通过 RVIZ 实时可视化机器人状态、传感器数据和地图信息，最终实现小车的巡航。

我们将深入了解先进的人工智能和机器人技术，培养工程实践能力，探索新的算法、传感器融合和人机交互等领域。基于 ROS2 构建机器人仿真环境，集成 RVIZ、Gazebo、Nav2 和 Cartographer（即：引入 Gazebo 进行场景建模，添加导航算法，实现机器人自主导航、运用 Cartographer 进行地图构建与定位，优化机器人路径规划同时使用 RVIZ 进行实时可视化，监控机器人状态与环境信息）。

最终，我们将搭建一个可复用的机器人仿真平台，为实际应用提供技术支持，同时提高机器人研发效率，降低实际测试成本力争实现为社会各个领域提供高效、安全、智能的解决方案。

2 工作目标和系统模块设计

2.1 工作目标

总体工作目标为，熟悉 ROS 2 框架的信息传输机制，了解其相对于上一代 ROS 1 的区别与改进，明确 ROS 2 的框架结构优势，初步掌握 ROS 2 信息传输机制的使用方法；

熟悉 Rviz、Gazebo 等仿真平台的使用，能够根据实际场景自定义在模拟器中仿真出相应的机械结构与运动方式；

了解 Slam 建图的基本原理，了解主流的寻路导航算法。

2.1.1 实验一

- 完成运行 ROS 2 需要的 ubuntu 环境的配置，安装 ROS 2 框架以及基于 ROS 2 的 publisher-subscriber 结构；
- 实现简单的节点之间传输数据的功能；
- 使用 Publisher-Subscriber 结构，完成两个 node 节点间传输数据功能传输文字与视频。

2.1.2 实验二

- 完成 URDF、rviz、Gazebo 的探索学习；
- 分别针对 rviz 和 gazebo 软件利用 urdf 格式语言创建自定义模型，创建激光雷达；
- 在 Gazebo 中搭建仿真环境，在 rviz 模拟器中使用键盘完成对小车的控制，并实时在 Gazebo 中完成构图。

2.1.3 实验三

- 使用自定义算法，完成寻路导航
- 利用自定义算法在 rviz 中设定起点和终点，完成自主导航
- 进一步在 rviz 中指定路径完成小车的循环运行实现巡视功能。

2.2 系统模块设计

2.2.1 实验一

首先设置工作空间，在该工作空间下设置实验一的文件夹，其树形结构如下图 1 所示。

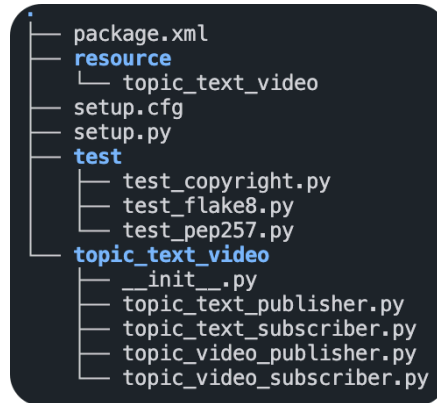


图 1 topic_text_video 文件夹的树形结构图

随后在该文件目录下设置 topic_text_video 文件夹作为根目录存放项目文件，其中四个文件的作用如下表格 1 所示。

表格 1 topic_text_video 文件夹下文件的作用和实现功能

文件名称	文件作用	实现功能
topic_text_publisher	建立发布节点提供文字交换信息到对应 topic	完成文本传输功能
topic_text_subscriber	建立订阅节点从对应 topic 上订阅文字信息	
topic_video_publisher	建立发布节点逐帧提供图像交换信息到对应 topic	完成视频传输功能
topic_video_subscriber	建立订阅节点从对应 topic 上订阅图像信息并组合成视频文件	

2.2.2 实验二

Laser_car_description 模块（表格 2）

表格 2 Laser_car_description 的功能展示

功能名称	模块功能
launch 模块	编写 rviz_gazebo_world.launch 负责将各个节点信息进行统合并启动程序，在完成各节点话题发布订阅之间的通讯的同时统合这些信息并且 rviz/gazebo 中打开对应的模型。

华中科技大学课程设计报告

功能名称	模块功能
urdf 模块	保存 urdf 格式文件用于存储机器人关节配置，部件配置，两轮差速配置和激光雷达配置等的基础配置信息
world 模块	保存在 gazebo 中手动搭建的机器人活动虚拟环境对应的.world 格式信息

2.2.3 实验三

Laser_car_cartographer 模块（表格 3）

基本描述：将 cartographer 启动模块和参数配置模块集合在一个模块里用于启动 cartographer 服务以及相关功能。

表格 3 Laser_car_cartographer 的功能展示

功能名称	模块功能	节点信息
config 模块	其中用 lua 格式的文件存储配置信息	完成文本传输功能
launch 模块	综合 cartographer 各节点之间的信息交流关系	在其中设立 cartographer_node 节点用于接收机器人在 gazebo 虚拟环境中的扫描的信息 occupancy_grid_node 节点订阅该信息并且将其拼接成 map 发布给 rviz 软件对应节点 rviz_node，并在过程中进行一定程度的参数调整。

Map 模块：用于将 cartographer 得到的地图进行存储。

Laser_car_navigation2 模块（表格 4）

基本描述：帮助 nav2 完成基础配置，将相应节点信息統合帮助正确启动 nav2

表格 4 Laser_car_navigation2 的功能展示

功能名称	模块功能
maps 模块	存储 cartographer 通过扫描 gazebo 虚拟空间得到的计算地图的 pgm
launch 模块	launch 格式文件用于定位包的位置；声明参数并获取配置文件路径；声明启动 launch 文件，传入地图路径、是否使用仿真时间及 nav2 参数文件

Navigation2_cmd 模块：

基本描述：主要用于实现功能在视图中指定路径，并且使得小车可以进行循环运行实现巡视功能的一系列代码。

3 关键技术及其解决思路

3.1 实验一

功能包 topic_text_video

(1) 文本传输:

使用 Publisher-Subscriber 结构, 分别创建发布者和订阅者结点, 发布的话题名为“chatter”, 消息类型为标准消息库 std_msgs.msg 中的文本消息类型 String。

以下为发布者的代码段展示:

```
1. """
2. 创建一个发布者节点
3. """
4. class PublisherNode(Node):
5.     def __init__(self, name):
6.         super().__init__(name) # ROS2 节点父类初始化
7.         self.pub = self.create_publisher(String, "chatter", 10) # 创建发布者对象 (消息类型、话题名、队列长度)
8.         self.timer = self.create_timer(1, self.timer_callback) # 创建一个定时器 (单位为秒的周期, 定时执行的回调函数)
9.         self.cnt=0 # 计数器
10.        def timer_callback(self): # 创建定时器周期执行的回调函数
11.            msg = String() # 创建一个String类型的消息对象
12.            msg.data = 'Call subscriber for '+str(self.cnt)+' times!'
13.            # 填充消息对象中的消息数据
14.            self.pub.publish(msg) # 发布话题消息
15.            self.get_logger().info('Publishing: "%s"' % msg.data)
16.            # 输出日志信息, 提示已经完成话题发布
17.            self.cnt+=1
```

