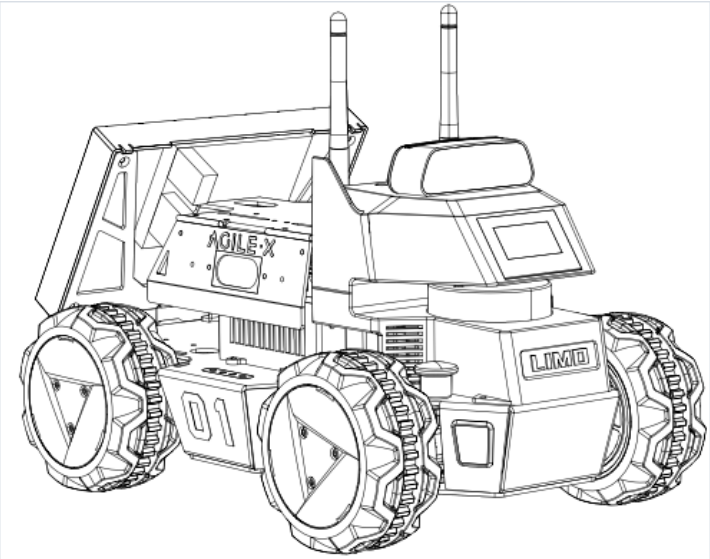
AGILEX ROBOTICS / PRODUCT MANUAL

LIMO

ROS 2 HUMBLE JETSON

产品用户使用和开发手册



计算平台	操作系统
Jetson Orin Nano	**Ubuntu 22.04**
开发框架	传感配置
ROS 2 Humble	**T-mini Pro + Dabai**

版本 1.1.3 2026.11

目 录

第 1 章 产品简介与快速上手 6
1.1 产品定位 6
1.2 随车物品与配置识别 7
1.3 核心性能参数 7
1.4 计算平台与 Jetson Orin Nano 8
1.5 部件名称与外部接口 9
1.6 功能亮点与车灯状态 9
1.7 外观、开关与电量指示 10
1.8 四种运动模态与切换方法 11
1.9 APP_Agilex 与首次移动 12
1.10 首次联网与登录 12
1.11 快速上手流程 13
第 2 章 安全与底盘电气信息 14
2.1 电池基本信息 14
2.2 充电流程 14
2.3 使用环境与机械安全 15
2.4 供电与通信拓扑 15
第 3 章 ROS 2 平台与底盘驱动 16
3.1 系统分层与工作空间 16
3.2 limo_base 驱动包 17
3.3 limo_msgs 接口 18
3.4 驱动启动与话题检查 18
3.5 Python 开发与速度控制 19
第 4 章 底盘运动学与四模态控制 21
4.1 四轮差速运动模型 21
4.2 履带运动模型 22
4.3 阿克曼运动模型 22
4.4 麦克纳姆轮运动模型 23
4.5 四模态选择建议 23
第 5 章 激光雷达建图 24

5.1 雷达参数与数据 24
5.2 GMapping 建图 25
5.3 Cartographer 原理 25
5.4 Cartographer 建图操作 26
5.5 RRT 自主探索建图 28
5.6 建图质量检查 28
第 6 章 ROS 2 激光导航与路径巡检 29
6.1 Nav2 组件 29
6.2 启动导航 30
6.3 建立初始位姿 30
6.4 单点与多点导航 31
6.5 导航前安全检查 31
6.6 路径记录与巡检 32
第 7 章 深度相机与雷达融合建图导航 37
7.1 Dabai 深度相机 37
7.2 RTAB-Map 建图 38
7.3 RTAB-Map 定位导航 38
第 8 章 视觉识别、巡线与交互应用 40
8.1 通用准备 40
8.2 文字识别 41
8.3 颜色识别 41
8.4 视觉巡线（FollowLine） 42
8.5 升降杆控制（选配） 42
8.6 ArUco 二维码识别与跟随 43
8.7 红绿灯识别 43
8.8 Ultralytics YOLO 目标检测 44
8.9 视觉综合实验建议 45
第 9 章 远程桌面、SSH 与多设备协同 47
9.1 网络准备与共享桌面文件 47
9.2 RDP 登录与 Windows 安全提示 47
9.3 SSH 终端、图形显示与多设备分工 49
9.4 常见连接问题与日常维护 50
第 10 章 SenseVoice 语音识别与运动控制（选配） 51

10.1 SenseVoice 模型与系统部署 51
10.2 语音转文字启动与验证 51
10.3 语音控制运动实验 52
10.4 故障排查与验收 53
第 11 章 MyCobot 机械臂控制（选配） 55
11.1 连接与工作空间检查 55
11.2 MoveIt 2 仿真预检查 55
11.3 真实机械臂滑块控制 56
11.4 模型跟随真实机械臂 56
11.5 GUI 与键盘控制 56
11.6 MoveIt 2 仿真规划 57
11.7 移动抓取 58
附录 A 整车三视图与外形尺寸 59
附录 B 基本操作与常用命令 60
B.1 Linux 基本操作 60
B.2 ROS 2 编译和工作空间 60
B.3 本机常用任务 60
B.4 常用路径 61
B.5 查看本机版本 61
附录 C ROS 2 系统架构 62
C.1 计算图核心概念 62
C.2 文件系统 62
C.3 话题、服务、动作和参数的选择 63
附录 D ROS 2 工作空间与开发基础 64
D.1 工作空间结构 64
D.2 创建功能包 64
D.3 编写 launch.py 65
D.4 RViz2 与 rqt 调试 65
D.5 最小开发闭环 65
附录 E 系统镜像烧录 67
E.1 获取正确镜像 67
E.2 microSD 卡烧录 67
E.3 eMMC/NVMe 或开发板模式 67

E.4 首次启动检查 67
附录 F 底盘固件升级 68
附录 G 建图与导航参数配置 69
G.1 GMapping 参数 69
G.2 Cartographer 参数 69
G.3 Nav2 定位与局部控制 69
附录 H 配套课程与资料 71
附录 I 故障排查 72
I.1 通用排查顺序 72
附录 J 使用检查表与版本说明 73
J.1 收货验收（约 30 分钟） 73
J.2 每次实验前检查表 73
J.3 内容校准说明 73
附录 K 依赖包一览 75
附录 L 配套资源包使用说明 77
L.1 包里有什么 77
L.2 拿到包后先做三件事 77
L.3 部署到车上 78
L.4 常见问题 78

第 1 章 产品简介与快速上手

LIMO 是一款面向机器人教育、功能研发和产品验证的四模态移动机器人平台。本版把计算单元升级为 Jetson Orin Nano，并在 Ubuntu 22.04 上使用 ROS 2 Humble。四种底盘模态仍然通过机械切换与 APP/遥控模式配合完成。

计算平台

Jetson Orin Nano

6 核 Arm Cortex-A78AE / 1024 CUDA 核心

开发环境

ROS 2 Humble

Ubuntu 22.04 / 分布式通信

激光雷达

T-mini Pro

360 度二维扫描 / 0.02-12 m

深度相机

ORBBEC Dabai

RGB-D / USB 连接

1.1 产品定位

LIMO 使用四个轮毂电机和可快速转换的轮系结构，在单一车体上实现四轮差速、履带、阿克曼与麦克纳姆四种运动模态。Jetson Orin Nano 负责底盘通信、雷达与相机数据处理，ROS 2 Humble 提供节点、话题、服务、动作与参数机制，便于从基础遥控逐步进入 SLAM、Nav2 导航和视觉识别开发。

典型任务	涉及能力	建议章节
开箱检查与首次移动	电池、电量表、模式灯、APP_Agilex	1.7-1.11
底盘程序开发	驱动节点、话题、消息、Twist 速度控制	第 3-4 章
激光建图与导航	T-mini Pro、GMapping、Cartographer、Nav2	第 5-6 章
视觉建图与识别	Dabai、RTAB-Map、视觉节点	第 7-8 章
远程实验与调试	WiFi、RDP 共享桌面、SSH 与 RViz2	第 9 章

1.2 随车物品与配置识别

标准配置包括 LIMO 高配版主体、电池、充电器和 APP_Agilex。开箱后先核对数量、接口和外观，再核对工控机与软件版本。不同批次的外壳、模块位置或可选附件可能存在差异，应以随车清单和实车丝印为准。

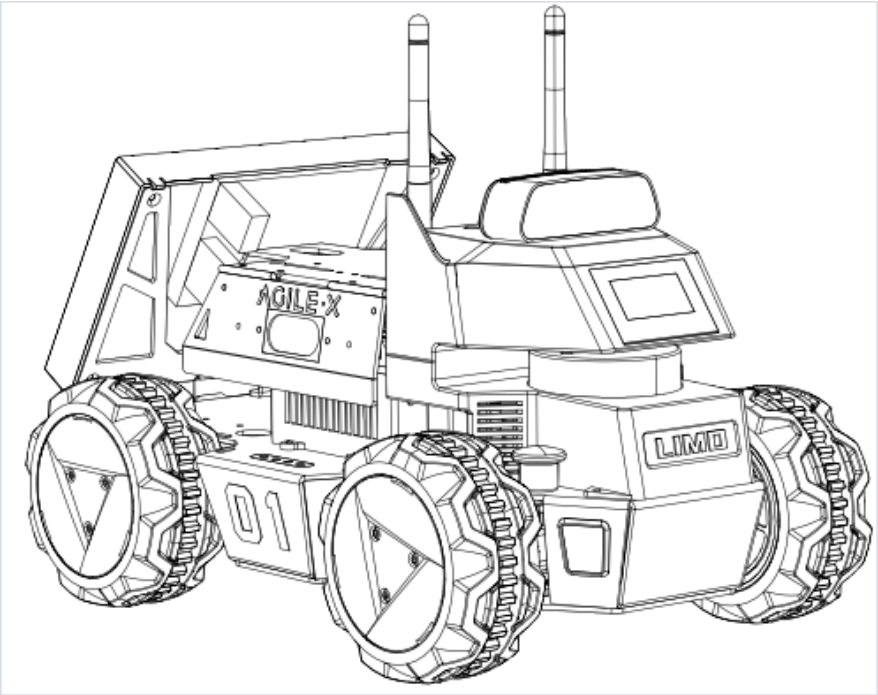


图 1-1 LIMO 高配版主体轮廓与越野轮配置示意

名称	数量	检查要点
LIMO 高配版主体（安装越野轮 x4）	x1	轮胎、插销、雷达、相机与前显示器无松动
电池	x1	输出与充电接口无破损，电量不低于提示阈值
充电器	x1	铭牌、插头与线缆完好，规格与电池匹配
APP_Agilex	x1	使用随车或官方渠道提供的版本完成蓝牙连接

> **确认本机的选配模块**
>
> 不同配置的车安装的选配模块不同。执行第 8.5、10、11 章的命令前，先用下面几条确认本机有没有对应硬件；也可以直接看车身：机械臂在车头左侧，升降杆在车顶，麦克风与声卡在车体后部。没有对应硬件时，相关章节只做原理阅读，不要执行真实硬件启动命令。

```
ls /dev/ttyACM*                                  # 机械臂（USB 串口设备）
lsusb | grep -iE 'qinheng|ch34'                   # 机械臂的 USB 转串口芯片
arecord -l                                       # 语音：是否有 USB 声卡/麦克风
ros2 pkg list | grep -E 'detect_ros|mycobot'      # 对应软件包是否已安装
ls ~/limo_voice_test ~/limo_voice_env            # 语音：SenseVoice 脚本、模型与虚拟环境
```

1.3 核心性能参数

机械与运动

项目	指标	项目	指标

项目	指标	项目	指标
外形尺寸	322 x 220 x 251 mm	轴距	200 mm
轮距	175 mm	自重	4.8 kg
负载	4 kg	最小离地间隙	24 mm
驱动方式	轮毂电机 4 x 14.4 W	空载最高车速	1 m/s
阿克曼最小转弯半径	0.4 m	最大爬坡角度	20 度

计算、电气与传感器

项目	配置	项目	配置
工控机	Jetson Orin Nano	操作系统	Ubuntu 22.04
ROS 版本	ROS 2 Humble	CPU	6 核 Arm Cortex-A78AE
GPU	1024 CUDA 核心	IMU	HI226
电池	10 Ah / 12 V 系统	工作时间	约 2.5 h
待机时间	约 4 h	通讯	WiFi、蓝牙
激光雷达	EAI T-mini Pro	深度相机	ORBBEC Dabai
显示器	前 1.54 英寸 OLED	后屏	7 英寸 1024 x 600 触控屏

电池标签上可能同时出现 12 V 系统电压与 11.1 V 标称电压。前者用于标识整车供电等级，后者是电池包的额定化学体系电压；充电截止电压为 12.6 V。不要混用不同电压或未经认证的电源。

1.4 计算平台与 Jetson Orin Nano

本机的核心计算单元为 NVIDIA Jetson Orin Nano，系统为 Ubuntu 22.04，机器人中间件为 ROS 2 Humble。底盘、雷达、相机与视觉功能采用统一的机器人任务逻辑，节点启动、话题检查与开发工作空间均使用 ROS 2 工具链。

项目	本配置信息	开发影响
计算平台	NVIDIA Jetson Orin Nano	ARM64 Linux，支持 CUDA 与 TensorRT 加速
CPU	6 核 Arm Cortex-A78AE	适合同时运行 Nav2、相机和轻量视觉模型
GPU	1024 CUDA 核心	可用于 YOLO、RGB-D 点云与视觉加速
操作系统	Ubuntu 22.04	与 ROS 2 Humble 的官方目标平台匹配
中间件	ROS 2 Humble	节点、话题与参数由 ros2 命令和 DDS 统一管理
内存与存储	以整机配置标签为准	运行大模型或保存地图前检查剩余空间

MyCobot 280 M5 机械臂（选配）

项目	指标	项目	指标

项目	指标	项目	指标
自由度	6	有效负载	250 g
工作半径	280 mm	重复定位精度	约 0.5 mm
自重	约 800 g	电源输入	12 V / 5 A
工作温度	-5 至 45 摄氏度	通信接口	USB Type-C

机械臂属于选配模块。若本机未安装机械臂，可保留对应章节作为扩展开发说明；若已安装，执行前必须确认 USB 设备号、机械臂电源、限位与工作空间，禁止在未确认姿态时快速拖动真实机械臂。

1.5 部件名称与外部接口

下图和清单用于快速定位调试接口、传感器、显示与控制部件。不同批次可能在模块位置、接口布置或选配项上存在差异，应以实车丝印和随车清单为准。

部件	作用	使用提示
WiFi/蓝牙天线	无线网络和 APP_Agilex 通信	不要用金属件遮挡或压折天线
前显示器与后屏幕	显示状态、触摸交互	后屏通常通过 USB 接口连接
深度相机	RGB-D 图像与视觉识别	保持镜头清洁，避免强逆光
T-mini Pro 雷达	二维激光扫描与 SLAM	避免碰撞和异物卡住旋转部件
轮毂电机与 RGB 车灯	运动驱动和模式指示	车灯颜色必须与当前模式一致
模式切换插销	四轮差速与阿克曼机械切换	切换后确认锁定到位
电池门与电源开关	电池安装、开关机和程序暂停	首次通电前确认电池接口牢固
海鸥门与 USB HUB	键盘、鼠标及调试设备接入	注意 HUB 总电流和多设备数量
扬声器与语音模块	语音交互（选配）	若未安装，语音章节仅作为扩展说明
升降杆控制接口	视觉触发升降杆（选配）	仅在车辆和升降杆完成联调后执行

1.6 功能亮点与车灯状态

LIMO 通过四个轮毂电机、快速切换轮系和集成传感器，在同一平台上覆盖教育演示、运动学实验、SLAM 与 Nav2 开发、视觉识别和可选机械臂研究。

•四模态快速切换：四轮差速、履带、阿克曼和麦克纳姆轮。•计算平台升级：Jetson Orin Nano 配合 ROS 2 Humble，适合更现代的节点与导航开发。•传感器集成：T-mini Pro 雷达、Dabai 深度相机、IMU、显示器和无线通信。•开放调试：侧门内提供 USB HUB，支持键盘、鼠标、相机和调试设备接入。•扩展开发：底盘接口可连接视觉、语音、升降杆与 MyCobot 机械臂等选配模块。

车灯状态	含义	处理建议
绿色常亮	阿克曼模式	确认插销拔起，软件和 launch 使用阿克曼入口
黄色常亮	四轮差速或履带模式	两种模态共用差速控制模型

车灯状态	含义	处理建议
蓝色常亮	麦克纳姆轮模式	确认四个麦克纳姆轮辊子朝向正确
红色常亮	程序暂停	按开关键恢复或检查节点状态
红色闪烁	低电量或主控报警	停止实验，检查电池和错误状态

1.7 外观、开关与电量指示

首次通电前，将车辆放在平整、无坡度的地面，收起外接线缆，确认轮胎和插销到位。车体后部开关用于开机与程序暂停，电量表用于快速判断是否具备正常实验条件。

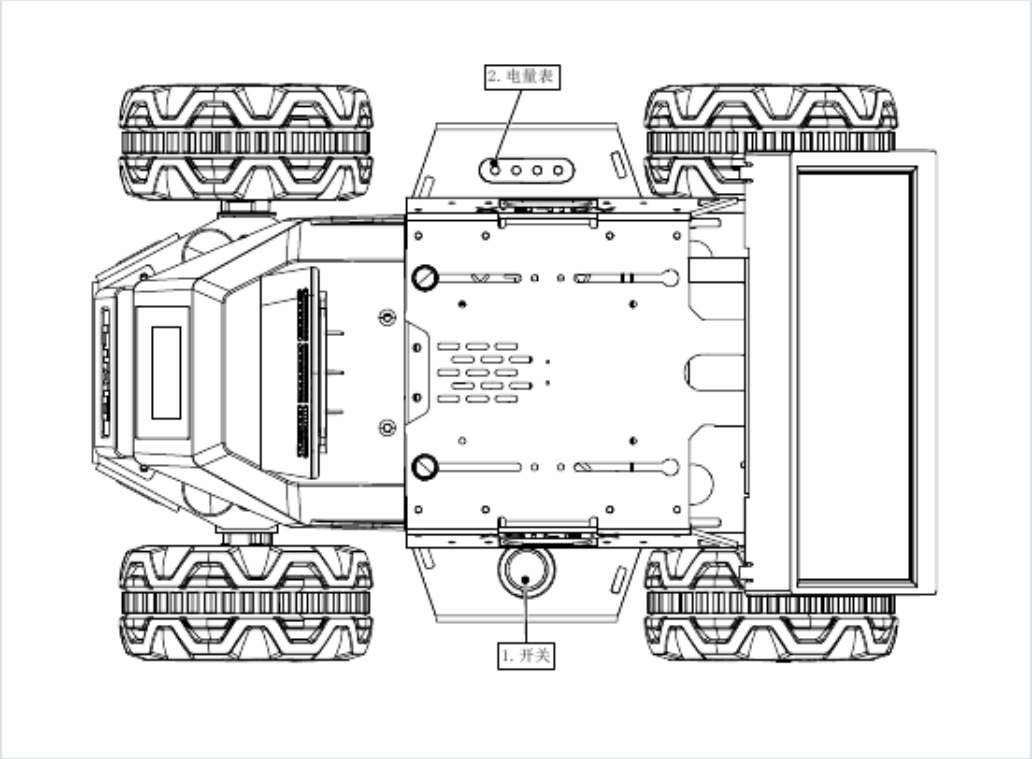


图 1-2 电量表与电源开关位置示意

1.检查电池接口、轮胎、插销和周围空间，确保车辆不会碰撞人员或障碍物。

2.长按开关键开机；短按用于暂停程序。观察电量表，末位红灯亮起时及时充电或更换电池。

3.观察前方插销和 RGB 车灯，确认底盘机械模态与 APP/遥控器所选模式一致。

4.首次测试使用低速，先验证前进、后退、左转、右转和停车，再进入建图或导航。

车灯状态	含义	处理建议
绿色常亮	阿克曼模式	确认插销已拔起，APP 或导航使用阿克曼入口
黄色常亮	四轮差速或履带模式	两种模态共用差速控制模型，按实际轮系操作
蓝色常亮	麦克纳姆轮模式	确认四个麦克纳姆轮的辊子朝向正确
红色常亮	程序暂停	按开关键恢复或检查程序状态
红色闪烁	低电量或主控报警	停止实验，检查电量与错误信息

1.8 四种运动模态与切换方法

四种模态共享底盘和轮毂电机，但机械姿态、转向方式与适用场景不同。模式切换完成后，必须同时核对车灯状态和软件模式，避免“机械已切换、APP 仍停留在旧模式”造成的异常运动。

模态	主要特点	典型场景	局限
四轮差速	四轮驱动，可原地转向，控制简单	通用实验、室内 SLAM	滑移转向，轮胎磨损较大

模态	主要特点	典型场景	局限
履带	越野能力强，适合小台阶和复杂地面	坡面、非结构化地面	磨损和阻力较大，侧门需抬起
阿克曼	前轮转向，接近汽车运动模型	自动驾驶、轨迹研究	转弯半径有限，需要零点校准
麦克纳姆轮	可横移、斜行与全向运动	狭小空间、全向控制	辊子接触不连续，平地要求较高

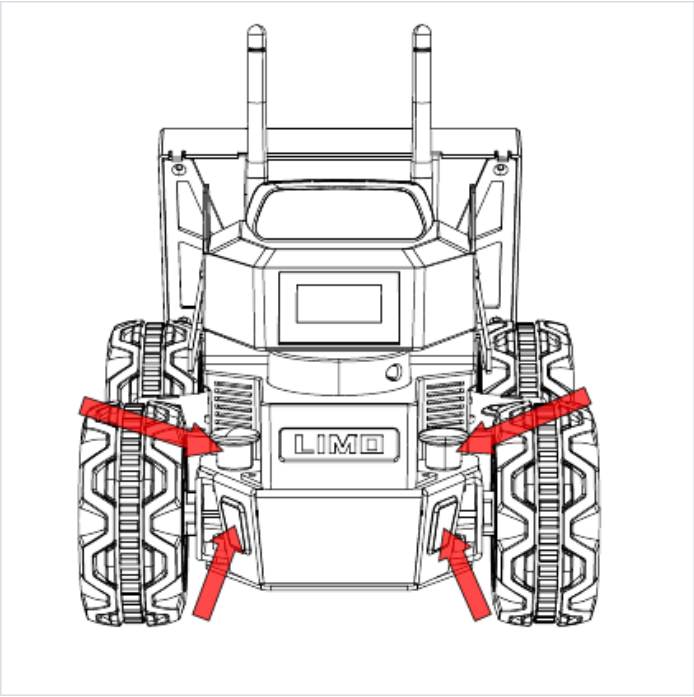


图 1-3 四轮差速/阿克曼模式切换插销位置

模态切换步骤

•阿克曼：拔起两侧插销，顺时针旋转约 30 度，使较长指示线朝前并锁定；车灯变绿后确认切换成功。**•四轮差速：**旋转至较短指示线朝前，调整轮胎孔位后将插销插入；车灯变黄后确认切换成功。**•履带：**先在四轮差速状态安装履带，建议先套后轮；抬起两侧侧门，防止行驶时剐蹭。**•麦克纳姆轮：**拆下轮毂盖和普通轮胎，保留轮毂电机；使四个辊子朝向车体中心，用 M3 x 5 螺丝安装，再在 APP/遥控器中切至对应模式。

