



“玉兔” L4 级自动驾驶 教学实验平台 使用说明书 V2.0



目 录

第 1 章 整车介绍与遥控操作	1
1.1 核心部件说明	1
1.2 遥控器操作说明	3
第 2 章 装调标定教学软件	4
2.1 超声波雷达调试	4
2.2 超声波雷达标定	10
2.3 前毫米波雷达安装	16
2.4 前毫米波雷达调试	22
2.5 前毫米波雷达标定	31
2.6 角毫米波雷达调试	43
2.7 激光雷达安装	50
2.8 激光雷达调试	56
2.9 激光雷达标定	57
2.10 单目相机调试	66
2.11 单目相机标定	67
2.12 环视相机调试	72
2.13 环视相机标定	72
2.14 组合导航调试	86
2.15 组合导航标定	87
2.16 线控底盘线控测试	94
2.17 线控底盘 CAN 通讯	100
2.18 线控底盘中位标定	113
2.19 毫米波雷达和相机联合标定	116
2.20 激光雷达和相机联合标定	124
第 3 章 自动驾驶教学软件	136
3.1 使用建议	136
3.2 录制数据包	136
3.3 PCD 点云地图制作	141



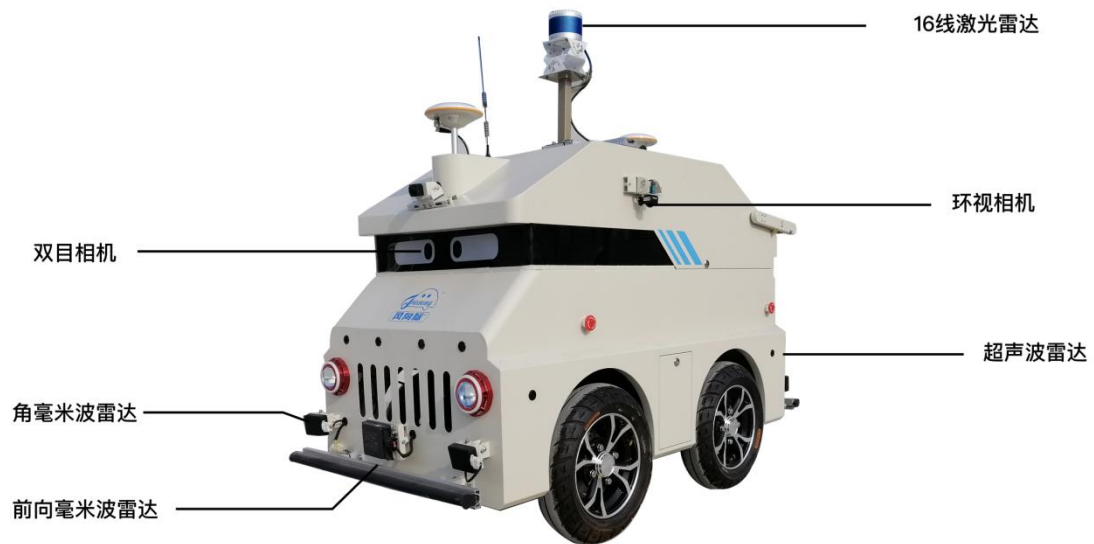
3.4 Waypoint 路径点制作	153
3.5 绘制 HD 高精地图	164
3.6 标记位置	180
3.7 道路测试-高级教学-方案一	188
3.8 道路测试-初级教学-方案一	211
3.9 道路测试-演示模式-方案一	217
3.10 道路测试-高级教学-方案二	219
3.11 道路测试-初级教学-方案二	236
3.12 道路测试-演示模式-方案二	242
3.13 道路测试-高级教学-OBU	245
3.14 道路测试-初级教学-OBU	264
第 4 章 故障排查与考核实训	276
4.1 实验一	276
4.2 实验二	278
4.3 实验三	281
4.4 实验四	283
4.5 实验五	286
4.6 实验六	288
4.7 实验七	290
4.8 实验八	292
4.9 实验九	294
4.10 实验十	297
4.11 实验十一	300
4.12 实验十二	302
4.13 实验十三	304
4.14 实验十四	307
4.15 实验十五	311
4.16 实验十六	315
4.17 实验十七	318



4.18 实验十八	320
4.19 实验十九	323
4.20 实验二十	326
4.21 实验二十一	328
4.22 实验二十二	333
4.23 实验二十三	338
4.24 实验二十四	344
第 5 章 自动驾驶仿真测试系统	351
5.1 使用准备	351
5.2 软件使用说明	353
第 6 章 V2X 车路协同 OBU 端	372
6.1 部件描述	372
6.2 设备接口描述	373
6.3 OBU 调试	373
6.4 OBU 与 PAD 通讯	378
6.5 V2X 预警事件触发	384

第 1 章 整车介绍与遥控操作

1.1 核心部件说明



车辆外观



内部瞰图



计算单元



组合导航



超声波雷达控制盒



单目相机



前毫米波雷达



角毫米波雷达



激光雷达



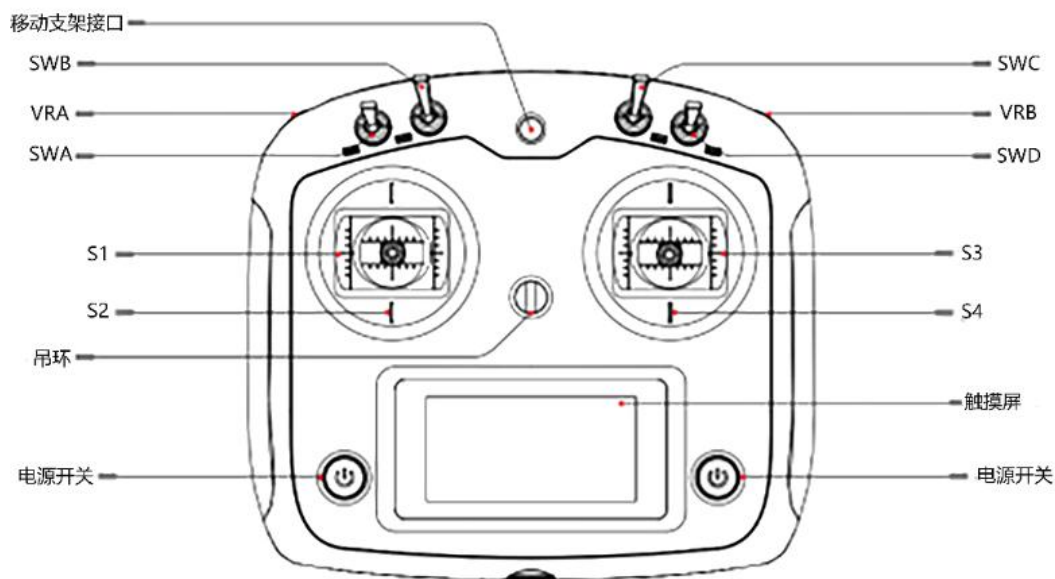
环视相机



设故板

主要部件

1.2 遥控器操作说明



SWA 为控制模式切换拨杆，两档位，以遥控器正面向上为例：**SWA** 拨杆处于上方时为遥控器控制模式，**SWA** 拨杆处于下方时为自动驾驶模式；

SWB 为档位切换拨杆，三档位，拨杆在中间时，车体处于 **N** 档，不接收前后运动控制信号；拨杆往上时切换至 **D** 档，底盘才能接收 **S4** 摇杆发送的前进运动控制信号，往前运动；拨杆往下时切换至 **R** 档，底盘才能接收 **S4** 摇杆发送的后退运动控制信号，往后运动；

VRA 为喇叭控制旋钮，自动回位，向下波动旋钮时使能喇叭；

VRB 为驻车请求控制旋钮，旋钮往上拨时发送驻车请求，启动驻车制动装置；旋钮往下拨时发送解除驻车请求，松开驻车制动装置；

S4 摇杆为油门控制，控制 **FR-07** 前进和后退的速度；**S3** 摇杆控制前轮的转向；

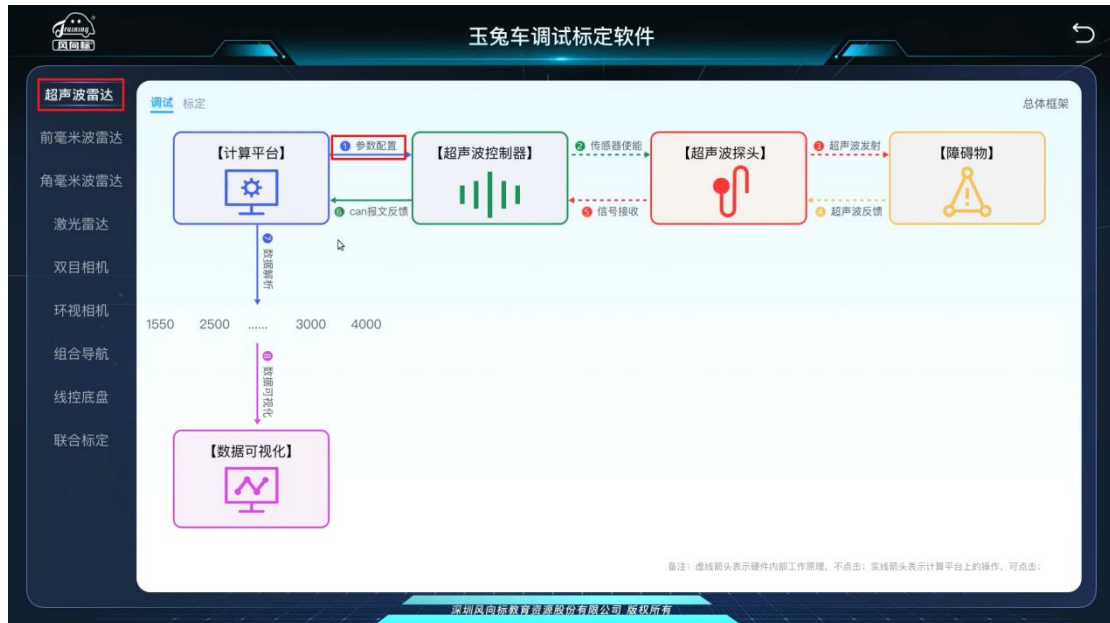
SWC 为 **S4** 摇杆高中低速控制模式，以前进档为例：**SWC** 处于最上方时 **S4** 摇杆控制车辆的最高速为低速模式行驶；**SWC** 处于中间位置时 **S4** 摇杆控制车辆为中速模式行驶；**SWC** 处于最下方位置时 **S4** 摇杆控制车辆为高速模式行驶；

电源开关 为遥控器电源控制开关，当遥控器处于关机状态同时长按显示器**两侧**电源开关控制遥控器开机；遥控器处于开机状态，同时长按显示器**两侧**电源开关进行关机，如遥控器接收机处于上电状态，同时长按显示器**两侧**电源开关无法进行关机，需要卸载电池进行关机；

第 2 章 装调标定教学软件

2.1 超声波雷达调试

1) 找到超声波雷达下的“参数配置”；




The screenshot shows the 'Yumi Car Debugging and Calibration Software' interface, specifically the '报文发送' (Message Sending) section. The '报文发送' (Message Sending) section includes fields for '波特率' (Baud Rate) set to 500k bps, '帧格式' (Frame Format) set to '标准帧' (Standard Frame), and '帧类型' (Frame Type) set to '数据帧' (Data Frame). Below these, there are fields for '帧ID(Hex)' (Frame ID (Hex)) set to 0X601, '数据长度dico' (Data Length dico) set to 0X3, and a '数据' (Data) field with a '发送' (Send) button. The '报文编写' (Message Writing) section includes a '工作模式' (Working Mode) dropdown set to '大于2m', a '总线值' (Bus Value) dropdown set to '0xb1', and a '2进制' (Binary) dropdown set to '10110001'. The '探头状态' (Sensor Status) dropdown is set to '开启' (On). The '数据' (Data) field is populated with a 16-bit binary sequence: 01 A2 BF. Below the '数据' field, there is a table showing the data in hexadecimal and binary format. The table has 8 rows (0-7) and 8 columns (7-0). The data is as follows:

	7	6	5	4	3	2	1	0
0	01	A2	BF					
1	01	A2	BF					
2	01	A2	BF					
3	01	A2	BF					
4	01	A2	BF					
5	01	A2	BF					
6	01	A2	BF					
7	01	A2	BF					

2) 在考核模式下需要根据协议自行计算开启所有超声波探头的报文并填写报文到数据部分，在非考核模式（演示模式）下选择配置选项，软件会自动计算所需



CAN 报文：

ID	DLC	Byte0	Byte1	Byte2
0x601	0x03	0xb1~0xba	0x10	0xff

其中数据区 Byte0~Byte2 意义如下：

一帧格式	意义简称内容
数据区第一字节	<p>0xb1 表示远距离（建议大于 2 米）索要测量长度数据，不间断返回距离数据，盲区 290nm 最远显示距离：5000nm</p> <p>0xb2 表示最远距离（建议大于 2 米）索要测量长度数据，返回一次距离数据，盲区 290nm 最远显示距离：5000nm</p> <p>0xb3 表示较远距离（建议 2 米内）索要测量长度数据，不间断返回距离数据，盲区 250nm 最远显示距离：5000nm</p> <p>0xb4 表示较远距离（建议 2 米内）索要测量长度数据，返回一次距离数据，盲区 250nm 最远显示距离：5000nm</p> <p>0xb5 表示稍远距离（建议 1.5 米内）索要测量长度数据，不间断返回距离数据，盲区 205nm 最远显示距离：5000nm</p>
	<p>0xb6 表示稍远距离（建议 1.5 米内）索要测量长度数据，返回一次距离数据，盲区 205nm 最远显示距离：5000nm</p> <p>0xb7 表示近距离（建议 1 米内）索要测量长度数据，不间断返回距离数据，盲区 200nm 最远显示距离：5000nm</p> <p>0xb8 表示近距离（建议 1 米内）索要测量长度数据，返回一次距离数据，盲区 200nm 最远显示距离：5000nm</p> <p>0xb9 表示近距离（建议 0.2 米内）索要测量长度数据，不间断返回距离数据，盲区为 130nm（此指令抗干扰性能比较差，对电源要求较高，若探头插上后，输出一直是 130nm 可微调对应接口的中周），最远显示距离：5000nm</p> <p>0xba 表示近距离（建议 0.2 米内）索要测量长度数据，返回一次距离数据，盲区为 130nm（此指令抗干扰性能比较差，对电源要求较高，若探头插上后，输出一直是 130nm 可微调对应接口的中周），最远显示距离：5000nm</p> <p>0xbb 表示雨天工作模式（建议雨天时使用该指令）索要测量长度数据，不间断返回距离数据，盲区为 290nm，最远探测距离 2500nm 左右，最远显示距离：5000nm</p> <p>0xbc 表示雨天工作模式（建议雨天时使用该指令）索要测量长度数据，返回一次距离数据，盲区为 290nm，最远探测距离 2500nm 左右，最远显示距离：5000nm</p>
数据区第 2-3 字节	<p>工作探头选择 2 字节定义如下：</p> <p>2 位 16 进制，先高字节，后低字节组成的 16 位来确定索要哪几个探头数据及机号，其中低 12 位（0~11）确定索要数据的探头号，高 4 位（12~15）确定机号。</p> <p>低 12 位中第 n 位为 1，则第 n+1 号探头需要输出距离数据 例如：0x10 0x35 则表示机号为 01 的模块返回 1 3 5 6 号探头的数据</p> <p>又如：0x10 0x00 则停止索要探头输出数据</p> <p>出厂默认机号为 01 号</p> <p>例如：此时需要 1~12 头数据，可发 b3 1f ff，约定 12 头数据全部需要，此时探头选择的 2 个字节必须是 1f ff；若需要探头停止工作，此时探头选择的 2 个字节是 10 00 可发送 b3 10 00（不需要距离数据时让探头停止工作，工作一段时间让探头停止工作几秒，都能提高探头使用寿命）</p> <p>注意：本产品功能只能全部输出 12 路探头的数据，不能选择哪几个探头工作，因此索要测距数据时该 2 个字节必须是 1f ff 需要探头停止工作时该 2 个字节必须是 10 00</p>

由上表指令可以看出探测距离越远，盲区越大，同时角度也越



3) 找到超声波雷达下的“CAN 报文反馈”，如果有 **CAN 报文不断刷新**则说明超声波雷达工作正常，否则异常；



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

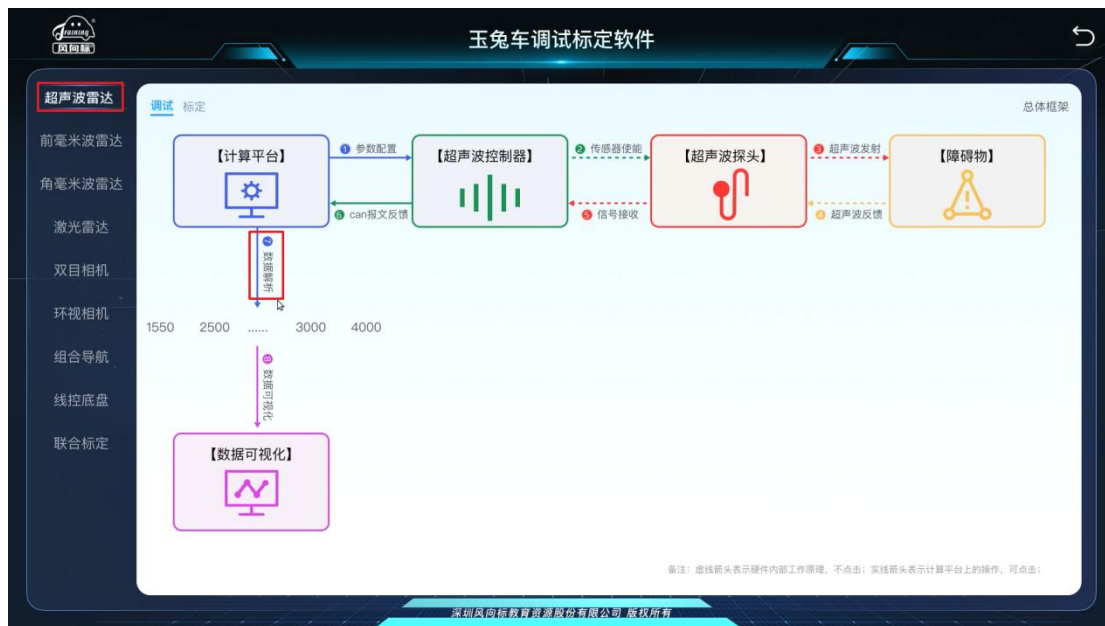
调试 标定

清除 Can报文反馈

序号	时间	can通道	传输方向	ID号	数据类型	帧格式	长度	数据
0	14:43:15.81	can1	接收	0x611	数据帧	标准帧	8	50 00 50 00 50 00 50 00
1	14:43:15.81	can1	接收	0x612	数据帧	标准帧	8	20 05 25 00 02 75 02 50
2	14:43:15.81	can1	接收	0x613	数据帧	标准帧	8	02 50 02 65 21 50 50 00
3	14:43:15.81	can1	接收	0x611	数据帧	标准帧	8	50 00 50 00 50 00 50 00
4	14:43:15.82	can1	接收	0x612	数据帧	标准帧	8	20 05 24 95 02 75 02 50
5	14:43:15.82	can1	接收	0x613	数据帧	标准帧	8	02 50 02 65 06 60 50 00
6	14:43:16.00	can1	接收	0x611	数据帧	标准帧	8	50 00 50 00 50 00 50 00
7	14:43:16.00	can1	接收	0x612	数据帧	标准帧	8	20 05 25 00 02 75 02 50
8	14:43:16.00	can1	接收	0x613	数据帧	标准帧	8	02 50 02 65 21 50 50 00
9	14:43:16.01	can1	接收	0x611	数据帧	标准帧	8	50 00 50 00 50 00 50 00
10	14:43:16.01	can1	接收	0x612	数据帧	标准帧	8	20 05 25 00 02 75 02 50
11	14:43:16.01	can1	接收	0x613	数据帧	标准帧	8	02 50 02 65 06 60 50 00

深圳风向往标教育股份有限公司 版权所有

4) 找到超声波雷达下的“数据解析”，点击播放按钮，可以抓取此刻 CAN 报文，在考核模式下需要自行计算报文中含义，在非考核模式（教学模式）下，软件会自动计算该报文的含义；



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

调试 标定

考核模式 ☒

读取报文

数据解析

ID号	d1 0x0611								d2 0x0612								d3 0x0613								
字节序号	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
原始报文(16进制)	15	50	25	00	30	00	40	00	15	50	25	00	30	00	40	00	15	50	25	00	30	00	40	00	从can网络中获取id为0x0611、0x0612、0x0613的报文数据
转换关系	↓ ↓ ↓ ↓								↓ ↓ ↓ ↓								↓ ↓ ↓ ↓								每个报文包含8个字节，序号分别为0-7
超声波序号	1路		2路		3路		4路		5路		6路		7路		8路		9路		10路		11路		12路		超声波ID号
解析后数据(10进制,mm)	<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		实际的12路超声波距离值

*换算公式：(d[i]/16*10+d[j]/16)*100+d[i+1]*16*10+d[j+1]*16 *备注：d指代ID号的报文，取d1、d2、d3，j指代报文中的字节序号，取0、2、4、6 考核模式需要自行解析CAN报文

深圳风向标教育装备股份有限公司 版权所有

玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

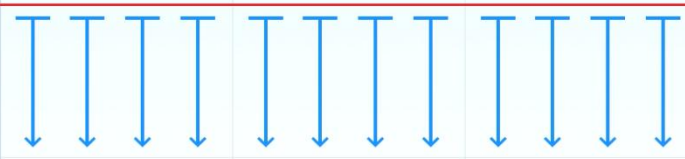
环视相机

组合导航

线控底盘

联合标定

调试 标定
考核模式 ☐ 抽取报文 || 数据解析

ID号	d1 0x0611								d2 0x0612								d3 0x0613								
字节序号	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
原始报文(16进制)	50	00	50	00	50	00	35	45	50	00	50	00	02	75	02	95	03	45	03	60	16	45	15	50	每个报文包含8个字节, 序号分别为0-7
转换关系																									原始报文字节到超声波序号的对应关系
超声波序号	1路		2路		3路		4路		5路		6路		7路		8路		9路		10路		11路		12路		超声波ID号
解析后数据 (10进制,mm)	5000		5000		5000		3545		5000		5000		0275		0295		0345		0360		1645		1550		实际的12路超声波距离值

*换算公式: $(d[i]/16*10+d[j]/16*10)*100+d[k]+1)*10*d[l]+1)*16$ *备注: d指代ID号的报文, 取d1, d2, d3, i指代报文中的字节序号, 取0, 2, 4, 6

在非考核模式下, 软件会自动解析CAN报文

深圳风向往标教育科技股份有限公司 版权所有

5) 找到超声波雷达下的“数据可视化”，此功能是把 12 个超声波探头测量的数据进行可视化，直观的看到车辆周围信息；





2.2 超声波雷达标定

2.2.1 场地要求

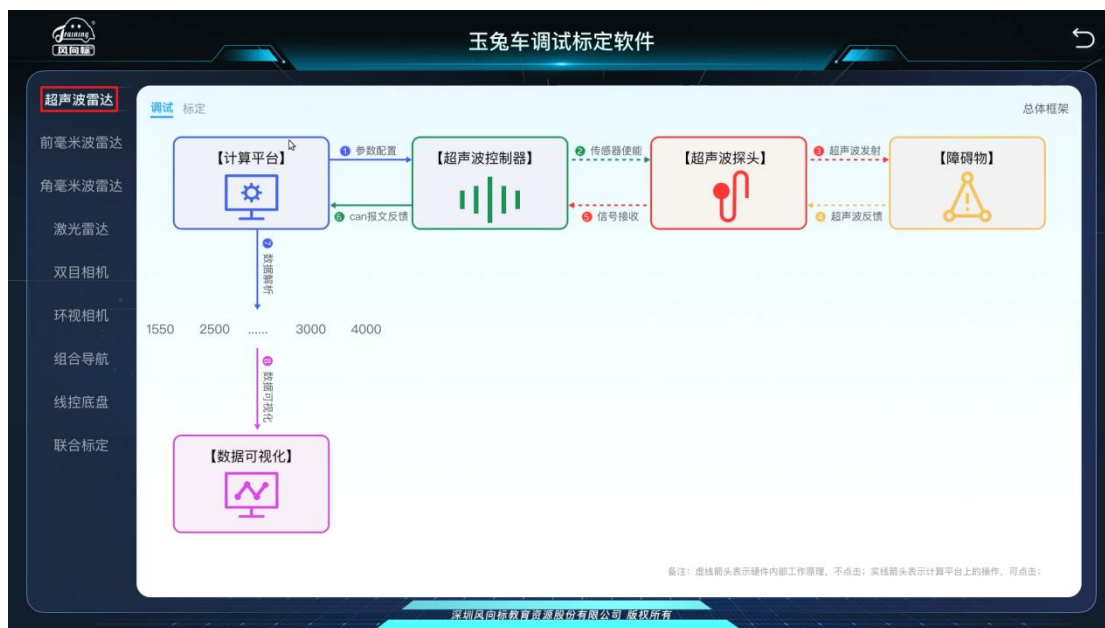
要求地面尽量水平，无沟槽无凸起物，标定区域为 **6m*3m** 以上的空旷区域。

2.2.2 准备工作

1) 准备激光测距仪，棋盘标定板。



2) 打开“装调标定教学软件”，找到“超声波雷达”一栏。



3) 点击“参数配置”，在数据一栏输入 **B3 1F FF**（或者输入 **B7 1F FF** 两者只是量程不一样，**B7** 量程短，在短距离精度高）后，点击“发送”按钮，让超声波探头工作。





4) 点击“CAN 报文反馈”，确认超声波探头是否工作，如果有报文不断刷新，则工作正常，否则异常。



5) 点击“标定”，选择需要标定探头编号（探头编号可以从“数据可视化”获得）。



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

调试 **标定**

超声波路数

类型	软件输出距离(mm)	误差(mm)	操作
盲区距离	500	200	设置
触发距离	1200	400	设置
最远距离	3500	500	设置



☒ 标定结果

*误差公式 $y=ax+b$

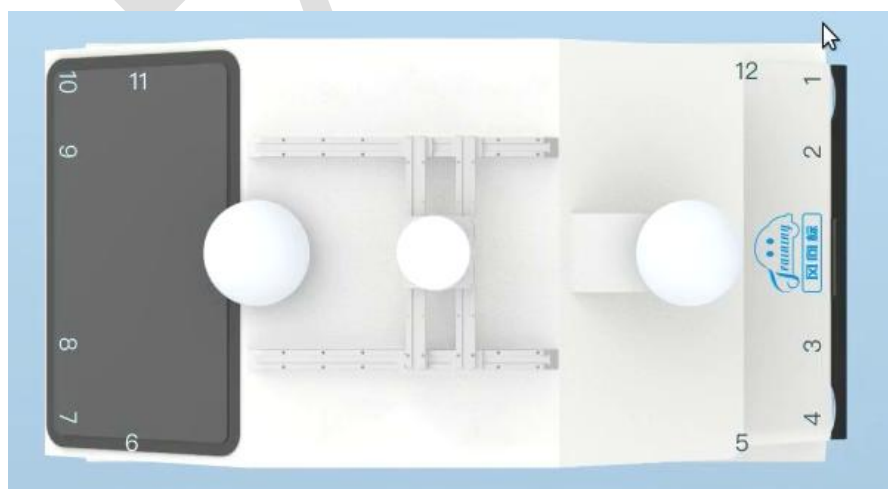
保存位置 保存

☒ 验证结果

验证

测量距离 误差距离

深圳风向往标教育科技股份有限公司 版权所有



6) 长按“DIST”按钮启动激光测距仪，点击“UNIT”按钮设置测量起点为尾部开始。



7) 移动棋盘标定板，至盲区距离、触发距离和最远距离。





盲区距离，理论值为 250mm，其含义是在 250mm 内的物体都无法测量，探头将一直返回测量距离为 250mm。

最远距离，理论值为 5000mm，其含义是在 5000mm 以外的物体都无法测量，探头将一直返回测量距离为 5000mm。

触发距离，其含义为在盲区距离和最远距离的这个范围内 250~5000mm 都有效，即传感器正常的测量范围，当物体在大于盲区距离小于触发距离的范围，程序会报警。

注！上述为厂家提供的理论值，可以根据实际情况进行更改。

8) 当移动到三个位置并将激光测距仪的数值到软件输入后，点击标定，生成误差公式的 a 和 b，以及画出 x 和 y 的对应图表（注意！激光测距仪测量完成后，需要移开，以免造成超声波探头测量距离错误）。



9) 在触发距离的范围内，任意移动棋盘标定板，将激光测距仪的读数填入软件，



点击验证。



2.3 前毫米波雷达安装

2.3.1 场地要求

要求地面尽量水平，无沟槽无凸起物。

2.3.2 准备工作

1) 准备内六角扳手套装、十字螺丝刀、角度尺、水平仪和防护手套并检查是否可以正常使用；



2) 点击电源开关（绿灯熄灭），关闭车辆电源；



2.3.3 安装过程

1) 使用内六角扳手，安装毫米波雷达并紧固螺栓；



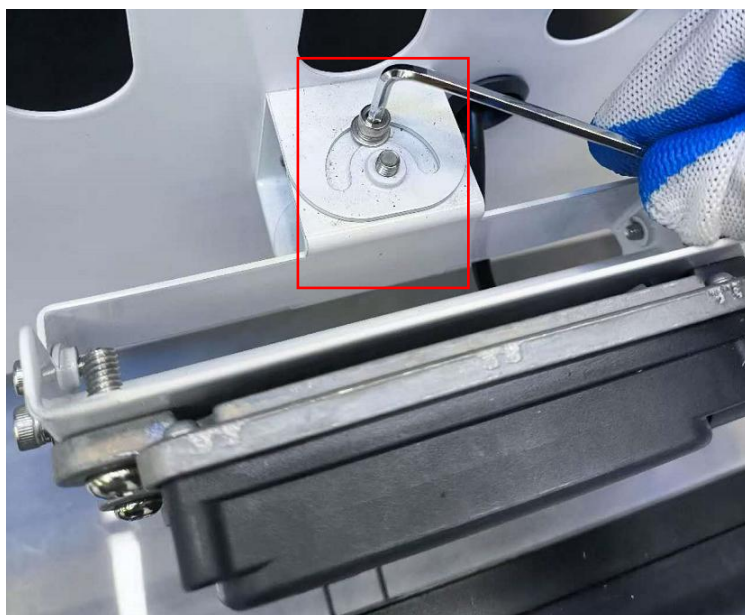
2) 安装毫米波雷达连接线束；



3) 在工具车上校 0 角度尺，使用角度尺测量毫米波雷达安装的横向水平角，90-角度尺测量值为横向水平角，此角度应在 $0 \pm 3^\circ$ 范围；



若角度不在 $0 \pm 3^\circ$ 范围，可以使用内六角扳手拧松螺栓，调节横向水平角，调节完成后，拧紧螺栓。

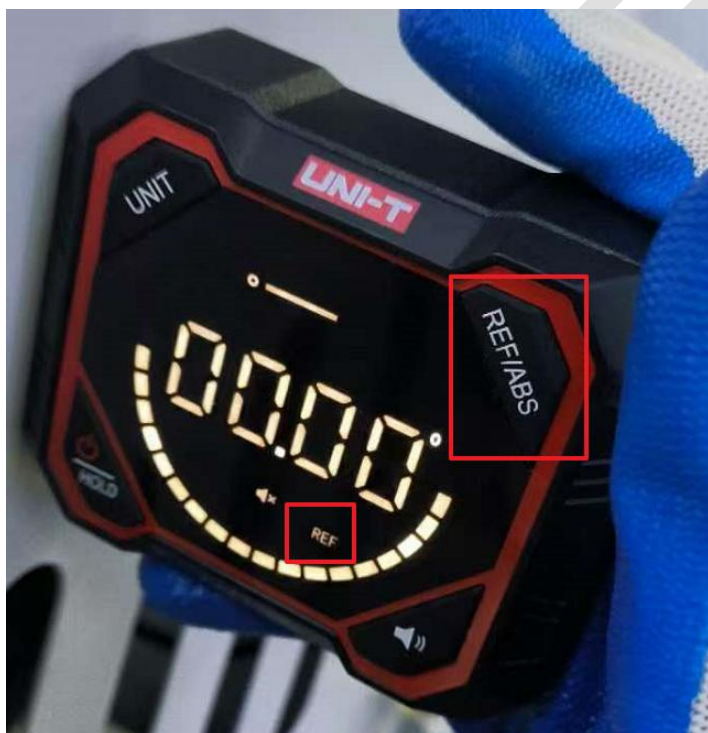


4) 长按水平仪电源按钮开机，长按水平仪 UNIT 按钮调节水平仪测量单位为度；





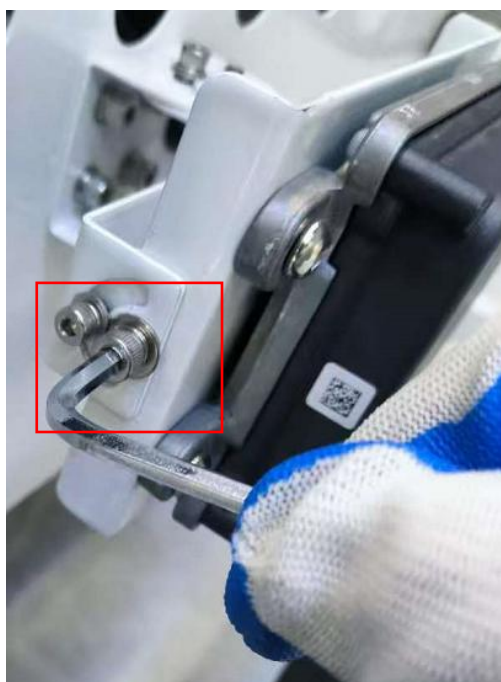
5) 按下 REF/ABS 按钮将水平仪设置为相对模式（用于测量相对水平），此时需要将水平仪放置在基准面（这里以毫米波雷达支架安装面为基准面）；



6) 将水平仪放置在毫米波雷达的检测面上，测量纵向水平角，此角度应在 $0 \pm 0.3^\circ$ 的范围；



若角度不在 $0 \pm 0.3^\circ$ 的范围，可以使用内六角扳手拧松螺栓，调节纵向水平角，调节完成后，拧紧螺栓。



2.4 前毫米雷达调试

1) 找到前毫米波雷达下的“参数配置”；



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

调试 标定

考核模式 ☒ 配置参数

☒ 报文发送

波特率: 500k bps 帧格式: 标准帧 帧类型: 数据帧

帧ID(Hex): 0X 200 数据长度: 0XB 数据: 01 A2 BF C3 D4 F5 A3 D1 发送

报文编写

雷达功率: 标准 0 000

输出类型: 无 0 00

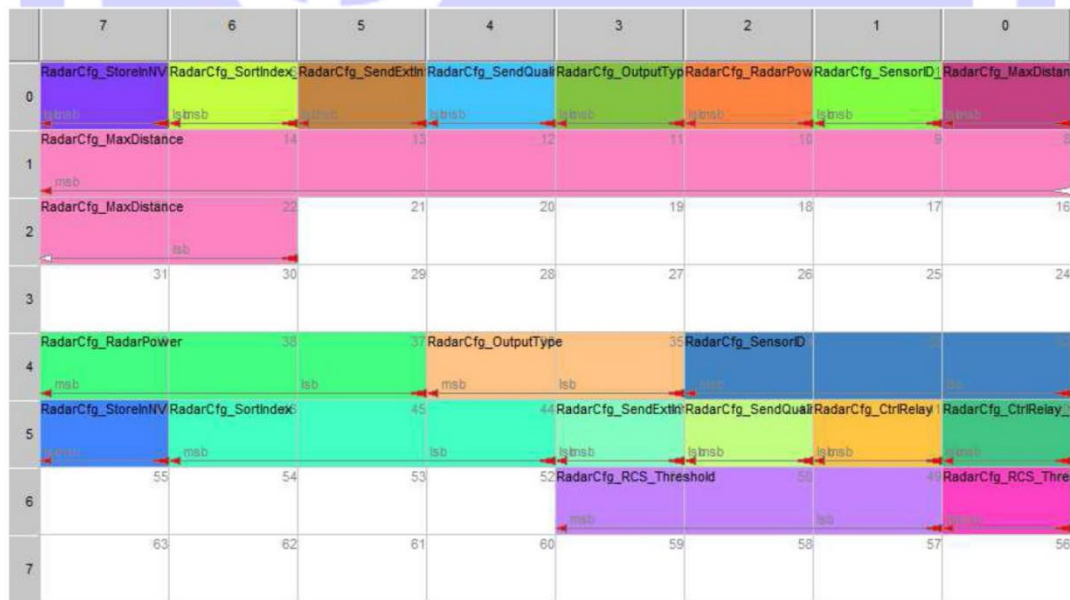
	7	6	5	4	3	2	1	0
0	7	6	5	4	3	2	1	0
1	15	14	13	12	11	10	9	8
2	23	22	21	20	19	18	17	16
3	31	30	29	28	27	26	25	24
4	35	34	33	32	31	30	29	28
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56

深圳风向往标教育股份有限公司 版权所有

2)在**考核模式**下根据协议自行计算所需配置的**CAN 报文**并填写报文到数据部分，在非**考核模式（教学模式）**下选择配置选项，软件会自动计算所需**CAN 报文**；

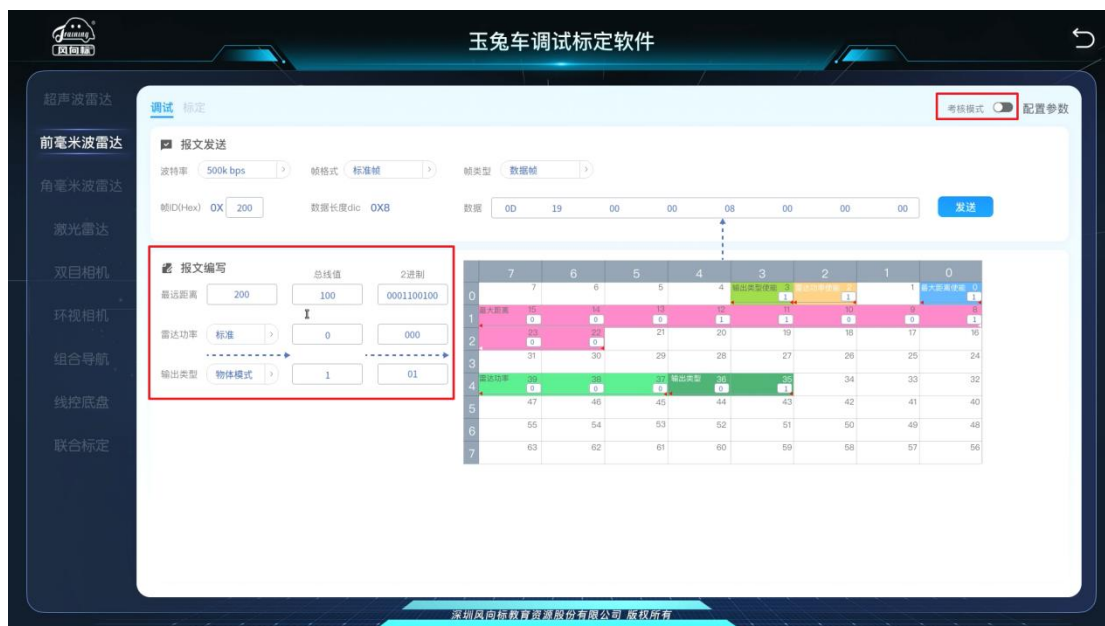
6.1 雷达配置 (0x200)

ARS/SRR传感器可以通过RadarCfg(0x200)来配置。参数可以单独更改或组合更改，每个参数都包含一个有效位（例如参数RadarCfg_MaxDistance的有效位为RadarCfg_MaxDistance_valid）。如果有效位被设置为valid（0x1），相应的参数将在ARS中更新，否则就被忽略。



信号	起始位	长度	最小值	最大值	分辨率	单位
RadarCfg_MaxDistance_valid	0	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_SensorID_valid	1	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_RadarPower_valid	2	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_OutputType_valid	3	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_SendQuality_valid	4	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_SendExtInfo_valid	5	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_SortIndex_valid	6	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_StoreInNVM_valid	7	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_MaxDistance	22	10	0	2046	2	m
RadarCfg_SensorID	32	3	0	7	1	传感器ID0x0~0x7

RadarCfg_OutputType	35	2	0	3	1	0x0: 无 0x1: objects 0x2: clusters
RadarCfg_RadarPower1	37	3	0	7	1	0x0: 标准 0x1: -3dB发射增益 0x2: -6dB发射增益 0x3: -9dB发射增益
RadarCfg_CtrlRelay_valid	4	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_CtrlRelay	41	1	0	1	1	0x0: 未激活 0x1: 激活
RadarCfg_SendQuality	42	1	0	1	1	0x0: 未激活 0x1: 激活
RadarCfg_SendExtInfo	43	1	0	1	1	0x0: 未激活 0x1: 激活
RadarCfg_SortIndex	44	3	0	7	1	0x0: 未排序 0x1: 按距离排序 0x2: 按RCS排序
RadarCfg_StoreInNVM	47	1	0	1	1	0x0: 未激活 0x1: 激活
RadarCfg_RCS_Threshold_valid	48	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_RCS_Threshold	49	3	0	7	1	0x0: 标准灵敏度 0x1: 高灵敏度
RadarCfg_InvalidClusters_valid2	52	1	0	1	1	0x0: 无效 0x1: 有效
RadarCfg_InvalidClusters2, 3	56	8	0	255	1	0x00: 无效 0x01: 启用所有Invalidclusters 0x02: 启用低RCS动态 0x04: 启用低RCS静态 0x08: 启用无效范围率 0x10: 启用范围<1m 0x20: 启用自我反射 0x40: 启用覆盖静止



3) 找到前毫米波雷达下的“CAN 报文反馈”，如果发现 CAN 报文反馈有报文在
不断刷新，则说明传感器工作正常，否则异常；



序号	时间	can通道	传输方向	ID号	帧类型	帧格式	长度	数据
226	15:40:34:30	can1	接受	0x60B	数据帧	标准帧	8	06 52 4C 2C 80 20 01 7C
227	15:40:34:30	can1	接受	0x60B	数据帧	标准帧	8	09 4F 8B FF 80 20 01 92
228	15:40:34:30	can1	接受	0x60B	数据帧	标准帧	8	0D 52 84 1D 80 20 01 85
229	15:40:34:30	can1	接受	0x60B	数据帧	标准帧	8	0E 50 E3 F3 80 20 01 73
230	15:40:35:29	can1	接受	0x60B	数据帧	标准帧	8	04 52 3C 54 80 20 01 8F
231	15:40:35:29	can1	接受	0x60B	数据帧	标准帧	8	05 52 44 82 80 1F E1 7F
232	15:40:35:29	can1	接受	0x60B	数据帧	标准帧	8	06 52 4C 2C 80 20 01 7C
233	15:40:35:29	can1	接受	0x60B	数据帧	标准帧	8	09 4F 8B FF 80 20 01 92
234	15:40:35:29	can1	接受	0x60B	数据帧	标准帧	8	0D 52 84 1D 80 20 01 85
235	15:40:35:29	can1	接受	0x60B	数据帧	标准帧	8	0E 50 E3 F3 80 20 01 73
236	15:40:35:29	can1	接受	0x60B	数据帧	标准帧	8	0F 4E A3 FF 80 20 01 9C
237	15:40:35:33	can1	接受	0x60B	数据帧	标准帧	8	10 51 D8 EF 80 20 01 7D
238	15:40:35:30	can1	接受	0x60B	数据帧	标准帧	8	11 52 44 3F 80 20 01 90
239	15:40:35:30	can1	接受	0x60B	数据帧	标准帧	8	00 4E F4 1D 80 20 01 8F
240	15:40:36:30	can1	接受	0x60B	数据帧	标准帧	8	00 4E F4 1D 80 20 01 8F
241	15:40:36:30	can1	接受	0x60B	数据帧	标准帧	8	01 50 1C 1B 80 20 01 76
242	15:40:36:30	can1	接受	0x60B	数据帧	标准帧	8	02 50 AB FE 80 20 01 80
243	15:40:36:30	can1	接受	0x60B	数据帧	标准帧	8	03 4E 84 12 80 20 01 94
244	15:40:36:30	can1	接受	0x60B	数据帧	标准帧	8	04 52 3C 54 80 20 01 8F
245	15:40:36:30	can1	接受	0x60B	数据帧	标准帧	8	05 52 44 82 80 1F E1 7D
246	15:40:36:30	can1	接受	0x60B	数据帧	标准帧	8	06 52 4C 2C 80 20 01 7B
247	15:40:36:30	can1	接受	0x60B	数据帧	标准帧	8	07 52 3C 96 80 20 01 85
248	15:40:36:30	can1	接受	0x60B	数据帧	标准帧	8	09 4F 8B FF 80 20 01 92
249	15:40:36:30	can1	接受	0x60B	数据帧	标准帧	8	0D 52 84 1D 80 20 01 86
250	15:40:37:23	can1	接受	0x60B	数据帧	标准帧	8	00 4E F4 1D 80 20 01 8E

4) 找到前毫米波雷达下的“数据解析”；





5) 点击播放按钮，可以抓取此刻 CAN 报文，当抓到报文后点击暂停按钮再进行 CAN 报文解析，在考核模式下需要自行计算报文中含义(在 6)有说明怎么计算)，在非考核模式（教学模式）下软件会自动计算该报文的含义；



6) 将数据部分的十六进制转换二进制填入字段表格；

0D = 0000 1101（不需要填入数据表格）

4E = 0100 1110

5C = 0101 1100

01 = 0000 0001



80 = 1000 0000

20 = 0010 0000

01 = 0000 0001

7C = 0111 1100

	7	6	5	4	3	2	1	0
0	7	6	5	4	3	2	1	0
1	15	14	13	12	11	10	9	8
2	23	22	21	20	19	18	17	16
3	31	30	29	28	27	26	25	24
4	39	38	37	36	35	34	33	32
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56

根据协议计算纵向距离、横向距离、纵向速度、横向速度和雷达散射面积（RCS）；

表45：Object_1_一般-信息内容(0x60B)

信号	起始位	长度	最小值	最大值	分辨率	单位
Object_ID	0	8	0	255	1	
Object_DistLong	19	13	-500	+1138.2	0.2	m
Object_DistLat	24	11	-204.6	+204.8	0.2	m
Object_VrelLong	46	10	-128.00	127.75	0.25	m/s
Object_DynProp	48	3	0	7	1	0x0: 移动 0x1: 静止 0x2: 来向 0x3: 可能静止 0x4: 未知 0x5: 横穿静止 0x6: 横穿移动 0x7: 停止
Object_VrelLat	53	9	-64.00	63.75	0.25	m/s
Object_RCS	56	8	-64.0	63.5	0.5	dBm ²

根据协议和字段表格，拼接每个数据字段的二进制，然后将二进制转换十进制；

注！存储方式为摩托罗拉（大端存储）

纵向距离：0 1001 1100 1011 = 2507

横向距离：100 0000 0001 = 1025

纵向速度：10 0000 0000 = 512



横向速度： $1\ 0000\ 0000 = 256$

雷达散射面积： $0111\ 1100 = 124$

根据协议的分辨率和最小值将十进制求得最终值：

纵向距离： $2507 * 0.2 - 500 = 1.4$

横向距离： $1025 * 0.2 - 204.6 = 0.4$

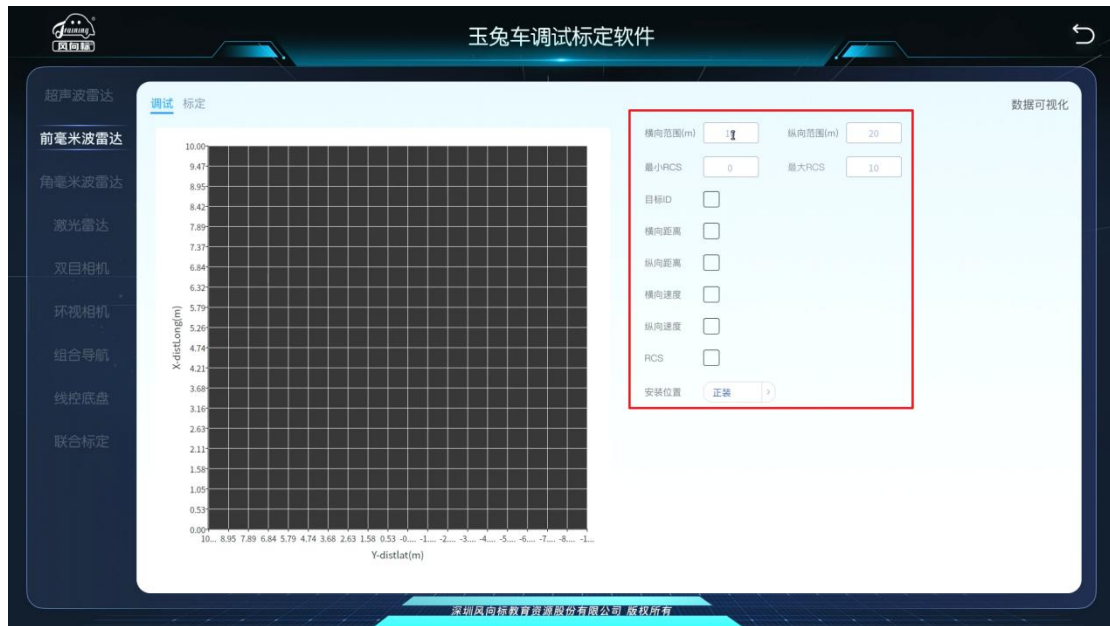
纵向速度： $512 * 0.25 - 128 = 0$

横向速度： $256 * 0.25 - 64 = 0$

雷达散射面积： $124 * 0.5 - 64 = -2.0$

7) 找到前毫米波雷达下的“数据可视化”，可以对检测物体的横向范围、纵向范围以及 RCS 雷达散射截面进行限制，还可以自由勾选所需显示数据；





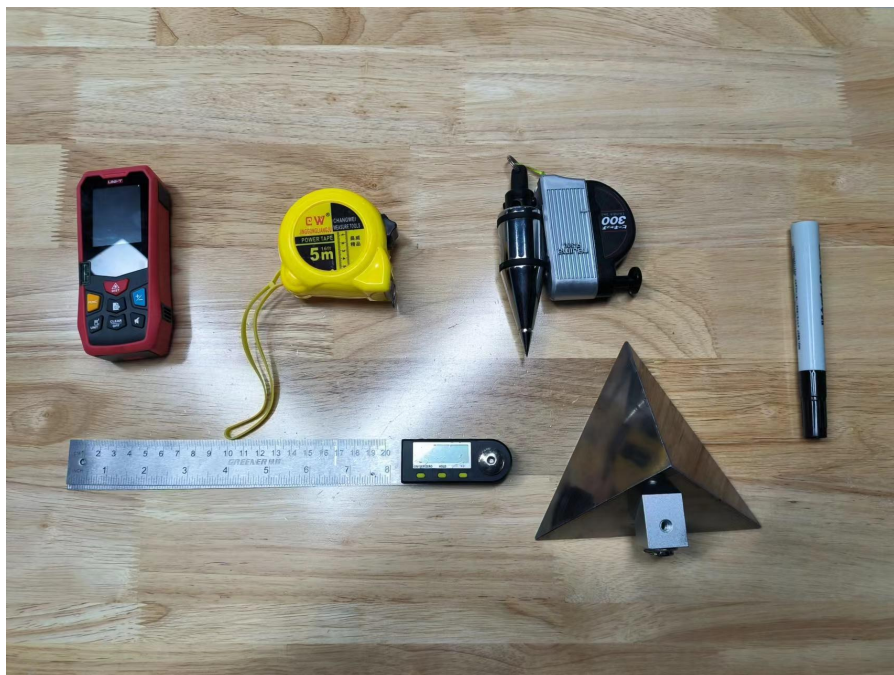
2.5 前毫米波雷达标定

2.5.1 场地要求

要求地面尽量水平，无沟槽无凸起物，标定区域为 7m*3m 以上的空旷区域。

2.5.2 准备工作

1) 准备激光测距仪、铅垂线、角度尺、卷尺、记号笔和角反射器，并检查是否可以正常使用；



- 2) 将车辆停稳，车头摆正；
- 3) 确保车辆没有放置重物，避免影响车辆水平；

2.5.3 标定过程

- 1) 调节角反射器高度，使角反射器中心与前毫米波雷达中心高度一致；



- 2) 使用铅锤线，将毫米波雷达检测面中心对应的前保险杠垂直到地面，并使用

记号笔标记，记作 A 点（**特别注意，使用铅垂线时不要被固定针扎到手**）；





3) 使用铅锤线，将毫米波雷达检测面中心旁边任意点对应的前保险杠垂直到地面，并使用记号笔标记，记作 B 点；

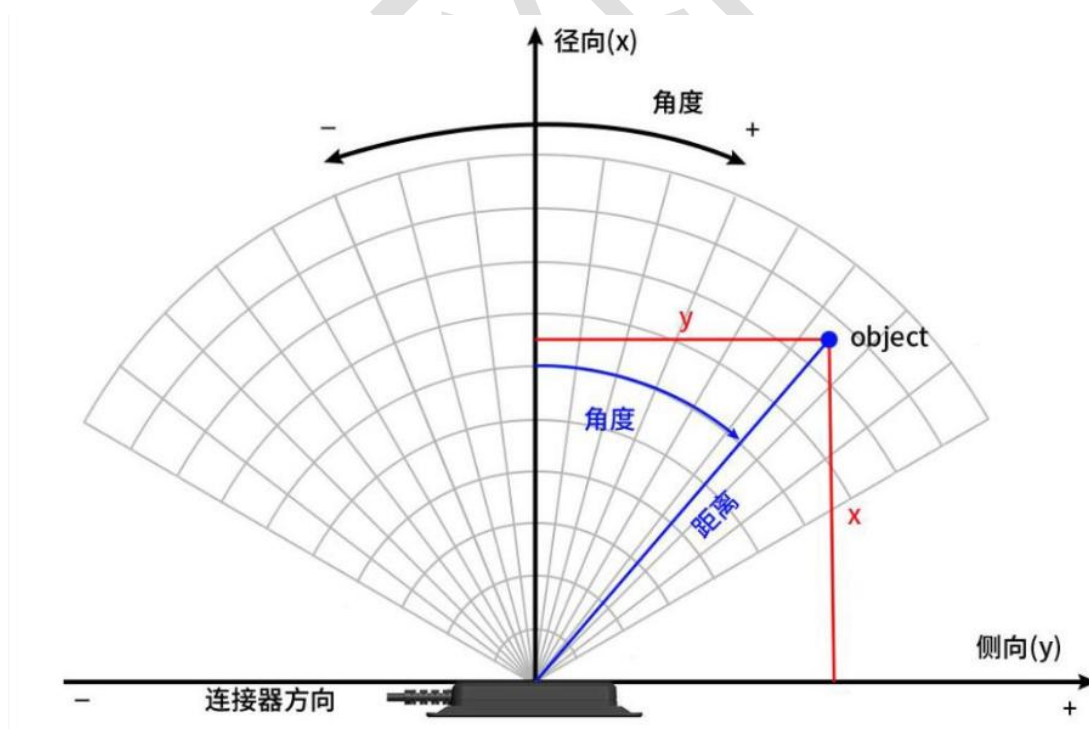


35



5) 使用角度尺，先校 0 然后调节成 90 度，画一条垂直 A 点的线，此线指向的就是毫米波雷达检测物体的 0 度方向（插接件指向正角度，反方向指向负角度）；





6) 使用激光测距仪配合摆放角反射器, 先将激光测距仪的测量起点调节至头部, 然后将角反射器放置距离毫米波雷达 3 米远的距离 (此距离可自定义, 但需要为 0.2 的倍数, 原因与毫米波雷达的分辨率有关), 需要考虑保险杠的宽度, 所以

激光测距仪测量 2.98 米左右即可。





7) 打开装调标定教学软件，进入前毫米波雷达界面；



8) 点击 CAN 报文反馈，查看是否有检测数据，用于验证毫米波雷达是否工作；



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

调试 标定
清除 Can报文反馈

序号	时间	can通道	传输方向	ID号	帧类型	帧格式	长度	数据
0	14:43:22:99	can1	接收	0x60B	数据帧	标准帧	8	1F 4E E3 FB 80 20 01 90
1	14:43:23:00	can1	接收	0x60B	数据帧	标准帧	8	12 4F 64 06 80 20 05 92
2	14:43:23:00	can1	接收	0x60B	数据帧	标准帧	8	17 4F 9B EF 7F DF C3 8E
3	14:43:23:00	can1	接收	0x60B	数据帧	标准帧	8	18 4F EB FE 80 20 01 A6
4	14:43:23:00	can1	接收	0x60B	数据帧	标准帧	8	04 52 A4 04 80 20 01 9A
5	14:43:23:00	can1	接收	0x60B	数据帧	标准帧	8	26 54 54 03 80 20 01 6C
6	14:43:23:00	can1	接收	0x60B	数据帧	标准帧	8	03 54 B3 F0 80 20 01 74
7	14:43:23:00	can1	接收	0x60B	数据帧	标准帧	8	08 55 C4 04 80 1F E1 72
8	14:43:23:00	can1	接收	0x60B	数据帧	标准帧	8	19 56 CB F8 80 1F E1 68
9	14:43:23:00	can1	接收	0x60B	数据帧	标准帧	8	10 57 B3 FF 80 20 01 68

深圳风向往标教育股份有限公司 版权所有

9) 点击数据可视化，查看是否有检测到摆放的角反射器；



10) 点击标定，填写摆放角反射器的距离和角度后点击标定，标定成功后，会反馈角反射器实际的距离和角度；



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

调试 标定

距离(m)	角度(°)	标定	结果	实际距离(m)	实际角度(°)
3.0	0	标定	• 标定成功	等待标定...	等待标定...

深圳风向标教育科技股份有限公司 版权所有



2.6 角毫米波雷达调试

1) 找到角毫米波雷达下的“参数配置”；





2)在考核模式下根据协议自行计算所需配置的CAN报文并填写报文到数据部分，在非考核模式（教学模式）下选择配置选项，软件会自动计算所需CAN报文；

具体ID 计算公式：每个雷达消息ID = 雷达ID * 0x10 + 基础消息ID。由于CAN总线可以挂载多个设备，每个设备有自己的ID。如上表中雷达ID默认为0，基础消息ID为0x200、0x400、0x60A、0x70B、0x70C。若雷达ID配置成1，则其Message ID分别为0x210，0x410，0x61A，0x71B，0x71C，以此类推。

表 6 雷达配置消息结构描述

参数	起始位置	长度(bit)	定义
DataType	0	7	1:雷达 ID 2:雷达版本 3:启动、停止目标信息输出 4:距离过滤 5:模式 6:雷达安装方向 7:目标输出选择 7e:内部测试使用 7f:保存参数
R/W	7	1	0:读取参数;1:写入参数
Parameter	8	56	根据 DataType 定义

启动、停止雷达目标信息输出格式如下表所示：

表 9 启动/停止目标信息数据输出格式

参数	起始位置	长度	值	定义
DataType	0	7	3	启动、停止目标信息输出
R/W	7	1	-	0:读取参数;1:写入参数

Parameter	8	1		0:停止输出;1:启动输出
Reserved	9	55	-	-

CAR28F 可以输出经过处理的目标数据（例如输出设定距离范围内的目标数据），也可直接输出原始的目标数据（检测范围内的所有目标数据），目前默认输出原始目标数据。目标输出选择格式如下表所示：

表 12 目标输出选择格式

参数	起始位置	长度	值	定义
DataType	0	7	7	目标输出选择
R/W	7	1	-	0:读取参数；1:写入参数
Parameter	8	1		0:处理后的目标数据 1:原始目标数据
Reserved	9	55	-	-





3) 找到角毫米波雷达下的“CAN 报文反馈”，如果在 CAN 报文中找到 ID 为 0x71C（左前）、0x72C（右前）、0x73C（左后）和 0x74C（右后）并且报文在不断刷新，则说明角毫米波雷达工作正常，否则异常；



The screenshot displays the "CAN 报文反馈" (CAN Message Feedback) table in the software. The table lists various CAN messages received from the radar, with specific IDs highlighted in red boxes.

序号	时间	can通道	传输方向	ID号	帧类型	帧格式	长度	数据
0	14:43:28:03	can1	接收	0x74C	数据帧	标准帧	8	01 64 00 A8 60 02 BC BF
1	14:43:28:03	can1	接收	0x74C	数据帧	标准帧	8	02 64 02 E4 56 02 C3 A1
2	14:43:28:03	can1	接收	0x72C	数据帧	标准帧	8	01 64 01 E7 5D 02 BC B7
3	14:43:28:03	can1	接收	0x72C	数据帧	标准帧	8	02 64 04 77 65 02 BC B1
4	14:43:28:03	can1	接收	0x71C	数据帧	标准帧	8	02 64 01 FA 5F 02 BC AD
5	14:43:28:03	can1	接收	0x73C	数据帧	标准帧	8	01 64 00 95 5F 02 BB C1
6	14:43:28:03	can1	接收	0x71C	数据帧	标准帧	8	03 64 02 EE 5C 03 29 AD
7	14:43:28:03	can1	接收	0x71C	数据帧	标准帧	8	01 64 04 52 51 02 BC B3
8	14:43:28:03	can1	接收	0x74C	数据帧	标准帧	8	01 64 00 A8 60 02 BC BF
9	14:43:28:04	can1	接收	0x74C	数据帧	标准帧	8	02 64 02 E1 56 02 C2 A2
10	14:43:29:02	can1	接收	0x73C	数据帧	标准帧	8	01 64 00 9F 5D 02 B3 C1
11	14:43:29:02	can1	接收	0x71C	数据帧	标准帧	8	02 64 01 FA 5F 02 BC AC
12	14:43:29:02	can1	接收	0x71C	数据帧	标准帧	8	03 64 02 F5 5C 03 29 AC
13	14:43:29:03	can1	接收	0x71C	数据帧	标准帧	8	01 64 04 52 52 02 BC B3
14	14:43:29:03	can1	接收	0x74C	数据帧	标准帧	8	01 64 00 B9 5F 02 BC BF
15	14:43:29:03	can1	接收	0x74C	数据帧	标准帧	8	02 64 02 F5 56 02 C4 A4
16	14:43:29:03	can1	接收	0x72C	数据帧	标准帧	8	01 64 01 E7 5D 02 BC B6
17	14:43:29:03	can1	接收	0x72C	数据帧	标准帧	8	02 64 04 77 65 02 BC B0
18	14:43:29:03	can1	接收	0x71C	数据帧	标准帧	8	02 64 01 FA 5F 02 BC AE
19	14:43:29:03	can1	接收	0x71C	数据帧	标准帧	8	03 64 02 F5 5C 03 29 AE
20	14:43:30:03	can1	接收	0x74C	数据帧	标准帧	8	01 64 00 BB 5F 02 BC C0
21	14:43:30:03	can1	接收	0x74C	数据帧	标准帧	8	02 64 03 21 4C 02 D3 A3
22	14:43:30:03	can1	接收	0x72C	数据帧	标准帧	8	01 64 01 E7 5D 02 BC B4
23	14:43:30:04	can1	接收	0x72C	数据帧	标准帧	8	02 64 04 77 65 02 BC AE
24	14:43:30:04	can1	接收	0x73C	数据帧	标准帧	8	01 64 00 95 5E 02 BB BF

4) 找到角毫米波雷达下的“数据解析”；




玉兔车调试标定软件

帧ID(HEX) 0x71C

数据 01 A2 BF C3 D4 F5 A3 D1

	7	6	5	4	3	2	1	0
0	7	6	5	4	3	2	1	0
1	15	14	13	12	11	10	9	8
2	23	22	21	20	19	18	17	16
3	15	14	13	12	11	10	9	8
4	39	38	37	36	35	34	33	32
5	45	44	43	42	41	40	39	38
6	63	62	61	60	59	58	57	56
7	63	62	61	60	59	58	57	56

2进制 00000000 00000000 00000000

总线值 0 0 0

解析后的数据

距离(m) 0

角度(°) 0

速度(m/s) 0

深圳风向往标教育股份有限公司 版权所有

5) 选择需要解析角毫米波的帧 ID，点击播放按钮，可以抓取此刻 CAN 报文，当抓到报文后点击暂停按钮再进行 CAN 报文解析，在**考核模式**下需要自行计算报文中含义（在 6）有说明怎么计算），在非考核模式（**教学模式**）下软件会自动计算该报文的含义；



6) 将数据部分的十六进制转成二进制填写到字段表格;

0x02 = 0000 0010

0x64 = 0110 0100

0x04 = 0000 0100

0x2C = 0010 1100

0x70 = 0111 0000

0x02 = 0000 0010

0xBC = 1011 1100

0xAF = 1010 1111

	7	6	5	4	3	2	1	0
0	7	6	5	4	3	2	1	0
1	15	14	13	12	11	10	9	8
2	距离	23	22	21	20	19	18	17
3	距离	31	30	29	28	27	26	25
4	角度	39	38	37	36	35	34	33
5				45	44	43	42	41
6		55	54	53	52	51	50	49
7	散射面积	63	62	61	60	59	58	57

根据协议计算距离、角度和速度;

参数	起始位置	长度	计算方法	取值范围
Cluster_Index	0	8		0~127
Cluster_RCSValue	8	8	Val*0.5-50	-50~30
Cluster_Range	16	16	Val*0.01	0~655
Cluster_Azimuth	32	8	Val-90	-90~90
Cluster_Vrel	48	11	Val*0.05-35	-35~35
Clusterl_RollCount	46	2		0~3

根据协议和字段表格提取距离、角度和速度的二进制并转成十进制；

注！存储方式为摩托罗拉（大端存储）

距离：0000 0100 0010 1100 = 1068

角度：0111 0000 = 112

速度：010 1011 1100 = 700

根据协议的计算方法进行计算求得最终值；

距离：1068 * 0.01 = 10.68

角度：112 - 90 = 22

速度：700 * 0.05 - 35 = 0

7) 找到角毫米波雷达下的“数据可视化”，可以对检测物体的横向范围、纵向范围以进行限制，还可以自由勾选所需显示数据；



2.7 激光雷达安装

2.7.1 场地要求

要求地面尽量水平，无沟槽无凸起物。

2.7.2 准备工作

1) 准备内六角扳手套装、卷尺、铅垂线、水平仪和防护手套并检查是否可以正常使用；



2) 点击电源开关（绿灯熄灭），关闭车辆电源；



2.7.3 安装过程

1) 使用内六角扳手，安装激光雷达并紧固螺栓；



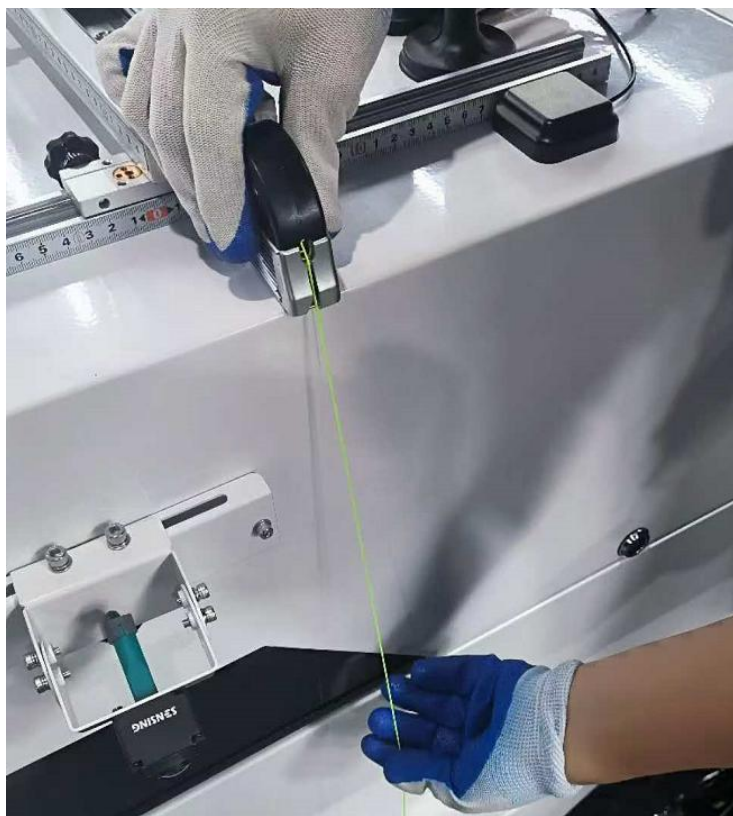
2) 连接激光雷达线束；



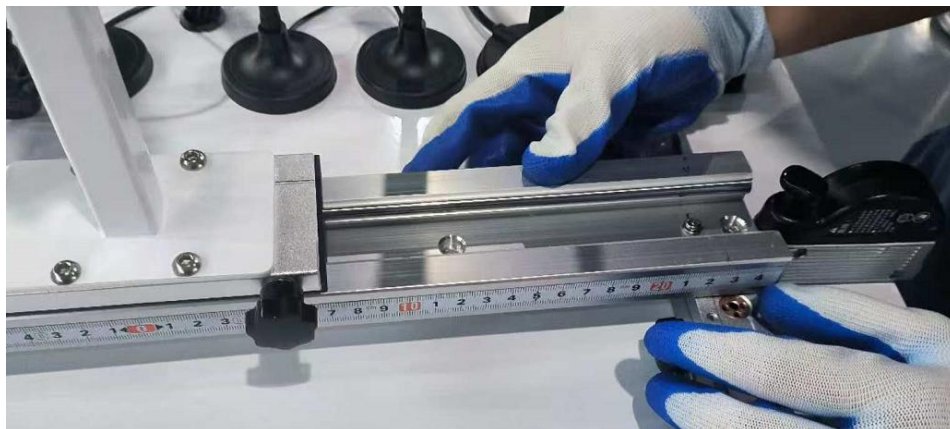
3) 将激光雷达中心移动 0 点处 (Y 偏移量为 0) ;



4) 摆放铅垂线, 使用卷尺将铅垂线调整到离后轴中心 0.33m (X 偏移量为 0.33);



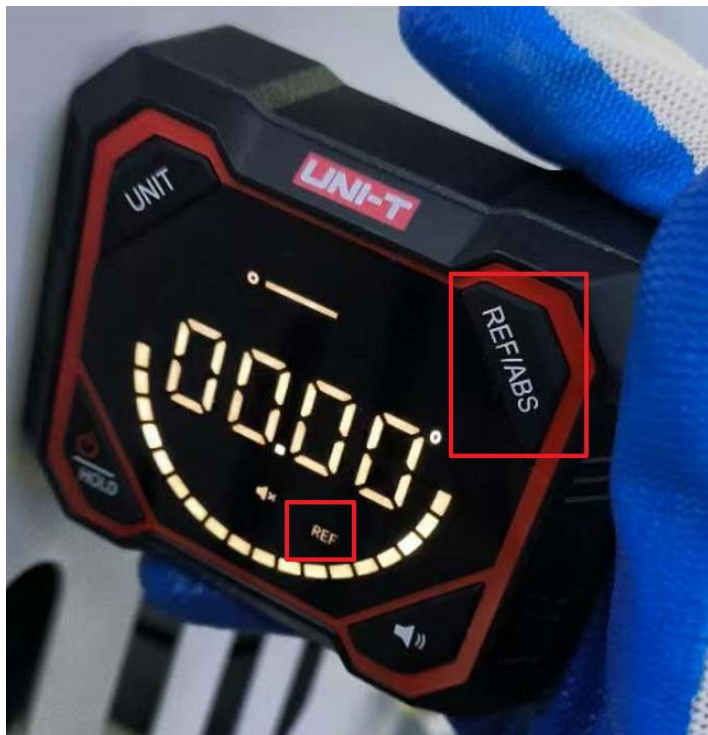
5) 调整激光雷达对齐铅垂线;



6) 长按水平仪电源按钮开机，长按水平仪 UNIT 按钮调节水平仪测量单位为度；



7) 按下 REF/ABS 按钮将水平仪设置为相对模式（用于测量相对水平），此时需要将水平仪放置在基准面（这里以激光雷达支架安装面为基准面）；



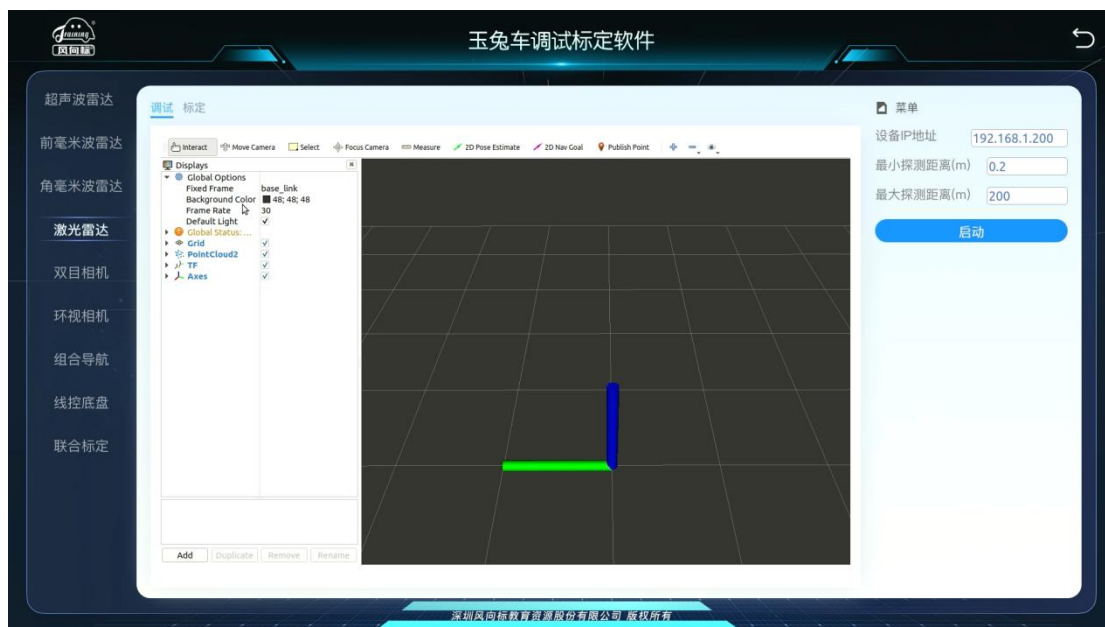
8) 将水平仪放置在激光雷达头部平面上，测量水平角，此角度应在 $0 \pm 0.3^\circ$ 的范围；



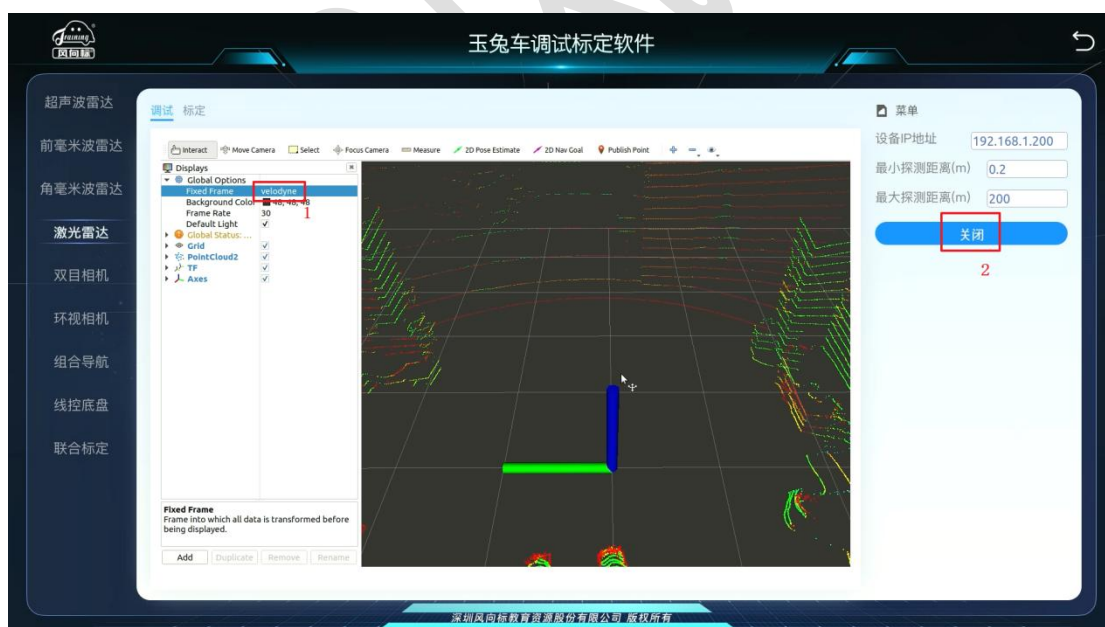


2.8 激光雷达调试

1) 找到激光雷达界面；



2) 点击启动，设置参考坐标系为 velodyne，如果有点云出现则说明激光雷达工作正常，否则异常；





2.9 激光雷达标定

2.9.1 场地要求

要求地面尽量水平，无沟槽无凸起物，标定区域为 10m*5m 以上的空旷区域。

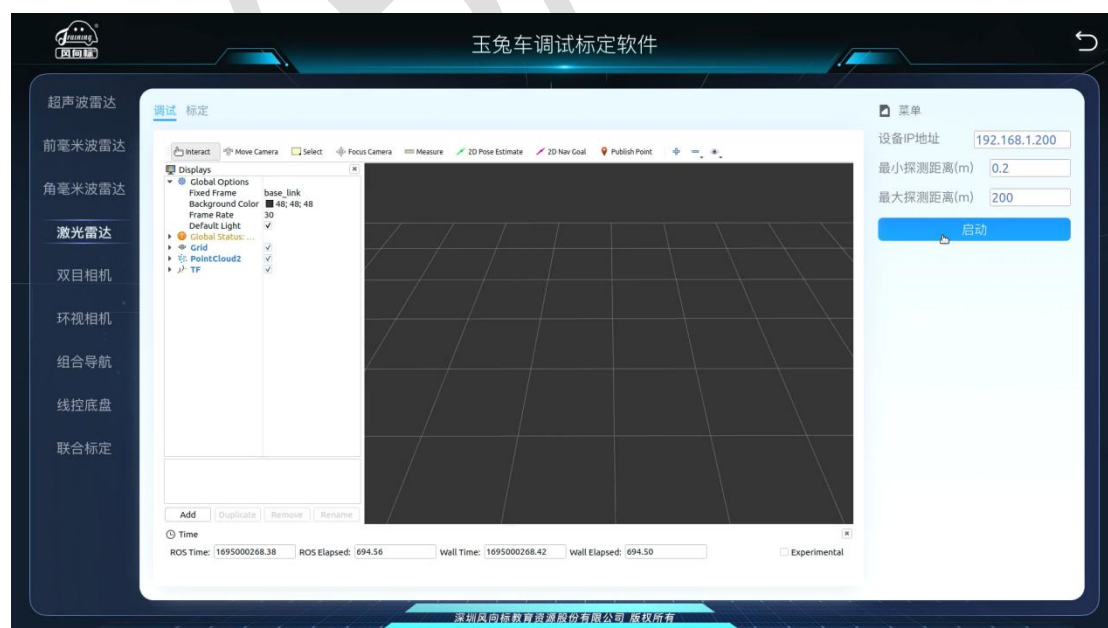
2.9.2 准备工作

1) 准备卷尺并检查是否可以正常使用；

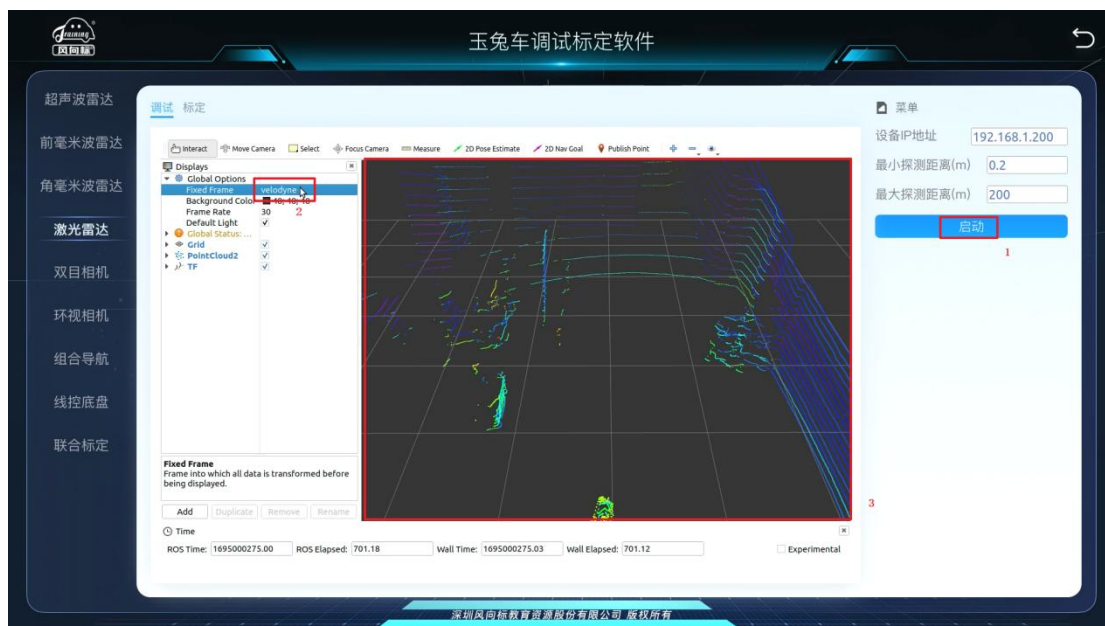


2.9.3 标定过程

1) 打开装调标定教学软件，进入激光雷达界面；

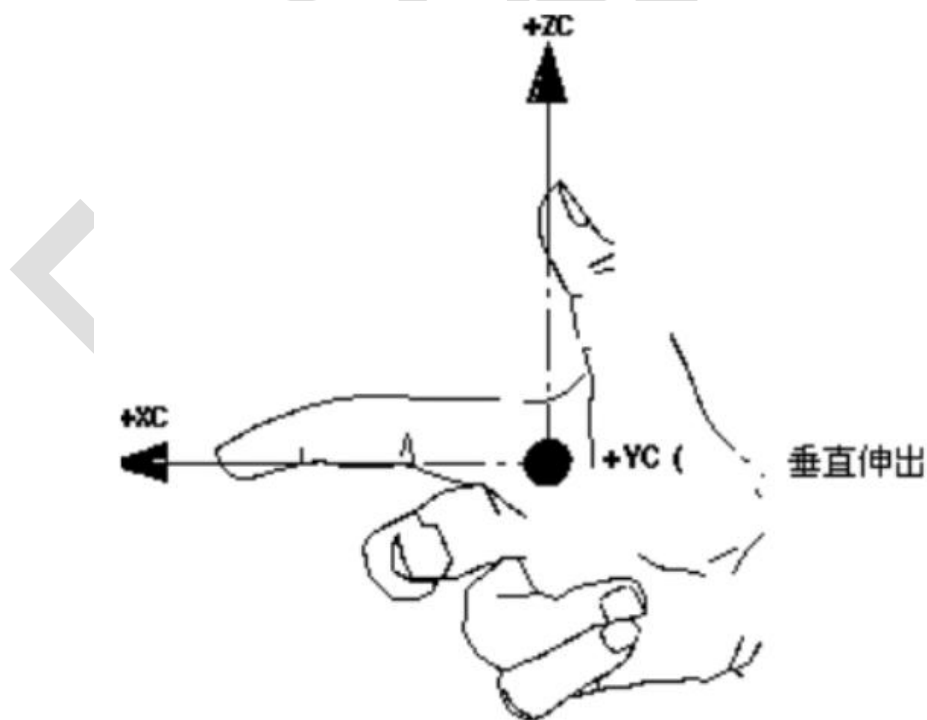


2) 点击启动，设置参考坐标系为 velodyne，此时会有点云出现；

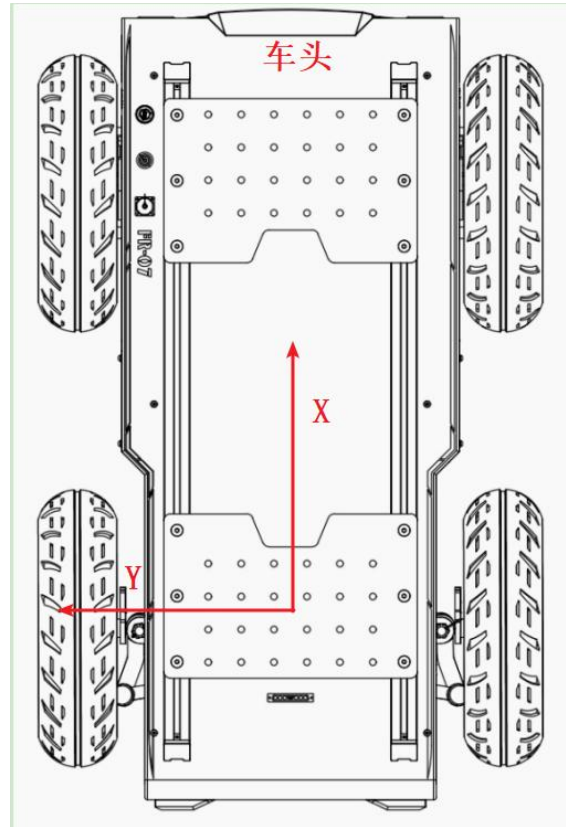


3) 根据右手定则测量 X 偏移;