以下为订阅者的代码段展示:

```
1. """
2. 创建一个订阅者节点
3. """
4. class SubscriberNode(Node):
5.     def __init__(self, name):
6.         super().__init__(name) # ROS2 节点父类初始化
7.         self.sub = self.create_subscription(\
8.             String, "chatter", self.listener_callback, 10) # 创建订阅者对象 (消息类型、话题名、订阅者回调函数、队列长度)
```



```
9.     def listener_callback(self, msg): # 创建回调函数，执行收到话题消息后
        对数据的处理
10.         self.get_logger().info('I heard: "%s"' % msg.data) # 输出日志
            信息，提示订阅收到的话题消息
```

(2) 视频传输：

使用 Publisher-Subscriber 结构，分别创建发布者和订阅者结点，发布的话题名为“image_raw”，类型为 sensor_msgs.msg 库中的图像消息类型 Image。

发布者中的图片首先通过 OpenCV 图像库中的 VideoCapture() 函数调用摄像头进行采集，然后通过 cv_bridge 中的 CvBridge() 函数将图像转换为 ROS2 的数据格式再发送出去。

订阅者通过 cv_bridge 中的 imgmsg_to_cv2() 函数将消息数据转换回 OpenCV 格式再展示出来。

以下为发布者的代码段展示：

```
1. """
2. 创建一个发布者节点
3. """
4. class PublisherNode_2(Node):
5.     def __init__(self, name):
6.         super().__init__(name)
7.         # ROS2 节点父类初始化
8.         self.publisher_ = self.create_publisher(Image, 'image_raw', 1
9.         0) # 创建发布者对象（消息类型、话题名、队列长度）
10.        self.timer = self.create_timer(1.0/30, self.timer_callback)
11.        # 创建一个定时器（单位为秒的周期，定时执行的回调函数）
12.        self.cap = cv2.VideoCapture(0)
13.        # 创建一个视频采集对象，驱动相机采集图像（相机设备号）
14.        self.cv_bridge = CvBridge()# 创建一个图像转换对象，用于稍后
15.        将OpenCV 的图像转换成 ROS 的图像消息
16.        def timer_callback(self):
17.            ret, frame = self.cap.read()# 一帧一帧读取图像
18.            frame = cv2.resize(frame, (640, 480)) # 将帧的大小调整为
19.            480p
20.            if ret == True: # 如果图像读取成功
21.                self.publisher_.publish(
22.                    self.cv_bridge.cv2_to_imgmsg(frame, 'bgr8')) # 发
23.                    布图像消息
24.                self.get_logger().info('Publishing video frame') # 输
25.                出日志信息，提示已经完成图像话题发布
```

以下为订阅者的代码段展示：

```
1. """
2. 创建一个订阅者节点
3. """
4. class SubscriberNode_2(Node):
5.     def __init__(self, name):
6.         super().__init__(name) # ROS2 节点父类初始化
7.         self.sub = self.create_subscription(
8.             Image, 'image_raw', self.listener_callback, 10) # 创建订
           阅者对象（消息类型、话题名、订阅者回调函数、队列长度）
9.         self.cv_bridge = CvBridge() # 创建一个图像转换对象，用于OpenCV 图
           像与ROS 的图像消息的互相转换
10.        def listener_callback(self, data):
11.            self.get_logger().info('Receiving video frame') # 输出
           日志信息，提示已进入回调函数
12.            image = self.cv_bridge.imgmsg_to_cv2(data, 'bgr8') # 将
           ROS 的图像消息转化成OpenCV 图像
13.            cv2.imshow("video", image) # 使
           用OpenCV 显示处理后的图像效果
14.            cv2.waitKey(10)
```

3.2 实验二

功能包：

cartographer(依赖)、cartographer_ros(依赖)、laser_car_description、
laser_car_cartographer

编写 URDF 文件创建激光雷达小车模型，在 Gazebo 中搭建仿真环境，通过 laser_car_description 功能包中的 rviz_gazebo_world.launch.py 文件启动激光雷达小车的 Gazebo 仿真。

rviz_gazebo_world.launch.py 如下：

```
1. import os
2. from launch import LaunchDescription
3. from launch.actions import ExecuteProcess
4. from launch_ros.actions import Node
5. from launch_ros.substitutions import FindPackageShare
6.
7. def generate_launch_description():
8.     robot_name_in_model = 'laser_car'
9.     package_name = 'laser_car_description'
10.    urdf_name = "laser_car_base.urdf"
11.    ld = LaunchDescription()
```

```
12.         pkg_share = FindPackageShare(package=package_name).find(packa
           ge_name)
13.         urdf_model_path = os.path.join(pkg_share, f'urdf/{urdf_name}'
           )
14.         gazebo_world_path = os.path.join(pkg_share, 'world/six_edge.w
           orld')
15.         # Start Gazebo server
16.         start_gazebo_cmd = ExecuteProcess(
17.             cmd=['gazebo', '--verbose', '-
           s', 'libgazebo_ros_init.so', '-
           s', 'libgazebo_ros_factory.so', gazebo_world_path],
18.             output='screen')
19.         # Launch the robot
20.         spawn_entity_cmd = Node(
21.             package='gazebo_ros',
22.             executable='spawn_entity.py',
23.             arguments=['-entity', robot_name_in_model, '-
           file', urdf_model_path ], output='screen')
24.         # Start Robot State publisher
25.         start_robot_state_publisher_cmd = Node(
26.             package='robot_state_publisher',
27.             executable='robot_state_publisher',
28.             arguments=[urdf_model_path]
29.         )
30.         # Launch RViz
31.         start_rviz_cmd = Node(
32.             package='rviz2',
33.             executable='rviz2',
34.             name='rviz2',
35.             output='screen',
36.             # arguments=['-d', default_rviz_config_path]
37.         )
38.         ld.add_action(start_gazebo_cmd)
39.         ld.add_action(spawn_entity_cmd)
40.         ld.add_action(start_robot_state_publisher_cmd)
41.         ld.add_action(start_rviz_cmd)
42.         return ld
```

在建图方面选择了 cartographer 框架，安装了 cartographer 与 cartographer_ros 两个功能包后，在 laser_car_cartographer 功能包中按照默认模板编写 cartographer 的参数文件 laser_car_2d.lua，然后编写 cartographer.launch.py 文件启动建图程序，最后通过 teleop_twist_keyboard 来调用键盘控制小车的移动完成建图的工作。