切换后复核

车灯只说明底盘当前识别的模态。开始遥控、建图或导航前，还要确认 APP/遥控器模式、机械轮系、launch 文件三者一致。

1.9 APP_Agilex 与首次移动

APP_Agilex 用于蓝牙连接、运动控制和模式选择。不同手机系统的安装入口可能不同，请使用包装资料或官方渠道提供的版本。蓝牙的有效控制距离约 10 m，实际距离受墙体、人体遮挡和手机天线影响。

1.开机后打开 APP_Agilex，进入蓝牙扫描页面。

2.选择以 LIMO_ 开头的设备并完成连接。出现连接失败时，先关闭其他已连接的手机或遥控程序。

3.选择与底盘机械模态一致的控制模式。首次移动建议使用低速，确认左右方向后再提高速度。

4.完成实验后先停车，停止 ROS 2 节点，关闭 APP 连接，再关闭车体电源。

操作区	作用	首次操作建议
左侧摇杆	前进、后退与速度控制	小幅度推动，确认车辆方向
右侧摇杆	转向或原地旋转	先短时轻推，观察转向方向
模式选择	阿克曼、四轮差速、麦克纳姆等	与机械插销和轮系保持一致
速度设置	调整控制杆速度范围	调试和建图时采用较低速度
蓝牙管理	扫描、重连或修改设备名称	重命名后需重新连接

导航时的控制模式

启动 Nav2 或 RTAB-Map 导航后，应让 ROS 2 节点接管速度控制。底盘不需要在 APP 上切换模式：limo_base 启动时会向底盘下发 0x421 帧自动进入指令模式，APP_Agilex 上没有"指令模式"按钮；导航期间应断开 APP 蓝牙，或让它停在非控制页面。不要在导航运行期间用 APP 摇杆持续接管车辆，否则可能产生冲突指令。

1.10 首次联网与登录

LIMO 的远程操作使用 RDP 共享桌面和 SSH 终端两条通道：图形界面通过 Windows 远程桌面查看，ROS 2 命令通过 SSH 执行。同一局域网内可多设备同时连接，详细步骤见第 9 章。

访问方式	适用场景	关键点
RDP 共享桌面	图形界面、RViz2 与多设备查看	双击 AgileX 共享桌面.rdp；192.168.0.242:3389；使用部署密码
SSH 终端	运行命令、传文件、查看日志	ssh agilex@192.168.0.242；GUI 需设置 DISPLAY 环境
RDP+SSH	多人协同、边看界面边执行命令	RDP 看界面，SSH 启停节点；统一控制权，避免重复启动

小车 IP 由路由器分配，可能变化：在车端执行 hostname -I 查看当前地址，并据此更新 AgileX 共享桌面.rdp 和 SSH 命令。

首次登录提示

首次连接桌面时需要小车已开机并完成系统启动。若连接后黑屏，先在远程窗口里移动鼠标或按一下 Shift 唤醒；仍无画面说明车上的桌面会话没有起来（例如未接显示器且没有登录过），到车上点亮屏幕登录一次再连。

1.11 快速上手流程

阶段	要完成的操作	通过标准
1. 通电前检查	电量、外接线缆、轮胎、插销、周围空间	无松动、无短路风险、运行区无障碍
2. 通电与模式	开机，核对车灯与 APP_Agilex 模式	三者与预期一致
3. 低速移动	前进、后退、左转、右转、停车	方向正确，无异常噪声和报警
4. 通信检查	终端检查节点、话题和电池状态	驱动节点稳定，/scan 与 /limo_status 正常
5. 进入实验	建图、导航或视觉功能	上一实验进程已关闭，场地满足安全要求

第 2 章 安全与底盘电气信息

LIMO 使用锂电池、轮毂电机和高速旋转的激光雷达。安全操作的重点是防止电池过放或过流、避免车辆挤压与跌落，并确保切换模态后由专人复核。发现异味、异常发热、冒烟或红色报警时，应立即停止实验并断开电池。

2.1 电池基本信息

项目	额定参数	项目	额定参数
典型容量	10 Ah	最小容量	10 Ah
标称电压	11.1 V	充电截止电压	12.6 V
放电截止电压	8.25 V	最大持续放电电流	10 A
推荐存储温度	-25 至 45 摄氏度	充电环境	0 摄氏度以上

电池有两个接口：黄色为输出接口，黑色为充电接口。运输状态下电池不一定满电，这是存储安全要求，不代表异常。长期不使用时，将电池与车体断开。

> **电池红线**
>
> 禁止拆解、短接、加热、投入火中或在高温环境中存储；禁止使用未认证充电器；禁止在电池几乎耗尽后才长期存放。出现鼓包、漏液、异味或异常发热时停止使用。

2.2 充电流程

1.先关闭车辆并取出电池，将黄色输出接口与车体分离。

2.将充电器接头连接到电池充电接口，再接通充电器电源。

3.根据指示灯确认充电状态：红灯为充电中，绿灯闪烁为即将充满，绿灯为已充满。

4.充满后先断开电池与充电器，再断开充电器电源。

指示灯	电池状态	操作
红色	充电中	保持通风，不覆盖充电器
绿色闪烁	即将充满	等待涓流充电完成
绿色	已充满	结束充电并断开接口

标准充电器为 12.6 V / 5 A。电池从放电截止状态充至接近满电约需 2.5 h，进入绿色状态后通常还会以约 0.1 A 涓流充电约 0.5 h。实际时间受电量、温度与电池状态影响。

> **禁止边充边用**
>
> 充电时必须将电池与车体分离，不能一边充电一边给 LIMO 供电。

2.3 使用环境与机械安全

•工作温度为 -10 至 40 摄氏度，相对湿度建议为 30% 至 80%。•设备不防水，禁止在雨、雪、积水或高湿凝露环境使用。•不要在腐蚀性、易燃性气体附近使用，也不要靠近明火或热源。•建议海拔不超过 1000 m，昼夜温差不超过 25 摄氏度。•不要推动车辆倒转电机，防止逆向发电损坏主板或电池。•未经技术支持确认，不要私自改装内部结构、供电或安全保护。

风险	典型原因	预防措施
碰撞或夹伤	高速遥控、侧门展开、场地人员靠近	设置安全区，首次低速，急停人员在场
跌落或翻车	坡度过大、边界无防护、轮系不匹配	在平整地面开始，坡道与台阶实验加防护
电池过放	长期关机存放、低电继续使用	低电及时充电，长期不用断开电池
过流保护	堵转、强行推车、外加负载过大	清除障碍，确认负载不超过 4 kg
雷达损伤	碰撞、异物卡住旋转部件	固定传感器，发现异响立即断电

2.4 供电与通信拓扑

电池为底盘、Jetson 计算单元和传感器提供系统供电，最大输出电流按 10 A 设计；USB HUB 的三个 USB 接口合计输出能力有限，不应外接大功率负载。LIMO 底盘与 Jetson 通过 UART 通信，APP_Agilex 通过蓝牙与底盘交互，雷达和相机通常通过 USB 接口连接到计算单元。

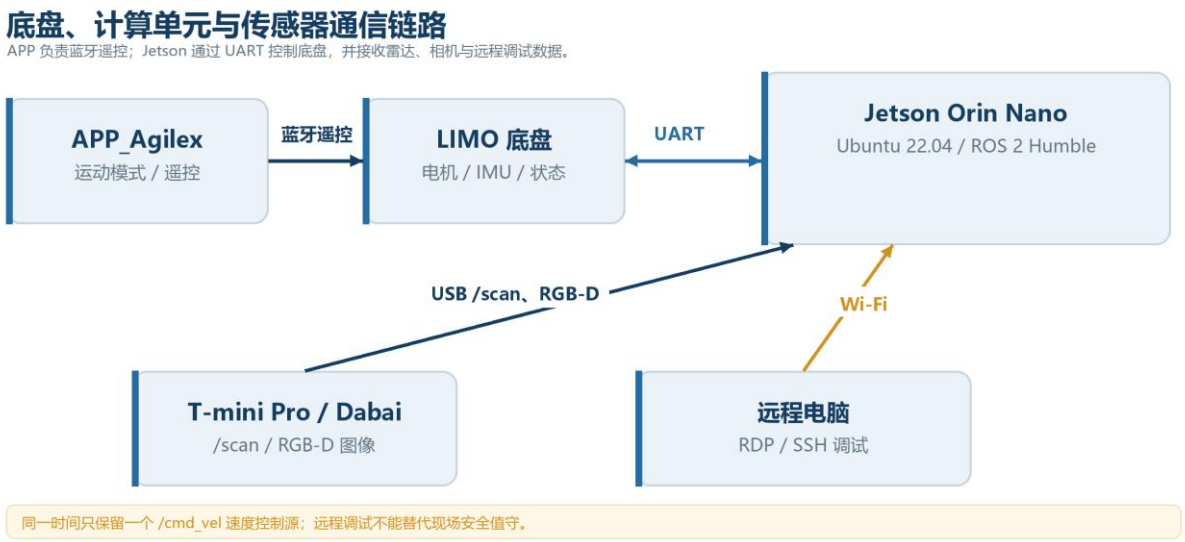


图 2-1 底盘、计算单元、传感器与远程调试链路示意

USB HUB 负载

HUB 端口用于键盘、鼠标、相机或调试设备时，应核算总电流。出现枚举失败、掉盘或重启时，先移除不必要的 USB 设备。

第 3 章 ROS 2 平台与底盘驱动

本配置的软件核心是 ROS 2 Humble。底盘驱动、雷达、相机、建图与导航由独立节点组成，通过话题、服务、动作和参数协作。开发前应建立“终端环境已 source、上一实验进程已关闭、当前模态与 launch 文件一致”的基本习惯。

3.1 系统分层与工作空间

层级	主要内容	作用
应用层	Cartographer、Nav2、RTAB-Map、limo_visions、trt_yolo	建图、定位、规划、识别与交互
通信层	话题、服务、动作、TF2、参数	连接各节点并同步传感器与控制数据
驱动层	limo_base、ydlidar_ros2_driver、orbbec_camera、aruco_ros	将硬件数据转换为 ROS 2 消息
系统层	Ubuntu 22.04、ROS 2 Humble、Jetson Orin Nano	运行环境、计算与硬件接口

```
~/agilex_ws/
  src/
    limo_ros2/
      limo_base/
      limo_msgs/
      limo_bringup/
    limo_visions/
    trt_yolo/
  install/
  build/
  log/
```

启动前环境

新开终端后，先加载 ROS 2 和本机工作空间。若提示找不到 limo_bringup、limo_base 或相关接口包，优先检查 setup.bash 是否加载，而不是直接修改包名。

```
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash
ros2 pkg list | grep -E 'limo|ydlidar|orbbec'
```

3.2 limo_base 驱动包

底盘驱动主要位于 ~/agilex_ws/src/limo_ros2/limo_base，用于串口连接、速度控制、里程计、IMU、车辆状态和 TF 发布。limo_msgs 提供配套的消息、服务与动作接口。

```
cd ~/agilex_ws/src/limo_ros2/limo_base
├── CMakeLists.txt
├── include/
├── launch/
│   ├── limo_base.launch.py
│   ├── open_ydlidar_launch.py
│   └── start_limo.launch.py
├── package.xml
├── scripts/
│   └── tf_pub.py
└── src/
    ├── limo_base_node.cpp
    ├── limo_driver.cpp
    ├── serial_port.cpp
    └── tf_pub.cpp
```

目录	内容	开发用途
include	头文件与驱动接口	阅读通信协议、串口与驱动结构
launch	节点启动文件	选择底盘、雷达或组合启动入口
scripts	Python 辅助脚本	TF 发布等辅助功能
src	C++ 源代码	底盘节点、驱动与串口实现

驱动核心函数与职责

函数或模块	作用
connect / serial_port	连接底盘串口并维护通信状态
readData / processRxData	读取串口数据并处理接收缓冲区
parseFrame / sendFrame	解析和封装底盘协议帧
setMotionCommand	设置速度、转向和运动模式命令
enableCommandedMode	使能程序控制，切换为 ROS 2 指令模式
publishOdometry	发布轮式里程计与位姿信息
publishLimoState	发布电池、模式、错误码等车辆状态
publishIMUData	发布 IMU 加速度、角速度与姿态
twistCmdCallback	接收 /cmd_vel 并转换为底盘控制量
processErrorCode	识别底盘错误和保护状态
角度换算函数	处理度、弧度与内外转角转换

函数名称可能随源码版本调整，修改前应先搜索节点中的版本号与头文件引用。禁止在未理解单位和坐标系时直接修改串口协议或控制模式枚举。

3.3 limo_msgs 接口

LimoStatus.msg

```
std_msgs/Header header
uint8 vehicle_state
uint8 control_mode
float64 battery_voltage
uint16 error_code
uint8 motion_mode
```

该消息用于观察底盘状态、控制模式、电池电压、错误码和运动模式。通过 /limo_status 可以在实验前确认车辆是否处于预期状态。

limo_msgs 实际提供的接口

```
# 确认接口包内容
ros2 interface list | grep limo_msgs
ros2 interface show limo_msgs/msg/LimoStatus
```

本镜像的 limo_msgs 只提供 msg/LimoStatus.msg；LimoSrv.srv 与 LimoAction.action 并不存在，limo_base 也不提供业务 service 或 action（只有 ROS 2 参数服务）。下发速度用 /cmd_vel（geometry_msgs/msg/Twist），读取状态用 /limo_status；需要自定义服务或动作时，应在自己的接口包里新建并编译。

3.4 驱动启动与话题检查

```
# 终端 A：启动底盘驱动
ros2 launch limo_base limo_base.launch.py
# 终端 B：查看节点、话题与状态
ros2 node list
ros2 topic list
ros2 topic echo /limo_status --once
```

话题	类型或用途	开发提示
/cmd_vel	geometry_msgs/msg/Twist	发布线速度与角速度，控制底盘运动
/imu	sensor_msgs/msg/Imu	提供加速度、角速度和姿态信息
/limo_status	LimoStatus	查看车辆状态、电量、错误码与运动模式
/parameter_events	rcl_interfaces/ParameterEvent	ROS 2 参数变化事件
/rosout	日志消息	检查节点日志与错误输出
/tf、/tf_static	TFMessage	坐标变换与静态安装关系
/odom	nav_msgs/msg/Odometry	轮式里程计与位姿估计

> **命令行控制示例**
>
> 以下命令只用于空旷场地的低速测试。车辆必须落地、周围无人，并已确认当前驱动和模态。

```
ros2 topic pub --once /cmd_vel geometry_msgs/msg/Twist \
  "{linear: {x: 0.10, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
# 发布停车命令
ros2 topic pub --once /cmd_vel geometry_msgs/msg/Twist \
  "{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
```

开发检查顺序

1.确认节点存在，且没有同一底盘驱动重复启动。

2.确认 /cmd_vel、/odom、/scan、TF 等所需话题持续发布。

3.用 /limo_status 查看电池、模式和错误码，排除底盘报警。

4.先低速发布短时命令并立即停车，再接入算法节点。

3.5 Python 开发与速度控制

本配置的底盘主驱动为 C++ 节点，Python 开发通过 ROS 2 的标准消息接口完成，不需要绕过驱动直接访问串口。只要 Python 节点按 geometry_msgs/msg/Twist 发布 /cmd_vel，就可以复用底盘驱动、Nav2 和视觉跟踪节点的控制链路。

```
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
class LimoPublisher(Node):
    def __init__(self):
        super().__init__('limo_python_demo')
        self.pub = self.create_publisher(Twist, '/cmd_vel', 10)
        self.count = 0
        self.timer = self.create_timer(0.1, self.tick)
    def tick(self):
        msg = Twist()
        msg.linear.x = 0.10
        msg.angular.z = 0.0
        self.pub.publish(msg)
        self.count += 1
        if self.count >= 10:
            self.pub.publish(Twist())
            self.destroy_timer(self.timer)
def main():
    rclpy.init()
    node = LimoPublisher()
    rclpy.spin(node)
    rclpy.shutdown()
if __name__ == '__main__':
    main()
```

开发方式	适用场景	关键接口
Python + rclpy	算法验证、订阅图像、发布速度	/cmd_vel、/limo_status、/scan

开发方式	适用场景	关键接口
C++ + rclcpp	高频控制、驱动二次开发	底盘驱动源码与自定义消息
ROS 2 launch	一次启动多个节点与参数	launch.py 与 YAML 配置
可视化调试	检查节点、TF、图像与地图	RViz2、rqt_image_view、rqt_graph

> **Python 控制安全**
>
> 首次运行将前进命令改为 0.05-0.10 m/s，并保证程序在异常退出前发布停车消息。不要让多个速度发布者同时控制车辆。

从脚本到 ROS 2 节点的验证顺序

1.启动底盘驱动，并在终端检查 /cmd_vel、/odom 和 /limo_status。

2.运行只发布零速度的 Python 节点，确认没有其他程序同时控制底盘。

3.将线速度改为 0.05-0.10 m/s，执行一个消息周期后立即发布停车。

4.使用 ros2 topic echo /cmd_vel 对照实际发布值，再接续视觉或导航算法。

```
# 运行 Python 节点前加载环境
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash
# 检查是否已有速度发布者
ros2 node list
ros2 topic info /cmd_vel --verbose
```

第 4 章 底盘运动学与四模态控制

四模态使用同一个四轮轮毂电机底盘，但转向约束、轮地接触方式和控制输入不同。运动学章节用于解释为什么不同模态需要不同的速度组合，以及出现偏航、打滑或轨迹误差时应该检查哪些物理量。所有公式均以车辆中心或后轴参考点为基准，实际符号方向应与驱动坐标约定一致。

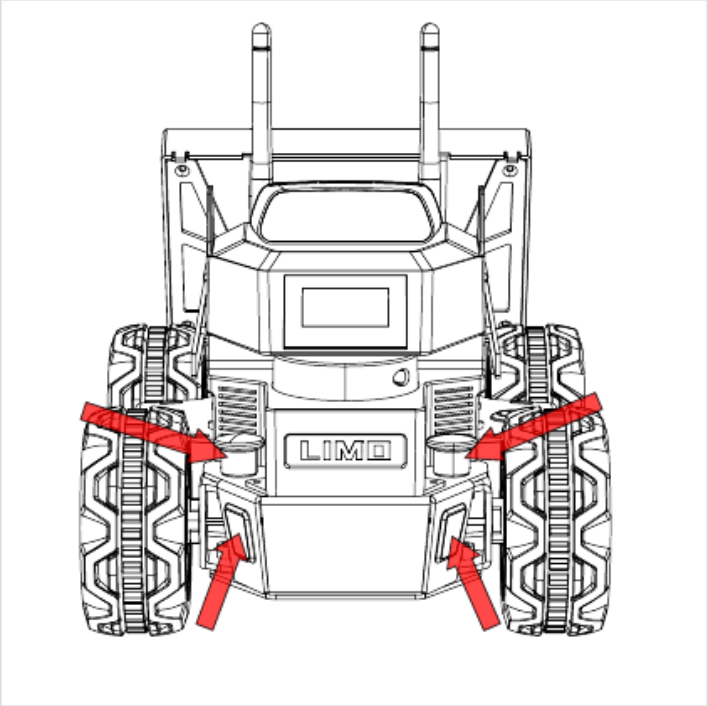


图 4-1 模式切换插销与四轮布置关系

4.1 四轮差速运动模型

四轮差速通过左右两侧车轮的速度差转向。直线运动时左右速度相同，原地旋转时左右速度大小相同、方向相反。四轮独立驱动但同侧车轮目标速度通常保持一致。

量	关系	说明
整车线速度 v	v = (vL + vR) / 2	vL、vR 为左右侧等效线速度
整车角速度 omega	omega = (vR - vL) / L	L 为左右轮等效轮距
左轮速度	vL = v - omega L / 2	根据期望 v、omega 反算
右轮速度	vR = v + omega L / 2	差速转向的核心关系
原地旋转	vR = -vL	v 近似为 0，omega 不为 0

在 ROS 2 中通过 /cmd_vel 的 linear.x 与 angular.z 表达运动意图，驱动节点完成轮速分配。不要直接把轮速写到串口，否则会绕过模式检查、限速和错误保护。

```
# 低速直线
ros2 topic pub --once /cmd_vel geometry_msgs/msg/Twist \
  "{linear: {x: 0.08}, angular: {z: 0.0}}"
# 低速原地转向
ros2 topic pub --once /cmd_vel geometry_msgs/msg/Twist \
  "{linear: {x: 0.0}, angular: {z: 0.25}}"
```

差速模式的限制

原地旋转会造成明显轮胎滑移和磨损。实验应短时、低速执行，完成测试后发布零速度。

4.2 履带运动模型

履带模式可以采用与四轮差速相同的左右速度差模型。不同点在于履带与地面的接触长度更大，转向需要剪切地面，摩擦阻力、侧向约束和磨损均高于轮式差速。

•直行时可把左右履带看作速度相同的两组连续接触单元。•转向依靠左右履带速度差，转弯中心通常存在滑移，不能等同于前轮转向。•硬质地面长时间原地旋转会加剧磨损，建议采用较大半径的连续转弯。•履带模式沿用差速 launch；安装履带时必须抬升侧门，防止结构剐蹭。