坐标系都采用右手定则，即伸出右手，摆出以下手势，此时食指指向的是 X 正方向（X 偏移），中指指向的是 Y 正方向（Y 偏移），大拇指指向的是 Z 正方向（Z 偏移），绕着大拇指旋转是偏航角（yaw），绕着中指旋转是俯仰角（pitch），绕着食指旋转是翻滚角（roll），顺时针旋转是负角度，逆时针旋转是正角度。



车辆后轴中心坐标系（base_link）是以车头为 X 正方向，车左侧为 Y 正方向，车顶为 Z 正向（此方向由开发者来定，不过这个方向为行业通用）。



激光雷达坐标系（velodyne）是以连接线束的反方向为 X 正方向，左侧为 Y 正方向，顶部为 Z 正方向（此方向不是固定的，根据传感器的驱动代码而定）。



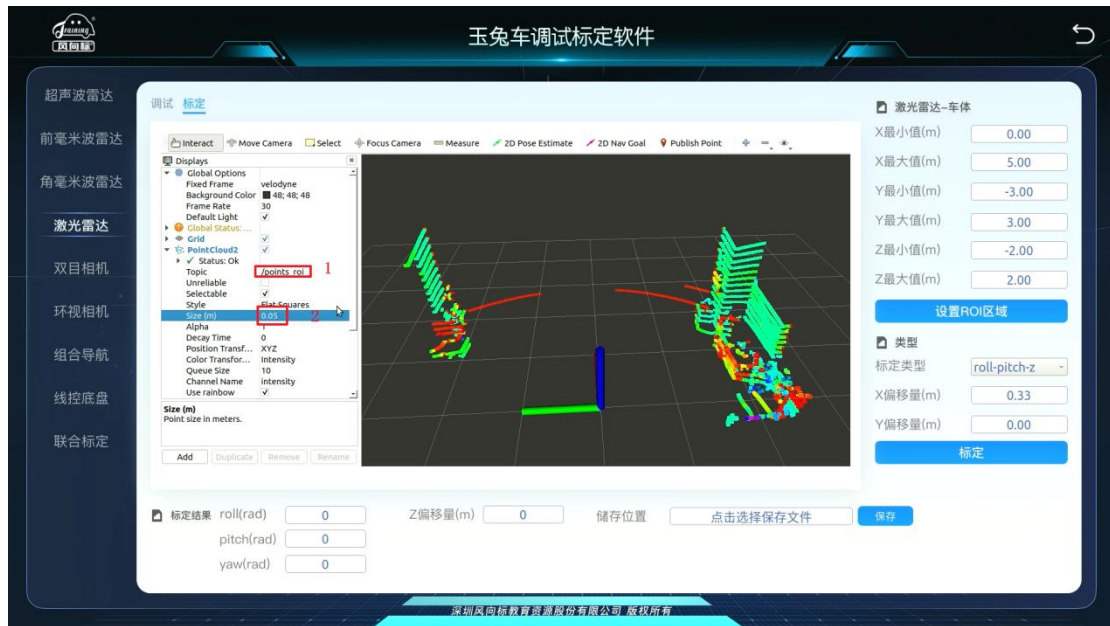


以车辆后轴中心坐标系（base_link）为原点，使用卷尺去测量激光雷达坐标系（velodyne）的 X 偏移。

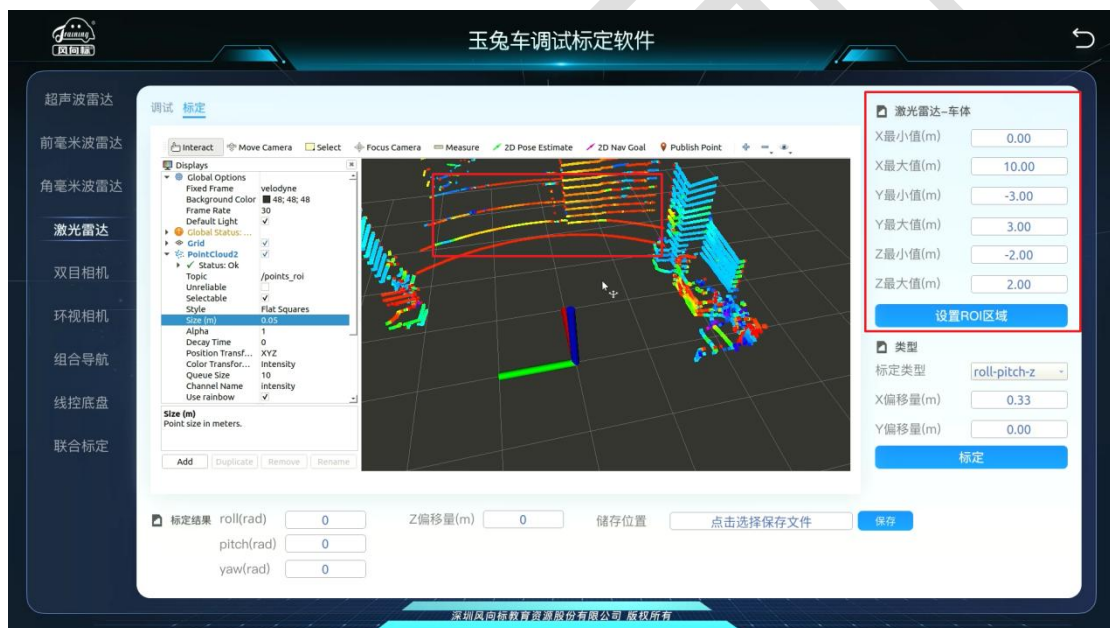


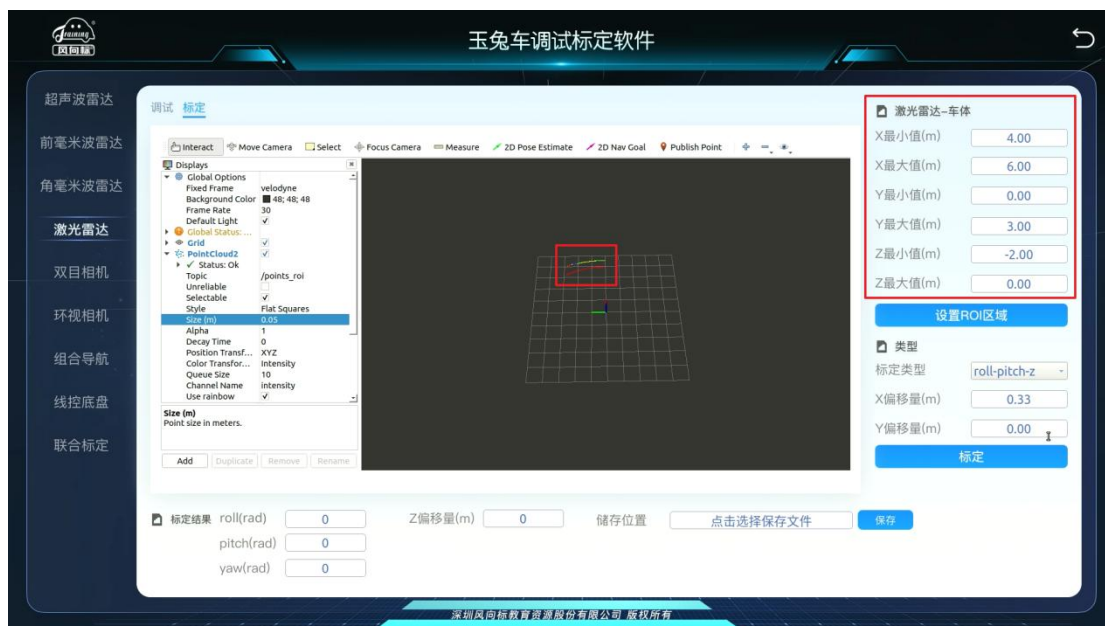
由上图可知 X 偏移为 0.33m 左右。

4) 切换显示点云的话题为/points_roi, 将显示点云的点尺寸设置为 0.05m 方便查看

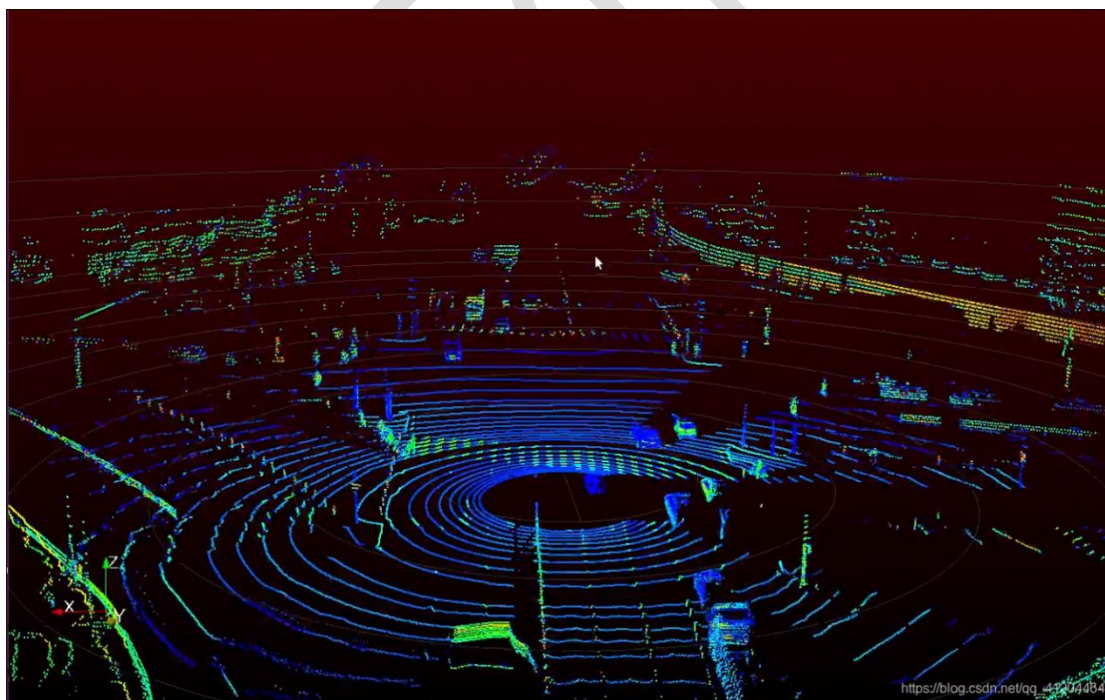


5) 设置 ROI 区域，最后使点云只剩下“地面点云”

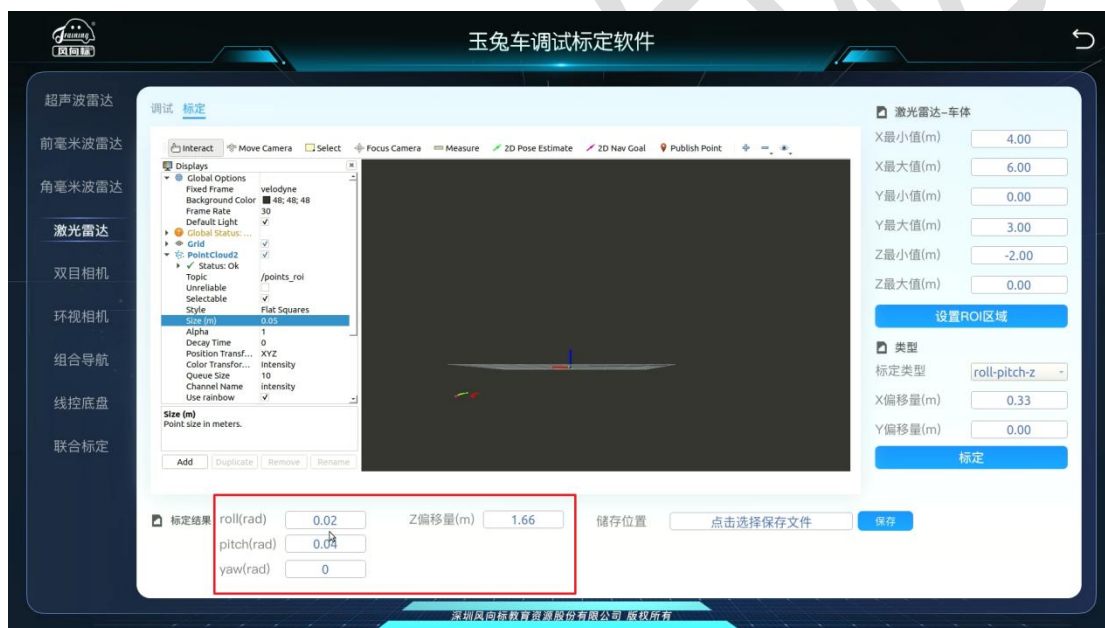
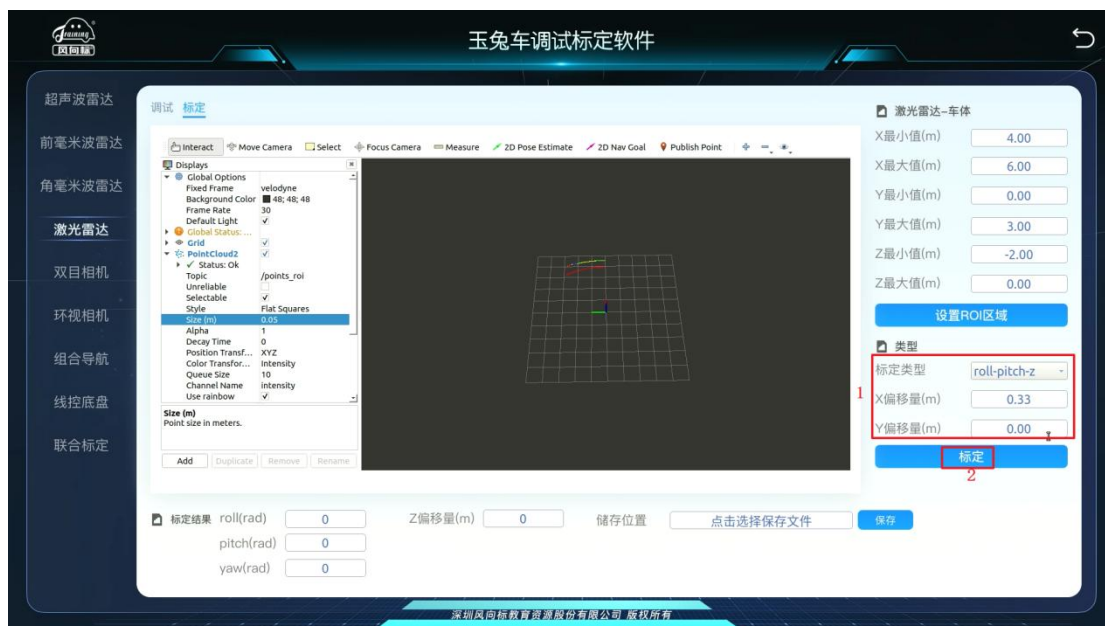




所谓的“地面点云”其实就是激光雷达扫描到地面的点云，这类点云的形状为环形，所以当你看到环形的点云，那说明这些扫描到了地面，下图是从网上找到的雷达点云图片，非常明显的看到很多“一圈一圈的点云”，没错这些就是地面点云。



6) 选择标定类型为 roll-pitch-z，填写 x 偏移量为 0.33，y 偏移量为 0 后点击标定

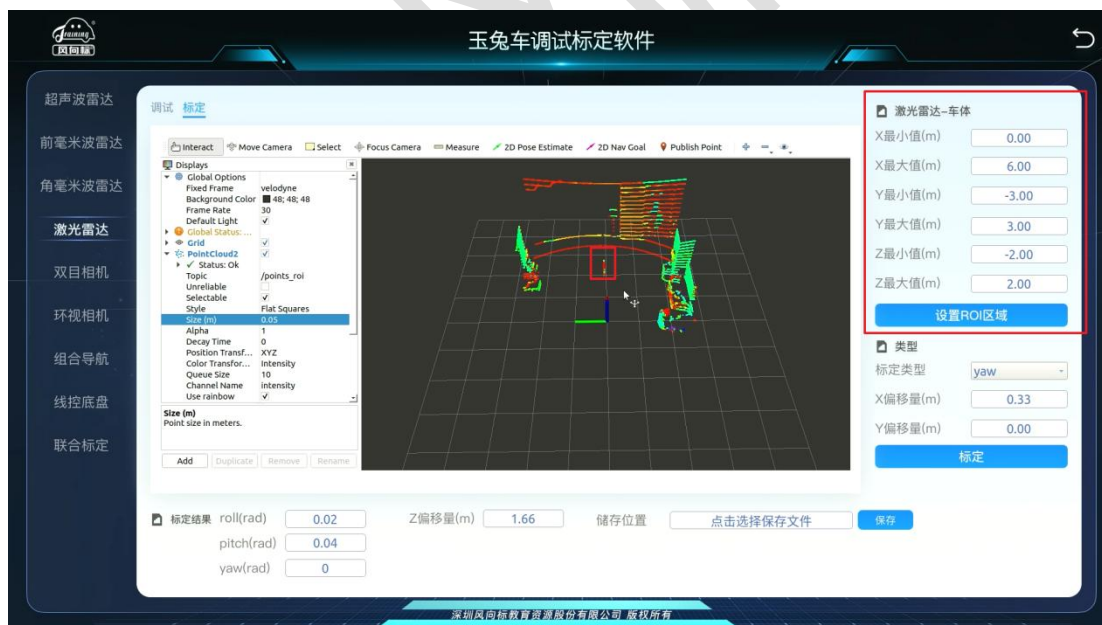


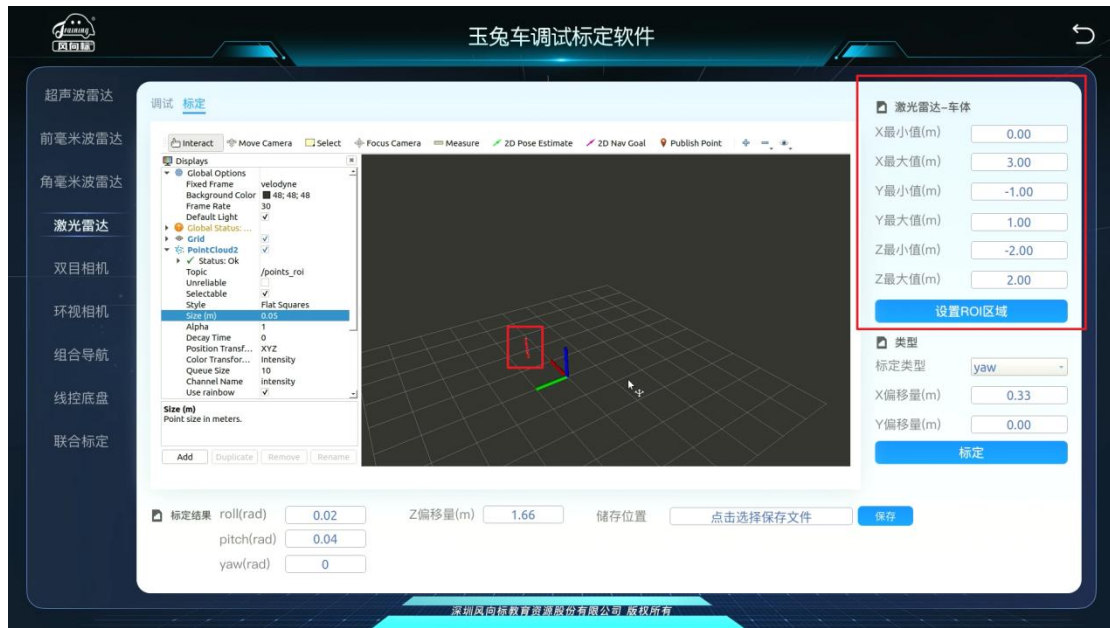
软件将自动算出当前激光雷达的 roll、pitch、Z 偏移量，其中 roll 和 pitch 的误差为正负 1 度，Z 偏移量会根据地面点云的数量而浮动，最高 10cm+ 误差，而且是激光雷达到地面的距离，不是到后轴中心，所以不能直接使用。

7) 使用激光测距仪将角反射器杆摆放到车头正前方 3m 处，摆放方法可以看 2.5.3 毫米波雷达标定

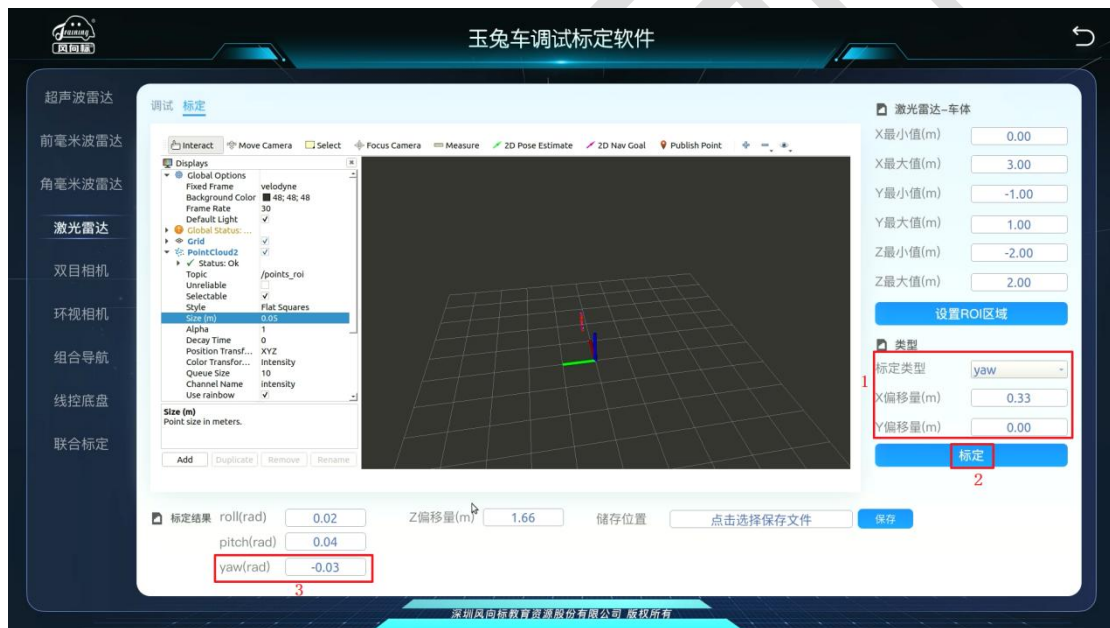


8) 设置 ROI 区域，最后使点云只剩下“角反射器杆”的点云

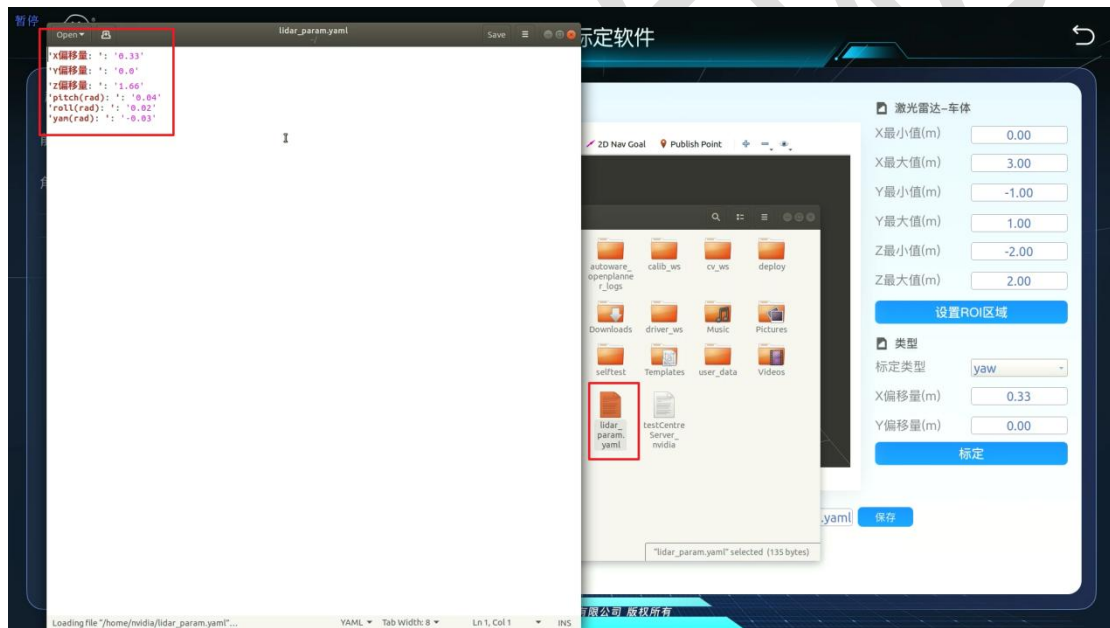
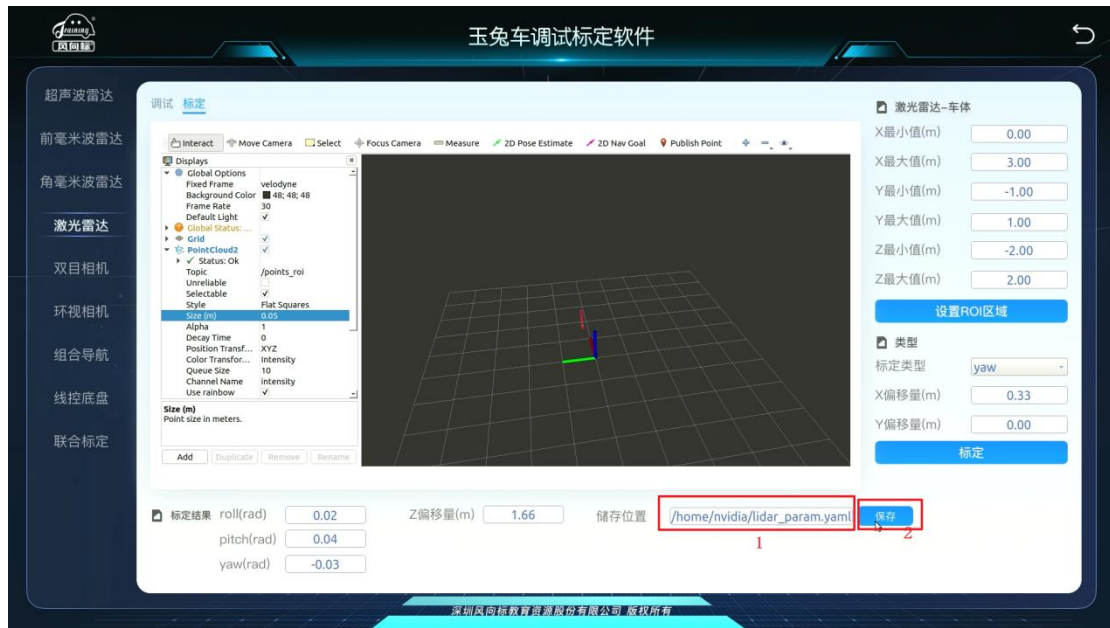




9) 设置标定类型为 yaw，填写 x 偏移量为 0.33，y 偏移量为 0 后点击标定



10) 保存标定结果



2.10 单目相机调试

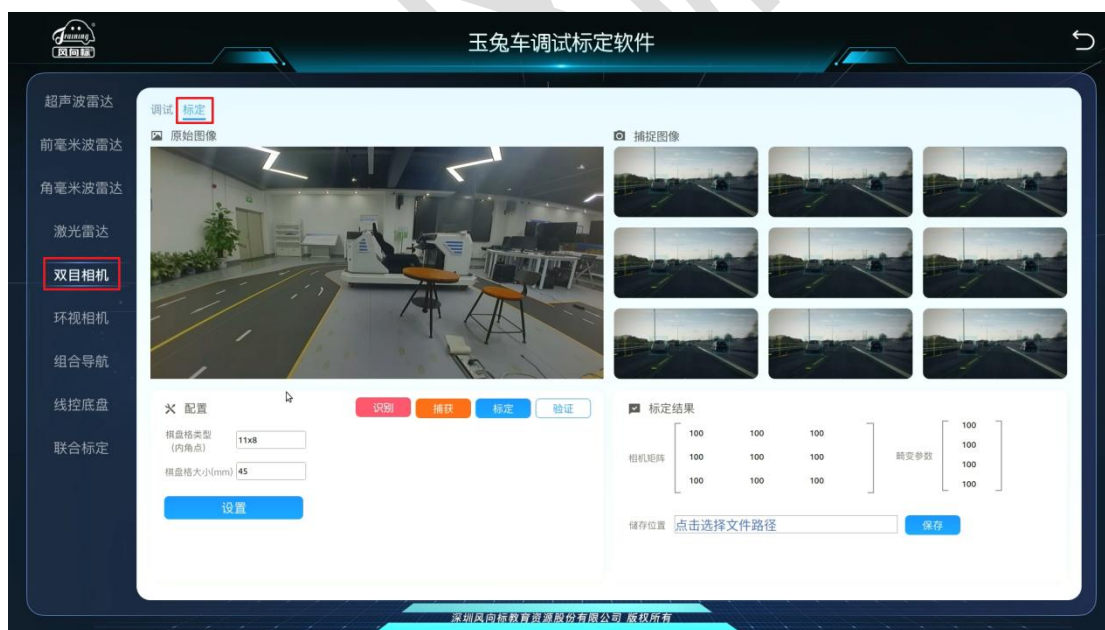
1) 找到单目相机，点击开启按钮；



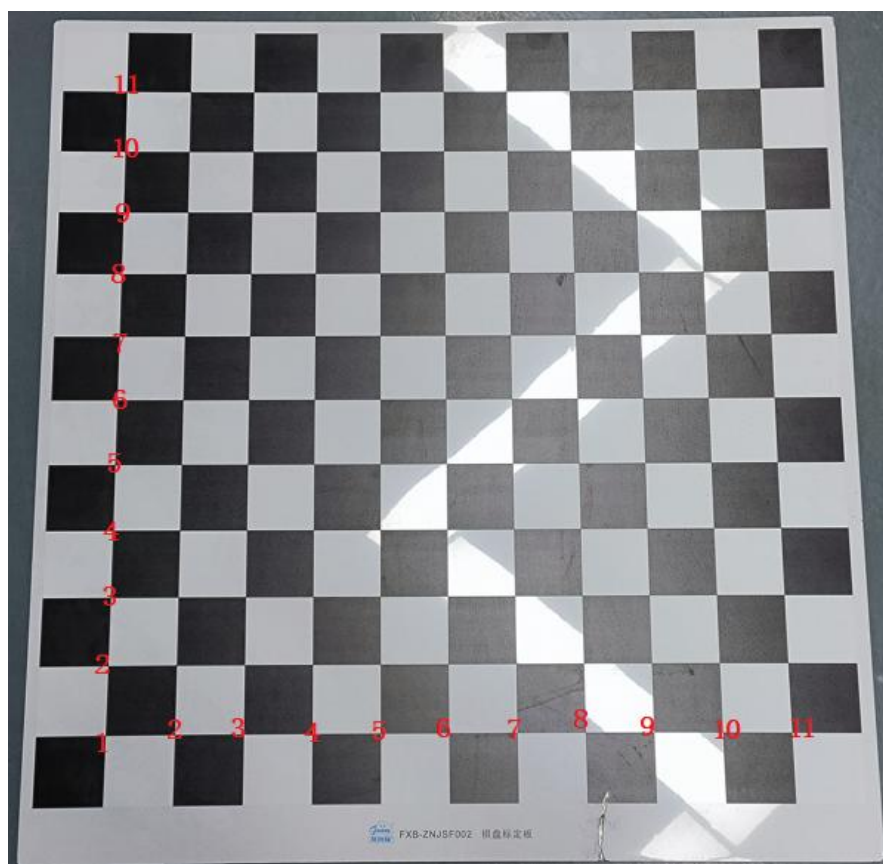
2) 如果有图像不断显示，则说明单目相机工作正常，否则异常；

2.11 单目相机标定

1) 找到单目相机下的标定；



2) 计算棋盘格内角点，使用卷尺测量单个格子宽度（单位 mm），填入软件后按下设置按钮，提示设置成功。





3) 将棋盘格标定板摆放相机前，确保棋盘格拍摄完整后，点击“识别”按钮，识别成功后画面中的棋盘格内角点将被彩色线连接，当识别成功后，点击“捕获”按钮保存本次识别数据（内角点像素位置）；

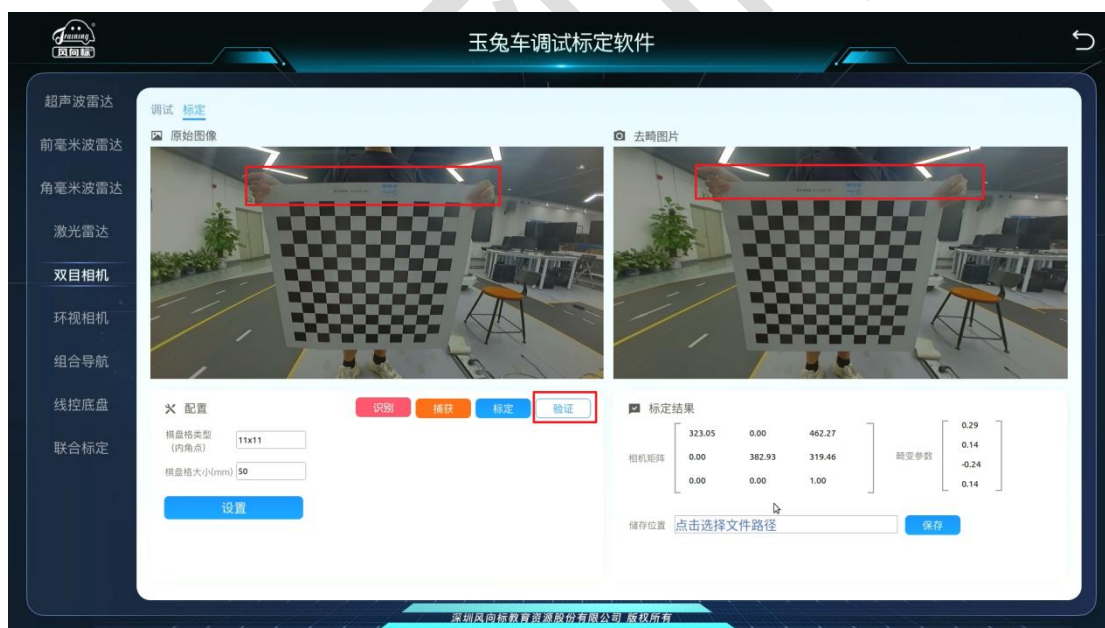




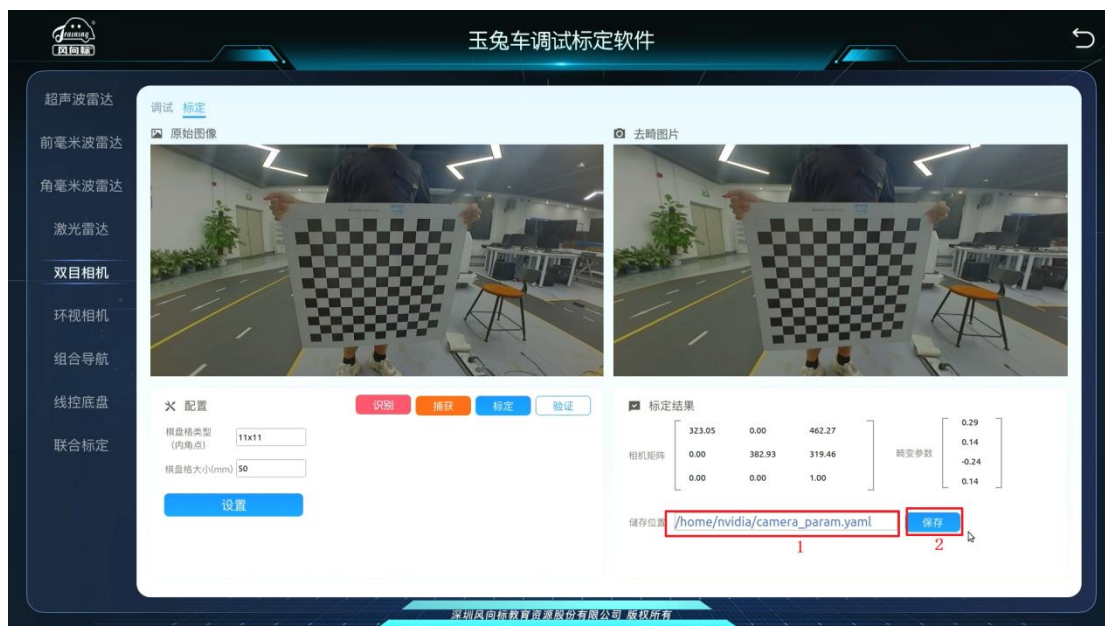
4) 当采集九次数据后，点击“标定”按钮，软件将会计算出相机内参矩阵和畸变系数；



5) 当标定成功后，即计算出相机矩阵和畸变系数后，可以点击“验证”按钮，观察标定前和标定后图像边缘的差异，发现标定后的图片被裁切了并且变形部分给校准了；



6) 保存标定参数；



2.12 环视相机调试

1) 找到环视相机，点击开启；



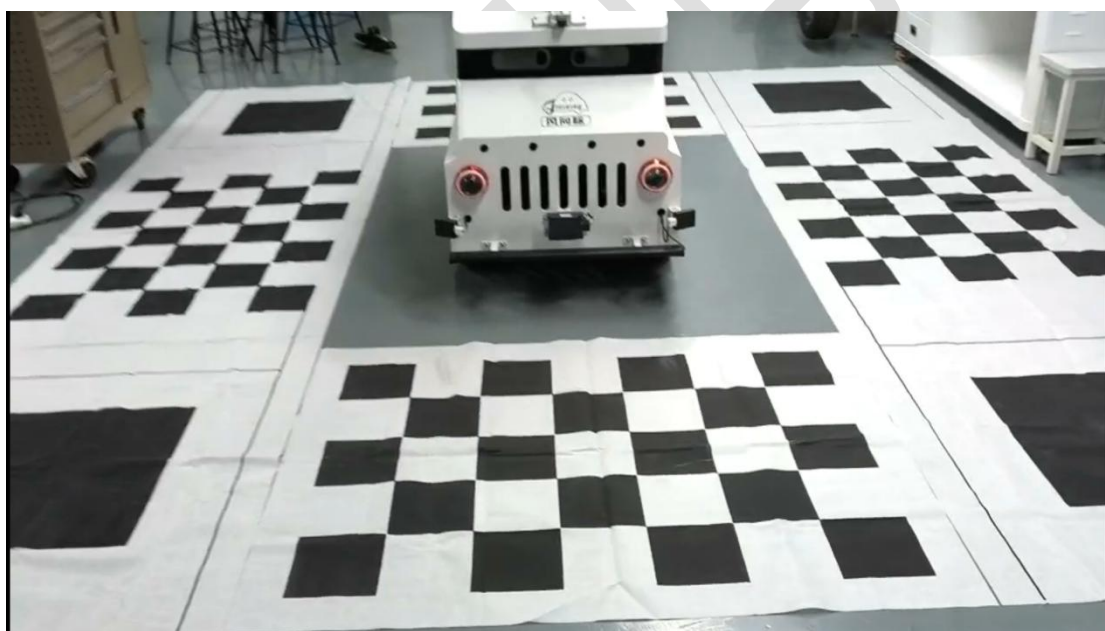
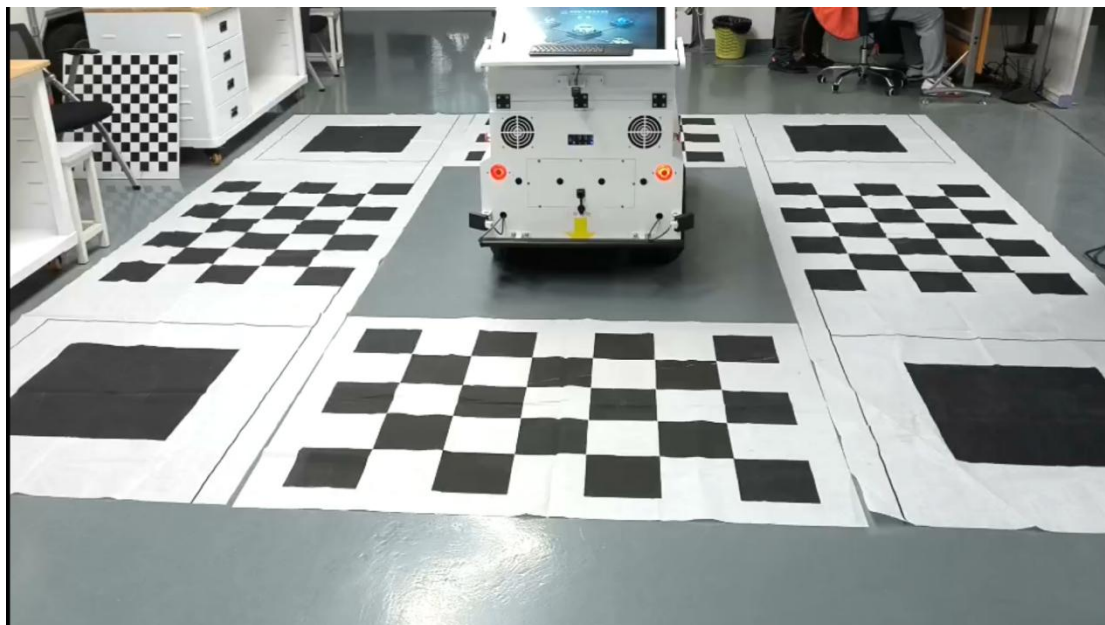
2) 如果有 4 个图像不断显示，则环视相机工作正常，否则异常；

2.13 环视相机标定

0) 从流程上环视标定的第一步是需要做相机内参标定（去畸变），本软件已经

默认做好了内参标定，所以以下环视画面都为去畸变的结果；

1) 摆放棋盘标定布，让车辆位于标定布中心；





2) 使用卷尺测量标定布尺寸，以下是测量到的数值：

大标定布

长：436 宽 118

小格子：20x20 大格子：60x60

大格子和小格子间距：58

左右边距：9 前后边距：30

小标定布

长：159 宽 119

小格子：20x20

边距 10

标定布离车辆左右距离：41

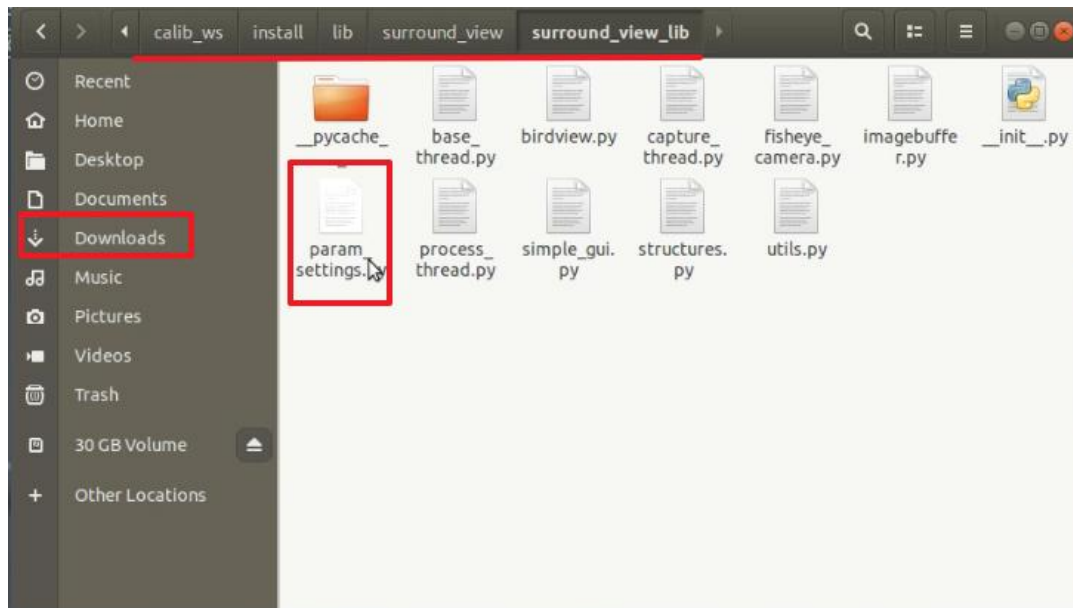
标定布离车辆前后距离：17

注意！测量单位都是 cm 厘米并且四舍五入，不能带有小数点

3) 找到

/home/nvidia/calib_ws/install/lib/surround_view/surround_view_lib/param_settings.

py 文件，根据实际摆放的不同距离进行填写。



```

Open  param_settings.py  Save
~/Downloads/calib_ws/install/lib/surround_view/surround_view_lib

#!/usr/bin/env python3
# coding=utf8

import cv2
import rospkg

camera_names = ["front", "back", "left", "right"]

# distance(cm)

# -----
# (shift_width, shift_height): how far away the birdview looks outside
# of the calibration pattern in horizontal and vertical directions
shift_w = 100
shift_h = 100

# size of the gap between the calibration pattern and the car
# in horizontal and vertical directions
inn_shift_w = 41
inn_shift_h = 17

# total width/height of the stitched image
total_w = 118 * 2 + 159 + 2 * shift_w
total_h = 436 + 2 * shift_h

# four corners of the rectangular region occupied by the car
# top-left (x_left, y_top), bottom-right (x_right, y_bottom)
xl = shift_w + 118 + inn_shift_w
xr = total_w - xl
yt = shift_h + 118 + inn_shift_h
yb = total_h - yt
# -----

# 转换成鸟瞰图后的图片尺寸
project_shapes = {
    "front": (total_w, total_h)
  
```

Python 3 Tab Width: 8 Ln 57, Col 59 INS


```
# 转换成鸟瞰图后的图片尺寸
project_shapes = {
    "front": (total_w, yt),
    "back": (total_w, yt),
    "left": (total_h, xl),
    "right": (total_h, xl)
}

# pixel locations of the four points to be chosen.
# you must click these pixels in the same order when running
# the get_projection_map.py script
project_keypoints = {
    "front": [(shift_w + 118 + 10, shift_h + 10),
              (shift_w + 118 + 10 + 140, shift_h + 10),
              (shift_w + 118 + 10, shift_h + 10 + 100),
              (shift_w + 118 + 10 + 140, shift_h + 10 + 100)],

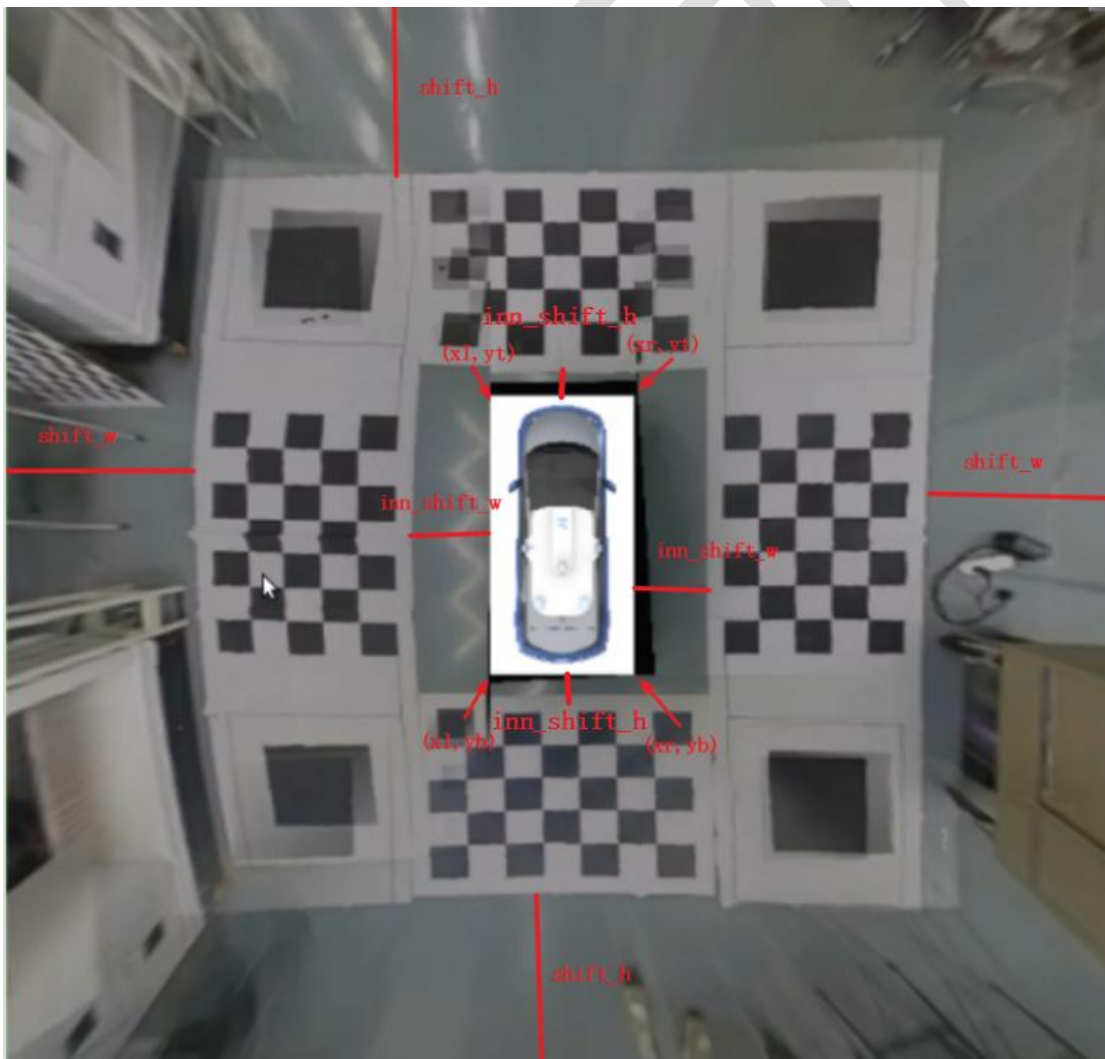
    "back": [(shift_w + 118 + 10, shift_h + 10),
              (shift_w + 118 + 10 + 140, shift_h + 10),
              (shift_w + 118 + 10, shift_h + 10 + 100),
              (shift_w + 118 + 10 + 140, shift_h + 10 + 100)],

    "left": [(shift_h + 30 + 60 + 58, shift_w + 9),
              (shift_h + 30 + 60 + 58 + 140, shift_w + 9),
              (shift_h + 30 + 60 + 58, shift_w + 9 + 100),
              (shift_h + 30 + 60 + 58 + 140, shift_w + 9 + 100)],

    "right": [(shift_h + 30 + 60 + 58, shift_w + 9),
              (shift_h + 30 + 60 + 58 + 140, shift_w + 9),
              (shift_h + 30 + 60 + 58, shift_w + 9 + 100),
              (shift_h + 30 + 60 + 58 + 140, shift_w + 9 + 100)]
}

car_image = cv2.imread(ros pkg.RosPack().get_path("surround_view") + "/imgs/car.png")
car_image = cv2.resize(car_image, (xr - xl, yb - yt))
```

Python 3 Tab Width: 8 Ln 57, Col 59 INS





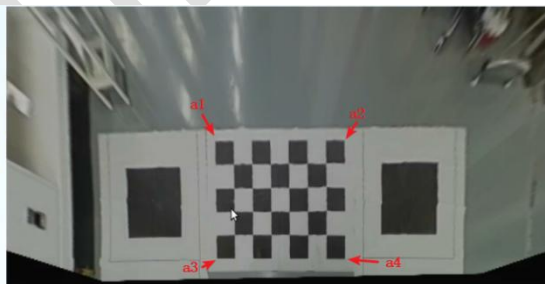
shift_w 和 **shift_h** 为标定布以外的距离（外距），此距离会影响输出整个融合图像的大小，这个值不需要测量，而是人为定义，此值越大需要的计算量就越大，所以这里 $\text{shift_w} = 100, \text{shift_h} = 100$;

inn_shift_w 和 **inn_shift_h** 为标定布到车辆的距离（内距），此距离需要手动测量，根据 2）中测量到的距离， $\text{inn_shift_w} = 41, \text{inn_shift_h} = 17$;

total_w 和 **total_h** 为融合图像输出的尺寸， $\text{total_w} = 2 \times \text{大标定布宽度} + \text{小标定布的宽度} + 2 \times \text{shift_w}$ ， $\text{total_h} = \text{大标定布长度} + 2 \times \text{shift_h}$ ，根据 2）中测量到的距离， $\text{total_w} = 2 \times 118 + 159 + 2 \times \text{shift_w}$ ， $\text{total_h} = 436 + 2 \times \text{shift_h}$;

xl 和 **xr** 为车辆左上、右上、左下和右下的 x 坐标值，此值决定在拼接图像中车辆的宽度，根据 2）中测量到的距离， $\text{xl} = \text{shift_w} + \text{大标定布宽度}(118) + \text{inn_shift_w}$ ， $\text{xr} = \text{total_w} - \text{xl}$;

yt 和 **yb** 为车辆左上、右上、左下和右下的 y 坐标值，此值决定在拼接图像中车辆的长度，根据 2）中测量到的距离， $\text{yt} = \text{shift_h} + \text{小标定布宽度}(119) + \text{inn_shift_h}$ ， $\text{yb} = \text{total_h} - \text{yt}$;



project_keypoints 填写的是 4 个环视相机进行透视变换所需特征点的坐标值，那么在软件进行点击获取特征点时，需要严格按照填写的顺序进行点击;

进行一次透视变换需要 4 组点对，每组点对由透视变换前坐标 **b1** 和透视变换后坐标 **a1** 组成，**b1** 可以通过软件点击获得像素坐标，**a1** 需要在脑海中想象这个图像转换成鸟瞰图（俯视角）的样子，然后根据实际测量值去计算，下面以环视相机和左环视相机为例，计算 **a1~a4** 的值;

前环视相机

$\text{a1_x} = \text{shift_w} + \text{大标定布宽度}(118) + \text{小标定布边距}(10)$

$\text{a1_y} = \text{shift_h} + \text{小标定布边距}(10)$

$\text{a2_x} = \text{shift_w} + \text{大标定布宽度}(118) + \text{小标定布边距}(10) + \text{小标定布中 7 个小格子的宽度之和}(7 \times 20 = 140)$



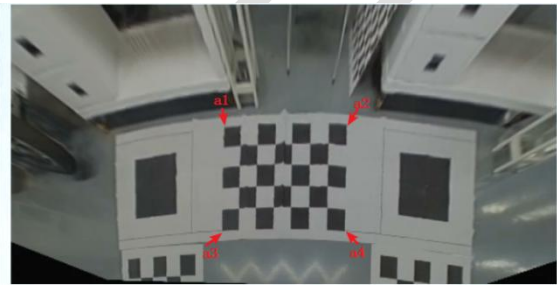
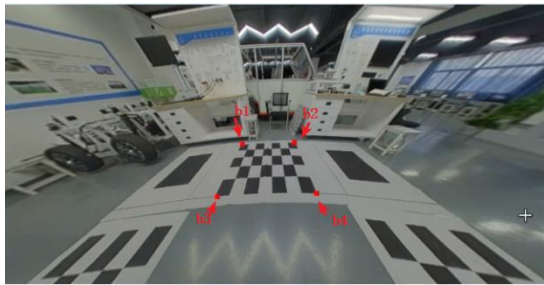
$a2_y = \text{shift_h} + \text{小标定布边距 (10)}$

$a3_x = \text{shift_w} + \text{大标定布宽度 (118)} + \text{小标定布边距 (10)}$

$a3_y = \text{shift_h} + \text{小标定布边距 (10)} + \text{小标定布中 5 个小格子的宽度之和 (5*20 = 100)}$

$a4_x = \text{shift_w} + \text{大标定布宽度 (118)} + \text{小标定布边距 (10)} + \text{小标定布中 7 个小格子的宽度之和 (7*20 = 140)}$

$a4_y = \text{shift_h} + \text{小标定布边距 (10)} + \text{小标定布中 5 个小格子的宽度之和 (5*20 = 100)}$



左环视相机

$a1_x = \text{shift_h} + \text{大标定布前后边距 (30)} + \text{大标定布大格子宽度 (60)} + \text{大格子和小格子间距 (58)}$

$a1_y = \text{shift_w} + \text{大标定布左右边距 (9)}$

$a2_x = \text{shift_h} + \text{大标定布前后边距 (30)} + \text{大标定布大格子宽度 (60)} + \text{大格子和小格子间距 (58)} + \text{小标定布中 7 个小格子的宽度之和 (7*20 = 140)}$

$a2_y = \text{shift_w} + \text{大标定布左右边距 (9)}$

$a3_x = \text{shift_h} + \text{大标定布前后边距 (30)} + \text{大标定布大格子宽度 (60)} + \text{大格子和小格子间距 (58)}$

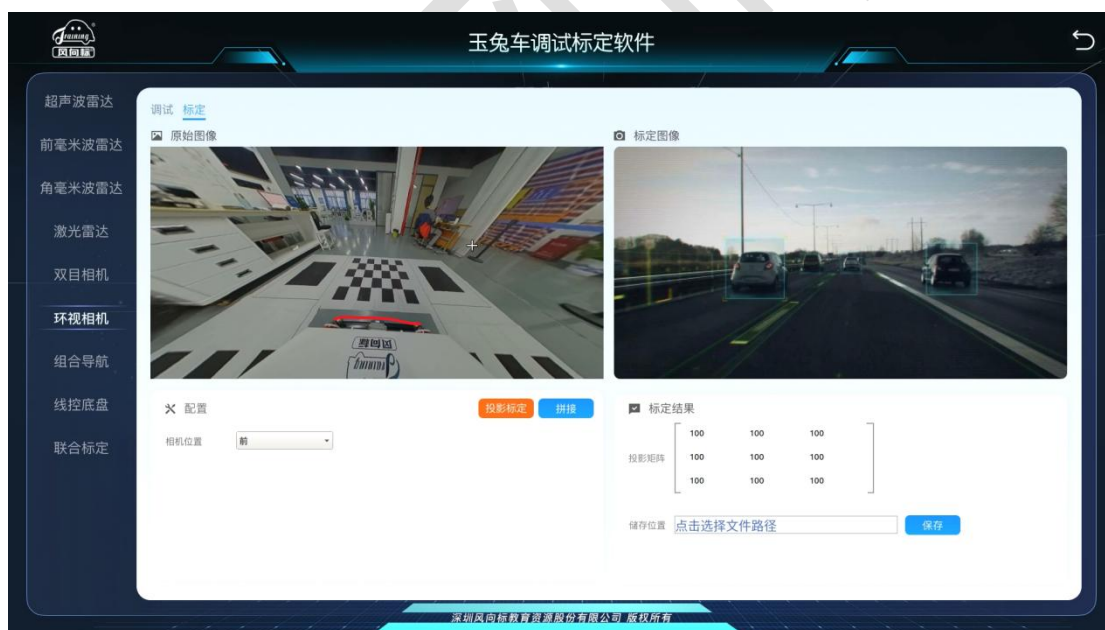
$a3_y = \text{shift_w} + \text{大标定布左右边距 (9)} + \text{小标定布中 5 个小格子的宽度之和 (5*20 = 100)}$

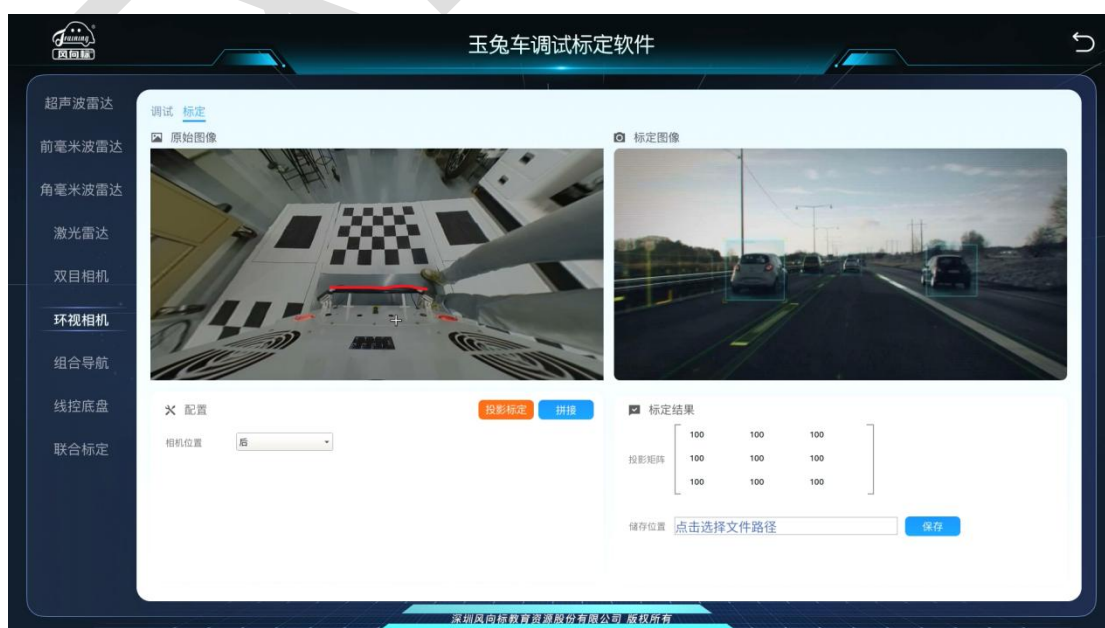
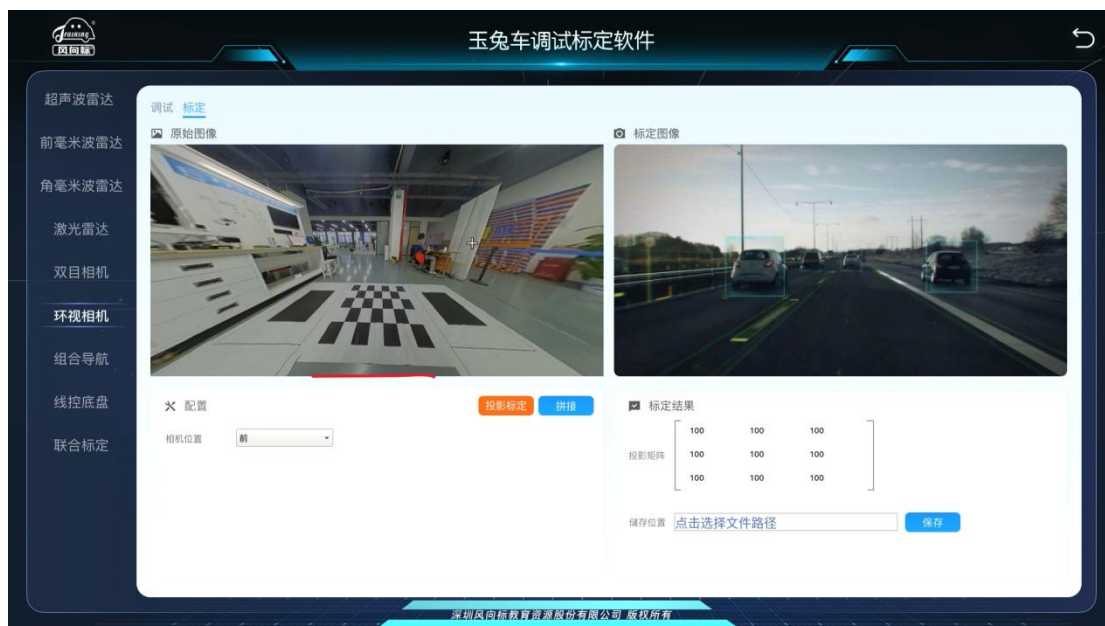
$a4_x = \text{shift_h} + \text{大标定布前后边距 (30)} + \text{大标定布大格子宽度 (60)} + \text{大格子和小格子间距 (58)} + \text{小标定布中 7 个小格子的宽度之和 (7*20 = 140)}$

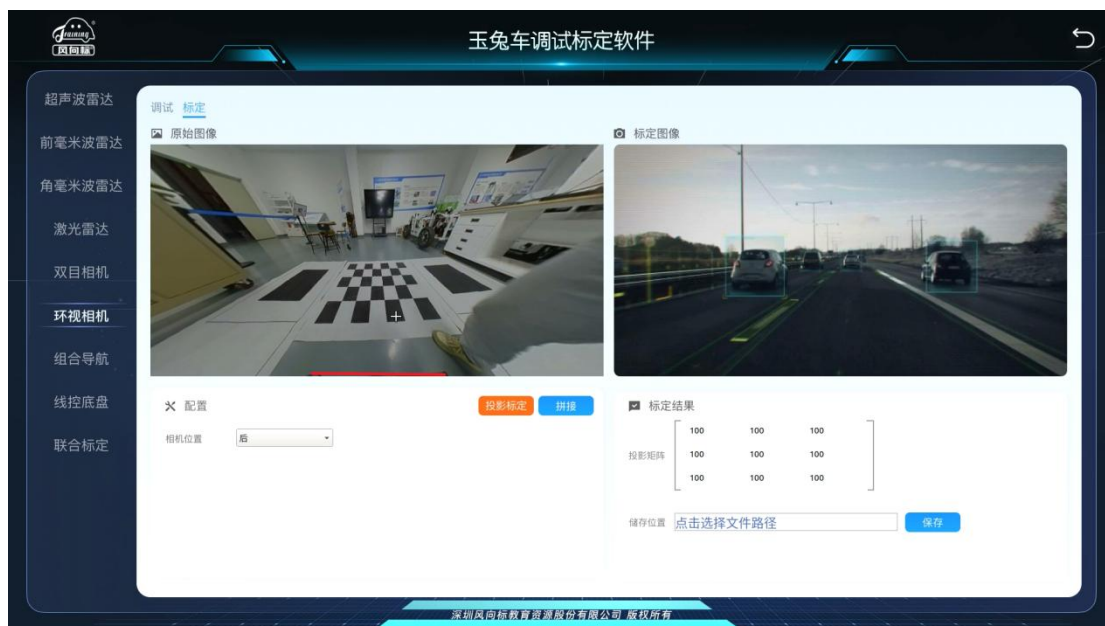
$a4_y = \text{shift_w} + \text{大标定布左右边距 (9)} + \text{小标定布中 5 个小格子的宽度之和 (5*20 = 100)}$

4) 当填写完配置文件后, 需要手动调整 4 个环视相机视角, 需要把相机视野的最低点调整为车身边缘, 以下有每个相机的例子;

•

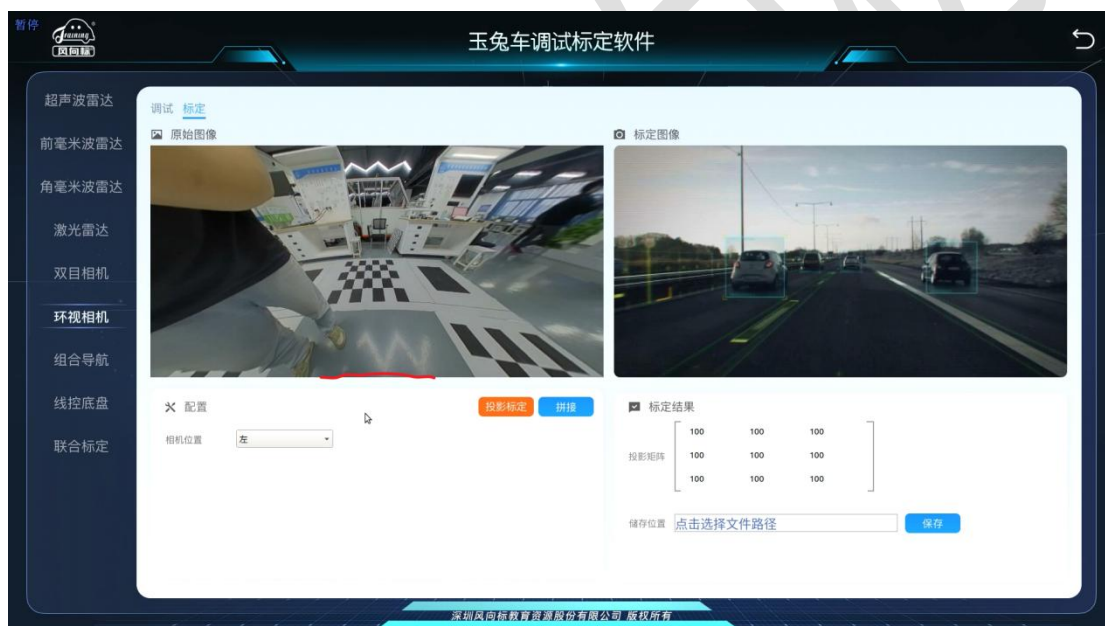


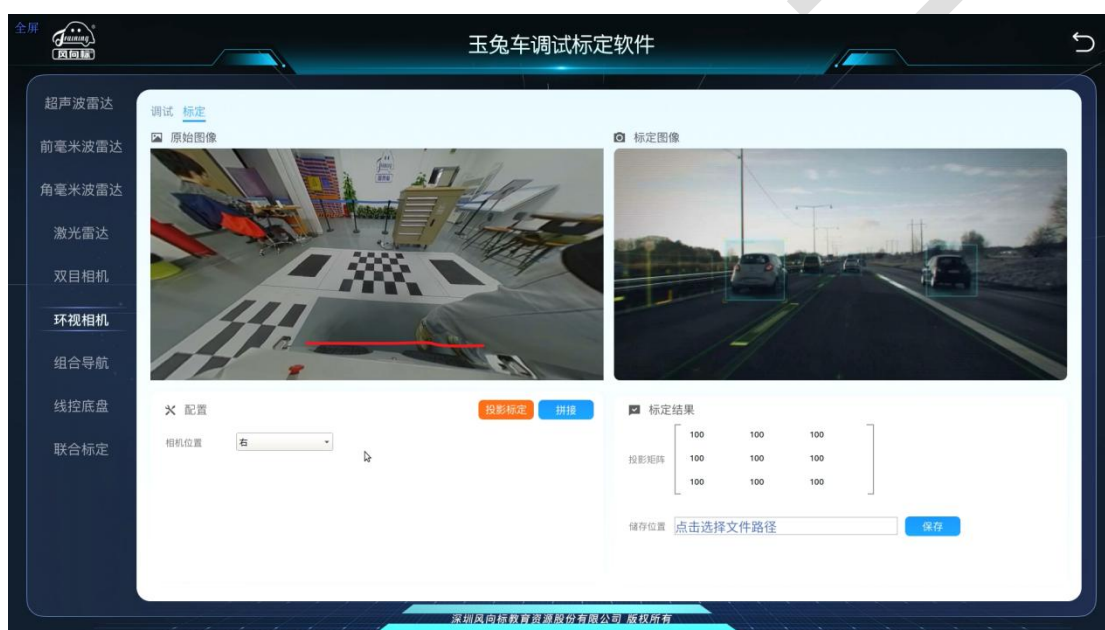




前后环视相机的视野最低点参照物是车辆保险杠；



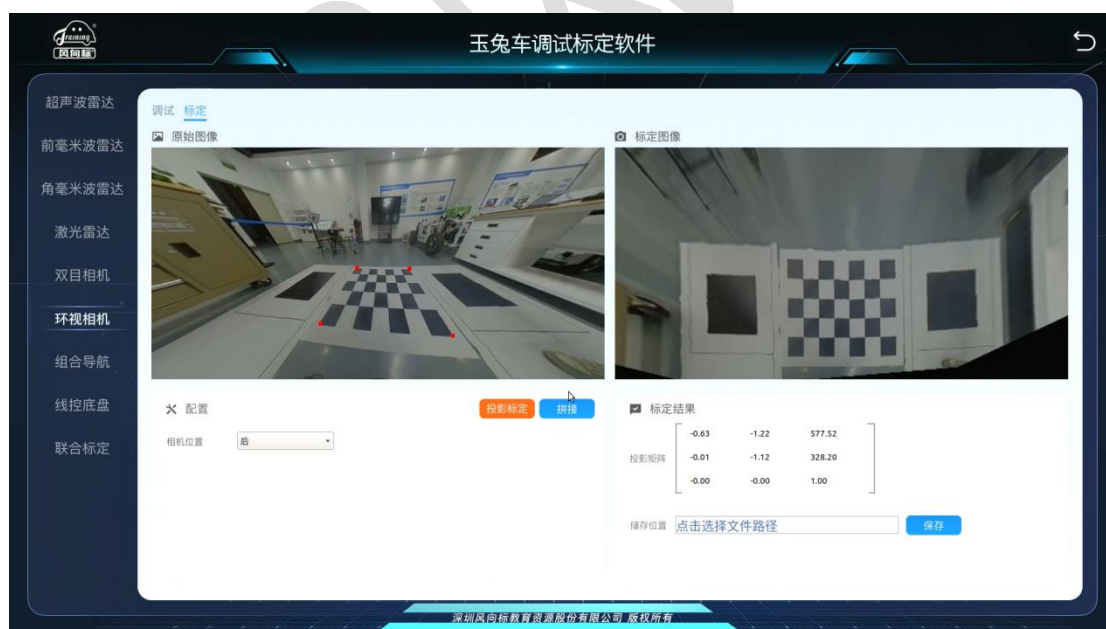
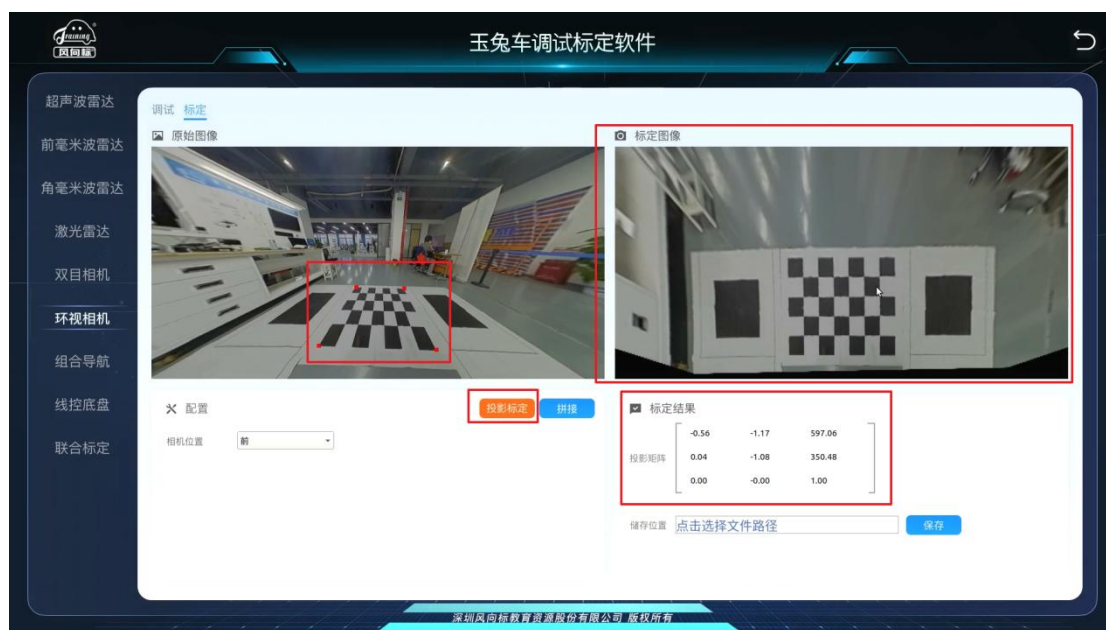


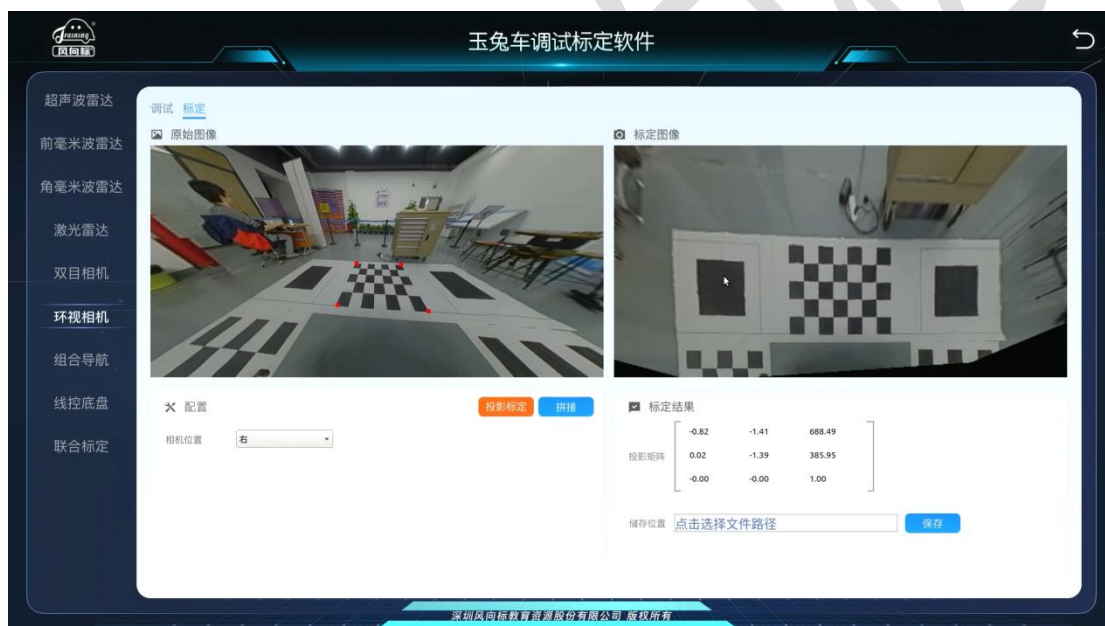
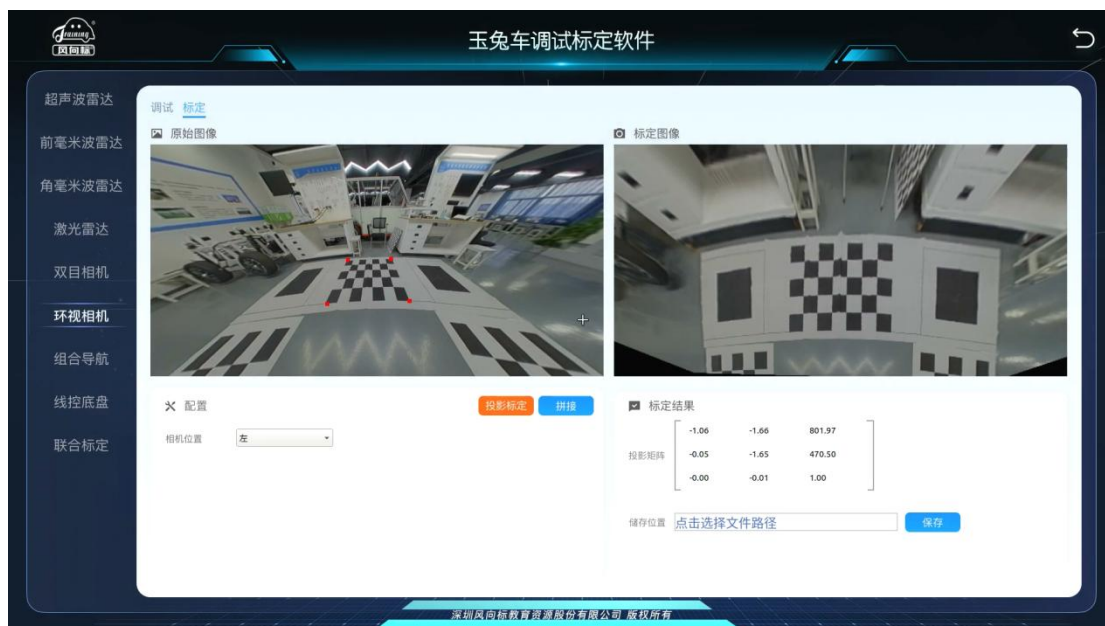




左右环视相机的视野最低点参照物是车辆左右测轮胎边缘；

5) 根据填写配置文件的特征点顺序，在软件上依次点击，点击完成后点击投影标定按钮进行透视变换，标定成功后会生成投影矩阵；当一个相机标定完成后，切换下一个相机进行标定时，需要先点击投影标定按钮，去除上一个相机标定点击的特征点，再进行点击特征点和点击投影标定按钮；





6) 当 4 个环视相机都完成投影标定后，点击拼接合成全景图像；



2.14 组合导航调试

1) 找到组合导航，点击开启；



2) 如果看到有数据刷新，则组合导航正常，否则异常；（**注意！在室内可能只会显示原始串口数据，其他部分为空白，属于正常现象，因为在室内无法接收卫星信号**）

2.15 组合导航标定

2.15.1 场地要求

要求地面尽量水平，无沟槽无凸起物。

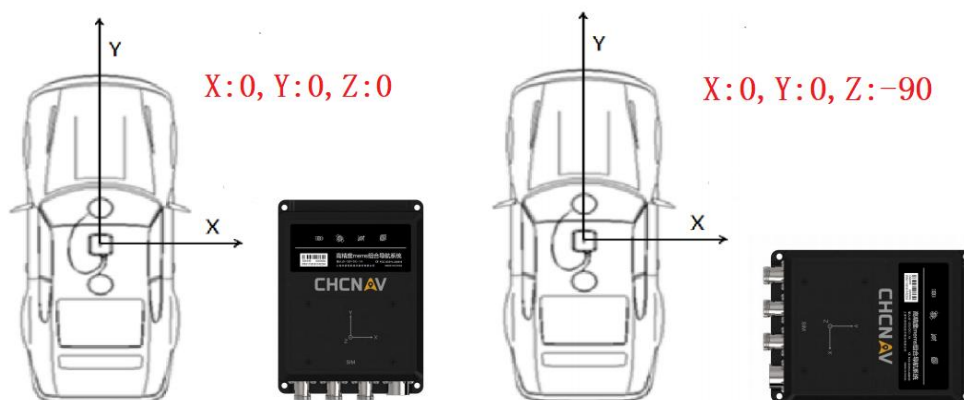
2.15.2 准备工作

1) 准备卷尺并检查是否可以正常使用。



2.15.3 标定过程

1) 由于惯导坐标系和车辆坐标一致（这里的车辆坐标系是组合导航厂家定义的，自驾软件的车辆坐标系不是这样，但是软件层面会做转换），所以惯导到车辆坐标系夹角 deg ，为 $X: 0, Y: 0, Z: 0$ 。如果惯导是朝顺时针旋转 N 度摆放，则 $X: 0, Y: 0, Z: -N$ 。如果惯导是朝逆时针旋转 N 度摆放，则 $X: 0, Y: 0, Z: N$ 。



2) 使用卷尺测量定位天线到后轮中心杆臂 m ，以定位天线为原点，车头朝向 Y

正，车辆右侧朝向 X 正，车顶朝向 Z 正，测量到后轮中心杆臂的偏移。



3) GNSS 定向基线与车辆坐标系夹角 deg, 天线前后安装, 则 X: 0, Y: 0, Z: 0。
 定位天线左, 定向右安装, 则 X: 0, Y: 0, Z: -90。定向天线左, 定位右安装,