华中科技大学课程设计报告

cartographer.launch.py 如下

```
1. import os
2. from launch import LaunchDescription
3. from launch.substitutions import LaunchConfiguration
4. from launch_ros.actions import Node
5. from launch_ros.substitutions import FindPackageShare
6.
7.
8. def generate_launch_description():
9.     # 定位到功能包的地址
10.     pkg_share = FindPackageShare(package='laser_car_cartographer'
11.     ).find('laser_car_cartographer')
12.     # =====运行节点需要的配置=====
13.     # 是否使用仿真时间, 我们用gazebo, 这里设置成true
14.     use_sim_time = LaunchConfiguration('use_sim_time', default='true')
15.     # 地图的分辨率
16.     resolution = LaunchConfiguration('resolution', default='0.05'
17.     )
18.     # 地图的发布周期
19.     publish_period_sec = LaunchConfiguration('publish_period_sec'
20.     , default='1.0')
21.     # 配置文件夹路径
22.     configuration_directory = LaunchConfiguration('configuration_
23.     directory', default= os.path.join(pkg_share, 'config') )
24.     # 配置文件
25.     configuration_basename = LaunchConfiguration('configuration_b
26.     asename', default='laser_car_2d.lua')
27.     rviz_config_dir = os.path.join(pkg_share, 'config')+"/cartogr
28.     apher.rviz"
29.     print(f"rviz config in {rviz_config_dir}")
30.     # =====声明三个节点,
31.     cartographer/occupancy_grid_node/rviz_node=====
32.     cartographer_node = Node(
33.         package='cartographer_ros',
34.         executable='cartographer_node',
35.         name='cartographer_node',
36.         output='screen',
37.         parameters=[{'use_sim_time': use_sim_time}],
38.         arguments=['-
39.         configuration_directory', configuration_directory,
40.         '-
41.         configuration_basename', configuration_basename])
42.     cartographer_occupancy_grid_node = Node(
```

```
34.         package='cartographer_ros',
35.         executable='cartographer_occupancy_grid_node',
36.         name='cartographer_occupancy_grid_node',
37.         output='screen',
38.         parameters=[{'use_sim_time': use_sim_time}],
39.         arguments=['-resolution', resolution, '-
publish_period_sec', publish_period_sec])
40.     rviz_node = Node(
41.         package='rviz2',
42.         executable='rviz2',
43.         name='rviz2',
44.         arguments=['-d', rviz_config_dir],
45.         parameters=[{'use_sim_time': use_sim_time}],
46.         output='screen')
47.     #=====定义启动文件=====
48.     ld = LaunchDescription()
49.     ld.add_action(cartographer_node)
50.     ld.add_action(cartographer_occupancy_grid_node)
51.     ld.add_action(rviz_node)
52.     return ld
```

最后调用 nav2_map_server 中的 map_saver_cli 将 cartographer 构建出的地图保存。

3.3 实验三

功能包： navigation2 （ 依赖 ）， laser_car_description ， laser_car_navigation2, navigation2_cmd。

首先对于使用 laser_car_description 功能包中的 gazebo_world.launch.py 启动 Gazebo 仿真环境，其内包含了激光雷达小车以及所处的地图环境。然后调用 laser_car_navigation2 功能包中的 laser_car_nav2.launch.py 启动 navigation2 导航模块，该文件会将在任务二中通过 cartographer 构建好的地图载入 Rviz2 中。

laser_car_nav2.launch.py 如下：

```
1. import os
2. from ament_index_python.packages import get_package_share_directory
3. from launch import LaunchDescription
4. from launch.actions import IncludeLaunchDescription
5. from launch.launch_description_sources import PythonLaunchDescription
   Source
6. from launch.substitutions import LaunchConfiguration
```

```
7. from launch_ros.actions import Node
8.
9. def generate_launch_description():
10.     #=====1. 定位到包的地址=====
11.     laser_car_navigation2_dir = get_package_share_directory('laser_car_navigation2')
12.     nav2_bringup_dir = get_package_share_directory('nav2_bringup')
13.     #=====2. 声明参数，获取配置文件路径=====
14.     # use_sim_time 这里要设置成true, 因为gazebo 是仿真环境，其时间是通过/clock 话题获取，而不是系统时间
15.     use_sim_time = LaunchConfiguration('use_sim_time', default='true')
16.     map_yaml_path = LaunchConfiguration('map', default=os.path.join(laser_car_navigation2_dir, 'maps', 'laser_car_map_2.yaml'))
17.     nav2_param_path = LaunchConfiguration('params_file', default=os.path.join(laser_car_navigation2_dir, 'param', 'laser_car_nav2.yaml'))
18.     rviz_config_dir = os.path.join(nav2_bringup_dir, 'rviz', 'nav2_default_view.rviz')
19.     #=====3. 声明启动 Launch 文件，传入：地图路径、是否使用仿真时间以及 nav2 参数文件=====
20.     nav2_bringup_launch = IncludeLaunchDescription(
21.         PythonLaunchDescriptionSource([nav2_bringup_dir, '/launch', '/bringup_launch.py']),
22.         launch_arguments={
23.             'map': map_yaml_path,
24.             'use_sim_time': use_sim_time,
25.             'params_file': nav2_param_path}.items(),
26.     )
27.     rviz_node = Node(
28.         package='rviz2',
29.         executable='rviz2',
30.         name='rviz2',
31.         arguments=['-d', rviz_config_dir],
32.         parameters=[{'use_sim_time': use_sim_time}],
33.         output='screen')
34.     return LaunchDescription([nav2_bringup_launch, rviz_node])
```

navigation2 的导航需要给定小车的初始姿态，方法有两种，第一种是使用 Rviz2 提供“2D Pose Estimate”，通过鼠标给小车一个初始的方位，第二种就是我利用 navigation2 的 API 提供小车的方位，该方案被我封装到 navigation2_cmd 功能包的 init_robot_pose.py 中，如下所示：

```
1. from geometry_msgs.msg import PoseStamped
2. from nav2_simple_commander.robot_navigator import BasicNavigator, TaskResult
3. import rclpy
4. from rclpy.duration import Duration
5. def main():
6.     rclpy.init()
7.     navigator = BasicNavigator()
8.     # Set our demo's initial pose
9.     initial_pose = PoseStamped()
10.    initial_pose.header.frame_id = 'map'
11.    initial_pose.header.stamp = navigator.get_clock().now().to_msg()
12.    initial_pose.pose.position.x = 0.0
13.    initial_pose.pose.position.y = 0.0
14.    initial_pose.pose.position.z = 0.0
15.    initial_pose.pose.orientation.w = 1.0
16.    navigator.setInitialPose(initial_pose)
17.    navigator.waitUntilNav2Active()
```

小车位姿初始化之后，对于基础任务，使用“Nav2 Goal”指定目标点的位姿，然后小车就会自动导航至目标点，实现单点导航。

对于进阶任务，由于 navigation2 的路点模式在 humble 版本没有循环模式，因此需要手动编写代码实现。

具体实现分为两步。

第一步就是获取多个目标点的位置，利用 ROS2 的 topic list 和 echo 工具可以知道使用 Rviz2 的“publish point”在点击目标点后，会发布一个“/clicked_point”的话题，数据类型为 geometry_msgs.msg 库中的 PointStamped，因此可以设计一个节点订阅此话题获取所需要的目标点的坐标，而姿态则规定为 1.0。获取对应的目标点之后将其保存在一个文件中，便于下一步使用 navigation2 的 API 进行导航。由于“publish point”所点击的目标点不会保留在 Rviz2 的地图上，因此还需要设计一个发布者将获取到的点重新发布回去，在 Rviz2 中显示相应的点，具体是用名为“visualization_marker_array”的话题传输目标点的位置，形状与颜色，用“text_marker_array”话题传递序号文本，在 Rviz2 中根据 topic 来 add 对应的插件。该步骤被写入 navigation2_cmd 的 set_up_inspection_points.py 中，具体如下：