场景	建议参数	原因
室内平地首次测试	线速度不超过 0.08 m/s	便于观察干涉和履带跑偏
台阶或坡面	先人工评估，低角速度接近	避免重心变化导致翻车
狭小空间	短时差速转向	履带滑移使急转更容易擦碰边界

4.3 阿克曼运动模型

阿克曼模式通过前轮转角改变车辆航向，内外前轮的转角不同，使各车轮运动方向尽量围绕后轴延长线上的同一瞬时转向中心。此模式更接近汽车运动学，适合轨迹跟踪和自动驾驶实验。

几何关系	含义	工程关注
转弯半径 R	参考点至瞬时转向中心的距离	LIMO 最小转弯半径约 0.4 m
轴距 Lw	前轴与后轴距离	影响相同转角下可达的转弯半径
轮距 W	左右轮中心距离	决定内外轮的转角差
内轮与外轮	内侧轮胎走更小半径	前轮连杆机构自动产生角度差
前轮转角 delta	tan(delta) 与车长和半径相关	小速度时近似成立，高速需考虑侧偏

•直线行驶需要左右前轮回正；发现车辆持续向一侧偏移时应检查舵机零点。•启动导航必须使用阿克曼入口，不能只修改 /cmd_vel 参数代替机械切换。•高速急转会导致侧向滑移，实验优先采用低速大半径轨迹。•如果机械插销拔起但车灯仍为黄色，应停止运动并回落插销重新切换。

4.4 麦克纳姆轮运动模型

麦克纳姆轮由轮毂和斜向被动辊子组成，四个独立驱动的轮子可以在不发生转向的前提下产生纵向、横向和旋转运动的组合。辊子的旋向和安装方向必须严格成对，否则横向速度分量会相互抵消。

运动意图	四个轮子的速度形态	结果
前进/后退	四轮同向转动	产生纵向线速度
左右横移	对角轮同向、另一对角反向	产生横向线速度
原地旋转	两侧轮系反向组合	产生角速度，车体中心平移近零
斜向运动	纵向与横向速度叠加	车体按合成方向平移

四轮速度分配依赖轮距、轴距、辊子安装角度和正方向定义。驱动源码中必须保持同一套符号约定；若只调换其中一个轮子的接线而不改速度矩阵，会出现横移方向错误、原地旋转中心和漂移。

> **安装检查**
>
> 将每个麦克纳姆轮的小辊子朝向车体中心，使用原厂指定螺丝固定。安装完成后先架空测试四个轮子的转向，落地后再低速验证横移和旋转。

4.5 四模态选择建议

模态	优点	局限	优先用途
四轮差速	控制简单、适应性强	转向滑移，轮胎磨损	SLAM、导航通用实验
履带	接地面积大，通过性好	阻力与磨损大，侧门需抬起	坡面、小台阶、非结构地面
阿克曼	接近汽车模型，轨迹平顺	转弯半径有限，需要零点标定	轨迹跟随、自动驾驶研究
麦克纳姆	可全向移动，机动性好	辊子接触不连续，平地要求高	狭小空间、全向控制

第 5 章 激光雷达建图

本配置使用 EAI T-mini Pro 360 度二维激光雷达。激光建图的目标是构建可被 Nav2 使用的二维栅格地图。建图质量不仅取决于算法，还取决于初始位姿、行驶速度、场地反光、动态行人和里程计稳定性。

5.1 雷达参数与数据

项目	最小	默认	最大	说明
测距频率	-	4000 Hz	-	每秒 4000 次测距
扫描频率	6 Hz	6 Hz	12 Hz	通常采用 6 Hz
测距范围	0.02 m	-	12 m	室内、80% 反射率条件
扫描角度	-	0-360 度	-	二维平面扫描
测距精度	-	20 mm	-	0.05-12 m 范围
角度分辨率	-	0.54 度	-	数据密度参考

雷达数据发布在 /scan 话题，消息类型为 sensor_msgs/msg/LaserScan。常见字段包括 angle_min、angle_max、angle_increment、range_min、range_max、ranges 与 intensities。

```
# 单独启动雷达
ros2 launch ydlidar_ros2_driver ydlidar.launch.py
# 新终端启动 RViz2，添加 LaserScan 并选择 /scan
rviz2
```

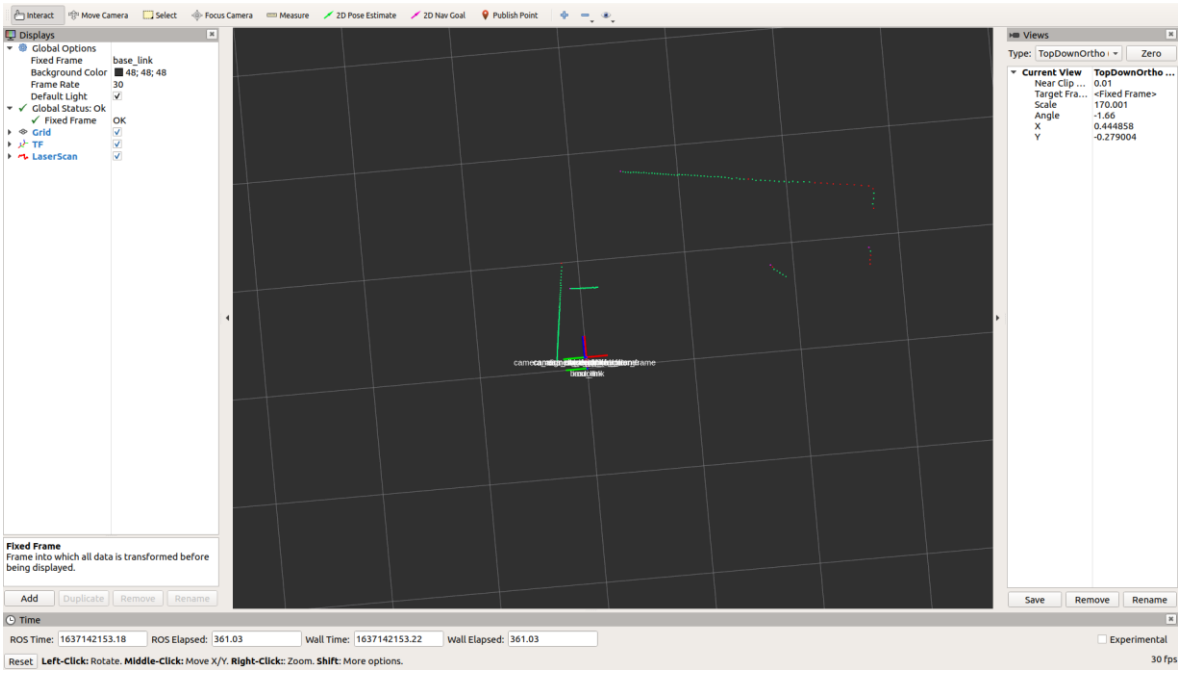


图 5-1 RViz2 中显示 /scan 的绿色激光点

不要重复启动雷达

集成入口 limo_start.launch.py 可能已经包含雷达驱动。若已运行组合启动，不要再运行单独的 ydlidar.launch.py，否则可能出现串口占用或重复话题。

5.2 GMapping 建图

GMapping 是基于粒子滤波的二维 SLAM 方法，常把轮式里程计与激光扫描结合，适合面积较小、结构相对清晰的室内场景。使用前先确认本机镜像是否提供对应的 ROS 2 GMapping 包与组合 launch（如 slam_gmapping、limo_gmapping.launch.py）；没有则直接使用本手册的 Cartographer 方案。

检查项	含义	建议
scan topic	输入激光数据	应为 /scan 或 launch 中重映射后的话题
odom topic	轮式里程计	检查 /odom 与 TF 连续更新
particles	粒子数量	资源充足时可提高，过大将增加 CPU 负载
delta	栅格地图分辨率	常用 0.05 m，过小会增大内存和计算量
maxRange	有效激光距离	应小于或等于雷达可靠测距范围

```
# 先检查当前镜像是否提供 GMapping
ros2 pkg list | grep -E 'gmapping|slam'
ros2 pkg executables slam_gmapping
# 若 limo_bringup 提供组合入口
ros2 launch limo_bringup limo_start.launch.py
ros2 launch limo_bringup limo_gmapping.launch.py
# 检查地图与激光
ros2 topic echo /map --once
ros2 topic hz /scan
```

镜像差异

若 ros2 pkg list 中没有 slam_gmapping 或 limo_gmapping.launch.py，说明当前镜像未集成该建图方式。不要用其他 ROS 版本的启动文件强行启动；可直接使用本手册后续的 Cartographer 章节。

5.3 Cartographer 原理

Cartographer 是基于图优化的 SLAM 系统，可拆分为 Local SLAM 与 Global SLAM。Local SLAM 用连续激光帧建立并维护子图 Submap；Global SLAM 在子图之间执行回环检测和位姿优化，减少长距离行驶后的累积误差。

模块	输入	输出	调参关注
Local SLAM	激光帧、里程计、IMU	局部子图与短期轨迹	扫描频率、运动速度、传感器标定
Global SLAM	已完成子图	回环约束与优化轨迹	回环半径、计算资源、场景相似度

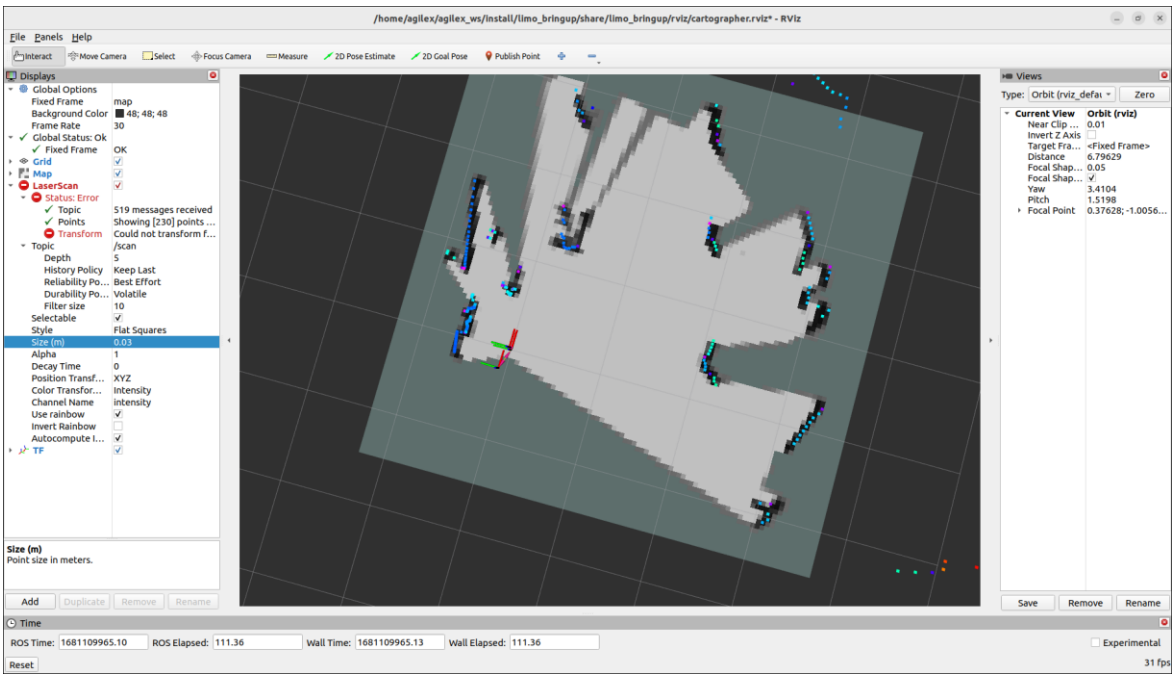


图 5-2 Cartographer 建图界面示例

5.4 Cartographer 建图操作

> **开始前**
>
> 清空场地障碍，保持正常室内光照；关闭上一次建图、导航和雷达进程。首次行驶采用低速，避免急转、原地高速旋转和人员穿行。
>
> 1.终端 A 启动底盘与雷达组合功能。2.终端 B 启动 Cartographer，等待 RViz2 打开。

3.在 RViz2 中检查 /scan、地图、TF 和机器人模型，确认初始位置无跳变。

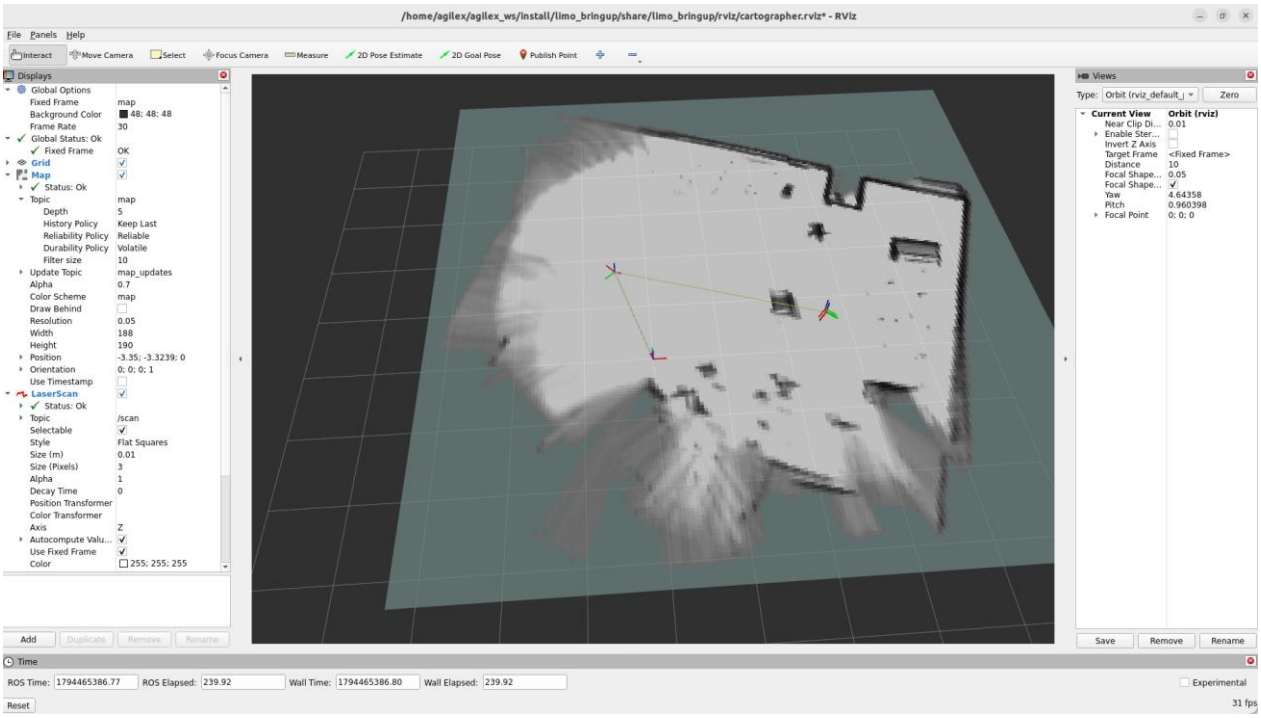


图 5-3 Cartographer 建图完成后的栅格地图与激光点重合效果

图为 Cartographer 运行约 4 分钟（面板 ROS Elapsed 239.92 s）时的 RViz2 画面：Fixed Frame 为 map，浅灰是已建栅格，白色小方块是当前帧 /scan。

第一组判据在左侧 Map 面板：Topic 为 map、Resolution 0.05、Width 188、Height 190，即约 9.4 m × 9.5 m、每格 5 cm；这两个数随场地和行驶范围变化，不是固定值（短距离试跑可能只有 91 × 127）。第二组判据看激光与墙线：白色点应紧贴灰色墙体边缘，出现双层墙或明显错位说明回环不足或里程计漂移，按 5.6 节降速并回到已见区域闭环。第三组看状态：Fixed Frame 无报错、LaserScan 的 Topic 为 /scan；若 RViz2 提示 map 不存在，先查 limo_cartographer.launch.py 日志里有没有 Inserted submap。

激光点完全不显示时，先把 LaserScan 的 Reliability Policy 改成 Best Effort：本机 /scan 的 QoS 是 BEST_EFFORT，RViz2 默认的 Reliable 收不到。

4.使用 APP_Agilex 低速遥控，先沿边界，再覆盖内部区域；尽量闭环回到起点。

5.地图边界稳定、回环重合后停车，保存地图。

```
# 终端 A：底盘与雷达
ros2 launch limo_bringup limo_start.launch.py
# 终端 B：Cartographer
ros2 launch limo_bringup limo_cartographer.launch.py
# 终端 C：保存地图
mkdir -p ~/agilex_ws/src/limo_ros2/limo_bringup/maps
cd ~/agilex_ws/src/limo_ros2/limo_bringup/maps
ros2 run nav2_map_server map_saver_cli -f map01
```

保存后通常得到 map01.pgm 与 map01.yaml。两个文件必须放在同一目录并保持文件名对应。YAML 中的 image 字段应指向实际 PGM 文件，分辨率通常由建图参数决定。

5.5 RRT 自主探索建图

RRT 通过随机采样和快速扩展树在未知区域生成探索路径，机器人可在人工设定边界后自动选择前沿目标并逐步覆盖环境。RRT 不是基础底盘功能，需要导航、地图更新、目标管理和探索节点同时就绪。

1.先验证人工遥控建图，确认地图、/scan、TF 和 Nav2 状态正常。

2.启动探索节点，在 RViz2 中设置四个环境边界点和一个搜索起始点。

3.让系统低速探索，观察机器人是否反复访问同一区域或进入狭窄死路。

4.覆盖完成后停止探索节点，保存最终地图并检查边界是否闭合。

```
# 检查探索包是否已集成
ros2 pkg list | grep -E 'rrt|exploration'
# 示例入口：实际 launch 名称以本机 rrt_exploration 包为准
ros2 launch limo_bringup limo_start.launch.py
ros2 launch limo_bringup limo_cartographer.launch.py
ros2 launch limo_bringup limo_nav2_diff.launch.py
ros2 launch rrt_exploration rrt.launch.py
```

> **探索安全**
>
> 自主探索可能主动进入未知区域和狭窄通道。实验场地应有实体边界，首次测试保持低速，人员不得站在车辆行进方向或雷达盲区内。

5.6 建图质量检查

现象	可能原因	处理
墙体双层或错位	速度过快、回环不足、里程计漂移	降速、回到已见区域闭环、检查轮径参数
地图断续	激光遮挡、反光、TF 中断	调整路线，检查 /scan 与 TF 频率
原点附近轨迹跳变	初始位姿未稳定或驱动重复	关闭所有进程后按单一入口重启
地图边缘像扇面	转弯过快或扫描频率不匹配	低速转弯，检查扫描频率和计算负载

第 6 章 ROS 2 激光导航与路径巡检

LIMO ROS 2 配置采用 Nav2 导航栈。导航系统需要地图、定位、传感器、底盘控制和目标管理共同工作；本节按 ROS 2 Humble 与 Nav2 的术语和命令描述系统。

6.1 Nav2 组件

组件	作用	调试关注
Map Server	加载 PGM/YAML 地图	地图路径、分辨率、原点
AMCL	在地图中估计机器人位姿	初始位姿、粒子分布、激光与地图重合
Planner Server	生成全局路径	代价地图、膨胀半径、路径是否合理
Controller Server	执行局部轨迹与控制	速度限制、障碍距离、振荡与恢复
Behavior Server	处理旋转、后退、等待等恢复行为	卡住时是否自动恢复
BT Navigator	组织导航行为树	单点、多点、失败重试流程

> **参数以本机为准**
>
> 本手册按 ROS 2 Humble 和 Nav2 统一表述；实际启用的 planner/controller 插件及参数，请以本机 launch 与 YAML 配置为准。

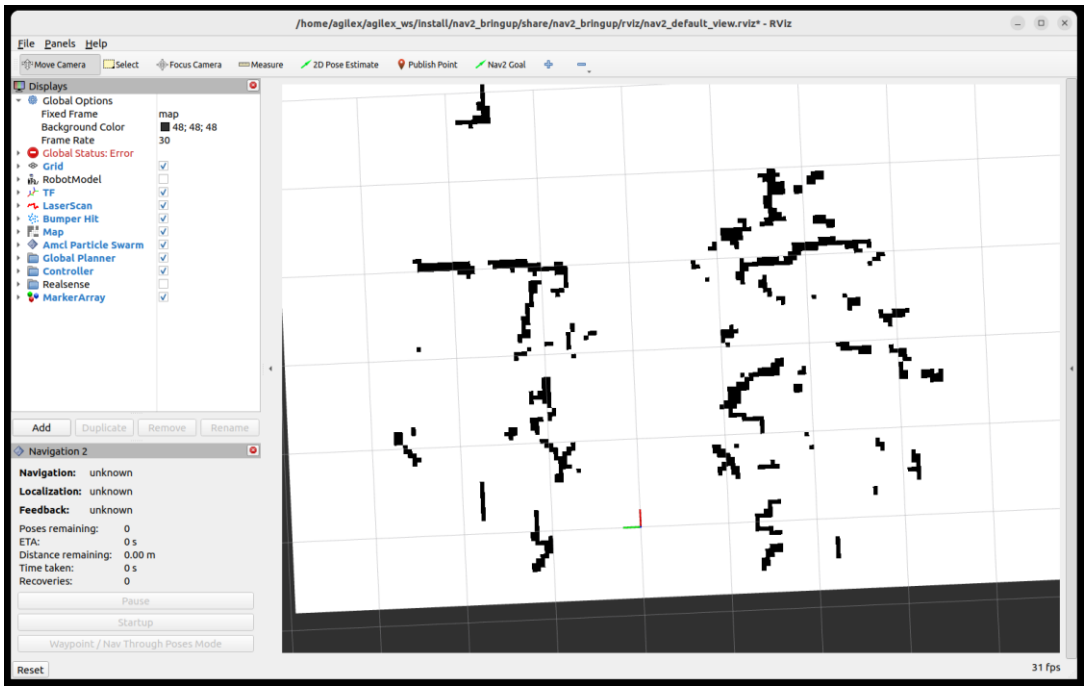


图 6-1 Nav2 与 RViz2 导航界面示例

6.2 启动导航

先启动底盘与雷达，再启动与当前运动模态匹配的导航 launch。四轮差速、履带和麦克纳姆通常使用差速控制入口；阿克曼必须使用阿克曼入口，不能混用。

```
# 通用终端 A
ros2 launch limo_bringup limo_start.launch.py
# 四轮差速 / 履带 / 麦克纳姆
ros2 launch limo_bringup limo_nav2_diff.launch.py
# 阿克曼
ros2 launch limo_bringup limo_nav2_ackermann.launch.py
```