则 X: 0, Y: 0, Z: 90。本车是以定位天线在后，定向天线在前安装。



4) 使用卷尺测量惯导到 GNSS 定位天线杆臂 m ，以惯导为原点，测量到 GNSS 定位天线臂杆的偏移。





5) 使用卷尺测量轮距和轴距 m。

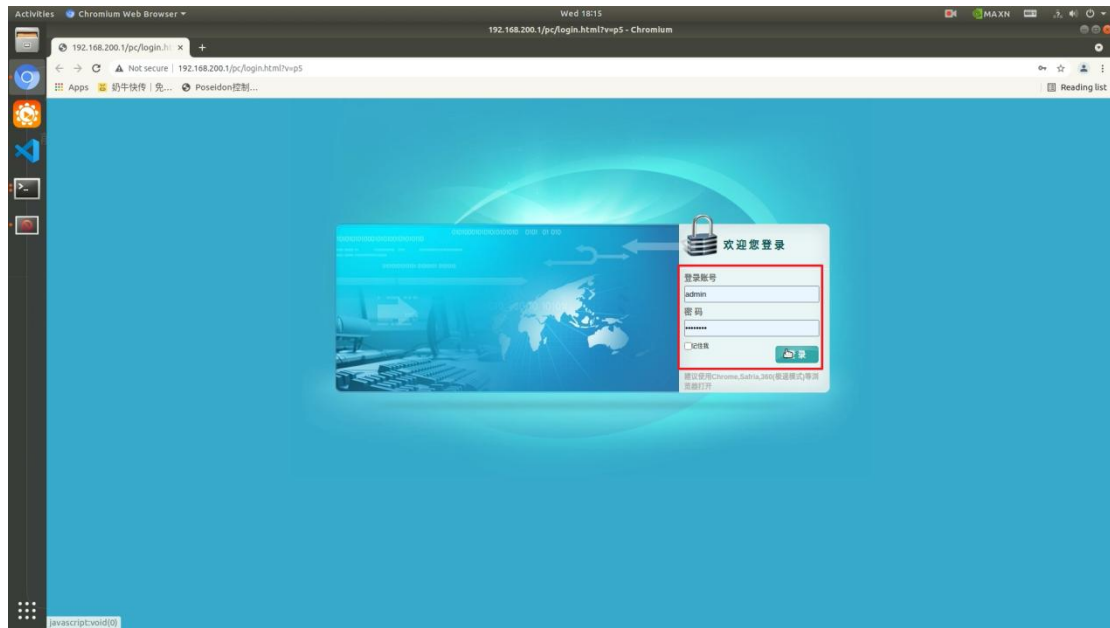




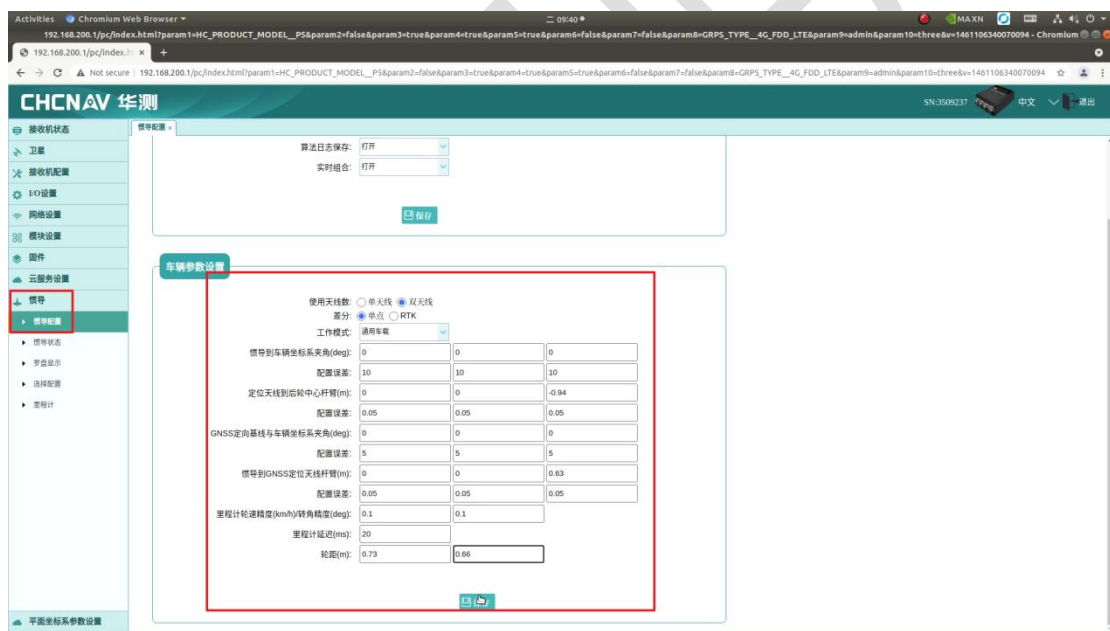
6) 连接组合导航 wifi, wifi 名在组合导航上有写明。



7) 打开浏览器, 输入 IP: 192.168.200.1 进入厂家提供的配置网页, 输入账号: admin, 密码: password 后点击登录。



8) 找到“惯导”下的“惯导配置”，将前面测量值填到“车辆参数设置”后，点击保存提示设置成功，就完成了组合导航标定（工作模式选择低速车载模式）。



9) 每一次修改车辆参数设置后，都需要进行跑车标定，需要将车辆速度超过 6km/h，进行口字型跑法，Ins 模式会从初始化到组合导航的状态跳转，如果速度过低则无法进行标定，所以需要尽量在空旷场地和较高的车速下进行，标定只



需要一次。



2.16 线控底盘线控测试

2.16.1 场地要求

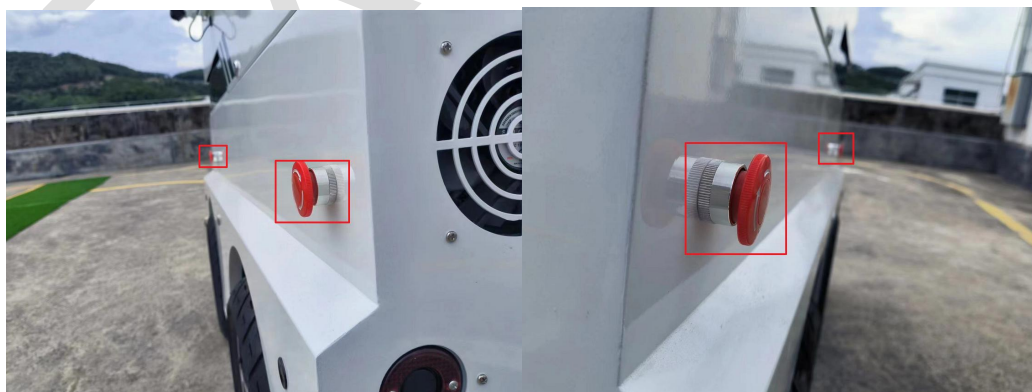
要求地面尽量水平，无沟槽无凸起物，四周空旷。

2.16.2 准备工作

- 1) 准备千斤顶、警示牌、警戒线和灭火器；
- 2) 摆放警示牌，拉起警戒线；
- 3) 使用千斤顶将车辆举升，确保后轮（驱动轮）离地；



- 4) 检查车身两侧的应急开关是否已弹出；



- 5) 开启遥控器，将控制模式设置为自动驾驶模式；



2.16.3 测试过程

2.16.3.1 驱动测试

- 1) 打开装调标定教学软件，进入线控底盘的线控测试界面；



2) 点击“线控驱动”，进入驱动测试页面；



3) 点击“制动力”进度条，释放制动；



4) 根据要求点击“请求速度”进度条的灰色部分加速，点击蓝色部分减速，点击“R 档按钮”车轮反转，点击“D 档按钮”车轮正转；



5) 查看当前车辆速度反馈，正转车速为正数，反转车速为负数，车速单位为千米每小时；



6) 在报告单上记录实际速度；

2.16.3.2 转向测试

1) 打开装调标定教学软件，进入线控底盘的线控测试界面；



2) 点击“线控转向”，进入转向测试页面；



3) 点击“制动力”进度条，释放制动；



4) 根据要求点击“请求角度”进度条的灰色部分递增角度，点击蓝色部分递减角度，点击“右转按钮”车轮右转，点击“左转按钮”车轮左转；



5) 查看当前车辆转向反馈，左转为正度数，右转为负度数，转向角度单位为度；



6) 在报告单上记录实际角度；

2.17 线控底盘 CAN 通讯

2.17.1 场地要求

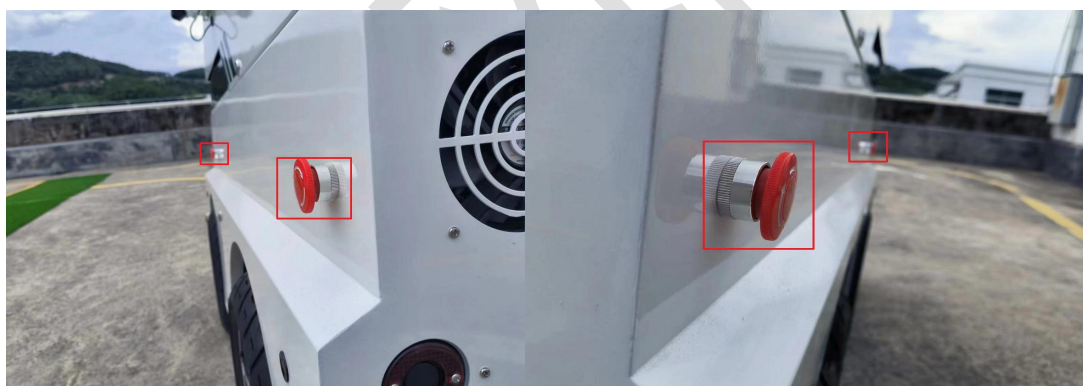
要求地面尽量水平，无沟槽无凸起物，四周空旷。

2.17.2 准备工作

- 1) 准备千斤顶、警示牌、警戒线和灭火器；
- 2) 摆放警示牌，拉起警戒线；
- 3) 使用千斤顶将车辆举升，确保后轮（驱动轮）离地；



- 4) 检查车身两侧的应急开关是否已弹出；



- 5) 开启遥控器，将控制模式设置为自动驾驶模式；



2.17.3 转向驱动测试过程

- 1) 打开装调标定教学软件，进入线控底盘的 CAN 通讯界面；



2) 将帧格式设置为拓展帧（由帧 ID 的长度可知，此帧为拓展帧）；



3) 根据提供的控制协议，计算 CAN 报文并控制底盘；

ctrl_cmd			0x18C4D2D0				10		8
信号描述	排列格式	起始字节	起始位	信号长度	数据类型	精度	偏移量	单位	信号值描述
目标档位	Intel	0	0	4	Unsigned	1	0		00: disable 01: P 档 02: R 档 03: N 档 04: D 档
目标车体速度	Intel	0	4	16	Unsigned	0	0	m/s	0.001m/s/bit;
目标车体转向角	Intel	2	20	16	signed	0.01	0	°	0.01°/bit;

假设需要控制目标档位 D 档，车体速度为 2.7612km/h，车体转角 0 度。

根据协议可知 D 档是 04，转换二进制为 0100，将值填入字段表格（**注意！起始位为 0**）。

	7	6	5	4	3	2	1	0
0	车体速度 7 0	6 0	5 0	4 0	目标档位 3 0	2 1	1 0	0 0
1	15 0	14 0	13 0	12 0	11 0	10 0	9 0	8 0
2	车体转角 23 0	22 0	21 0	20 0	19 0	18 0	17 0	16 0
3	车体转角 31 0	30 0	29 0	28 0	27 0	26 0	25 0	24 0
4	39	38	37	36	车体转角 35 0	34 0	33 0	32 0
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56

因为控制协议的控制速度单位为 m/s，所以 $2.7612\text{km/h}=0.767\text{m/s}$

协议说明每 bit 位对应的速度为 0.001m/s，那么要求出 0.767m/s 对应多少个 bit 位， $0.767/0.001 = 767$ ，将 767 转换二进制为 10 1111 1111，为了凑够 16 位需要补零，0000 0010 1111 1111，将值填入字段表格（**注意！起始位为 4**）。

	7	6	5	4	3	2	1	0
0	车体速度 7 1	6 1	5 1	4 1	目标档位 3 0	2 1	1 0	0 0
1	15 0	14 0	13 1	12 0	11 1	10 1	9 1	8 1
2	车体转角 23 0	22 0	21 0	20 0	19 0	18 0	17 0	16 0
3	车体转角 31 0	30 0	29 0	28 0	27 0	26 0	25 0	24 0
4	39	38	37	36	车体转角 35 0	34 0	33 0	32 0
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56

转向角度为 0 度，协议说明每 bit 位对应的角度为 0.01 度，那么需要求出 0 度对应多少个 bit 位， $0/0.01 = 0$ ，将 0 转换二进制为 0，为了凑够 16 位需要补零，0000 0000 0000 0000，将值填入字段表格（**注意！起始位为 20**）。

	7	6	5	4	3	2	1	0
0	车体速度 7 1	6 1	5 1	4 1	目标档位 3 0	2 1	1 0	0 0
1	15 0	14 0	13 1	12 0	11 1	10 1	9 1	8 1
2	车体转角 23 0	22 0	21 0	20 0	19 0	18 0	17 0	16 0
3	车体转角 31 0	30 0	29 0	28 0	27 0	26 0	25 0	24 0
4	39	38	37	36	车体转角 35 0	34 0	33 0	32 0
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56

最后只需将字段表格中的二进制转换为 16 进制，然后填入数据点击发送即可控制底盘。

1111 0100 = F4

0010 1111 = 2F

0000 0000 = 00

0000 0000 = 00

0000 0000 = 00



可以通过反馈确认报文是否计算正确。





假设需要控制目标档位 D 档，车体速度为 0km/h，车体右转 20 度。

目标档位二进制为 0100

车体速度二进制为 0000 0000 0000 0000

	7	6	5	4	3	2	1	0
0	车体速度 7 0	6 0	5 0	4 0	目标档位 3 0	2 1	1 0	0 0
1	15 0	14 0	13 0	12 0	11 0	10 0	9 0	8 0
2	车体转角 23 0	22 0	21 0	20 0	19 0	18 0	17 0	16 0
3	车体转角 31 0	30 0	29 0	28 0	27 0	26 0	25 0	24 0
4	39	38	37	36	车体转角 35 0	34 0	33 0	32 0
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56

协议转向没有说明如何控制方向，根据我们实测**左转为正角度，右转为负角度**。

正角度的计算方法和上面一致，这里主要说明如何计算负角度。

转向角度为 20 度，协议说明每 bit 位对应的角度为 0.01 度，那么需要求出 20 度对应多少个 bit 位， $20/0.01 = 2000$ ，将 2000 转换二进制为 111 1101 0000，为了凑够 16 位需要补零，0000 0111 1101 0000，计算到这里都和前面说明的一样，现在说明如何控制转向方向，协议上有说明转向的数据类型是 signed 有符号的，



原码的最高位表示符号位（1 表示负数，0 表示正数），则 1000 0111 1101 0000（-0000 0111 1101 0000），但是计算机底层的二进制都是采用补码形式（因为补码可以比原码多表示一个数，在有符号的情况下），**注意！正数的原码、反码和补码都是一样的，负数有所不同，反码是在原码的基础上，除符号位外其余都取反，补码是在反码的基础上+1。**

原码：1000 0111 1101 0000

反码：1111 1000 0010 1111

补码：1111 1000 0011 0000

将补码填入字段表格（**注意！起始位为 20**）。

	7	6	5	4	3	2	1	0
0	车体速度 7 0	6 0	5 0	4 0	目标档位 3 0	2 1	1 0	0 0
1	15 0	14 0	13 0	12 0	11 0	10 0	9 0	8 0
2	车体转角 23 0	22 0	21 0	20 0	19 0	18 0	17 0	16 0
3	31 1	30 0	29 0	28 0	27 0	26 0	25 1	24 1
4	39	38	37	36	车体转角 35 1	34 1	33 1	32 1
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56

最后只需将字段表格中的二进制转换为 16 进制，然后填入数据点击发送即可控制底盘。

0000 0100 = 04

0000 0000 = 00

0000 0000 = 00

1000 0011 = 83

0000 1111 = 0F



可以通过反馈确认报文是否计算正确。





2.17.4 灯光测试过程

1) 选择帧 ID 为 0x18C4D7D0，灯光控制



2) 将帧格式设置为拓展帧（由帧 ID 的长度可知，此帧为拓展帧）；



3) 根据提供的控制协议，计算 CAN 报文并控制底盘；

io_cmd			0x18C4D7D0				50		8
信号描述	排列格式	起始字节	起始位	信号长度	数据类型	精度	偏移量	单位	信号值描述
I/O 控制使能	Intel	0	0	1	Unsigned	1	0		0 = 关闭 1 = 打开
转向灯开关	Intel	1	10	2	Unsigned	1	0		0 = 全关 1 = 开启左转向灯 2 = 开启右转向灯
示廓灯开关	Intel	1	13	1	Unsigned	1	0		0 = 关闭 1 = 打开

假设车辆需要开启左转向灯和示廓灯，根据协议转换成二进制填入协议表格

I/O 控制使能：1

转向灯开关：01

示廓灯开关：1

	7	6	5	4	3	2	1	0
0	7	6	5	4	3	2	1	I/O 控制使能 0
1	15	14	13 示廓灯开关 1	12	11 转向灯开关 0	10 1	9	8
2	23	22	21	20	19	18	17	16
3	31	30	29	28	27	26	25	24
4	39	38	37	36	35	34	33	32
5	47	46	45	44	43	42	41	40
6	55	54	53	52	51	50	49	48
7	63	62	61	60	59	58	57	56

将协议表格的二进制一行一行提取，空白地方为 0

0000 0001 = 01

0010 0100 = 24

0000 0000 = 00

0000 0000 = 00

0000 0000 = 00

将 16 进制填入数据部分，点击发送，对应车灯将会点亮



2.18 线控底盘中位标定

2.18.1 场地要求

要求地面尽量水平，无沟槽无凸起物，四周空旷。

2.18.2 准备工作

- 1) 准备千斤顶、警示牌、警戒线和灭火器；
- 2) 摆放警示牌，拉起警戒线；
- 3) 使用千斤顶将车辆举升，确保后轮（驱动轮）离地；



- 4) 检查车身两侧的应急开关是否已弹出；



- 5) 开启遥控器，将控制模式设置为自动驾驶模式；



2.18.3 标定过程

1) 打开装调标定软件，找到线控底盘下“中位标定”；



2) 拖动转向控制条，目测轮胎居中后，点击“开始标定”按钮；



3) 打开车辆盖子，找到“转向控制线束断路”按钮，按下断开转向控制线束；



4) 点击“结束标定”按钮；



5) 恢复“转向控制线束断路”按钮；



6) 可以遥控车辆看看能否走直线，不能则还需完成步骤 1~5，直至车辆能正常走直线。

2.19 毫米波雷达和相机联合标定

2.19.1 场地要求

要求地面尽量水平，无沟槽无凸起物，标定区域为 7m*3m 以上的空旷区域。

2.19.2 准备工作

1) 准备激光测距仪、铅垂线、角度尺、卷尺、记号笔和角反射器，并检查是否可以正常使用；



- 2) 将车辆停稳，车头摆正；
- 3) 确保车辆没有放置重物，避免影响车辆水平；

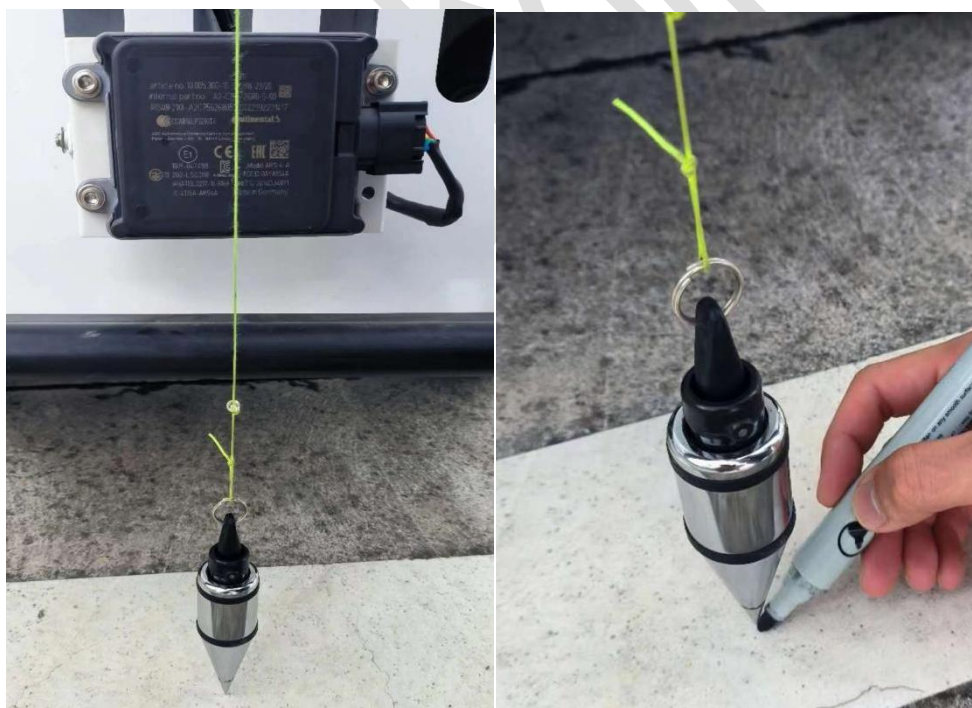
2.19.3 标定过程

- 1) 调节角反射器高度，使角反射器中心与前毫米波雷达中心高度一致；



- 2) 使用铅锤线，将毫米波雷达检测面中心对应的前保险杠垂直到地面，并使用

记号笔标记，记作 A 点（**特别注意，使用铅垂线时不要被固定针扎到手**）；



3) 使用铅锤线，将毫米波雷达检测面中心旁边任意点对应的前保险杠垂直到地面，并使用记号笔标记，记作 B 点；



4) 使用角度尺将 AB 两点连接;



5) 使用角度尺, 先校 0 然后调节成 90 度, 画一条垂直 A 点的线, 此线指向的就是毫米波雷达检测物体的 0 度方向 (插接件指向正角度, 反方向指向负角度);

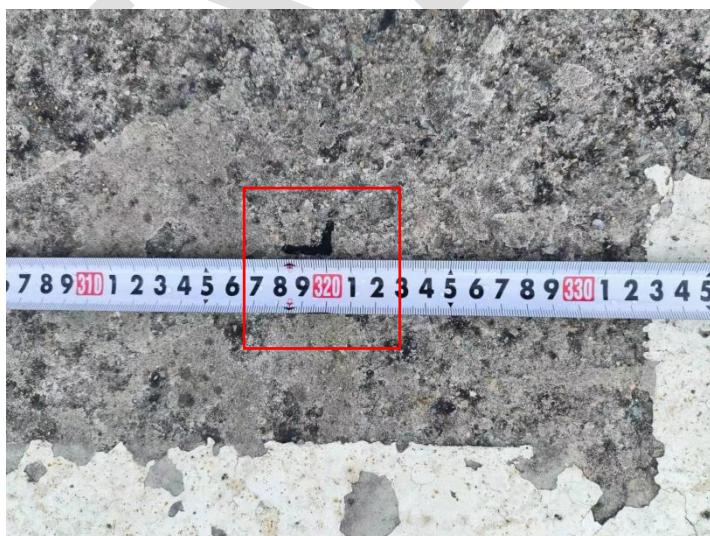


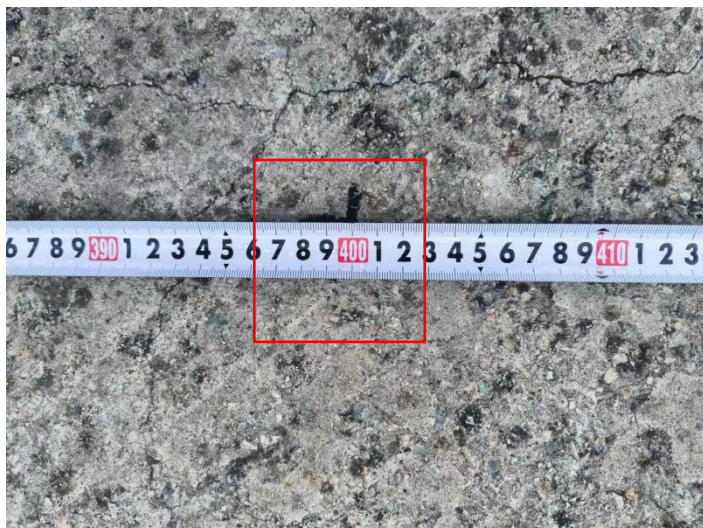


6) 将 A 点向左向右各延长 0.4 米，向左延长点记作 C 点，向右延长点记作 D 点，使用角度尺画一条垂直 C 点和 D 点的线；



7) 使用卷尺，以 A 点、C 点和 D 点作为起点，在垂直方向的 3.2 米、4 米和 4.8 米处使用记号笔标记，一共标记 9 个点；





8) 打开装调标定教学软件，进入联合标定界面；





9) 依次将角反射器摆放在 9 个标记点，每摆放一次，需要到相机画面上（左边画面）点击**角反射器垂直到地面的点**和毫米波雷达检测物体画面上（右边画面）点击检测角反射器的坐标点，由这两个点构成一个点对，**注意！将角反射器摆放到一个标记点时，点击两边画面得到一个点对，一共需要摆放 9 次得到 9 个点对**；



10) 当完成 9 个点对后，点击标定，会计算出单应矩阵，此矩阵的作用是将毫米波雷达检测的坐标点经过单应矩阵后转换到相机画面上对应物体垂直地面上的点；





11) 标定成功后，将标定参数保存到指定位置；



12) 标定成功后，在报告单上记录单应矩阵；

13) 当标定成功后，点击验证，可以看到毫米波雷达和相机的融合画面，蓝色点在角反射器（金属物体）垂直到地面的位置，说明标定效果好；



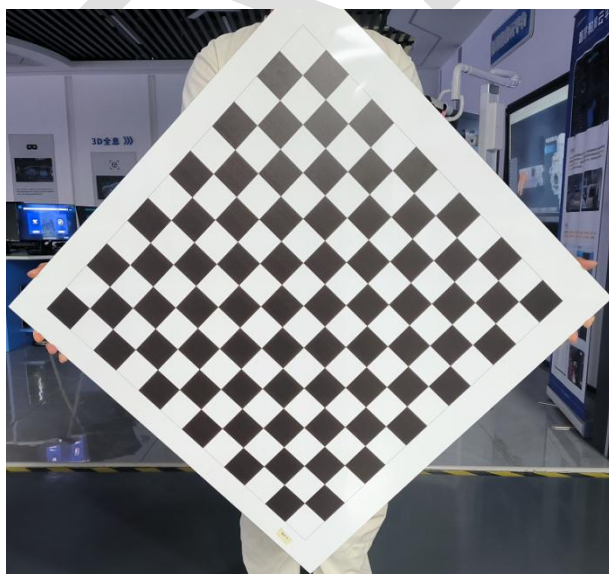
2.20 激光雷达和相机联合标定

2.20.1 场地要求

要求地面尽量水平，无沟槽无凸起物，标定区域为 7m*3m 以上的空旷区域；

2.20.2 准备工作

1) 准备指定棋盘格；



2.20.3 标定过程

1) 打开装调标定教学软件，进入联合标定界面；

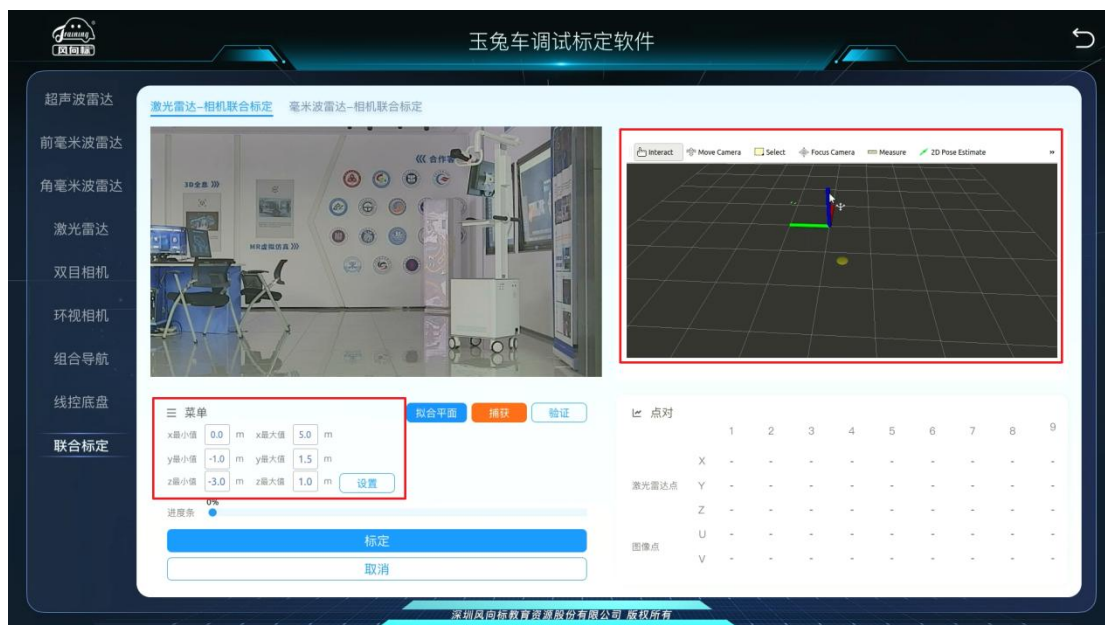


2) 设置合适的感兴趣区域 (ROI) 将环境障碍物的点云去除；



以上图为例，右侧显示点云的网格中每一格表示 1m，我们标定时，只要车前面的点云，所以我们需要把 x 最小值设置为 0，我们可以看到车左侧 1.5m 内是干净的，所以 Y 最大值为 1.5，车右侧是 1m 是干净的，所以 Y 最小值为 1，当我们不断调整 XYZ 的最大值和最小值（填写值后需要点击设置才生效），直到把周围

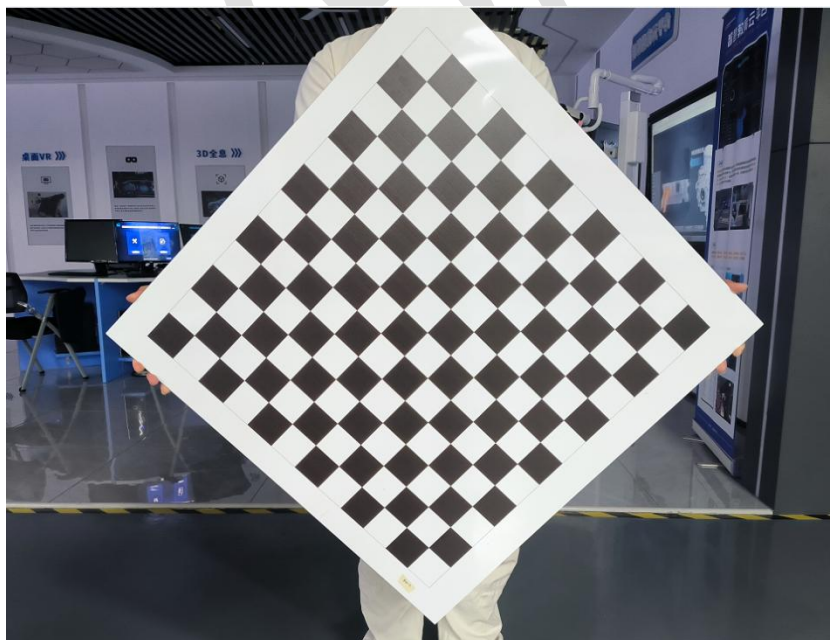
环境的点云尽可能清除干净。下图是清除环境点云后的效果。

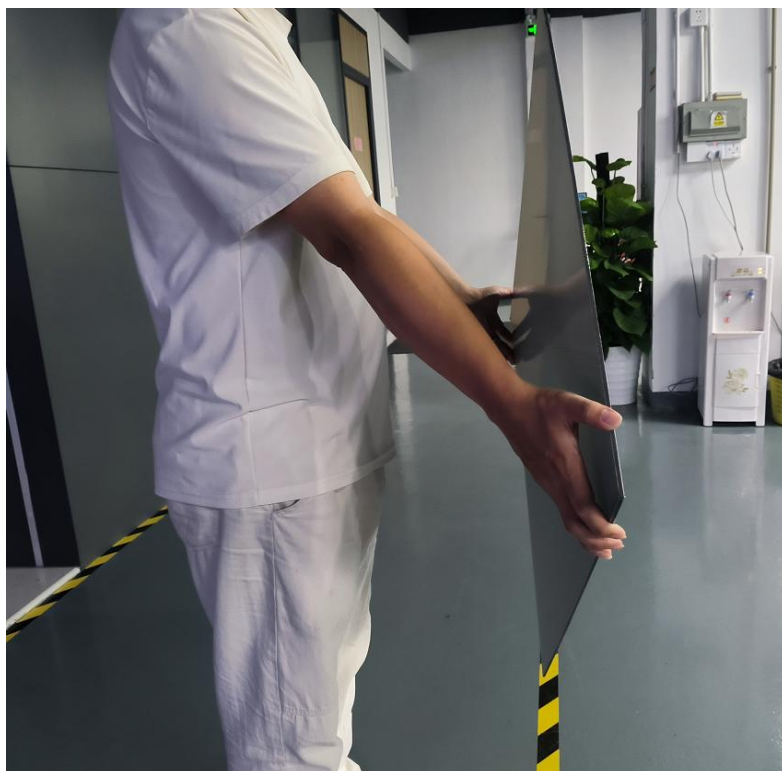


简单理解，设置一个 XYZ 的最大值和最小值，右侧将会把激光雷达的点云中的点，在我们设置的范围内将显示，其余将丢弃，右侧中有一个三色坐标系，红色指向 X 正，绿色指向 Y 正，蓝色指向 Z 正。

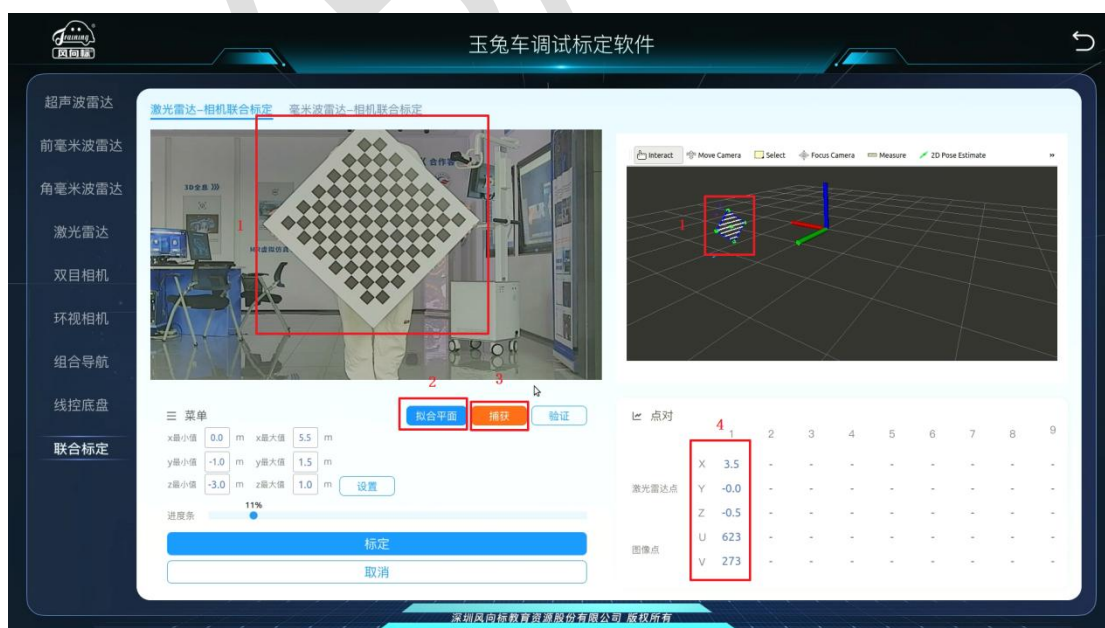
3) 下面需要拿起标定板在相机面前站 9 个位置，每站在一个位置时需要在软件上点击拟合平面，当确认拟合成功时，点击捕获。

拿标定板时需要把标定板的角正直朝上，同时把手往前伸，不要靠着身体。

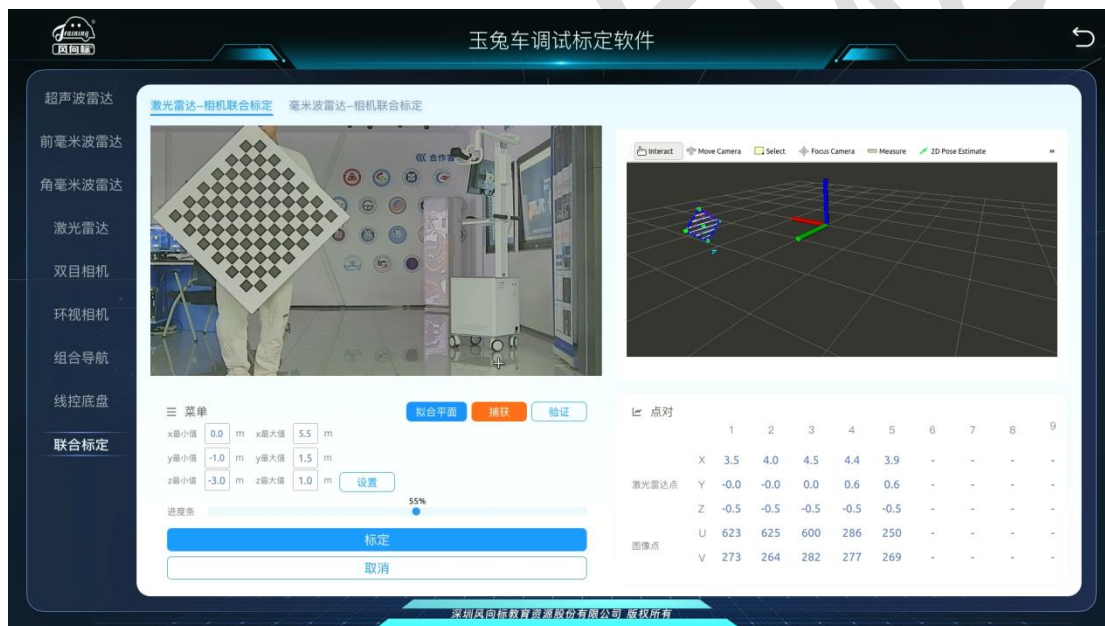
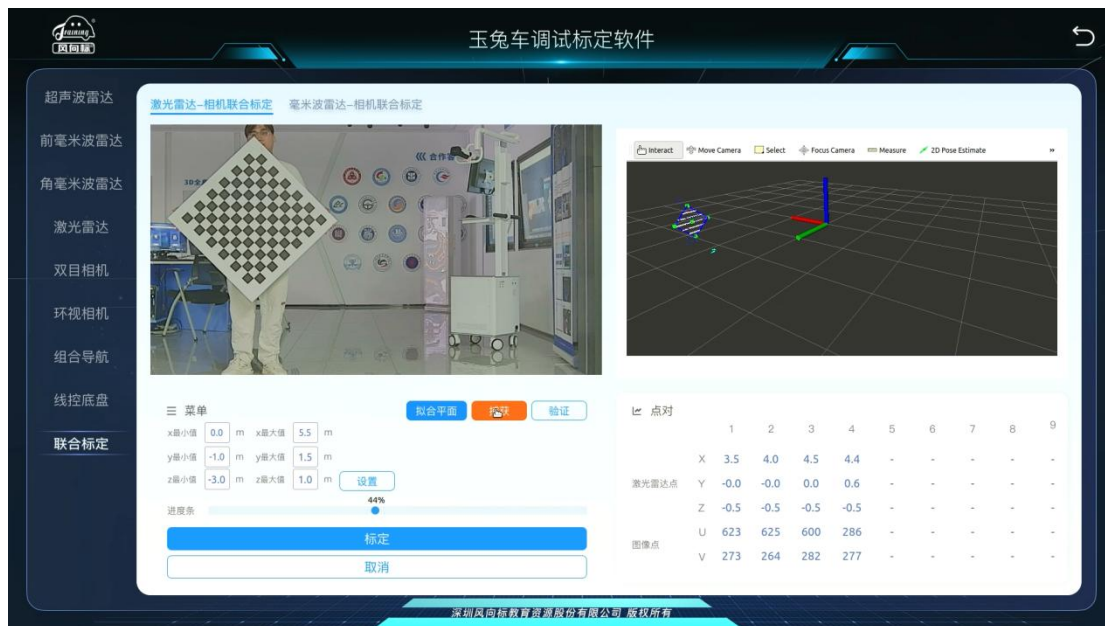


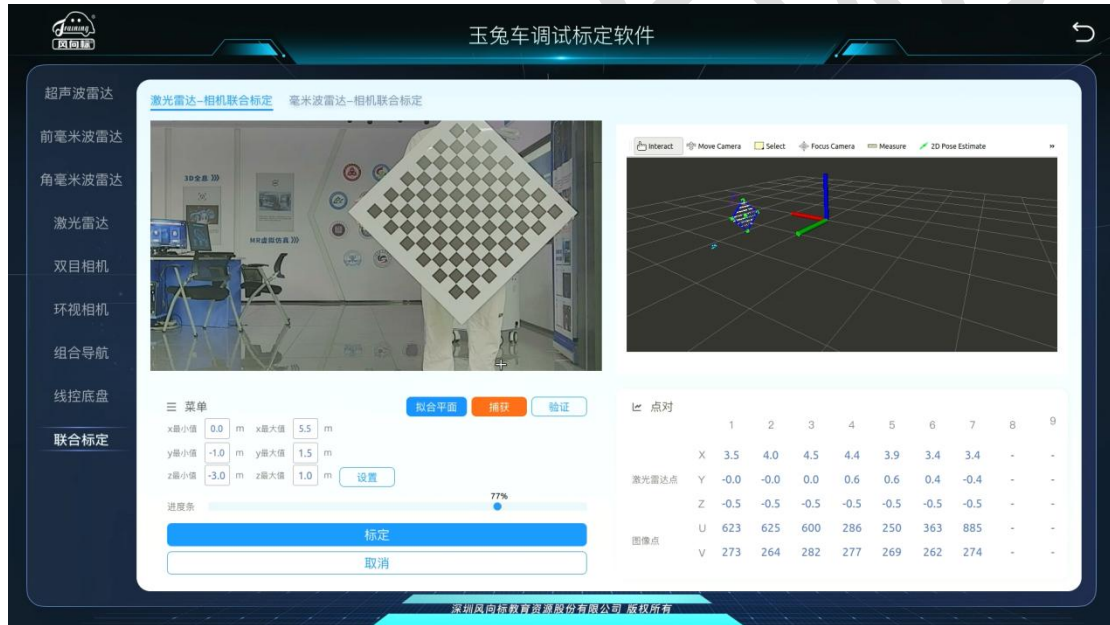
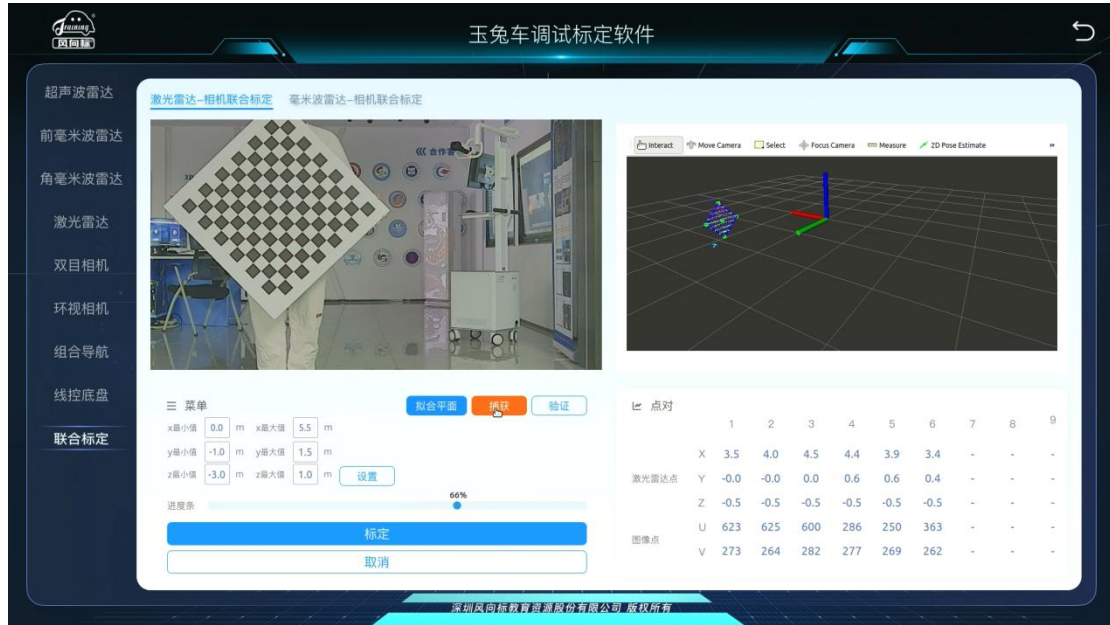


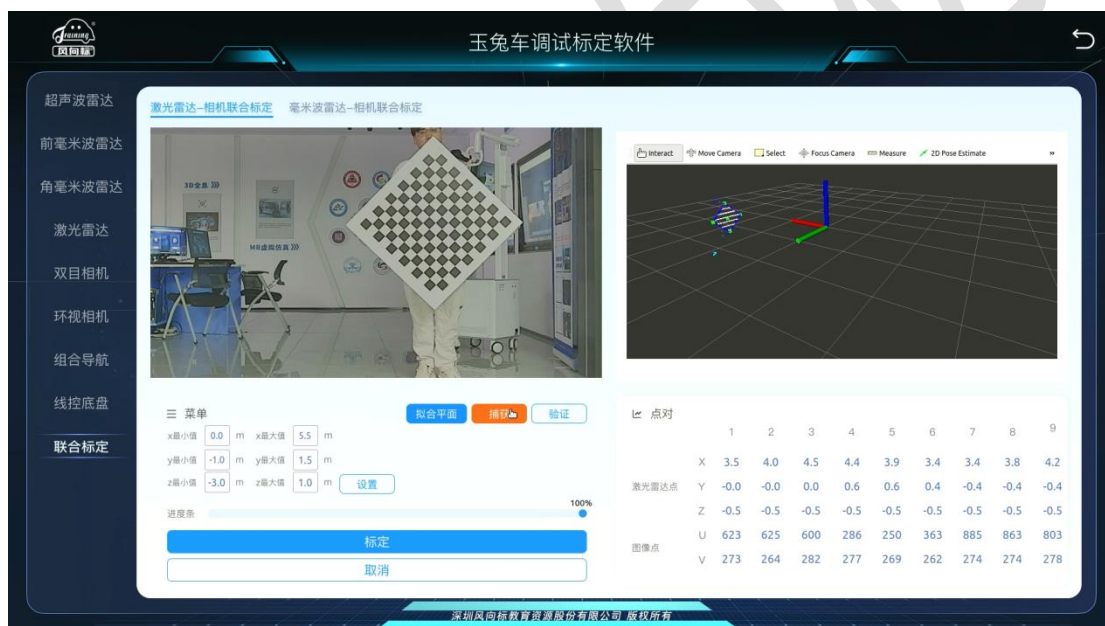
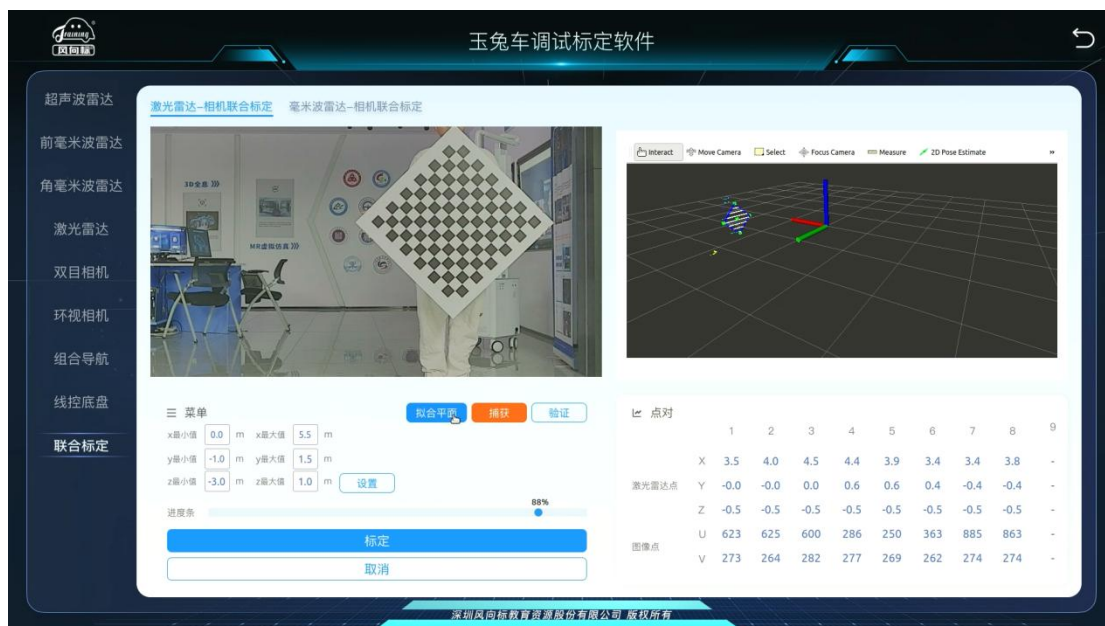
下面将介绍 9 个位置应该怎么站，你需要在相机画面下，能拍全棋盘格在相机画面的中间位置、最左边位置和最右边位置同时需要保证激光雷达能扫描全棋盘格的上半部分（能满足此要求下，离相机越近越好），然后在中间位置、最左边位置和最右边位置为起点采集一次后，再每往后退一步采集一次，9 个位置就是这么得的，下面是我们采集 9 次不同站位的图片。







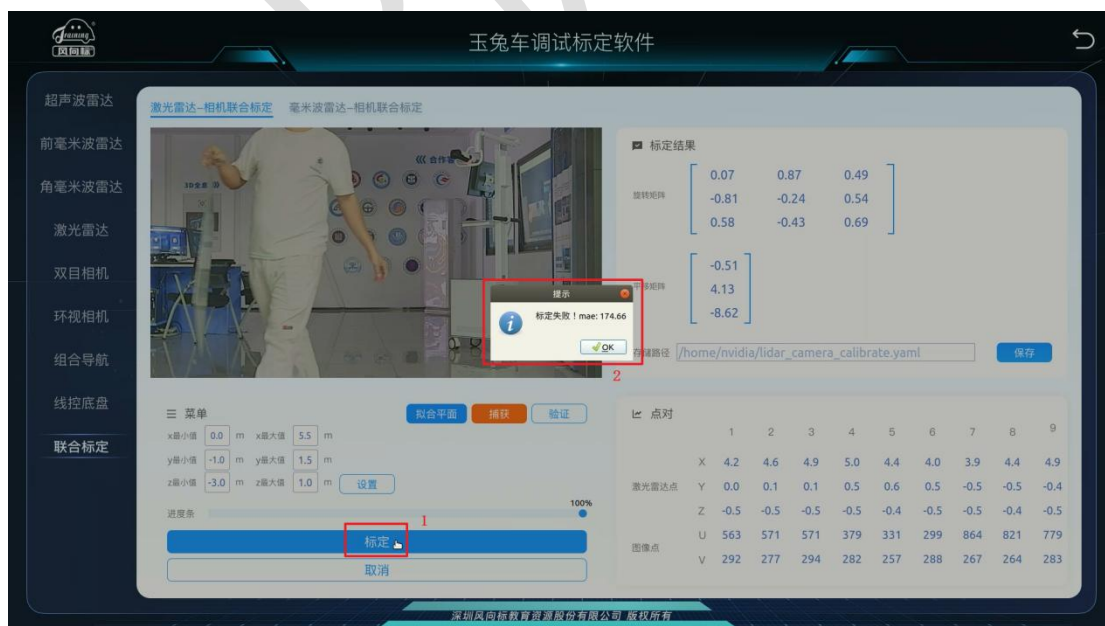




注意！如果提示拟合平面失败或拟合效果不好（如果右侧没有形成棋盘格此时摆放的形状或形状差异较大则认为拟合效果不好），请不要点击捕获，应点击取消拟合后，重新调整站位后再点击拟合平面。

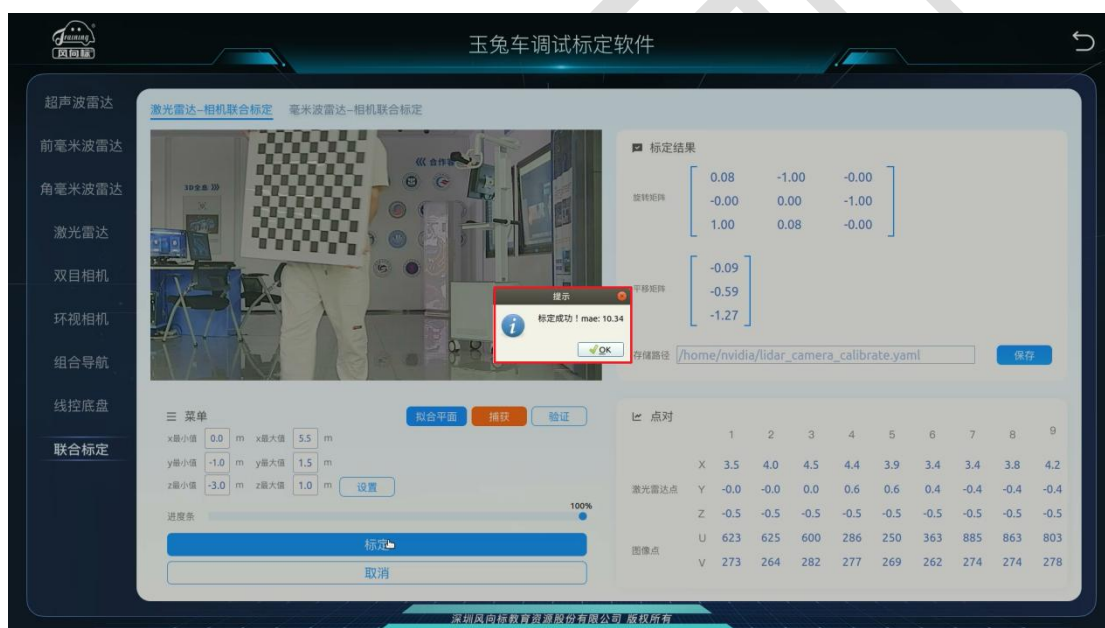


4) 当我们采集 9 次后，点击标定按钮，如果标定成功则提示标定成功，如果标定失败，你可以尝试点击验证按钮，看一下标定失败的融合画面是怎么样的，会看到点云和物体并不重合，没关系点击取消按钮，重新再来一遍就行，一般标定失败的原因就是 9 个点的站位不好或者拟合效果不好但点击了捕获，我们在测试时也不能一次就标定成功，但多尝试几次就成功了，所以不要灰心再来一遍就行。





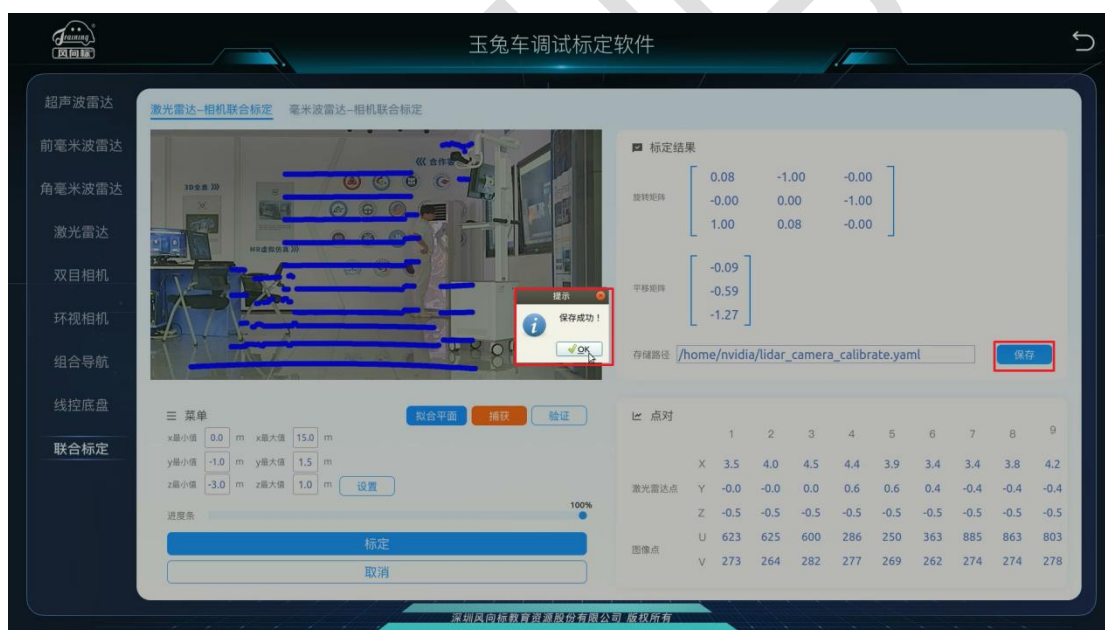
下面是标定成功的融合图像效果，mae 越小融合的效果就会越好。







5) 如果标定成功请点击保存，软件会把此时计算出来的 3×3 的旋转矩阵和 3×1 的平移向量保存到指定路径下的指定文件。



第 3 章 自动驾驶教学软件

3.1 使用建议

自动驾驶测试的场地周围需要有一定固定物体（例如房屋、树木等），不宜过于空旷，**不推荐在非常空旷的操场或空地**进行自动驾驶测试，这可能会造成定位失败，如果没有办法，推荐在测试场地周围摆放一些标识牌、旗帜等高大的物体。最后，进行自动驾驶测试时**必须携带遥控器**，并**随时做好紧急接管**。

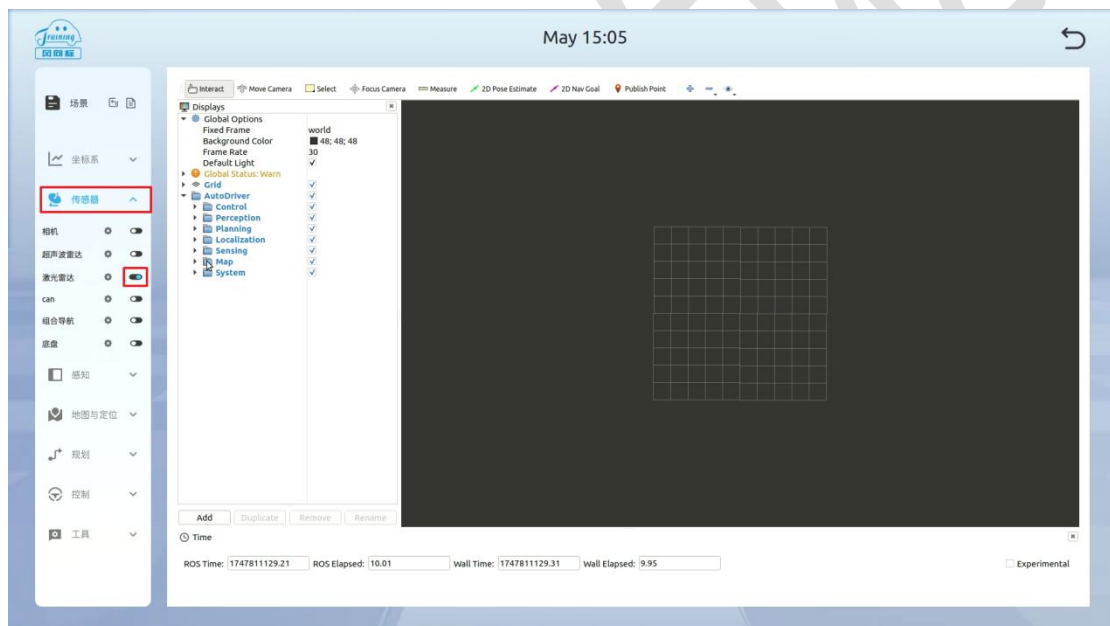
3.2 录制数据包

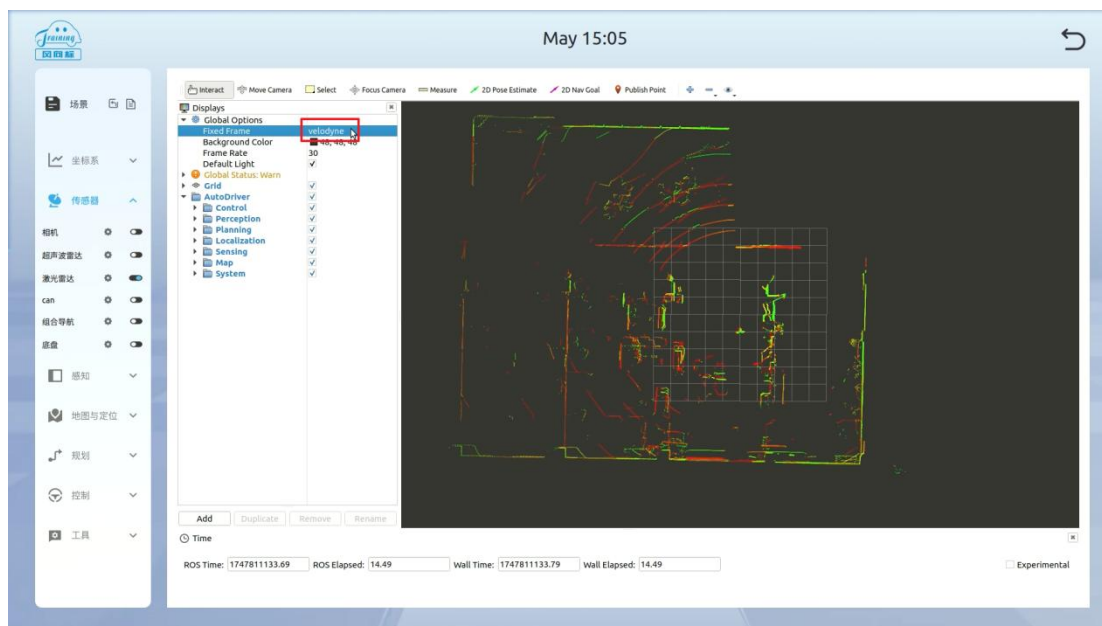
0) 在关闭所有软件和终端后再操作本小节。

1) 使用遥控器将车辆行驶到起始区域；

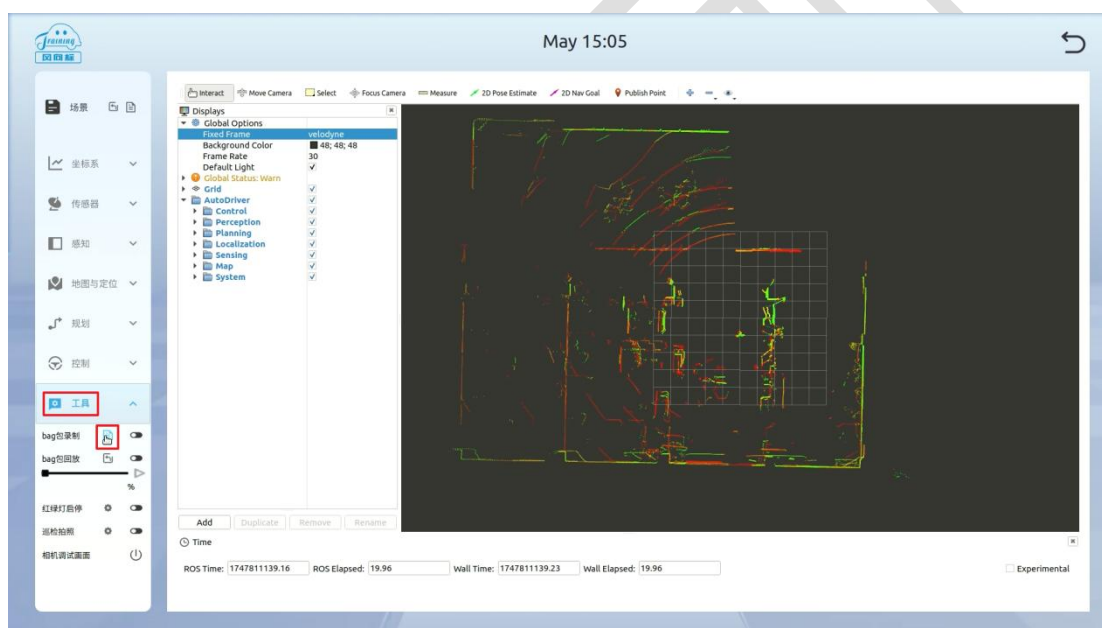


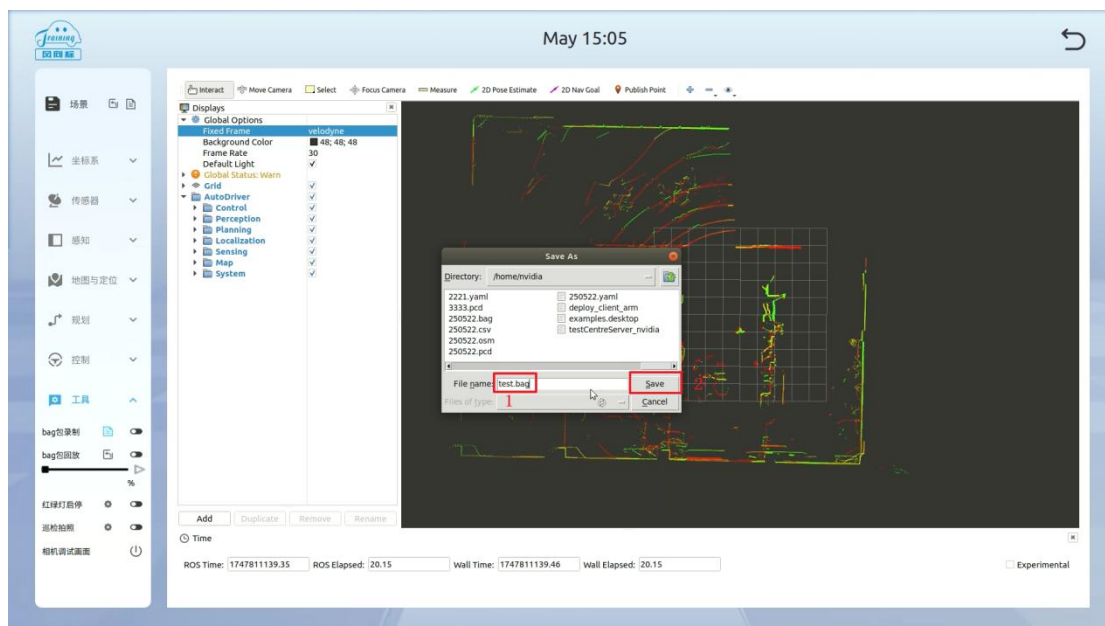
2) 打开自动驾驶教学软件，选择高级教学模式，找到“传感器”下的“激光雷达”，点击齿轮图标可以进行配置参数，勾选则启动激光雷达驱动，此时不会有
点云出现，因为参考坐标系没有填写为 velodyne；



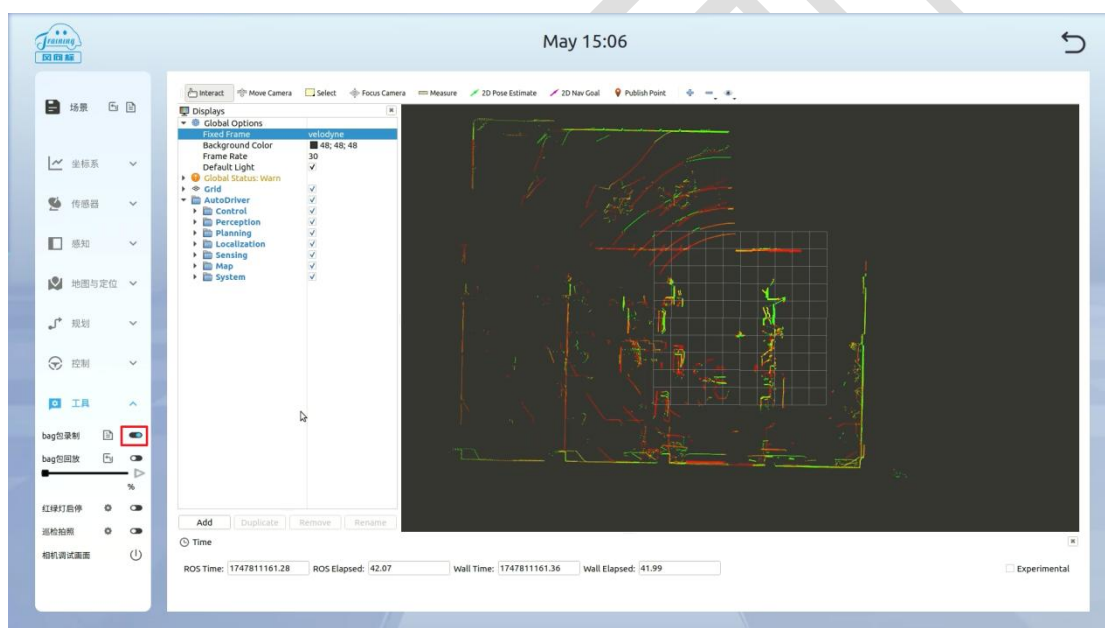


3) 找到工具下的“bag 包录制”，点击旁边的图标，选择保存路径及文件名；

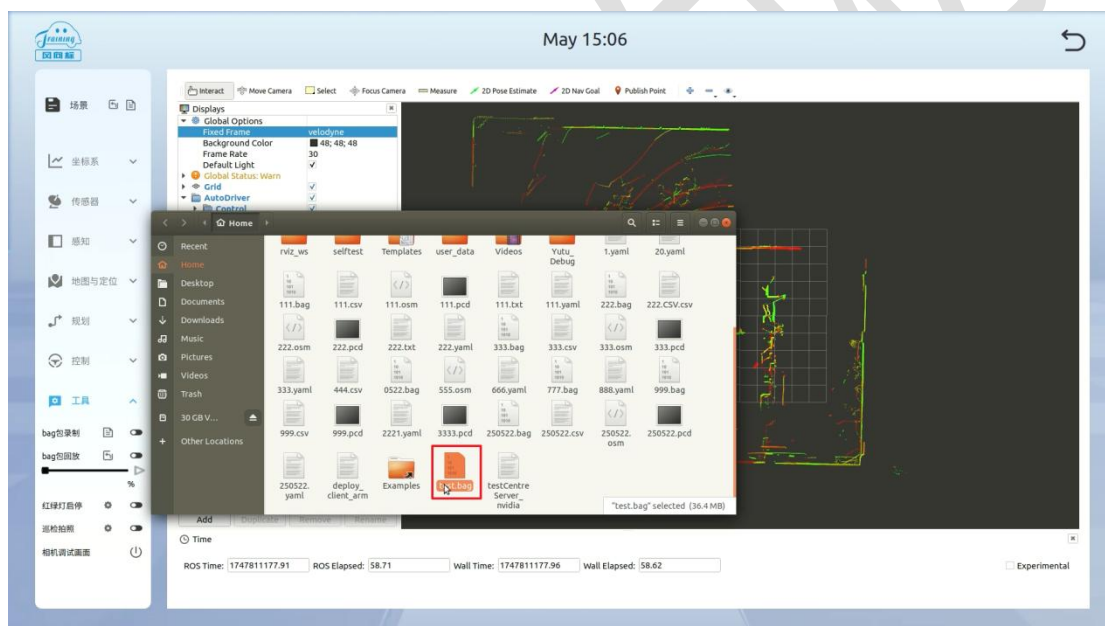
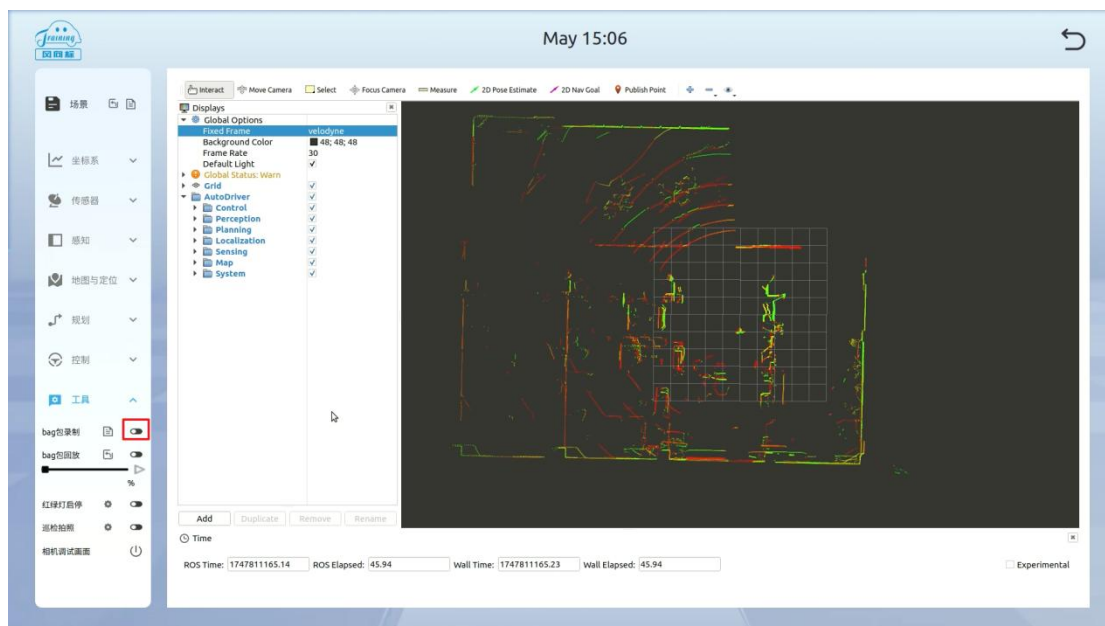




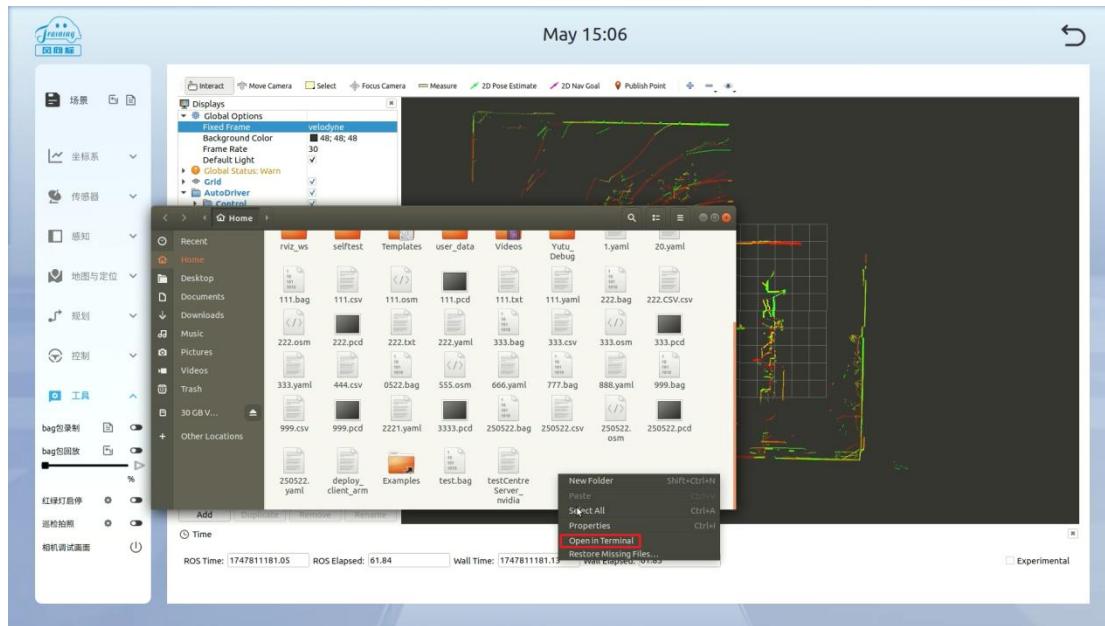
4) 勾选启动“bag 包录制”功能，此时需要遥控车辆行驶所需的路径；



5) 当车辆完成所需路径的行驶后，取消勾选“bag 包录制”功能，则会生成 bag 包到之前填写的保存路径；



6) 打开保存数据包的文件夹，在空白地方，右键选择“Open in Terminal”后会打开终端，输入命令 `roslaunch jf_training jf_training.launch` 数据包名字，查看数据包的信息；



```
nvidia@oceanstar: ~
File Edit View Search Terminal Help
nvidia@oceanstar:~$ rosbag info test.bag
path:      test.bag
version:   2.0
duration:  3.6s
start:     May 21 2025 15:06:01.91 (1747811161.91)
end:       May 21 2025 15:06:05.52 (1747811165.52)
size:      34.7 MB
messages:  44
compression: none [36/36 chunks]
types:     rosgraph_msgs/Log [acffd30cd6b6de30f120938c17c593fb]
           sensor_msgs/PointCloud2 [1158d486dd51d683ce2f1be655c3c181]
topics:    /points_raw 36 msgs : sensor_msgs/PointCloud2
           /rosout 8 msgs : rosgraph_msgs/Log (2 connections)
```

7) 记录激光雷达的帧数（注意！是数据发布的总数量，不是帧率）到报告单；

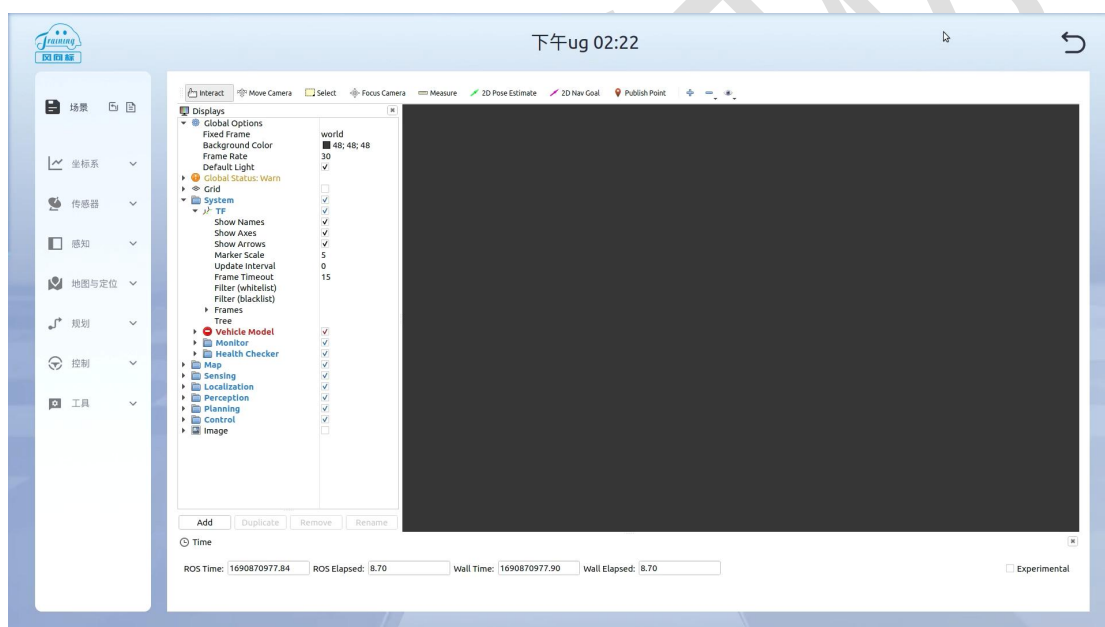
```
nvidia@oceanstar: ~
File Edit View Search Terminal Help
nvidia@oceanstar:~$ rosbag info test.bag
path:      test.bag
version:   2.0
duration:  3.6s
start:     May 21 2025 15:06:01.91 (1747811161.91)
end:       May 21 2025 15:06:05.52 (1747811165.52)
size:      34.7 MB
messages:  44
compression: none [36/36 chunks]
types:     rosgraph_msgs/Log [acffd30cd6b6de30f120938c17c593fb]
           sensor_msgs/PointCloud2 [1158d486dd51d683ce2f1be655c3c181]
topics:    /points_raw 36 msgs : sensor_msgs/PointCloud2
           /rosout 8 msgs : rosgraph_msgs/Log (2 connections)
```