华中科技大学课程设计报告

```
1. import rclpy                                # ROS2 Python 接口库
2. from rclpy.node import Node                 # ROS2 节点类
3. from geometry_msgs.msg import PointStamped
4. from visualization_msgs.msg import MarkerArray, Marker
5. import pickle
6. import os
7. import random
8. """
9. 创建一个转发者节点
10. """
11. class Retransmission_Node(Node):
12.     def __init__(self, name):
13.         super().__init__(name)                # ROS2 节点父类初始化
14.         #接收选中的点
15.         self.sub = self.create_subscription(PointStamped, "/clicked_point", self.listener_callback, 10) # 创建订阅者对象（消息类型、话题名、订阅者回调函数、队列长度）
16.         #转发被选中的点序列
17.         self.pub = self.create_publisher(MarkerArray, 'visualization_marker_array', 10)
18.         self.text_pub = self.create_publisher(MarkerArray, 'text_marker_array', 10)
19.         self.marker_array = MarkerArray()
20.         self.text_marker_array=MarkerArray()
21.         self.id=1
22.         file_path=os.path.dirname(os.path.realpath(__file__))
23.         self.goal_poses_file=(file_path.split('/')[0])+"/src/navigation2_cmd/points_temp.pkl"
24.         self.goal_poses=[]
25.     def add_marker(self,msg):
26.         #添加标记点
27.         marker = Marker()
28.         marker.header.frame_id = msg.header.frame_id
29.         marker.type = Marker.SPHERE
30.         marker.action = Marker.ADD
31.         marker.pose.position.x = msg.point.x
32.         marker.pose.position.y = msg.point.y
33.         marker.pose.position.z = msg.point.z
34.         marker.pose.orientation.w=1.0
35.         marker.scale.x = 0.2
36.         marker.scale.y = 0.2
37.         marker.scale.z = 0.2
38.         marker.color.a = 0.5
```


华中科技大学课程设计报告

```
39.         marker.color.r = 1.0
40.         marker.color.g = 1.0
41.         marker.color.b = 0.0
42.         marker.id=self.id
43.         self.marker_array.markers.append(marker)
44.         self.pub.publish(self.marker_array)
45.     def add_text_marker(self,msg):
46.         #添加标记
47.         marker = Marker()
48.         marker.header.frame_id = msg.header.frame_id
49.         marker.type = Marker.TEXT_VIEW_FACING
50.         marker.action = Marker.ADD
51.         marker.pose.position.x = msg.point.x
52.         marker.pose.position.y = msg.point.y
53.         marker.pose.position.z = msg.point.z
54.         marker.pose.orientation.w = 1.0
55.         marker.scale.z = 0.1
56.         marker.color.a = 0.75
57.         marker.color.r = 0.0
58.         marker.color.g = 0.0
59.         marker.color.b = 0.0
60.         marker.id=self.id
61.         marker.text=str(self.id)
62.         self.text_marker_array.markers.append(marker)
63.         self.text_pub.publish(self.text_marker_array)
64.     def listener_callback(self, msg):                                # 创建回调函数，执行收到话题消息后对数据的处理
65.         self.get_logger().info('I heard: "%s"' % msg)                # 输出日志信息，提示订阅收到的话题消息
66.         self.add_marker(msg)
67.         self.add_text_marker(msg)
68.         self.id=self.id+1
69.         #存入文件中
70.         self.goal_poses.append([msg.point.x,msg.point.y,msg.point.z]
71.         )
71.         with open(self.goal_poses_file, 'wb') as f:
72.             pickle.dump(self.goal_poses, f)
73. def main(args=None):                                                # ROS2 节点主入口main 函数
74.     rclpy.init(args=args)                                           # ROS2 Python 接口初始化
75.     node = Retransmission_Node("clickpoint_sub")                   # 创建ROS2 节点对象并进行初始化
76.     rclpy.spin(node)         # 循环等待ROS2 退出
77.     node.destroy_node()      # 销毁节点对象
78.     rclpy.shutdown()
```


华中科技大学课程设计报告

```
35.         if feedback :
36.             print('Executing current waypoint: ' +
37.                   str(feedback.current_waypoint + 1) + '/' +
                   str(len(goal_poses)))
38.             now = navigator.get_clock().now()
39.             # 导航超时
40.             if now - nav_start > Duration(seconds=600.0):
41.                 navigator.cancelTask()
42.             result = navigator.getResult()
43.             if result == TaskResult.SUCCEEDED:
44.                 print('Goal succeeded!')
45.             elif result == TaskResult.CANCELED:
46.                 print('Goal was canceled!')
47.             elif result == TaskResult.FAILED:
48.                 print('Goal failed!')
49.             else:
50.                 print('Goal has an invalid return status!')
51.             pass
52.         except KeyboardInterrupt:
53.             navigator.cancelTask()
54.             print("循环被中断")
55.             break
56.     exit(0)
```