地图配置路径

需要更换地图时不必修改任何文件，直接在导航 launch 后追加参数覆盖，例如 ros2 launch limo_bringup limo_nav2_diff.launch.py map:=$HOME/map.yaml。本配置默认加载 limo_bringup/maps/map0818.yaml；地图文件在 install/limo_bringup/share/limo_bringup/maps 下，以本机实际文件为准。

6.3 建立初始位姿

1.在 RViz2 中确认地图、/scan、TF、机器人模型与代价地图显示正常。

2.使用 2D Pose Estimate 在地图上给出车辆的位置与朝向。

3.低速遥控车辆旋转或小范围移动，使激光点与墙体轮廓重合。

4.在粒子云收敛、轮廓基本重合后再发布导航目标。

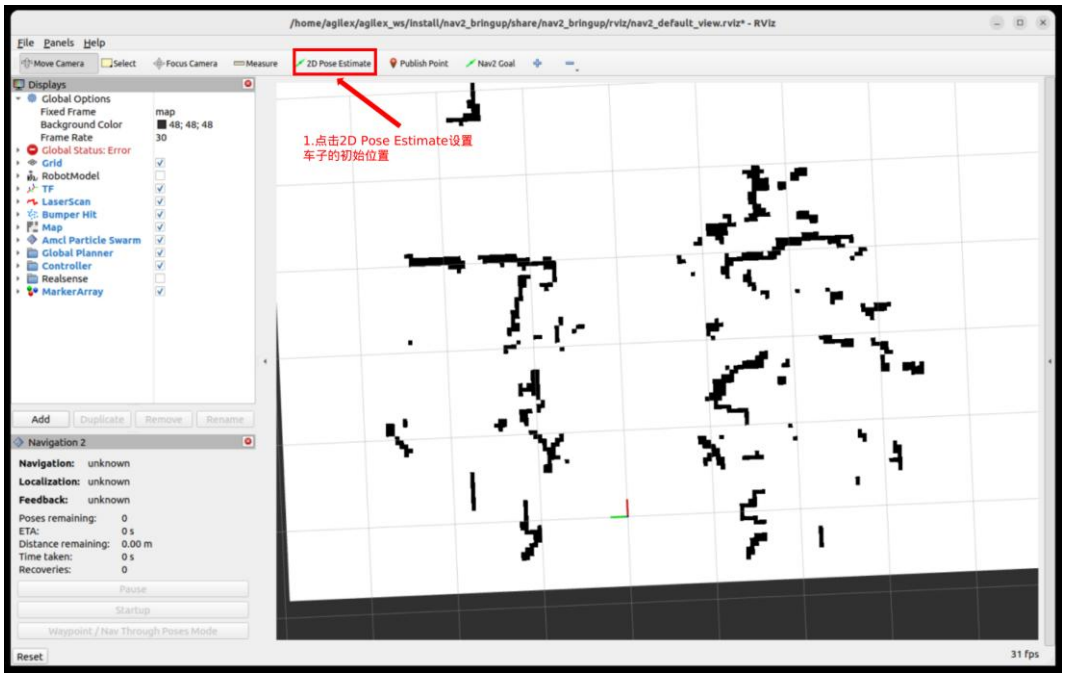


图 6-2 通过初始位姿与激光轮廓完成地图对齐

如果激光与地图始终不重合，不要用导航目标强行测试。先检查地图是否为当前场地、起始位置是否与建图坐标一致、TF 是否完整，以及手机摇杆是否仍处于手动接管状态。

6.4 单点与多点导航

点击 Nav2 Goal，在地图上选择目标位置与朝向；系统将生成全局路径和局部控制轨迹。车辆开始运动后，保持底盘处于指令模式（limo_base 启动时已通过 0x421 进入），并断开 APP_Agilex 蓝牙或让它停在非控制页面，避免手机摇杆与导航节点同时发布速度。

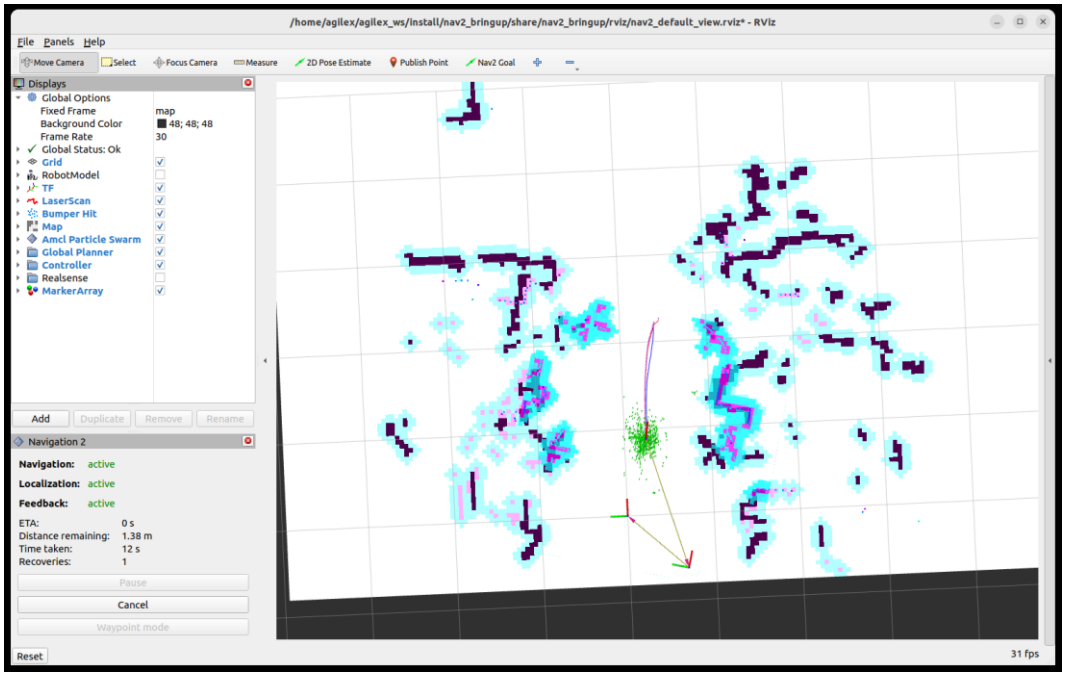


图 6-3 多点导航与开始执行按钮示例

•全局路径应避开墙体与膨胀区域；若穿过障碍，先检查地图和代价地图参数。•局部路径频繁摆动时，降低速度、检查雷达遮挡和控制器参数。•卡住后观察行为树是否触发旋转、后退或等待恢复。•多点导航前先测试单点，确认每个途经点位于可通行区域。

检查项	正常现象	异常处理
定位粒子	集中在机器人附近并逐渐稳定	重新设置初始位姿或检查地图
激光与地图	轮廓基本重合	检查坐标系、地图版本和起始位置
全局路径	连续且避障	检查地图、代价地图与目标点
底盘响应	速度平滑、无明显振荡	降低速度并检查控制器参数
APP 状态	指令模式，不持续手动摇杆	切换模式后重新发送目标

6.5 导航前安全检查

1.清空路径与人行通道，确保急停人员能够迅速接近车辆。

2.确认电量足以完成路线，避免低电时速度限制或突然停止。

3.核对地图版本和当前场地，移走建图后新增的临时障碍。

4.从短路线和低速参数开始，确认目标朝向和停车行为后，再执行多点任务。

6.6 路径记录与巡检

路径巡检先录制一次人工驾驶路线，再由 limo_path_patrol 节点按 CSV 复跑。巡检不使用 Nav2，也不使用 /wheel/odom；录制和跟随命令都必须显式指定 odom_topic:=/odom。

> 本次实机结果
>
> 新路线已经录制并完成基础校验，但自动巡检启动后出现“有速度指令、里程计不动”。/limo_status 显示约 8.9 V、error_code=3，低两位表示低电量保护。应先关机充电，不能用提高速度或放宽路线阈值绕过故障。

6.6.1 运行边界与接口

同一时间只允许一个 /cmd_vel 发布者。巡检期间不要启动 MyCobot GUI、listen_real_service、mycobot_adapter、Nav2、手机 App、键盘遥控或其他会控制底盘的程序。

功能	实际接口	说明
底盘节点	/limo_base_node	接收 /cmd_vel，发布 /odom 和 /limo_status
雷达节点	/ydlidar_ros2_driver_node	发布 /scan
里程计	/odom	路径录制和跟随都必须使用该话题
雷达	/scan	跟随节点的前方障碍保护输入
速度控制	/cmd_vel	同一时间只能保留一个发布者

> 不要使用 /wheel/odom
>
> 本车里程计话题是 /odom。所有 record_path 和 follow_path 命令都应加入 odom_topic:=/odom；/wheel/odom 在本车上不存在。

6.6.2 启动前检查

每次上电后先检查底盘状态、里程计、雷达和速度发布者。

检查项	通过标准	异常处理
/limo_status	control_mode=1、error_code=0，电压明显高于低电量范围；正常约 12 V	关机充电或检查底盘报警
/odom	持续更新	重启底盘驱动并检查串口
/scan	持续更新	重启雷达并检查 USB 连接
/cmd_vel	录制前为 0 或仅有键盘遥控；巡检期间仅有 follower	关闭多余控制程序

```
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash

ros2 topic echo /limo_status --once
ros2 topic echo /odom --once
ros2 topic echo /scan --once
ros2 topic info /cmd_vel --verbose
ros2 node list | grep limo_path_patrol
ps -eo pid,cmd | grep '[f]ollow_path'
```

正常情况下只能存在一个 /limo_path_patrol_follower。重复启动 follower 会让两个节点同时创建 start、stop 服务和 /cmd_vel 发布者，可能导致控制指令冲突。

> 低电量判定
>
> battery_voltage 约 8.9 V 且 error_code=3 时，立即停止测试、关机充电。error_code=3 = 0x0001 | 0x0002，不是路线参数错误。

6.6.3 录制并校验路线

步骤 1：启动底盘和雷达。在终端 A 执行并保持运行，确认 /limo_base_node 和 /ydlidar_ros2_driver_node 已出现。

```
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash
ros2 launch limo_bringup limo_start.launch.py
```

```
ros2 node list | grep -E 'limo_base|ydlidar'
```

步骤 2：开始录制新路线。在终端 B 执行。新录制结果会覆盖旧路线，录制器把第一条 /odom 样本作为路线原点。

```
mkdir -p ~/.ros/limo_path_patrol
ros2 launch limo_path_patrol record_path.launch.py \
  path_file:="$HOME/.ros/limo_path_patrol/lab_route.csv" \
  odom_topic:=/odom
```

看到 First odometry sample received; route started. 后再开始驾驶。CSV 中的 x、y、yaw 都是相对该原点的位置和朝向。

步骤 3：启动键盘遥控。在终端 C 执行。连续按 q 降低速度，不要使用默认高速录制。

```
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

按键	功能
i	前进
逗号 (,)	后退
j	左转
l	右转
k	立即停车
q	降低整体速度

步骤 4：驾驶并录制路线。把小车放在固定起点、摆正车头，先按 i 直线前进一小段，再开始正常转弯；到达终点后按 k 停车。

- 保持低速、连续和平滑，不突然加速、不急转弯。
- 开始记录后不要先按 j 或 l 原地旋转，也不要横穿已经走过的路线。
- 路线周围不要有新增固定障碍；第一段应是从静止直接缓慢前进。

步骤 5：保存路线。回到终端 B 按 Ctrl+C。只有看到 Saved ... points, route length ... m 才表示保存完成；没有 Saved 时不要继续巡检。

步骤 6：关闭键盘遥控。回到终端 C 按 Ctrl+C，再执行 ros2 topic info /cmd_vel --verbose，确认 teleop_twist_keyboard 已消失。

步骤 7：校验路线文件。以下脚本只读取 CSV，不会让小车运动。

```
python3 - <<'PY'
import csv
import math
from pathlib import Path

path = Path.home() / ".ros/limo_path_patrol/lab_route.csv"
with path.open(newline="", encoding="utf-8-sig") as f:
    rows = [(float(r["x"]), float(r["y"]), float(r["yaw"]))
            for r in csv.DictReader(f)]

if len(rows) < 2:
    raise SystemExit("FAIL: route has fewer than 2 points")

steps, turns = [], []
for a, b in zip(rows, rows[1:]):
    steps.append(math.hypot(b[0] - a[0], b[1] - a[1]))
    turns.append(abs(math.atan2(math.sin(b[2] - a[2]),
                                math.cos(b[2] - a[2]))))

route_length = sum(steps)
start_end = math.hypot(rows[-1][0] - rows[0][0],
                       rows[-1][1] - rows[0][1])
max_step = max(steps)
max_turn = max(turns)

cumulative = 0.0
initial_rotation = 0
for step, turn in zip(steps, turns):
    if cumulative > 0.20:
        break
    if step < 0.005 and turn > math.radians(3.0):
        initial_rotation += 1
    cumulative += step

print(f"points={len(rows)}")
print(f"route_length_m={route_length:.3f}")
print(f"start_end_distance_m={start_end:.3f}")
print(f"step_min_max_m={min(steps):.3f}..{max_step:.3f}")
print(f"max_turn_deg={math.degrees(max_turn):.1f}")
print(f"initial_rotation_segments={initial_rotation}")

failures = []
if len(rows) < 20:
    failures.append("too few points")
if route_length < 1.0:
    failures.append("route too short")
if max_step > 0.20:
    failures.append("large position jump")
if initial_rotation > 0:
    failures.append("rotation-only start")
if failures:
    print("FAIL:", ", ".join(failures))
    raise SystemExit(1)
print("PASS: route meets basic patrol checks")
PY
```

校验指标	本次结果
路线点数	160
路线长度	8.691 m
起点到终点距离	0.996 m
相邻点间距	0.050–0.060 m
最大相邻点转角	3.9°
起步原地旋转片段	0
结论	PASS

6.6.4 启动路径跟随与自动巡检

步骤 8：启动路径跟随节点。继续使用终端 B。该步骤只加载路线并等待服务，小车不会运动。

```
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash
ros2 launch limo_path_patrol follow_path.launch.py \
  path_file:="$HOME/.ros/limo_path_patrol/lab_route.csv" \
  odom_topic:=/odom \
  linear_speed:=0.05
```

参数	本次值	要求
linear_speed	0.05 m/s	首次复跑固定使用低速
obstacle_distance	0.35 m	前方障碍小于该距离自动停车
max_cross_track_error	0.60 m	偏离路线过大自动停车
goal_tolerance	0.15 m	到达终点附近自动结束
autostart	false	防止节点启动后立即运动

步骤 9：把小车放回录制起点。地面位置和车头方向要与录制开始时一致，并确认路线周围没有新增障碍、键盘遥控已经关闭、/cmd_vel 只剩 follower。

> 起点判定
>
> follow_path 在调用 start 服务时会把当前位姿作为路线新原点。即使小车被搬回去、/odom 数值没有回到录制时的数值，只要地面位置和车头方向正确，仍可正常开始。

步骤 10：通过服务启动巡检。在终端 C 执行。成功时返回 success=True 和 Patrol started.；节点等待约 2 秒后开始发布 /cmd_vel。

```
ros2 service call /limo_path_patrol/start \
  std_srvs/srv/Trigger "{}"
```

步骤	作用	小车是否运动
步骤 8：启动 follow_path	读取 CSV，连接 /odom、/scan，创建服务并等待 start	不运动
步骤 10：调用 start 服务	记录当前起点，重置路线进度，等待 2 秒后发布 /cmd_vel	开始运动

6.6.5 前 10 秒检查与停止

巡检不是调用 start 后就结束。启动后应立即检查运动指令和里程计是否同时变化。

```
ros2 topic echo /cmd_vel --once
ros2 topic echo /odom --once
sleep 3
ros2 topic echo /odom --once
ros2 topic echo /limo_status --once
```

观察结果	判断与处理
/cmd_vel 约 0.05 m/s，/odom 位置连续变化	继续巡检并观察终点停车
/cmd_vel 持续非零，但 /odom 数秒不变	立即调用 stop、发布零速度并检查 /limo_status
/limo_status 显示 8.9 V、error_code=3	低电量保护：关机充电，不要提高速度
没有 /cmd_vel 或 scan is stale	停止巡检，检查 follower 日志、雷达和重复进程

```
ros2 service call /limo_path_patrol/stop \
  std_srvs/srv/Trigger "{}"

ros2 topic pub --once /cmd_vel geometry_msgs/msg/Twist \
  "{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
```

正常停止时调用 stop 服务即可，follower 程序仍然保留，可再次调用 start。紧急停止时优先按物理急停，也可以在终端 B 按 Ctrl+C 结束 follow_path，再调用 stop 并发布零速度。

error_code=3 的低位定义如下：0x0001 和 0x0002 均为低电量；0x0004 为遥控器失联；0x0008、0x0010、0x0020、0x0040 分别为 1–4 号电机驱动错误；0x0100 为驱动状态错误。

常见现象	优先检查	处理
/wheel/odom 没有数据	实际里程计是 /odom	所有命令加入 odom_topic:=/odom
出现 Cross-track error	起点或车头方向不一致	回录制起点对齐车头，必要时重录
录制后没有 Saved	是否按 Ctrl+C，点数是否至少为 2	重新录制并保存
/cmd_vel 有多个发布者	Nav2、teleop、App、重复 follower	只保留一个巡检发布者
有 /cmd_vel 但车不动	/limo_status 电压、错误码、控制模式	error_code=3 时关机充电
Laser scan is stale	雷达节点和 /scan	重启雷达并检查数据
No odometry received	/odom	重启底盘并检查串口
start 返回 already running	上次巡检未 stop 或 follower 重复	先 stop，再检查 follower 数量
第一次复跑偏离	起点方向不对或路线开头旋转	重录路线，不要放宽 0.60 m 阈值

6.6.6 验收与本次恢复

一条可自动复跑的路线应从固定起点开始，车头方向固定，开始记录后先直线前进，低速连续驾驶，不原地旋转、不急转弯、不横穿原路线，并且航线周围没有新增固定障碍。路线文件不要手工编辑，重新录制比修改 CSV 更可靠。

阶段	通过标准
录制前	环境已加载；/limo_base_node、/ydlidar_ros2_driver_node、/odom、/scan 正常；control_mode=1、error_code=0；没有多余 /cmd_vel 发布者
录制后	出现 Saved；CSV 至少 20 个点、路线长度大于 1 m、最大相邻点间距不超过 0.20 m、起步没有原地旋转
自动巡检前	键盘遥控已关闭；/cmd_vel 只有 limo_path_patrol_follower；只有一个 follower；小车回到录制起点并摆正车头；人员和急停就位
自动巡检后	start 返回 success=True；/cmd_vel 约 0.05 m/s；/odom 位置连续变化；到达终点或调用 stop 后及时停车

本次恢复步骤：小车已经关机充电，充电完成后按以下顺序恢复。

- 开机，等待底盘和网络启动。
- source /opt/ros/humble/setup.bash 和 source ~/agilex_ws/install/setup.bash。
- 启动 limo_start.launch.py。
- 执行 ros2 topic echo /limo_status --once，确认 error_code=0、电压恢复正常。
- 确认 /odom 和 /scan 都有数据。
- 启动 follow_path.launch.py，保持 linear_speed:=0.05、autostart=false。
- 把小车放回录制起点并摆正车头。
- 关闭所有其他 /cmd_vel 发布者。
- 调用 start 服务。
- 前 10 秒确认 /cmd_vel 非零且 /odom 连续变化；若仍不变化，立即停止并重新检查 /limo_status。

第 7 章 深度相机与雷达融合建图导航

ORBBEC Dabai 提供彩色与深度信息。RTAB-Map 将 RGB-D 数据、里程计和可选激光数据结合起来，构建三维环境地图并支持视觉定位。与纯激光导航相比，RGB-D 方案对纹理和光照更敏感，因此需要控制车速、保持相机清洁并保证计算负载稳定。

7.1 Dabai 深度相机

项目	参数	项目	参数
深度距离	0.3-3 m	基线	40 mm
深度 FOV	H 67.9 度 / V 45.3 度	彩色 FOV	H 71 度 / V 43.7 度
深度分辨率	最高 640 x 400 at 30 FPS	彩色分辨率	最高 1920 x 1080 at 30 FPS
精度	1 m 处约 6 mm	延迟	30-45 ms
数据传输	USB 2.0 或以上	供电	USB
工作温度	10-40 摄氏度	尺寸	59.6 x 17.4 x 11.1 mm

```
# 启动相机
ros2 launch orbbec_camera dabai.launch.py
# 查看可用话题
ros2 topic list | grep -E 'color|depth|camera'
# 可视化图像
ros2 run rqt_image_view rqt_image_view
```

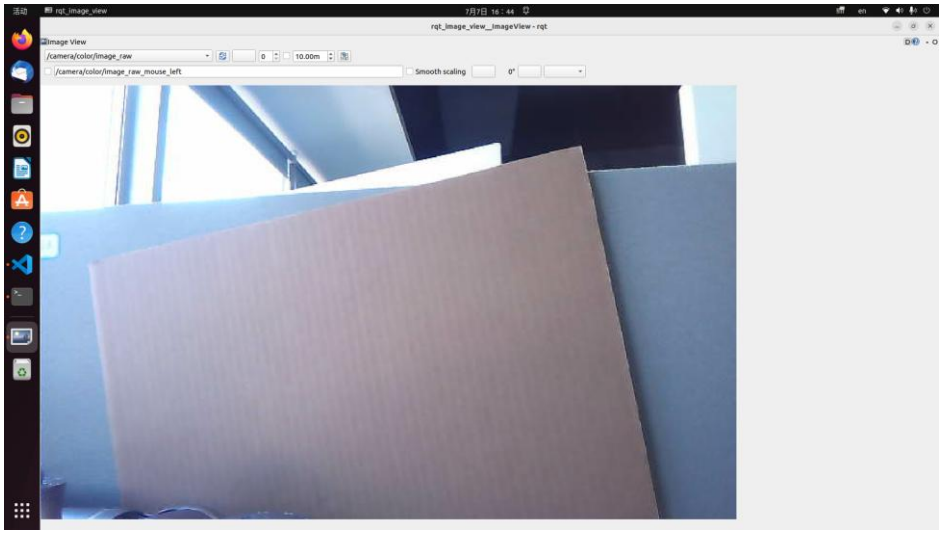


图 7-1 使用 rqt_image_view 检查相机画面

7.2 RTAB-Map 建图

RTAB-Map 的核心是外观与几何约束结合的地图管理。它通过关键帧、回环检测和图优化提高长路线的一致性，并可使用 RGB-D 数据构建稠密地图。计算资源不足时，算法会增加地图管理负担，因此应使用适当分辨率并控制处理频率。

数据库安全：limo_rtab_rgbd.launch.py 默认以 rtabmap -d 启动建图，会删除并重建 ~/.ros/rtabmap.db。需要重新建图时可直接执行；需要加载已有地图时，应先备份数据库，并使用 localization:=true，不能再次以默认建图模式启动。