3.3 PCD 点云地图制作

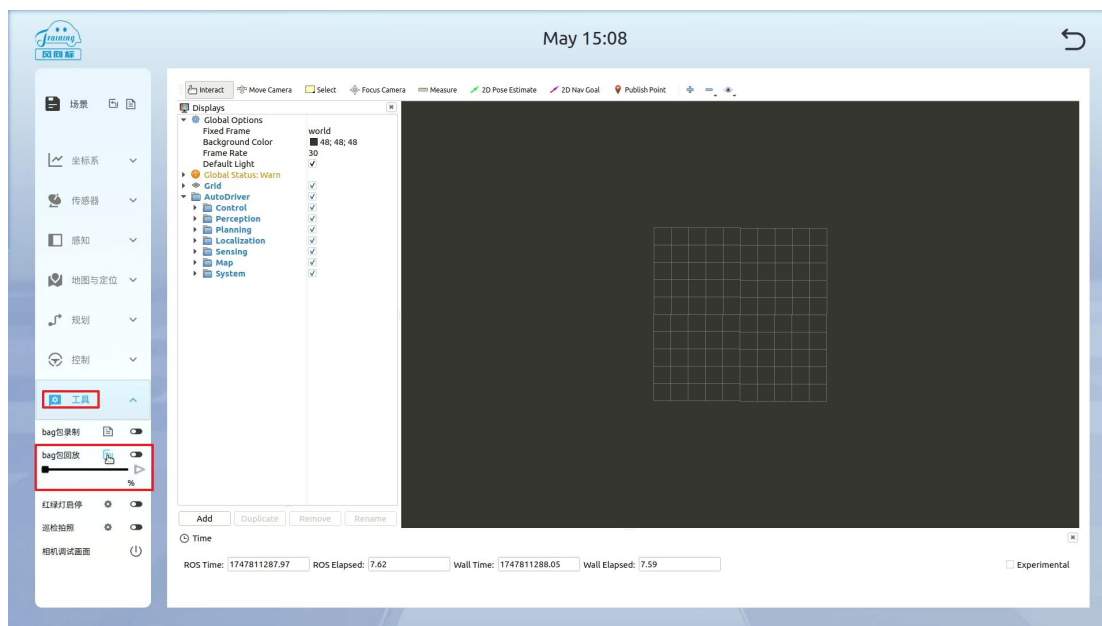
0) 在关闭所有软件和终端后再操作本小节。



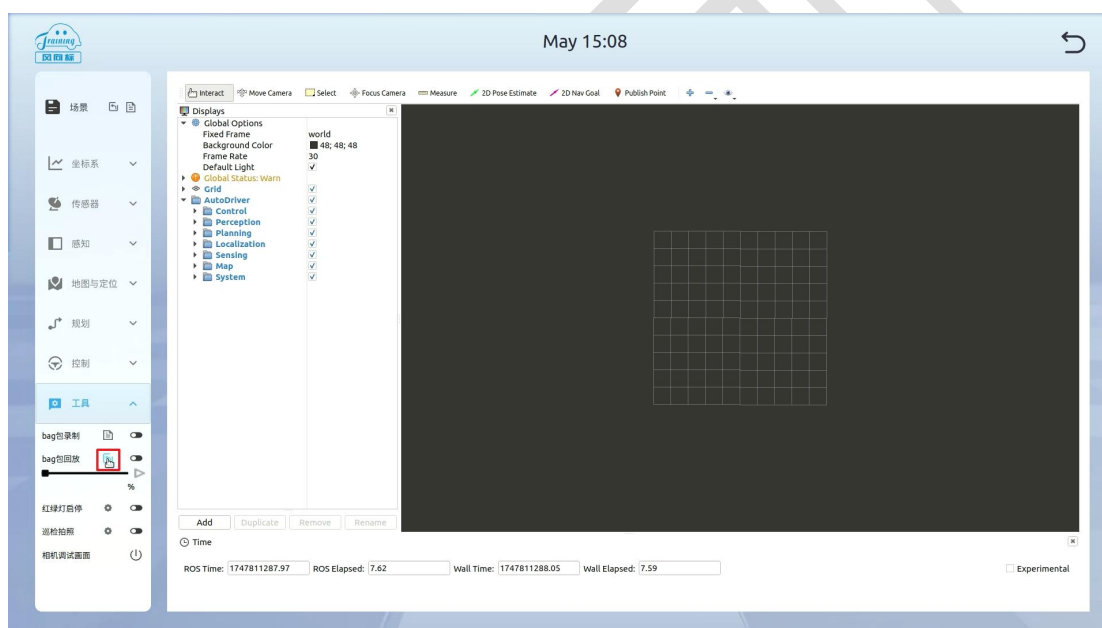
1) 打开自动驾驶教学软件，选择“高级教学”模式。

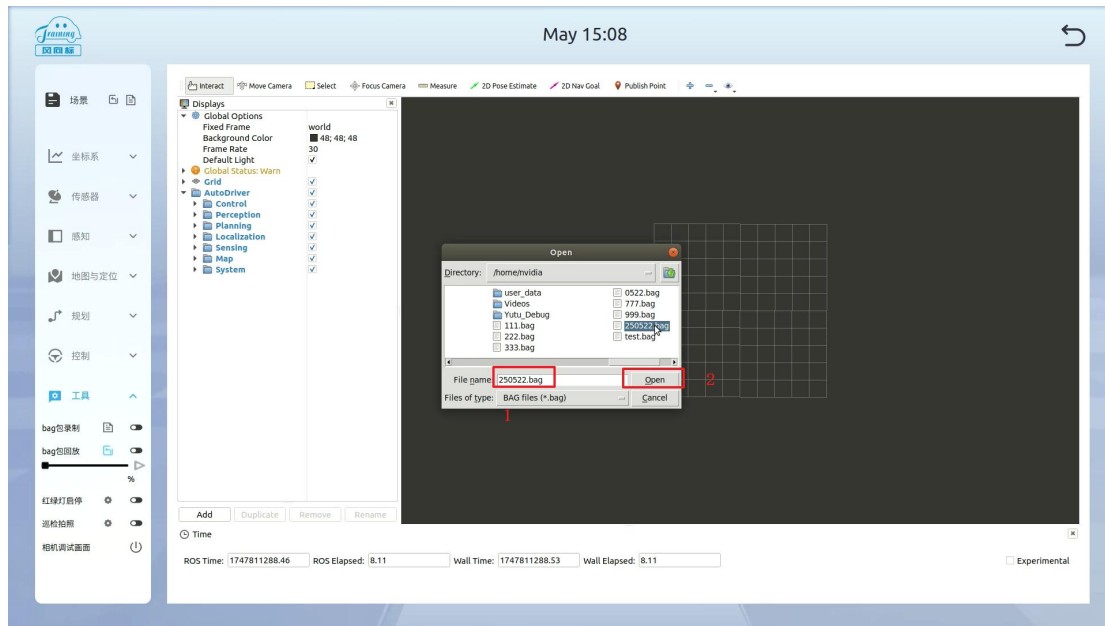


2) 双击“工具”，找到下面的“bag 包回放”功能。

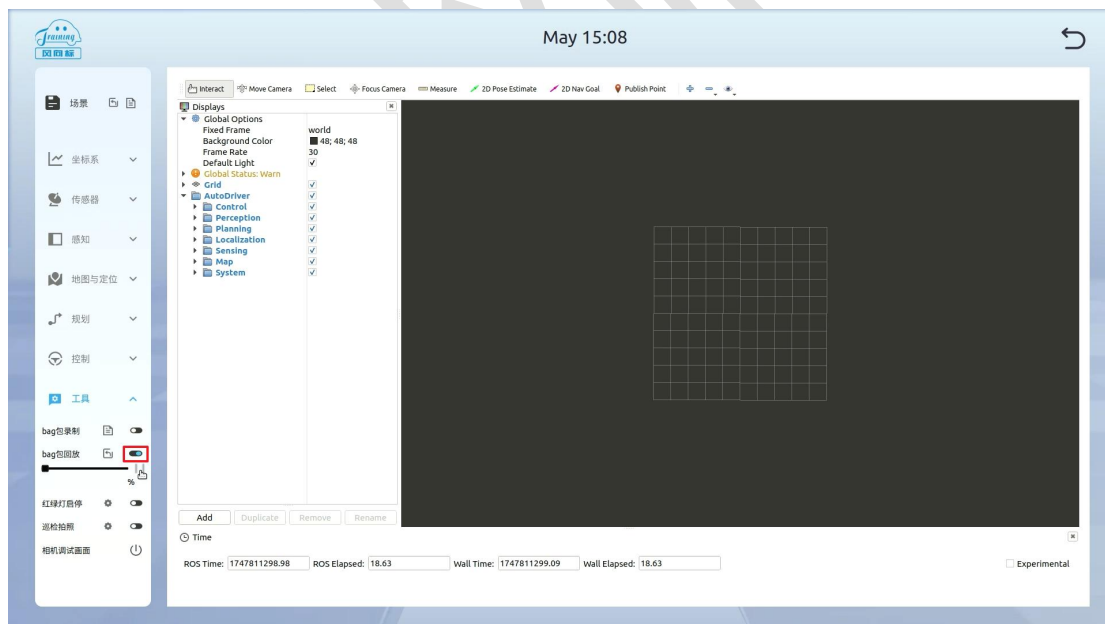


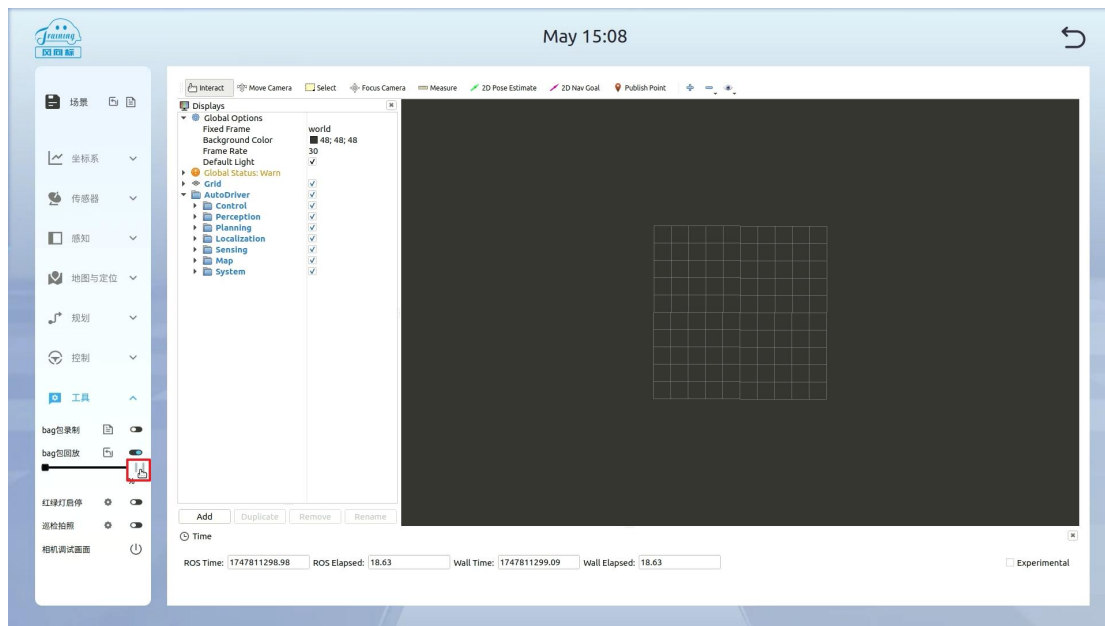
3) 点击导入 bag 按钮，选择预先采集好的传感器数据 bag 文件。



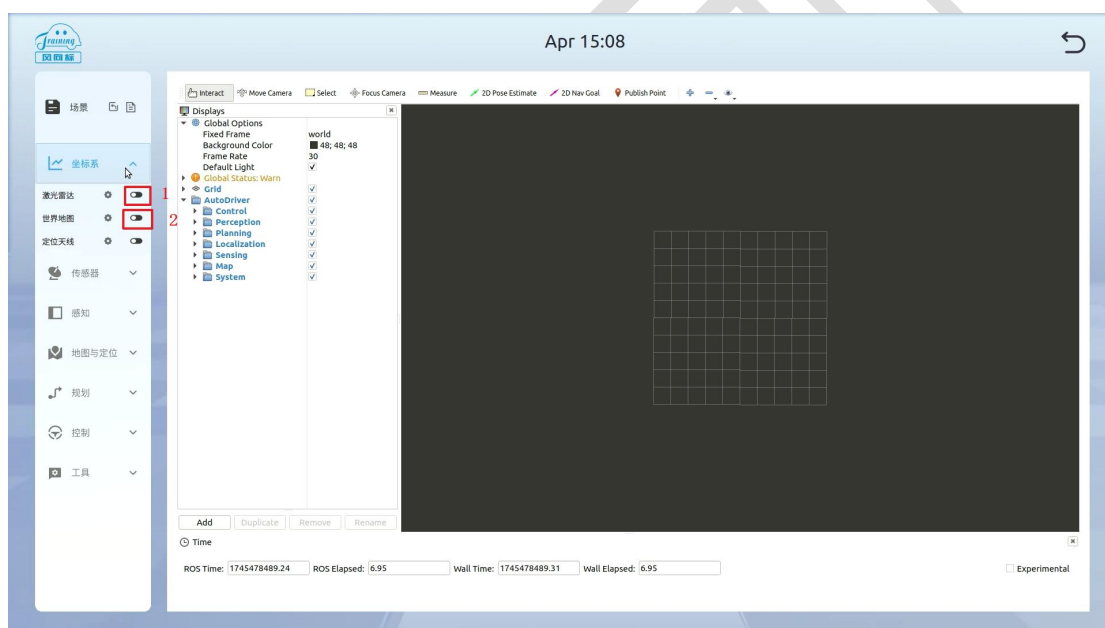


4) 点击启动 bag 包回放功能，当进度条 1%时（有进度就行，不必纠结必须 1%），点击进度条旁边的暂停按钮，此操作的目的是让一小部分传感器数据先发布，可以让接下来启动其他功能时可以获取到数据，不至于没有完全启动，而一直等待数据获取。

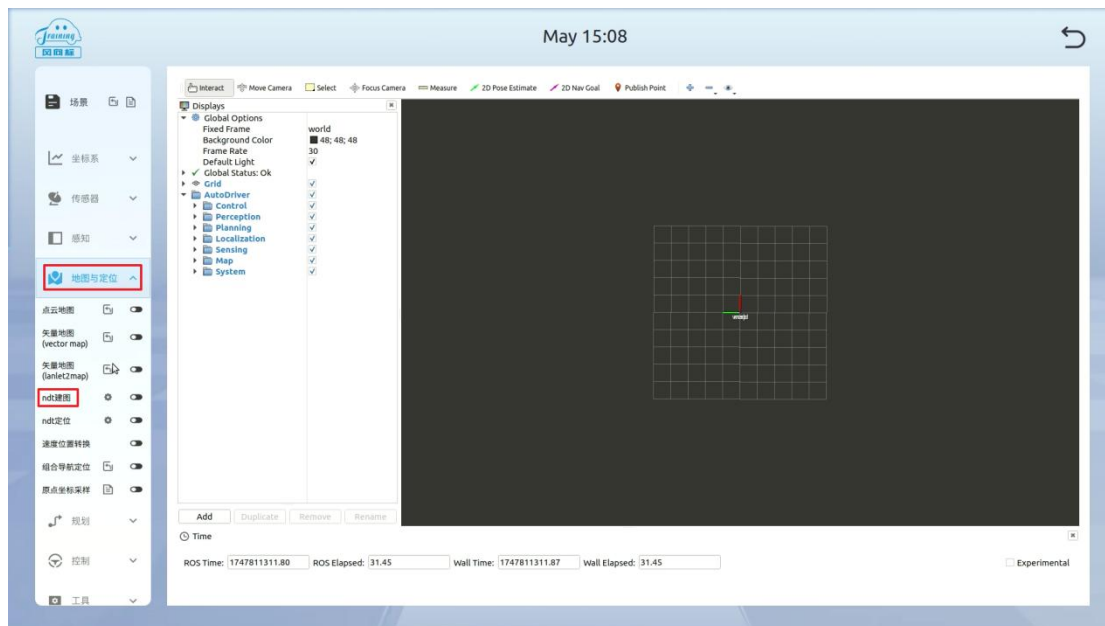




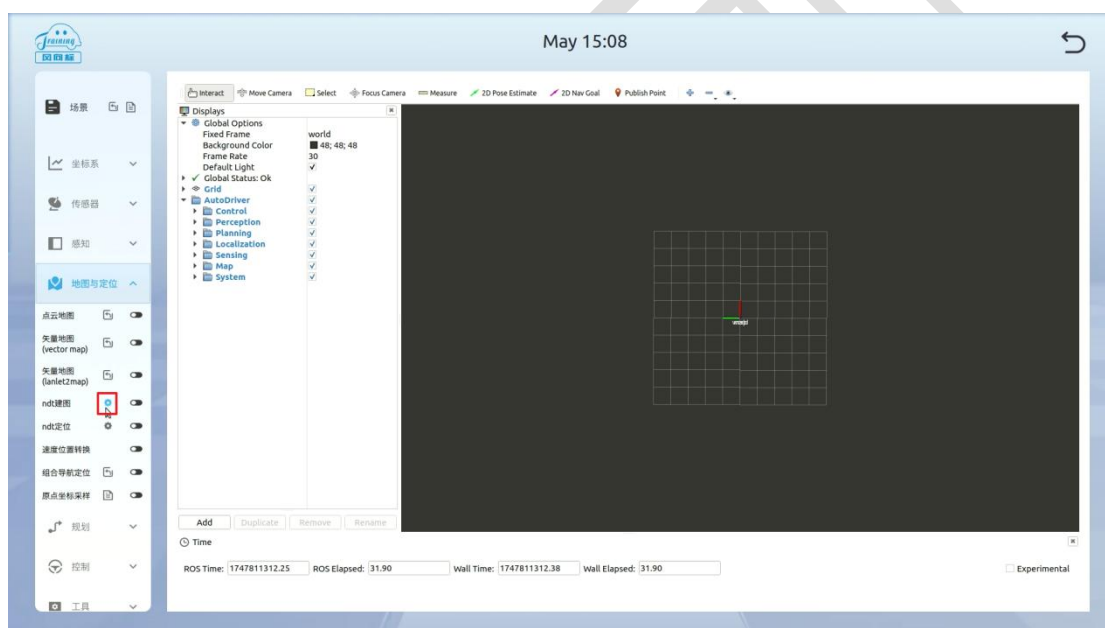
5) 接下来双击“坐标系”，勾选启动“激光雷达”和“世界地图”。

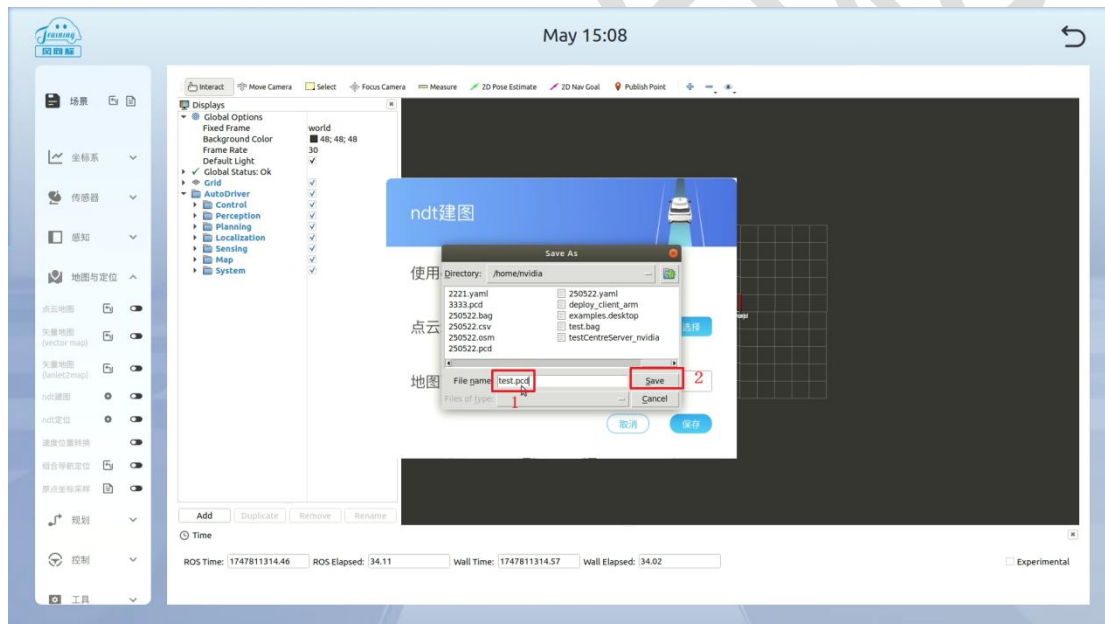
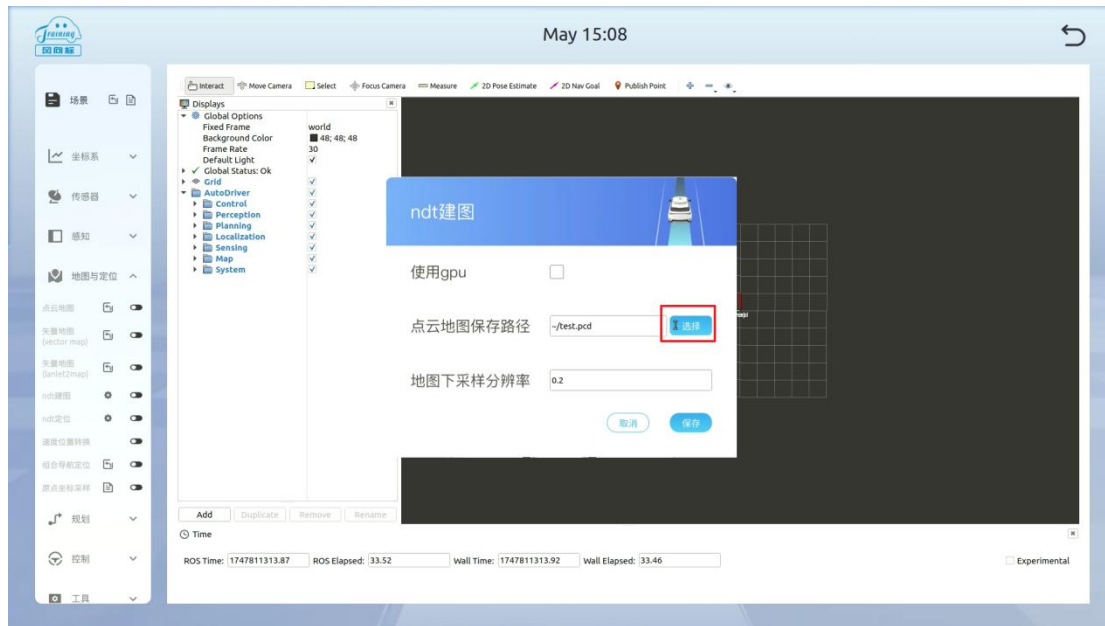


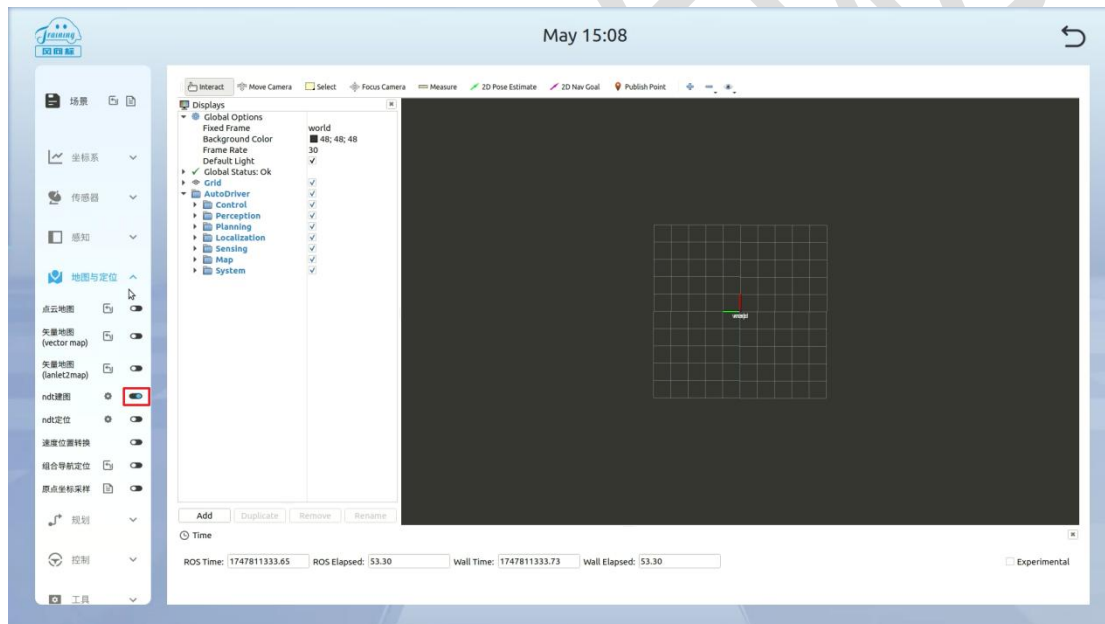
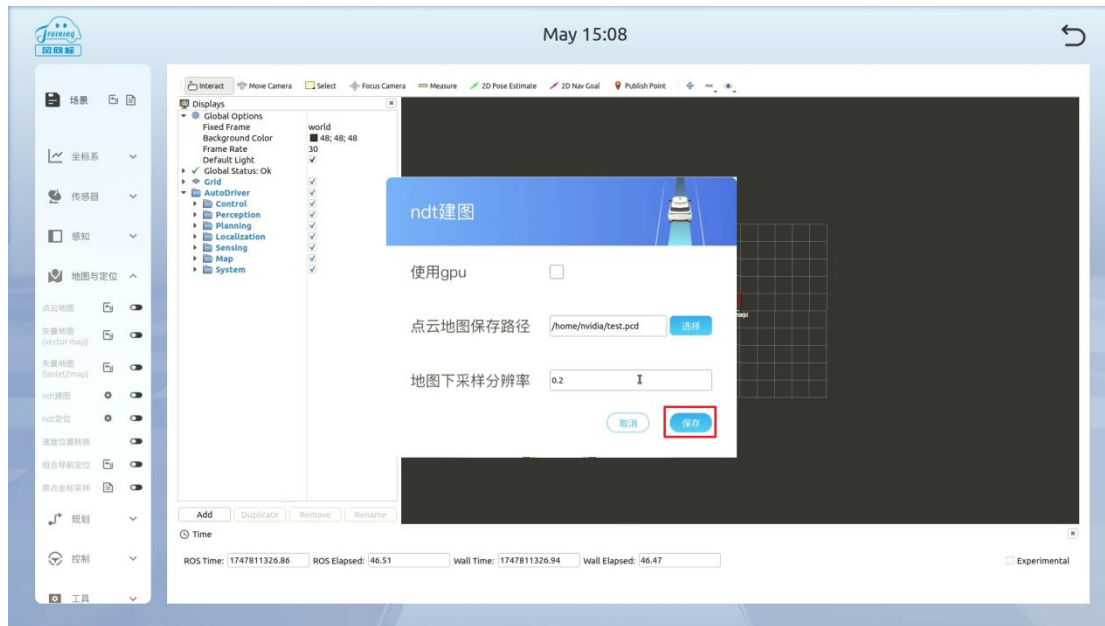
6) 接下来双击“地图与定位”，找到“ndt 建图”功能。



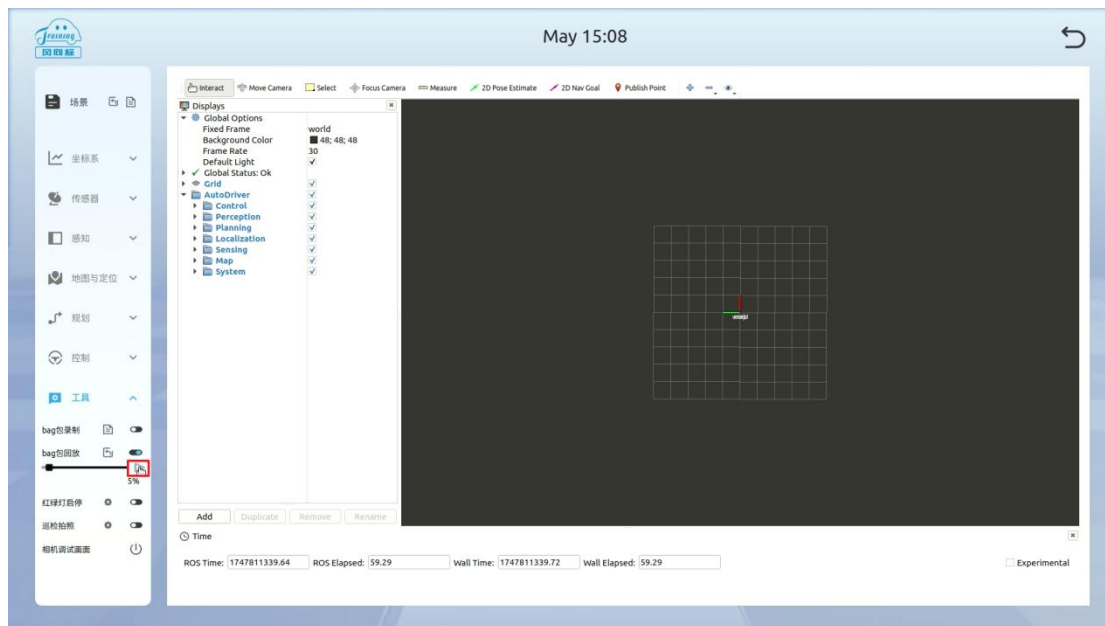
7) 点击配置按钮，填写 PCD 地图的保存路径以及保存文件名后勾选启动。



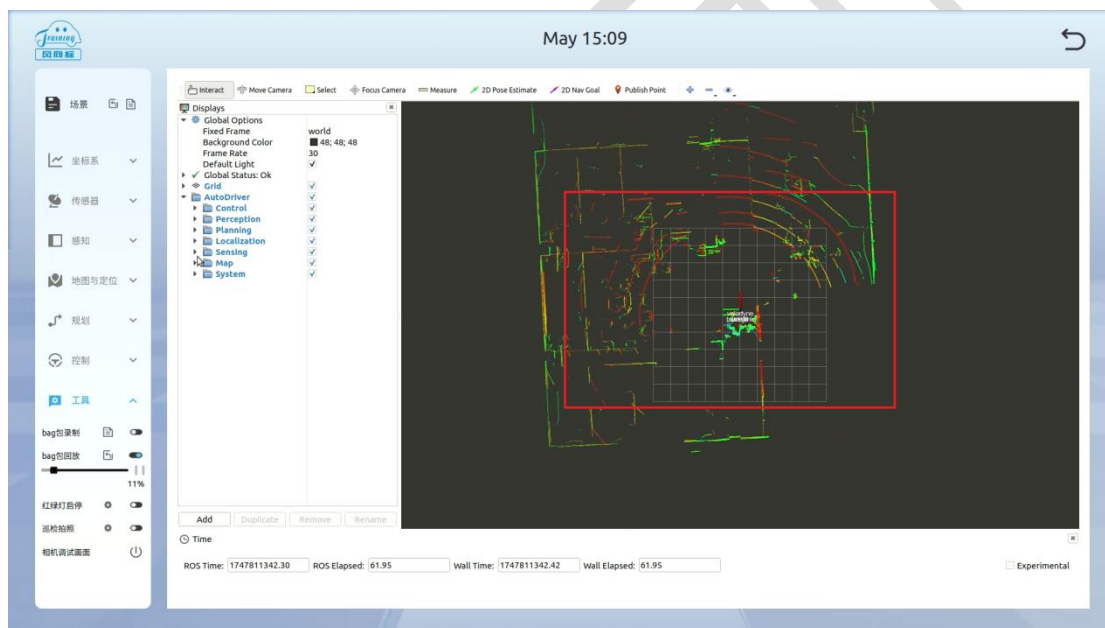




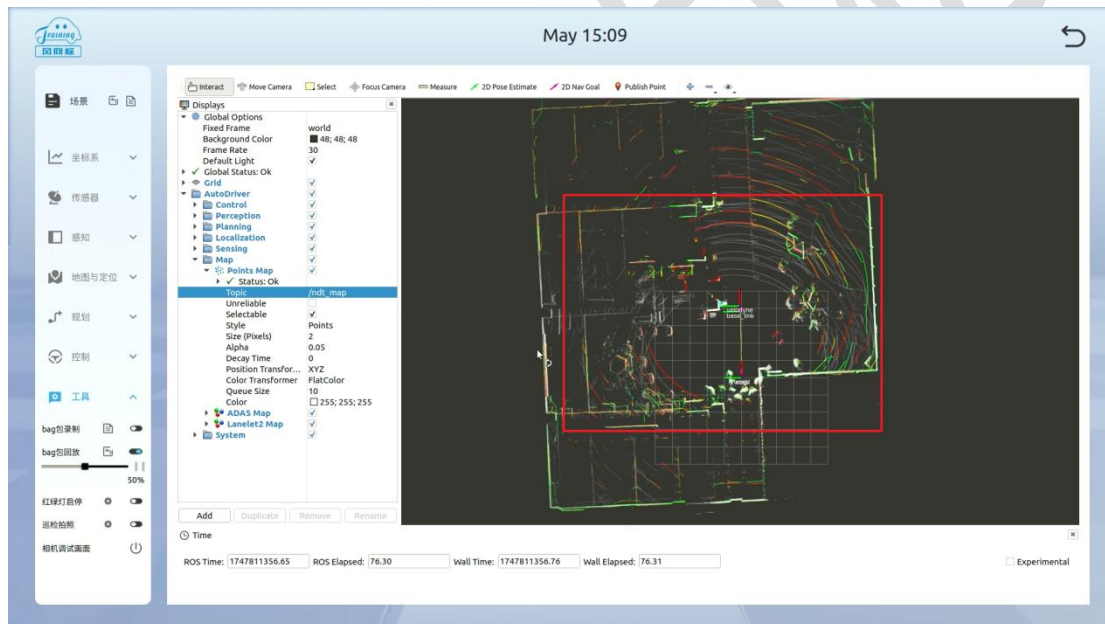
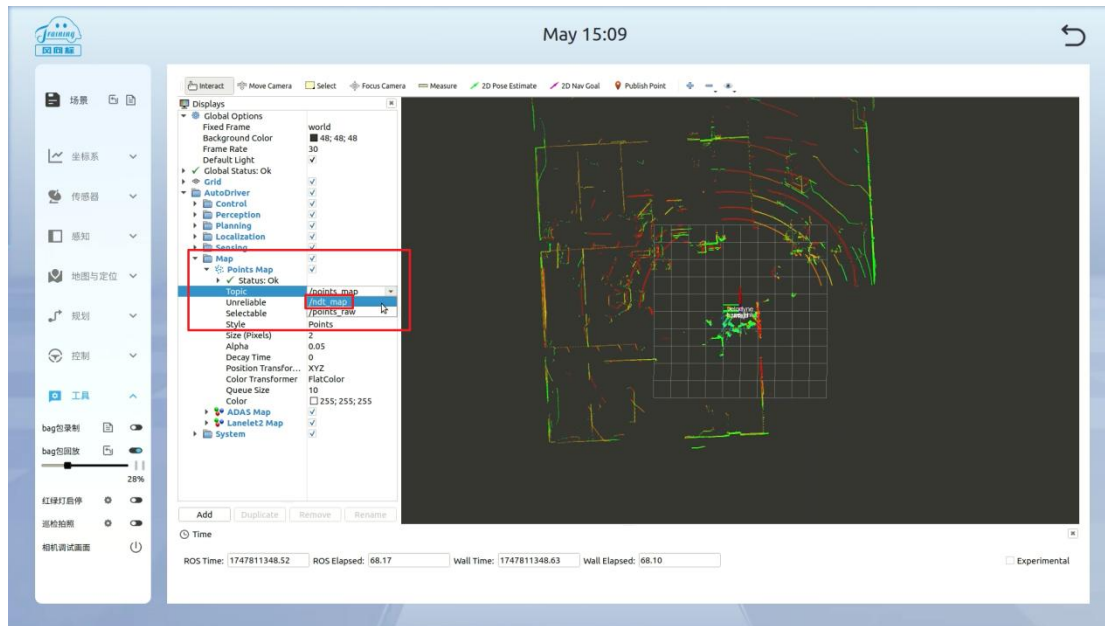
8) 接下来将继续播放传感器 bag 包。



点击播放后会看到有彩色的点云数据出现，但没有看到白色的点云地图。



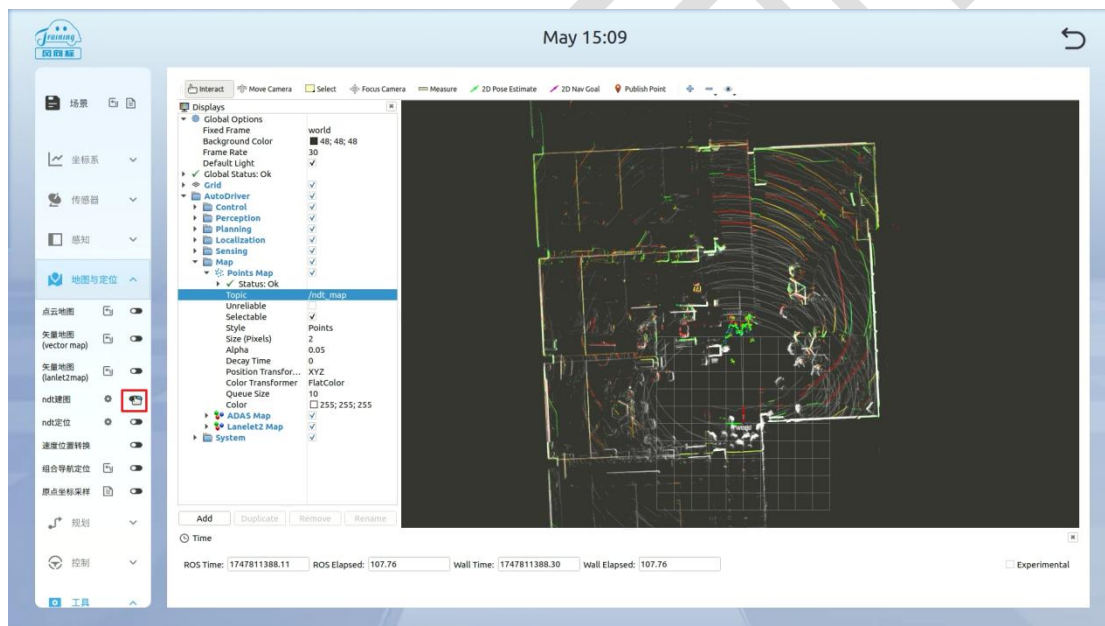
这是因为显示点云地图的插件没有选择正确的话题名，可以找到 Map 下的 Points Map 的 Topic，将/points_map 选择为/ndt_map。

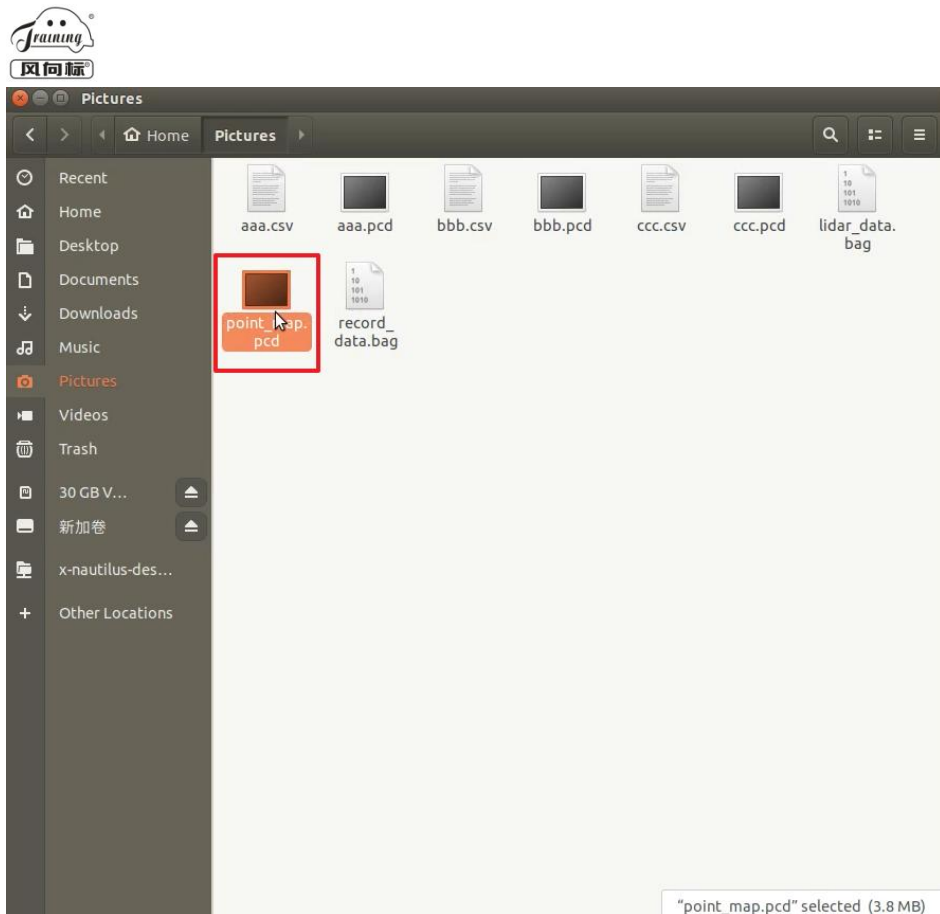


9) 按下键盘 win 键，查看建图输出终端，当终端没有新的日志刷新（不必纠结 **Processed** 和 **Input** 数字一致，有时会因为建图程序处理不过来，而导致一部分数据丢失，录制场景越大，所需花费的时间越长），则取消勾选按钮，等待 5~10s 后出现提示 pcd 地图保存成功，说明 pcd 地图已经创建成功。

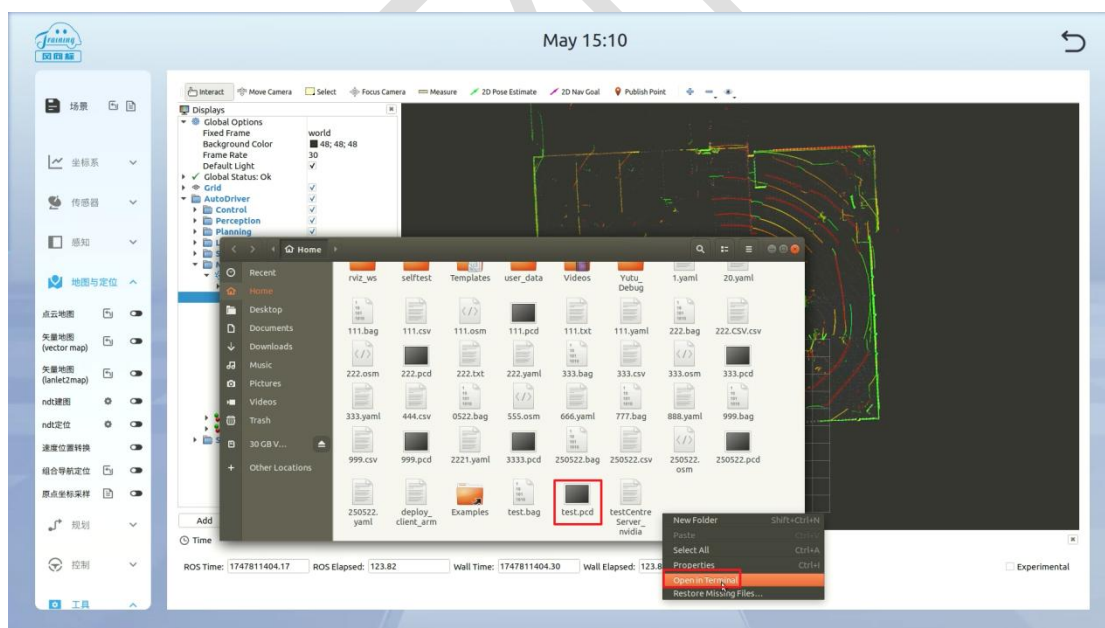
```

/home/nvidia/driver_ws/install/share/autoware_param_config/launch/pub_save_mapping_c...
File Edit View Search Terminal Help
0.000227143  0.00651194  0.999979  1.18511
              0          0          0          1
shift: 0.0689393
-----
(Processed/Input): (1403 / 1404)
-----
Sequence number: 4401
Number of scan points: 23831 points.
Number of filtered scan points: 3045 points.
transformed_scan_ptr: 23831 points.
map: 111772 points.
NDT has converged: 1
Fitness score: 0.0514902
Number of iteration: 2
(x,y,z,roll,pitch,yaw):
(2.40964, 0.0926649, -0.0506555, -7.87181e-05, -0.00643746, 0.0140558)
Transformation Matrix:
    0.014851    0.999869   -0.00643789    2.73165
   -0.99989    0.0148512  -1.17697e-05    0.0972885
   8.38424e-05  0.00643735    0.999979    1.18644
                0          0          0          1
shift: 0.060048
-----
(Processed/Input): (1404 / 1404)
  
```

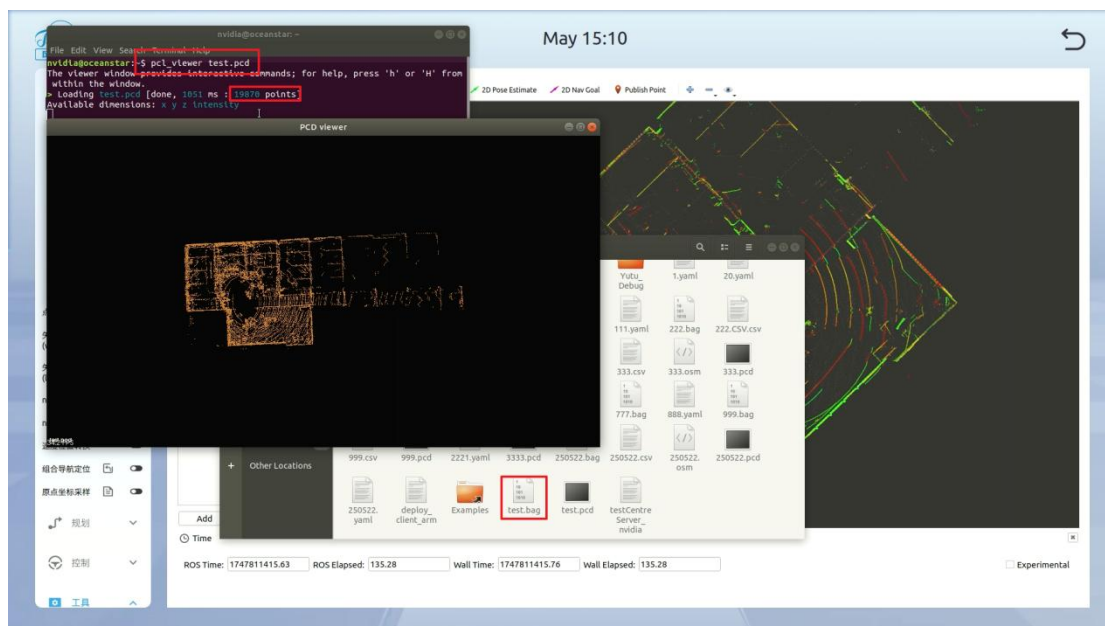




10) 在 PCD 文件夹的空白处，右键选择“Open in Terminal”打开终端。



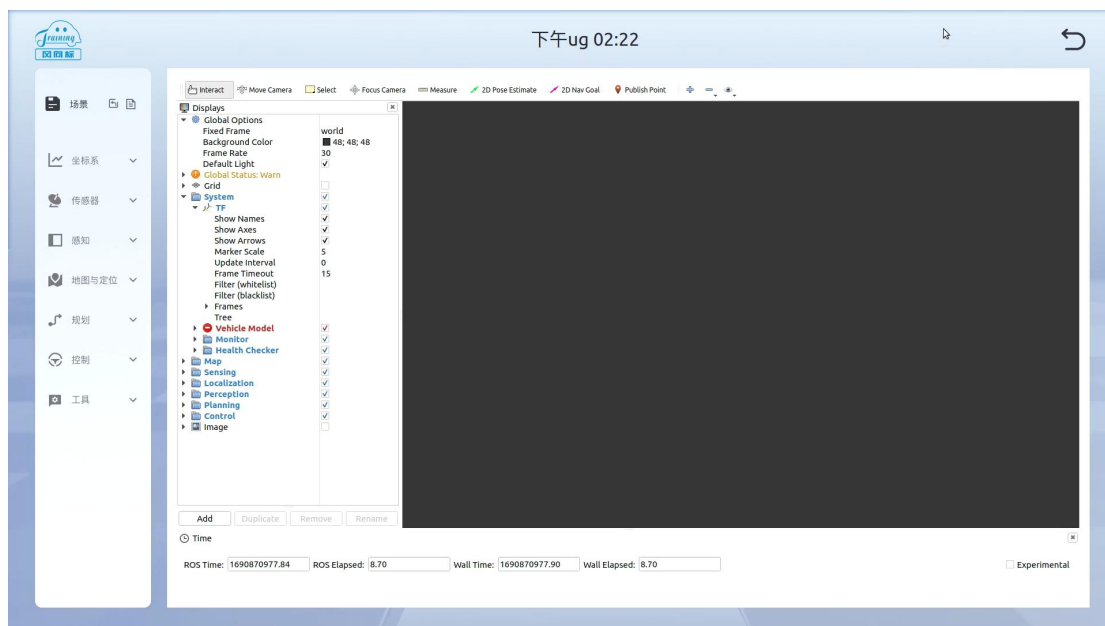
11) 输入命令 `pcl_viewer` 文件名，查看点云并获取点云点数。



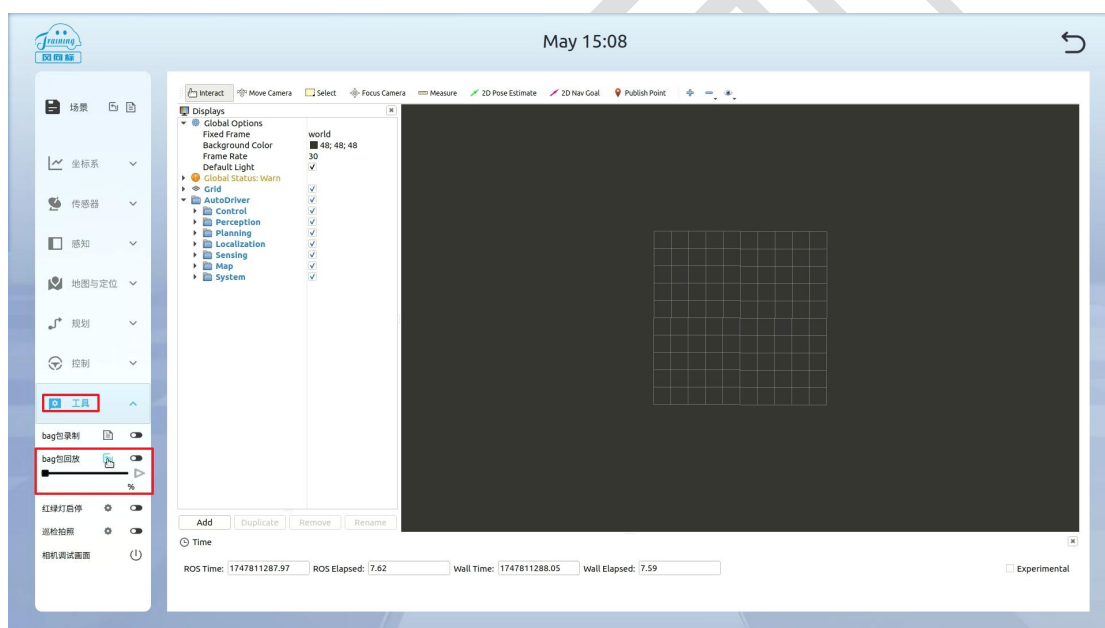
3.4 Waypoint 路径点制作

- 0) 在关闭所有软件和终端后再操作本小节。
- 1) 打开自动驾驶教学软件，选择“高级教学”模式。

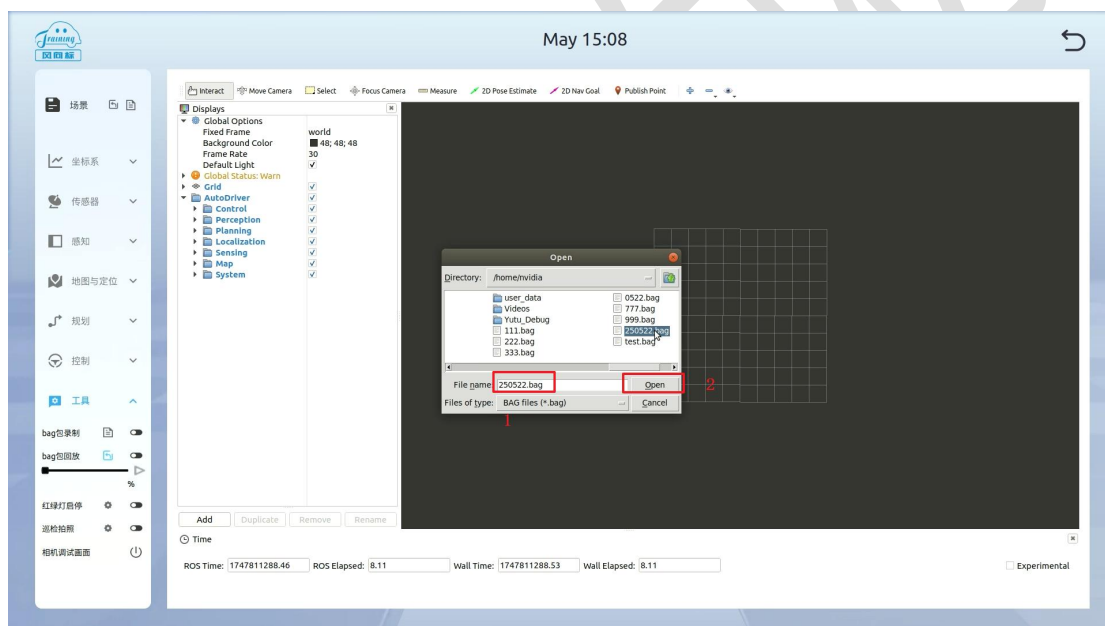
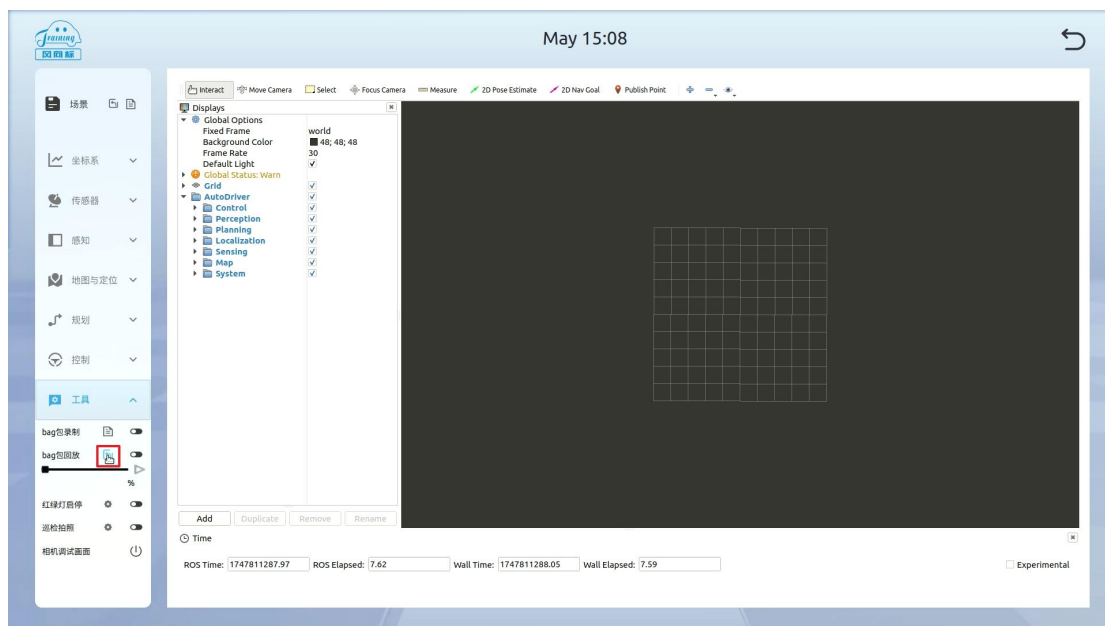




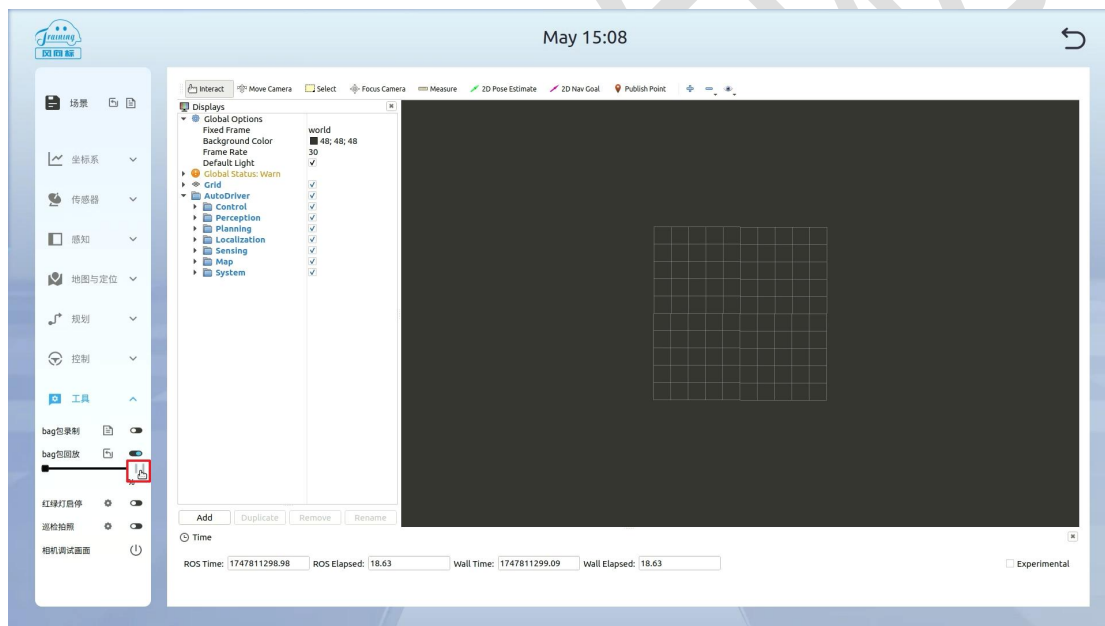
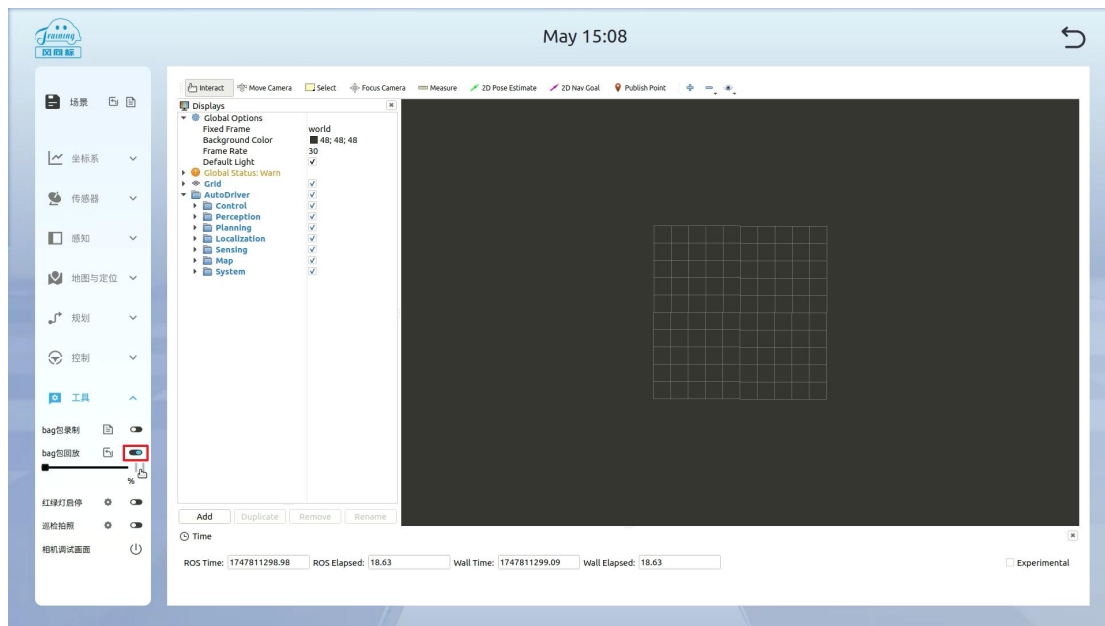
2) 双击“工具”，找到下面的“bag 包回放”功能。



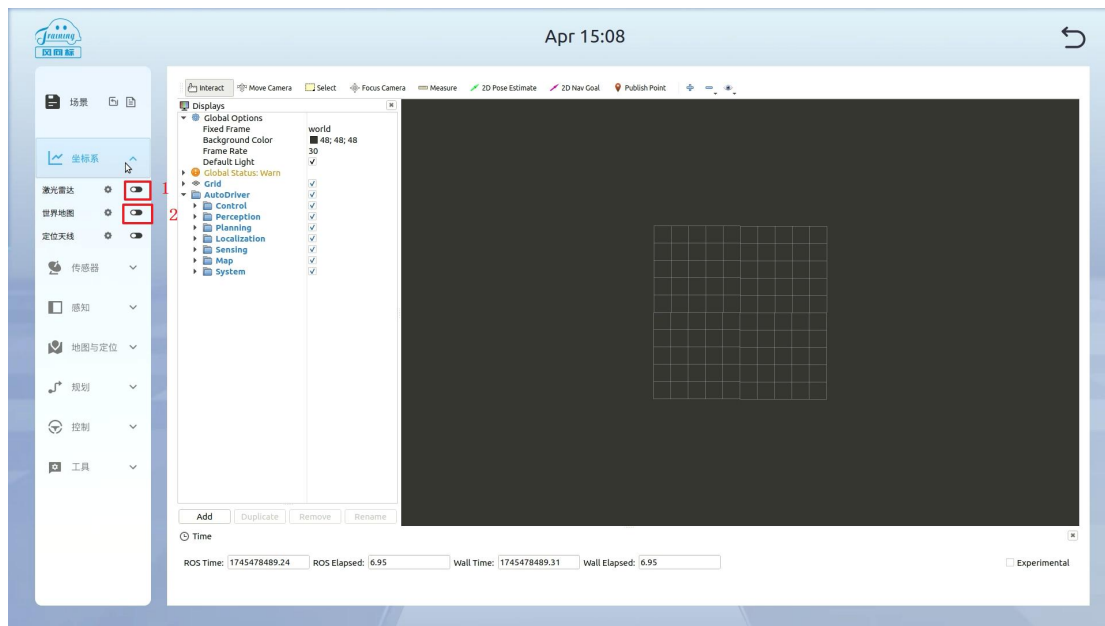
3) 点击导入 bag 按钮，选择预先采集好的传感器数据 bag 文件。



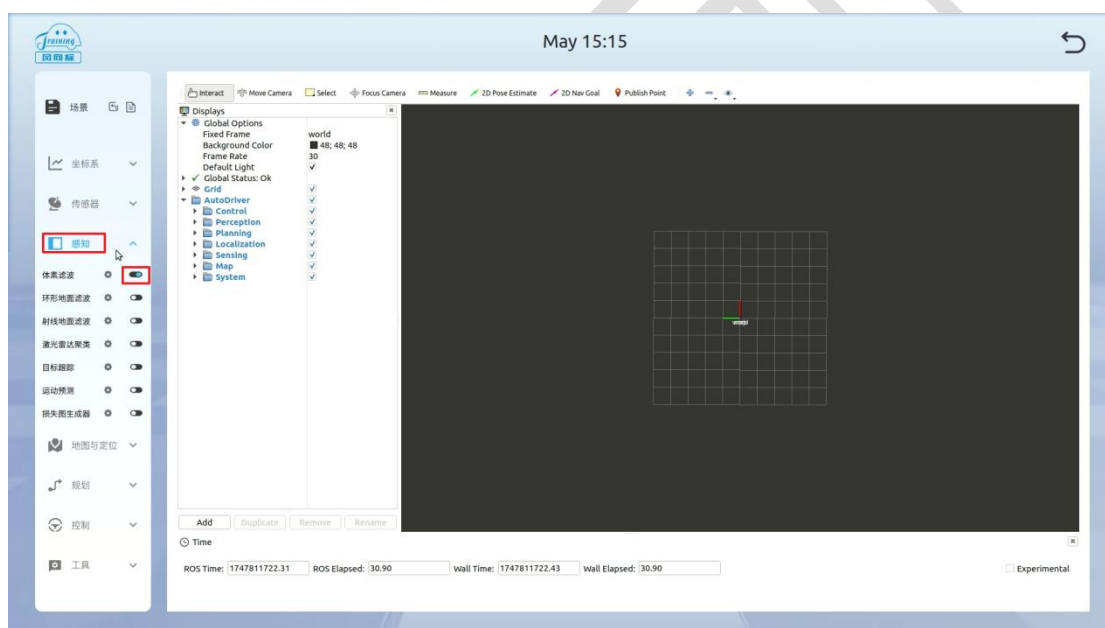
4) 点击启动 bag 包回放功能，当进度条 1%时（有进度就行，不必纠结必须 1%），点击进度条旁边的暂停按钮，此操作的目的是让一小部分传感器数据先发布，可以让接下来启动其他功能时可以获取到数据，不至于没有完全启动，而一直等待数据获取。



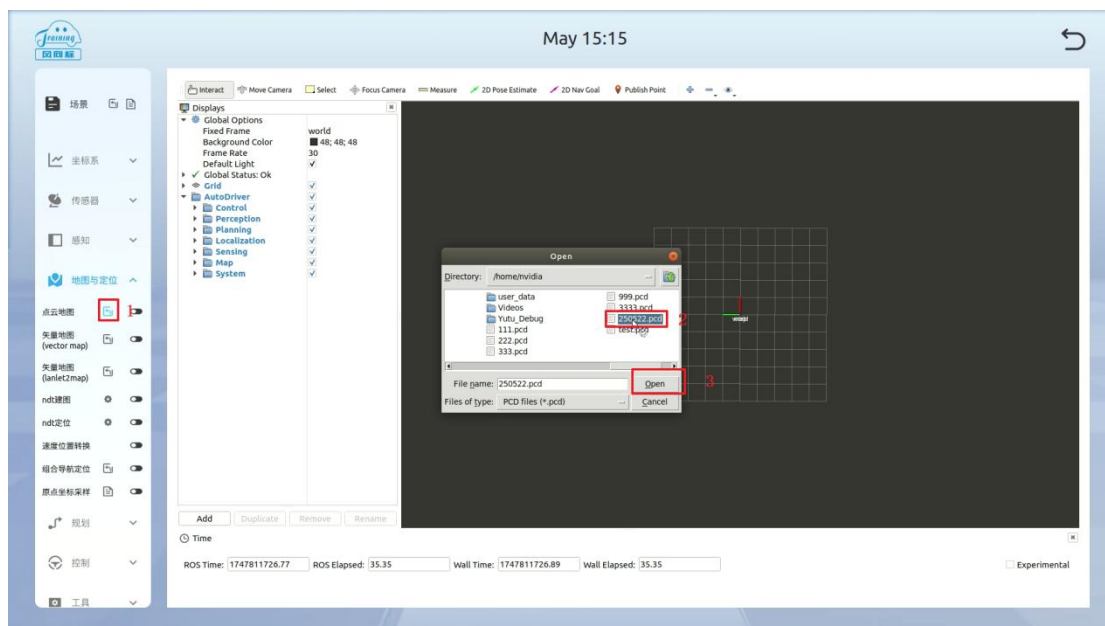
5) 接下来双击“坐标系”，勾选启动“激光雷达”和“世界地图”。



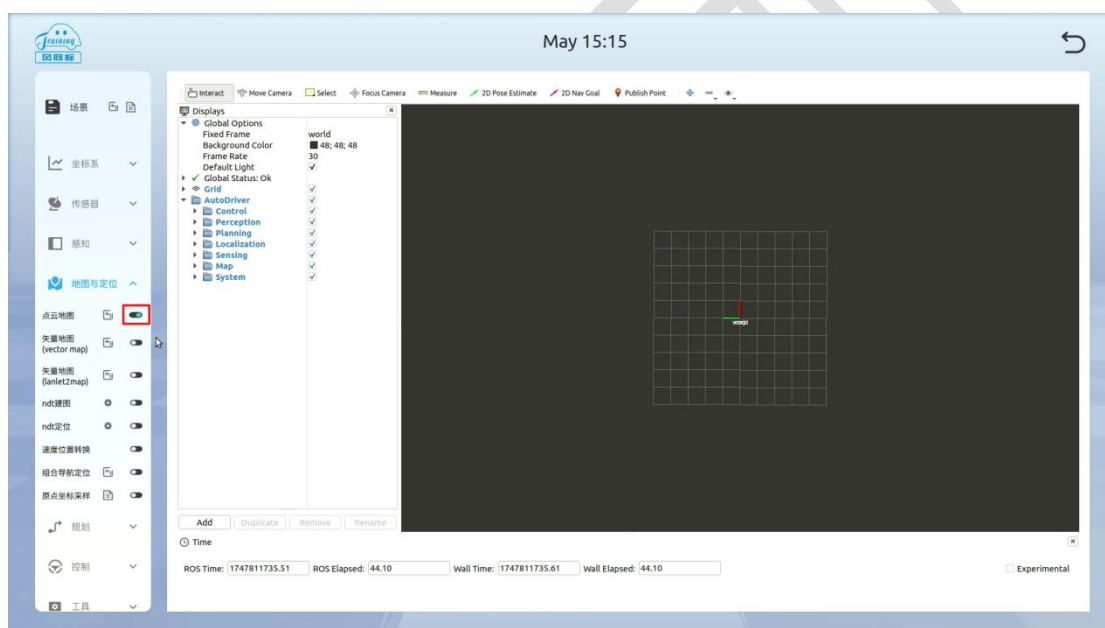
6) 双击“感知”，勾选启动“体素滤波”。



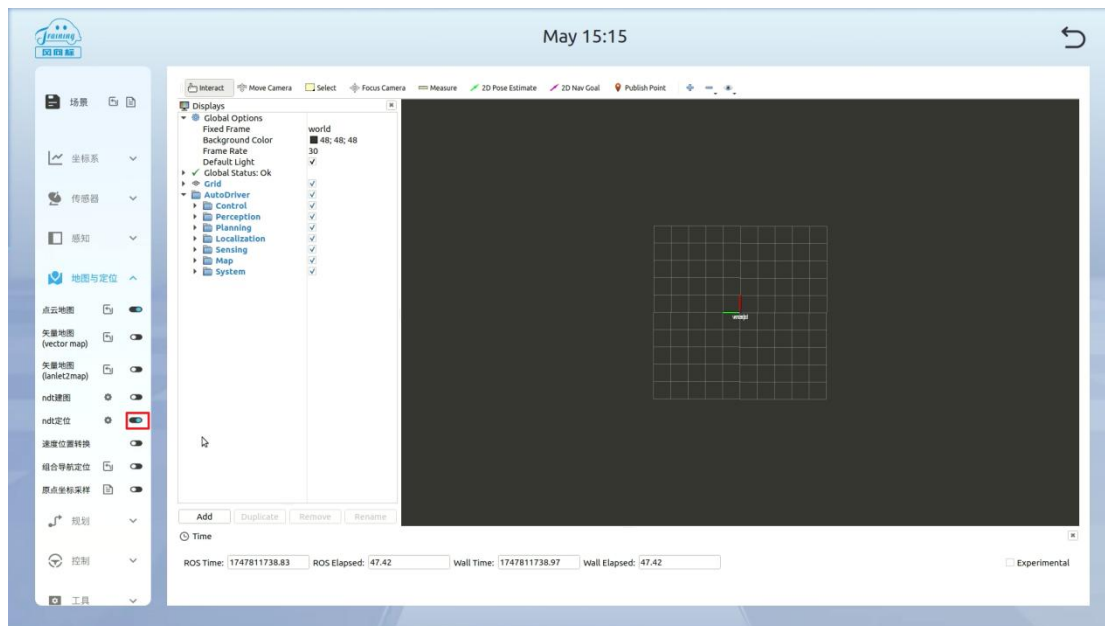
7) 双击“地图与定位”，找到“点云地图”，选择 PCD 点云地图文件后。



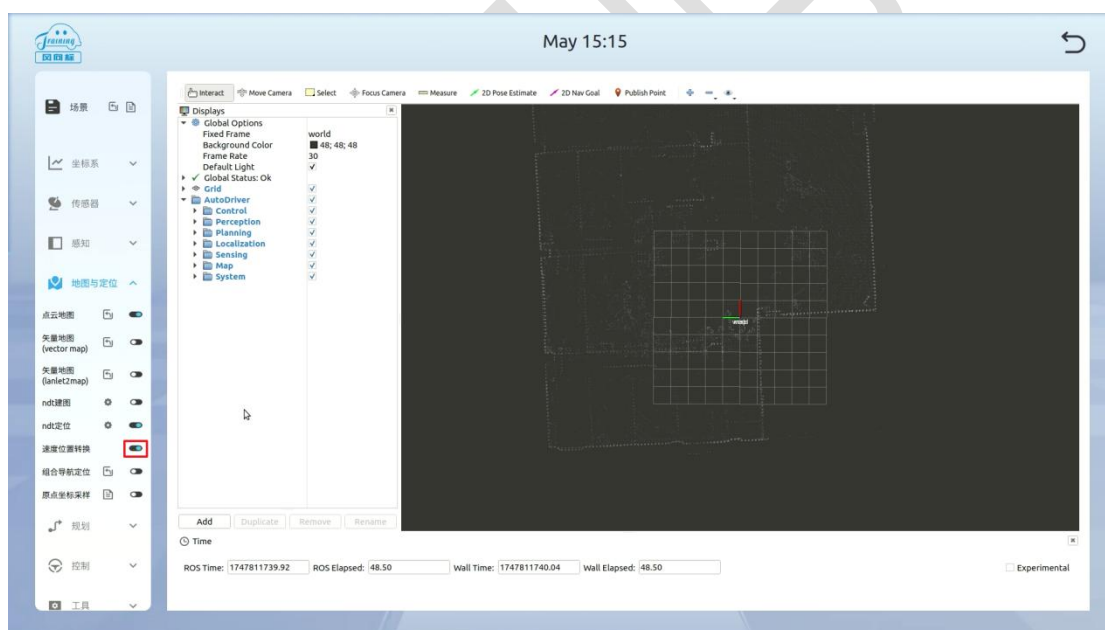
8) 接下来勾选启动“点云地图”功能。



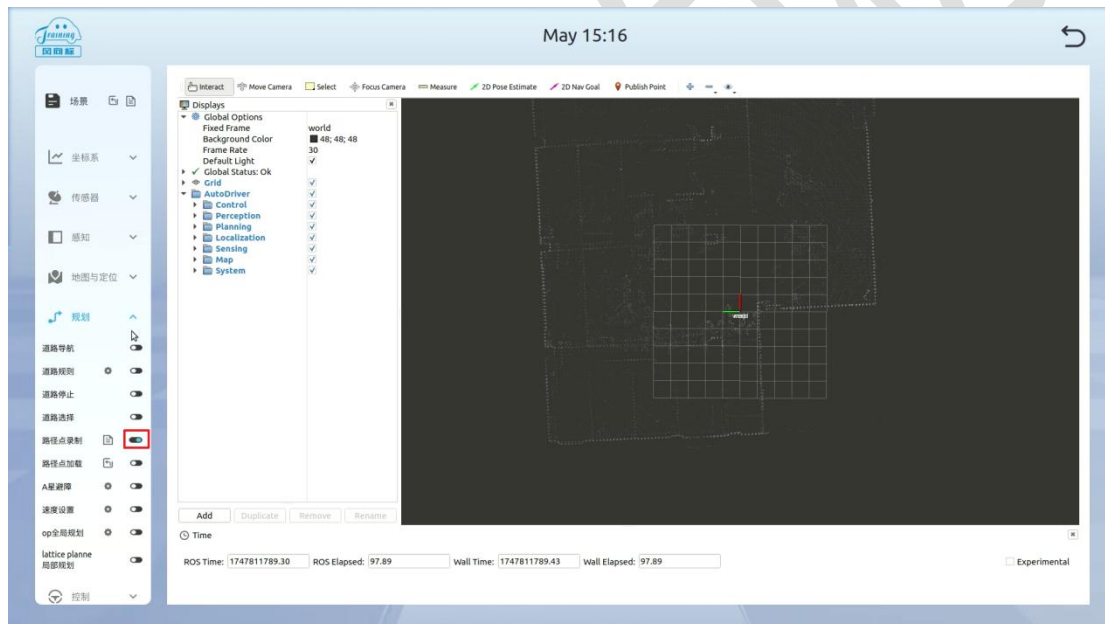
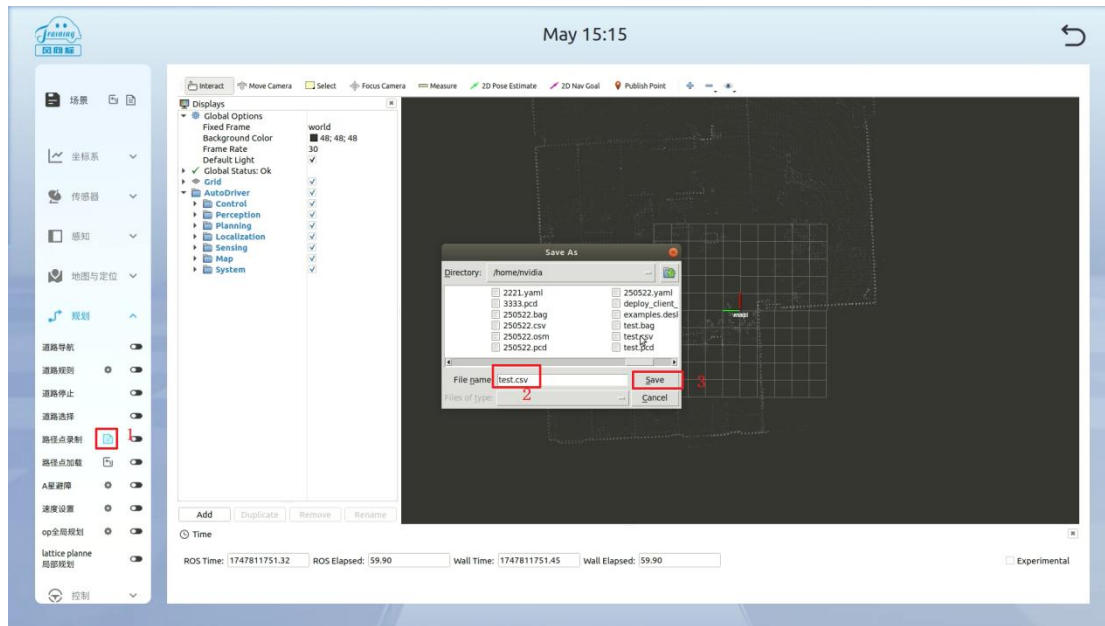
9) 接下来双击“地图与定位”，找到“ndt 定位”并勾选启动。



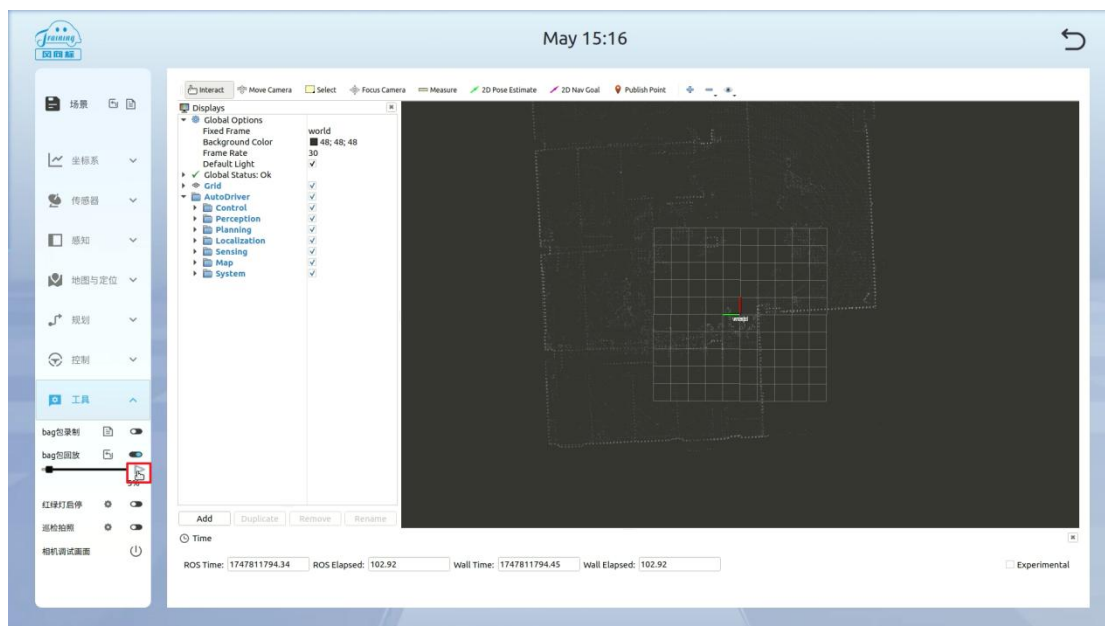
10) 接下来勾选启动“速度位置转换”功能，此功能作用是将 **ndt** 定位输出的速度和位置信息的话题名进行转换，因为接下来的路径点录制需要订阅该话题名。



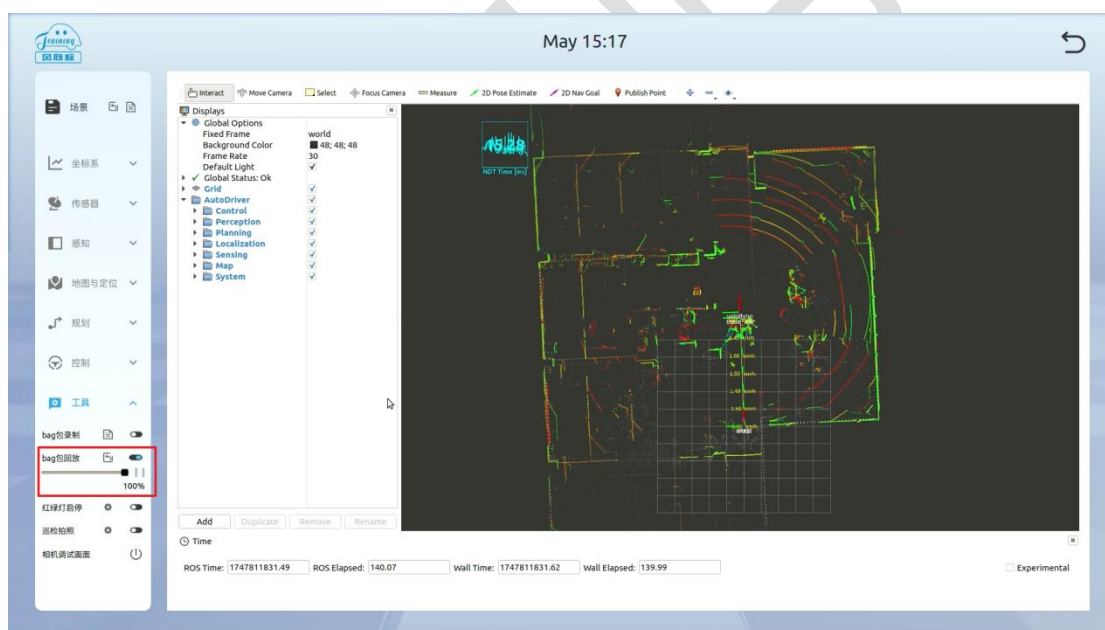
11) 双击“规划”，找到“路径点录制”，选择保存路径和文件名并勾选启动。

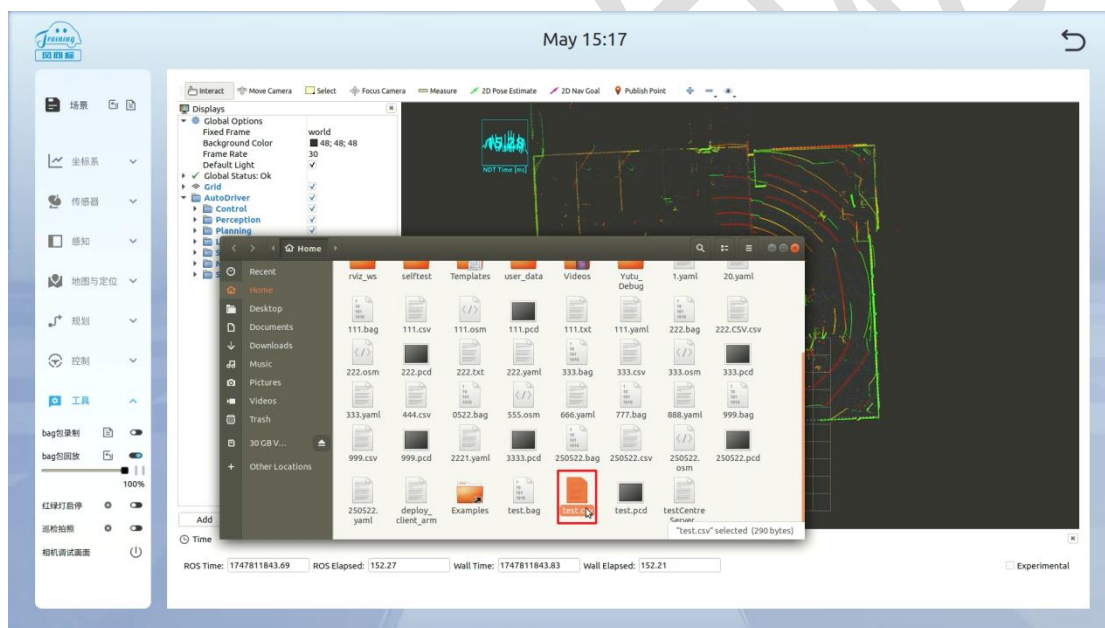
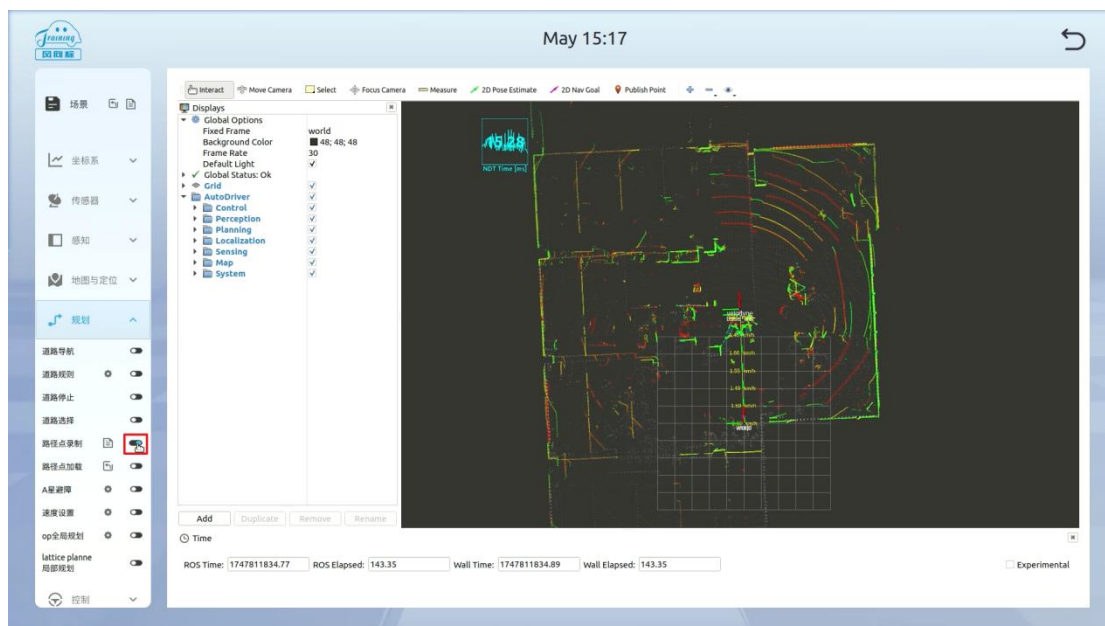