4 系统实现

4.1 实验一

均在 `topic_text_video` 功能包中实现。其结构如下所示。

(1) 文本传输：

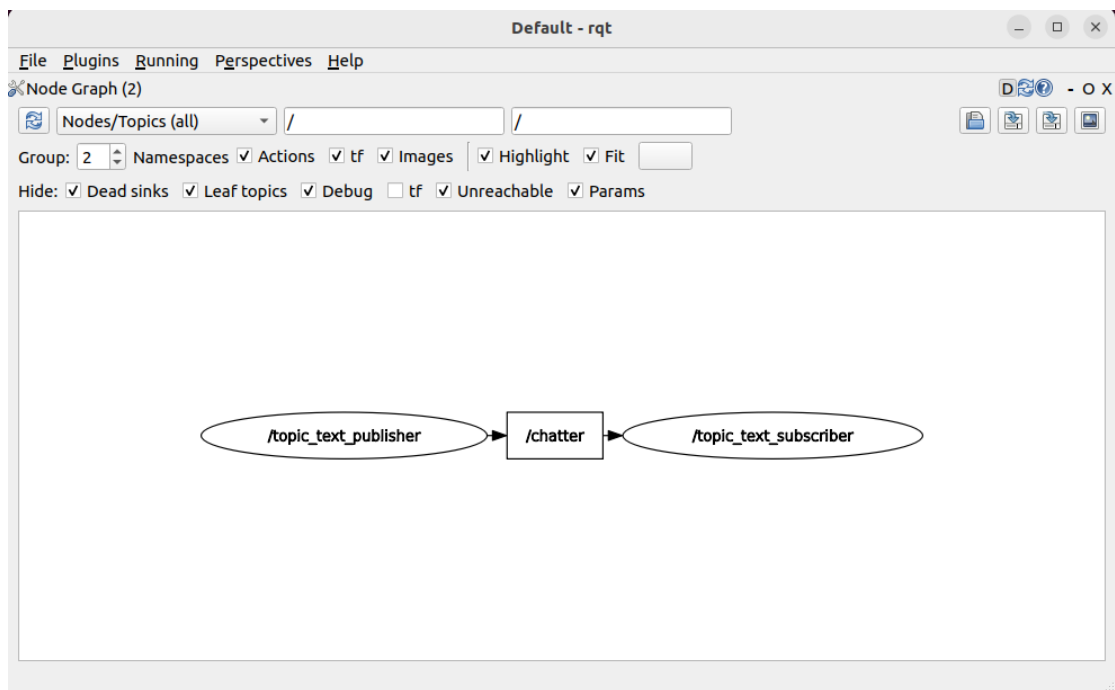


图 2 文本传输 RQT 图

(2) 视频传输：

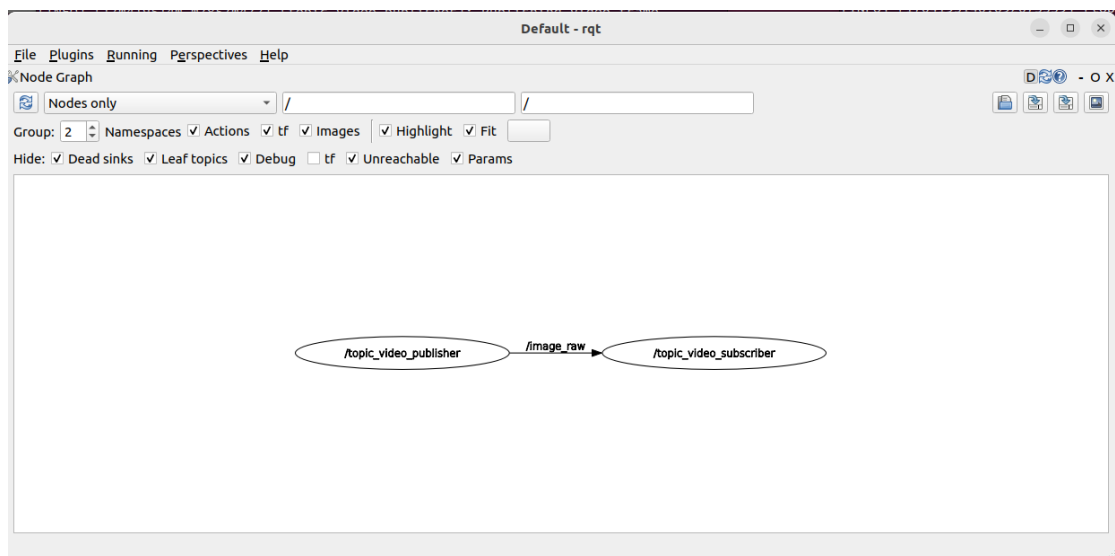


图 3 视频传输 RQT 图

4.2 实验二

在 `cartographer` , `cartographer_ros` 、 `laser_car_description` 、 `laser_car_cartographer` 中实现, 其中 `laser_car_description` 负责 URDF 文件描述的激光雷达小车在 Rviz2 与 Gazebo 中的展示以及 Gazebo 仿真器加载预先搭建好的地图。 `laser_car_cartographer` 负责调用 `cartographer` , `cartographer_ros` 的工具, 启动建图程序。

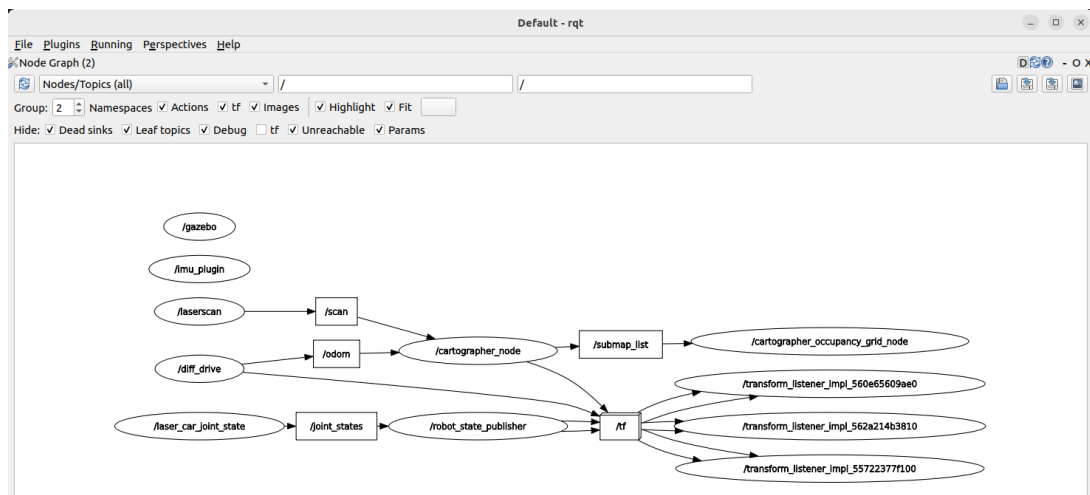


图 4 实验二 RQT 图

4.3 实验三基础

在 `navigation2` , `laser_car_description` , `laser_car_navigation2` , `navigation2_cmd` 功能包中实现。 `laser_car_description` 负责 URDF 文件描述的激光雷达小车在 Gazebo 中的展示以及 Gazebo 仿真器加载预先搭建好的地图。 `laser_car_navigation2` 负责在 Rviz2 中展示任务二中构建的地图并调用 `navigation2` 的导航功能, `navigation2_cmd` 提供初始化位姿的功能。

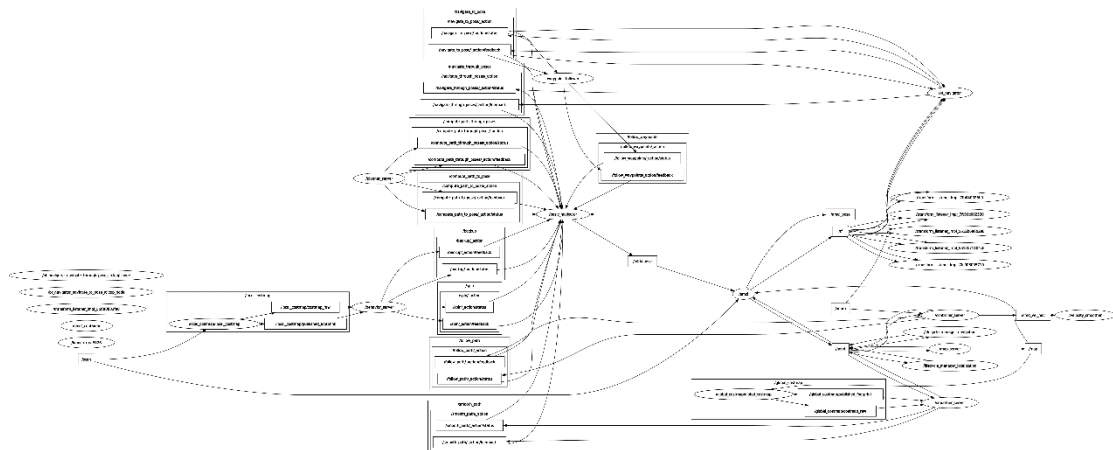


图 5 实验三基础 RQT 图

4.4 实验三进阶

在 `navigation2`，`laser_car_description`，`laser_car_navigation2`，`navigation2_cmd` 功能包中实现。`laser_car_description` 负责 URDF 文件描述的激光雷达小车在 Gazebo 中的展示以及 Gazebo 仿真器加载预先搭建好的地图。`laser_car_navigation2` 负责在 Rviz2 中展示任务二中构建的地图并调用 `navigation2` 的导航功能，`navigation2_cmd` 提供初始化位姿的功能，获取多个目标点的功能以及启动小车循环导航到多个目标点的功能。