```
# 终端 A：底盘与雷达
ros2 launch limo_bringup limo_start.launch.py
# 终端 B：深度相机
ros2 launch orbbec_camera dabai.launch.py
# 终端 C：RTAB-Map RGB-D 建图
ros2 launch limo_bringup limo_rtab_rgbd.launch.py
```

> **地图保存位置**
>
> 建图完成后正常结束节点。数据库通常位于 ~/.ros/rtabmap.db。~/.ros 是隐藏目录，可在文件管理器按 Ctrl+H 显示。
>
> 实验结束
>
> 前应把数据库复制到项目目录并备份。

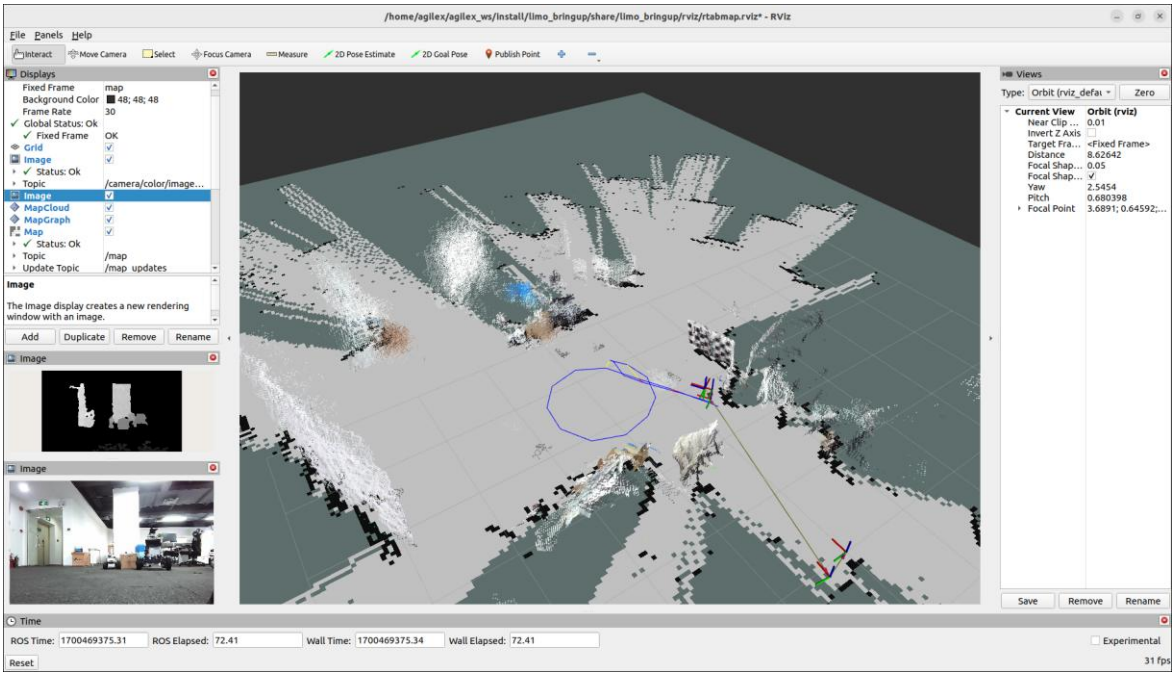


图 7-2 RTAB-Map 建图与地图可视化界面

7.3 RTAB-Map 定位导航

```
# 终端 A：底盘与雷达
ros2 launch limo_bringup limo_start.launch.py
# 终端 B：深度相机
ros2 launch orbbec_camera dabai.launch.py
# 终端 C：载入数据库并进入定位模式
ros2 launch limo_bringup limo_rtab_rgbd.launch.py localization:=true
# 终端 D：启动 Nav2 导航
ros2 launch limo_bringup limo_rtab_nav2.launch.py
```

使用视觉定位时，通常不需要像纯激光导航那样先在 RViz2 中手工校正二维位姿。启动 localization:=true 前必须确认 ~/.ros/rtabmap.db 是本次任务对应的数据库，并等待当前画面与地图定位状态稳定后再发送 Nav2 目标。

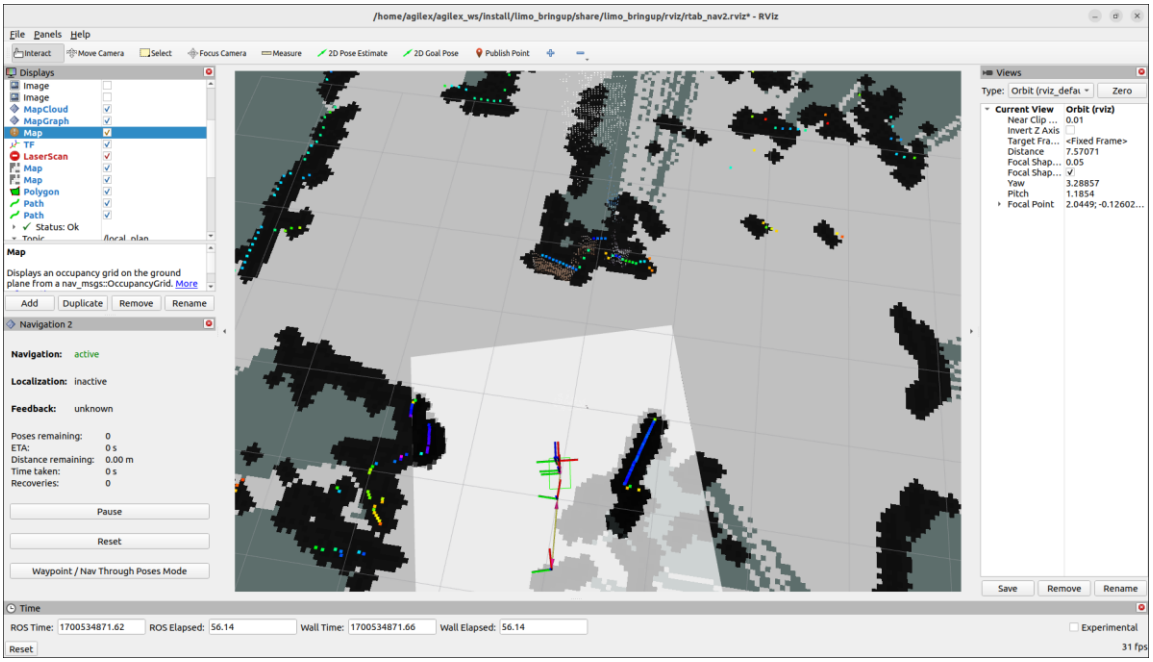


图 7-3 RTAB-Map 导航目标与路径显示

问题	检查方向
定位无法启动	数据库路径、数据库版本、launch 参数 localization:=true
当前位置识别错误	光照变化、纹理不足、相机遮挡、相似场景
地图更新慢或卡顿	降低相机分辨率或处理频率，关闭不必要节点
导航不启动	定位状态、Nav2 生命周期、TF 和地图是否就绪

第 8 章 视觉识别、巡线与交互应用

视觉节点均依赖 Dabai 相机。每次实验只启动一个相机驱动，并在新终端中检查图像话题后再启动识别节点。视觉算法受光照、目标尺寸、背景颜色和相机角度影响，示例命令用于功能验证，不应直接视为生产级参数。

8.1 通用准备

```
# 启动相机
ros2 launch orbbec_camera dabai.launch.py
# 检查图像话题
ros2 topic list | grep -E 'color|image'
# 可视化并选择图像话题
ros2 run rqt_image_view rqt_image_view
```

一次只运行一个相机驱动

若已运行 RTAB-Map 或其他包含相机的 launch，再启动 dabai.launch.py 可能造成设备占用。先确认当前进程，再决定单独启动还是复用已有相机节点。

8.2 文字识别

文字识别节点获取 RGB 图像并执行灰度化、二值化等预处理，再调用 tesseract 识别文字。需要注意：镜像自带的 C++ 节点 ros2 run limo_visions recognition 只发布标注图像 /image_hsv，不发布 /ocr/result；需要结构化结果时使用下面的前置安装与 Python 脚本（/ocr/result 为 JSON，/ocr/image 为带框图像）。提高成功率的要点是正对目标、提高对比度并减少反光。

```
# 前置：tesseract 与 Python 绑定（只需一次）
sudo apt install tesseract-ocr tesseract-ocr-chi-sim tesseract-ocr-eng
pip3 install --user pytesseract
# 终端 A：启动相机
ros2 launch orbbec_camera dabai.launch.py
# 终端 B：文字识别，输出 /ocr/result 与 /ocr/image
cd /home/agilex/agilex_ws/src/limo_visions/scripts
python3 limo_recognition_text_read.py
```

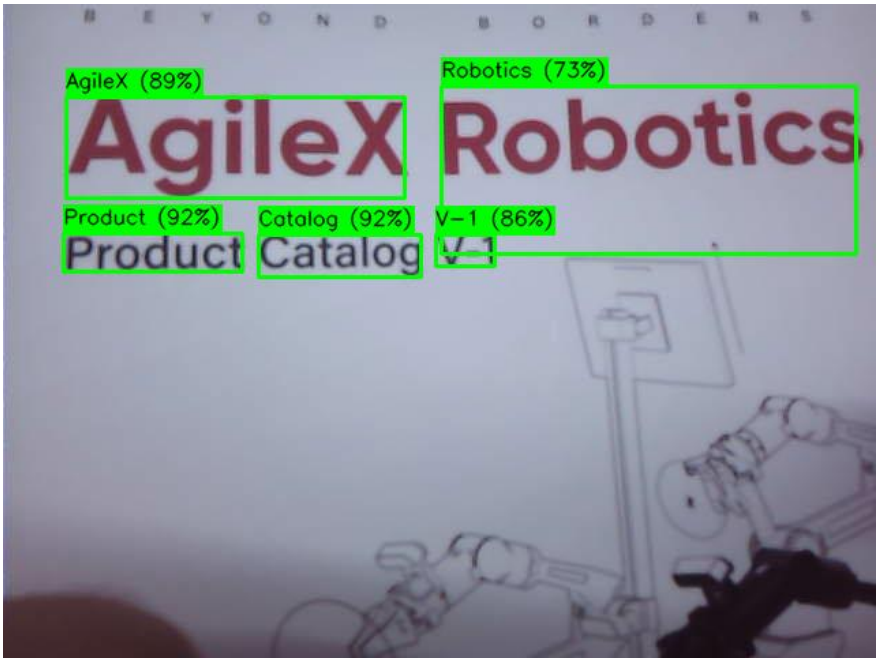


图 8-1 文字识别实验画面

8.3 颜色识别

颜色识别把相机 RGB 图像转换为适合颜色分割的空间，按预设阈值提取目标区域，再通过轮廓、边界框或中心点输出识别结果。它只判断画面中哪里出现指定颜色，不一定改变底盘速度。

```
ros2 launch orbbec_camera dabai.launch.py
ros2 run limo_visions recognition
```

调试点	影响	建议
HSV 色相范围	颜色类别是否被选中	在固定光源下取样重新标定
饱和度与亮度	阴影和反光是否误检	限制高亮区域并增加形态学处理
最小轮廓面积	小色块噪声	按目标实际像素大小设置阈值
相机曝光	颜色漂移和过曝	关闭自动曝光或锁定参数

8.4 视觉巡线（FollowLine）

本机 limo_visions 的 object_detect 可执行文件实际运行 FollowLine 节点。它订阅 /camera/color/image_raw，发布 /image_hsv 和 /cmd_vel。该节点不是只读的颜色识别工具，一旦识别到线路就可能直接驱动底盘；测试前必须架空车轮或清空场地，并确认没有其他速度发布者。

```
ros2 launch orbbec_camera dabai.launch.py
ros2 run limo_visions object_detect
# object_detect 即 FollowLine，会发布 /cmd_vel
```

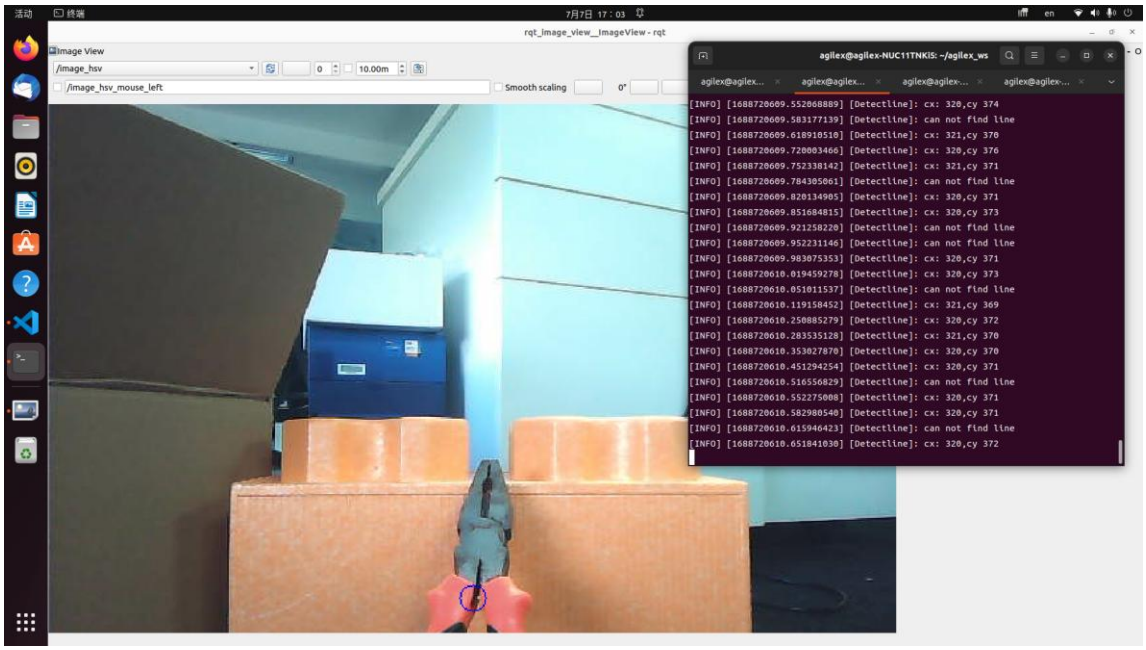


图 8-2 object_detect / FollowLine 视觉巡线节点

8.5 升降杆控制（选配）

选配升降杆控制通过相机识别升降杆上方的二维标记，估计车辆与升降杆的距离。当距离进入设定阈值时，视觉节点向 /chatter_updown 发布触发消息，使升降杆抬起，并从产生触发到车辆通过的短暂时间内保持开放。本机没有安装 detect_ros 时说明未配置升降杆，按选配跳过本节命令。

```
ros2 launch orbbec_camera dabai.launch.py
ros2 launch detect_ros agx_ar_pose.launch.py
# 若镜像未提供该 launch，可按包内实际可执行文件运行
ros2 run detect_ros agx_ar_pose
```

> **升降杆联调**
>
> 该功能涉及外部执行机构。必须由现场人员确认升降杆限位、供电和急停，不能仅根据图像中的触发值让车辆高速通过。

现象	可能原因	调参方向
左右摆动	比例增益过大	降低角速度比例项
偏离后回不来	增益过小或图像阈值错误	提高增益，检查线路掩模
转弯冲出路外	线速度过快、前瞻距离过短	降低速度，增加预测距离
完全无线	相机话题或 launch 名称不匹配	检查图像话题与包内 launch

8.6 ArUco 二维码识别与跟随

ArUco 标记由黑白几何编码组成，aruco_ros 负责检测标记并估计位姿。应保证标记平整、尺寸一致、周围有留白，且摄像头能完整看到标记。

```
ros2 launch orbbec_camera dabai.launch.py
ros2 launch aruco_ros single.launch.py
ros2 run limo_visions move_to_ar
```

> **标记测试**
>
> 打印标记时不要裁掉白色边界，也不要使用高反光纸张。训练前先检查标记 ID 与启动参数一致。

8.7 红绿灯识别

红绿灯节点将图像转换到 HSV 空间，按色相区间提取红色、黄色和绿色区域，再进行形态学处理、轮廓与外接圆检测，最后在图像上标出候选灯色，标注结果发布到 /image_hsv。环境光、曝光和红色背景是主要干扰因素。

```
ros2 launch orbbec_camera dabai.launch.py
ros2 run limo_visions traffic_light
```

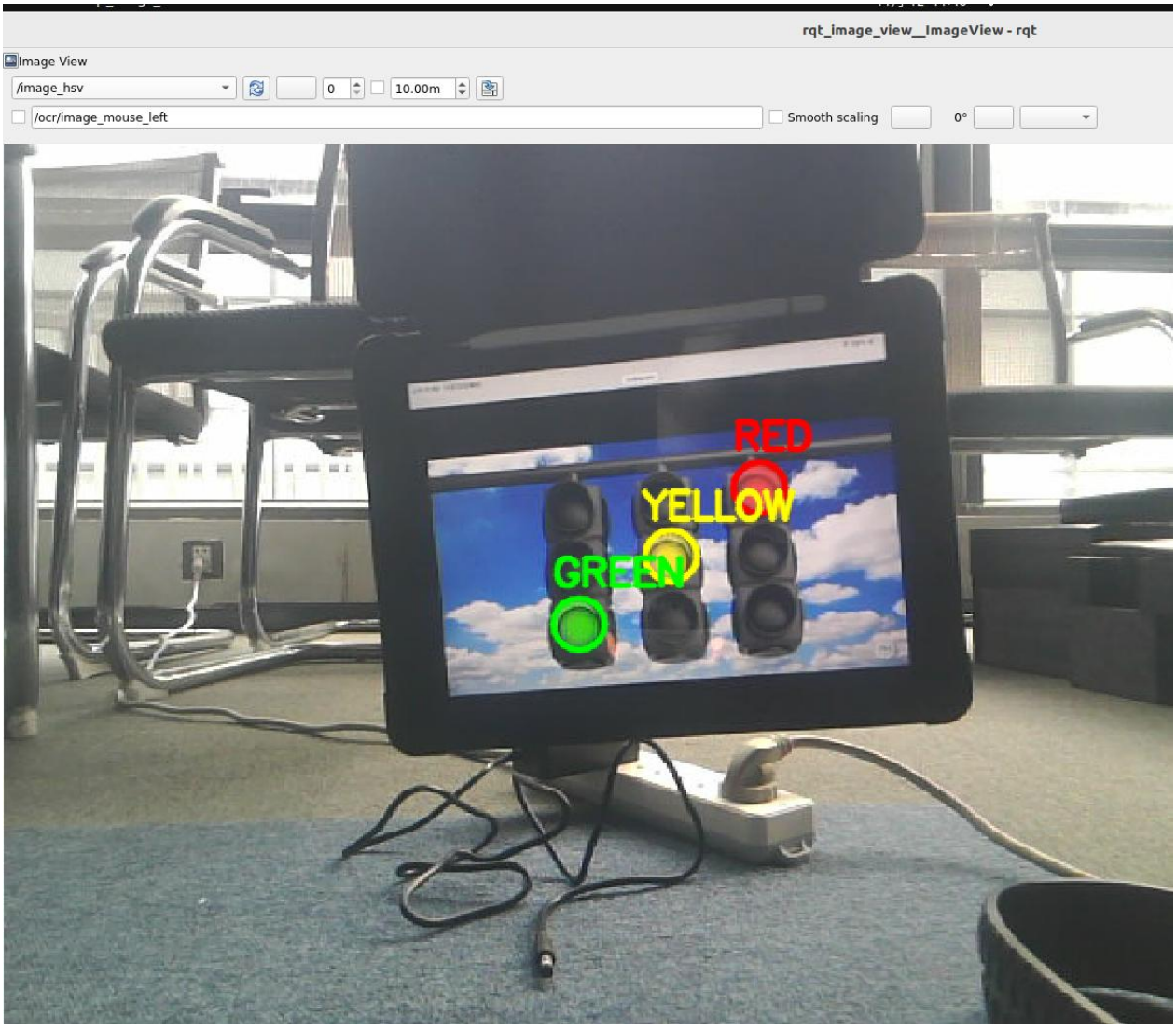


图 8-3 在 rqt_image_view 中查看 /image_hsv 的红绿灯标注结果

图中是在 rqt_image_view 里选择 /image_hsv 后看到的画面：同一帧里三个灯位分别被标成 GREEN、YELLOW、RED，说明颜色分割、轮廓与外接圆检测都在正常工作。

判读要点：圆圈应落在灯体上；圈落在背景上或某个灯位没有圈时，先调整 HSV 色相区间与最小轮廓面积，再锁定相机曝光、避免窗光直射。三个灯同时被标出只表示画面里出现了这三类颜色区域，是否按红灯停车要由上层逻辑再判一次。窗口第二行的 /ocr/image_mouse_left 是保存图片用的路径框，与本图判读无关。

8.8 Ultralytics YOLO 目标检测

Jetson Orin Nano 可使用 TensorRT 加速的 YOLO 节点进行对象检测。首次运行前应确认模型文件、类别配置和相机话题与 launch 文件一致。