12) 最后取消暂停播放 bag。

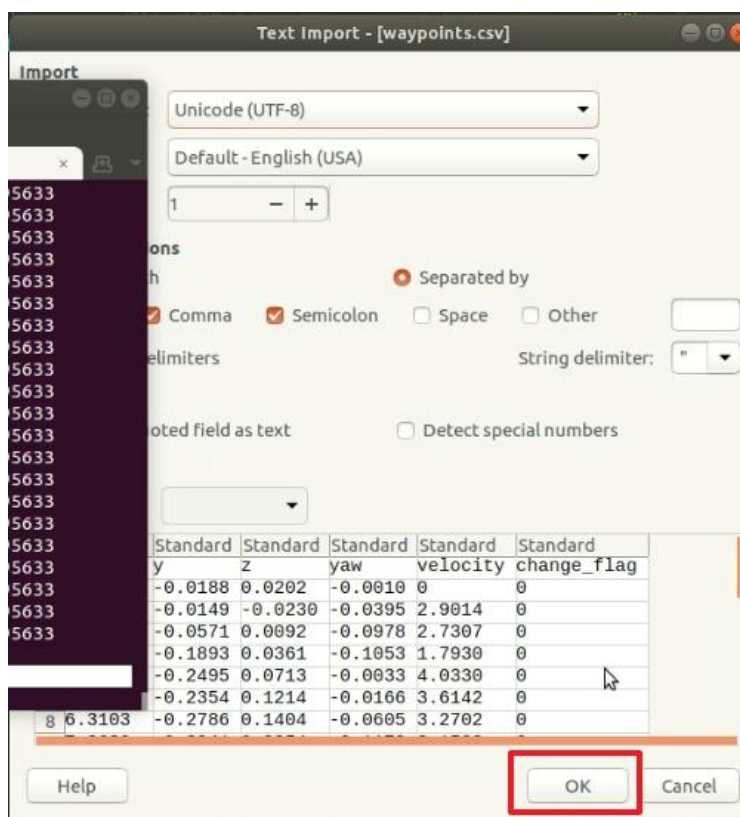
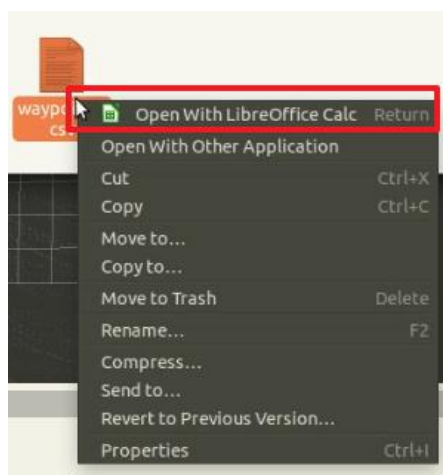


13) 当播放进度条到达 100%时或者认为路径已经完成，可以取消勾选路径点录制，路径点文件会保存到填写的指定路径中。





•



15) 将路径点的速度设置成相同速度，刚开始调试推荐设置为 2，等调试稳定后再慢慢增加速度，最大不要超过 5，最小为 1（速度单位 km/h）。

Activities LibreOffice Calc

File Edit View Insert Format Styles Sheet Data Tools Window Help

LibreOffice Calc

E3:E28

	A	B	C	D	E	F	G	H
1	x	y	z	yaw	velocity	change_flag		
2	-0.005	-0.0188	0.0202	-0.001	2	0		
3	1.0285	-0.0149	-0.023	-0.0395	2	0		
4	2.0682	-0.0571	0.0092	-0.0978	2	0		
5	3.081	-0.1893	0.0361	-0.1053	2	0		
6	4.1194	-0.2495	0.0713	-0.0038	2	0		
7	5.1944	-0.2354	0.1214	-0.0165	2	0		
8	6.3103	-0.2786	0.1404	-0.0605	2	0		
9	7.363	-0.3941	0.2254	-0.1179	2	0		
10	8.3731	-0.4666	0.2605	-0.0285	2	0		
11	9.4236	-0.4512	0.2431	0.0179	2	0		
12	10.5268	-0.4078	0.2793	0.0193	2	0		
13	11.5585	-0.435	0.2887	-0.0717	2	0		
14	12.6166	-0.5054	0.3422	-0.0597	2	0		
15	13.6218	-0.5429	0.3621	-0.0432	2	0		
16	14.6883	-0.5725	0.3098	-0.0048	2	0		
17	15.7618	-0.5618	0.3979	0.0052	2	0		
18	16.8076	-0.548	0.4166	0.0039	2	0		
19	17.8363	-0.5211	0.4204	0.0165	2	0		
20	18.8946	-0.4975	0.4465	0.0012	2	0		
21	19.9067	-0.52	0.4868	-0.0368	2	0		
22	20.9348	-0.5492	0.4747	-0.08	2	0		
23	21.9917	-0.6411	0.5111	-0.0308	2	0		
24	23.0411	-0.6452	0.5487	-0.0162	2	0		
25	24.0758	-0.6754	0.5716	-0.0457	2	0		
26	25.1277	-0.7227	0.6007	-0.0578	2	0		
27	26.1495	-0.7928	0.632	-0.1068	2	0		
28	27.202	-0.8725	0.6373	-0.035	2	0		
29								

16) 使用快捷键 `ctrl + s` 保存，保存时选择“Use Text CSV Format”。



3.5 绘制 HD 高精地图

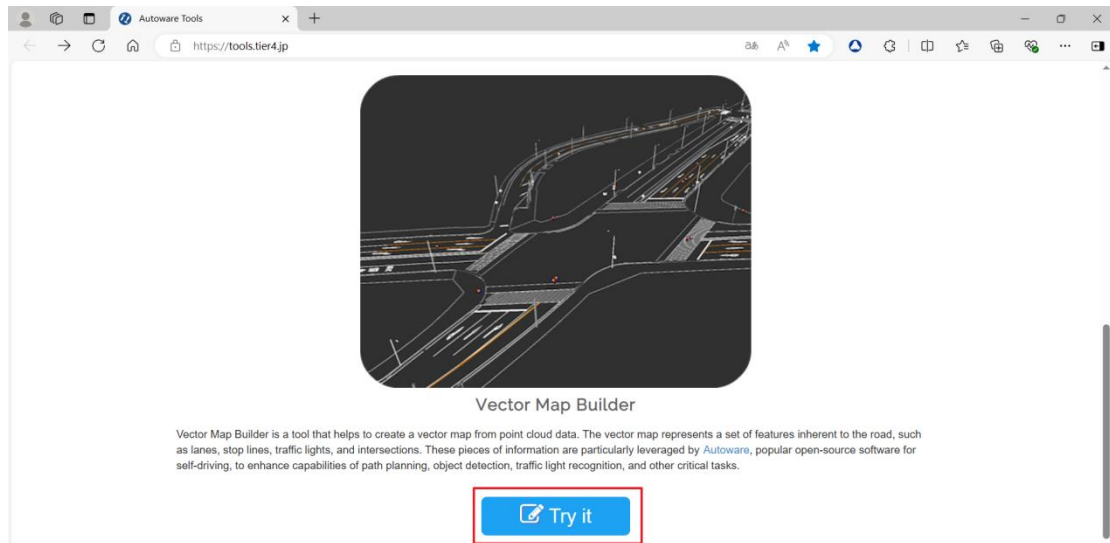
注！本小节需要到其他电脑进行操作。

需要注册一个 TIER IV 的账号。

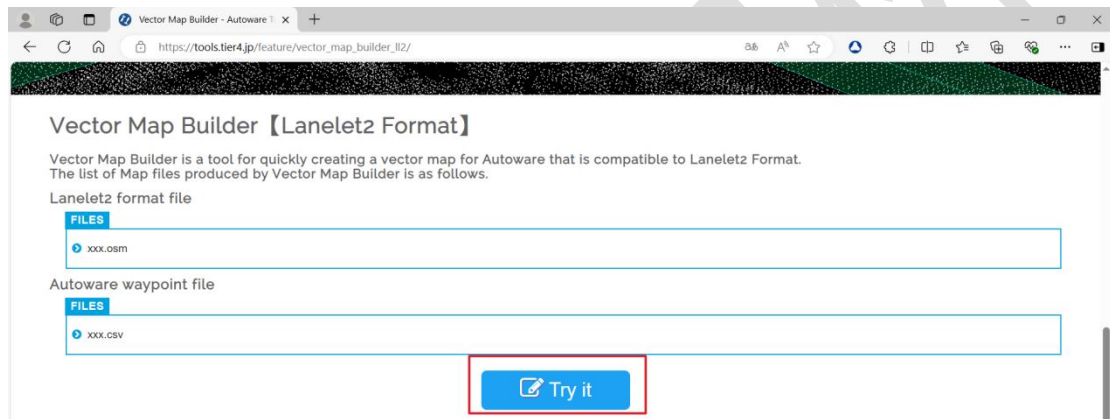
注册账号网站: <https://account.tier4.jp/registration>

在线绘制地图的网站: <https://tools.tier4.jp/>

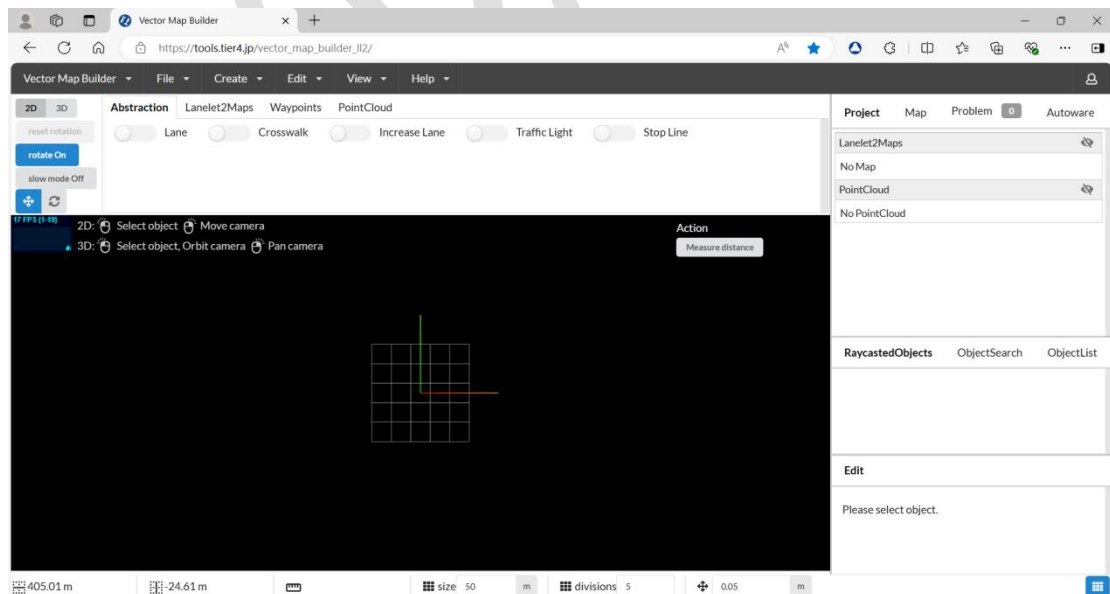
使用 U 盘将 3.3 和 3.4 小节制作好的 PCD 文件和 CSV 文件拷贝到画图电脑。



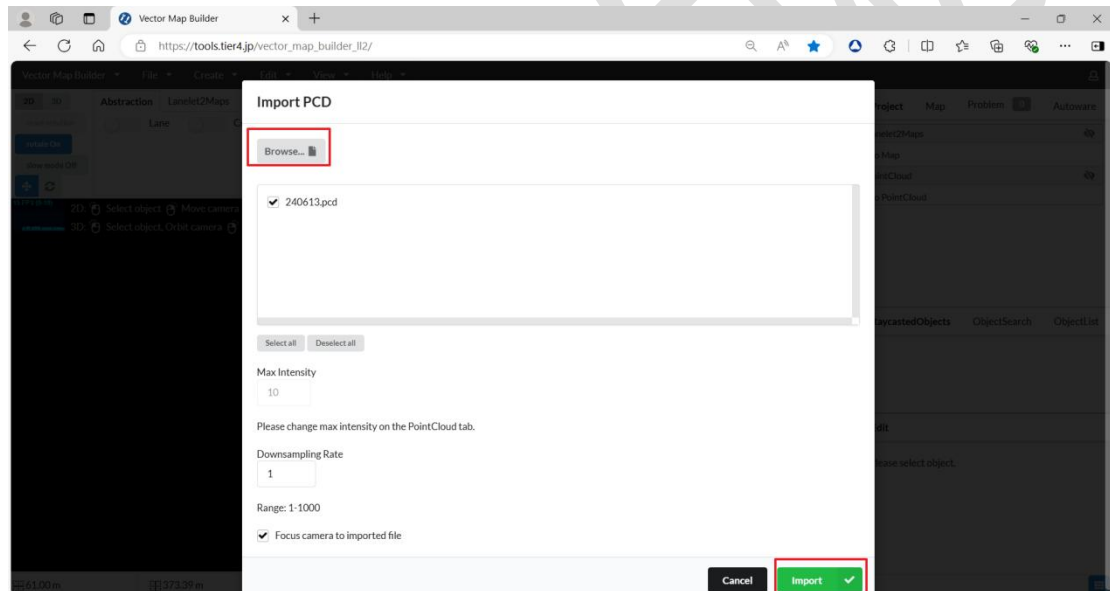
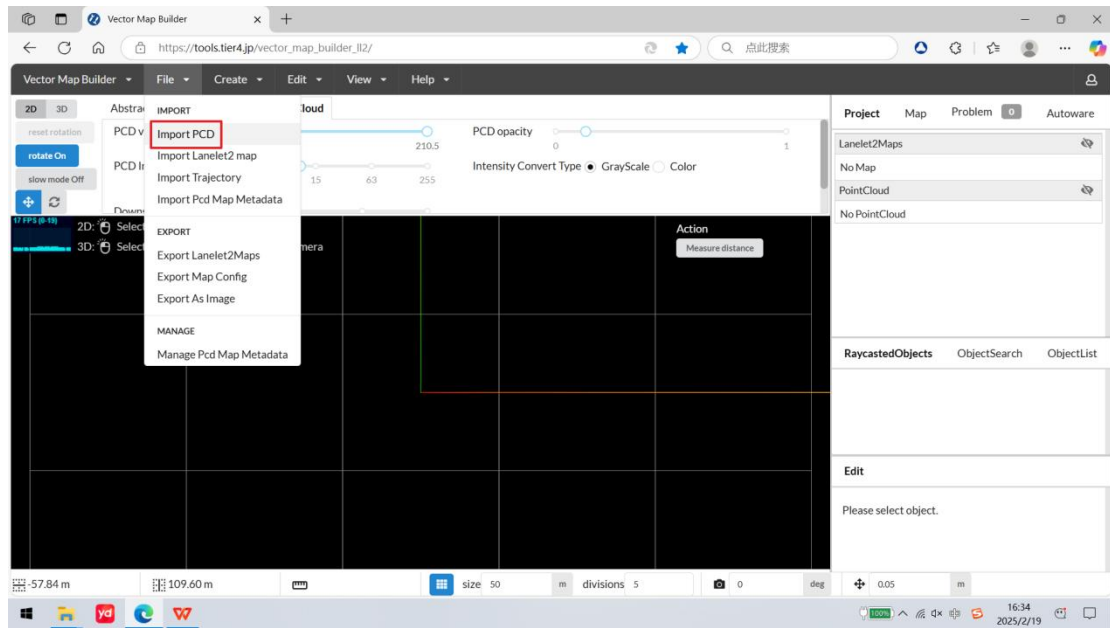
点击完成后会出现以下页面。

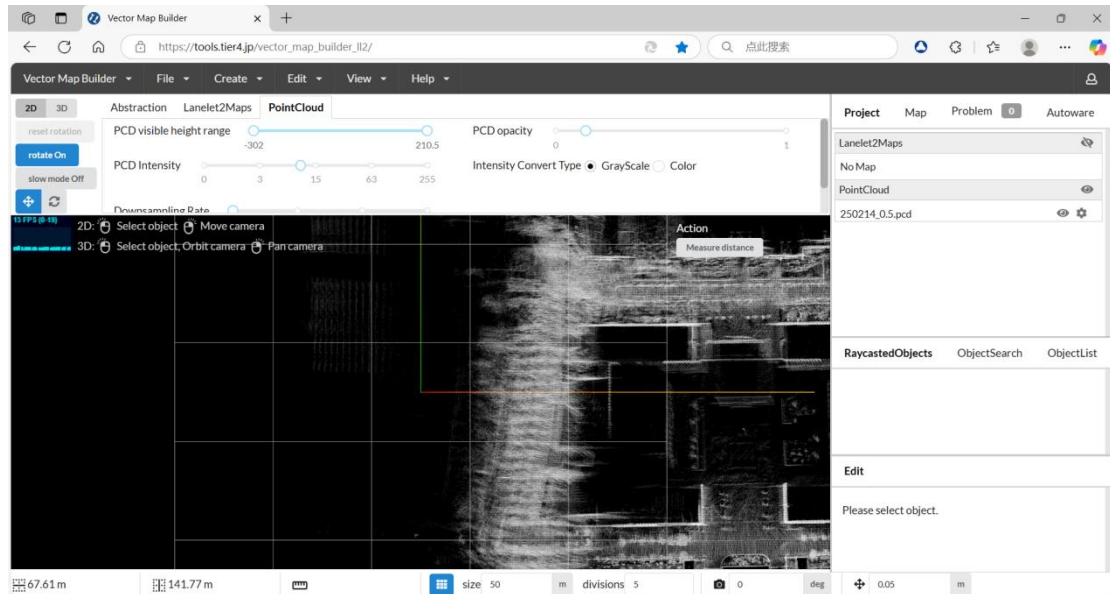


再次点击 Try it 后，会出现以下画面。

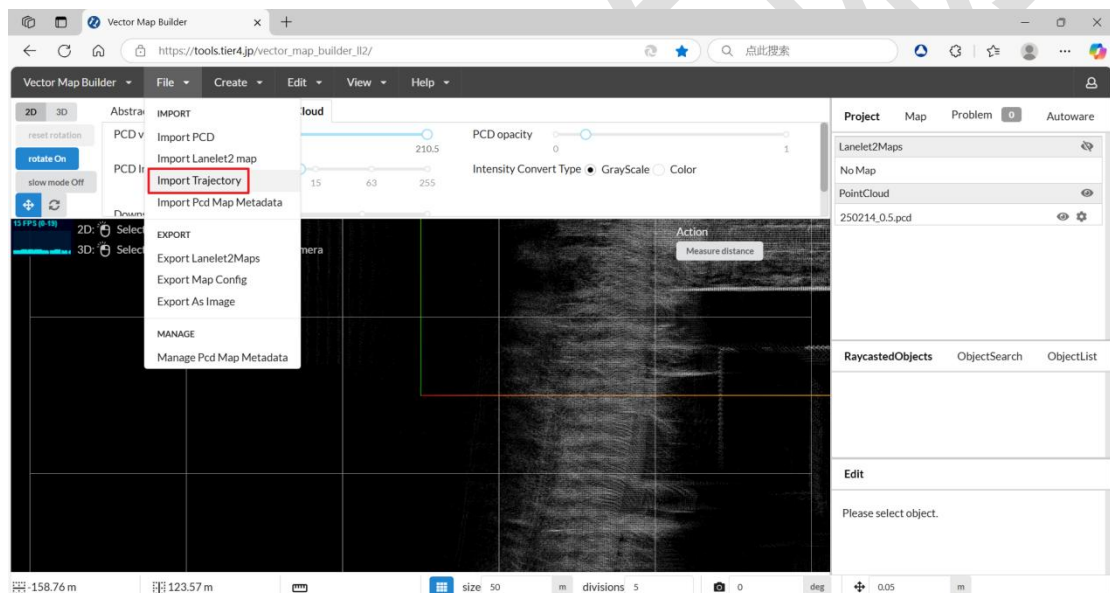


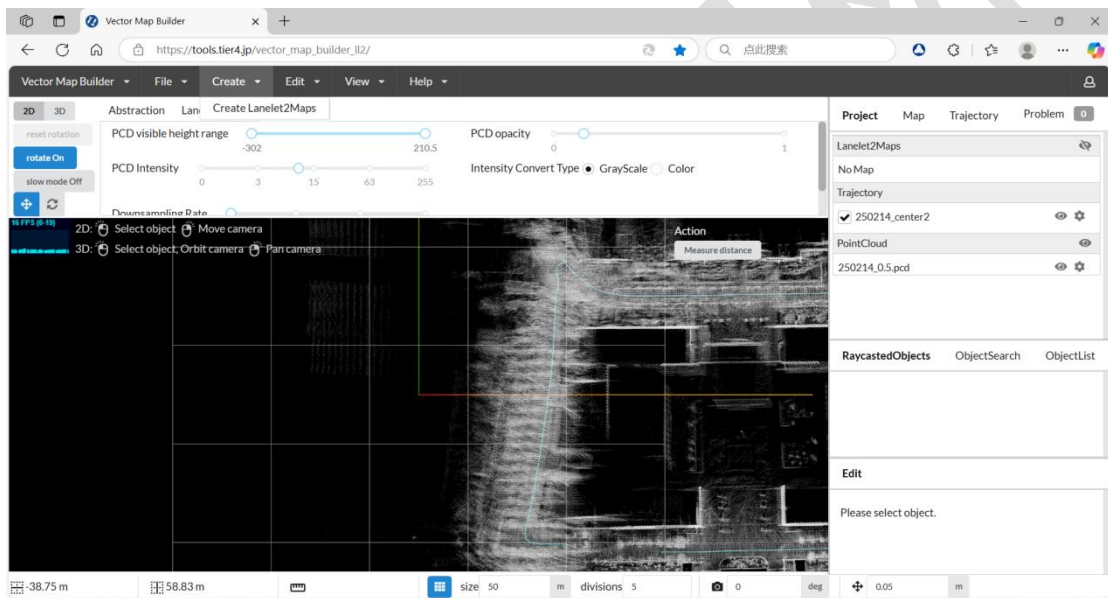
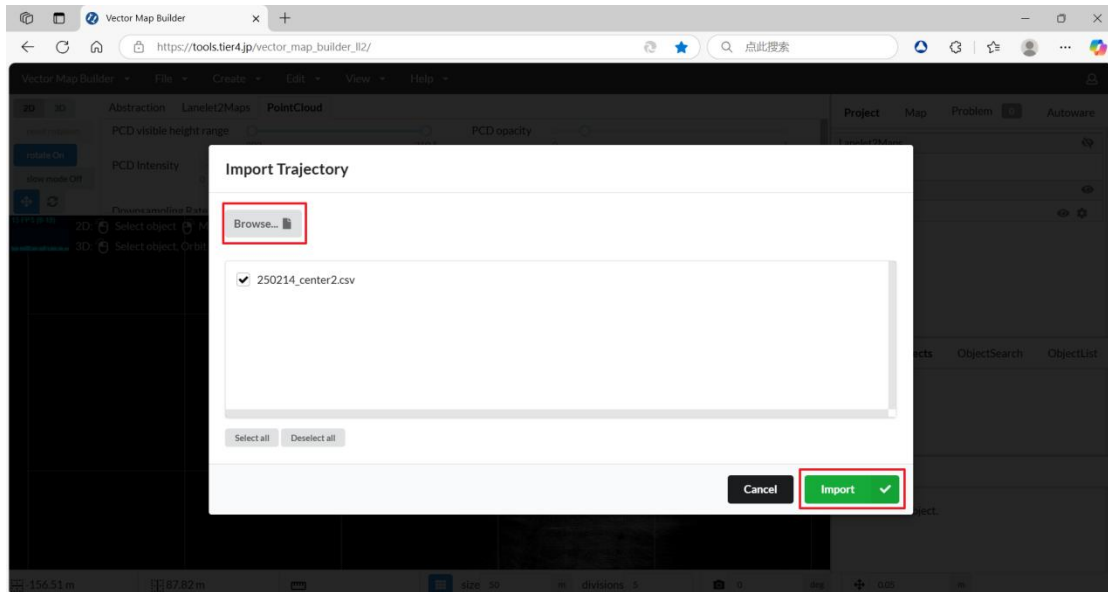
1) 点击 File，点击 Import PCD 导入 PCD 文件。



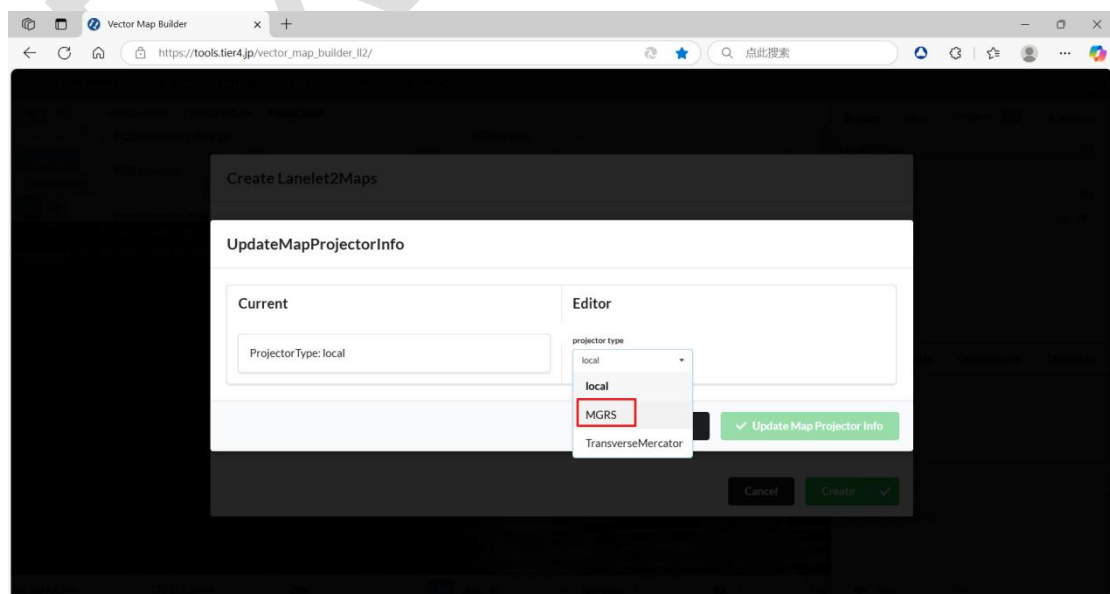
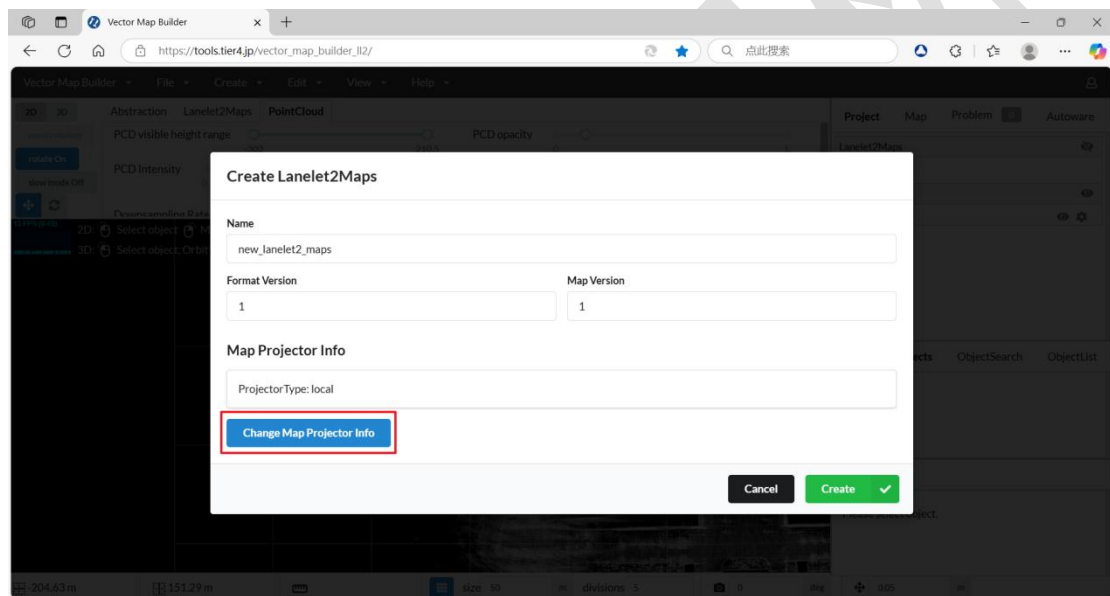
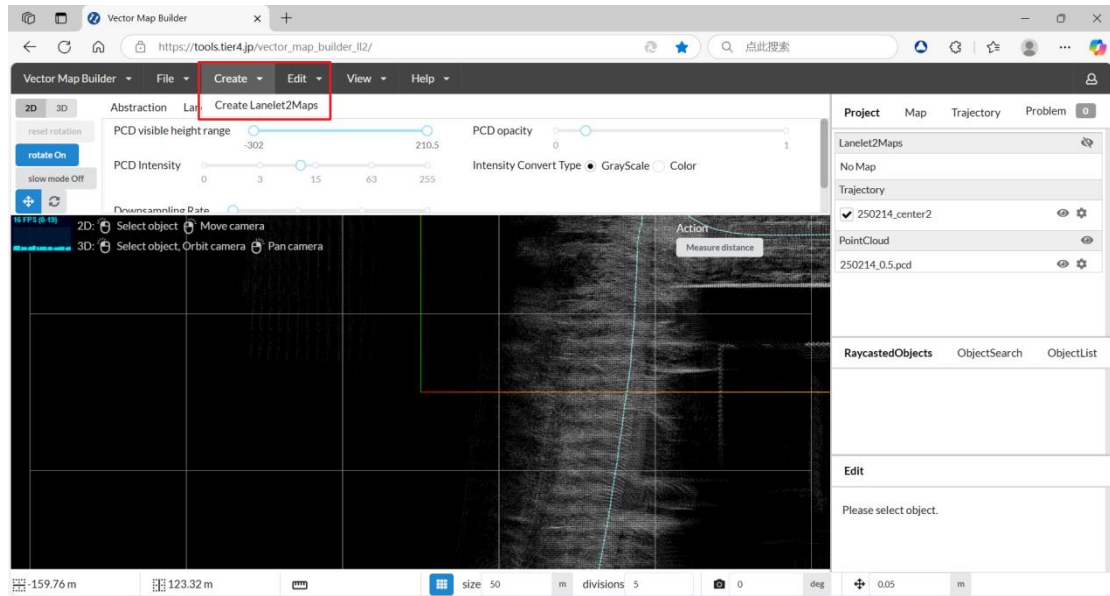


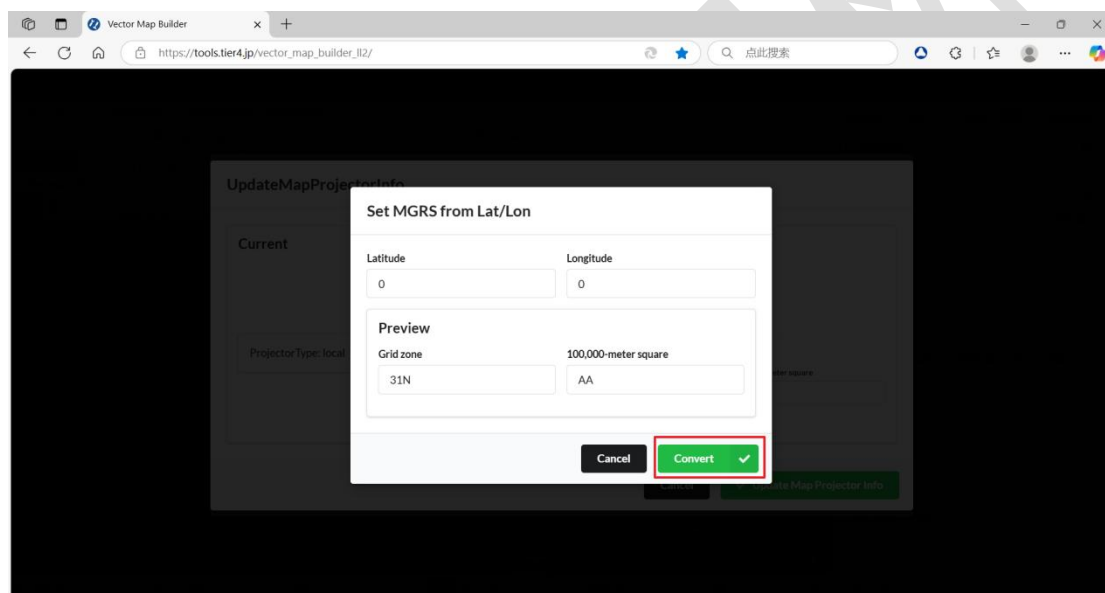
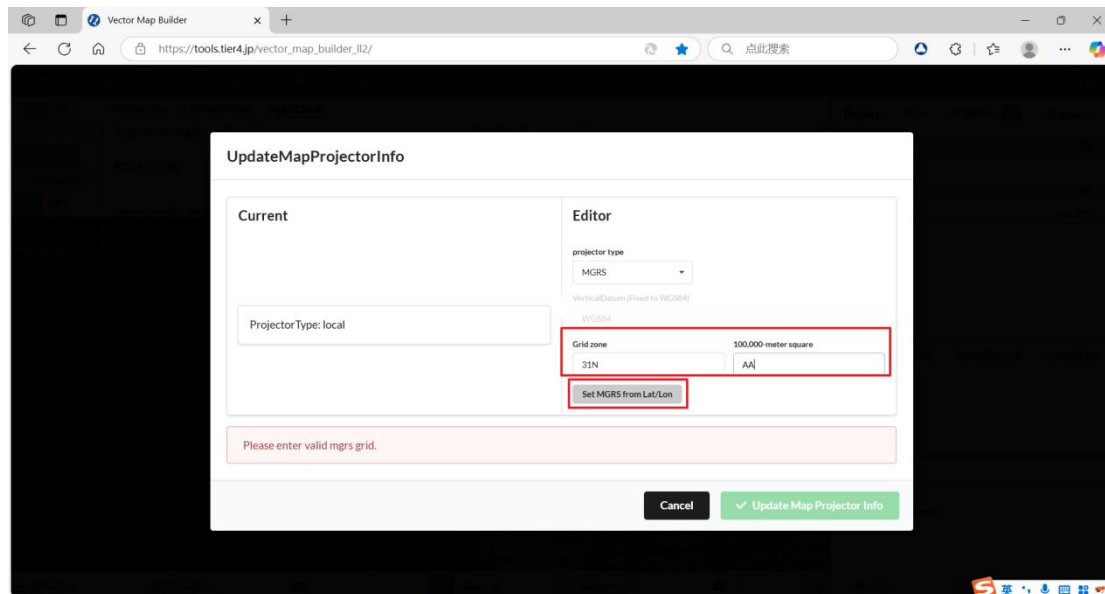
2) 点击 File，点击 Import Trajectory 导入 CSV 文件。

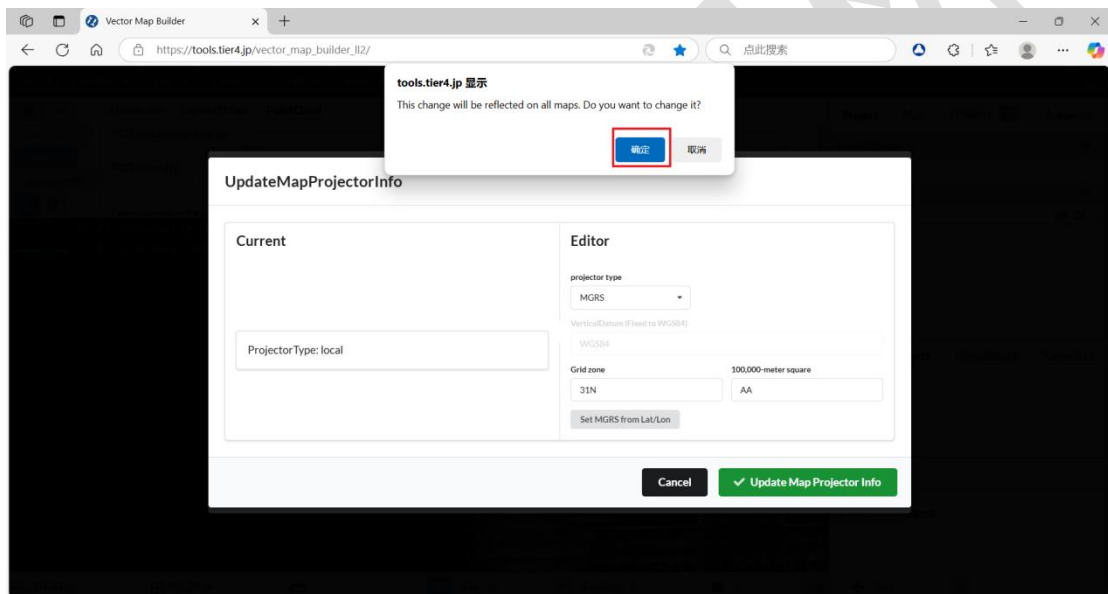
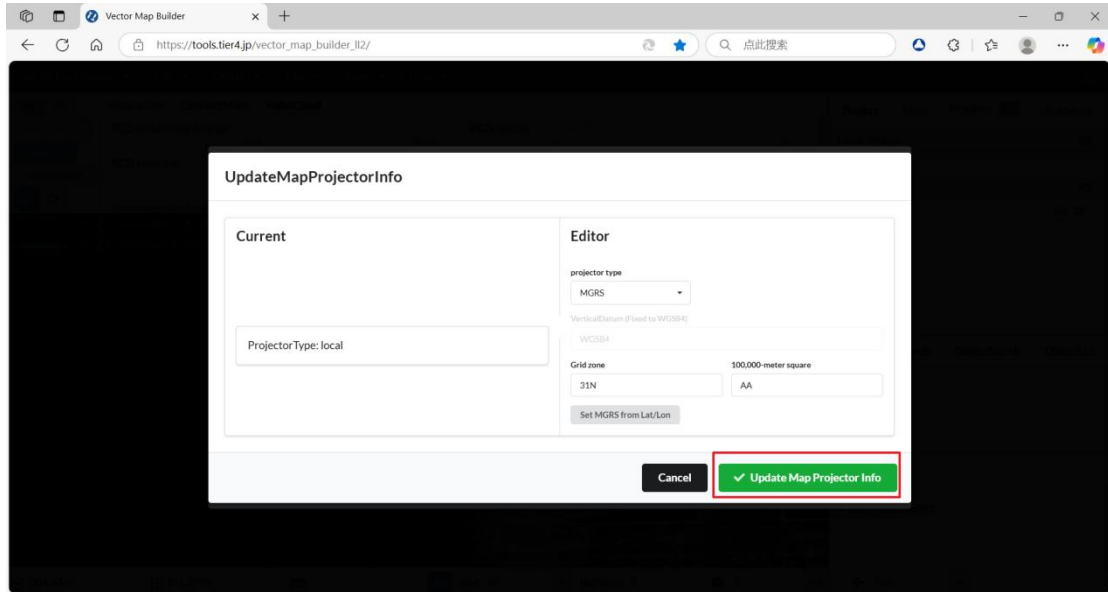


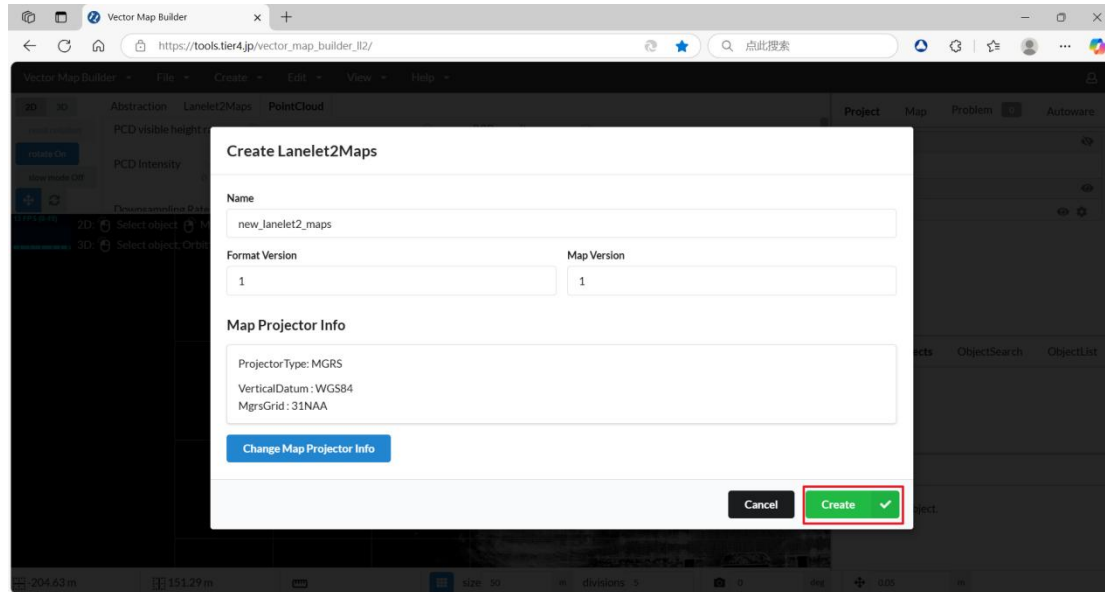


3) 点击 Create, 点击 Create Lanelet2Maps 创建一个 Lanelet2 地图(这步没有做好, 下面的操作都不能执行), 需要将 MGRS 填写为 31N AA。

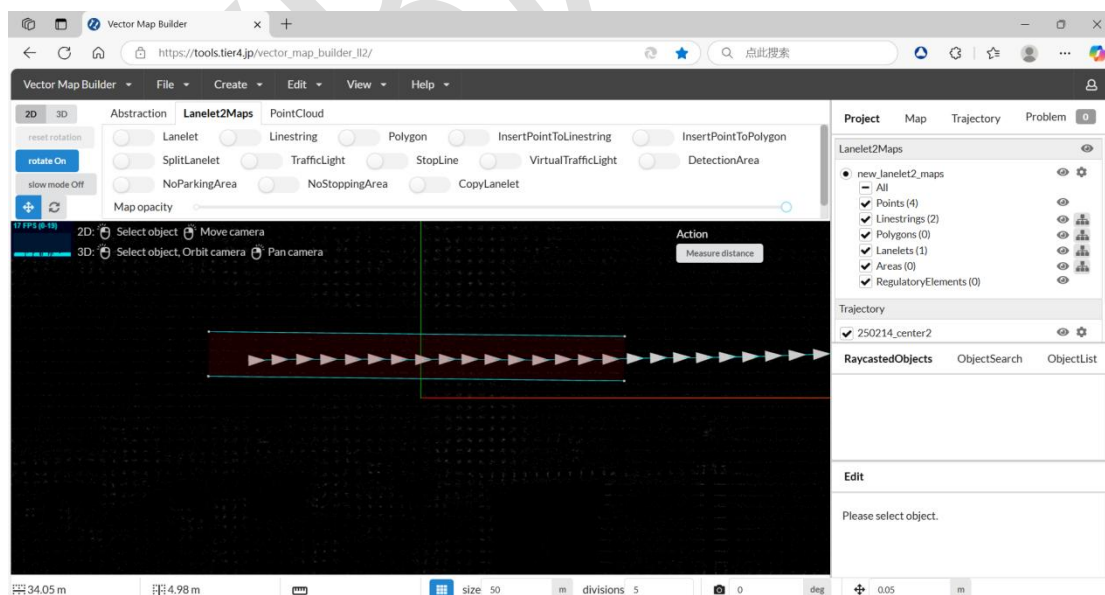
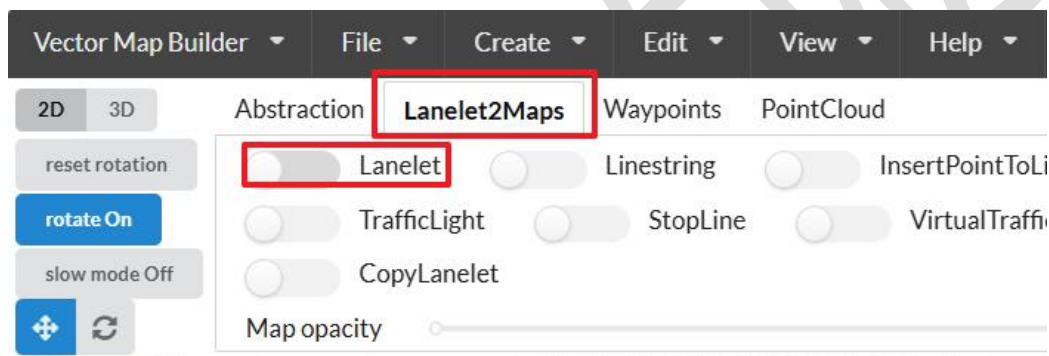




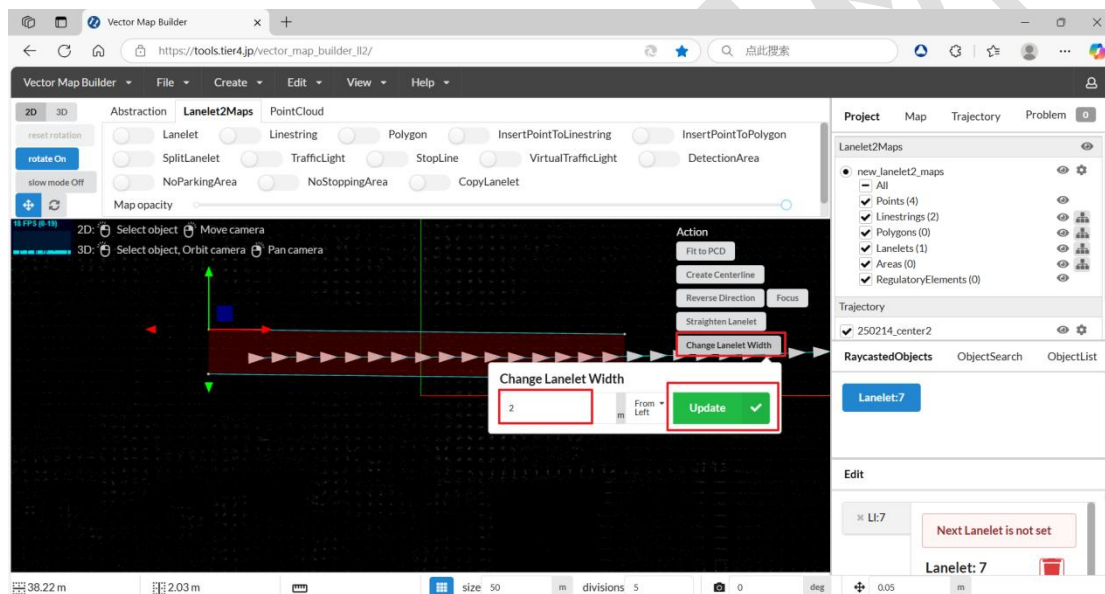
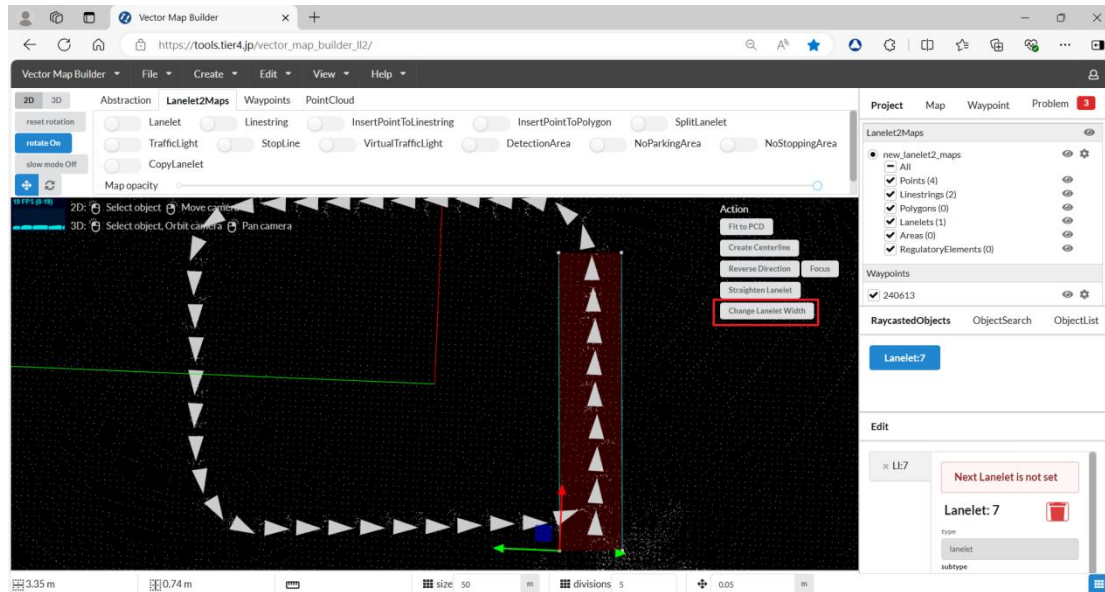




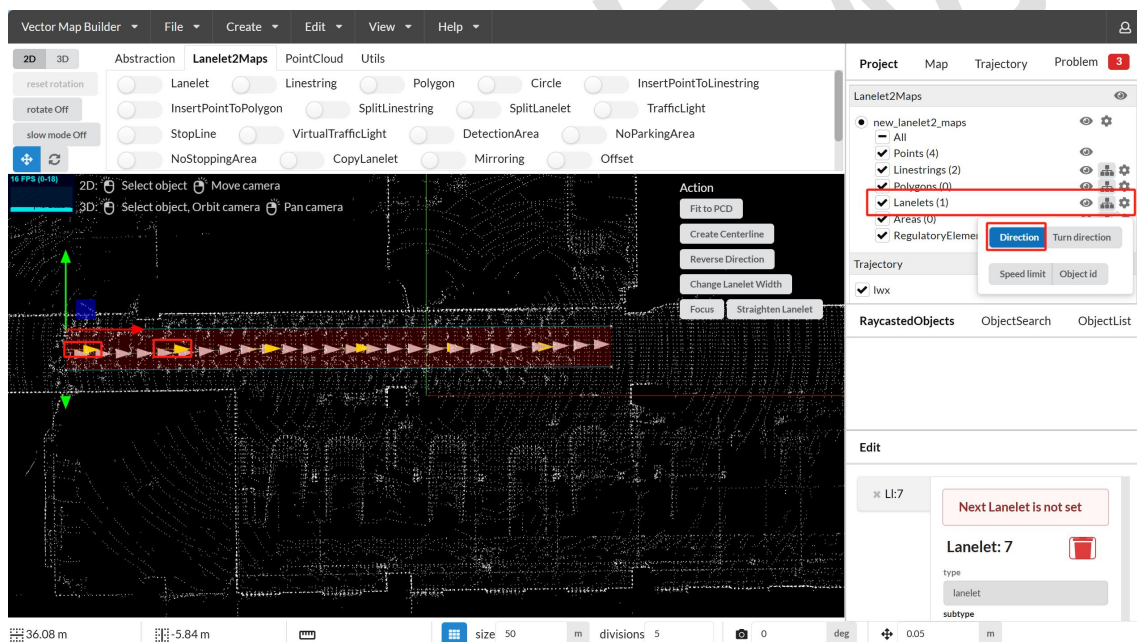
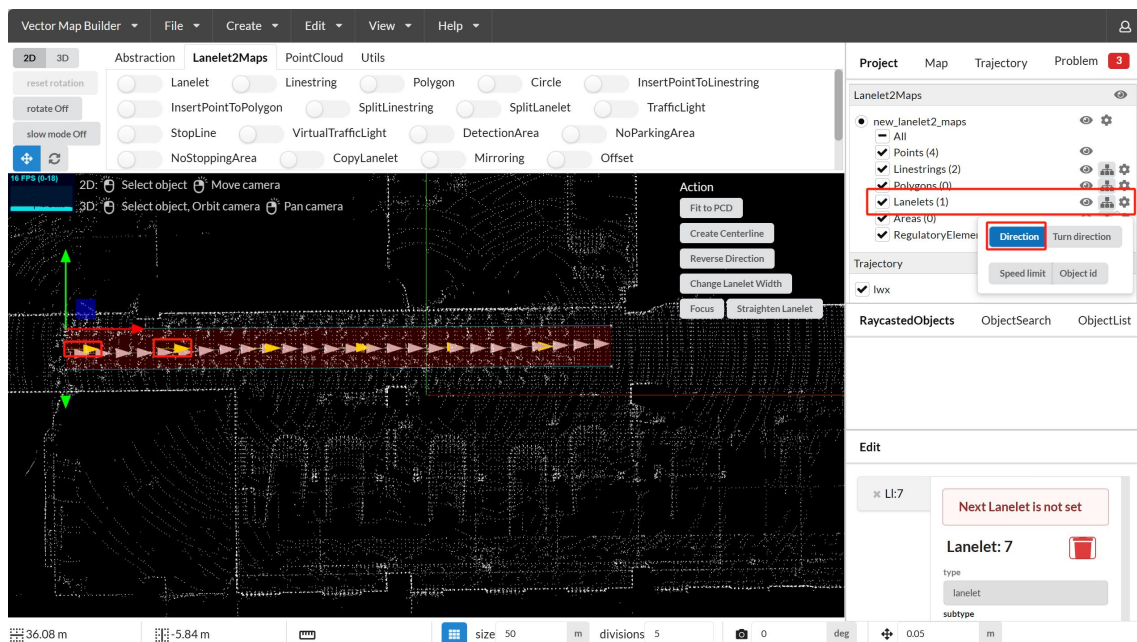
4) 点击 Lanelet2Maps 下的 Lanelet 绘制直线车道。



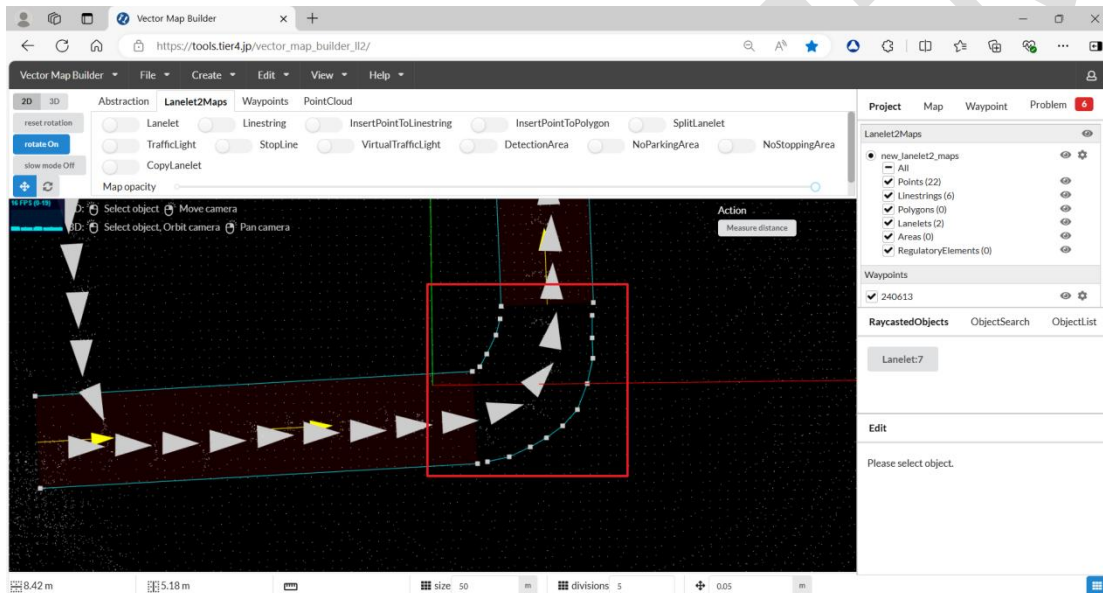
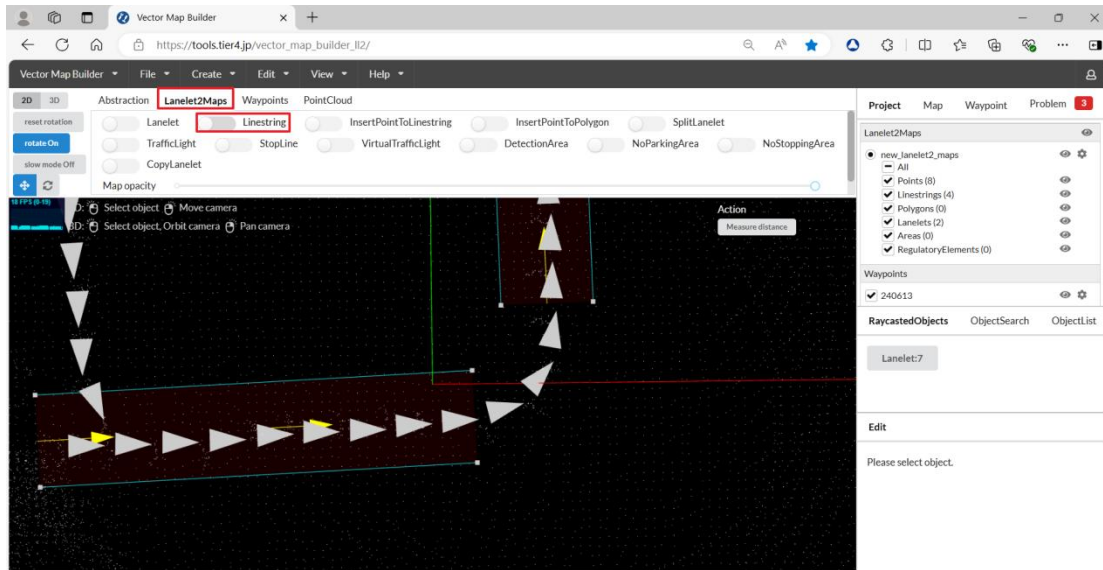
5) 点击车道，在 Action 中选择修改车道宽度。



6) 查看车道的行驶方向。

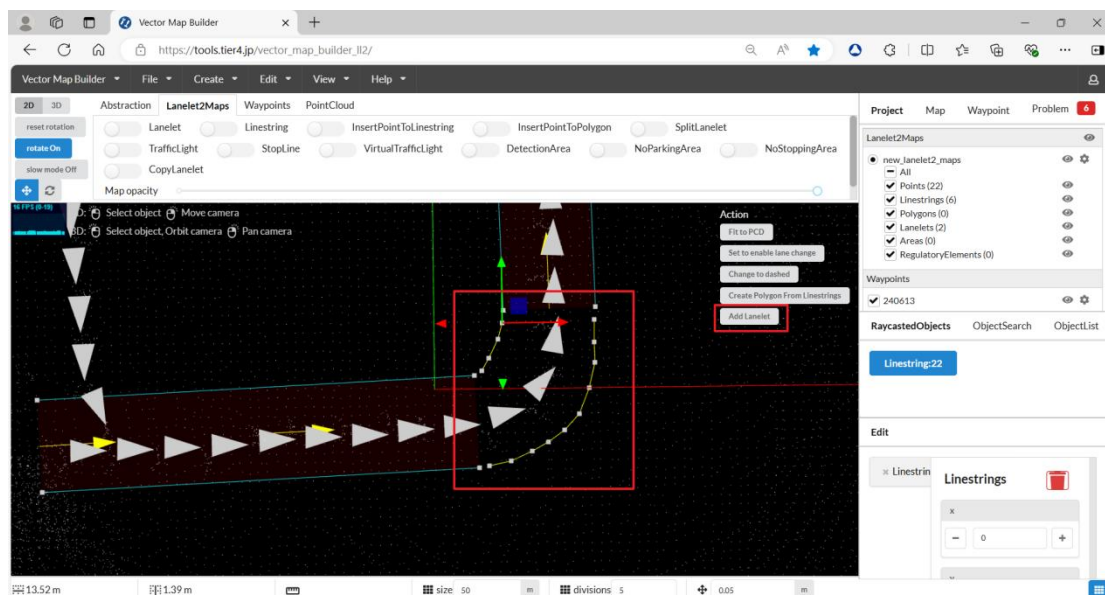


7) 点击 Lanelet2 下的 Linestring，绘制曲线车道。

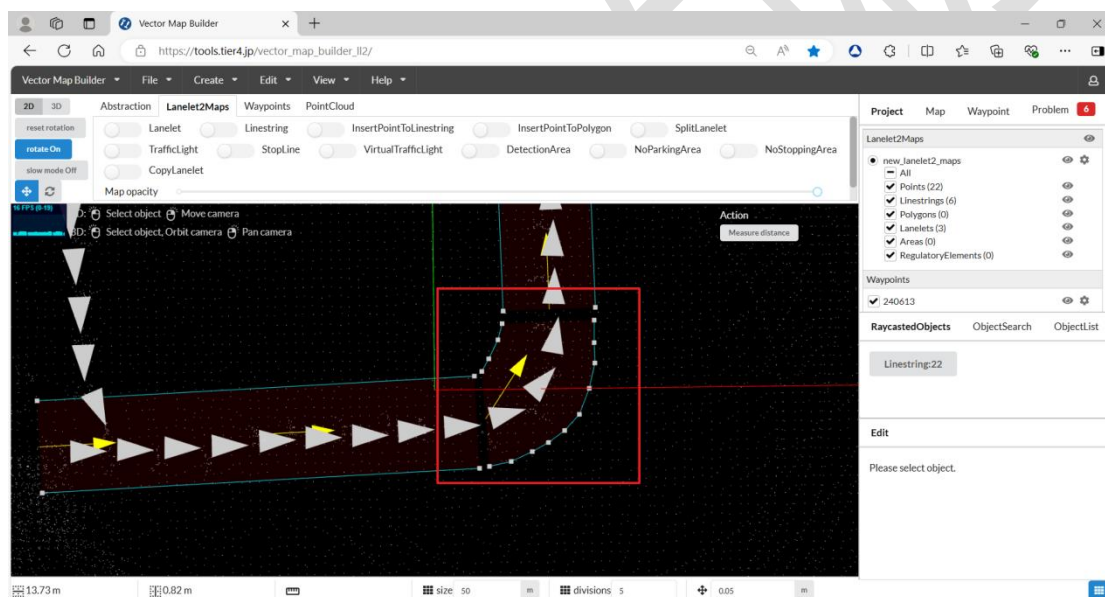


先点击一条 linestring（点击线，不要点击点），然后按住 Shift 并点击另外一条 linestring，在 Action 中选择 Add lanelet 创建一个车道。

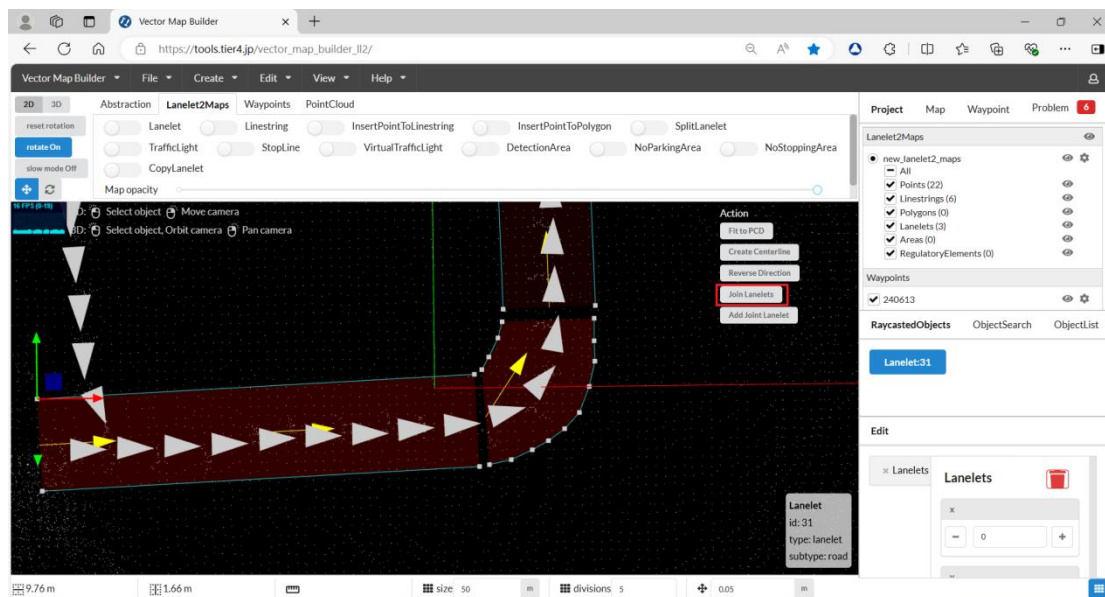
注意！先点左边后点右边的 linestring 会生成向前行驶的车道，反之向后行驶的车道。



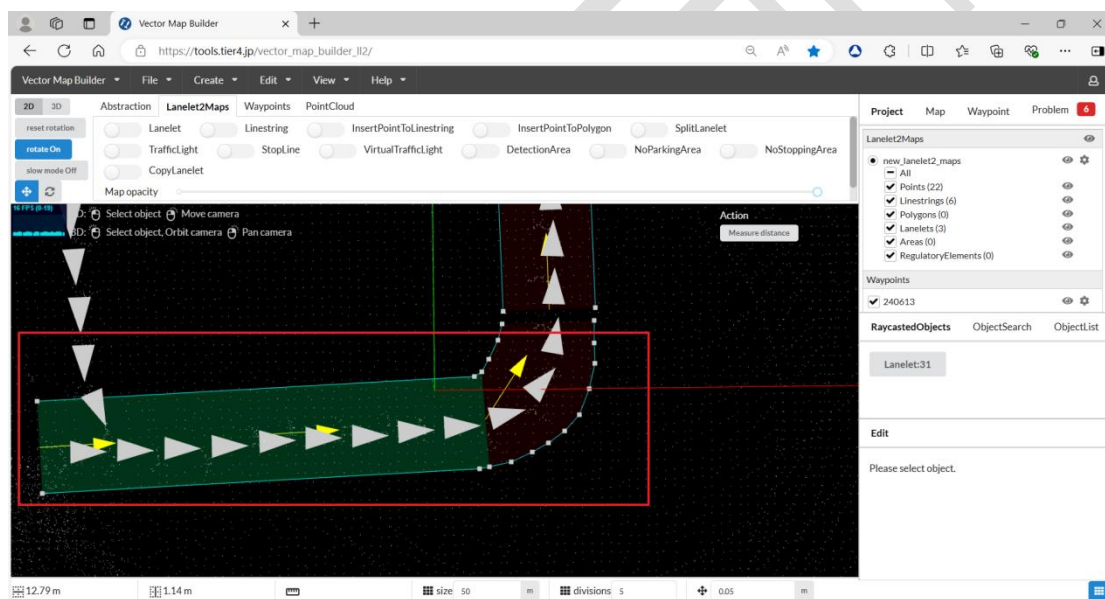
创建后的结果。



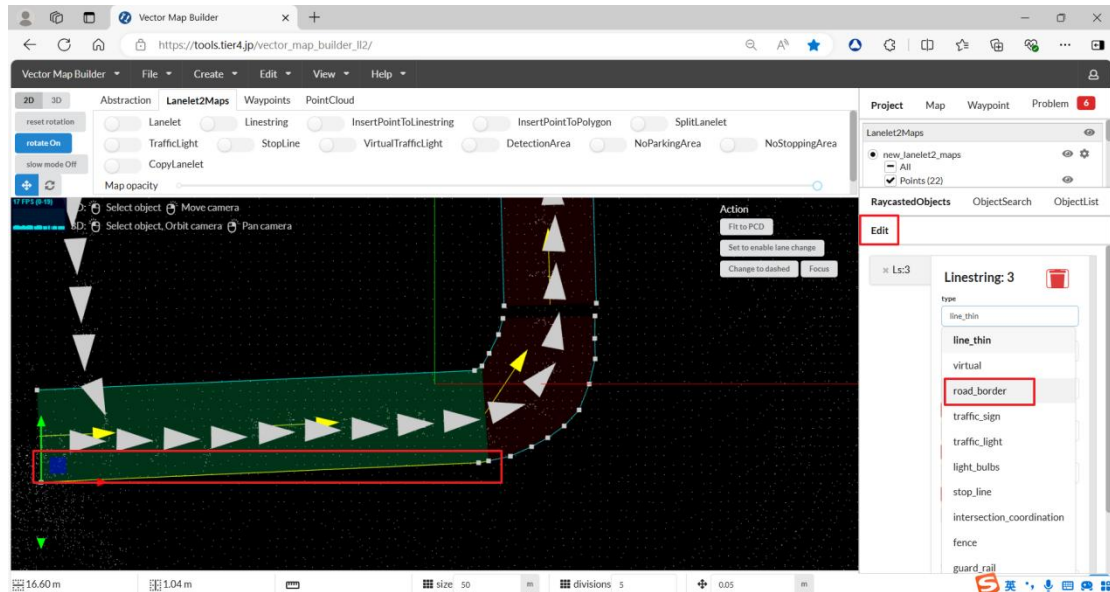
8) 点击一个车道后, 按住 Shift 并点击另外一个车道, 在 Action 中选择 Join Lanelets 合并两个车道。



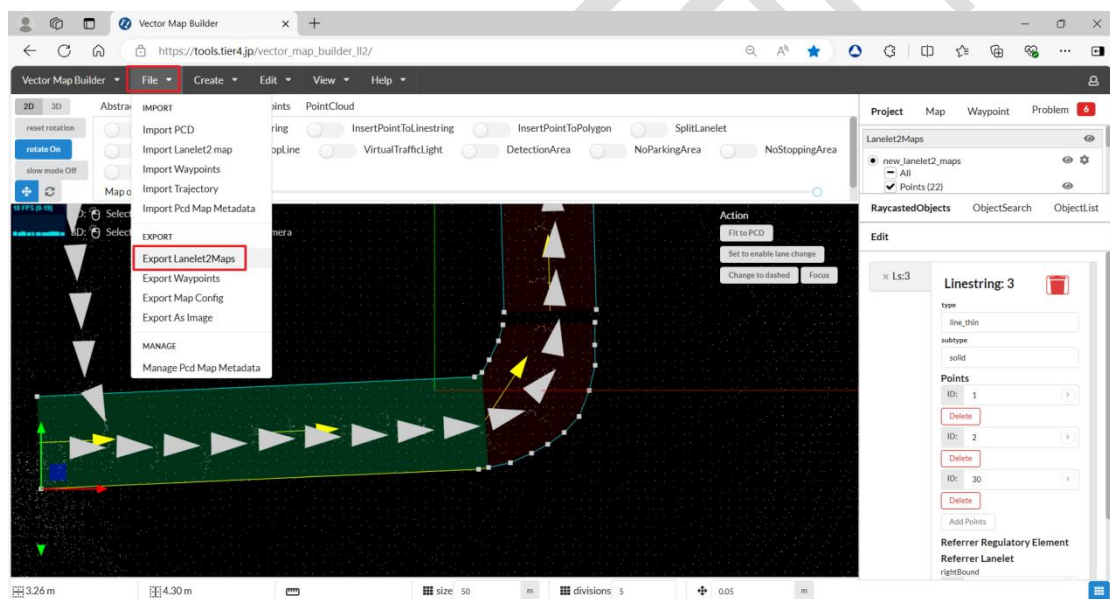
合并后的结果，当车道为绿色时，说明该车道正常，红色时说明该车道没有下一个车道进行连接。



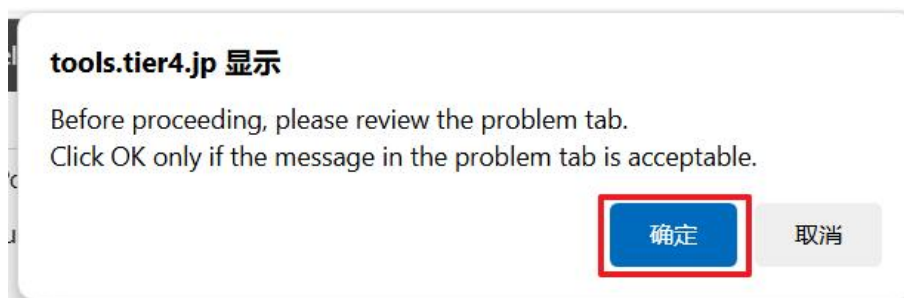
9) 点击车道边线，设置车道边缘为 road_border 路沿（只需设置两边，多车道并齐也一样，只需设置最外的边缘）。

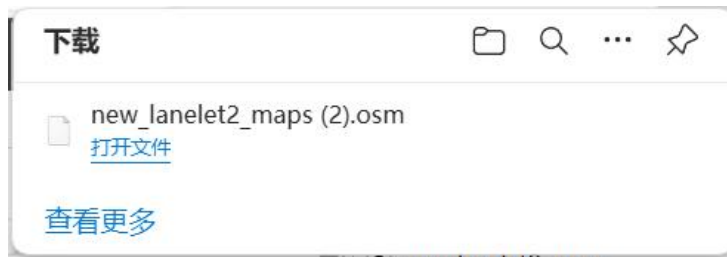
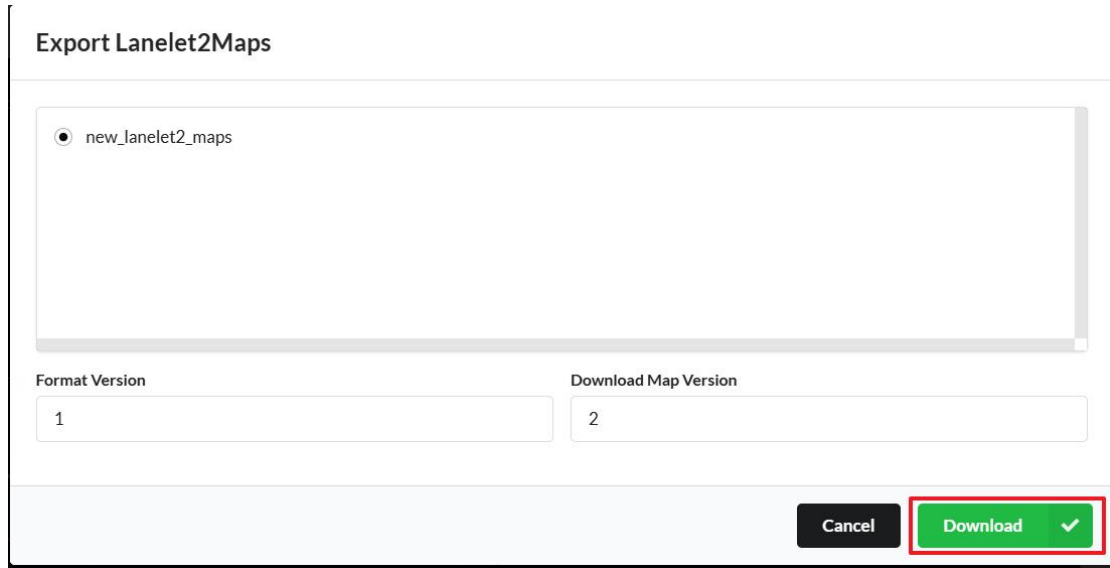


10) 当上述步骤都操作完成后, 需要导出 osm 地图, 点击 File, 点击 Export Lanelet2Maps, 导出 lanelet2 地图。



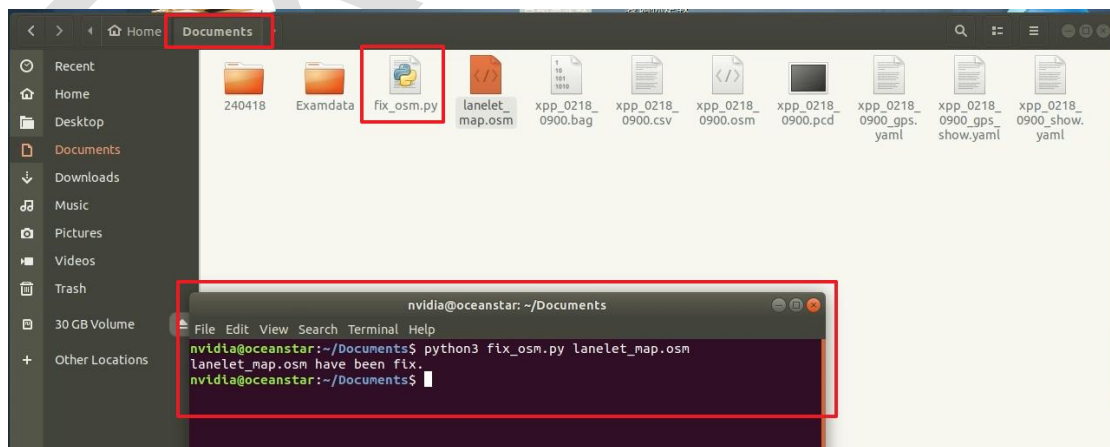
点击确定。





注意！osm 文件名只能由数字和字母和下划线（_）组成，其他字符请手动删除，例如括号和空格。

11) 将 osm 文件使用 U 盘拷贝到车辆的计算单元上面，使用快捷键 `ctrl + alt + t` 打开终端后，输入命令 `cd ~/Documents` 切换到该目录，执行完成后输入命令，`python3 fix_osm.py lanelet_map.osm` 修复地图工具的高度问题。成功则提示 xxx have been fix. 其余状态则失败，请联系技术人员解决。



3.6 标记位置

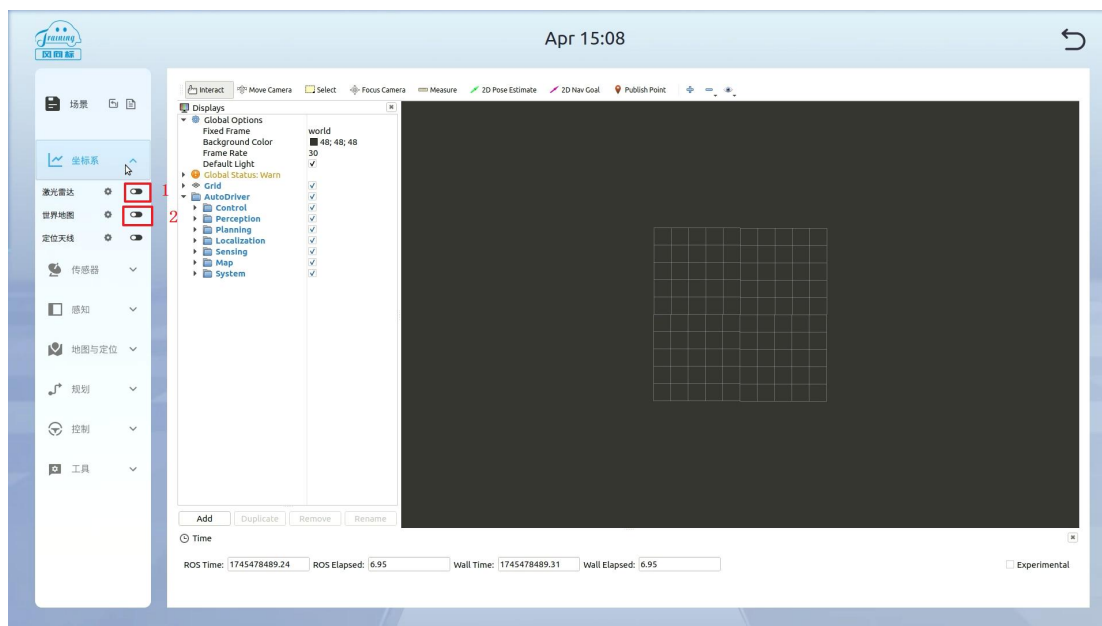
0) 在关闭所有软件和终端后再操作本小节。

1) 使用遥控器将车辆行驶到起始区域。

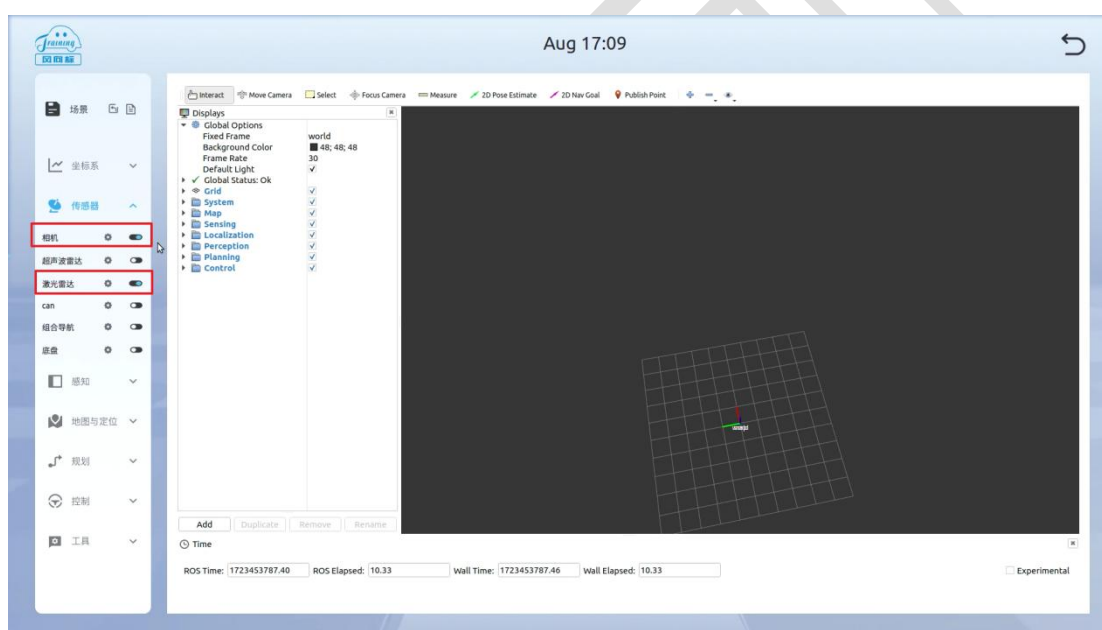


2) 打开自动驾驶教学软件，选择“高级教学”模式，找到坐标系下的“激光雷达”和“世界地图”功能并勾选启动，运行成功后会出现三色坐标系。

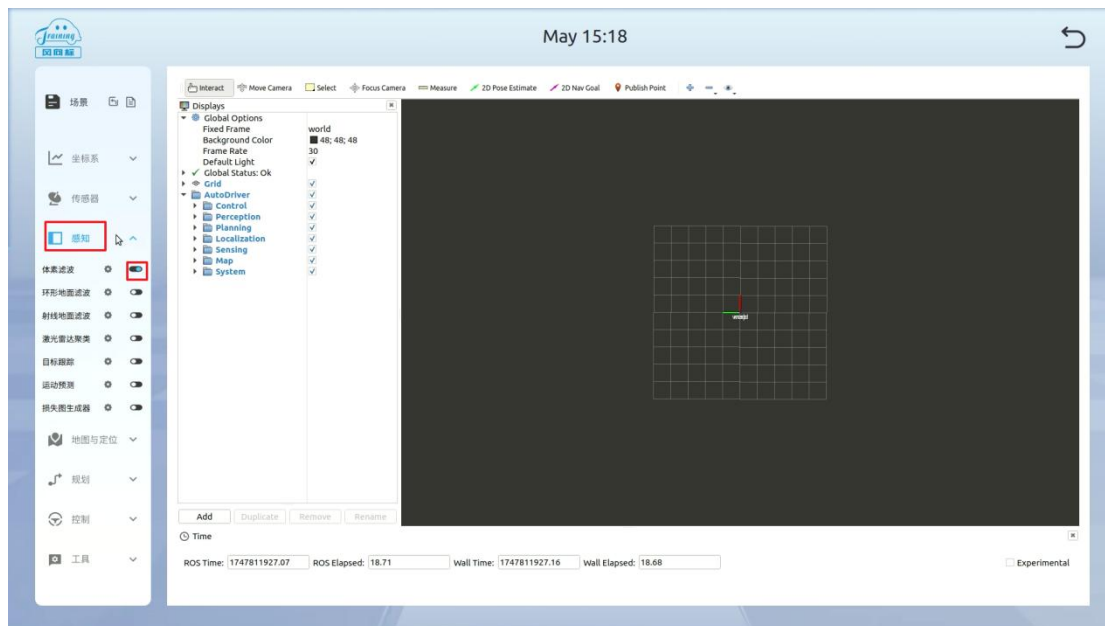




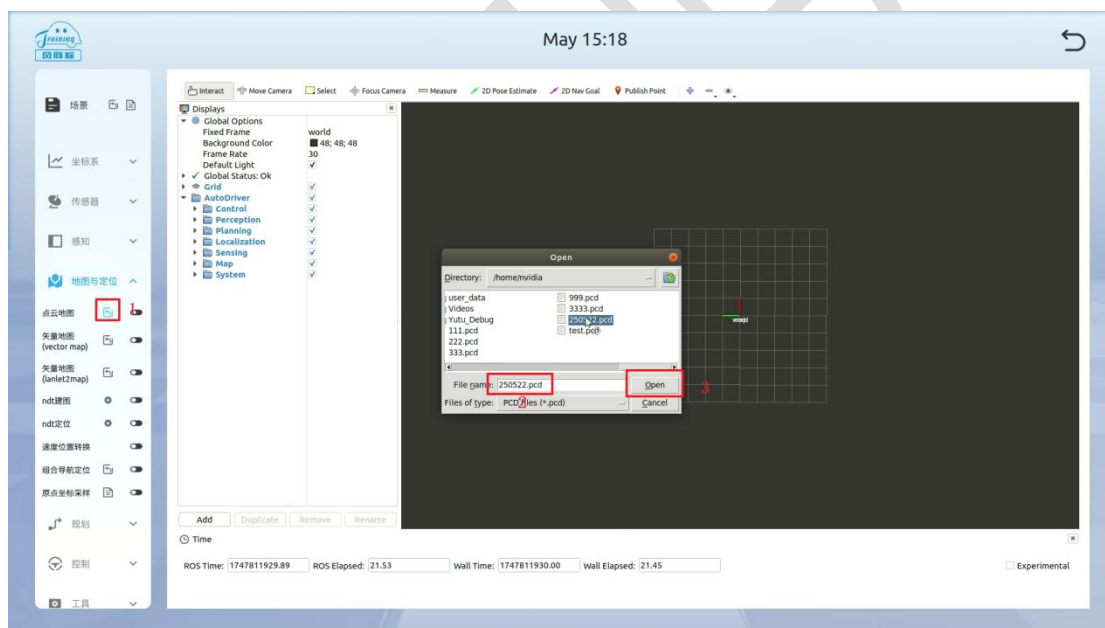
3) 找到传感器下的“相机”和“激光雷达”并勾选启动。

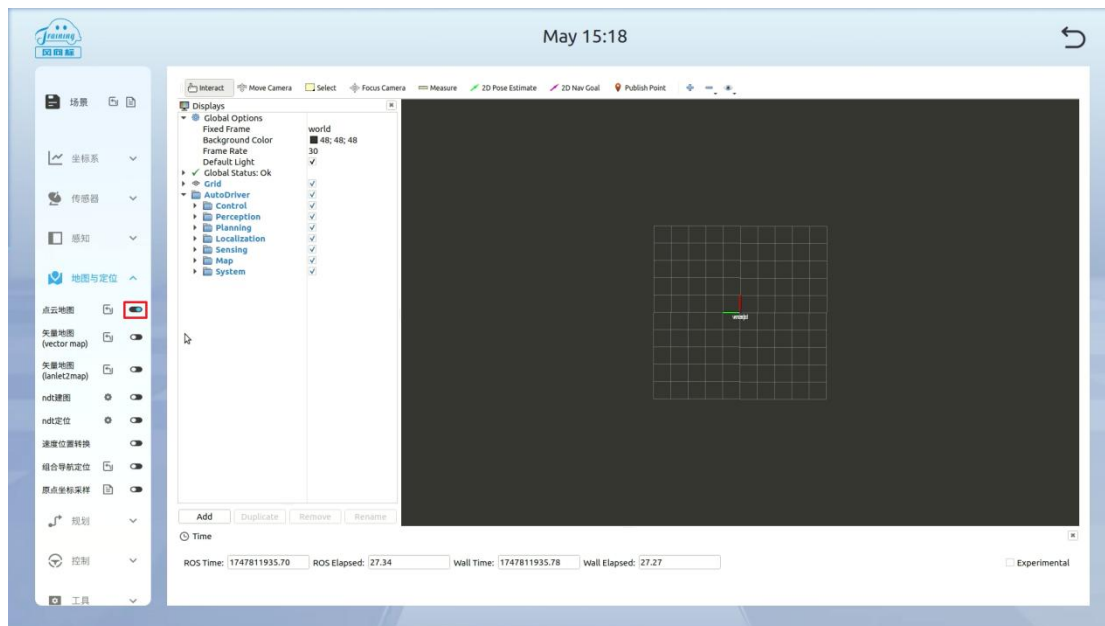


4) 找到“感知”下的“体素滤波”并勾选启动。

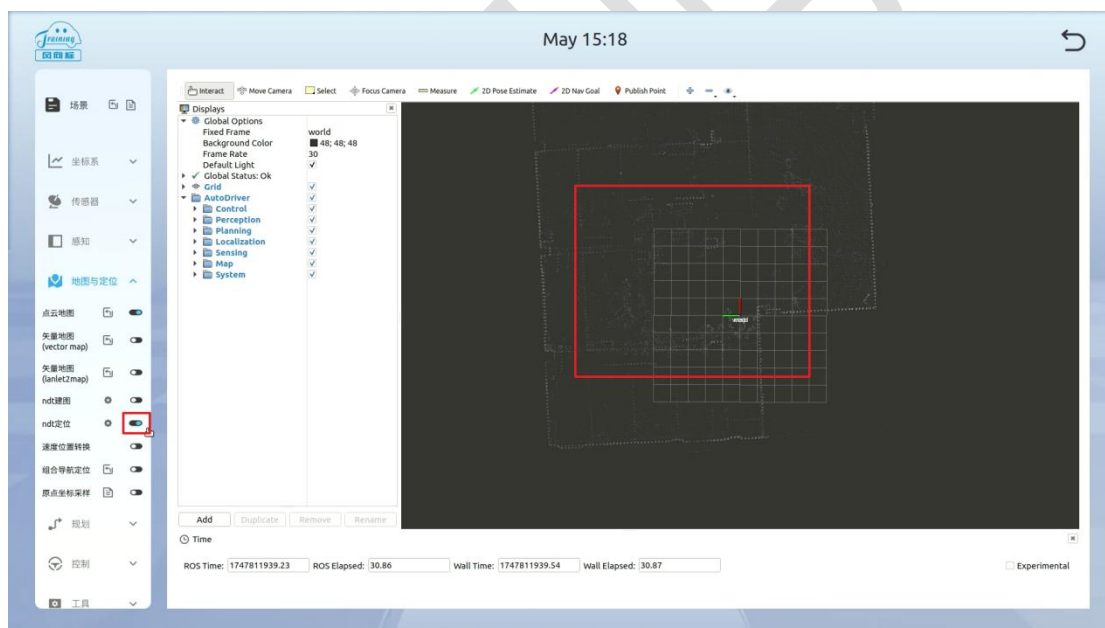


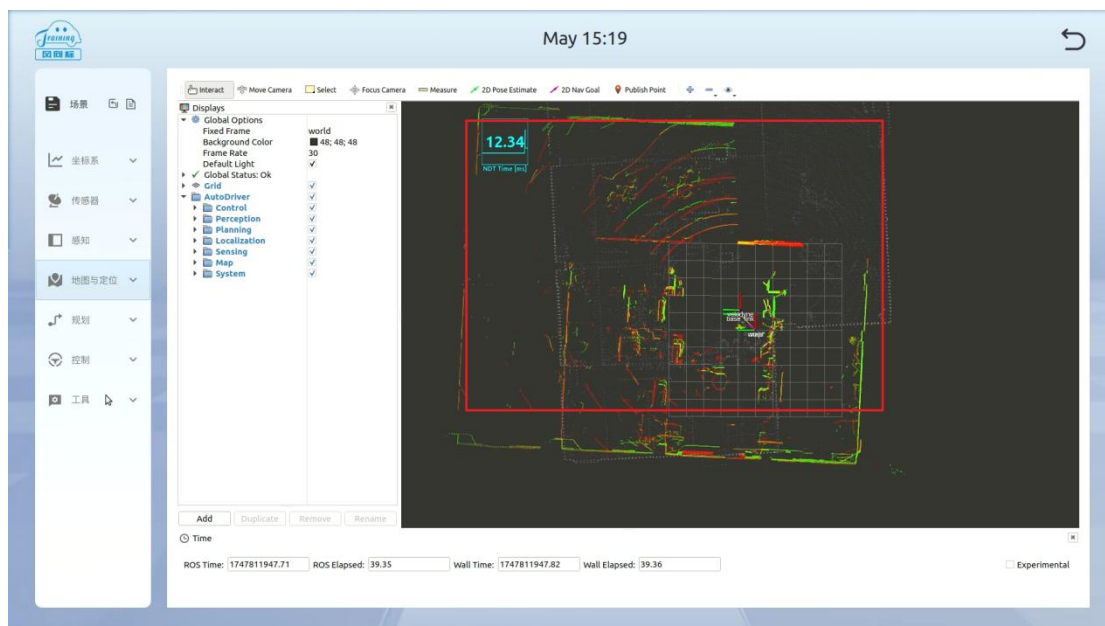
5) 找到“地图与定位”下的“点云地图”，先点击选择地图按钮，选择制作好的 PCD 后，勾选启动。