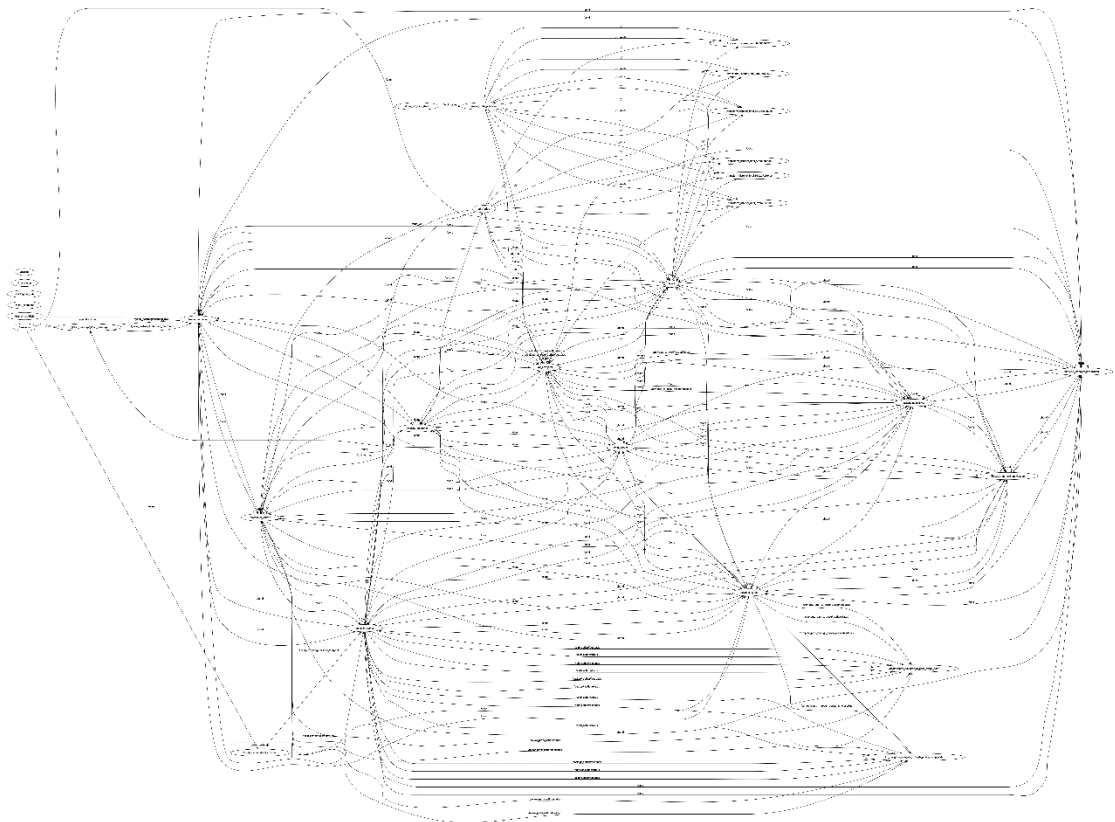
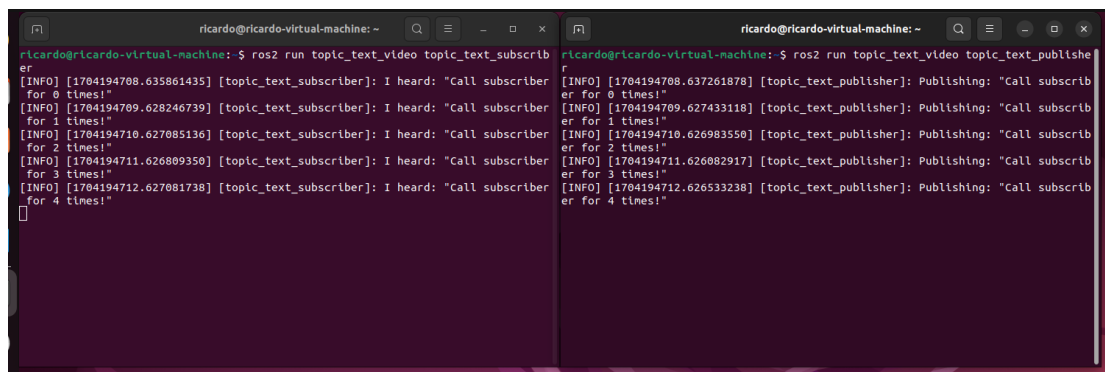


图 6 实验三进阶 RQT 图

5 系统测试及结果分析

5.1 实验一

(1) 文本传输:



```

ricardo@ricardo-virtual-machine: ~
ricardo@ricardo-virtual-machine: $ ros2 run topic_text_video topic_text_subscrib
[INFO] [1704194708.635861435] [topic_text_subscriber]: I heard: "Call subscriber
for 0 times!"
[INFO] [1704194709.628246739] [topic_text_subscriber]: I heard: "Call subscriber
for 1 times!"
[INFO] [1704194710.627085136] [topic_text_subscriber]: I heard: "Call subscriber
for 2 times!"
[INFO] [1704194711.626809350] [topic_text_subscriber]: I heard: "Call subscriber
for 3 times!"
[INFO] [1704194712.627081738] [topic_text_subscriber]: I heard: "Call subscriber
for 4 times!"
ricardo@ricardo-virtual-machine: ~
ricardo@ricardo-virtual-machine: $ ros2 run topic_text_video topic_text_publish
[INFO] [1704194708.637261878] [topic_text_publisher]: Publishing: "Call subscrib
er for 0 times!"
[INFO] [1704194709.627433118] [topic_text_publisher]: Publishing: "Call subscrib
er for 1 times!"
[INFO] [1704194710.626983550] [topic_text_publisher]: Publishing: "Call subscrib
er for 2 times!"
[INFO] [1704194711.626802917] [topic_text_publisher]: Publishing: "Call subscrib
er for 3 times!"
[INFO] [1704194712.626533238] [topic_text_publisher]: Publishing: "Call subscrib
er for 4 times!"
    
```

图 7 文本传输结果图

结果分析：由图 7 所演示的传输结果来看，topic_text_subscriber 结点成功接收到 topic_text_publisher 发布的文本信息。

(2) 视频传输:

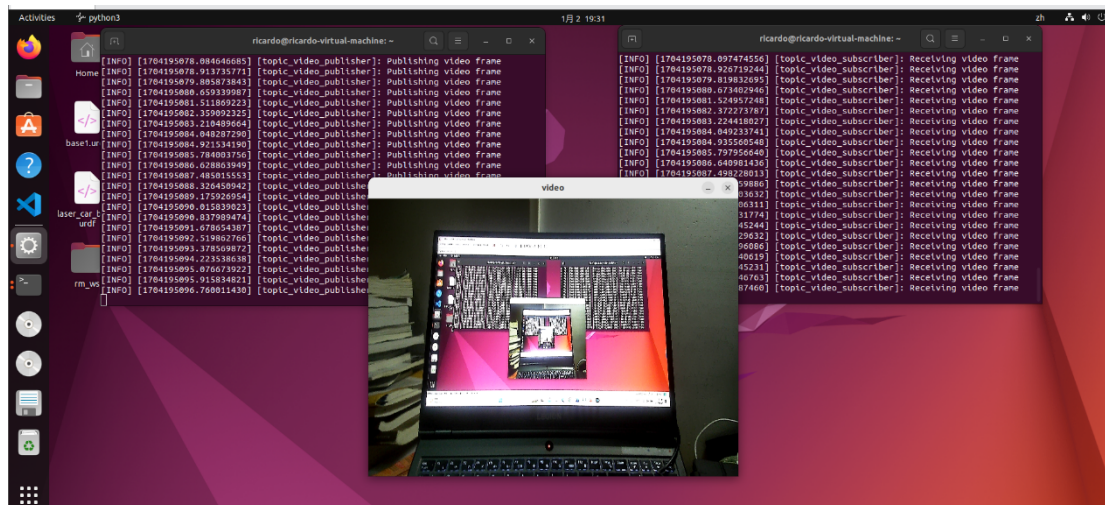


图 8 视频传输结果图

结果分析：由图 8 演示的结果来看，topic_video_subscriber 结点成功接收到 topic_video_publisher 发布的视频信息，但由于“publisher-subscriber”结构的限制，以及虚拟机处理图片信息的速度较慢，导致视频较为卡顿，且延迟较高。