```
ros2 launch orbbec_camera dabai.launch.py
ros2 launch trt_yolo yolov8n_det.launch.py
```

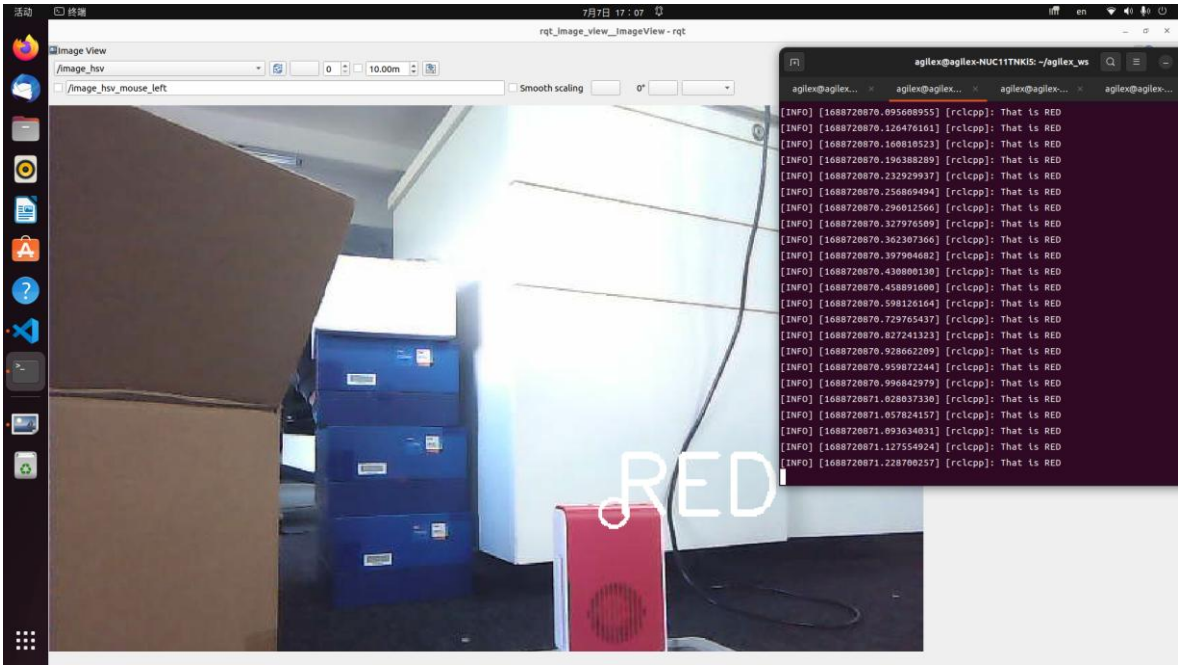


图 8-4 YOLO 实时对象检测界面

下图为另一组实测画面：在 rqt_image_view 中选择 /yolov8/detection_image 后，节点对椅子目标输出检测框和类别标签，可用于确认图像话题和检测结果是否正常。

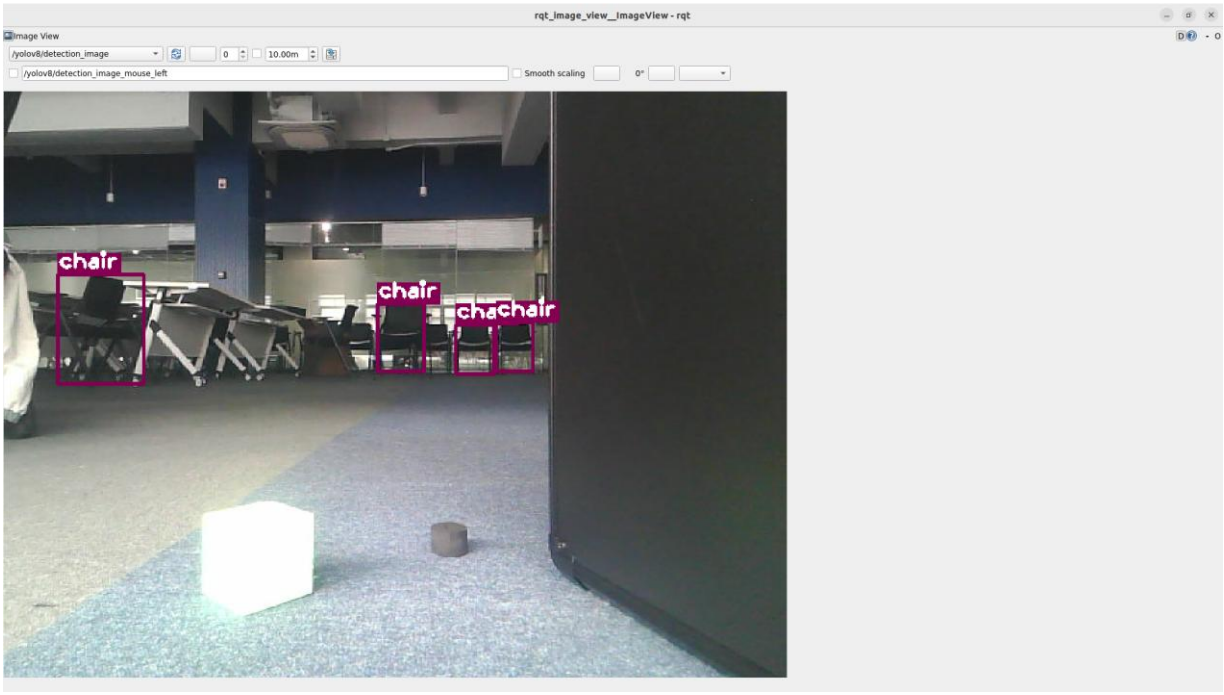


图 8-5 Ultralytics YOLO 对椅子目标的检测结果

应用	启动入口	观察结果
文字识别	limo_recognition_text_read.py	/ocr/result 与终端输出

应用	启动入口	观察结果
颜色识别	ros2 run limo_visions recognition	目标区域、分类或图像标注
视觉巡线	ros2 run limo_visions object_detect	线路中心、/image_hsv 与 /cmd_vel 控制
升降杆	ros2 launch detect_ros agx_ar_pose.launch.py	距离阈值与 /chatter_updown 触发
视觉巡线	ros2 launch limo_deeplearning follow_lane.launch.py	线路偏移与 /cmd_vel 控制
ArUco	aruco_ros single.launch.py + move_to_ar	标记 ID、位姿与跟随动作
红绿灯	ros2 run limo_visions traffic_light	图像中的灯色标记
YOLO	trt_yolo yolov8n_det.launch.py	检测框、类别与置信度

8.9 视觉综合实验建议

实验	输入	目标输出	安全要求
文字读取	正面标牌	识别文本与置信结果	车辆保持静止
颜色巡检	不同色块	颜色类别和目标中心	先验证颜色阈值，再接入速度
视觉巡线	连续线条	偏移、角速度和控制量	设置最大速度与出线停车

实验	输入	目标输出	安全要求
红绿灯联动	红绿目标	灯色状态与停车条件	识别到红灯必须优先停车
YOLO 检测	目标物体	类别、检测框与置信度	检测和规划解耦，避免误检直接急转

综合实验应先验证单一视觉输出，再允许节点发布 /cmd_vel。建议增加“超时未识别即停车”和“单条消息最大持续时间”两项保护，避免旧识别结果持续驱动车辆。

第 9 章 远程桌面、SSH 与多设备协同

LIMO 的远程调试由两条通道组成：RDP 共享桌面负责查看和操作 Ubuntu 桌面、RViz2 与图形工具，SSH 终端负责运行 ROS 2 命令、传输文件和查看日志。两条通道可以同时使用，也可以从多台电脑同时连接。

9.1 网络准备与共享桌面文件

小车和所有控制电脑必须连接到同一个可互通局域网。优先使用实验室内网，不要使用会隔离设备的访客 WiFi。如果小车 IP 发生变化，先在车端执行 hostname -I 获取地址，再更新共享桌面文件。

1. 在小车终端执行 hostname -I，记录当前局域网 IP。
2. 将 Windows 电脑连接到同一路由器或实验内网。
3. 双击 AgileX 共享桌面.rdp。当前配置默认指向 192.168.0.242:3389，用户名为 agilex。
4. 如果地址变化，右键 .rdp 文件选择“编辑”，修改“计算机”字段并保存。

项目	当前配置	说明
桌面入口	AgileX 共享桌面.rdp	无需安装额外远程桌面客户端
远程地址	192.168.0.242:3389	IP 变化时修改 .rdp 中的计算机地址
登录账号	agilex	密码使用部署时设置的小车桌面密码
网络要求	同一局域网	电脑能够访问小车的 3389 端口



图 9-1 AgileX 共享桌面.rdp 连接入口

> **多设备同时连接**
> 可以把 AgileX 共享桌面.rdp 复制到多台 Windows 电脑，车端和电脑位于同一局域网时可同时连接。建议一人负责图形界面或 RViz2，一人通过 SSH 执行命令，另一台设备用于观察或记录。多人共用同一桌面时，应先确认当前由谁启动节点、谁发送目标，避免重复启动或同时控制车辆。

9.2 RDP 登录与 Windows 安全提示

以下两个提示来自 Windows 对 .rdp 文件和 xrdp 自签名证书的安全检查。只有在确认小车 IP、网络环境和设备归属后，才应继续连接；在公共网络或地址不匹配时应取消。



图 9-2 Windows 提示远程连接发布者未知

- 核对“远程计算机”是否为小车当前 IP，例如 192.168.0.242。
- 确认处于受控实验网络，并且该 .rdp 文件来自项目维护者。
- 通常不需要勾选智能卡或 WebAuthn；确认无误后点击“连接”。



图 9-3 xrdp 自签名证书与主机名提示

证书提示如何判断

提示中的请求地址应是小车 IP，证书名称通常为 agilex-desktop。该证书来自 xrdp 自签名配置，因此 Windows 会报告“证书上的服务器名错误”和“证书来自不受信任的证书验证机构”。仅在地址、设备和网络均确认可信时点击“是”；如不希望重复询问，可在首次验证后勾选“不再询问”。

1. 在 xrdp 登录窗口输入用户名 agilex 和桌面密码。

2. 如进入 Ubuntu 登录界面，再选择 agilex 并输入系统密码。
3. 登录后保持桌面在线。关闭 RDP 窗口可能导致该桌面启动的图形程序退出。
4. 确认能够看到任务栏、应用窗口和网络状态。

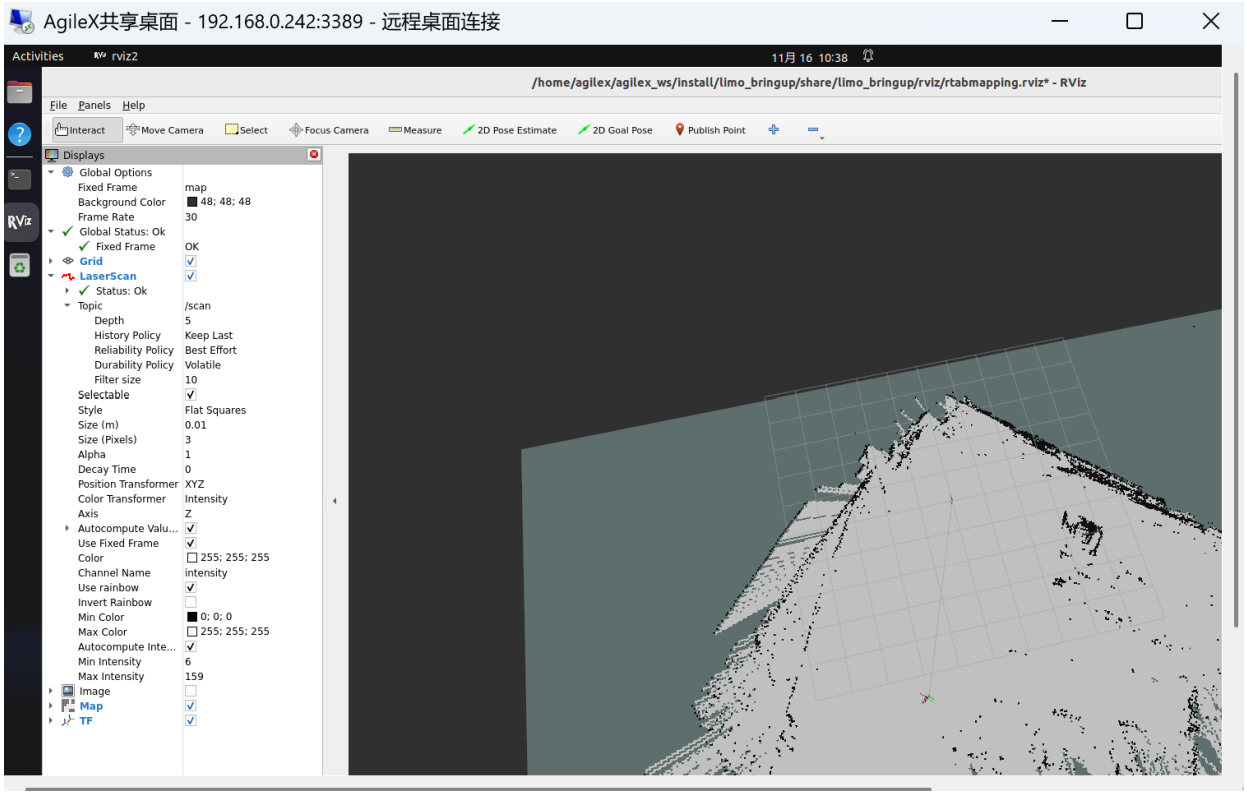


图 9-4 通过 RDP 打开 LIMO 桌面并运行 RViz2

图 9-4 是连接成功的参照状态：窗口标题显示 AgileX 共享桌面 - 192.168.0.242:3389，Ubuntu 桌面中的 RViz2 正在显示地图、激光和坐标信息。

9.3 SSH 终端、图形显示与多设备分工

RDP 和 SSH 可以同时使用。RDP 显示桌面结果，SSH 执行命令并保留终端日志。新建 SSH 终端后先加载 ROS 2 和工作空间环境。

```
ssh agilex@192.168.0.242
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash
ros2 node list
ros2 topic list
```

如果通过 SSH 启动 rviz2、rqt 或机械臂滑块等图形程序，先保持 RDP 桌面在线，再在当前 SSH 终端执行以下四行：

多台车或多台电脑接入同一网络时，ROS 2 默认的 ROS_DOMAIN_ID 都是 0，会互相收到对方的话题（包括 /cmd_vel）。多设备同时实验前，先约定一个私有 domain 并在所有设备上设成相同的值：export ROS_DOMAIN_ID=42，写入 ~/.bashrc 后重新打开终端生效。

```
# 先确认 display 号：ls /tmp/.X11-unix 里是 X0 还是 X1，下面以 :0 为例
export DISPLAY=:0
export XAUTHORITY=/run/user/1000/gdm/Xauthority
export XDG_RUNTIME_DIR=/run/user/1000
rviz2
```

通道	最适合的任务	结果显示位置
RDP 共享桌面	查看桌面、RViz2、rqt、滑块和地图	Windows 远程桌面窗口
SSH 终端	运行 ros2 命令、查看日志、传输文件	SSH 终端
RDP + SSH	一人观察界面、一人执行命令并协同排障	桌面与终端同时使用

设备	建议分工	注意事项
电脑 A	RDP 查看 RViz2、地图和 GUI	避免与其他设备重复点击导航目标
电脑 B	SSH 启动节点、查看日志	每个新终端都要 source 环境
电脑 C	RDP 观察或记录实验结果	共用桌面时不要争抢同一窗口操作
现场人员	观察车辆、急停和物理边界	远程操作不能替代现场安全值守

多设备操作规则

多台设备可同时连接，但同一实验只保留一个导航目标发布者、一个速度控制源和一份日志记录。开始行驶前由现场负责人确认控制权归属，结束后统一停车并关闭所有节点。

9.4 常见连接问题与日常维护

现象	优先检查	处理
RDP 无法连接	同网段、IP 变化、3389 服务	ping 小车地址；用 hostname -I 核对新 IP
出现发布者警告	远程计算机地址和 .rdp 来源	仅在可信实验网络确认后点击连接
证书名称不匹配	请求 IP 与证书名	确认地址和 agilex-desktop 均正确后再继续
SSH 无图形窗口	RDP 是否在线、DISPLAY/XAUTHORITY	保持 RDP 在线并在当前 SSH 终端重新 export
多设备操作冲突	是否重复启动节点或点击目标	明确一名主操作者，其余设备只观察

周期	检查项目	标准
每次实验前	网络、电量、轮胎、相机与雷达	连接稳定，无松动、无遮挡
每次实验后	停止节点、备份地图与日志	后台无残留进程，关键数据已备份
每周	USB/CAN 线缆、侧门、散热和磁盘	无破损，温度与空间正常
长期存放	电池约 50% 电量并分离	定期补电，避免过放

第 10 章 SenseVoice 语音识别与运动控制（选配）

本车 ROS 2 语音链路采用 SenseVoice-Small 离线模型、Sherpa-ONNX 推理、SoX 噪声处理和 ROS 2 节点。识别文本发布到 /voice/text，运动节点再把固定命令词转换为 /cmd_vel。

10.1 SenseVoice 模型与系统部署

项目	本机配置	作用
识别模型	SenseVoice-Small INT8 ONNX	中文、英文等离线语音识别
模型日期	2024-07-17	中英日韩粤多语种版本
模型大小	约 229 MB	适合 Jetson 本地持续运行
推理框架	Sherpa-ONNX 2.0.1	ARM64 CPU 离线推理
语音前端	arecord + SoX	16 kHz 单声道录音、降噪和归一化
ROS 2 输出	/voice/text	发布识别文字
运动输出	/cmd_vel	由独立运动节点发布

选择 SenseVoice-Small 的原因是：相对大型云端 ASR 或 1B 以上模型，它更轻量、启动快，能够完全离线运行，同时支持中文和英文指令；对语音控制而言，固定命令词和短句识别比长篇连续转写更重要。FireRedASR、Qwen3-ASR 等更高精度方案需要更大的模型与计算资源，适合作为后续高性能版本，而不是当前 Jetson 实时控制的首要部署。

资源	路径	说明
模型文件	~/limo_voice_test/models/sense-voice/model.int8.onnx	INT8 ONNX 模型
词表文件	~/limo_voice_test/models/sense-voice/tokens.txt	模型 token 映射
Python 环境	~/limo_voice_env	安装 sherpa-onnx 与语音依赖
识别节点	~/limo_voice_test/limo_voice_asr_sensevoice.py	录音、降噪、识别并发布文本
运动节点	~/limo_voice_test/limo_voice_command_node.py	解析文本并发布受限速度

如重新部署 Python 环境，在虚拟环境中安装 Sherpa-ONNX：

```
source ~/limo_voice_env/bin/activate
python -m pip install sherpa-onnx
python -c "import sherpa_onnx; print(sherpa_onnx.__version__)"
```

> **模型离线运行**
> 完成模型和依赖安装后，ASR、噪声处理与 ROS 2 消息发布均不需要联网。模型文件路径和 tokens.txt 必须同时存在，移动节点或重新部署时应整体复制。

10.2 语音转文字启动与验证

终端	用途	保持状态
终端 A	启动 SenseVoice 识别节点	持续运行，周期性录音

终端	用途	保持状态
终端 B	查看 /voice/text	持续查看识别结果
终端 C	运动控制节点	先关闭运动权限验证，再开启
终端 D	底盘驱动	语音运动测试时启动

```
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash
source ~/limo_voice_env/bin/activate

python ~/limo_voice_test/limo_voice_asr_sensevoice.py \
  --ros-args \
  -p device:=plughw:0,0 \
  -p duration_sec:=4 \
  -p interval_sec:=6.0
```

启动后先加载模型，终端显示 SenseVoice-Small model ready。随后每一轮先保持安静约 0.5 秒，再说一条短命令。录音结束后节点自动执行 SoX 降噪、SenseVoice 推理并发布 /voice/text。

```
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash
ros2 topic echo /voice/text
```

中文命令	English command	预期结果
前进	forward	/voice/text 输出前进命令
后退	back	输出后退命令
左转	left	输出左转命令
右转	right	输出右转命令
停止	stop	输出停车命令

指标	原始录音	SoX 预处理后
峰值电平	-22.6 dBFS	-3.0 dBFS
RMS 电平	-40.2 dBFS	-17.0 dBFS
4 kHz 以上能量占比	35.36%	0.08%
削波样本	0	0

上表来自本车 USB 麦克风实测，说明 SoX 处理主要降低了高频噪声并提升了有效语音电平。最新处理后录音位于 /tmp/limo_voice_clean.wav，原始录音位于 /tmp/limo_voice_raw.wav，可用 aplay 试听。

10.3 语音控制运动实验

先把 APP 切换到指令模式，并停止其他 /cmd_vel 发布节点。运动节点默认关闭运动权限，用于先验证文本解析；确认后再把 enable_motion 设为 true。

```
# 终端 D：标准底盘与雷达
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash
ros2 launch limo_bringup limo_start.launch.py

# 终端 C：先禁止运动，验证文本解析
source ~/limo_voice_env/bin/activate
python ~/limo_voice_test/limo_voice_command_node.py \
  --ros-args -p enable_motion:=false -p odom_topic:=/odom
```

确认 /voice/text 能收到命令后，停止终端 C 并以低速重新启动。标准 limo_start 链路的前进和后退使用 /odom 累计里程，每次目标默认 1 米；若使用 limo_start_ekf.launch.py，应把里程计主题设为 /odometry/filtered。达到目标后运动节点自动停车。

里程计选择：标准 limo_start.launch.py 链路使用 /odom；limo_start_ekf.launch.py 链路使用 /odometry/filtered。请根据实际启动链路设置 odom_topic 参数。