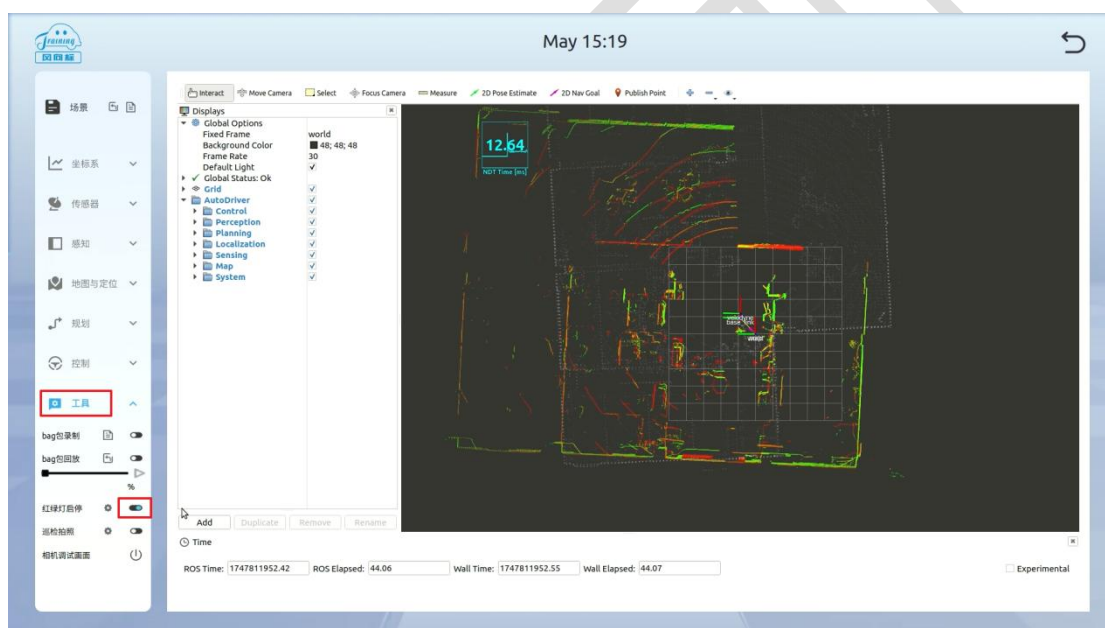


6) 当白色的点云地图加载成功后，找到“地图与定位”下的“ndt 定位”并勾选启动，成功后会看到彩色点云出现。

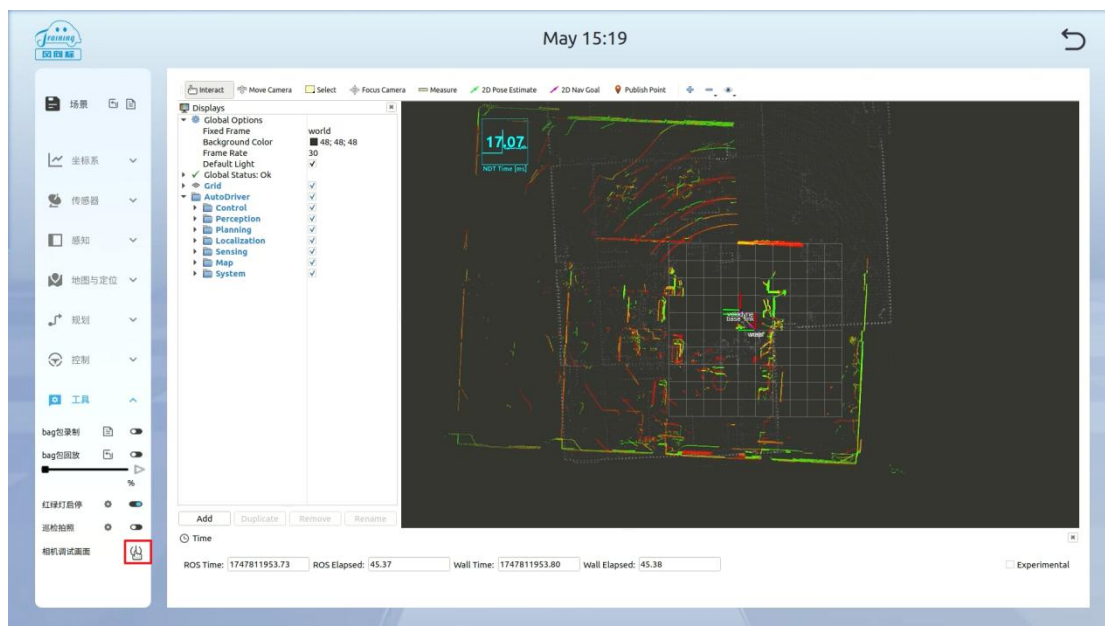




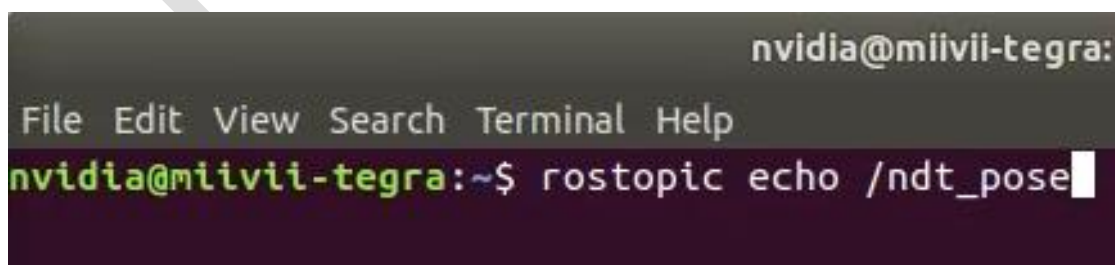
7) 找到“工具”下的“红绿灯启停”并勾选启动。



8) 找到“工具”下的“相机调试画面”，点击旁边的启动按钮，等待一会后会弹出相机调试窗口，选择/img_detection，用于测试是否成功识别出红绿灯。



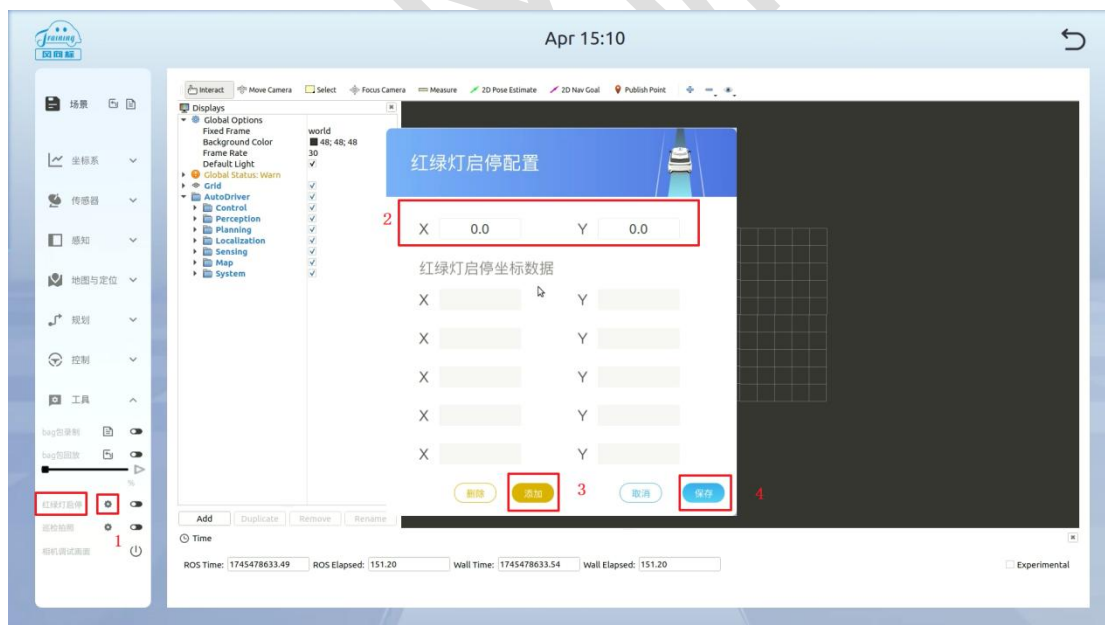
9) 使用快捷键 `Ctrl + Alt + t` 打开终端，输入 `rostopic echo /ndt_pose` 读取当前车辆在地图位置。

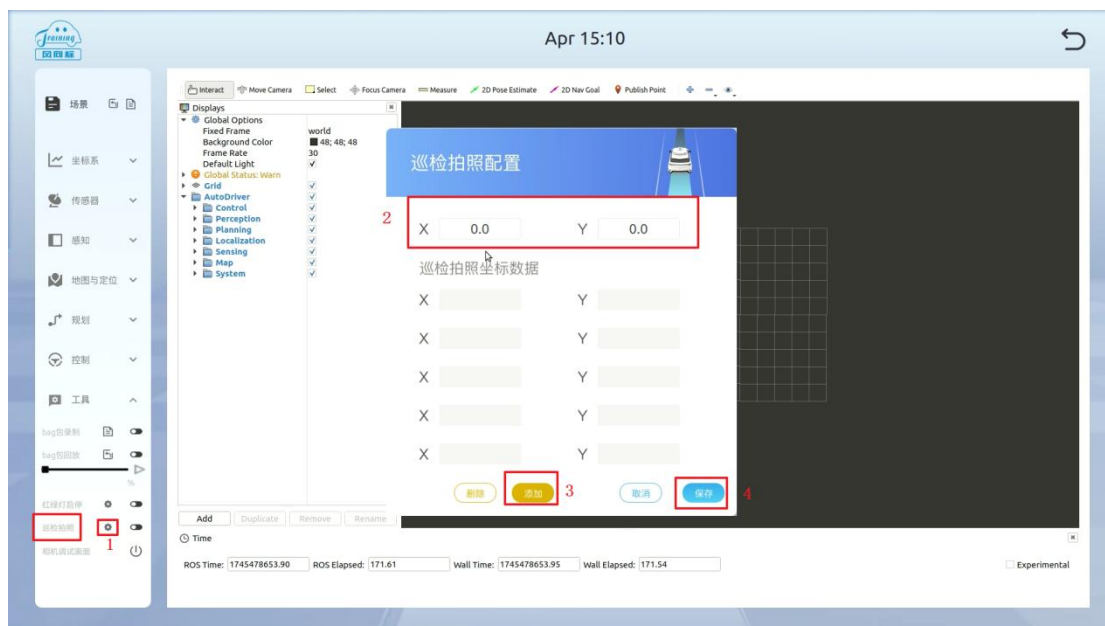


```
nvidia@miivii-tegra: ~
File Edit View Search Terminal Help

orientation:
  x: 0.00460313901572
  y: -0.0172708153947
  z: -0.0452863736845
  w: 0.998814134064
---
header:
  seq: 455
  stamp:
    secs: 1723453865
    nsecs: 334700000
  frame_id: "map"
pose:
  position:
    x: -0.114143103361
    y: 0.250378400087
    z: -0.00814700126648
  orientation:
    x: 0.00732934612327
    y: -0.0174483069741
    z: -0.0454426720166
    w: 0.998787665537
---
```

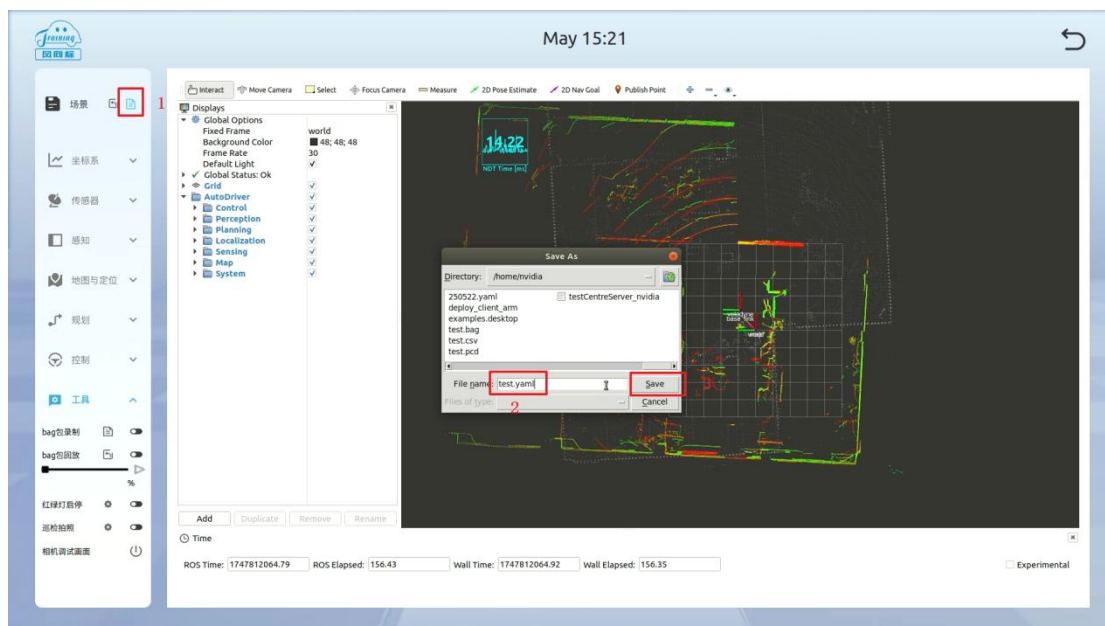
10) 根据场地指示，遥控车辆到达指定巡检点和识别红绿灯停止线处，记录此刻车辆坐标位置到“红绿灯启停”和“巡检拍照”的配置页面处，保留小数点一位，四舍五入（注意！红绿灯启停需要取消勾选才可配置）。





11) 选择导入 Lanelet2 矢量地图文件路径和 CSV 路径点文件路径后，点击保存参数按钮，保存参数。





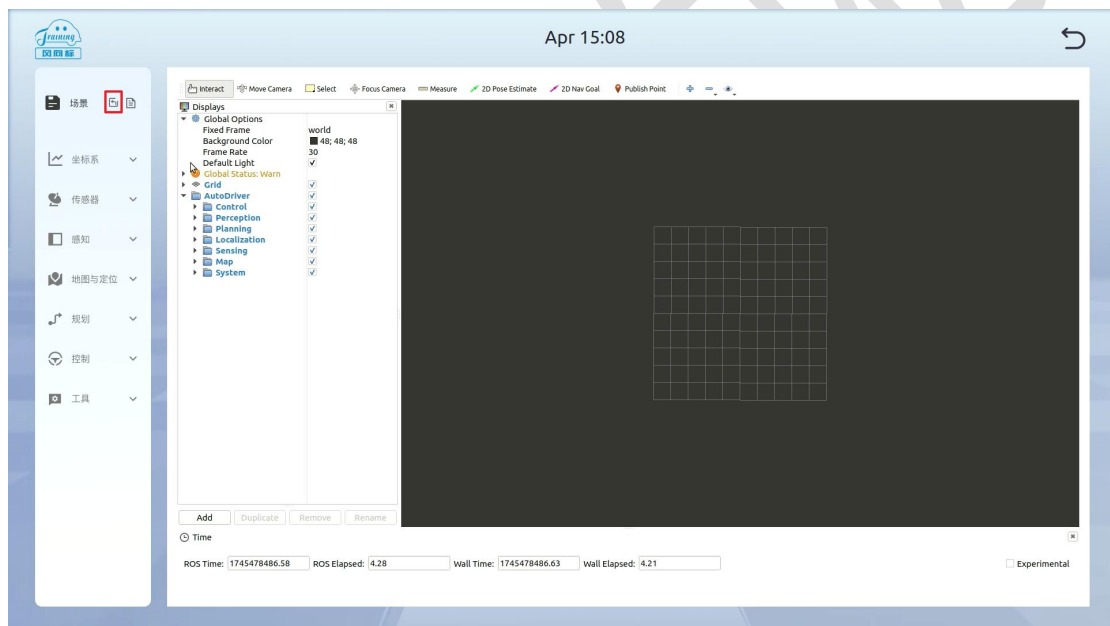
3.7 道路测试-高级教学-方案一

0) 在关闭所有软件和终端后再操作本小节；

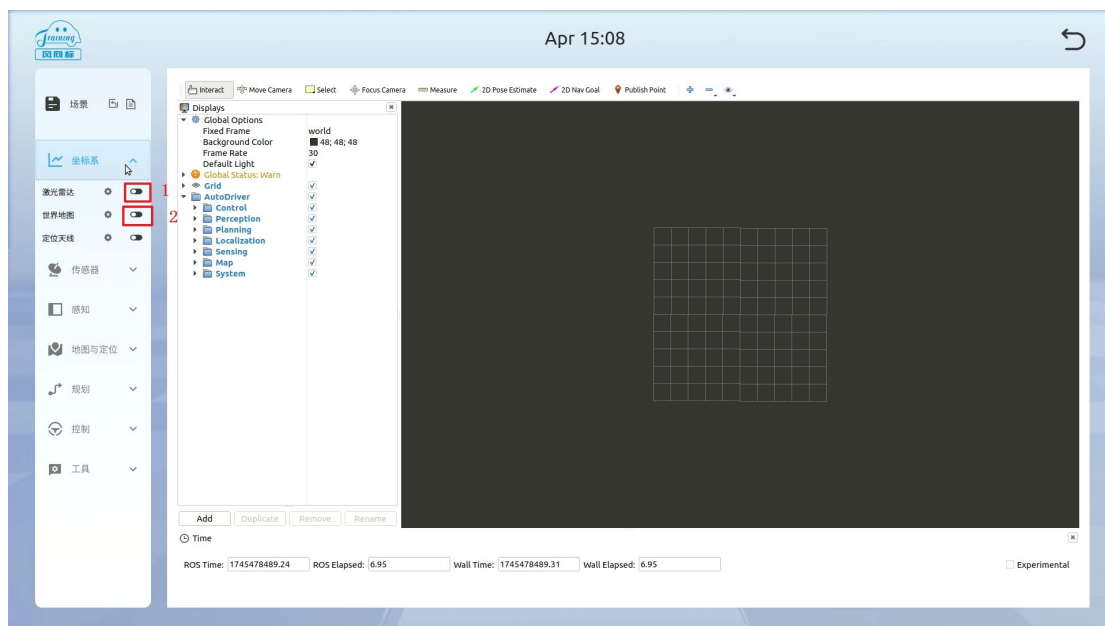
1) 使用遥控器将车辆行驶到起始区域；



2) 打开自动驾驶教学软件，选择“高级教学”模式，导入从 3.6 小节导出的参数；

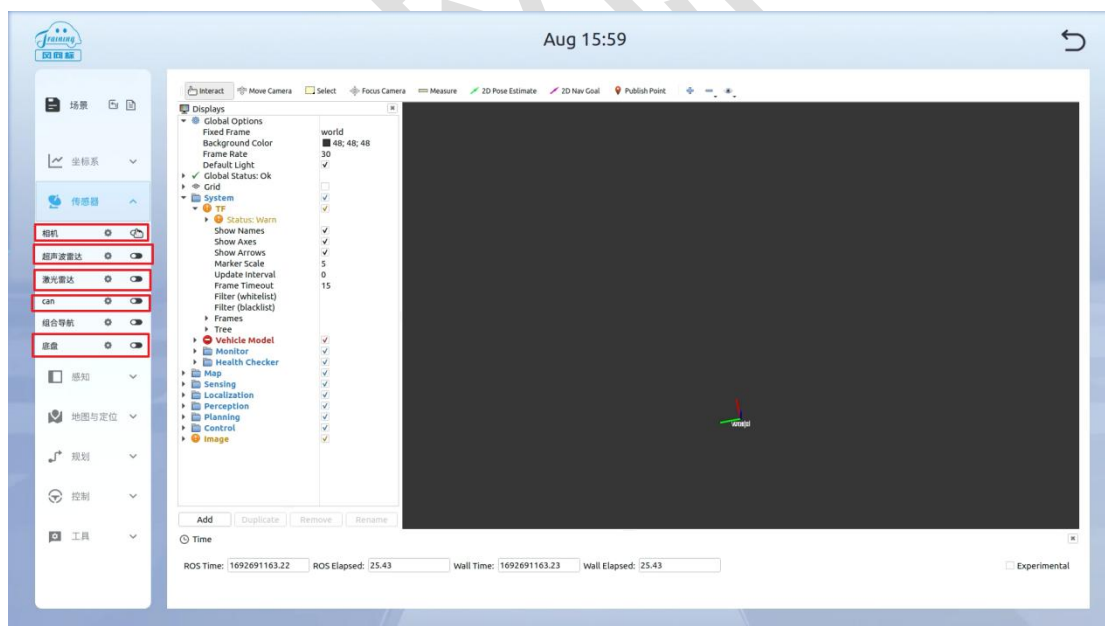


3) 找到坐标系下的“激光雷达”和“世界地图”功能并勾选启动，运行成功后会出现三色坐标系：

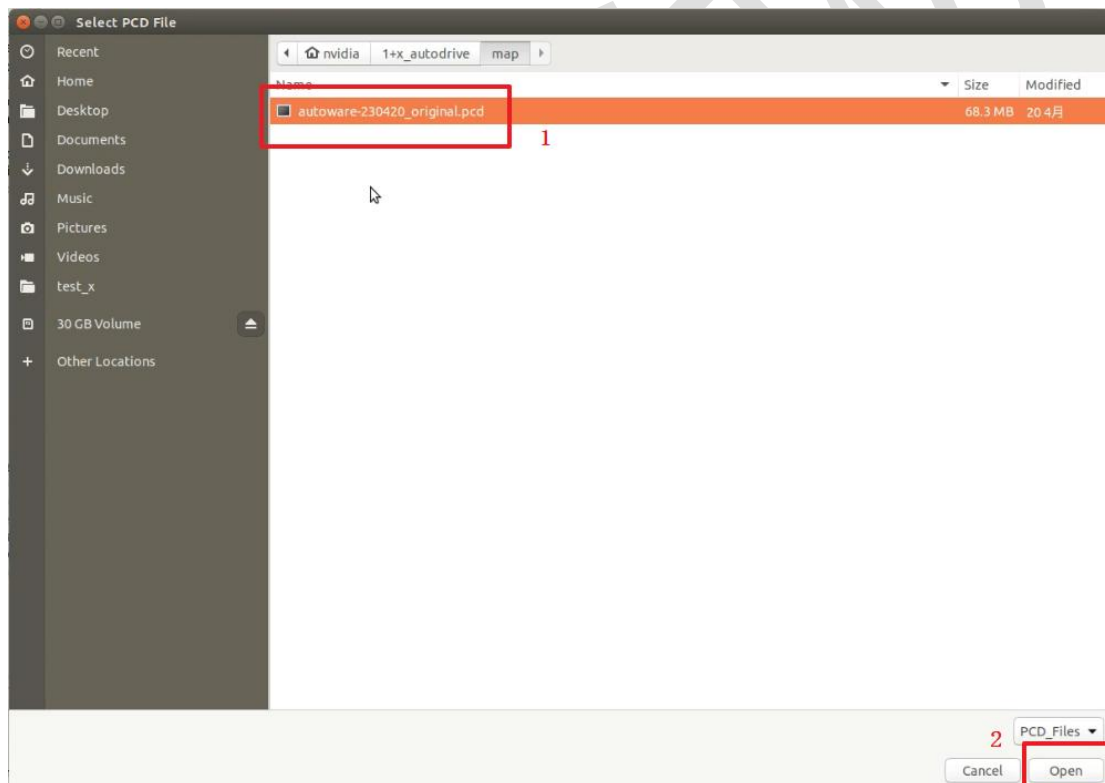
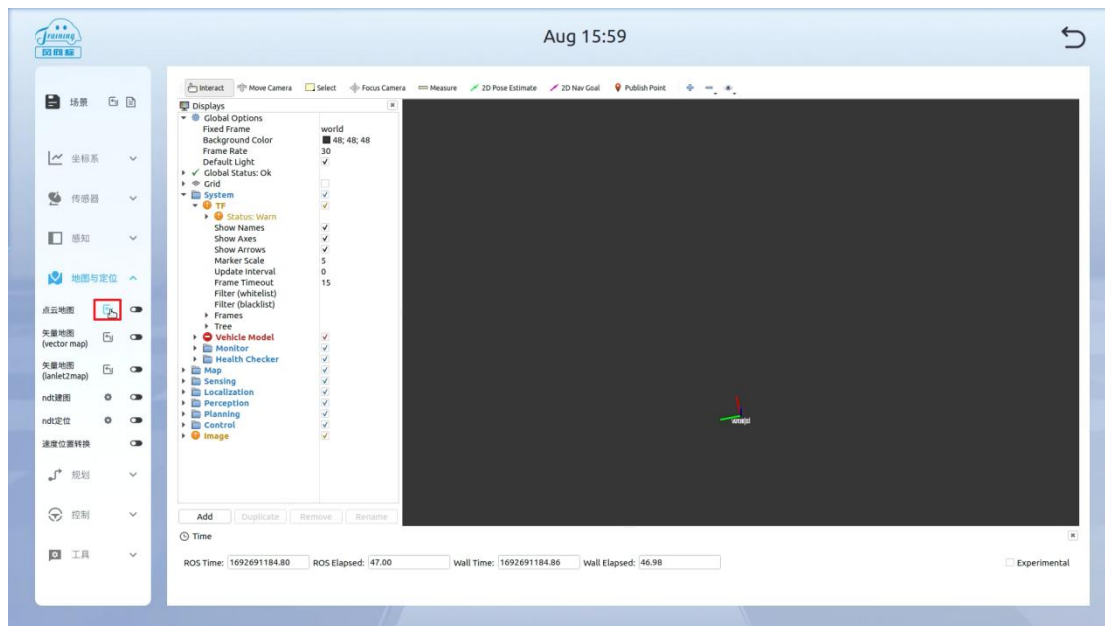


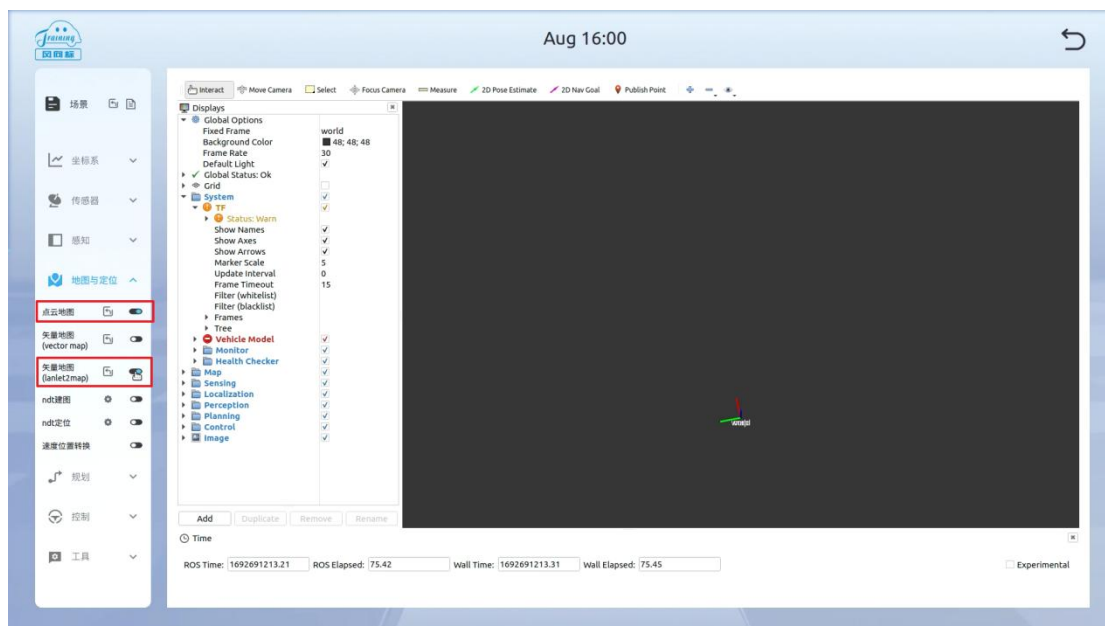
4) 找到传感器下的“相机”、“超声波雷达”、“激光雷达”、“CAN”和“底盘”并勾选启动它们，后台会启动对应的驱动程序；

注意！如果没有做 3.6 标记位置或不需要红绿灯识别或巡检点检测，请不要勾选相机。

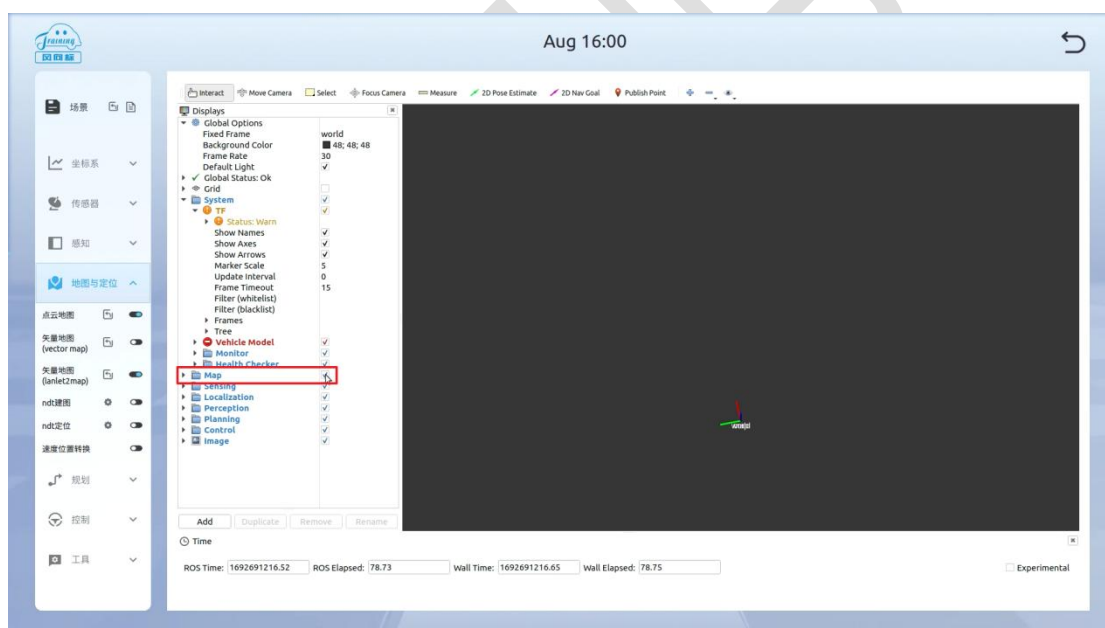


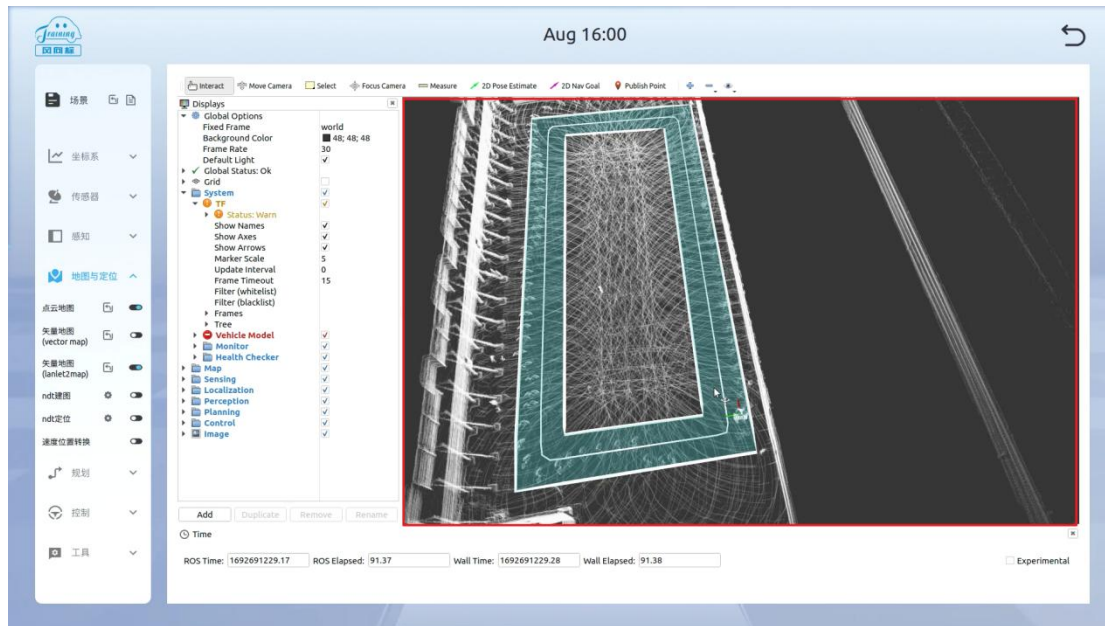
5) 找到地图与定位下的“点云地图”和“矢量地图 (lanelet2map)”功能，勾选启动点云地图和矢量地图 (lanelet2map)，后台会启动对应的地图加载程序（如果没有导入 3.6 小节导出的参数，那这里的地图需要点击旁边的配置按钮，手动导入对应的地图文件）；



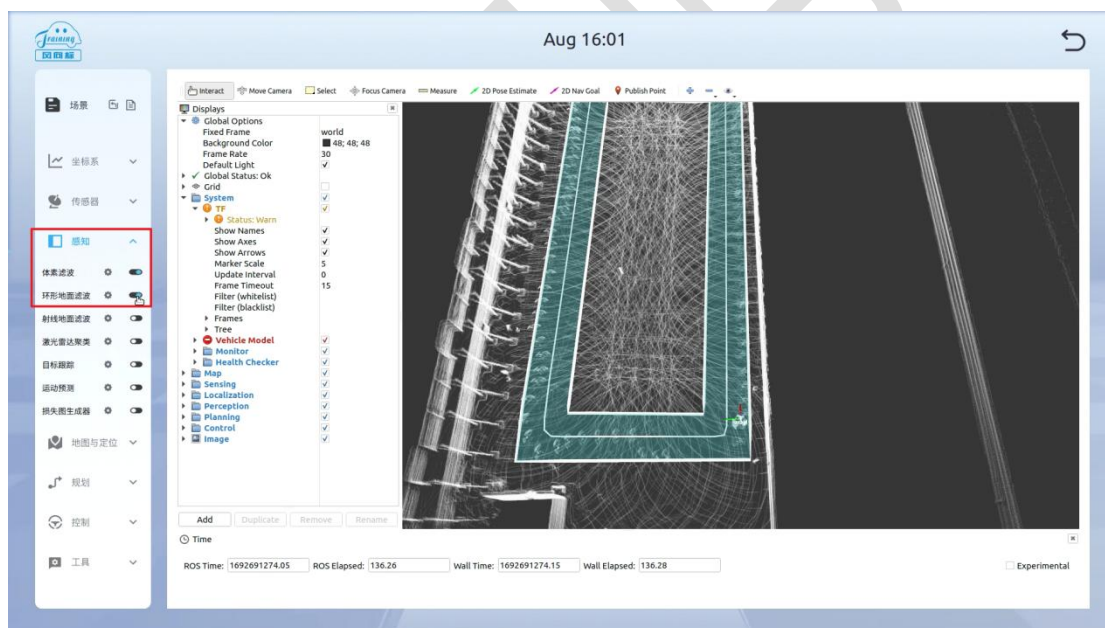


6) 等待几秒后重启地图显示插件，当出现白色点云和青色车道表示地图加载成功；

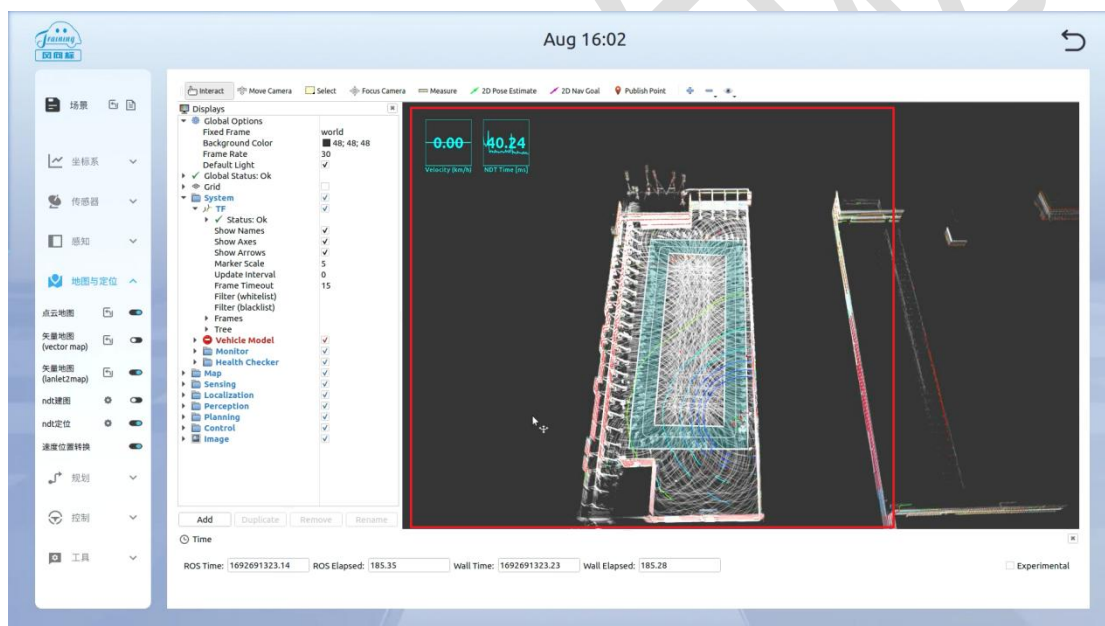
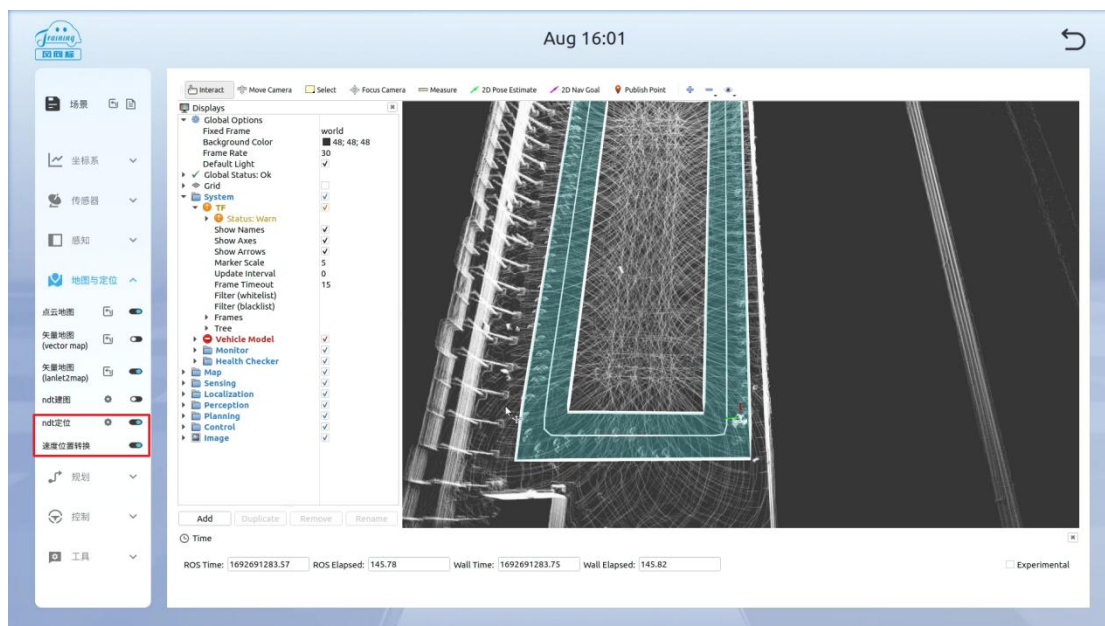




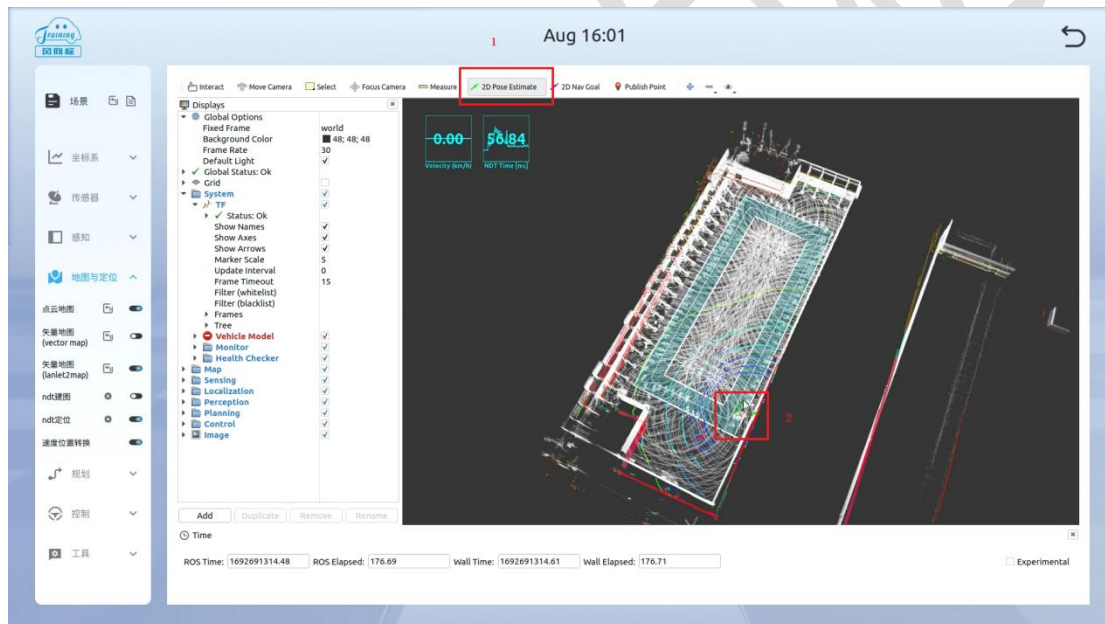
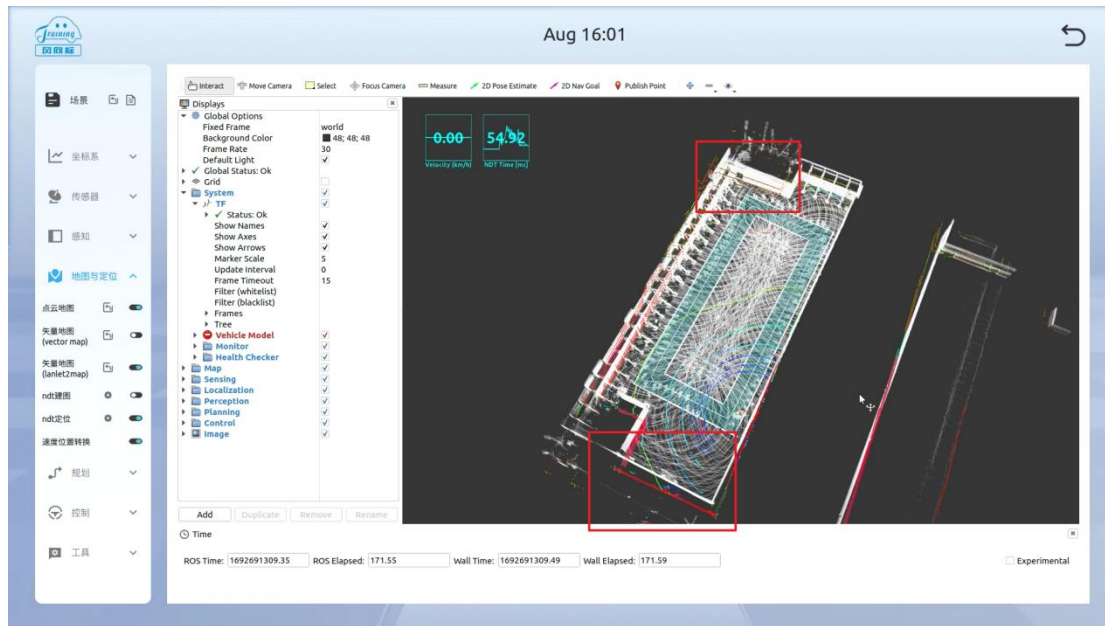
7) 找到“感知”下的“体素滤波”和“环形地面滤波”功能并勾选启动，后台会启动对应的算法程序；



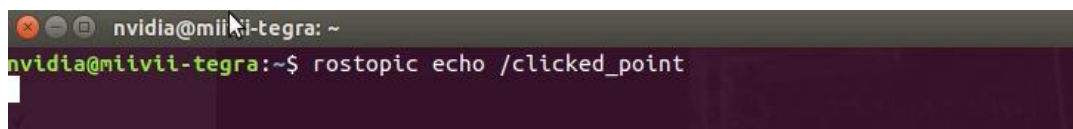
8) 找到“地图与定位”下的“ndt 定位”和“速度位置转换”功能并勾选启动，后台会使用当前的点云数据和 PCD 地图进行匹配定位车辆的位置（默认不使用 GNSS 的情况）；



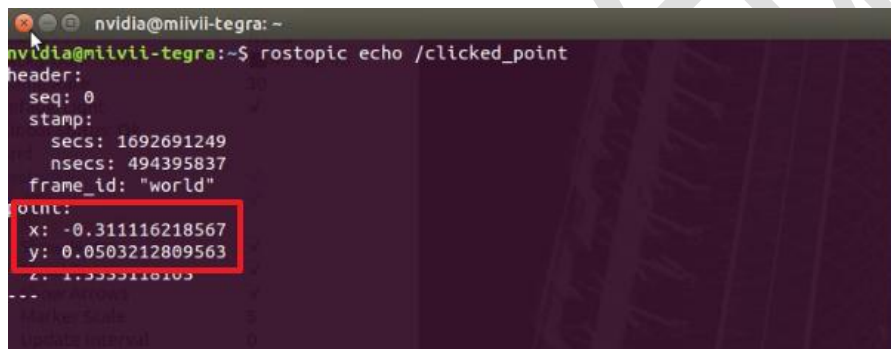
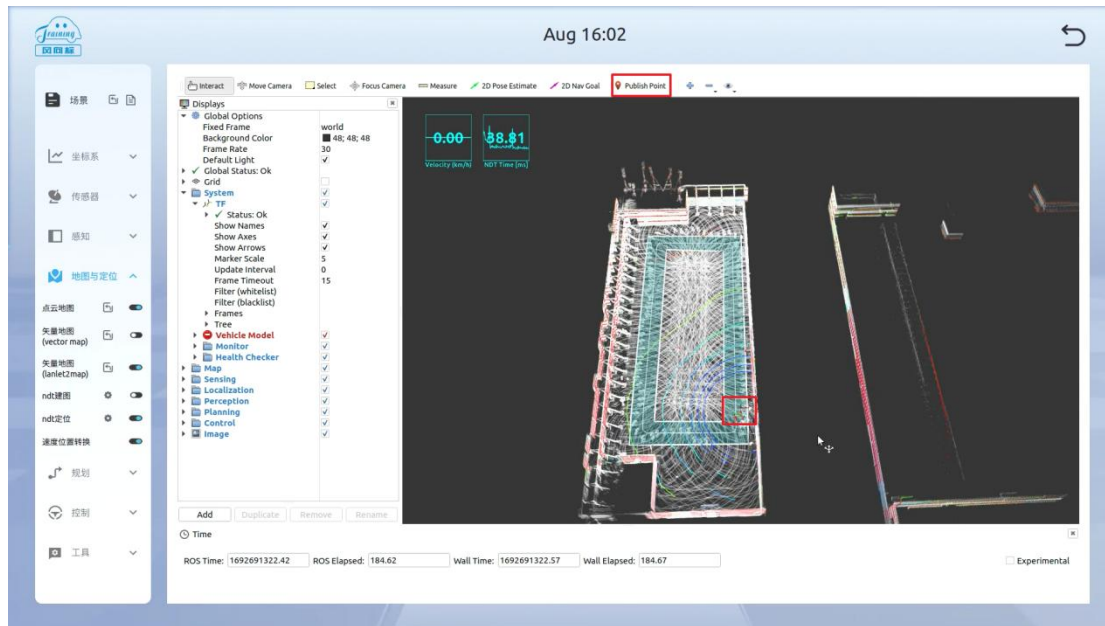
注意！可以通过点云与地图的匹配程度，来判断定位是否准确，如果定位不准，请先取消勾选 **ndt** 定位后，确认车辆移动到起始点位置后，再勾选启动 **ndt** 定位，也可以点击“**2D Pose Estimate**”按钮手动给车辆设置起始点（此方法难度较大）；



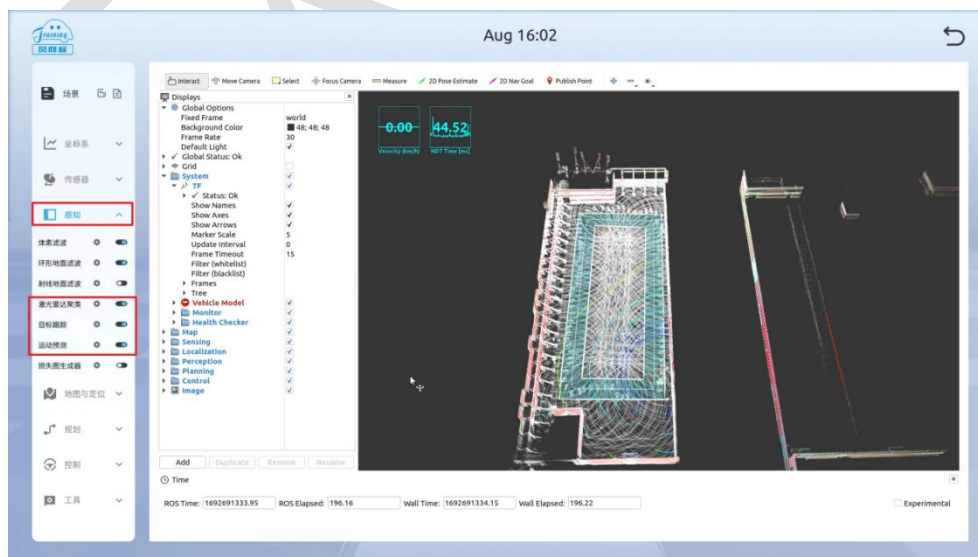
9) 使用快捷键 **ctrl+alt+t** 打开终端，输入命令 **rostopic echo /clicked_point** 后按下回车；

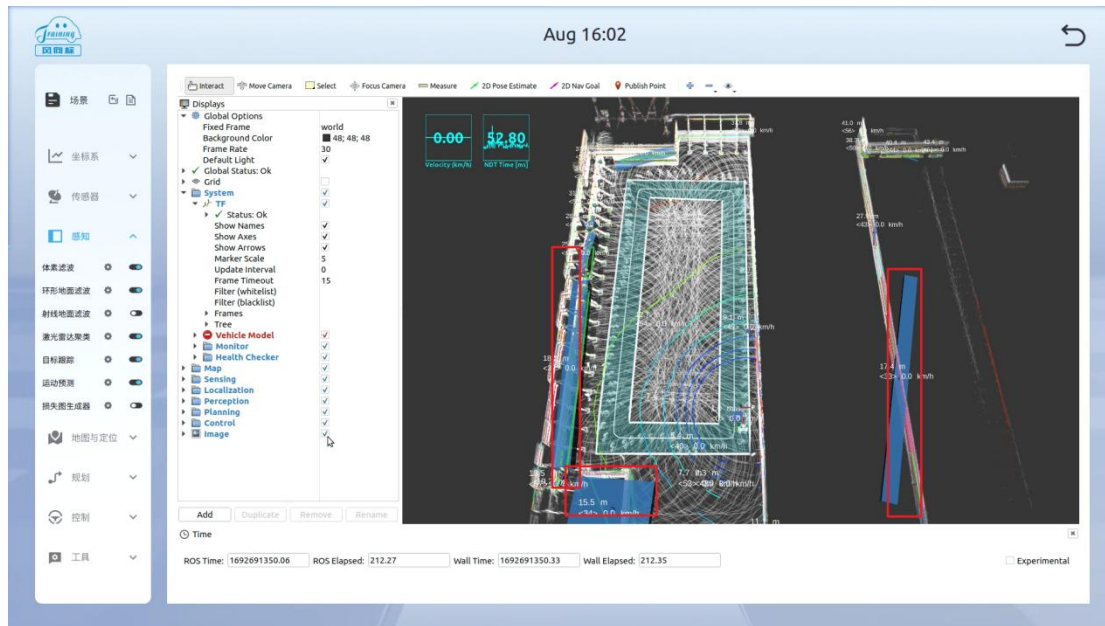


10) 点击“Publish Point”按钮后点击三色坐标，使用快捷键 **alt+tab** 切换到步骤 9 打开的终端，记录 xy 起始点坐标到报告单上；

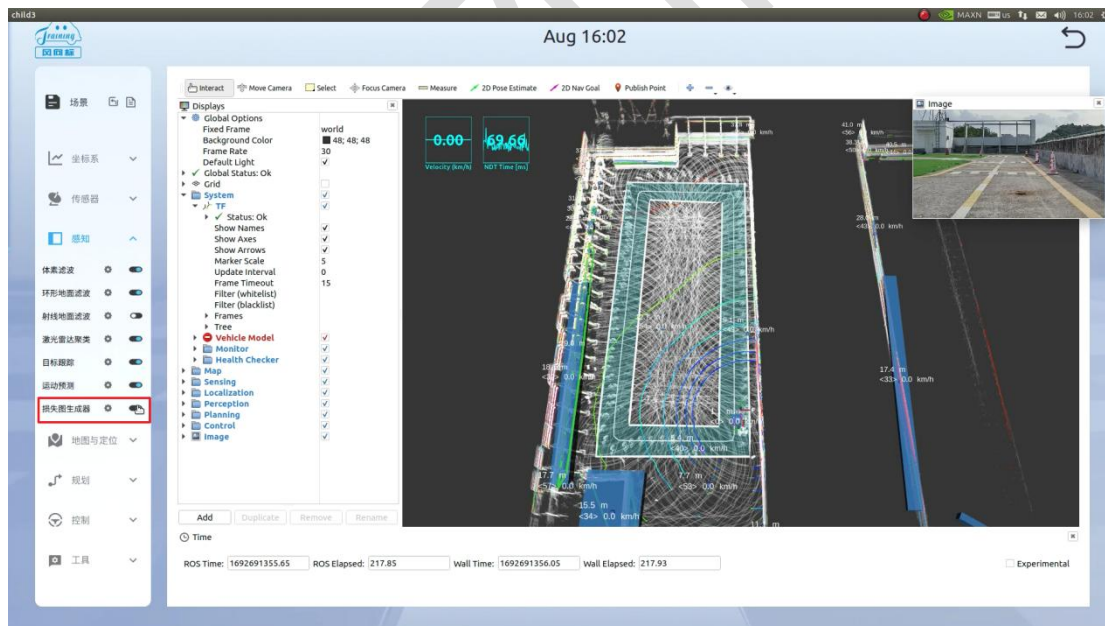


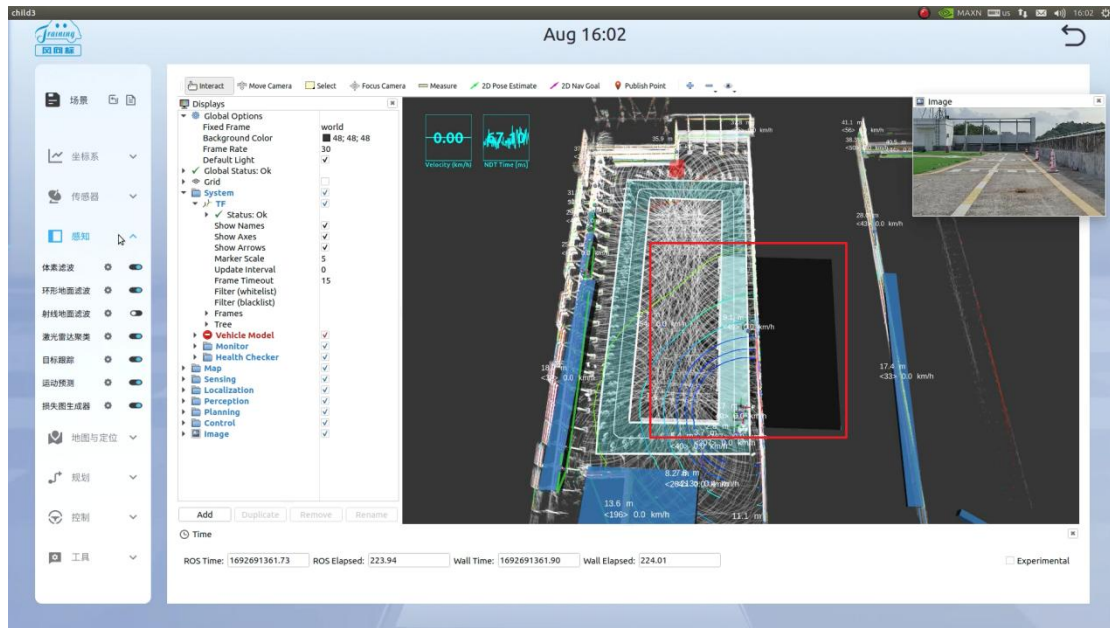
11) 找到“感知”下的“激光雷达聚类”、“目标跟踪”和“运动预测”功能并勾选启动，等待几秒钟后，在高精地图上会看到物体检测跟踪的效果，说明后台算法启动成功；





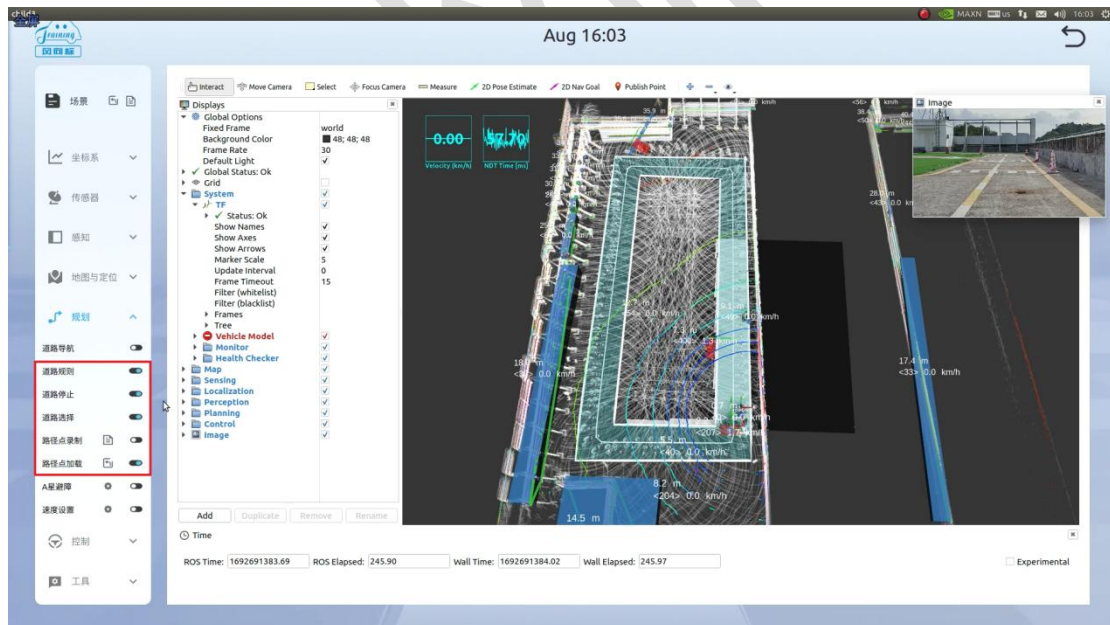
12) 找到“感知”下的“损失图生成器”功能并勾选启动，等待几秒后会出现黑白方形的地图出现，此地图被称为代价地图（损失地图），用于为路径规划算法提供场景信息，当出现代价地图时说明后台算法启动成功；

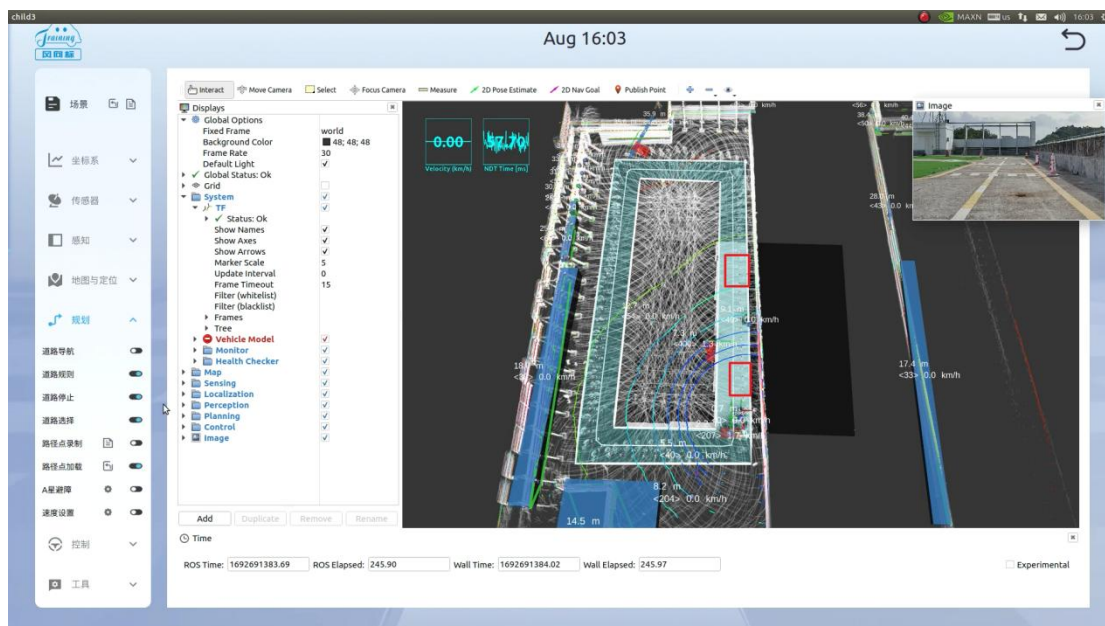




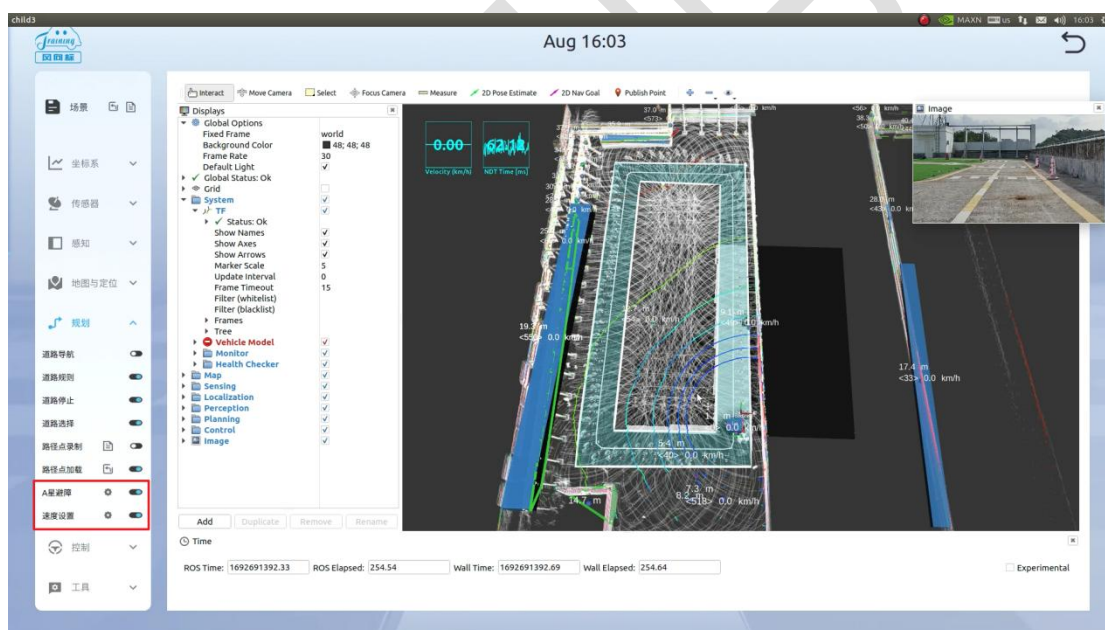
13) 找到“规划”下的“路径点加载”，勾选启动（如果没有导入 3.6 小节导出的参数，则需要点击旁边的图标，选择需要加载的 csv 文件）；

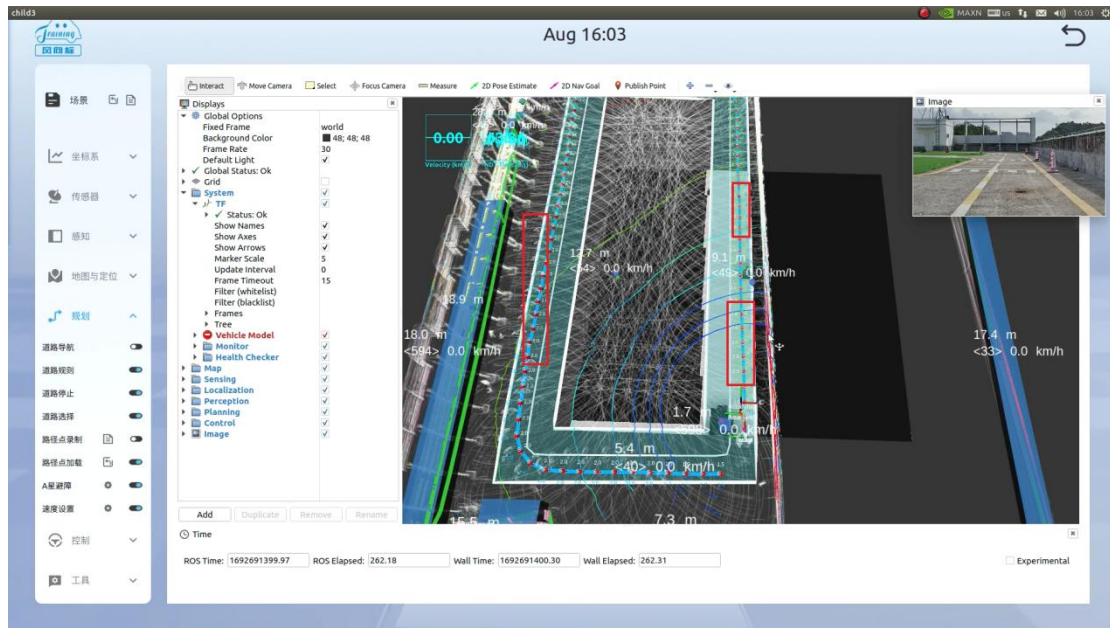
14) 找到“规划”下的“道路选择”、“道路停止”和“道路规则”功能并勾选启动，等待几秒后，看到路径点出现说明后台算法启动成功；



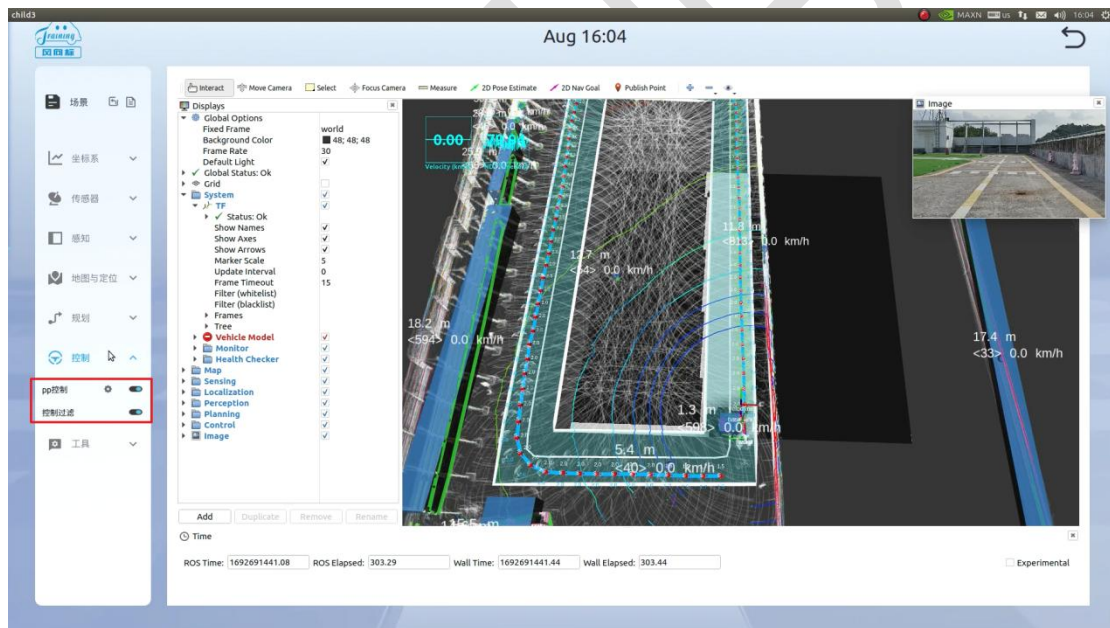


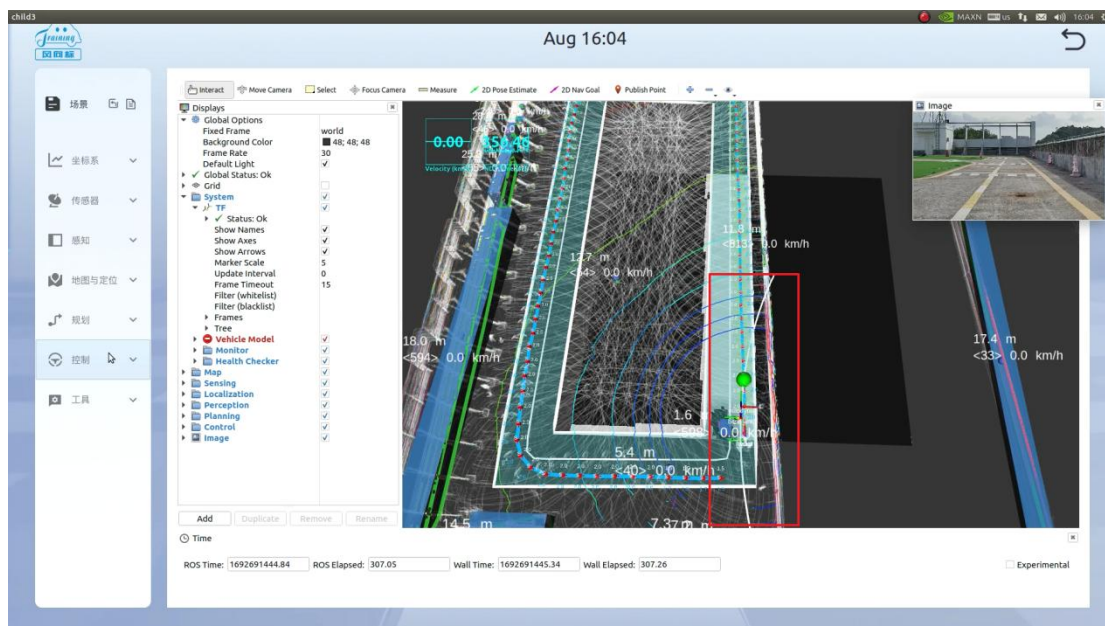
15) 找到“规划”下的“A星避障”和“速度设置”功能并勾选启动，等待几秒后，看到路径点被一条蓝色线连接时，说明后台算法启动成功；