5.2 实验二

使用 cartographer 进行建图。

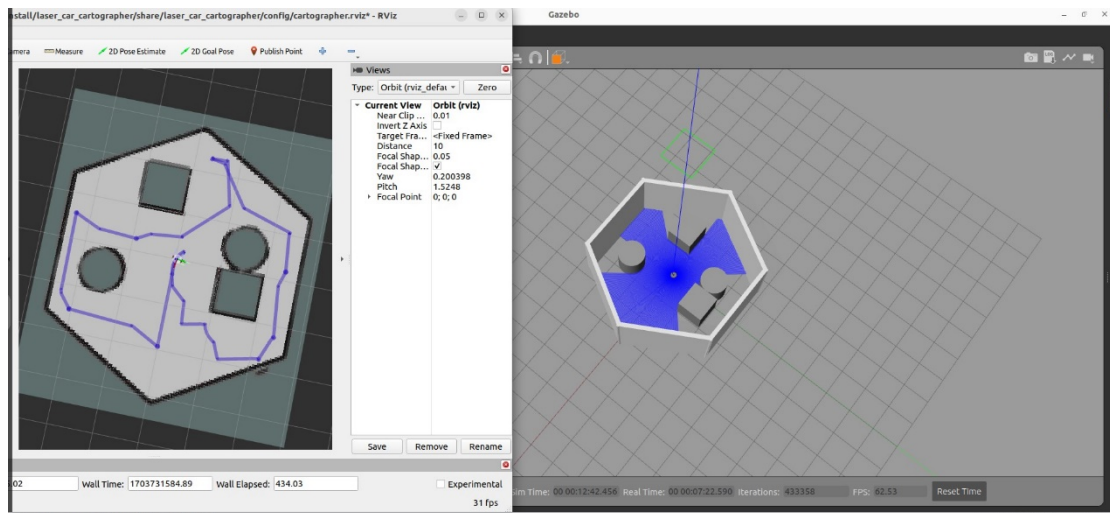


图 9 cartographer 建图结果

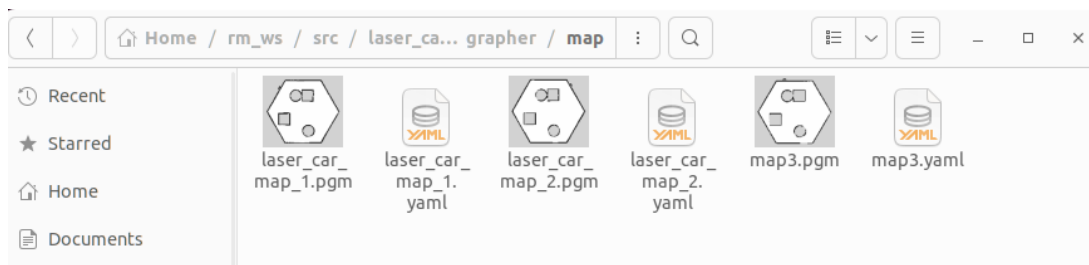


图 10 地图存储结果

结果分析：在 Gazebo 仿真软件中成功展示出了预先设置好的地图以及激光雷达小车模型，如图 9 所示。在 Rviz2 模拟器中也成功通过键盘控制小车的移动完成了当前地图的绘制工作，地图也成功保存，如图 10 所示。

5.3 实验三基础

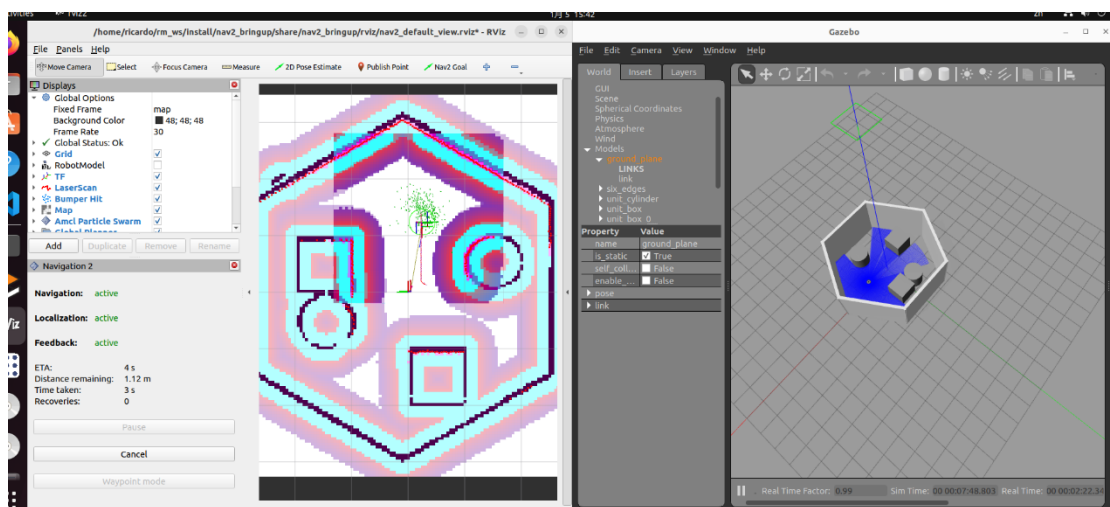


图 11 指定目标导航结果图

结果分析：图 11 表示成功实现指定目标点及其位姿之后，小车自动导航至

6 心得体会

CS2004 王博涵

本次能力培养我收获颇深，我接触到了一个完全新的领域——ROS2，从入门开始学起，了解并尝试熟练使用 ROS2 来完成各项任务。

我首先研究了 Linux 和 ROS2 的相关指令系统，让自己在使用过程中，能够尽量正确、高效的完成既定的任务。同时我也对 ROS2 和 ROS1 的区别做了详细的了解，让我明白了 ROS 在演变过程中所改进、取舍的部分和原因。

我对 ROS2 的了解始于实验一的先导——收发者的通信，我使用了 ROS2 中自带的 demo 进行测试，在 terminal 中，通过分别运行 `ros2 run demo_nodes_py listener` 指令和 `ros2 run demo_nodes_cpp talker` 指令，来实现最基础的通信功能；随后我对实验中使用 Publisher-Subscriber 结构进行实现文字和视频的传输，仿照基本的实现方式，我提升了自己针对特定问题的解决能力和处理信息的能力。