```
python ~/limo_voice_test/limo_voice_command_node.py \
  --ros-args \
  -p enable_motion:=true \
  -p odom_topic:=/odom \
  -p linear_speed:=0.10 \
  -p backward_speed:=0.08 \
  -p angular_speed:=0.25 \
  -p pulse_sec:=0.7 \
  -p forward_distance_m:=1.0 \
  -p backward_distance_m:=1.0 \
  -p max_travel_sec:=20.0
```

语音文本	运动行为	停止机制
前进 / forward	以 0.10 m/s 前进，按里程累计 1 m	达到 1 m 或 20 s 超时停车
后退 / back	以 0.08 m/s 后退，按里程累计 1 m	达到 1 m 或 20 s 超时停车
左转 / left	以 0.25 rad/s 左转	0.7 s 后自动停车
右转 / right	以 0.25 rad/s 右转	0.7 s 后自动停车
停止 / stop	立即输出零速度	无需等待计时结束
其他文本	不产生运动	保持零速度

语音运动安全边界
首次测试使用架空车轮；落地测试时前方至少保留 2 m 空间。只允许一个节点发布 /cmd_vel，不得同时使用 APP 摇杆、Nav2、巡线或键盘遥控。odom 中断、未知命令、节点退出或超过最大行驶时间时，运动节点必须持续发布零速度。现场人员应保留立即断电能力。

开始实验前用 ros2 topic info /cmd_vel --verbose 检查速度发布者，并用 ros2 topic echo /odom 确认里程计持续更新。使用 EKF 时应检查 /odometry/filtered。标准 limo_base 链路使用 /odom。前进方向异常时先检查底盘模态与电机配置，不要仅靠反向命令长期替代正确配置。

10.4 故障排查与验收

现象	优先检查	处理
ModuleNotFoundError	虚拟环境是否激活	source ~/limo_voice_env/bin/activate
模型加载失败	模型与 tokens 路径	检查 model.int8.onnx 和 tokens.txt

现象	优先检查	处理
没有识别文本	麦克风设备、录音电平和降噪结果	aplay /tmp/limo_voice_clean.wav
识别词不准确	说话距离、环境噪声、命令词	缩短命令，靠近麦克风，坚持固定词表
有文字但不运动	enable_motion 与 /cmd_vel	先设 enable_motion:=true
按 1 米目标不准确	轮式里程计与打滑	检查 /odom；在防滑地面复测
同时运行就乱动	多个速度发布者	关闭 APP、Nav2、巡线或其他遥控节点
转动方向相反	角速度符号	增加 -p invert_angular:=true

验收标准

中文和英文固定命令都能稳定进入 /voice/text；一次前进或后退的目标距离约 1 m 且自动停车；停止命令立即生效；未知文本、里程计中断和节点退出均不会维持运动；多设备查看时只有一个速度控制源。

第 11 章 MyCobot 机械臂控制（选配）

MyCobot 280 M5 通过六个关节实现抓取和姿态控制，可采用关节角 API、图形滑块、键盘、模型跟随与 MoveIt 2 等方式。标准 LIMO 未安装机械臂时，本章作为选配扩展参考；执行真实机械臂命令前必须确认本机已安装对应硬件和软件包。

项目	参数	项目	参数
自由度	6	有效负载	250 g
工作半径	280 mm	重复定位精度	约 0.5 mm
机械臂自重	约 800 g	输入电源	12 V / 5 A
通信	USB Type-C	工作温度	-5 至 45 摄氏度

11.1 连接与工作空间检查

```
source /opt/ros/humble/setup.bash
source ~/agilex_ws/install/setup.bash
ls -l /dev/ttyACM*
groups
ros2 pkg list | grep -E 'mycobot|moveit|cobot'
```

检查项	通过标准	处理
USB 设备	出现稳定 ttyACM 端口	重新插拔，禁止使用频繁变化的端口
串口权限	当前用户属于 dialout	重新登录使权限生效
姿态	关节无顶死，线缆无拉扯	先人工调整至安全姿态
供电	12 V / 5 A 稳定	不要在带载运动中热插拔
工作空间	抓取目标在可达半径内	避免碰撞车身或相机

> **首次运行**
>
> 将机械臂底座固定，清空 280 mm 工作半径内的障碍，手放在电源附近。先运行仿真，再以低速小角度执行真实机械臂。

11.2 MoveIt 2 仿真预检查

```
ros2 launch mycobot_280_moveit2 demo.launch.py   # RViz2 会随 demo 一起启动
```

本机 MoveIt 2 demo 使用 ros2_control 的 mock_components 假硬件控制器，只验证 URDF、关节名称、Planning Group 和碰撞模型，不会驱动真实机械臂。先在仿真中确认关节方向、限位和末端工作空间，再进行真实硬件检查。

11.3 真实机械臂滑块控制

该 launch 直接连接真实 /dev/ttyACM0，并已包含 slider_control 节点，不能再次执行 ros2 run mycobot_280 slider_control。首次运动应清空 280 mm 工作半径并小角度操作。

```
ros2 launch mycobot_280 slider_control.launch.py \
  port:=/dev/ttyACM0 baud:=115200
```

> **快速拖动风险**
>
> 真实机械臂会跟随滑块目标运动。首次拖动采用小角度逐步执行，不要在连接真实机械臂后突然将模型关节拖到远端。

11.4 模型跟随真实机械臂

mycobot_follow.launch.py 已包含 follow_display 节点，不需要再次 ros2 run 启动。

```
ros2 launch mycobot_280 mycobot_follow.launch.py \
  port:=/dev/ttyACM0 baud:=115200
```

启动后缓慢手动掰动机械臂，RViz2 模型应同步变化。若关节方向相反或延迟很大，停止拖动并检查机械臂固件、波特率和关节方向配置。

11.5 GUI 与键盘控制

```
# 默认只读观察；确认工作空间后再显式允许运动
ros2 launch mycobot_280 simple_gui.launch.py enable_motion:=false
ros2 launch mycobot_280 simple_gui.launch.py enable_motion:=true

# 键盘控制
ros2 launch mycobot_280 teleop_keyboard.launch.py \
  port:=/dev/ttyACM0 baud:=115200
ros2 run mycobot_280 teleop_keyboard
```

simple_gui 默认 enable_motion=false，只用于观察；确认机械臂工作空间和限位后，才能显式设置为 true。键盘控制在 teleop launch 提供服务后再运行 teleop_keyboard。首次运动应使用低步长并保持断电能力。

11.6 MoveIt 2 仿真规划

本机 MoveIt 2 demo 使用 mock_components 假硬件，适合验证 URDF、Planning Group、轨迹规划和碰撞模型，不会直接驱动真实机械臂。仿真规划成功也不代表现场无碰撞，执行真实硬件前仍需检查夹爪、线缆、相机和车身模型。

```
ros2 launch mycobot_280_moveit2 demo.launch.py
```

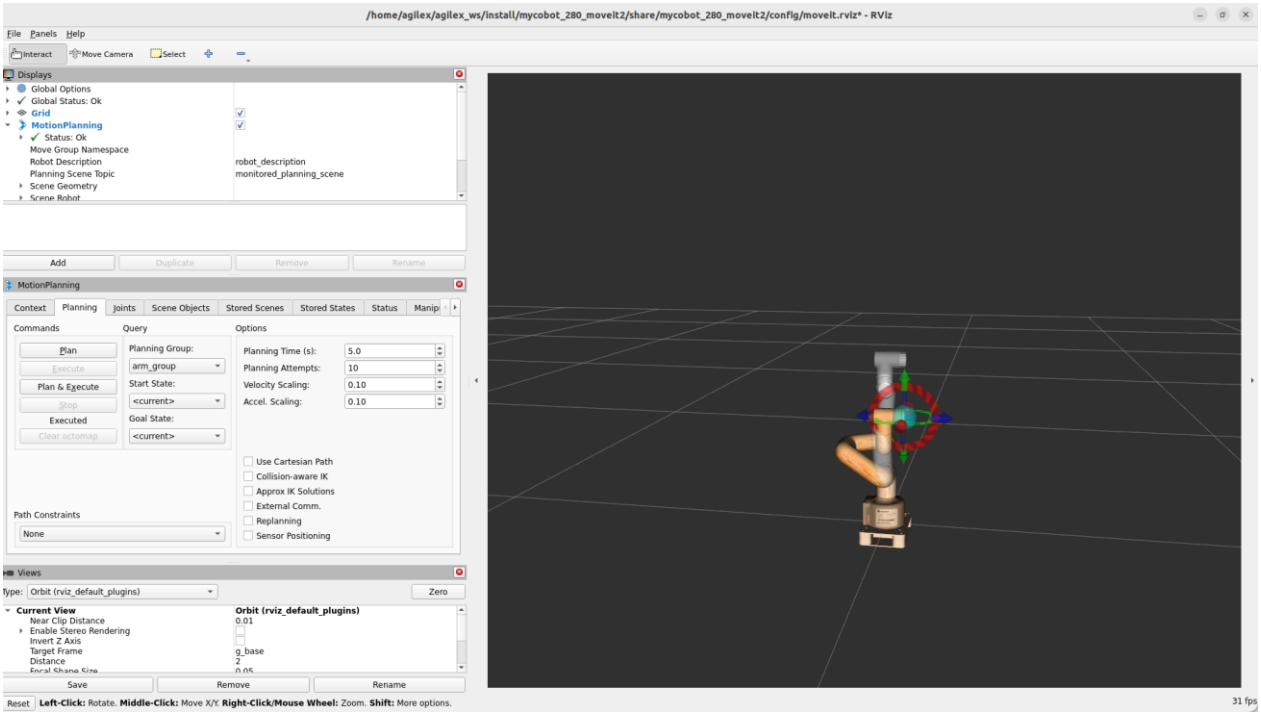


图 11-1 MoveIt 2 MotionPlanning 面板与 MyCobot 280 规划场景

图为 MoveIt 2 的 MotionPlanning 面板与 MyCobot 280 模型：Displays 中 MotionPlanning 与 Global Status 均为 Ok，Planning Group 为 arm_group，Start State 与 Goal State 都是 <current>；窗口标题显示本机使用的配置文件是 mycobot_280_moveit2/share/mycobot_280_moveit2/config/moveit.rviz。

先看三处判断环境是否就绪：面板 Status 是否 Ok、末端标记是否落在 280 mm 工作半径内（不要穿到桌面、车身或相机）、Velocity 与 Accel Scaling 是否为小值（示例为 0.10，与本节“低速执行”的要求一致）。判据是“先 Plan 出轨迹再 Execute”：未规划时 Execute 按钮为灰色，执行过程中关节应连续、无线缆拉扯、无碰撞告警。

面板显示正常不等于现场安全：碰撞模型只覆盖 URDF 里的连杆，夹爪线缆、相机支架和车身并不在其中，执行前仍需人工目视确认工作空间。

步骤	在 MoveIt 2 中完成的操作	通过标准
加载模型	确认 URDF 和关节名称	模型与真实机械臂结构一致
设置目标	拖动末端标记到目标位姿	目标在工作空间内

步骤	在 MoveIt 2 中完成的操作	通过标准
规划	检查轨迹是否避开车身	路径连续、碰撞模型正确
执行	低速执行并观察真实机械臂	无碰撞、无拉扯、关节不越限

11.7 移动抓取

移动抓取把 Nav2 导航与机械臂抓取组合：车辆先到达抓取点，再调用机械臂轨迹，最后离开抓取区域。实现时应把导航入口统一为 Nav2，并核对任务脚本中的坐标、坐标框架和抓取姿态。

1.在目标场地完成建图和 Nav2 导航测试。

2.在 RViz2 中记录抓取点和离开点，确认坐标与朝向。

3.将目标点、机械臂抓取姿态和等待时间写入任务脚本。

4.先空载运行，验证到达、停车、抓取和退出顺序。

5.再放置轻质目标进行低速抓取，现场人员保持急停能力。

兼容入口默认行为是配置检查，不会直接执行移动抓取。必须先修复配置中的 missing 项并得到 ready: true，再显式传入 --execute。

```
ros2 launch limo_bringup limo_start.launch.py
ros2 launch limo_bringup limo_nav2_diff.launch.py

cd /home/agilex/agilex_ws/src/set_nav_point
# 1. 仅检查导航点、抓取点和机械臂姿态配置
python3 more_task_node.py --check

# 2. 只有检查结果为 ready: true 后才执行任务
python3 more_task_node.py --execute
```

抓取顺序

必须先停车再执行机械臂，避免底盘运动与机械臂同时动作。脚本最终版本应包含导航失败、抓取失败和超时的停车/复位分支。

附录 A 整车三视图与外形尺寸

整车外形尺寸为 322 x 220 x 251 mm。下图为整车三视图，用于布置实验场地、检查箱体尺寸和确认传感器突出部位。选配机械臂、升降杆或线缆会增加实际占用空间。

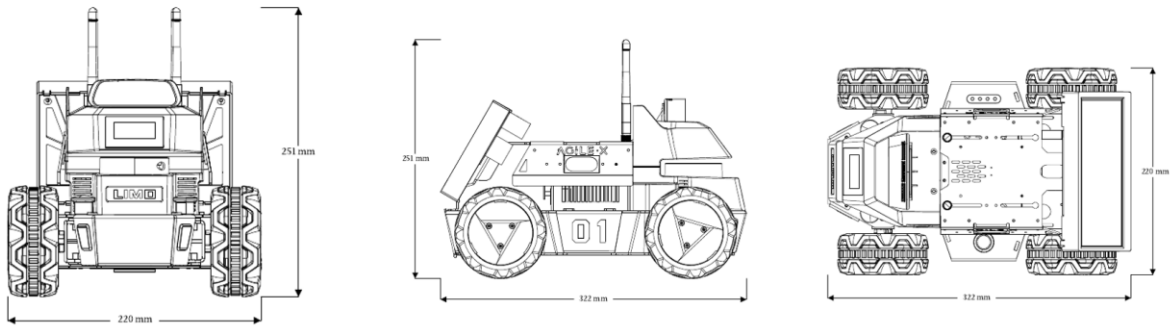


图 A-1 LIMO 正视图、侧视图与俯视图

尺寸项	数值	使用说明
长度	322 mm	不含外接线缆和完全展开的侧门
宽度	220 mm	安装麦克纳姆轮或履带后复核实际占地
高度	251 mm	雷达、天线和选配模块可能导致高度增加
轴距	200 mm	运动学计算和控制器参数的基础量
轮距	175 mm	差速与麦克纳姆速度分配的基础量

附录 B 基本操作与常用命令

B.1 Linux 基本操作

操作	命令	示例
查看当前位置	pwd	pwd
切换目录	cd	cd ~/agilex_ws
返回上级	cd ..	cd ../src
查看文件	ls / ls -la	ls -la ~/.ros
创建目录	mkdir -p	mkdir -p ~/agilex_ws/src
复制 / 移动	cp / mv	cp map01.yaml backup/
终止程序	Ctrl+C	停止 launch 或视觉节点
显示隐藏文件	Ctrl+H	文件管理器显示 .ros

B.2 ROS 2 编译和工作空间

任务	命令
加载 ROS 2	source /opt/ros/humble/setup.bash
加载本机工作空间	source ~/agilex_ws/install/setup.bash
编译整个工作空间	cd ~/agilex_ws && colcon build --symlink-install
编译单个包	colcon build --packages-select limo_base
创建 Python 包	ros2 pkg create learning_limo --build-type ament_python --dependencies rclpy
启动 launch	ros2 launch package launch_file.launch.py
运行节点	ros2 run package executable

B.3 本机常用任务

任务	命令
加载环境	source /opt/ros/humble/setup.bashsource ~/agilex_ws/install/setup.bash
启动底盘	ros2 launch limo_base limo_base.launch.py
启动底盘与雷达	ros2 launch limo_bringup limo_start.launch.py
单独启动雷达	ros2 launch ydlidar_ros2_driver ydlidar.launch.py
查看激光数据	ros2 topic echo /scan --once
查看底盘状态	ros2 topic echo /limo_status --once

任务	命令
发布停车速度	ros2 topic pub --once /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.0}, angular: {z: 0.0}}"

任务	命令
Cartographer	ros2 launch limo_bringup limo_cartographer.launch.py
保存地图	ros2 run nav2_map_server map_saver_cli -f map01
差速/全向导航	ros2 launch limo_bringup limo_nav2_diff.launch.py
阿克曼导航	ros2 launch limo_bringup limo_nav2_ackermann.launch.py
启动相机	ros2 launch orbbec_camera dabai.launch.py
RTAB-Map 建图	ros2 launch limo_bringup limo_rtab_rgbd.launch.py
RTAB-Map 定位	ros2 launch limo_bringup limo_rtab_rgbd.launch.py localization:=true
RTAB-Map 导航	ros2 launch limo_bringup limo_rtab_nav2.launch.py

B.4 常用路径

内容	路径
底盘驱动	~/agilex_ws/src/limo_ros2/limo_base
消息与服务	~/agilex_ws/src/limo_ros2/limo_msgs
建图与导航入口	~/agilex_ws/src/limo_ros2/limo_bringup/launch
地图目录	~/agilex_ws/src/limo_ros2/limo_bringup/maps
视觉脚本	/home/agilex/agilex_ws/src/limo_visions/scripts
RTAB-Map 数据库	~/.ros/rtabmap.db

> **路径核对**
>
> 不同镜像的包名或目录可能略有差异。使用 tab 补全、ros2 pkg prefix 和 find 核对实际路径，不要为了匹配示例而修改系统目录。

B.5 查看本机版本

报障、核对资料或准备重装镜像前，先用下面四条命令确认本机版本，并把输出一起保存：

```
head -2 /etc/os-release                              # 系统版本：Ubuntu 22.04.5 LTS
head -1 /etc/nv_tegra_release                        # JetPack：R36.4.3
echo $ROS_DISTRO                                     # ROS 版本：humble
git -C ~/agilex_ws/src/limo_ros2 log -1 --oneline    # 驱动源码版本
```

不同批次的镜像可能使用不同版本的驱动源码，实际版本以本机输出为准。

附录 C ROS 2 系统架构

ROS 2 系统按操作系统、通信中间层和应用层分层。本机使用 ROS 2 Humble，节点发现依赖 DDS，不再需要单独启动 ROS Master。

层级	ROS 2 Humble 对应
操作系统	Ubuntu 22.04 + ROS 2 Humble
通信	DDS、ROS 2 话题/服务/动作
节点发现	DDS 自动发现，无中心节点
工作空间	colcon 工作空间 + colcon build
启动	.launch.py Python 配置
可视化	rviz2、rqt_graph、rqt_image_view

C.1 计算图核心概念

概念	含义	本车示例
节点 Node	执行一项功能的进程	limo_base、Nav2、相机驱动、视觉节点
消息 Message	有严格数据结构的数据	Twist、LaserScan、Image、Odometry
话题 Topic	多对多的异步发布/订阅	/cmd_vel、/scan、/imu
服务 Service	请求-应答的同步调用	保存地图、坐标变换查询
动作 Action	带目标、反馈、结果的长任务	导航到目标点、机械臂轨迹
参数 Parameter	节点运行时配置	速度限制、地图路径、AMCL 粒子数
TF2	随时间变化的坐标树	map -> odom -> base_link -> sensor

C.2 文件系统

组成	作用
Package 功能包	节点、库、launch、参数和执行文件的基本单元
package.xml	声明包名、版本、依赖和维护信息
CMakeLists.txt	C/C++ 包或消息接口包的构建规则
setup.py / setup.cfg	Python 包的安装与入口点
msg / srv / action	自定义通信接口
launch	一次启动多个节点并加载参数
config / param	YAML 参数与算法配置

C.3 话题、服务、动作和参数的选择

需求	推荐机制	原因
持续发布速度、激光、图像、状态	话题	高频、异步、可多订阅者
保存地图、查询一次性结果	服务	需要明确请求和应答
导航、抓取、长时运动	动作	需要进度反馈和可取消能力
调整限速、地图和算法阈值	参数	启动期或运行期配置

附录 D ROS 2 工作空间与开发基础

D.1 工作空间结构

```
~/dev_ws/
  src/       # 功能包源代码
  build/     # 编译中间文件
  install/   # 编译后的安装空间
  log/       # 编译日志
source /opt/ros/humble/setup.bash
cd ~/dev_ws
colcon build --symlink-install
source install/setup.bash
```

每个新终端都要加载 ROS 2 和工作空间环境。安装空间不是源代码目录，修改 src 后必须重新 colconbuild，或使用 --symlink-install 支持部分 Python 和资源文件的软链接更新。

D.2 创建功能包

```
cd ~/dev_ws/src
ros2 pkg create learning_limo \
 --build-type ament_python \
 --dependencies rclpy std_msgs geometry_msgs
cd ~/dev_ws
colcon build --packages-select learning_limo
source install/setup.bash
```

创建 talker.py 与可执行入口

上一步创建的 Python 包还需要添加实际节点代码，并在 setup.py 注册 talker 入口，否则 ros2 run 和 launch.py 都找不到该可执行文件。

```
# 文件：~/dev_ws/src/learning_limo/learning_limo/talker.py
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class Talker(Node):
    def __init__(self):
        super().__init__('learning_talker')
        self.publisher = self.create_publisher(
            String, '/learning_topic', 10)
        self.create_timer(1.0, self.publish_message)
        self.count = 0

    def publish_message(self):
        self.count += 1
        self.publisher.publish(
            String(data=f'Hello LIMO {self.count}'))

def main():
    rclpy.init()
    node = Talker()
    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        pass
    finally:
        node.destroy_node()
```

```
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

在自动生成的 setup.py 中设置 console_scripts 入口：

```
entry_points={
    'console_scripts': [
        'talker = learning_limo.talker:main',
    ],
}
```

保存后回到 ~/dev_ws 执行 colcon build --packages-select learning_limo，再加载 install/setup.bash。之后 ros2 run learning_limo talker 才能启动。

D.3 编写 launch.py

```
from launch import LaunchDescription
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch_ros.actions import Node
from ament_index_python.packages import get_package_share_directory
def generate_launch_description():
    return LaunchDescription([
        Node(
            package='learning_limo',
            executable='talker',
            output='screen',
            parameters=[{'use_sim_time': False}],
        ),
    ])
```

D.4 RViz2 与 rqt 调试

```
rviz2
rqt_graph
rqt_image_view
ros2 run tf2_tools view_frames
# TF 树（首次使用前先安装）
sudo apt install ros-humble-rqt-tf-tree
ros2 run rqt_tf_tree rqt_tf_tree
```

工具	用途	典型检查
rviz2	三维显示	TF、激光、地图、机器人模型、导航目标
rqt_graph	节点关系图	谁发布 /cmd_vel，谁订阅 /scan
rqt_image_view	相机图像	RGB、Depth 话题是否存在
tf2_tools view_frames	生成 TF 树 PDF	map、odom、base_link 是否完整连接

D.5 最小开发闭环

一个可复现的 LIMO ROS 2 开发闭环应至少包含：加载环境、启动底盘、验证状态、发布短时命令、接续算法和停止清理。下面的顺序适合新建任何视觉、运动学或导航实验。