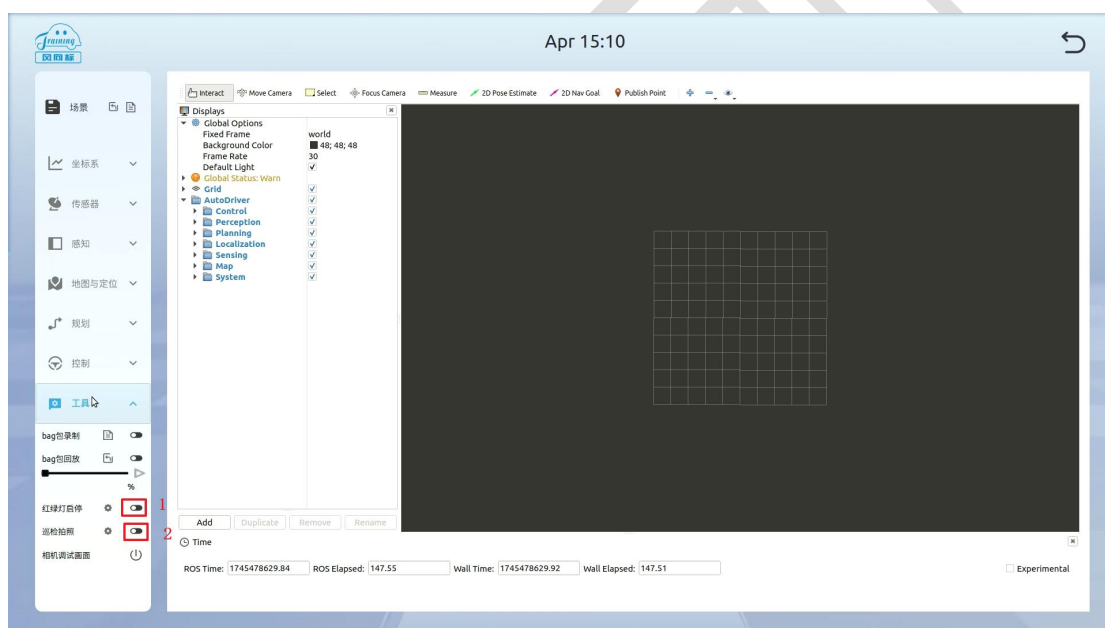


16) 找到“控制”下的“PID 控制”和“控制过滤”功能并勾选启动，当出现转向弧线和跟踪大绿点说明后台算法启动成功；





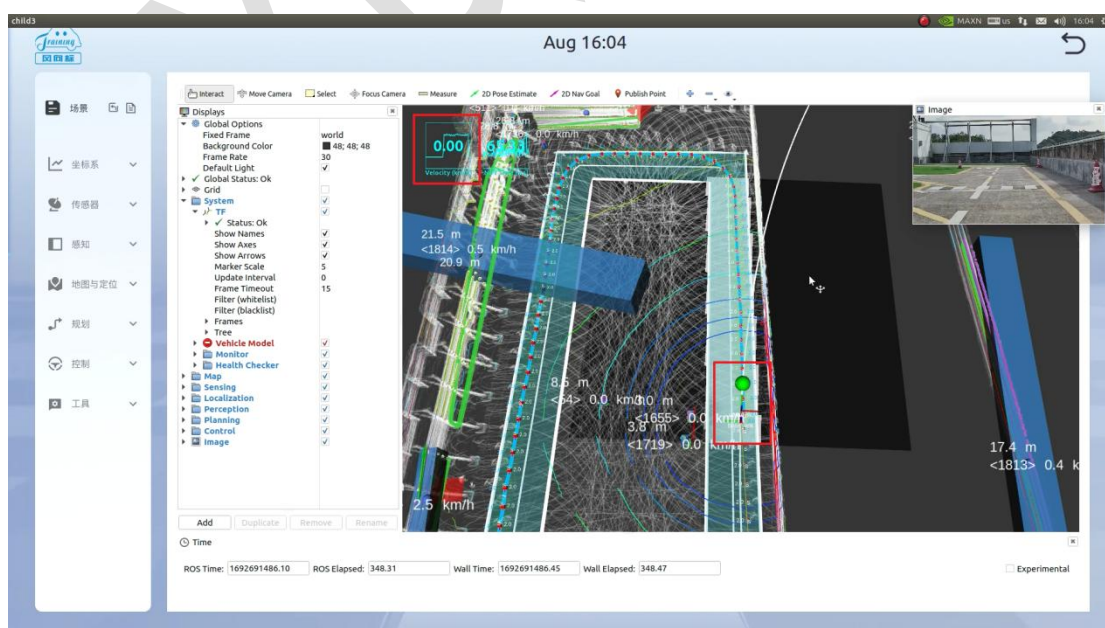
17) 找到“工具下”的“红绿灯启停”和“巡检拍照”功能并勾选启动；

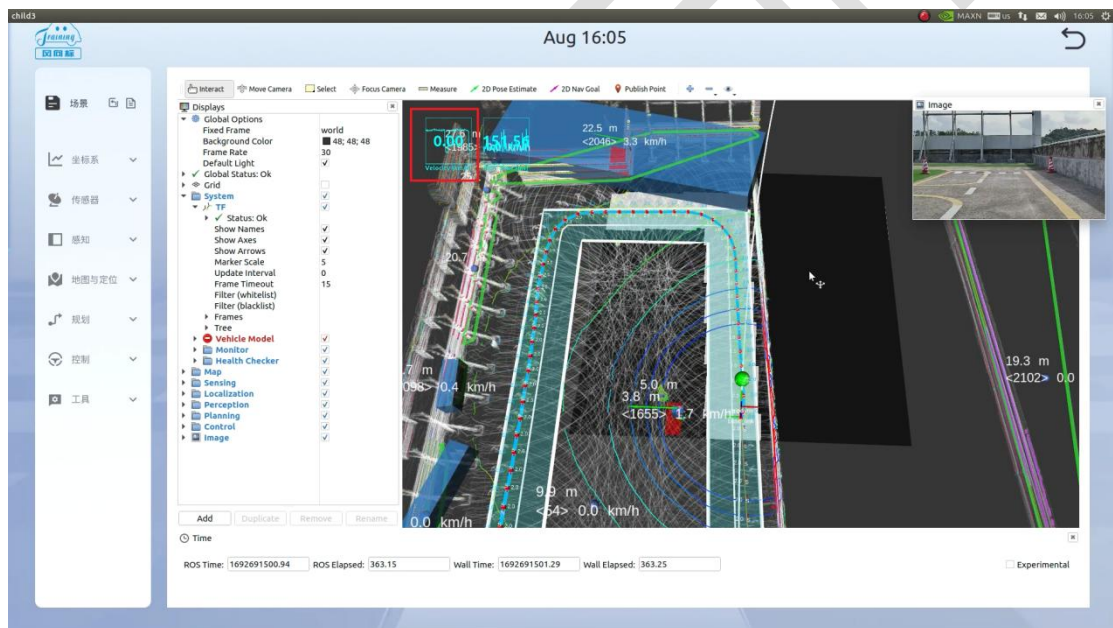


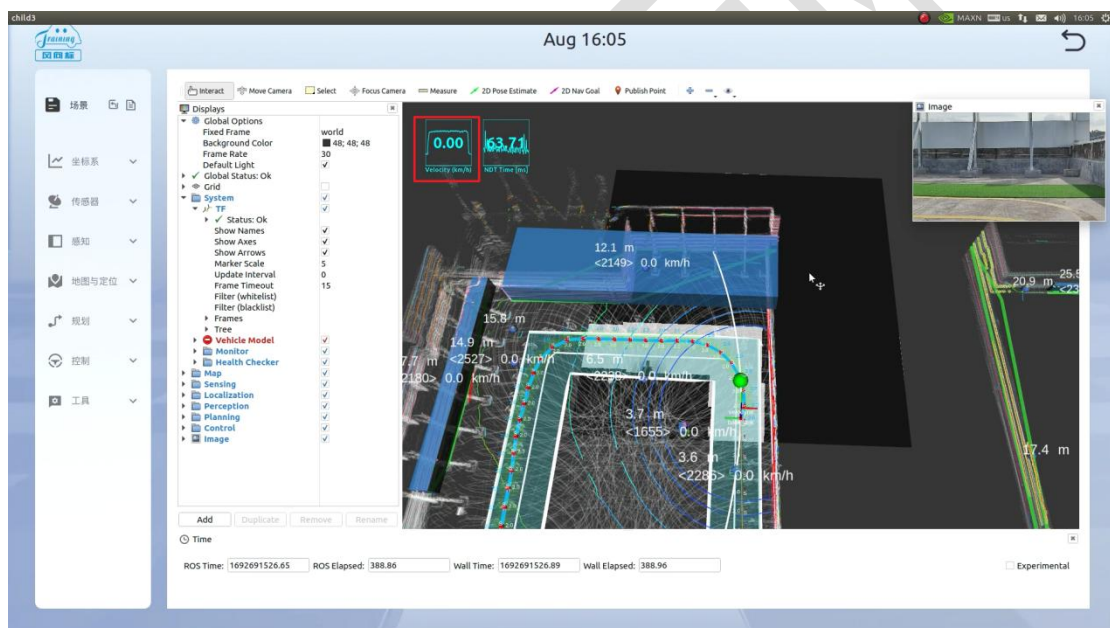
18) 将遥控器控制模式设置为自动驾驶模式，车辆将会按照指定路径行驶；



到达每个巡检点时会停留 3 秒并采集此刻单目相机的视觉画面

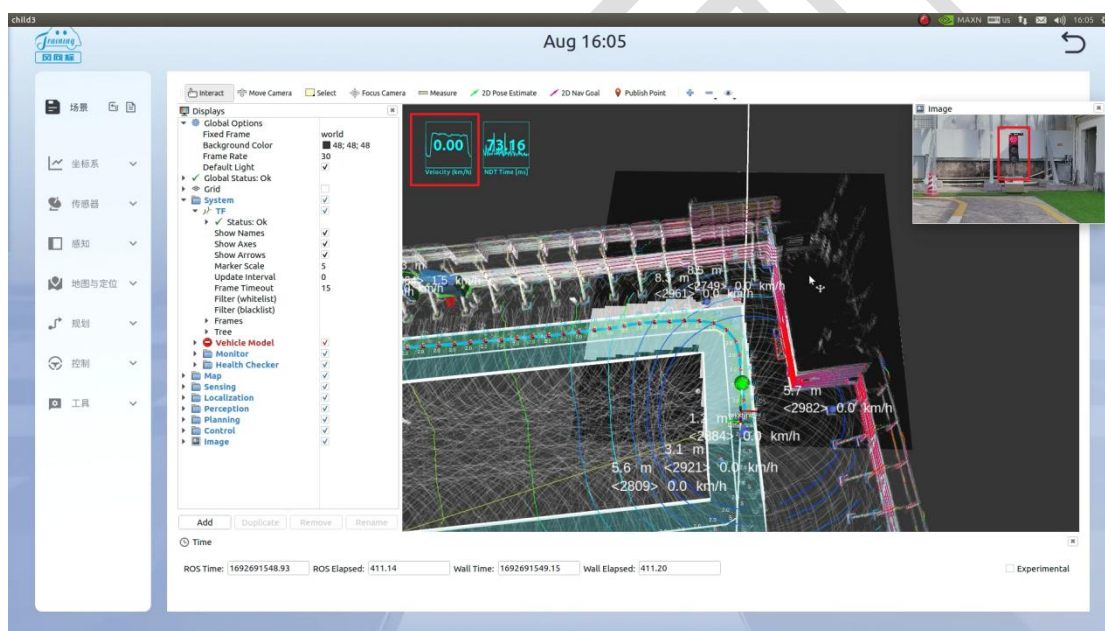




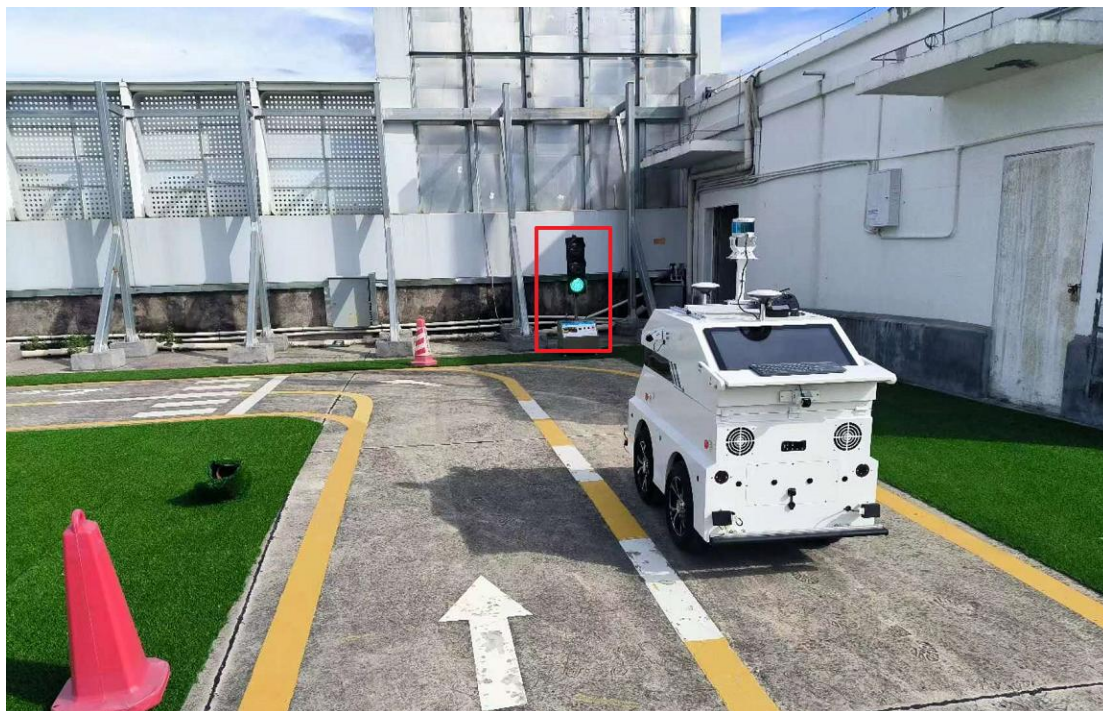




在红绿灯检测区域内，检测到红灯时车辆停车，绿灯时车辆行驶

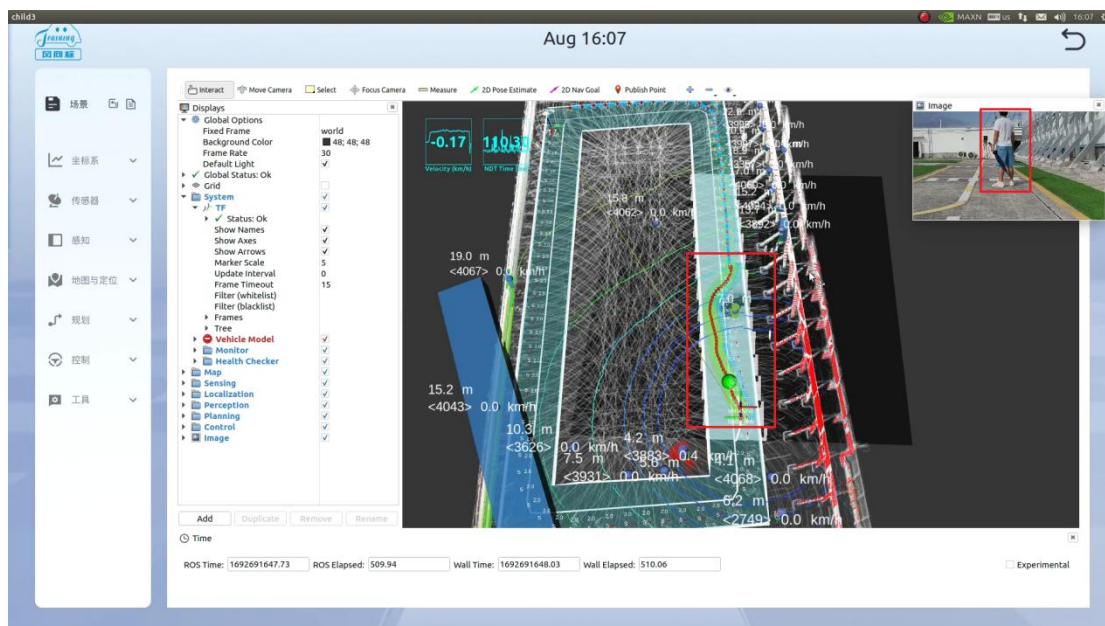






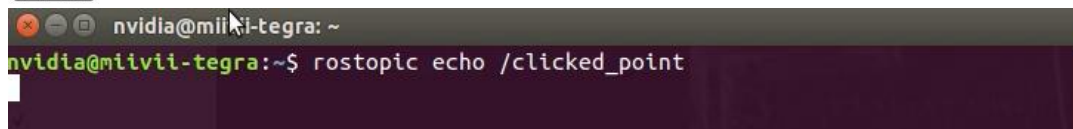
当车辆前方遇到障碍物时，路径规划算法开始计算避障路径，控制算法会根据避障路径行驶，最终避开障碍物



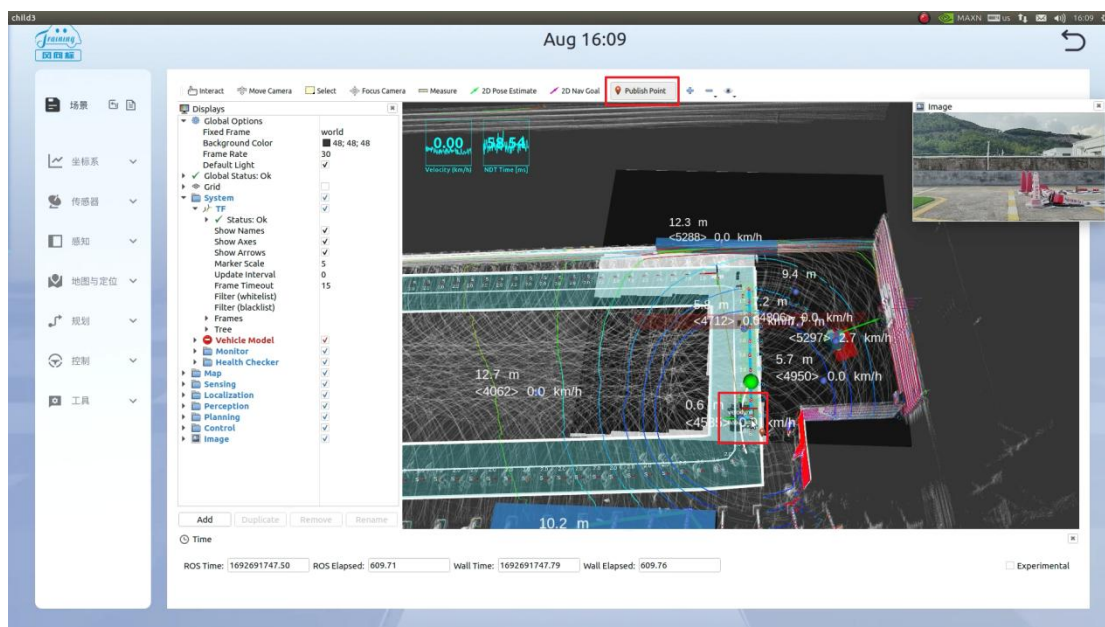




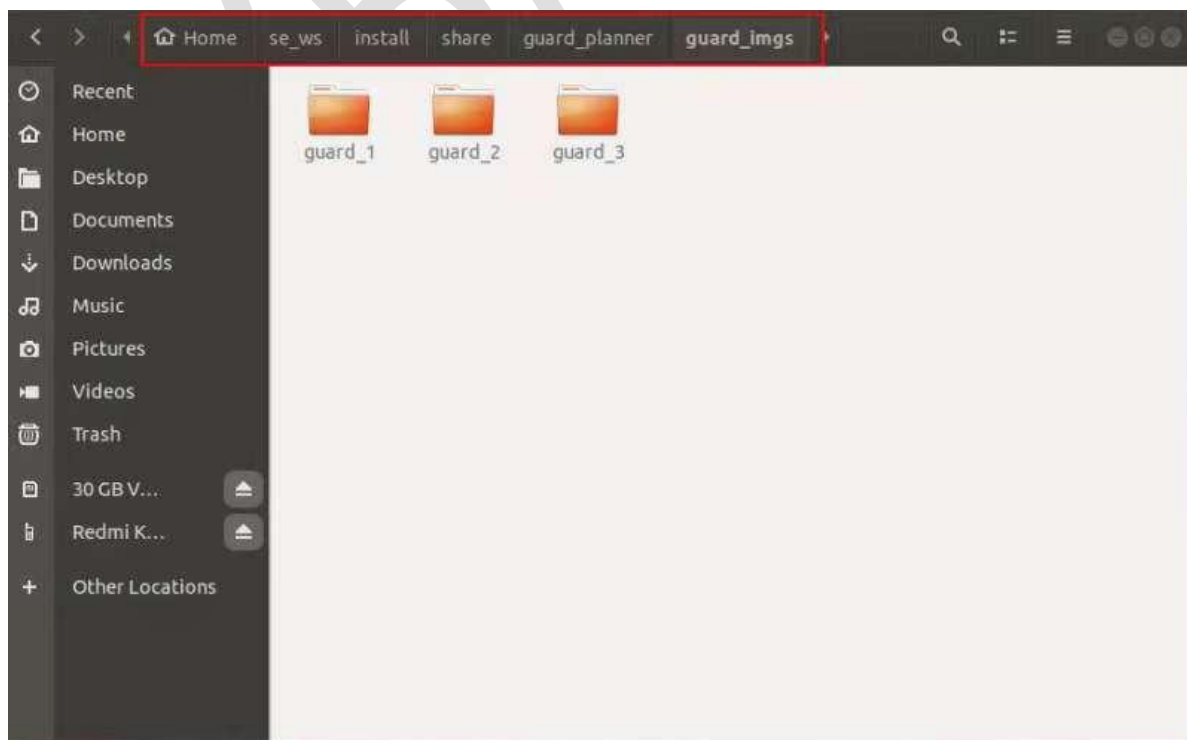
19)当车辆行驶到终点时,使用快捷键 `ctrl+alt+t` 打开终端,输入命令 `rostopic echo /clicked_point` 后按下回车;



20) 点击“Publish Point”按钮后点击三色坐标，使用快捷键 **alt+tab** 切换到步骤 19 打开的终端，记录 **xy** 终点坐标到报告单上；



21) 找到 `/home/nvidia/driver_ws/install/share/guard_planner/guard_imgs` 文件夹，里面有到达三个巡检点时采集单目视觉画面的图像，记录每个巡检点采集图像的数量到报告单上；





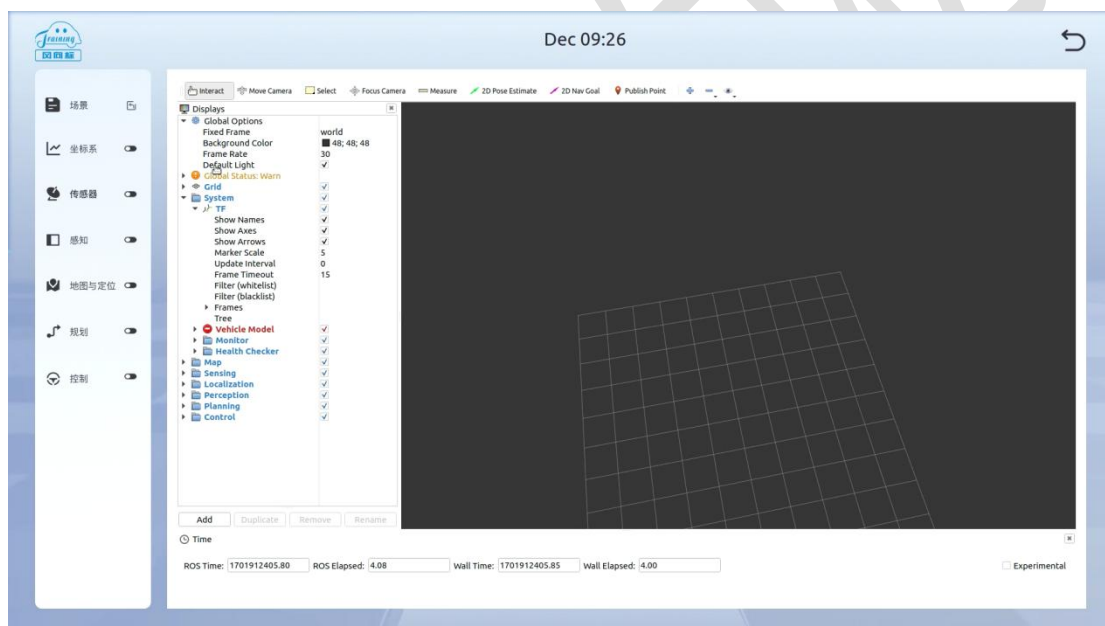
3.8 道路测试-初级教学-方案一

0) 在关闭所有软件和终端后再操作本小节；

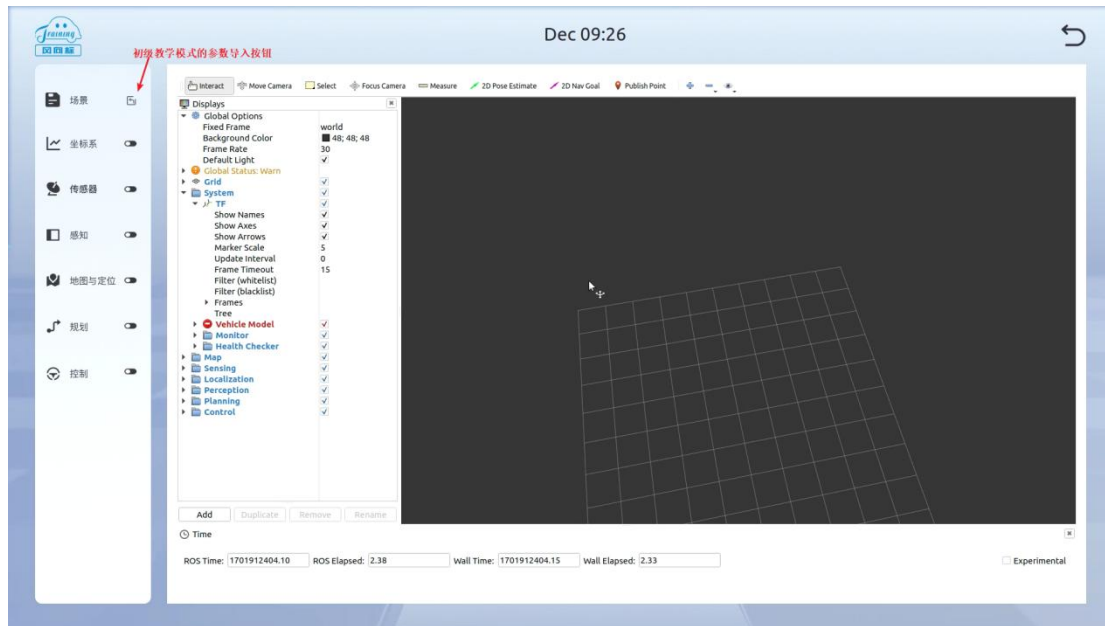
1) 使用遥控器将车辆行驶到起始区域；



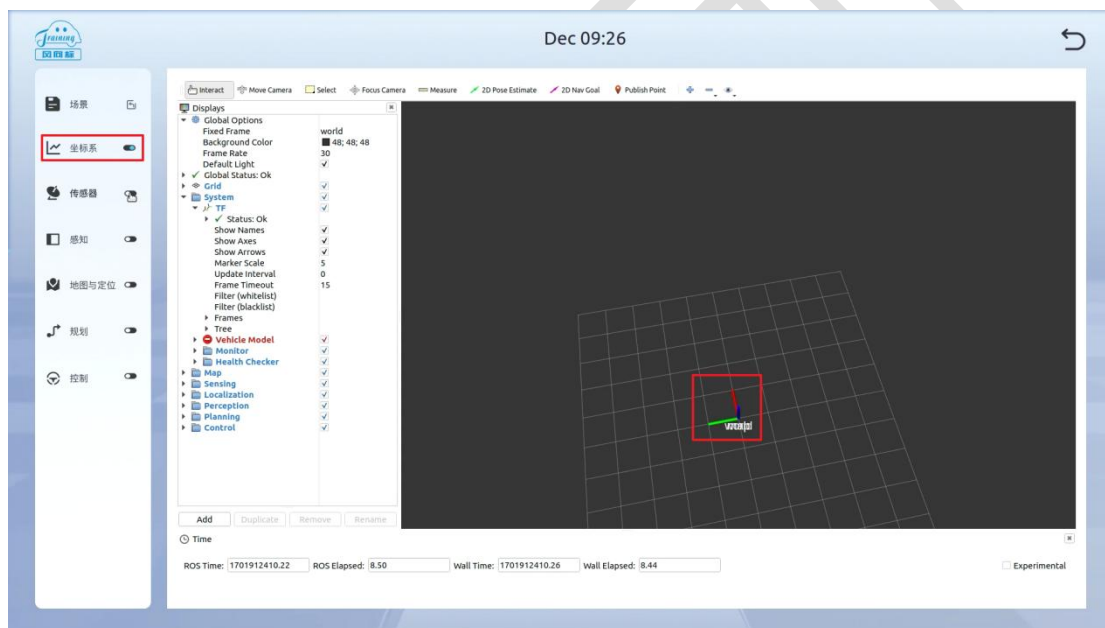
2) 打开自动驾驶教学软件，选择“初级教学”模式；



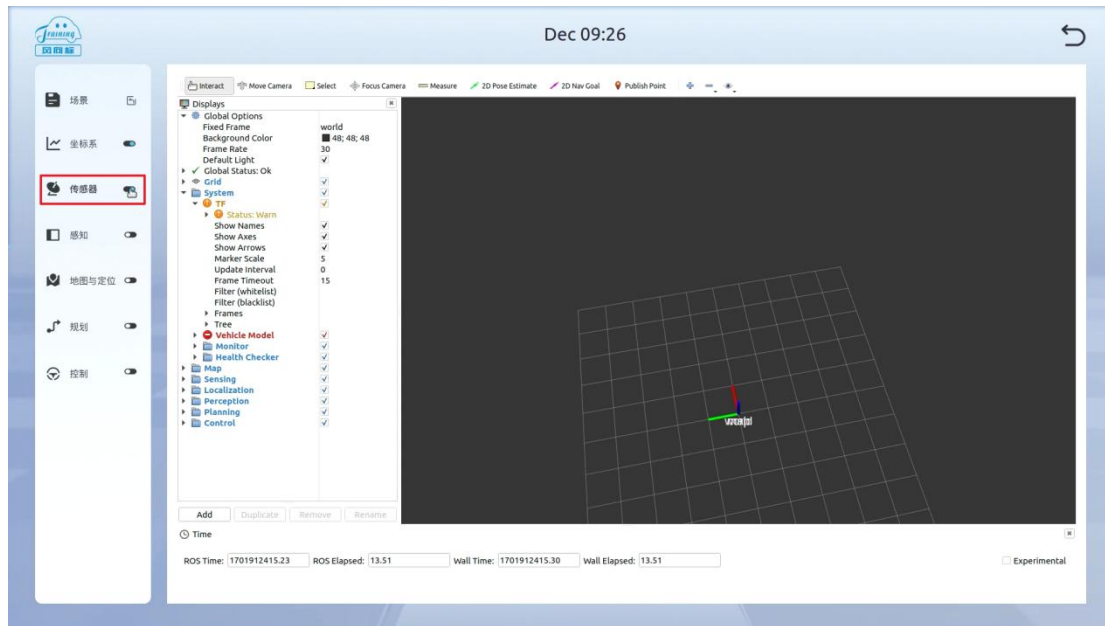
3) 导入从 3.6 小节导出的参数，选择方案一；



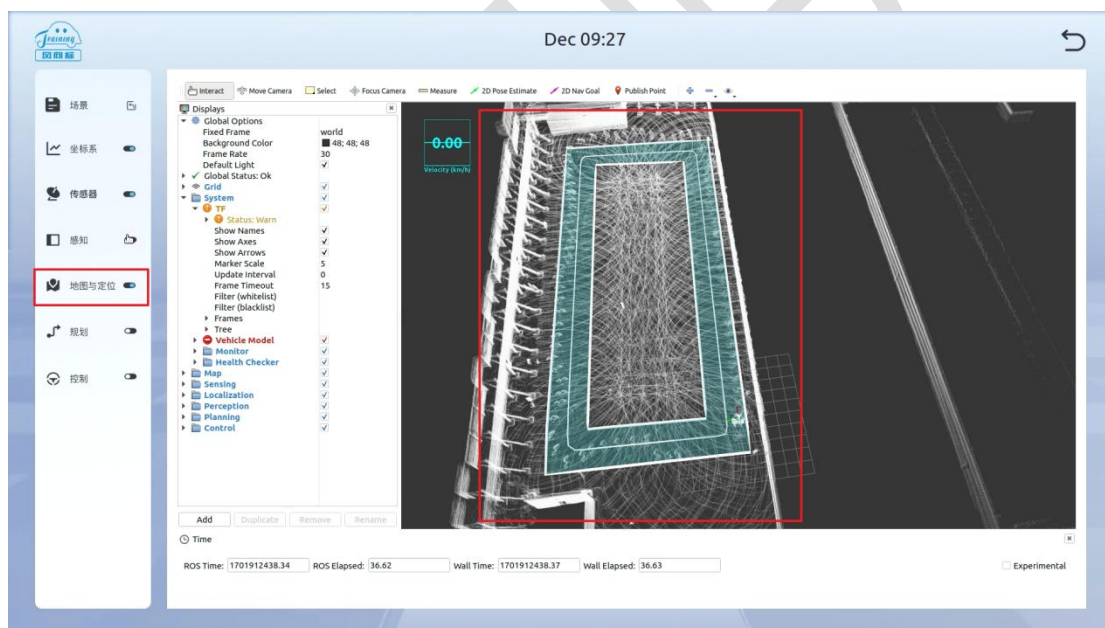
4) 勾选启动“坐标系”，启动成功后会出现“三色坐标系”；



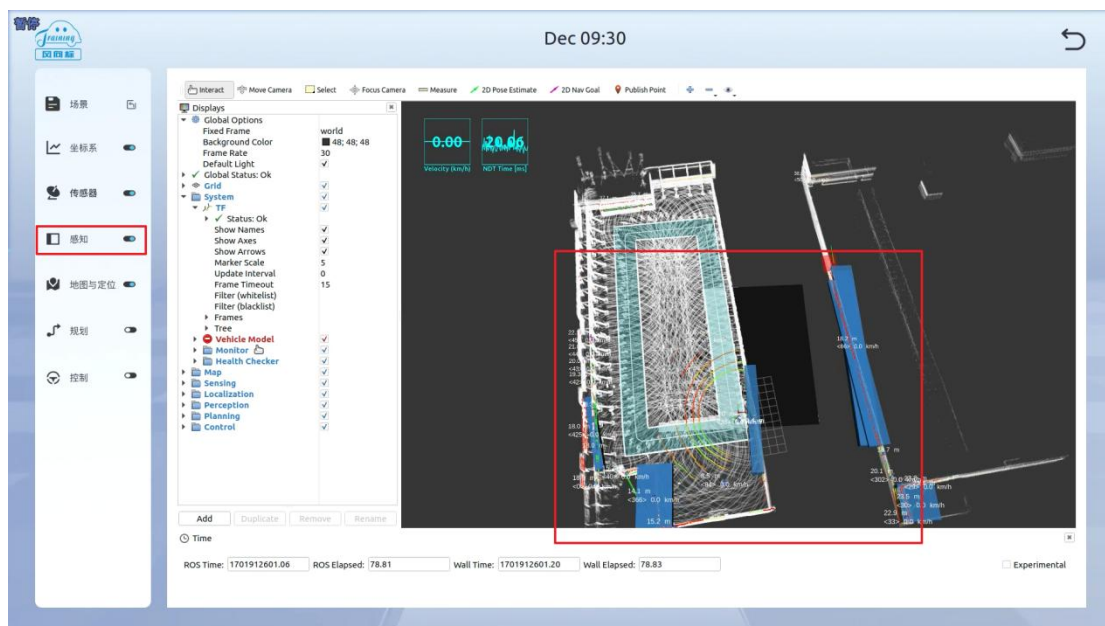
5) 勾选启动“传感器”；



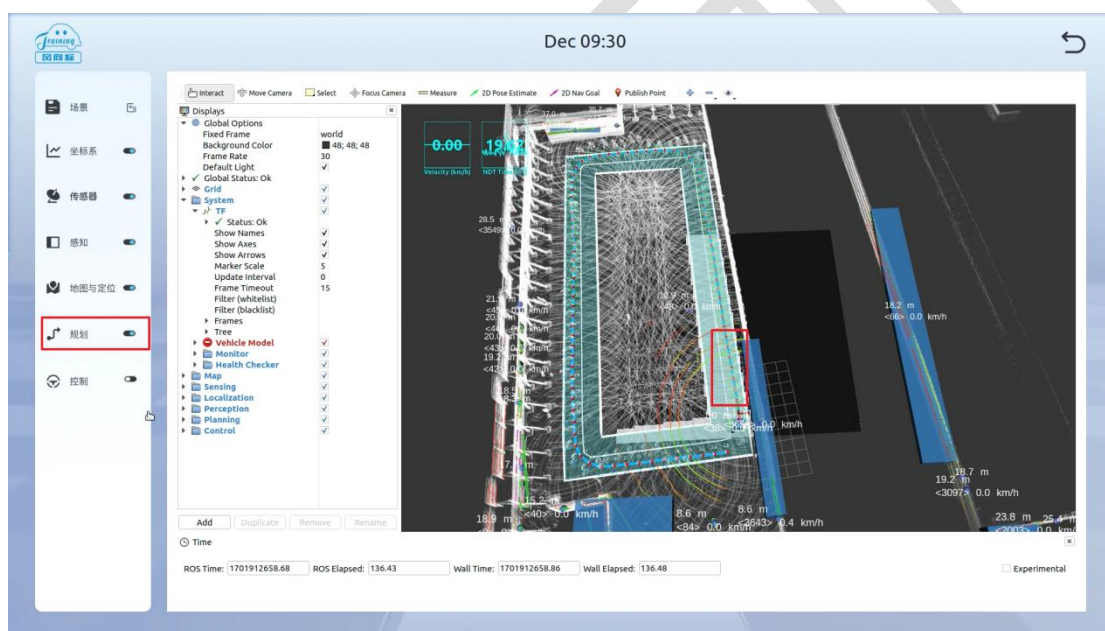
6) 勾选启动“地图与定位”，启动成功后会出现白色的 PCD 地图和青绿色的 HD 高精地图；



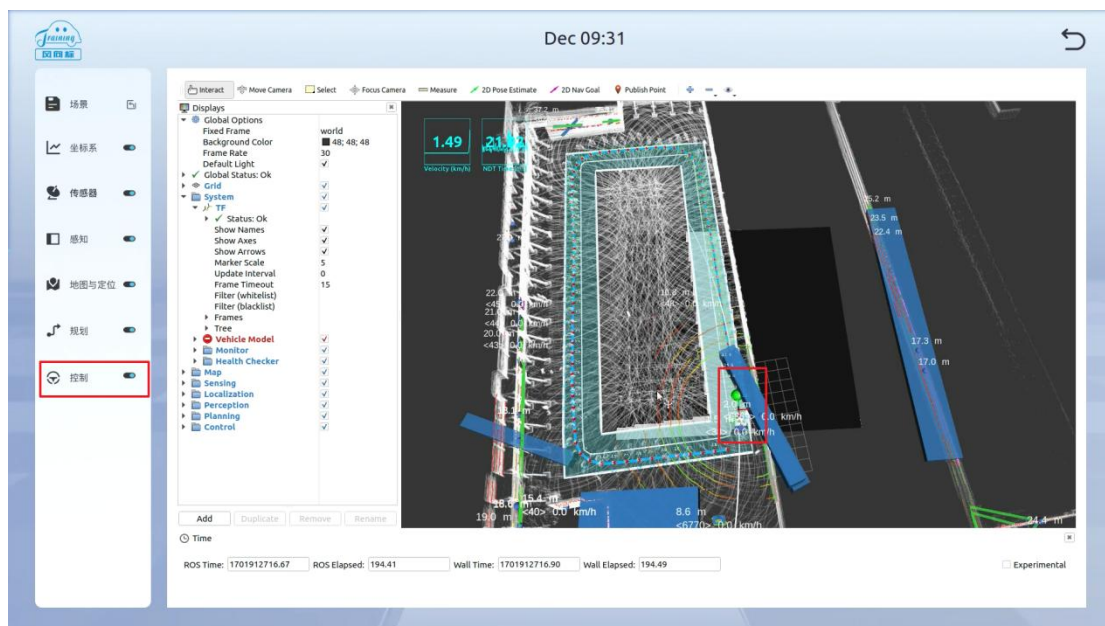
7) 勾选启动“感知”，启动成功后会出现蓝色矩形和黑白地图；



8) 勾选启动“规划”，启动成功后会出现蓝色全局路线；



9) 勾选启动“控制”，启动成功后会出现大绿点和转向弧线；



10) 将遥控器控制模式设置为自动驾驶模式，车辆将会按照指定路径行驶；



3.9 道路测试-演示模式-方案一

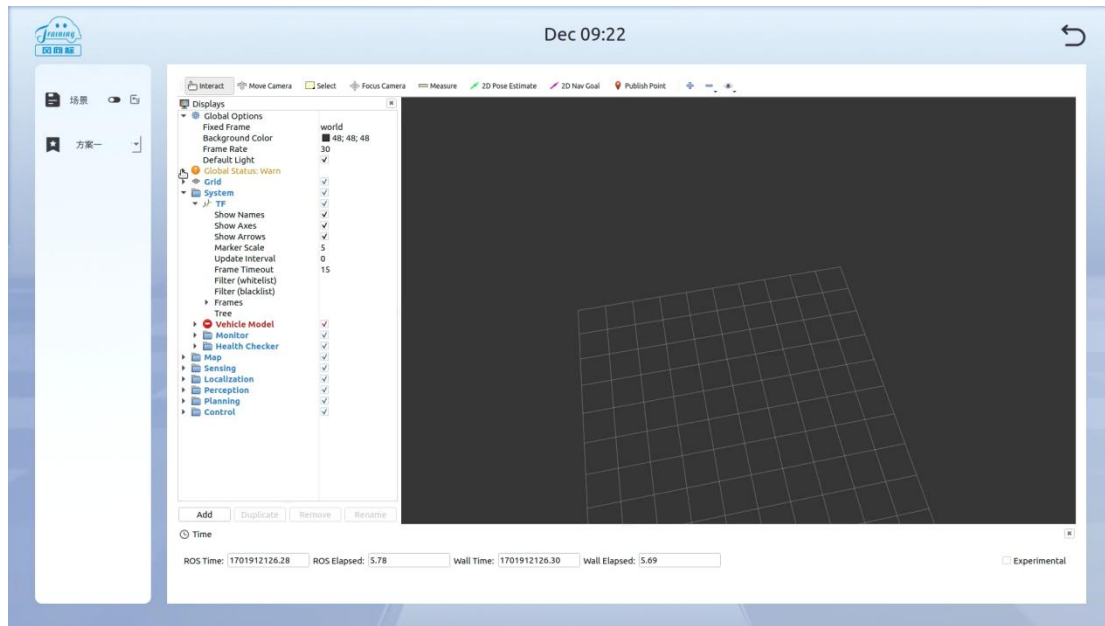
0) 在关闭所有软件和终端后再操作本小节；

1) 使用遥控器将车辆行驶到起始区域；

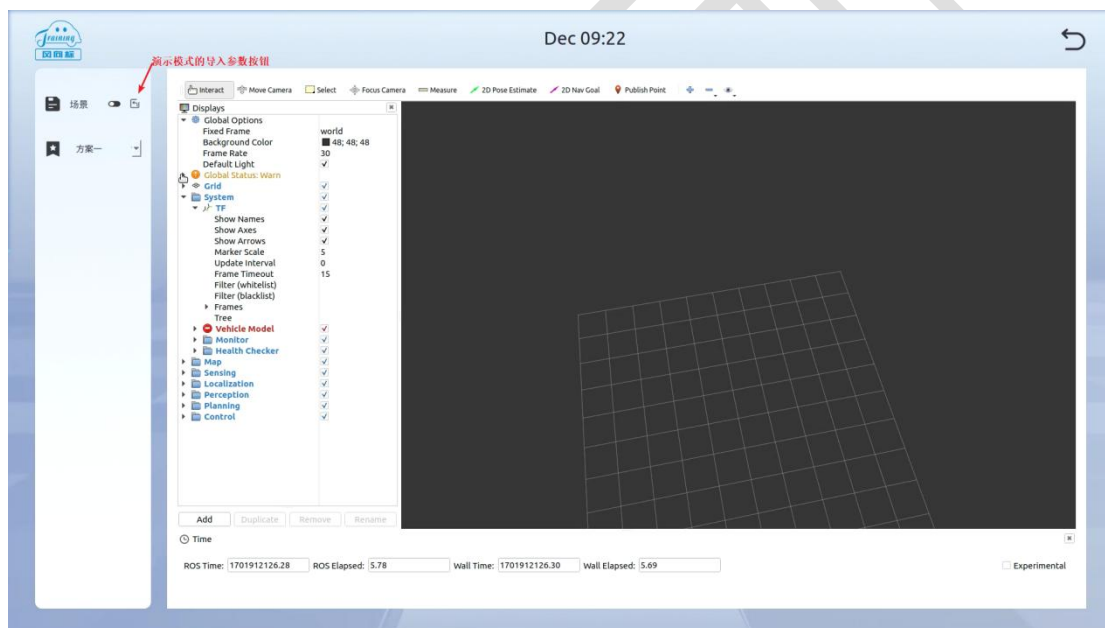


2) 打开自动驾驶教学软件，选择“演示”模式；





3) 导入从 3.6 小节导出的参数，选择方案一；



4) 点击一键启动按钮，启动成功后会出现白色 PCD 地图、青绿色高清 HD 地图、黑白地图、蓝色全局路线、大绿点和转向弧线，说明所有算法启动成功；



5) 将遥控器控制模式设置为自动驾驶模式，车辆将会按照指定路径行驶；



3.10 道路测试-高级教学-方案二

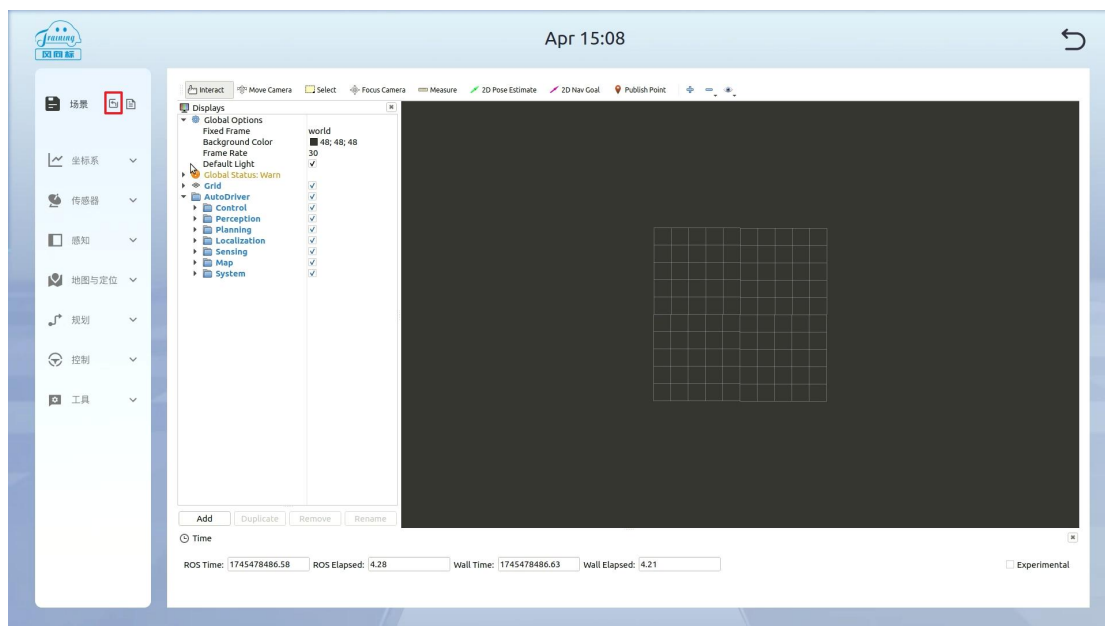
0) 在关闭所有软件和终端后再操作本小节；

1) 使用遥控器将车辆行驶到起始区域；

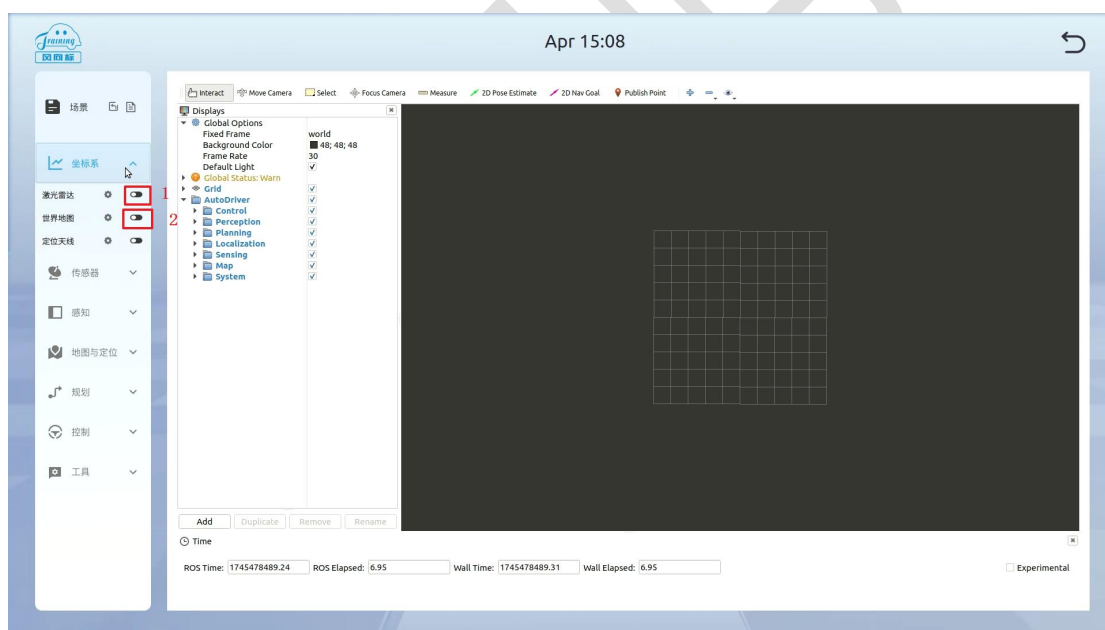


2) 打开自动驾驶教学软件，选择“高级教学”模式，导入从 3.6 小节导出的参数；



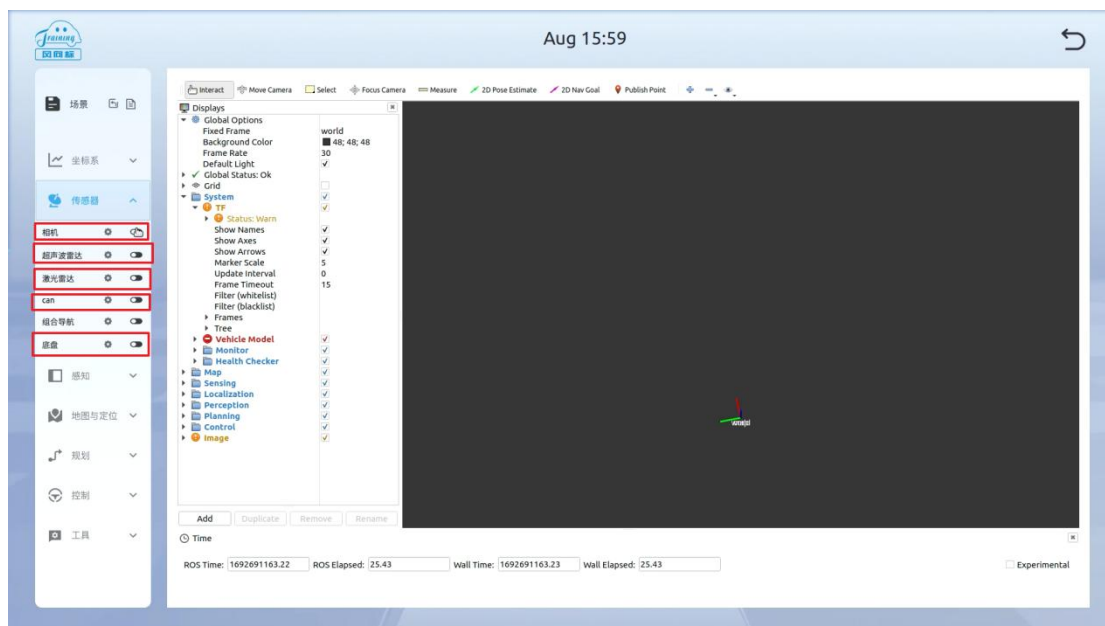


3) 找到“坐标系”下的“激光雷达”和“世界地图”功能并勾选启动，运行成功后会出现三色坐标系；

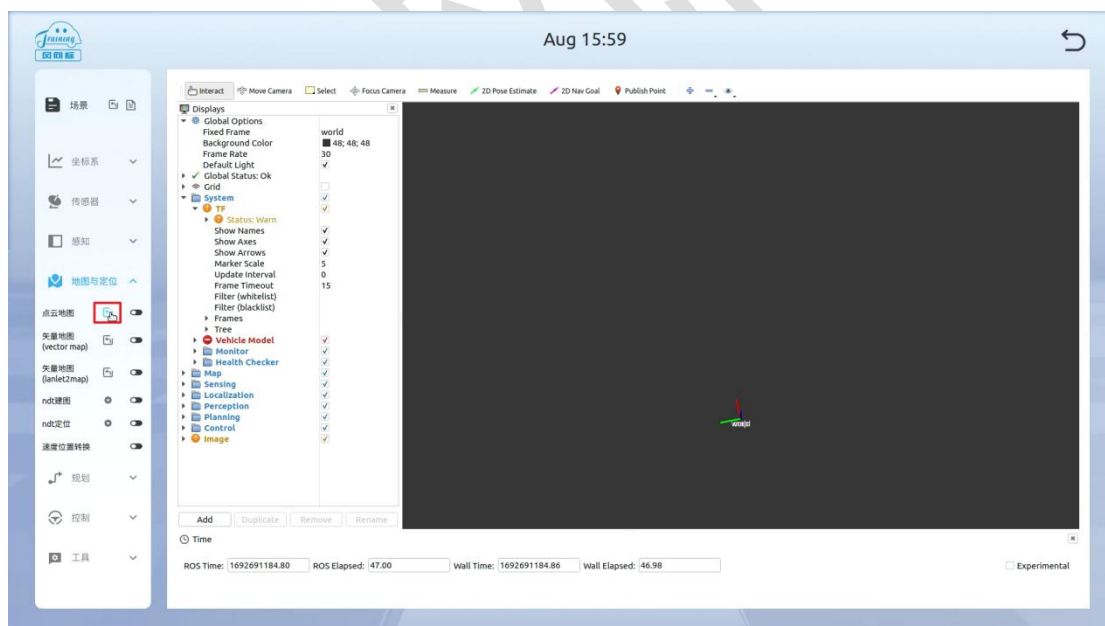


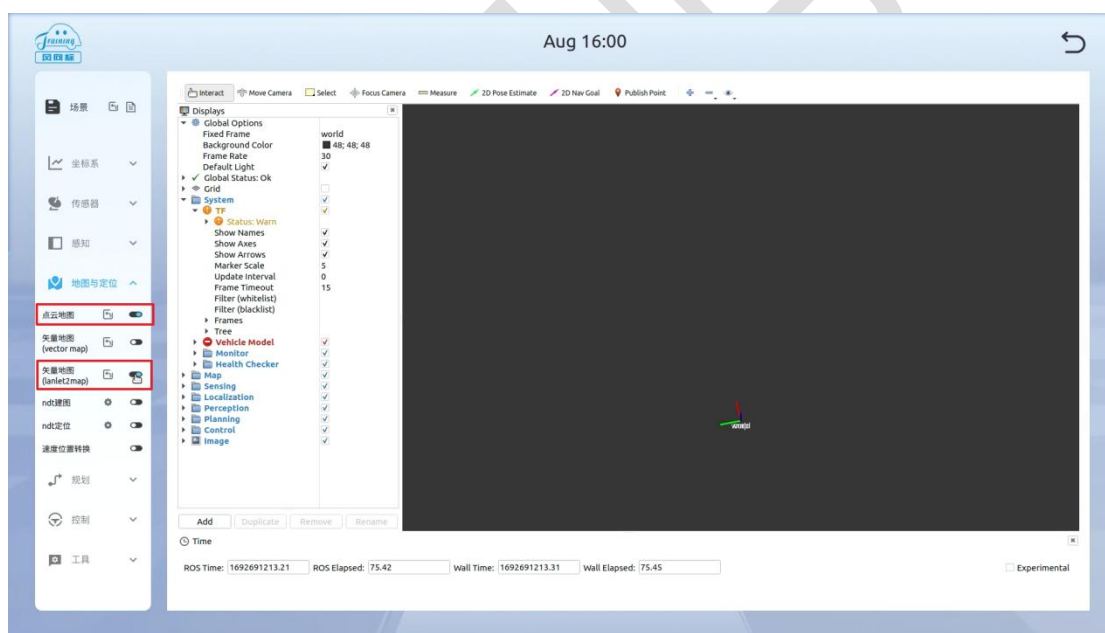
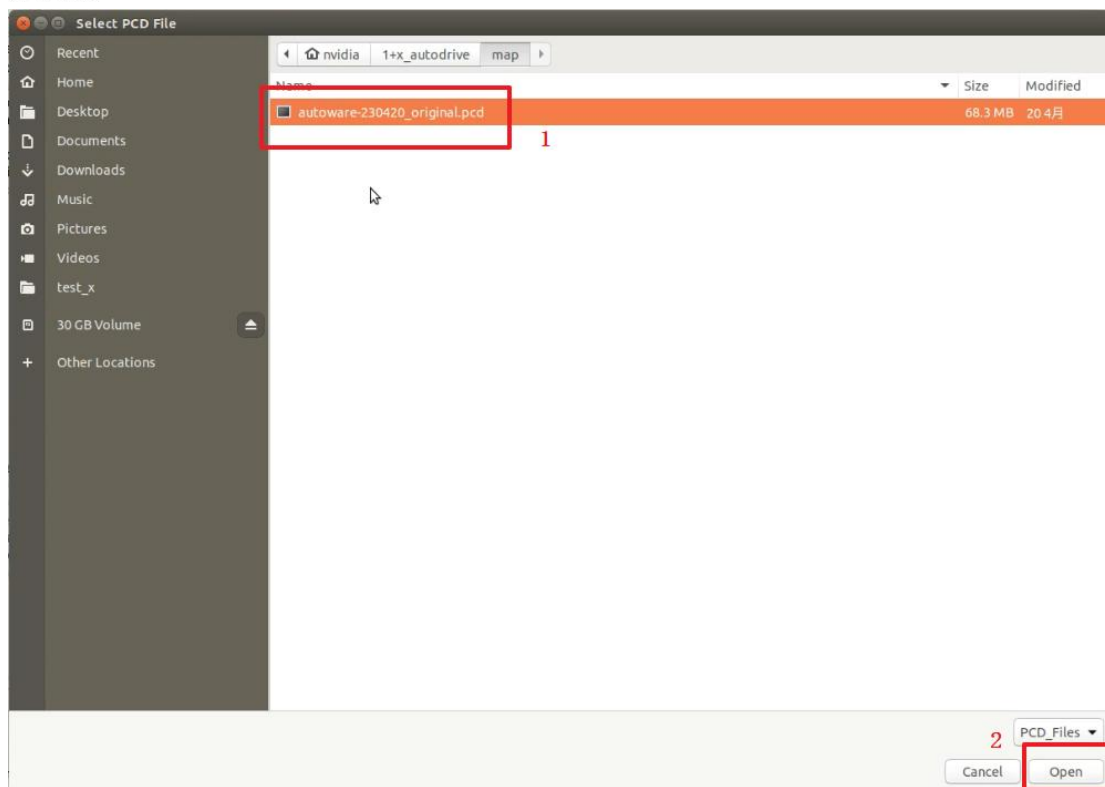
4) 找到传感器下的“相机”、“超声波雷达”、“激光雷达”、“CAN”和“底盘”并勾选启动它们，后台会启动对应的驱动程序；

注意！ 如果没有做 3.6 标记位置或不需要红绿灯识别或巡检点检测，请不要勾选相机。

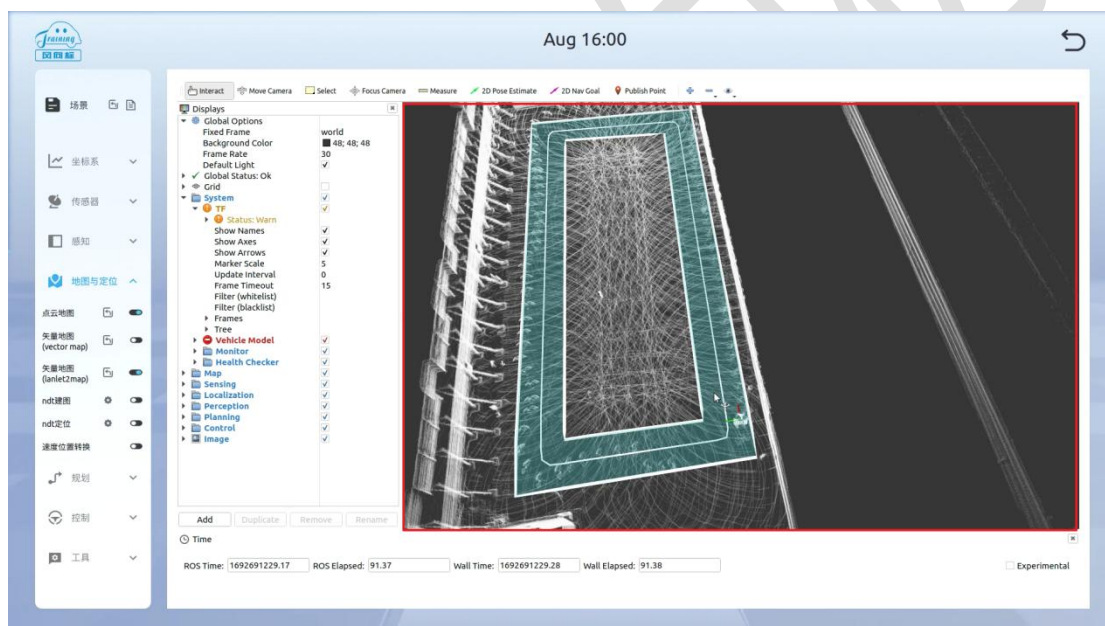
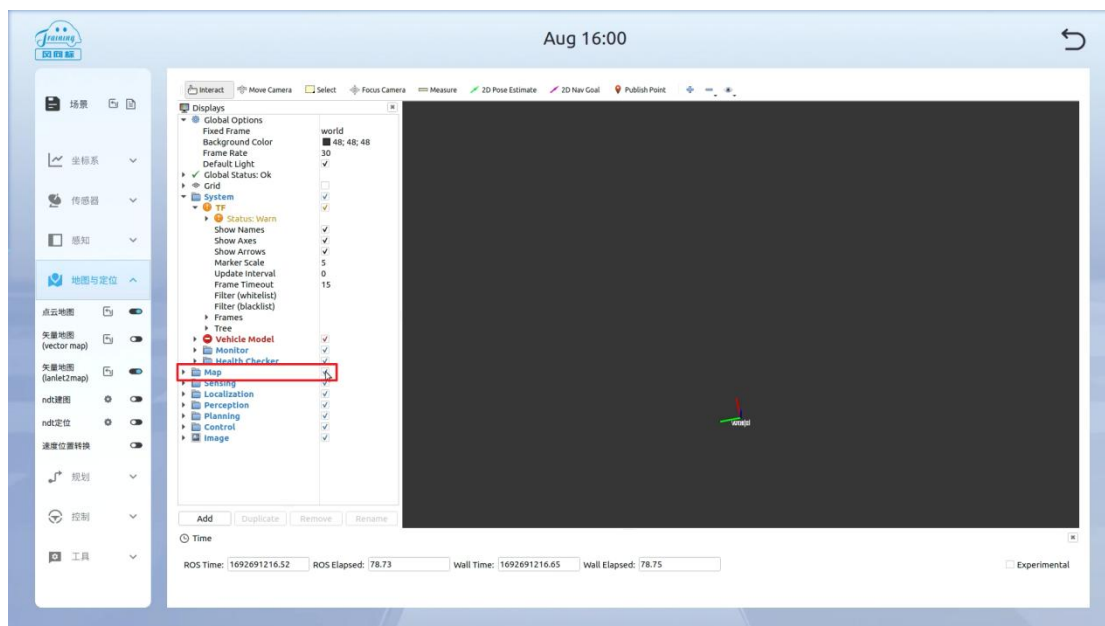


5) 找到地图与定位下的“点云地图”和“矢量地图 (lanelet2map)”功能，勾选启动点云地图和矢量地图 (lanelet2map)，后台会启动对应的地图加载程序（如果没有导入 3.6 小节导出的参数，那这里的地图需要点击旁边的配置按钮，手动导入对应的地图文件）；

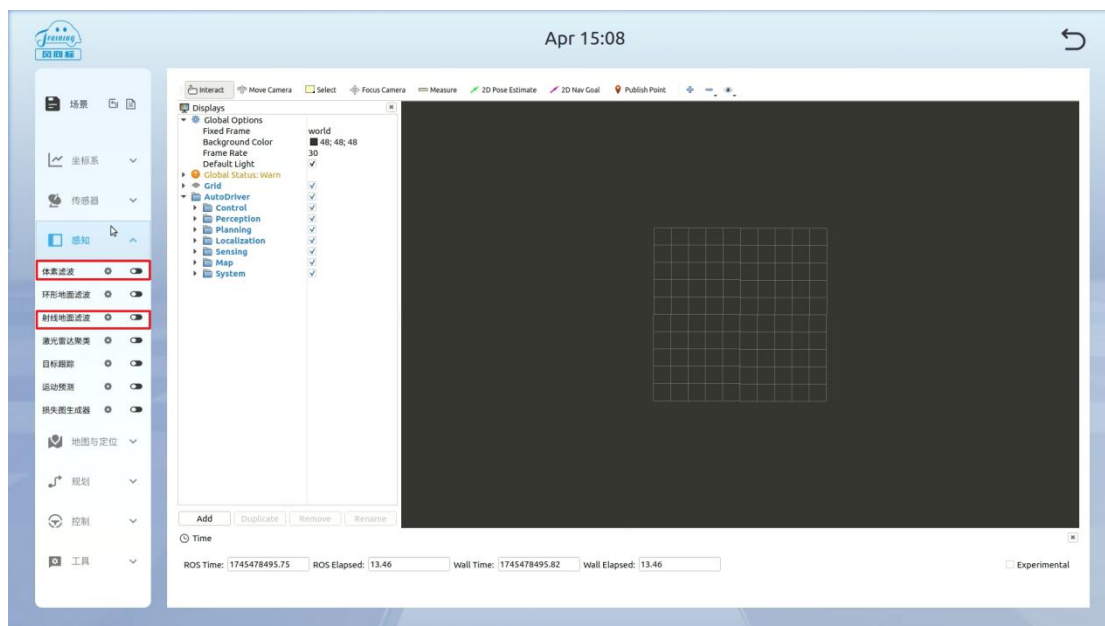




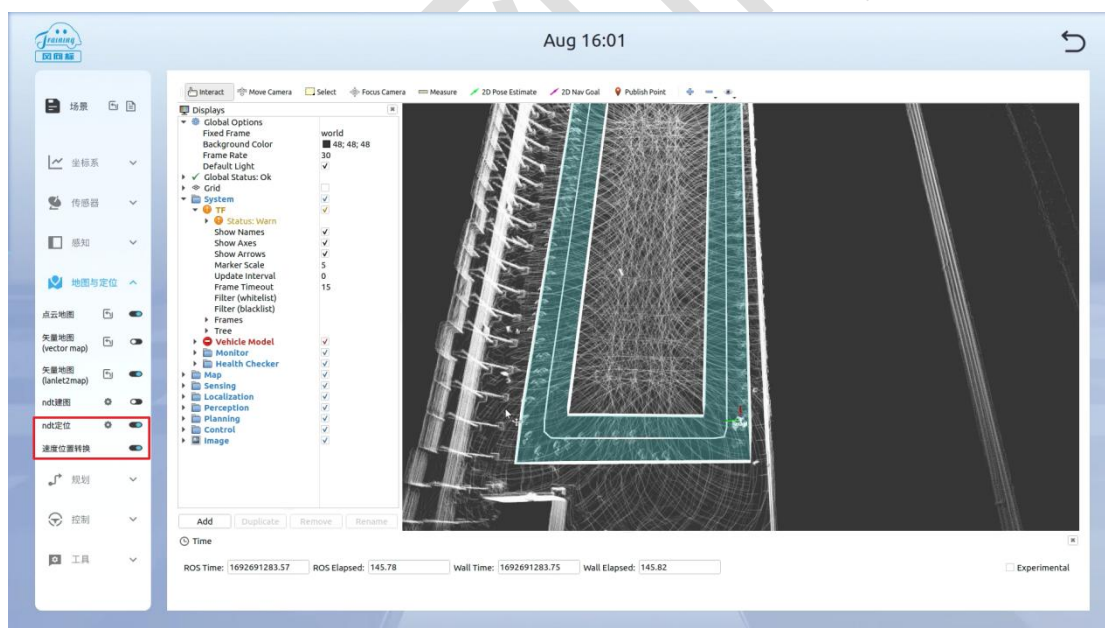
6) 等待几秒后重启地图显示插件，当出现白色点云和青色车道表示地图加载成功；

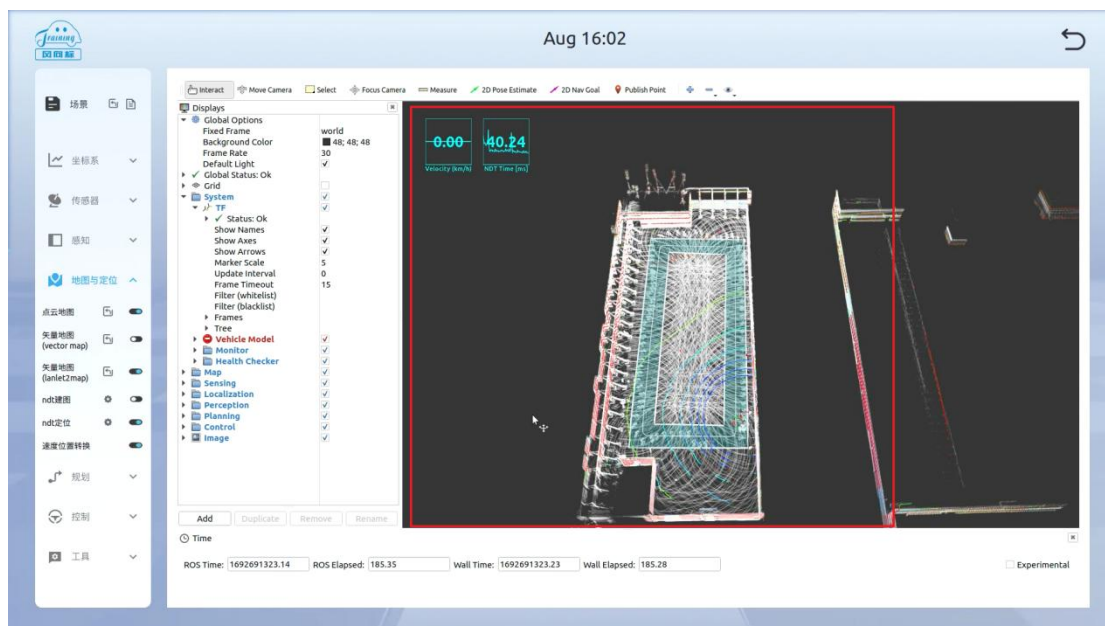


7) 找到“感知”下的“体素滤波”和“射线地面滤波”功能并勾选启动，后台会启动对应的算法程序；

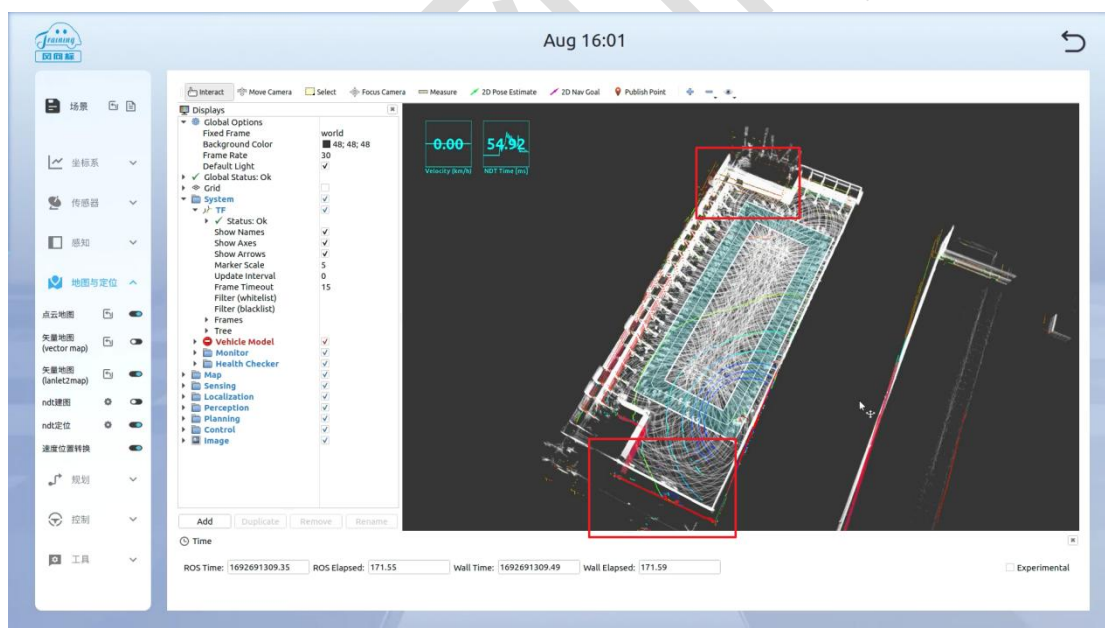


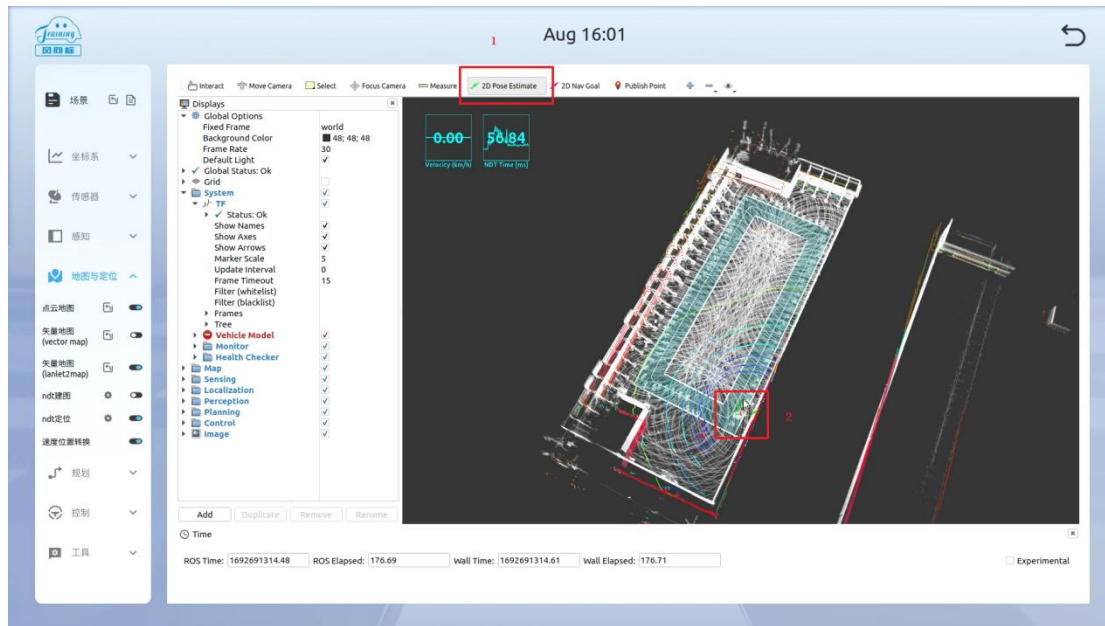
8) 找到“地图与定位”下的“ndt 定位”和“速度位置转换”功能并勾选启动，后台会使用当前的点云数据和 PCD 地图进行匹配定位车辆的位置（默认不使用 GNSS 的情况）；



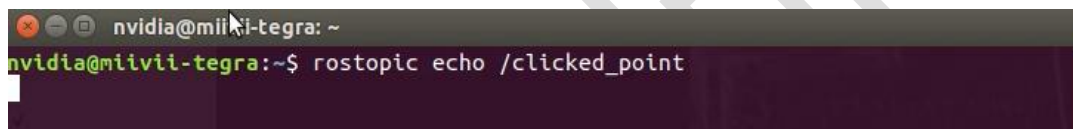


注意！可以通过点云与地图的匹配程度，来判断定位是否准确，如果定位不准，请先取消勾选 **ndt** 定位后，确认车辆移动到起始点位置后，再勾选启动 **ndt** 定位，也可以点击“**2D Pose Estimate**”按钮手动给车辆设置起始点（此方法难度较大）；

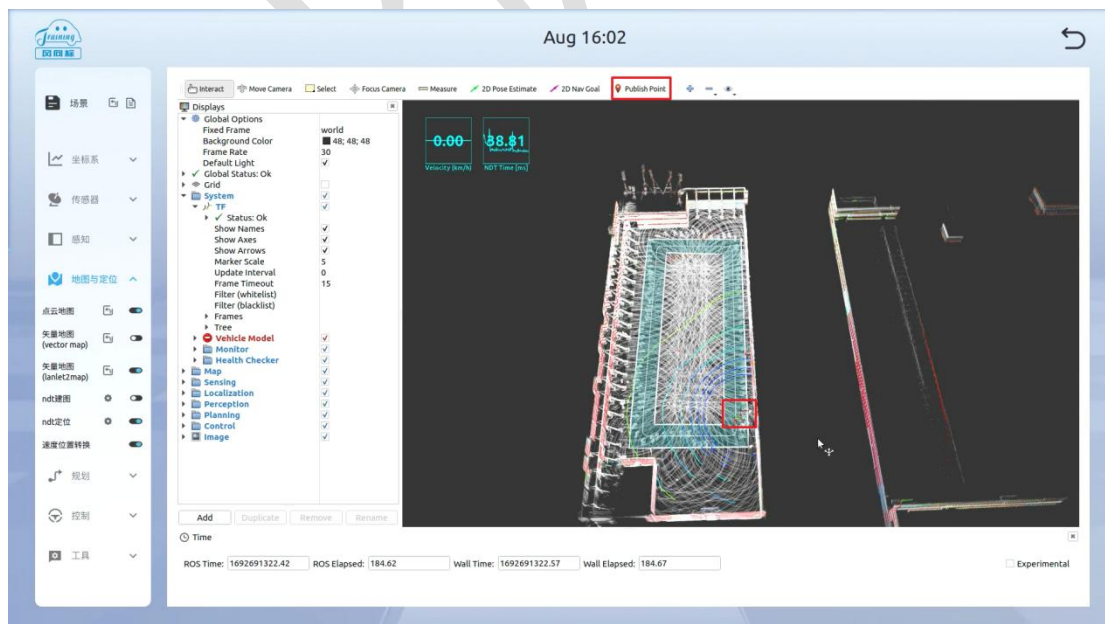




9) 使用快捷键 **ctrl+alt+t** 打开终端，输入命令 **rostopic echo /clicked_point** 后按下回车；

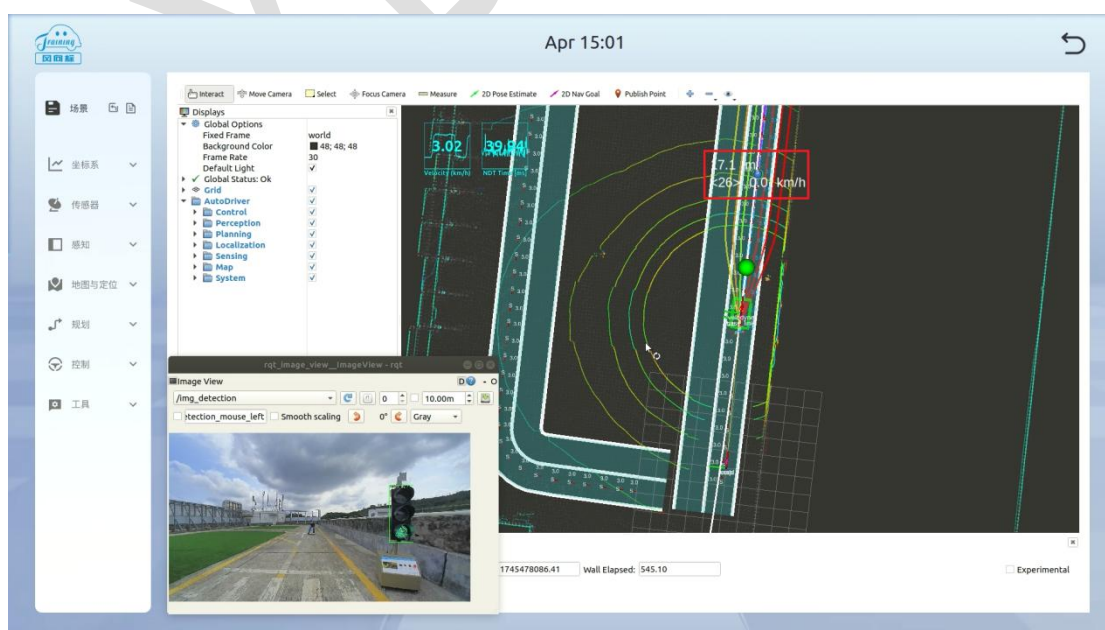
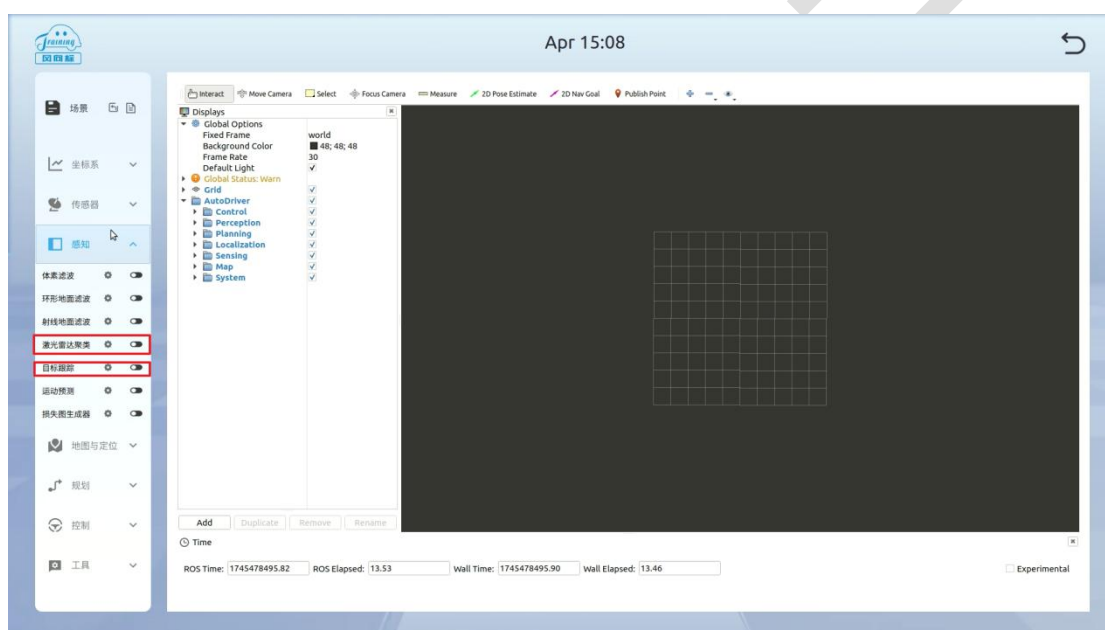


10) 点击“Publish Point”按钮后点击三色坐标，使用快捷键 **alt+tab** 切换到步骤 9 打开的终端，记录 **xy** 起始点坐标到报告单上；



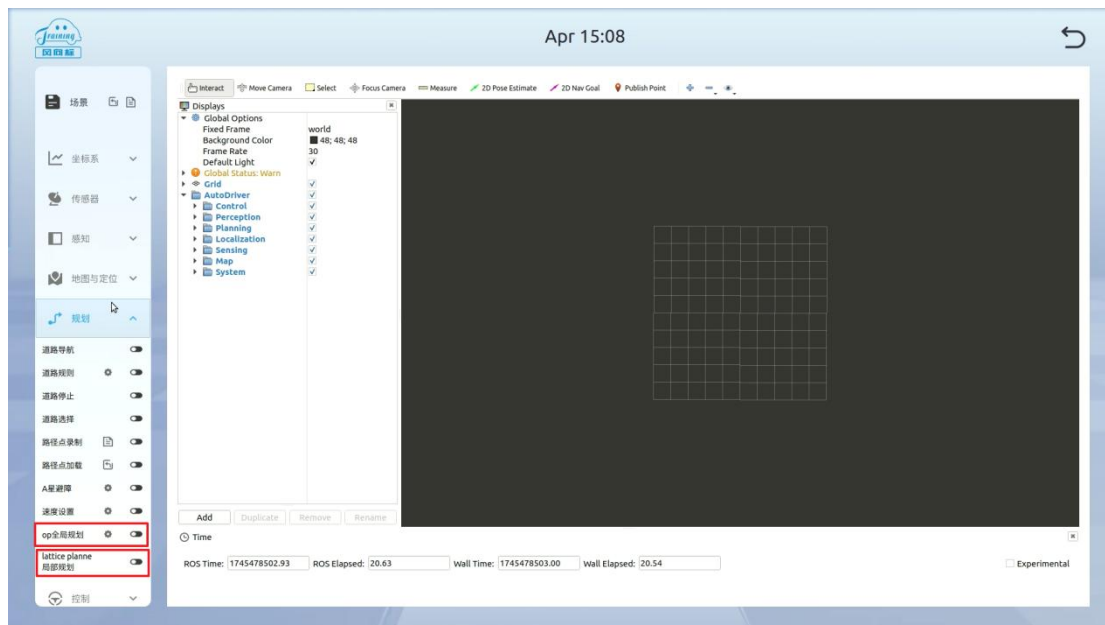

```
nvidia@mlivii-tegra: ~
nvidia@mlivii-tegra:~$ rostopic echo /clicked_point
header:
  seq: 0
  stamp:
    secs: 1692691249
    nsecs: 494395837
  frame_id: "world"
data:
  x: -0.311116218567
  y: 0.0503212809563
  z: 1.5555118103
```

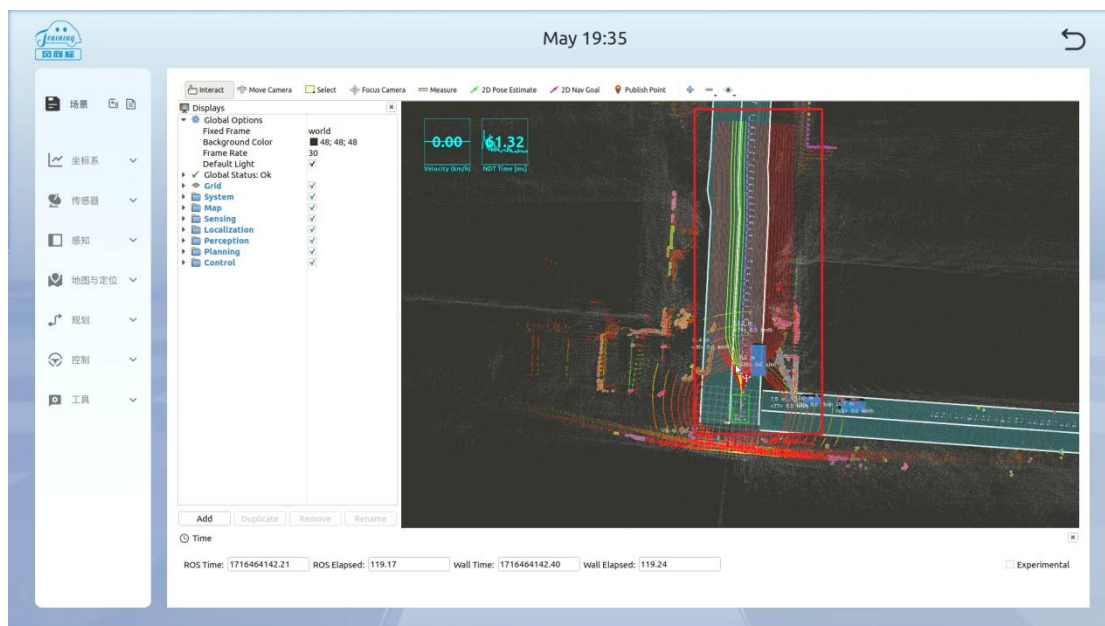
11) 找到“感知”下的“激光雷达聚类”和“目标跟踪”功能并勾选启动，等待几秒钟后，在高精地图上会看到物体检测跟踪的效果，说明后台算法启动成功；