在实验 2、3 中，为了能够更好的实现，我和李学森沟通，在不会的问题上查阅资料，询问同学，尽量完美的实现建模和寻路的小车模型。我首先学习了基础的坐标变换和车辆构造的基本信息，随后深入了解了 URDF 统一机器人建模语言，并且能够熟练的使用该建模语言完成基本的建模操作。这其中我也遇到了许多困难和问题，比如我的虚拟机内存不足，在运行 RVIZ2 时经常崩溃，或是 ROS2 的安装版本不对，未使用 humble 而导致系统出现大量报错等等，这些问题都在查阅资料和耐心的 debug 之后得以解决。在这个过程当中，不仅提升了我解决问题的能力，也让我在面对突发问题时，能够保持清醒的头脑，在关键为题上抓住要点，逐一攻破。

同时我也更加深入的明晰了节点的发布和使用的思路 and 流程，与实验一相比，我加深了对节点的认识，同时也提升了代码能力。

在随后的仿真模块中，我首先学习了如何将 URDF 与实际的物理模型相连接，在最开始的学习当中，我无法理解，URDF 将部件编写后如何带入实际的物理模型来达到真实的使用效果，但在学习了实际的场景之后，我发现在编程的时候需要有逻辑的将所有的物理情况考虑进去。诸如悬浮的小车、雷达和车身分离、轮

子不转只能漂移的场景时万万不能出现的。

Gazebo 的学习和使用让我吃尽了苦头，由于我的 Linux 系统问题，我尝试了许多镜像，都无法安装此应用程序，最后找到了一台台式机的 Linux 的系统才解决了问题，导致我的进度放缓，也拖慢了小组的进度。因此通过这个问题，我也明白了提前准备好环境的重要性，在小组的协作中，任何的准备工作都非常的重要，是百倍于自己完成任务时的，因此在下次小组活动中我也会多加注意（或许是在读研的时候）。

同时我也学习并且融合了差速器、IMU 惯性测量单元、LiDAR 激光雷达等等的使用，增强了我的动手能力和代码能力。同时也增强了，在空间构建中所必不可少的空间想象能力。

最终我学习了最难理解的，也是最难实现的一部分，即导航功能的实现。我先后学习了整体的建图和全局定位系统中的 SLAM 算法，深刻理解了在这个过程当中，各个模块的运作机制，利于后续编程。同时我学习了如何下载并使用 Nav2 导航，同时将其与实验二中完成设计的小车相融合。

最终，在小组的通力合作下我们完成了这项设计。作为本科期间的最后一次小组的设计作业，我希望尽自己最大的努力为小组做贡献，同时我在此过程中，也收获了很多，学习了优秀同学处理问题的思路 and 他们的关注点，这有利于我提升自己的能力和今后处理问题的步骤和思路。

最后感谢小组的全体成员，感谢课程组的全体老师。

CS2004 李学森

在本次系统能力培养课程中，我受益匪浅。我从零开始，学习 ROS2 框架，掌握了 ROS2 的基础操作，也在 Rviz2 与 Gazebo 模拟器中实现了激光雷达小车以及地图的搭建，SLAM 绘制地图，以及最终的导航功能。这些任务不仅让我掌握了 cartographer, navigation2 等功能包的使用方式，还让我对于机器人系统的工作方式有了更加深刻的理解。在这些任务当中难度最大的就是任务三的进阶版，由于我采用的 ROS2 版本为 humble 版本，对应的 navigation2 的功能包中只有单点导航以及不包含循环的路点导航模式，因此需要自己编写代码实现多点循环导航的功能。所以我查看 nav2 网站上给出的命令控制 API 以及 Rviz2 模拟器中不

同工具发布的话题信息，根据样例代码仿写程序，实现多点循环导航以及目标点在 Rviz2 中的展示。这段时间的学习，增强了我对机器人系统的兴趣，为我未来的发展提供助力。

CS2006 齐突然

本次系统学习让我全面的了解了 ros2 体系的有趣之处，通过对于相关知识的学习，生拉硬拽般地了解了一个自己完全没接触过地全新领域，提升了对于现代前沿技术体系的认知。这一段时间的学习锻炼了我的意志，磨练了个人良好品格，拓宽了自己的视野，提升了自己的学习能力，无论 ros2 系统知识学习程度如何，至少在考研超长空档期之后第一时间巩固了一些基础技能，为将来进一步学习其他知识打下了良好的基础；并且能够和小组同学们进行友好的合作，我们组员之间通过共同努力很快的完成了设定的学习目标，积累了小组合作的经验。

CS2008 曾睿孜

我的学习和实践中，我深刻认识到 ROS 2 相对于 ROS 1 的重大改进，尤其是在信息传输机制方面的创新。ROS 2 采用了 Data Distribution Service (DDS) 作为其通信中间件，与 ROS 1 中的 ROS Master 相比，DDS 提供了更强大、灵活且实时的通信机制。DDS 的发布-订阅模型使得 ROS 2 能够更好地支持分布式系统，提高了通信的可靠性和性能。

ROS 2 的框架结构也经历了显著的改进，引入了更加模块化的设计，使得各个功能单元可以更独立地开发和部署。这种设计带来的一个重要优势是 ROS 2 的可扩展性大大提高，适用于更广泛的应用场景。此外，ROS 2 在支持实时性需求和多语言开发方面也有所改进，使得其在嵌入式系统和复杂项目中表现更为出色。在仿真方面，Rviz 和 Gazebo 等仿真平台为 ROS 2 提供了强大的工具，使得开发者能够在虚拟环境中模拟机械结构和运动方式。通过这些仿真工具，我能够在模拟器中根据实际场景灵活地定义和测试机器人的行为，从而加速开发周期和降低实际部署的风险。

对于 SLAM 建图和导航，我了解了基本原理，并熟悉了主流的寻路导航算法。SLAM 技术对于机器人在未知环境中的自主导航至关重要，而 ROS 2 为 SLAM 算法

华中科技大学课程设计报告

的集成提供了便利的框架。通过在仿真环境中测试不同的 SLAM 算法，我更好地理解它们的优缺点以及在实际机器人应用中的适用性。

总之，通过深入学习 ROS 2 框架、仿真平台的使用以及 SLAM 建图导航等相关知识，我对机器人系统开发的整个生命周期有了更全面的了解。这些知识的掌握不仅提高了我的技术水平，也为我在实际项目中更好地设计、开发和部署机器人系统提供了坚实的基础。

参考文献

- [1] <https://fishros.com/d2lros2/#/>, 动手学 ROS2, 鱼香 ROS
- [2] <https://blog.csdn.net/msql9895070/article/details/120427788>, 详解 ROS 2 的安装步骤, 阿木实验室
<https://blog.csdn.net/u011832219/article/details/125872070>, ROS2 学习、ROS2 概述, 敲代码的雪糕
- [3] https://blog.csdn.net/qq_29923461/article/details/120413799, ROS2 入门教程—创建一个简单的订阅者和发布者, Roar 冷颜
- [4] https://blog.csdn.net/weixin_43903639/article/details/126593762, URDF + Gazebo + Rviz 仿真, LyaJpunov
- [5] https://blog.csdn.net/ost_csdn/category_9655312.html, ROS 合集, ost_ros 博客
- [6] https://blog.csdn.net/qq_27865227/article/details/125069399, 动手学习 ROS2, 鱼香 ROS