1.在一个终端启动底盘或组合 launch，并确认没有重复进程。

2.在第二个终端使用 ros2 topic list、ros2 topic hz 和 ros2 topic echo 检查输入输出。

3.运行自写节点，先只打印数据，不发布速度。

4.接管 /cmd_vel 后设置低速和停车超时，再进行闭环实验。

5.保存 launch、YAML、日志和地图，按 Ctrl+C 停止全部节点。

```
ros2 launch limo_base limo_base.launch.py
ros2 node list
ros2 topic list
ros2 topic hz /scan
ros2 run learning_limo talker
```

附录 E 系统镜像烧录

烧录前确认

先确认本机使用 microSD、eMMC 还是 NVMe 启动，并记录当前系统备份和网络配置。Jetson 型号、存储介质和镜像版本不匹配会造成无法启动或驱动缺失。

E.1 获取正确镜像

1.从设备供应商或官方资料获取与 Jetson Orin Nano、Ubuntu 22.04 和 LIMO 驱动匹配的镜像。

2.核对镜像 SHA256 或发布说明，避免下载不完整或把旧 Jetson Nano 镜像用于 Orin Nano。

3.备份工作空间、地图、RTAB-Map 数据库、标定文件和课程资料。

4.准备稳定电源、读卡器或 USB 数据线和足够的存储空间。

E.2 microSD 卡烧录

步骤	操作	检查
1	安装 balenaEtcher	从 https://etcher.balena.io/ 获取对应版本
2	选择官方镜像文件	确认文件类型和大小
3	选择目标存储卡	只选正确的可移动设备，避免擦除电脑硬盘
4	执行 Flash 并校验	等待写入和 Verify 完成
5	安全弹出后插入 Jetson	首次启动连接显示器或串口

E.3 eMMC/NVMe 或开发板模式

若配置使用 eMMC 或 NVMe，使用 NVIDIA SDK Manager 或供应商提供的恢复模式流程。不同载板和模块的按键与 USB 接口不同，必须按对应硬件说明进入恢复模式，不能照搬旧版 Nano 的通用步骤。

E.4 首次启动检查

```
source /opt/ros/humble/setup.bash
printenv ROS_DISTRO
ros2 pkg list | grep -E 'limo|ydlidar|orbbec|nav2'
ls -l ~/agilex_ws
```

恢复判断

启动后先检查显示、网络、USB、串口、ROS 2 版本和驱动包，再进行地图和机械臂功能。若启动失败，停止反复上电并保留串口日志。

附录 F 底盘固件升级

底盘固件升级用于修复通信、运动控制或保护逻辑。固件版本必须与本机的底盘硬件和 ROS 2 镜像匹配，不要使用其他型号或其他批次的固件。升级期间保持电源稳定，不得移动车辆。

1.记录当前固件版本、底盘型号和报错现象，关闭所有 ROS 2 和 APP 控制。

2.从官方渠道获取对应的升级软件和固件文件，核对型号与版本。

3.在关机状态下按设备说明进入固件升级模式，等待计算单元正常启动。

4.打开升级工具，选择正确的串口和固件文件。

5.执行升级并等待完成，不要拔电源或关闭串口。

6.升级成功后关闭串口，重新上电，检查底盘状态、模式灯和 /limo_status。

> 固件升级工具通常在 Windows 个人电脑上运行，不在车上执行。使用前核对工具版本、目标串口和固件文件，并按发布说明完成升级；不要照搬 Linux 串口或 chmod 命令。

检查项	正确结果
串口	选择底盘实际端口，不与雷达或机械臂串口混用
固件文件	型号、版本号与发布说明匹配
进度	升级过程持续供电，工具无超时
升级后状态	模式灯正常，/limo_status 能持续发布

工具名称可能变化

工具名称可能随镜像版本变化；应使用供应商为 Orin Nano / ROS 2 配置提供的最新刷写工具与步骤。

附录 G 建图与导航参数配置

ROS 2 Humble 采用 Nav2 生命周期节点和 YAML 参数文件；不同镜像版本的参数名与插件机制可能不同。本节保留调参逻辑和参考值，具体键名必须在本机 YAML 中核对。

```
find ~/agilex_ws/src/limo_ros2/limo_bringup -type f \
  \( -name '*.yaml' -o -name '*.lua' -o -name '*.launch.py' \) | sort
```

G.1 GMapping 参数

参数概念	参考值	作用
particles	30	粒子数量，越大越稳定但更耗 CPU
delta	0.05	栅格分辨率，单位 m
linearUpdate	1.0	平移超过该距离处理一次扫描
angularUpdate	0.5	旋转超过该角度处理一次扫描
maxRange	按雷达有效距离	限制参与建图的最远激光
minimumScore	0.0	扫描匹配最低分数，过低易产生错误匹配
resampleThreshold	0.5	基于有效粒子数的重采样阈值

G.2 Cartographer 参数

参数	常见配置	说明
map_frame	map	地图父坐标系
tracking_frame	base_link 或 IMU frame	实际追踪坐标系
published_frame	odom 或 base_link	发布位姿的父坐标系
provide_odom_frame	true	是否由 Cartographer 发布 odom
use_odometry	true/false	是否融合轮式里程计
num_laser_scans	1	本配置使用单台二维雷达
lookup_transform_timeout_sec	0.2	TF 查询超时
submap_publish_period_sec	0.3	子图发布周期

G.3 Nav2 定位与局部控制

参数组	关注项	调参建议
AMCL	min/max_particles、误差、更新距离	粒子过少会丢定位，过多会增加计算负载
Controller	max_vel_x、max_vel_theta、加速度	先降速，再调角速度与加速度
Goal tolerance	xy_goal_tolerance、yaw_goal_tolerance	按场地精度调和停车行为

参数组	关注项	调参建议
Costmap inflation	inflation_radius	保证车身与行人安全距离
Planner	planner plugin、路径代价	先确认地图正确，再改路径偏好
Recovery	旋转、后退、等待行为	狭小场地应减少后退与大幅旋转

参数修改原则

一次只改变一组参数，保留修改前版本并记录效果。地图、TF、传感器话题异常时，不要通过增大速度或放宽碰撞约束来掩盖问题。

附录 H 配套课程与资料

配套课程与资料随版本更新，建议以官方渠道提供的最新版本为准。学习时应重点掌握机器人结构、运动学、传感器原理与算法思路；具体命令、工作空间、launch 文件与导航包以本手册的 ROS 2 写法为准。

学习模块	建议练习	本手册对应章节
产品与底盘	模式切换、电池充电、APP 低速控制	第 1-2 章
驱动与通信	话题检查、Twist 控制、状态订阅	第 3 章
运动学	差速、阿克曼、麦克纳姆速度验证	第 4 章
激光与导航	建图、初始位姿、单点与多点导航	第 5-6 章
视觉与融合	RTAB-Map、文字、颜色、巡线与 YOLO	第 7-8 章
语音与机械臂	录音识别、模型跟随、MoveIt 2 与移动抓取	第 10-11 章

课程资料、镜像、固件和示例包可能随版本更新。下载时应记录版本号与发布日期，先备份现有系统，再替换整机镜像；不要把旧版软件包直接复制进 ROS 2 工作空间。

附录 I 故障排查

现象	优先检查	处理
上电无反应、电量表不亮	电池是否插到位、电量是否耗尽	重新插紧输出接口；充电后再试
发 /cmd_vel 车辆不动	APP/遥控是否切到 ROS 2 指令模式、/odom 数值是否冻结	底盘 MCU 挂死：整车断电重上电；再确认 APP 或遥控器未抢控制
蜂鸣器长鸣	遥控器或 APP 是否断开	失联报警：重新连接或断开后按开关键；软件层关不掉，需保持连接或物理处理
终端反复刷 Invalid frame! Check sum failed	是否有多个进程同时读底盘串口	sudo fuser -v /dev/ttyTHS1 只保留一个 limo_base；串口波特率 460800
RViz2 里看不到 /scan 点云	/scan 的 QoS 是否匹配	把 LaserScan 的 Reliability Policy 改为 Best Effort（默认 Reliable 收不到）
SSH 里 rviz2 起不来或报 cannot open display	DISPLAY / XAUTHORITY 是否设置、车上是否已有桌面会话	按 9.2 设置环境变量；先在车上或远程桌面登录一次
多台车同网时互相干扰、车自己动	ROS_DOMAIN_ID 是否相同（默认都是 0）	所有设备设成同一个私有 domain，见 9.1
ros2 launch 找不到包	是否 source 了 ROS 与工作空间	重新加载 setup.bash，核对包名
雷达无 /scan	USB、串口占用、驱动日志	关闭重复驱动后重新连接雷达
相机无图像	USB 带宽、相机占用、话题名	移除冗余 USB 设备，只保留一个相机驱动
导航没有全局路径	地图、初始位姿、目标点、代价地图	重新设置位姿，确认目标点在可通行区域
定位粒子不收敛	地图版本、起始位置、map→odom	重新估计位姿，检查 TF 与地图是否对应
车辆原地摆动	控制器参数、激光是否被遮挡	降低速度与增益，检查雷达视野
磁盘写满、日志暴涨	~/.ros 与日志目录	清理旧 bag、地图备份与日志，保留必要数据

I.1 通用排查顺序

1.停止所有正在运行的相关节点，记录错误日志最后 20 行。

2.确认 ROS 环境和当前模态，查看 node、topic 与 status。

3.一次只启动一个功能入口，验证传感器和底盘后再叠加算法。

4.保留复现步骤、包版本、launch 文件和关键日志，便于进一步定位。

```
ros2 node list
ros2 topic list
ros2 topic hz /scan
ros2 topic echo /limo_status --once
ros2 pkg prefix limo_bringup
```

附录 J 使用检查表与版本说明

J.1 收货验收（约 30 分钟）

新车拆箱后按顺序做一遍，任何一项不通过都先别急着做实验，按附录 I 排查或联系供应商。命令都在车上终端执行，需要先 source 工作空间。

检查项	怎么做	通过标准
外观与配件	对照 1.2 表格清点；检查轮胎、雷达、相机、外壳与侧门	无破损、无缺件、螺丝无松动
电池与接线	取出电池看电量表或万用表，插紧黄色输出接口	电压不低于 11.1 V，接口无松动
开机与电量表	长按开关键开机，观察前置屏幕与电量表	正常进入系统，无红灯报警
四种模态与车灯	依次切换四轮差速/阿克曼/麦克纳姆，观察车灯	车灯颜色与机械模态一致
雷达数据	ros2 launch ydlidar_ros2_driver ydlidar.launch.py；再执行 ros2 topic hz /scan	频率约 6 Hz，frame_id 为 laser_link
相机图像	ros2 launch orbbec_camera dabai.launch.py；ros2 topic list | grep -E "color|depth"	能看到彩色与深度话题
APP 遥控	APP_Agilex 蓝牙连接，低速前进、后退、转向、停车	方向正确，能正常停车
联网与登录	按 9.1-9.3 连接局域网，并用 RDP 共享桌面或 SSH 登录	能进入桌面或命令行
建图自检	按 5.4 慢速走一小圈后保存地图	生成 map.pgm 与 map.yaml，墙体闭合
关机顺序	停车 → 停止节点 → 关闭 APP → 关机 → 断开电池	无异常报错，电量表熄灭

J.2 每次实验前检查表

[ ] 电池电量充足，输出接口与充电接口连接正确

[ ] 轮胎、插销、履带或麦克纳姆轮与当前模式一致

[ ] LIMO 位于平整场地，行驶区无人和临时障碍

[ ] APP_Agilex 已连接，并在低速模式下完成方向测试

[ ] 终端已经 source ROS 2 Humble 与本机工作空间

[ ] 同一设备没有重复启动底盘、雷达或相机驱动

[ ] 已准备急停、断电人员和必要防护措施

J.3 内容校准说明

本手册内容以本机实机配置为基线（Jetson Orin Nano、Ubuntu 22.04、ROS 2 Humble），并结合实机验证结果整理。参数、界面与包名可能随镜像版本变化，实验前请以实机命令输出为准。

对于源资料中可能因版本更新而变化的内容，如插件名称、地图参数、相机话题和视觉脚本路径，本手册强调先在本机 launch/YAML/ROS 话题中核对。设备实际发行版本、校准参数与现场安全规范优先于示例文本。

J.4 修订记录

版本	日期	主要修改
1.1.3	2026.11	重写 6.6 路径巡检操作，补充 YOLO 检测图，修复图 2-1、图号、固件升级说明和目录残留。

附录 K 依赖包一览

本手册用到的 ROS 2 包和脚本分成两类：一类随镜像预装，照着命令就能跑；一类本机没有预装（或只在选配硬件上才有）。下表在 LIMO（Jetson Orin NX / Ubuntu 22.04 / ROS 2 Humble）上逐条核对过，遇到没有的按最后一列处理，不要硬套命令。

包 / 文件	用途	本机预装	未预装时怎么办
limo_base / limo_bringup / limo_msgs	底盘驱动、启动入口、消息定义	是	—
ydlidar_ros2_driver	T-mini Pro 激光雷达	是	—
orbbec_camera	Dabai 深度相机	是	—
limo_visions	颜色、红绿灯、ArUco、文字识别与巡线节点	是	—
trt_yolo	TensorRT 加速的 YOLO 检测	是	—
aruco_ros	ArUco 标记识别（8.7）	是	—
robot_localization + rf2o_laser_odometry	抗打滑建图的 EKF 与激光里程计	是	按 5.5 增量包安装
mycobot_280 / mycobot_280_moveit2 / mycobot_280_moveit2_control	机械臂滑块、跟随、MoveIt 2 与同步节点（11.x）	是	—
slam_gmapping / limo_gmapping.launch.py	GMapping 建图（5.2）	否	改用 Cartographer（5.3–5.4）
rrt_exploration	RRT 自主探索（5.5）	否	用 Cartographer 建图 + Nav2 多点导航
agilex_pure_pursuit	路径记录与复现（6.6）	否	用 Nav2 航点复现
detect_ros	升降杆标记识别（8.5）	否（选配硬件）	未装升降杆就不执行
rqt_tf_tree	TF 树可视化（D.4）	否	sudo apt install ros-humble-rqt-tf-tree
voice	旧版语音包（第 10 章已改用 SenseVoice）	否	按第 10 章的 SenseVoice 流程
limo_path_patrol	路径录制、CSV 校验与自动跟随	当前资源包提供	检查 launch 与服务；未安装时按 6.6 和附录 L 核对

下面这张表把手册里出现的启动命令与脚本按用途归到一起，方便按“我现在要做什么”反查；标注“随包提供”的脚本放在配套资源包的对应目录里，其余由镜像预装。

脚本 / 命令	位置	用途	章节
limo_base.launch.py	limo_base	启动底盘驱动：串口、/cmd_vel、/odom、/imu、/limo_status、TF	3.4
limo_start.launch.py	limo_bringup	底盘 + 雷达组合入口（已含 ydlidar）	5.1、5.4、6.2

脚本 / 命令	位置	用途	章节
limo_cartographer.launch.py	limo_bringup	Cartographer 2D 建图，输出 /map 与 submap	5.4
limo_nav2_diff / limo_nav2_ackermann.launch.py	limo_bringup	Nav2 导航（差速用 diff，阿克曼用 ackermann）	6.2
limo_rtab_rgbd.launch.py	limo_bringup	RTAB-Map RGB-D 建图，落库 ~/.ros/rtabmap.db	7.2
limo_rtab_nav2.launch.py	limo_bringup	RTAB-Map 定位 + Nav2 导航	7.3
ros2 run limo_visions recognition	limo_visions	颜色识别，发布标注图 /image_hsv（不发 /ocr/result）	8.3
ros2 run limo_visions object_detect	limo_visions	FollowLine 巡线节点：订阅 /camera/color/image_raw，发布 /image_hsv 与 /cmd_vel	8.4
ros2 run limo_visions traffic_light	limo_visions	红绿灯识别，按 HSV 标 RED/YELLOW/GREEN	8.8
ros2 run limo_visions move_to_ar	limo_visions	ArUco 跟随：对准标记并移动到设定距离	8.7
limo_recognition_text_read.py	limo_visions/scripts（随包提供）	文字识别：输出 /ocr/result 与 /ocr/image	8.2
yolov8n_det.launch.py	trt_yolo	TensorRT YOLOv8 检测（首次需 serialize）	8.9
slider_control.launch.py	mycobot_280	滑块控制关节角（内置节点，参数 port/baud）	11.2、11.3
mycobot_follow.launch.py	mycobot_280	模型跟随真机（内置 follow_display）	11.4
simple_gui / teleop_keyboard.launch.py	mycobot_280	GUI 输入关节角、键盘控制末端	11.5
mycobot_280_moveit2 demo.launch.py	mycobot_280_moveit2	MoveIt 2：mock_components 假硬件 + RViz2 规划面板	11.2、11.6
ros2 run mycobot_280_moveit2_control sync_plan	mycobot_280_moveit2_control	把规划结果同步到真实机械臂	11.6
more_task_node.py	set_nav_point（随包提供）	多点任务入口：导航到点后调用机械臂抓取	11.7
limo_voice_asr_sensevoice.py	~/limo_voice_test（随包提供）	SenseVoice 语音转文字，发布 /voice/text	10.2
limo_voice_command_node.py	~/limo_voice_test（随包提供）	把 /voice/text 转成 /cmd_vel 运动命令	10.3
tf_pub.py	limo_base/scripts	发布 base_link→laser_link 等静态 TF	3.2

附录 L 配套资源包使用说明

本手册正文引用了一部分不在系统镜像里的文件：Windows 远程桌面配置、SenseVoice 语音脚本与模型、抗打滑建图的 EKF 增量包、文字识别与多点任务脚本，以及本手册的 Word 版。它们统一放在配套资源包（随车 U 盘提供，同一份内容也会发布到项目仓库）里，下文说明包内结构、校验方法和部署步骤。

L.1 包里有什么

下表按目录列出内容与用法；“对应章节”列给出该文件在正文里的出处。

目录 / 文件	内容	怎么用	对应章节
manual/	本手册 Word 版	直接打开；改版以此为准	全文
rdp/AgileX 共享桌面.rdp	Windows 远程桌面连接配置	拷到 Windows 双击；IP 变了右键“编辑”改“计算机”	9.1–9.2
docs/Windows 远程桌面连接 LIMO 小车教程.pdf	RDP 图文教程（含黑屏、VNC error 排查）	电脑上阅读	9.2
docs/抗打滑建图_EKF 融合里程计.md	EKF 融合里程计的完整说明	电脑上阅读	5.5
ekf/	抗打滑建图增量包：install.sh、3 个 launch、3 个参数文件、rf2o 日志补丁	在能 ssh 的电脑上执行 bash ekf/install.sh	5.5
scripts/limo_recognition_text_read.py	文字识别脚本	拷到车上 limo_visions/scripts，再按 8.2 装 tesseract	8.2
scripts/more_task_node.py	多点任务入口	拷到车上 set_nav_point/	11.7
voice/limo_voice_asr_sensevoice.py	录音 → SoX 降噪 → SenseVoice → /voice/text	拷到车上 ~/limo_voice_test	10.2
voice/limo_voice_command_node.py	口令 → /cmd_vel	同上，配合 10.3 使用	10.3
voice/models/sense-voice/	SenseVoice-Small 模型（229 MB，含 tokens.txt）	整个目录拷到 ~/limo_voice_test/models/	10.1
voice/noise.prof	SoX 噪声剖面	拷到 ~/limo_voice_test	10.2
docker/	ROS 2 开发容器草案与说明	见 docker/README.md	附录 D

L.2 拿到包后先做三件事

1.校验完整性。解压后进入包目录执行 sha256sum -c SHA256SUMS（Windows 用 certutil -hashfile <文件> SHA256 逐个比对），输出 OK 才算完整；模型文件 229 MB，传输中断最容易坏在这里。

2.解压到电脑本地目录，不要在 U 盘或网络盘里直接运行脚本。

3.打开包内 README.md，确认这次要用的是哪几项，再按 L.3 部署。

L.3 部署到车上

（1）脚本类（8.2 文字识别、11.7 多点任务）。改好权限再运行：

```
scp scripts/limo_recognition_text_read.py agilex@<小车 IP>:/home/agilex/agilex_ws/src/limo_visions/scripts/
scp scripts/more_task_node.py agilex@<小车 IP>:/home/agilex/agilex_ws/src/set_nav_point/
ssh agilex@<小车 IP> "chmod +x ~/agilex_ws/src/limo_visions/scripts/*.py ~/agilex_ws/src/set_nav_point/*.py"
```

（2）抗打滑建图增量包（5.5）。在能 ssh 到车的电脑上执行，脚本会自动同步文件、编译 limo_bringup 并校验安装结果：

```
bash ekf/install.sh
# 启动（三选一，不要同时跑）
ros2 launch limo_bringup limo_start_ekf.launch.py
ros2 launch limo_bringup limo_cartographer_ekf.launch.py
ros2 launch limo_bringup limo_rtab_rgbd_ekf.launch.py
```

（3）语音模块（10.x）。先把脚本和模型放到车上，再建虚拟环境：

```
scp -r voice/*.py voice/models voice/noise.prof agilex@<小车 IP>:~/limo_voice_test/
ssh agilex@<小车 IP>
python3 -m venv ~/limo_voice_env && source ~/limo_voice_env/bin/activate
python -m pip install sherpa-onnx numpy     # SoX 用 sudo apt install sox
python ~/limo_voice_test/limo_voice_asr_sensevoice.py   # 另开终端看 /voice/text
```

（4）Windows 远程桌面（9.2）。把 rdp/AgileX 共享桌面.rdp 拷到 Windows，双击即可；若提示无法连接，先在车上执行 hostname -I 确认 IP，再右键该文件“编辑”，把“计算机”改成新的 IP:3389。

L.4 常见问题

模型缺失或路径不对：报错里通常带 FileNotFoundError 或找不到 model.int8.onnx。确认 ~/limo_voice_test/models/sense-voice/ 下同时有 model.int8.onnx 与 tokens.txt，两个文件缺一不可。

仓库里拉不下模型：GitHub 单文件上限 100 MB，229 MB 的模型请用发布附件（Release）或 git lfs，最省事的是直接用随车 U 盘里的 zip。

ekf/install.sh 报 ssh 连接失败：先在电脑上确认 ssh agilex@<小车 IP> 能登录，且已 source 过 ROS 环境；脚本依赖这台电脑能直连小车。

脚本无法执行：scp 过去后补 chmod +x；用 python3 显式指定解释器也可以绕过执行位问题。