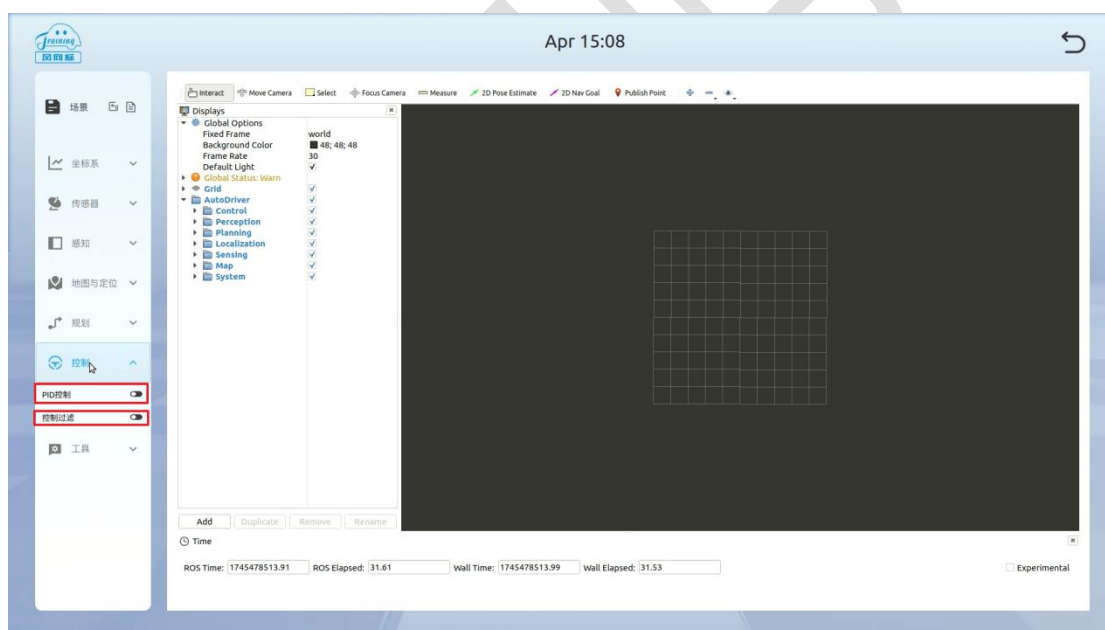


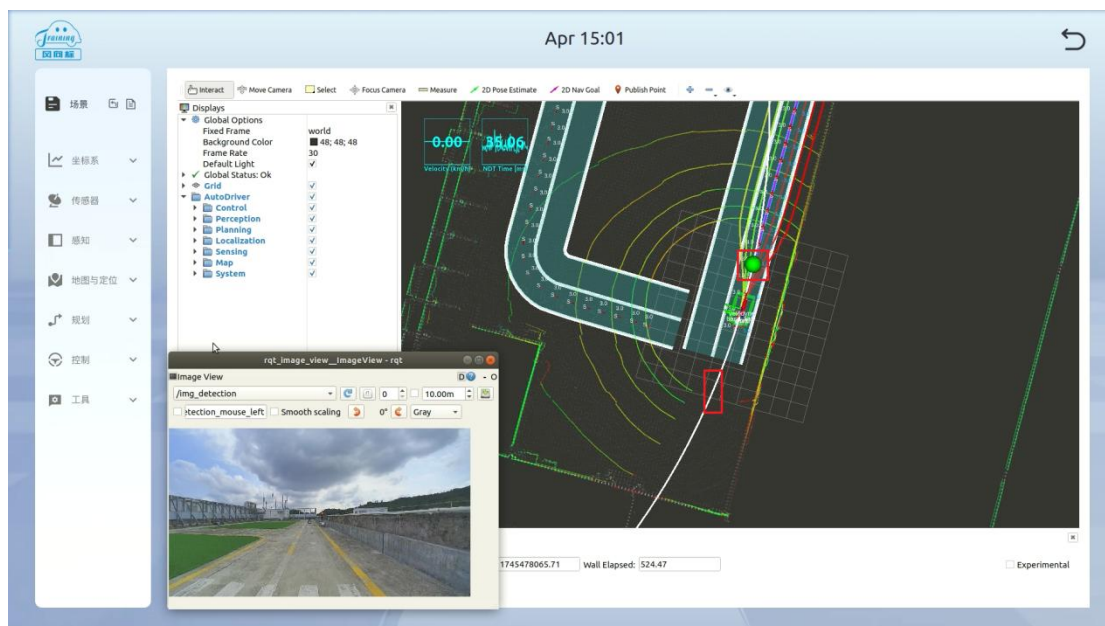
12) 找到“规划”下的“op 全局规划”和“lattice planne 局部规划”功能并勾选启动，等待几秒后，点击“2D Nav Goal”按钮，在高精地图同一个车道上，选择终点位置点击并拖动（左键点击后不松手，使用触摸板拖向车道方向后再松手）；



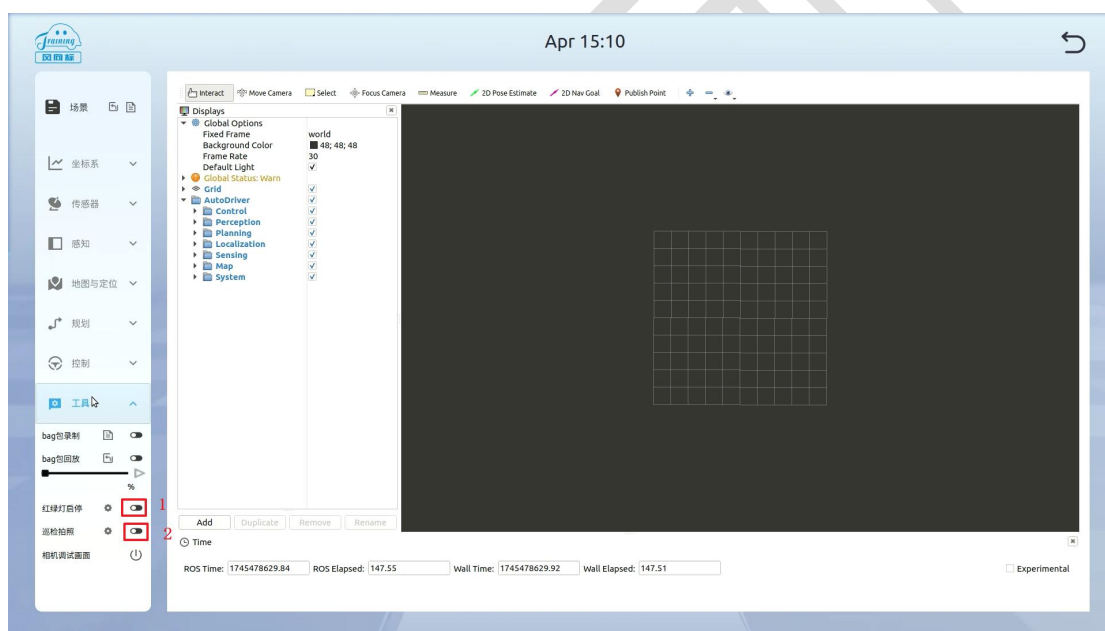


13) 找到“控制”下的“PID 控制”和“控制过滤”功能并勾选启动，当出现转向弧线和跟踪大绿点说明后台算法启动成功；





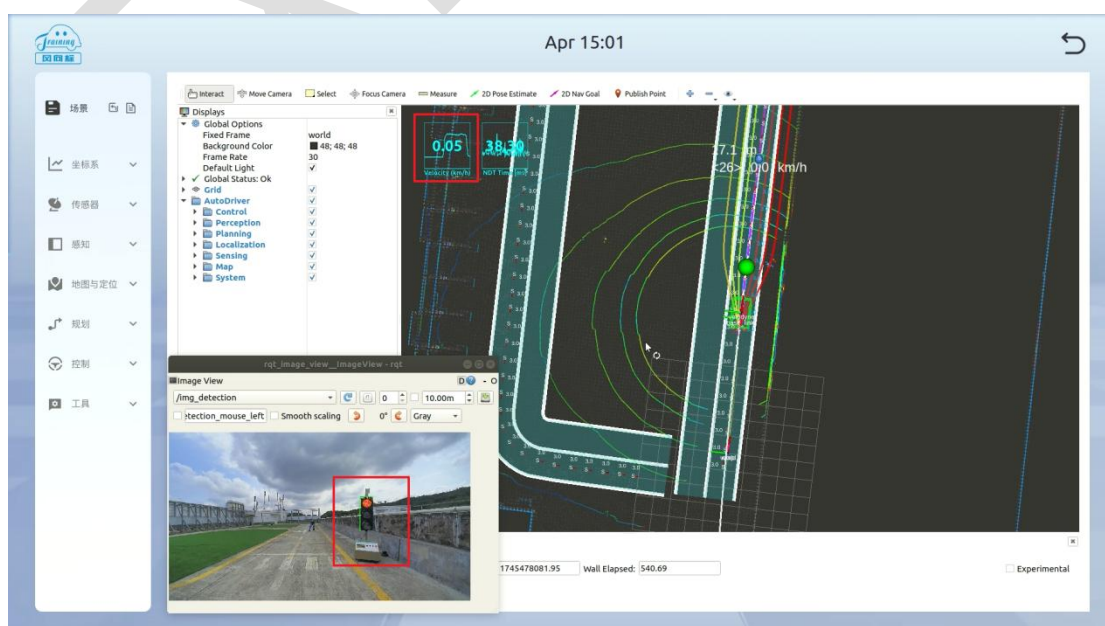
14) 找到“工具下”的“红绿灯启停”和“巡检拍照”功能并勾选启动；

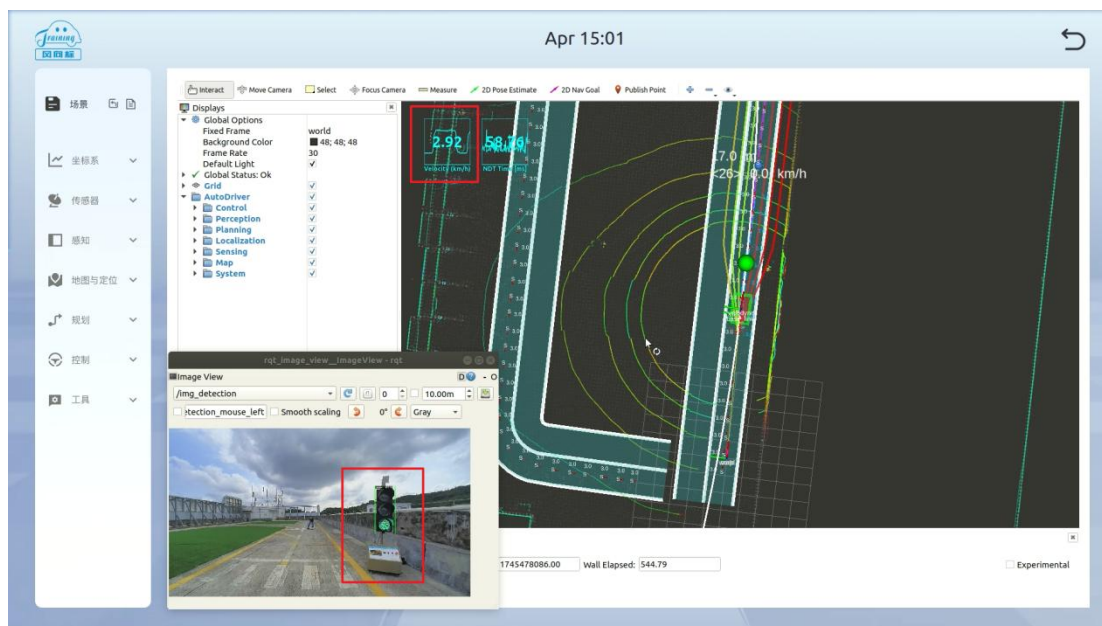


15) 将遥控器控制模式设置为自动驾驶模式，车辆将会按照指定路径行驶；

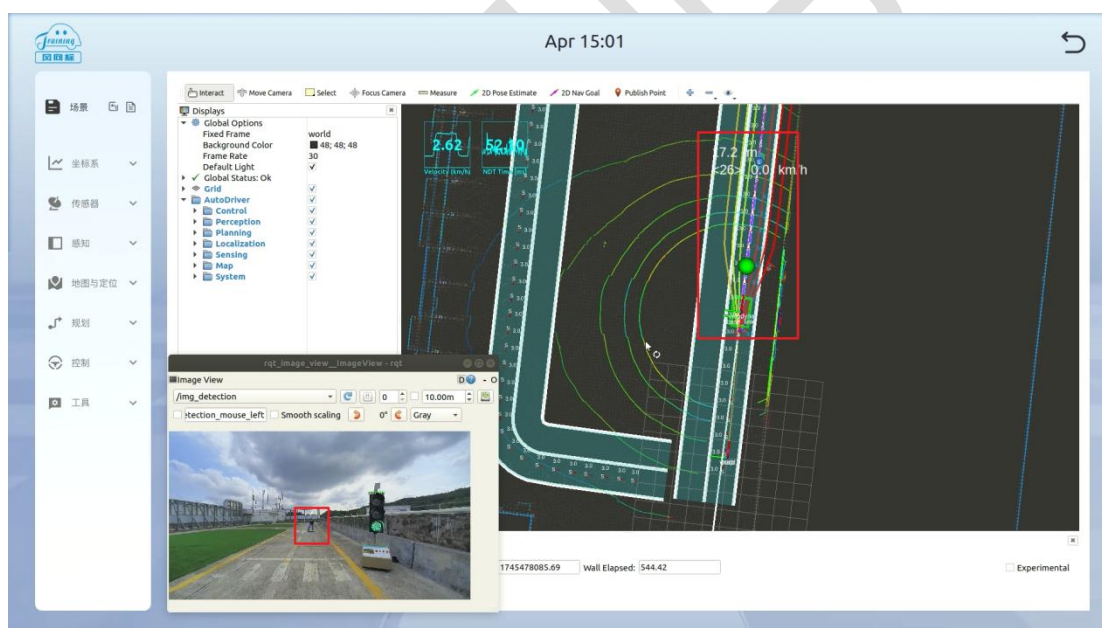


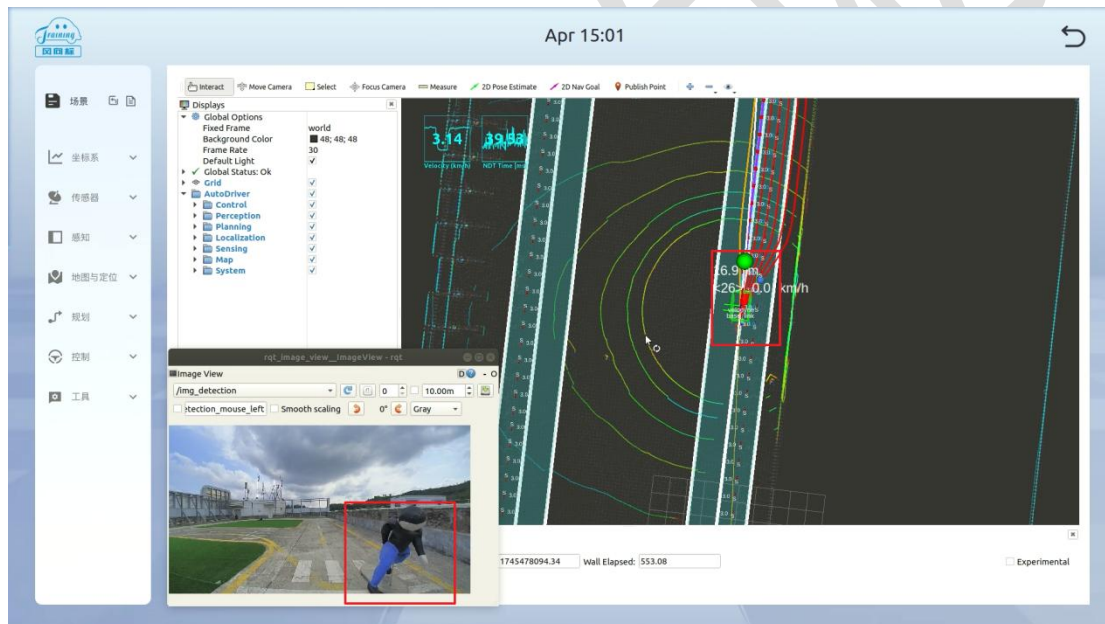
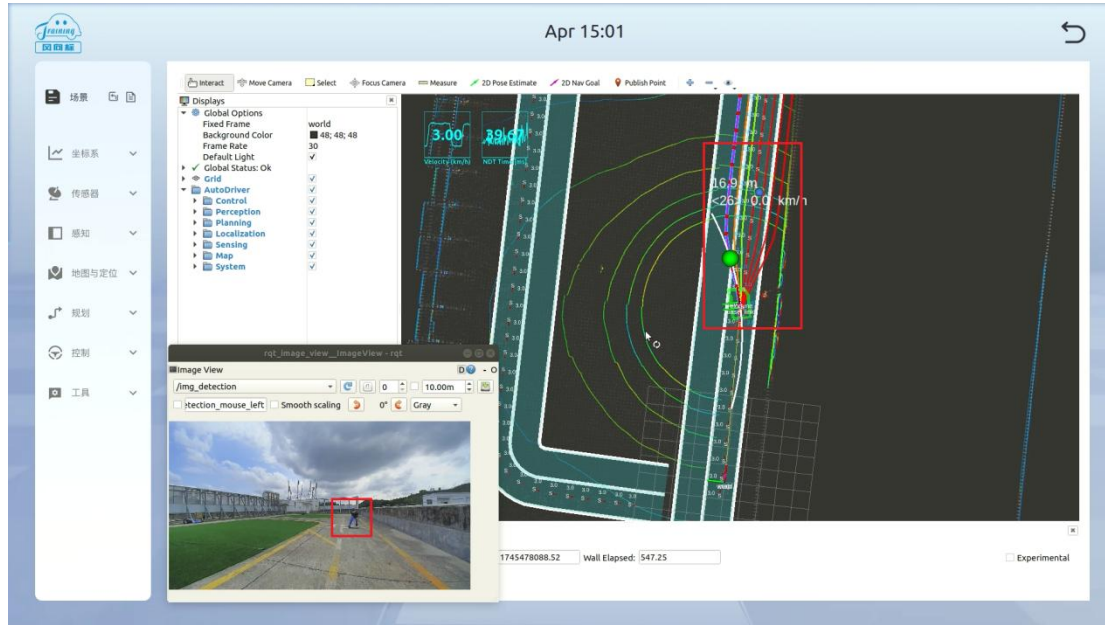
到达每个巡检点时会停留 3 秒并采集此刻单目相机的视觉画面
在红绿灯检测区域内，检测到红灯时车辆停车，绿灯时车辆行驶



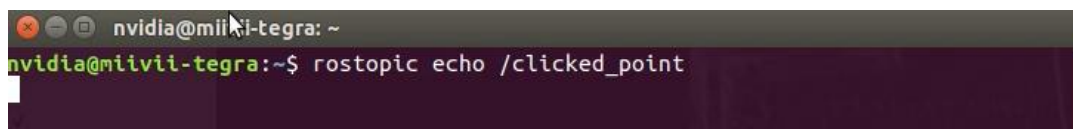


当车辆前方遇到障碍物时，路径规划算法开始计算避障路径，控制算法会根据避障路径行驶，最终避开障碍物

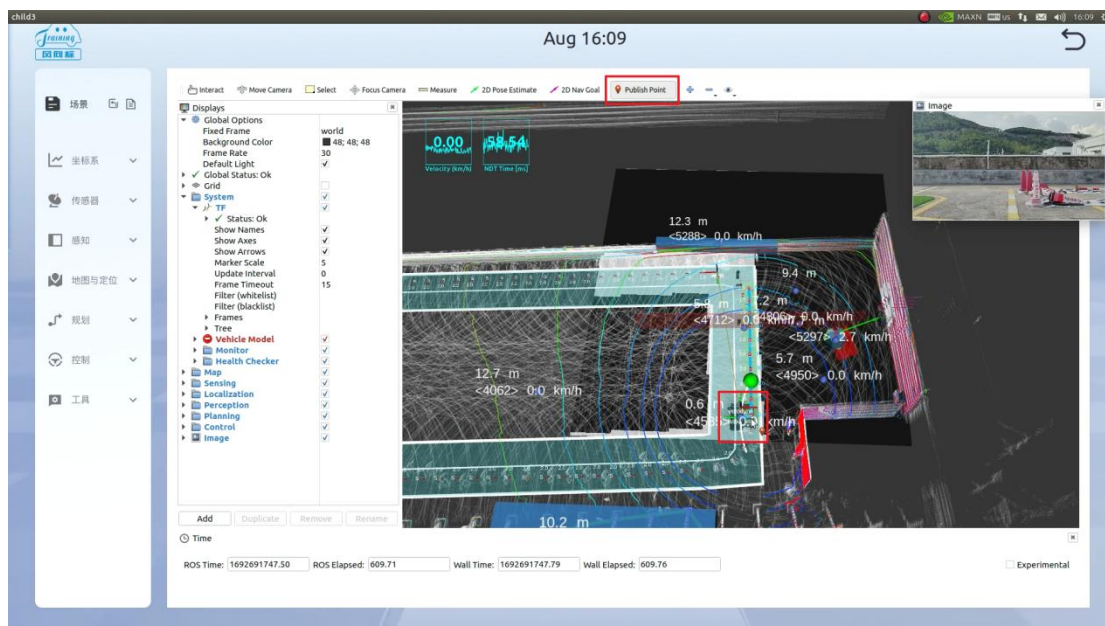




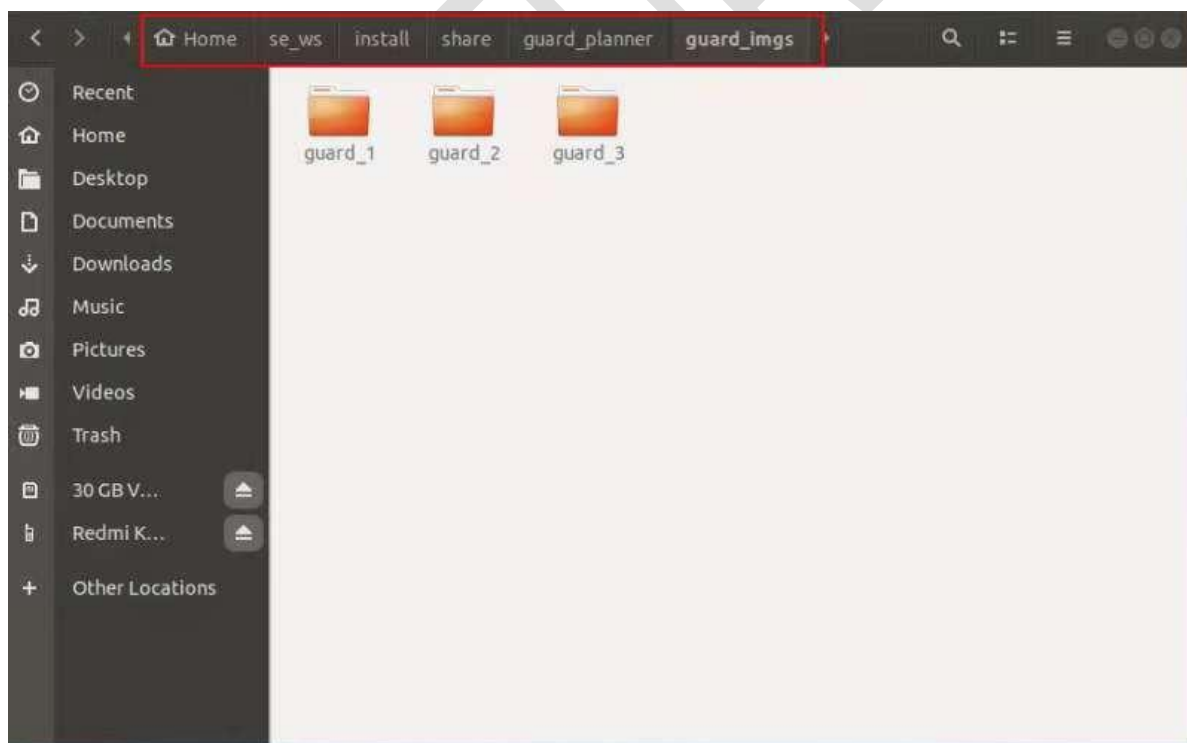
16) 当车辆行驶到终点时, 使用快捷键 `ctrl+alt+t` 打开终端, 输入命令 `rostopic echo /clicked_point` 后按下回车;

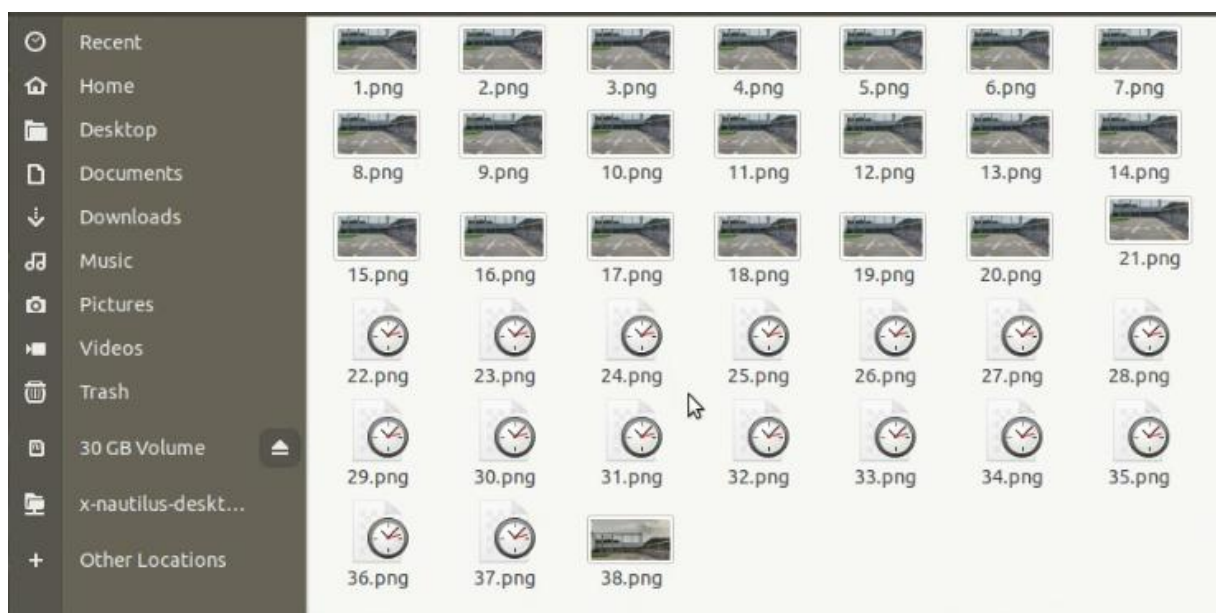


17) 点击“Publish Point”按钮后点击三色坐标, 使用快捷键 `alt+tab` 切换到步骤 16 打开的终端, 记录 xy 终点坐标到报告单上;



18) 找到/home/nvidia/driver_ws/install/share/guard_planner/guard_imgs 文件夹，里面有到达三个巡检点时采集单目视觉画面的图像，记录每个巡检点采集图像的数量到报告单上；





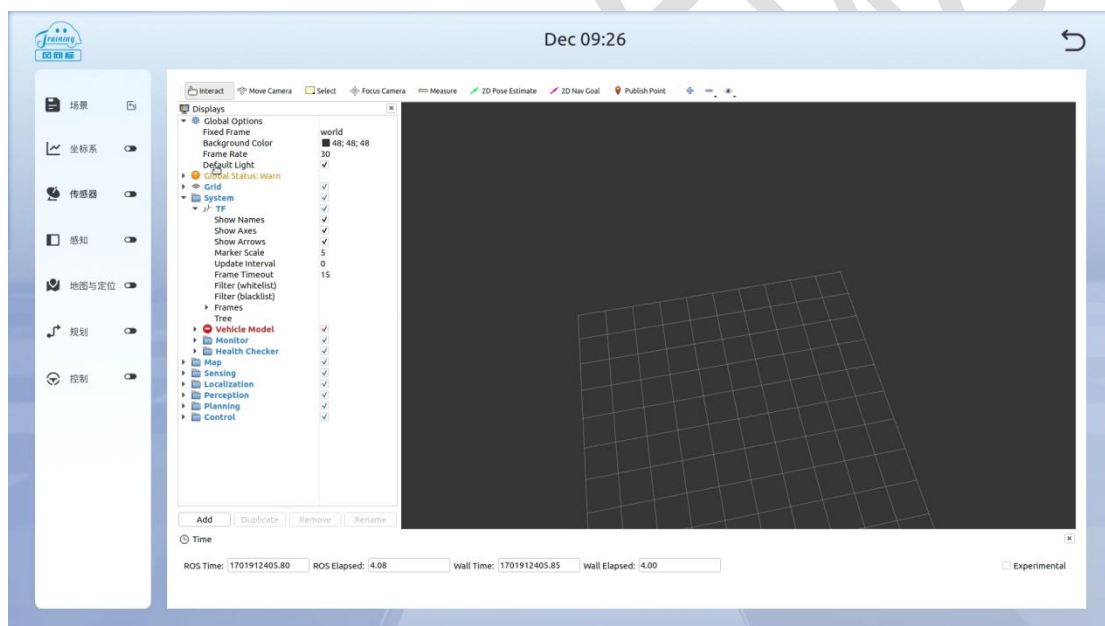
3.11 道路测试-初级教学-方案二

0) 在关闭所有软件和终端后再操作本小节；

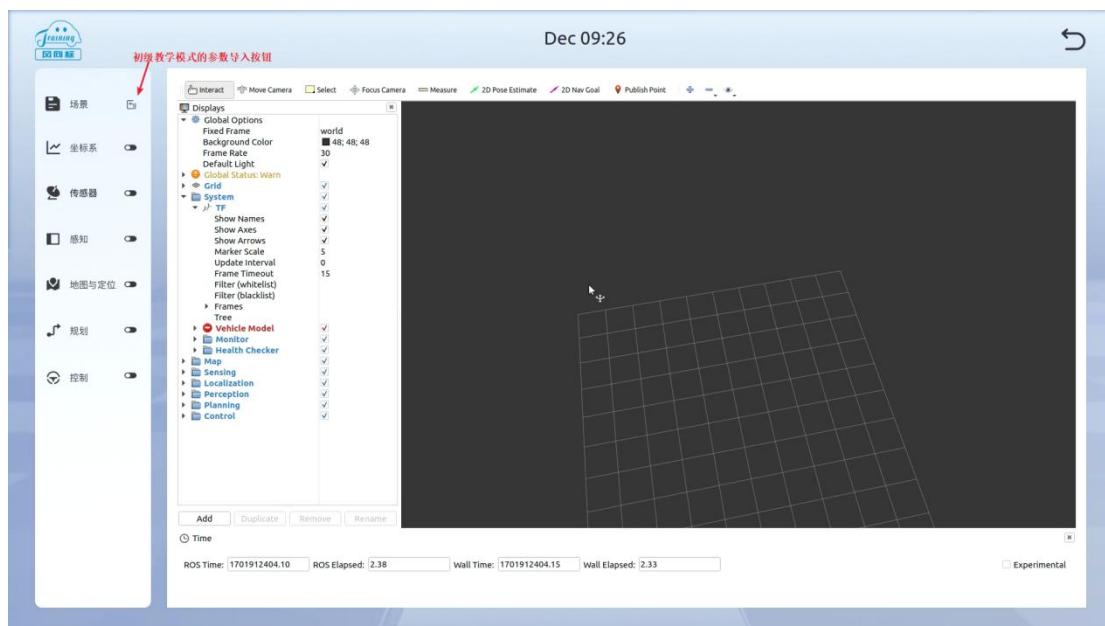
1) 使用遥控器将车辆行驶到起始区域；



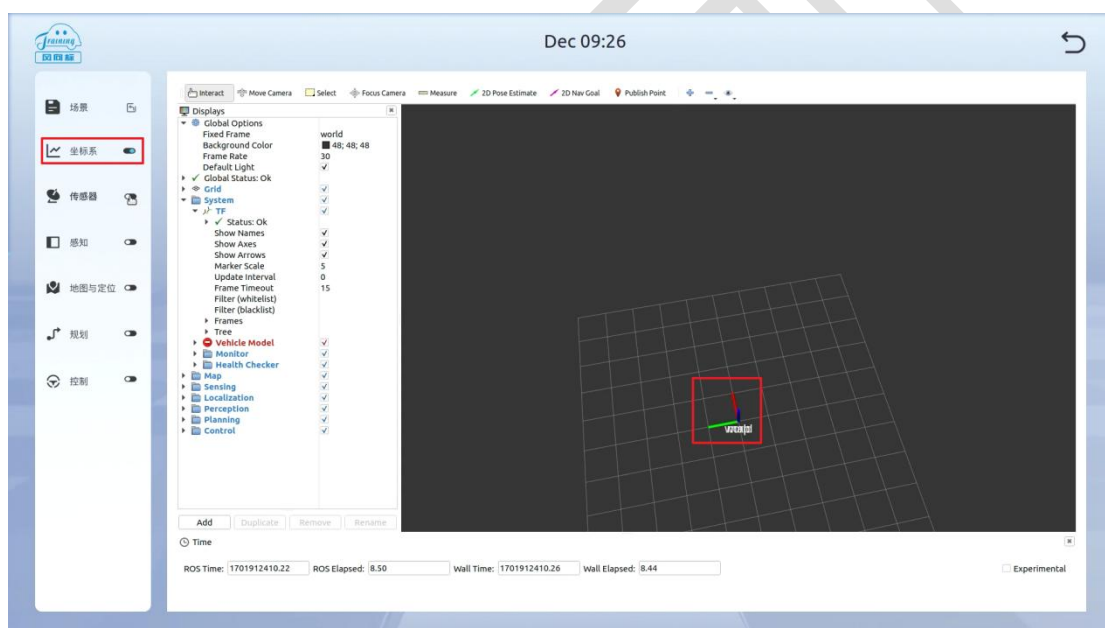
2) 打开自动驾驶教学软件，选择“初级教学”模式；



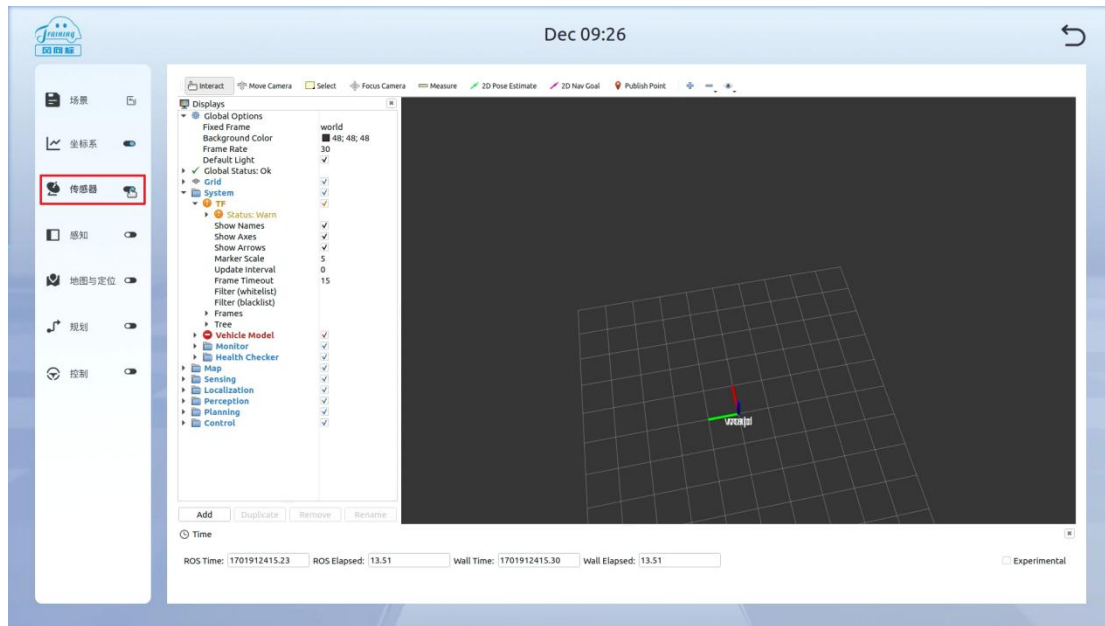
3) 导入从 3.6 小节导出的参数，选择方案二；



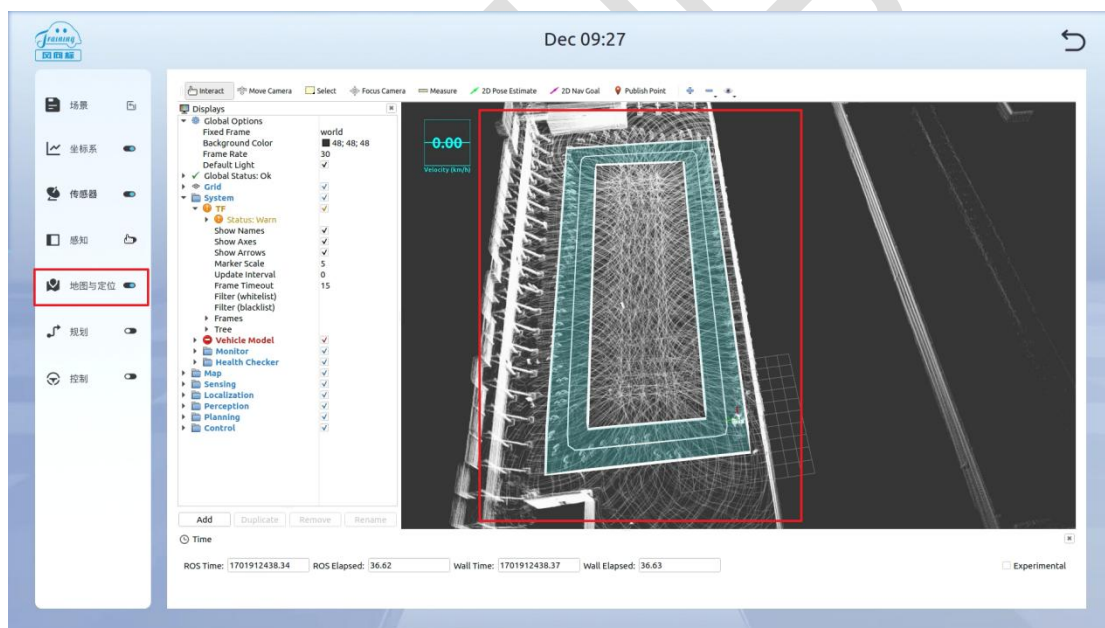
4) 勾选启动“坐标系”，启动成功后会出现“三色坐标系”；



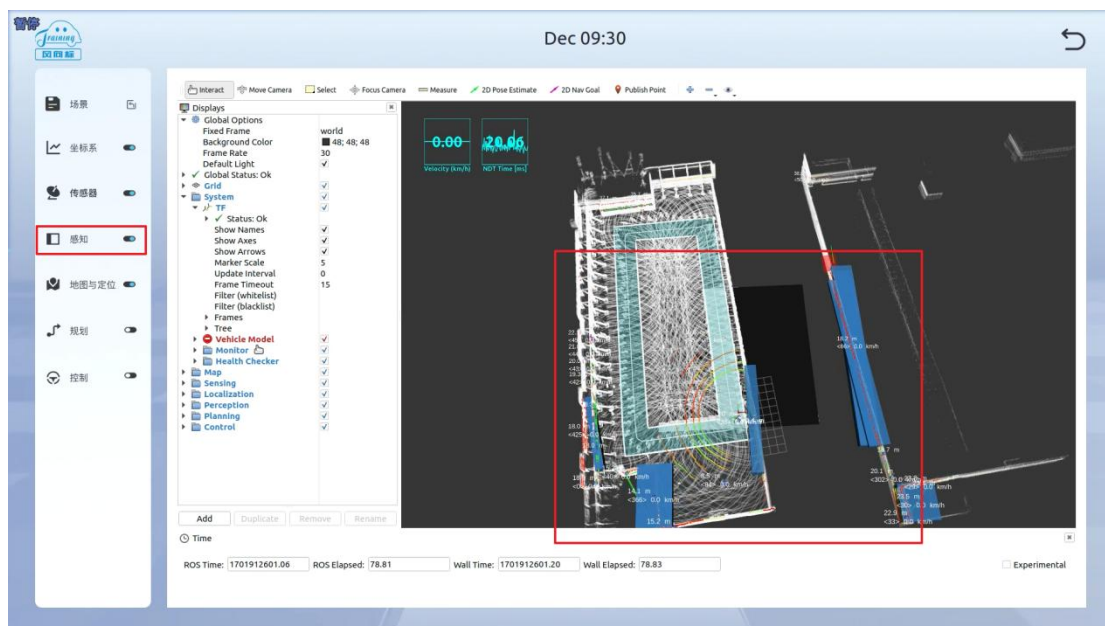
5) 勾选启动“传感器”；



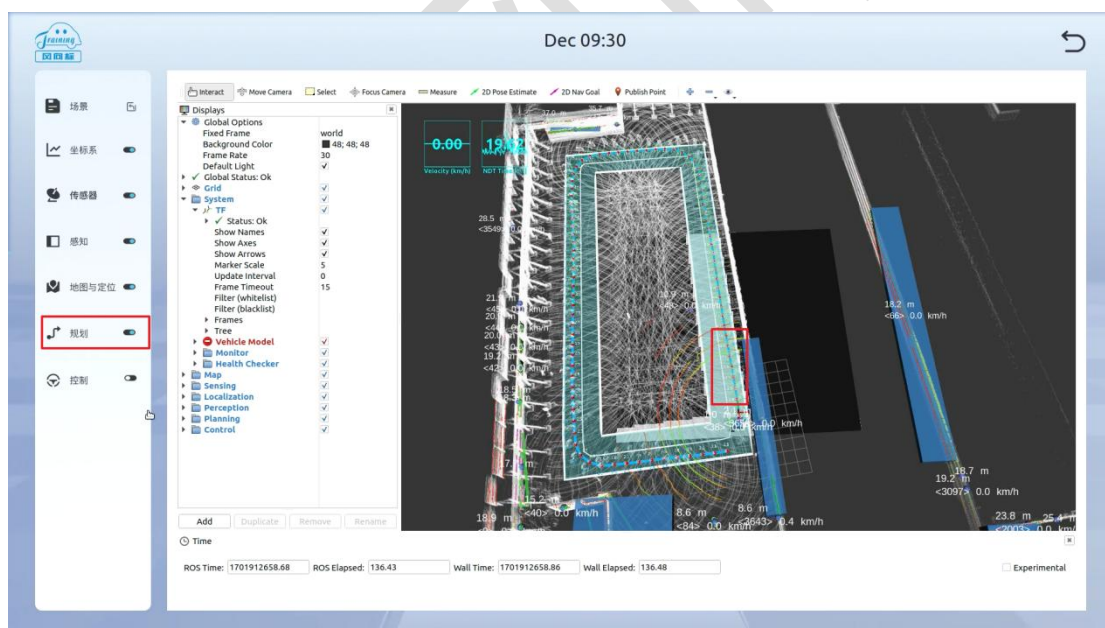
6) 勾选启动“地图与定位”，启动成功后会出现白色的 PCD 地图和青绿色的 HD 高精地图；



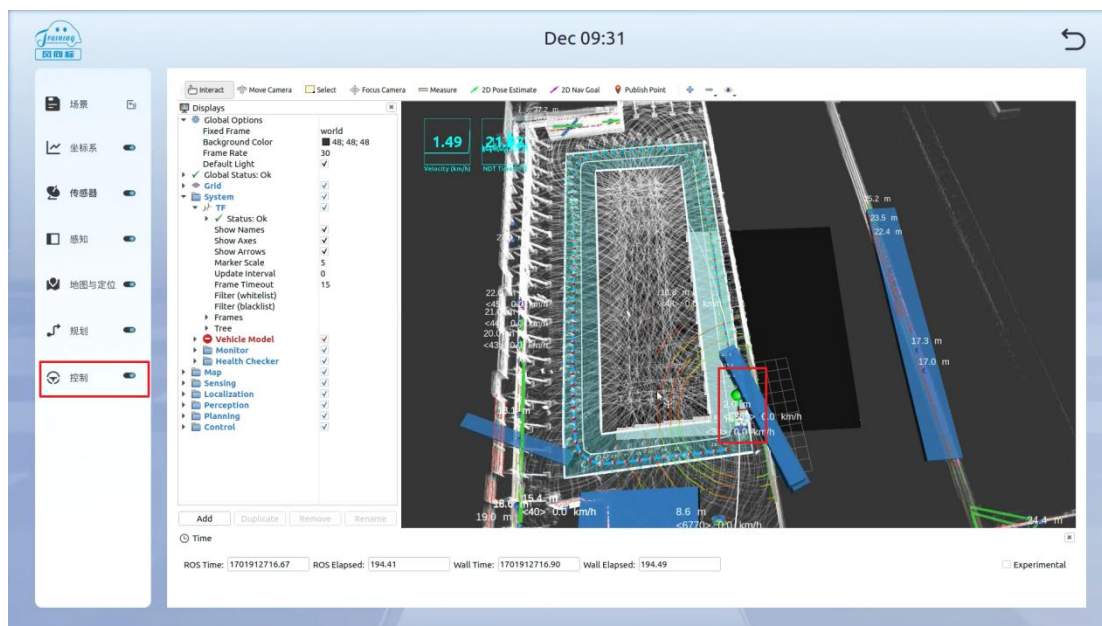
7) 勾选启动“感知”，启动成功后会出现蓝色矩形；



8) 勾选启动“规划”，点击“2D Nav Goal”按钮，在高精地图同一个车道上，选择终点位置点击并拖动（左键点击后不松手，使用触摸板拖向车道方向后再松手），启动成功后会出现多条不同颜色的候选路线；



9) 勾选启动“控制”，启动成功后会出现大绿点和转向弧线；



10) 将遥控器控制模式设置为自动驾驶模式，车辆将会按照指定路径行驶；



3.12 道路测试-演示模式-方案二

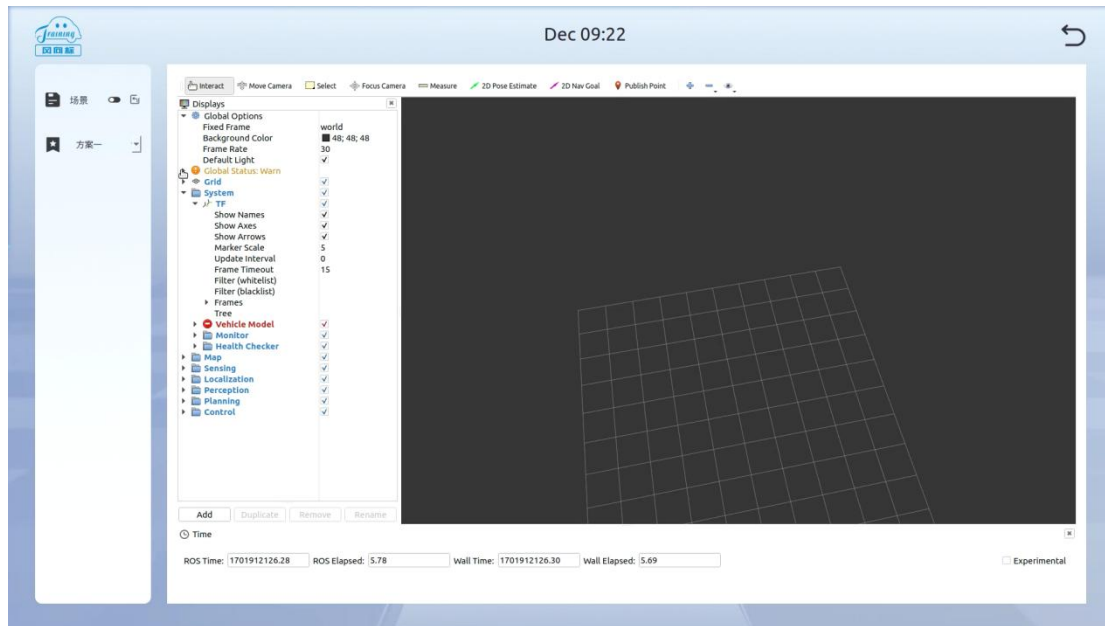
0) 在关闭所有软件和终端后再操作本小节；

1) 使用遥控器将车辆行驶到起始区域；

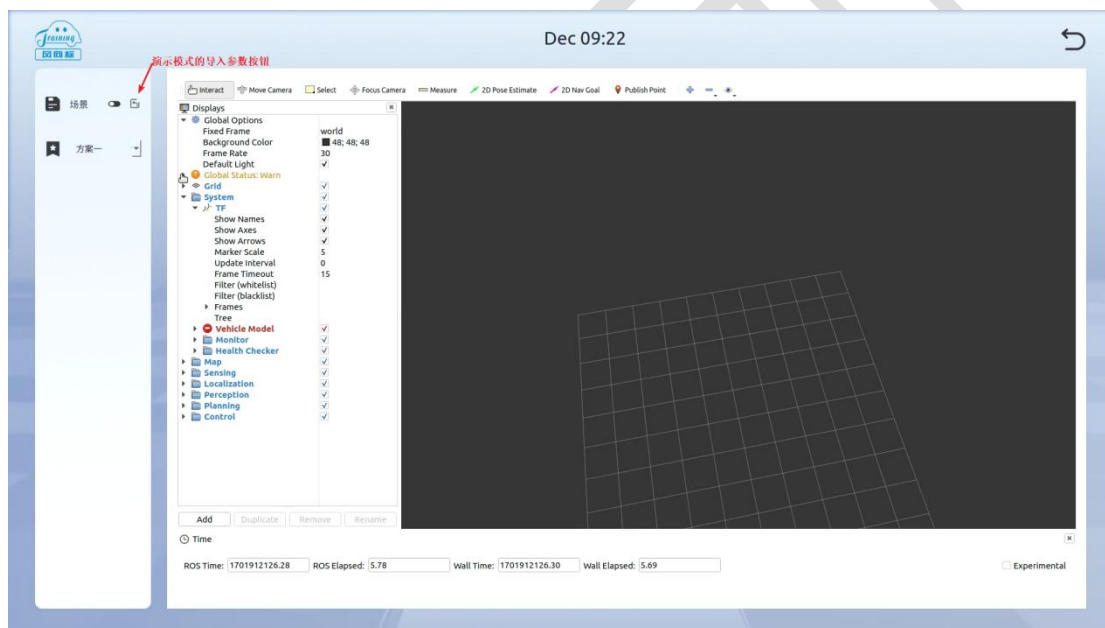


2) 打开自动驾驶教学软件，选择“演示”模式；





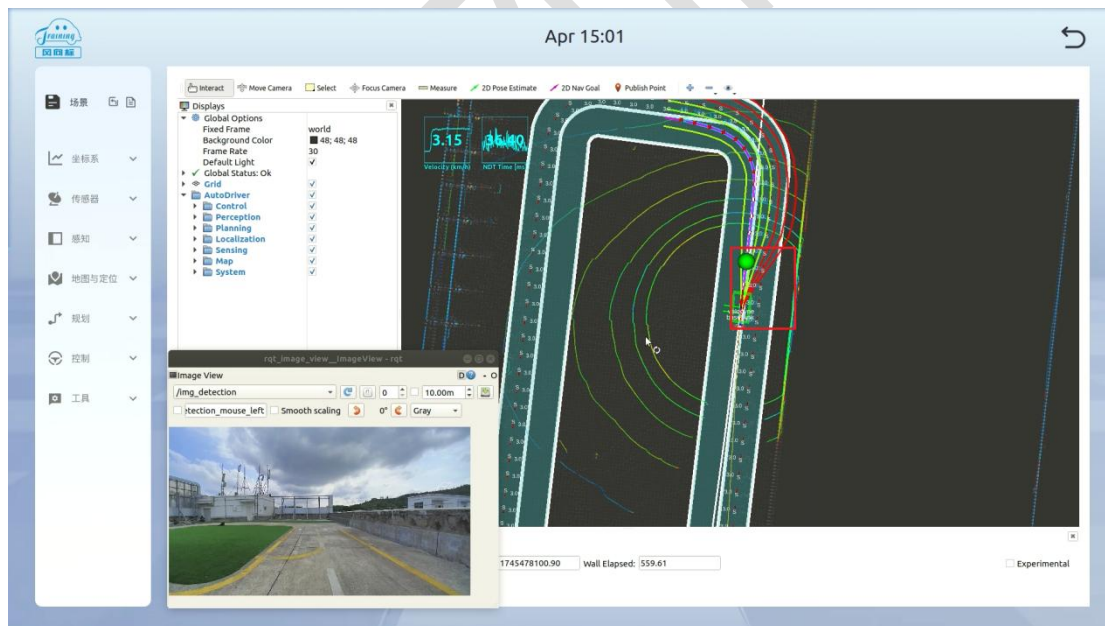
3) 导入从 3.6 小节导出的参数，选择方案二；



4) 点击一键启动按钮，启动成功后会出现白色 PCD 地图、青绿色高清 HD 地图，说明所有算法启动成功；



5) 点击“2D Nav Goal”按钮，在高精地图同一个车道上，选择终点位置点击并拖动（左键点击后不松手，使用触摸板拖向车道方向后再松手），启动成功后会出现多条不同颜色的候选路线和大绿点和转向弧线；



6) 将遥控器控制模式设置为自动驾驶模式，车辆将会按照指定路径行驶；

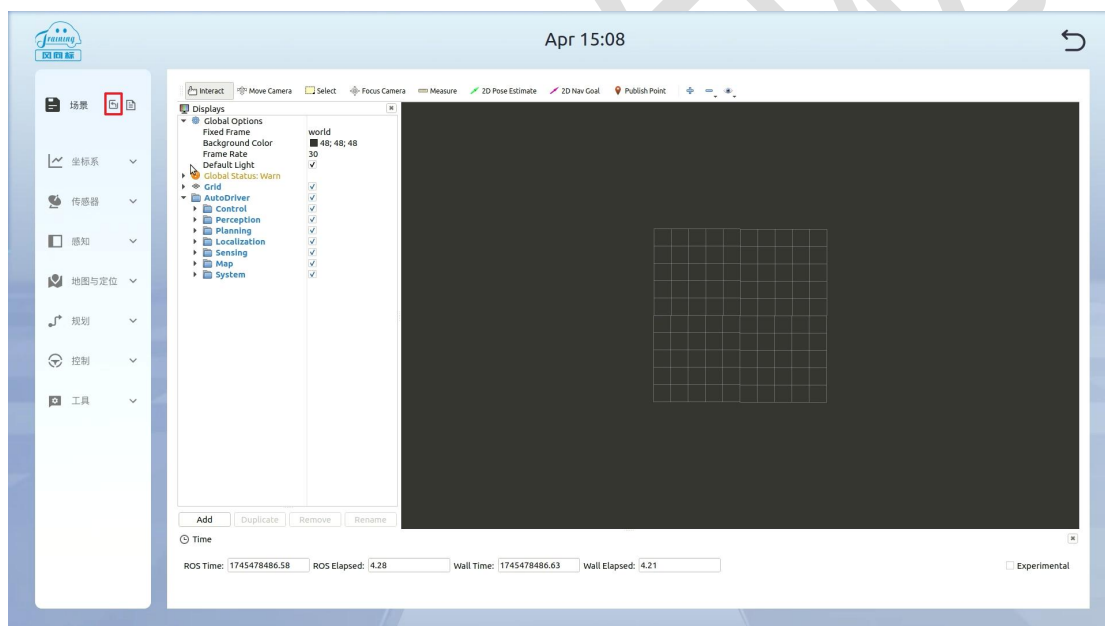


3.13 道路测试-高级教学-OBV

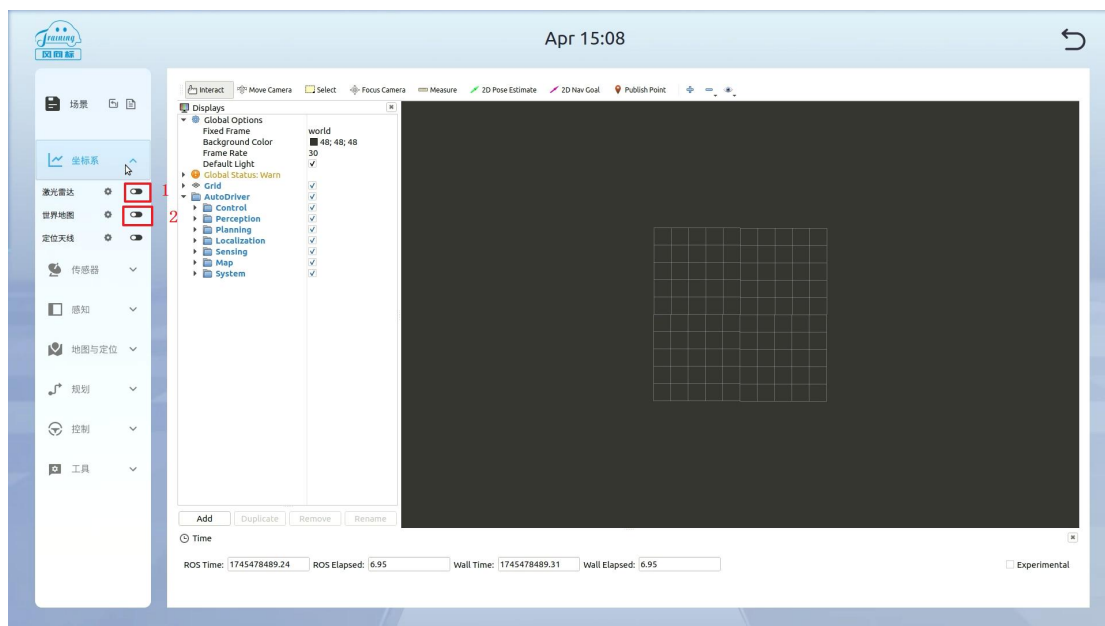
- 0) 在关闭所有软件和终端后再操作本小节;
- 1) 使用遥控器将车辆行驶到起始区域;



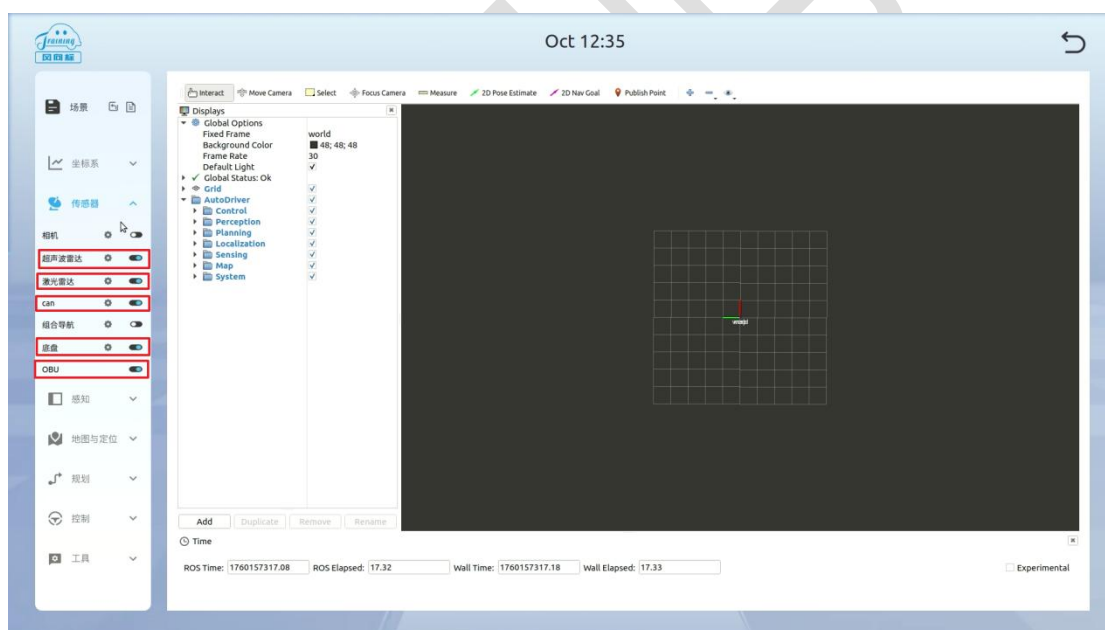
- 2) 打开自动驾驶教学软件, 选择“高级教学”模式, 导入从 3.6 小节导出的参数;



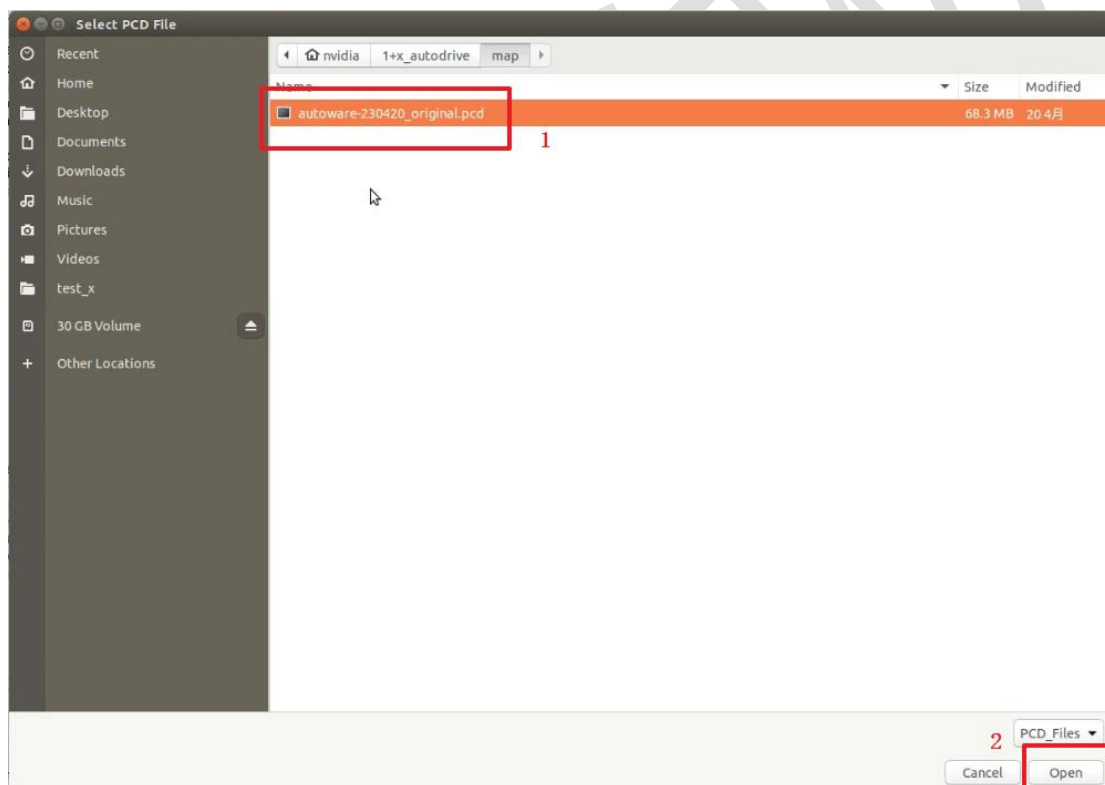
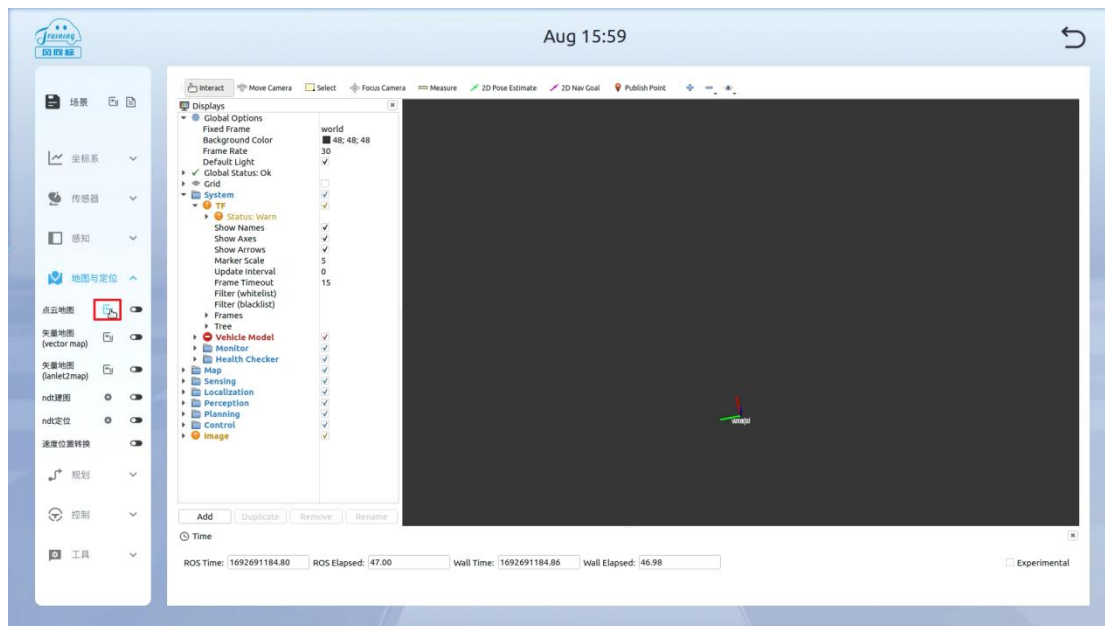
3) 找到“坐标系”下的“激光雷达”和“世界地图”功能并勾选启动，运行成功后会现三色坐标系；

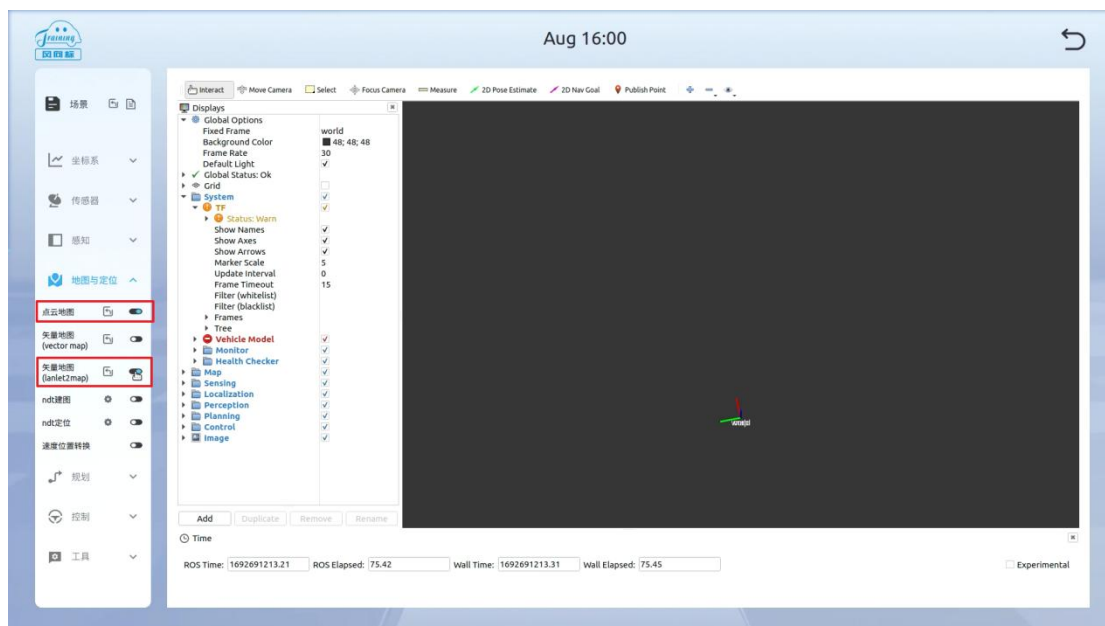


4) 找到传感器下的“超声波雷达”、“激光雷达”、“CAN”、“底盘”和“OBU”并勾选启动它们，后台会启动对应的驱动程序；

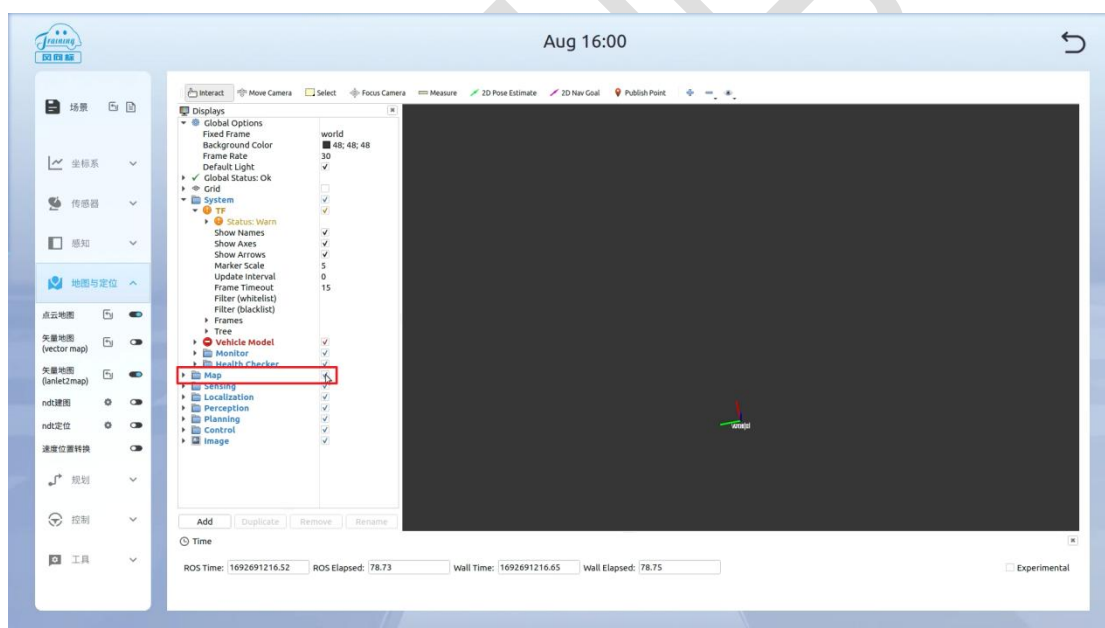


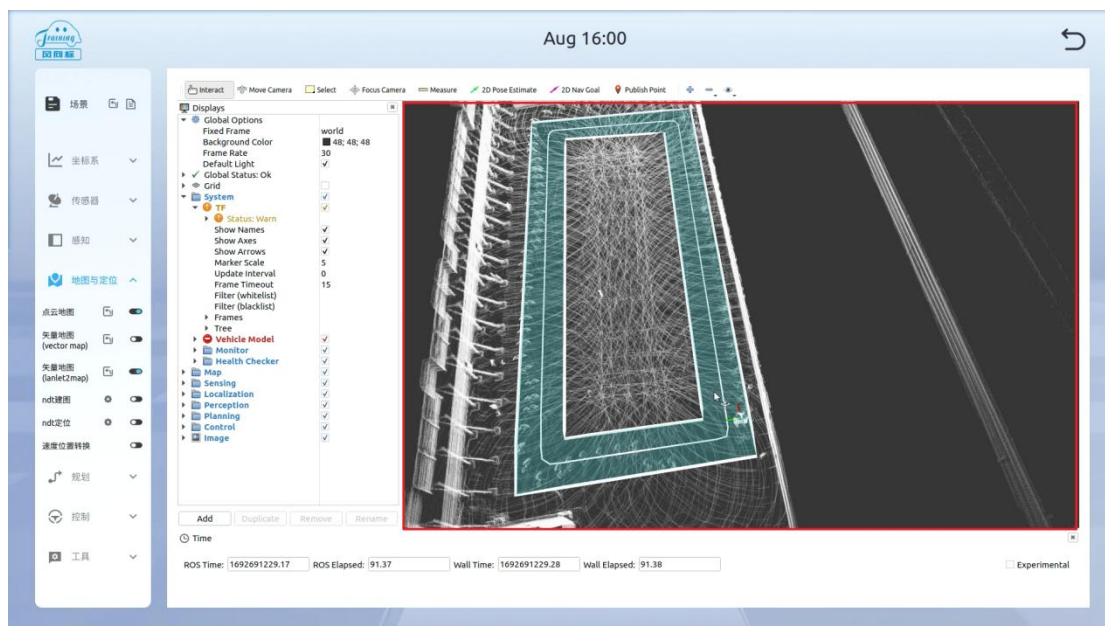
5) 找到地图与定位下的“点云地图”和“矢量地图 (lanelet2map)”功能，勾选启动点云地图和矢量地图 (lanelet2map)，后台会启动对应的地图加载程序 (如果没有导入 3.6 小节导出的参数，那这里的地图需要点击旁边的配置按钮，手动导入对应的地图文件)；



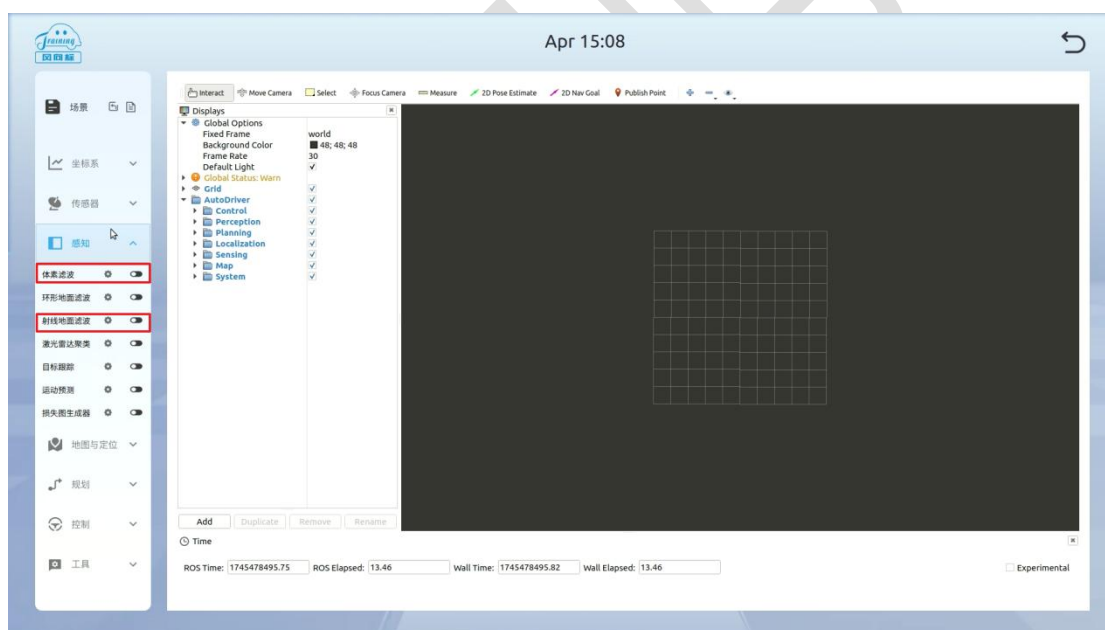


6) 等待几秒后重启地图显示插件，当出现白色点云和青色车道表示地图加载成功；

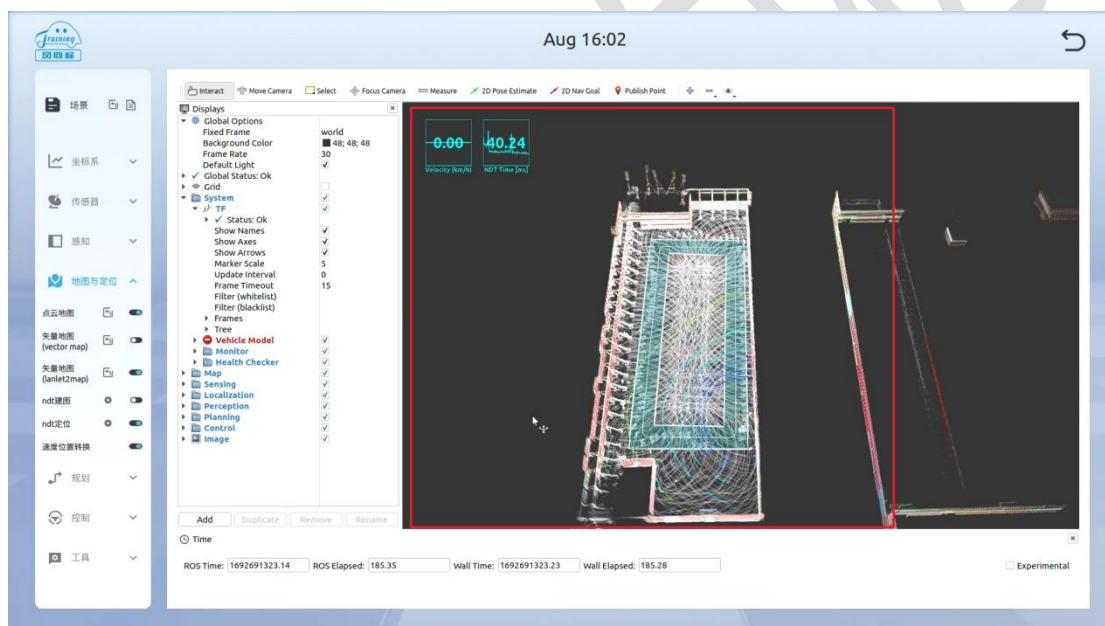
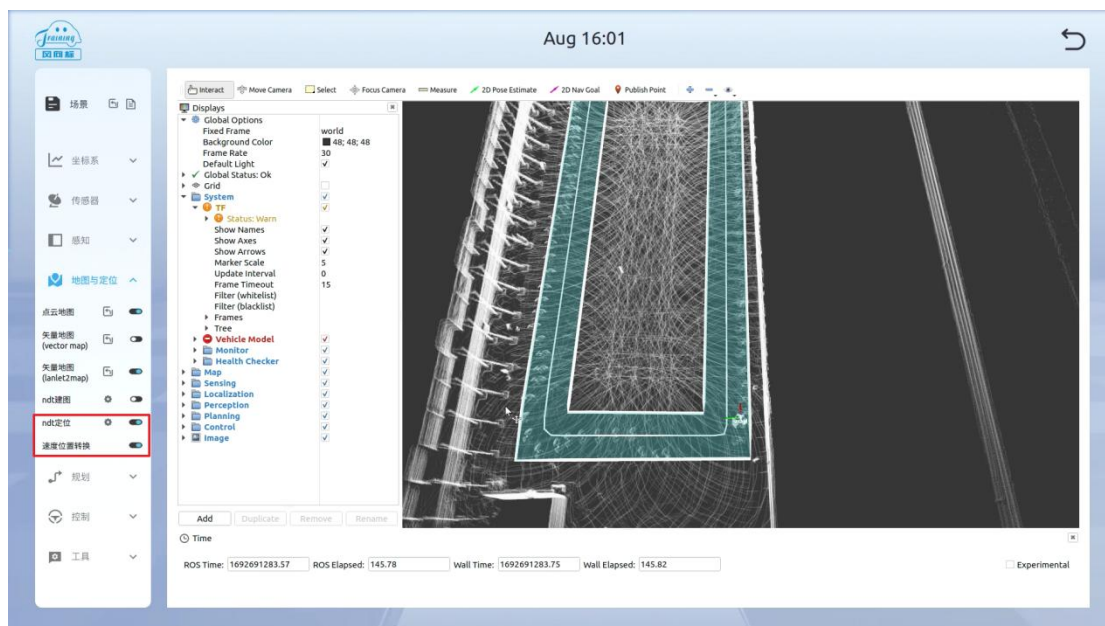




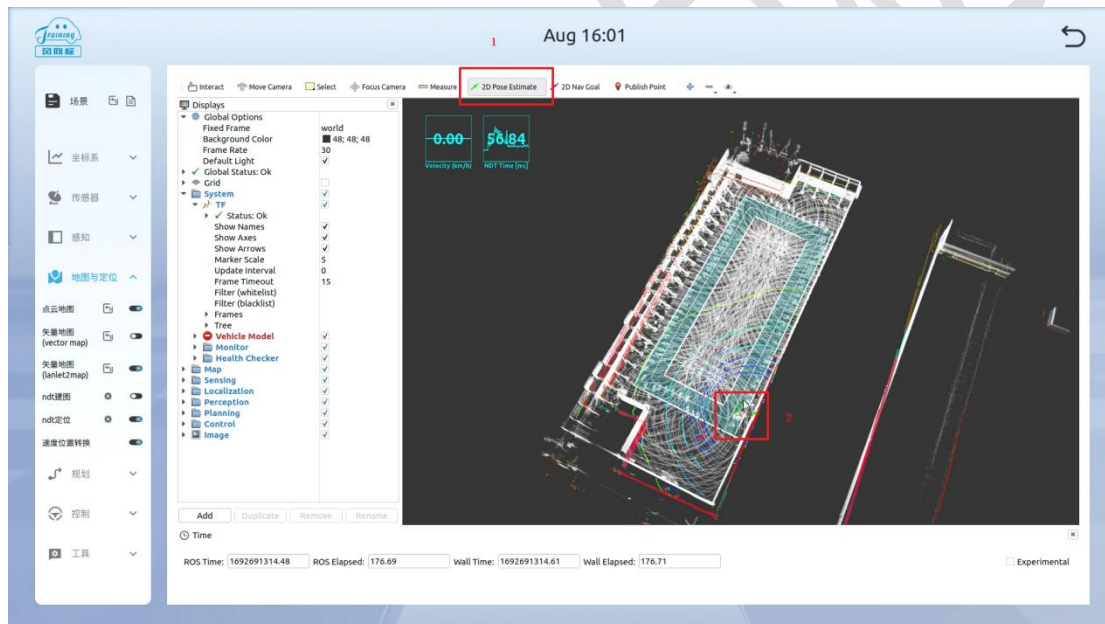
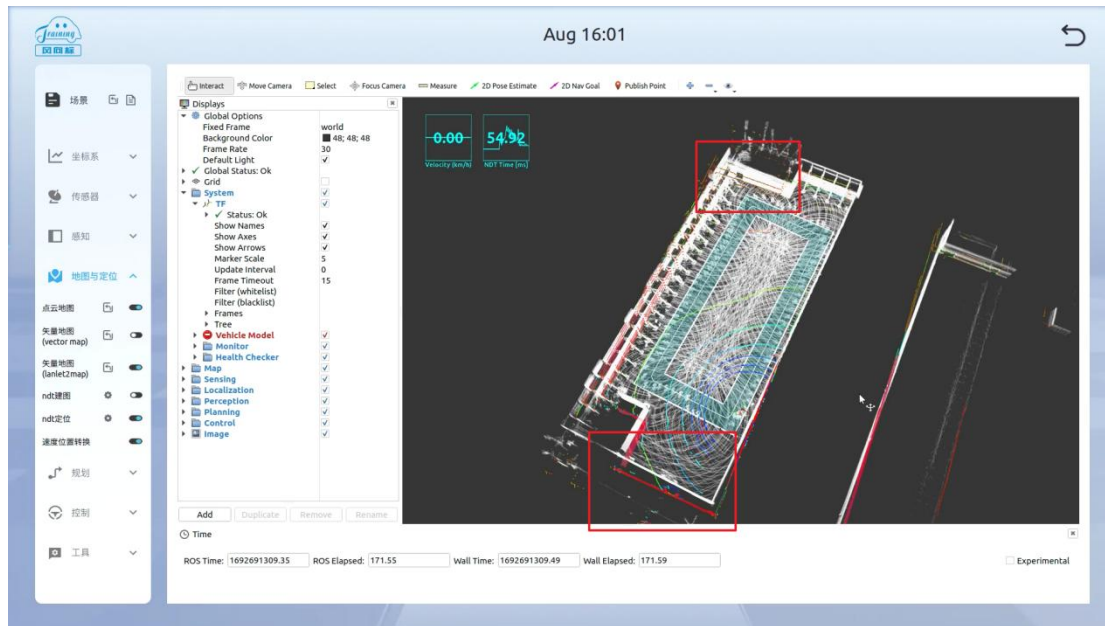
7) 找到“感知”下的“体素滤波”和“射线地面滤波”功能并勾选启动，后台会启动对应的算法程序；



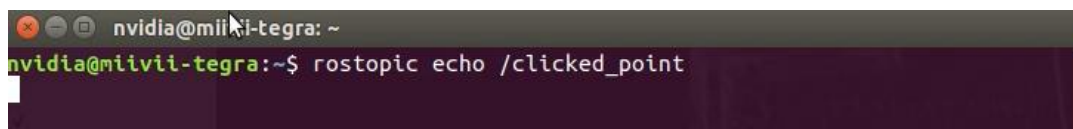
8) 找到“地图与定位”下的“ndt 定位”和“速度位置转换”功能并勾选启动，后台会使用当前的点云数据和 PCD 地图进行匹配定位车辆的位置（默认不使用 GNSS 的情况）；



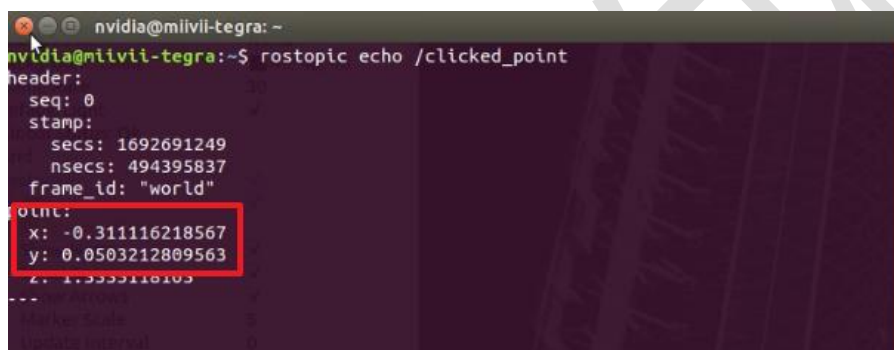
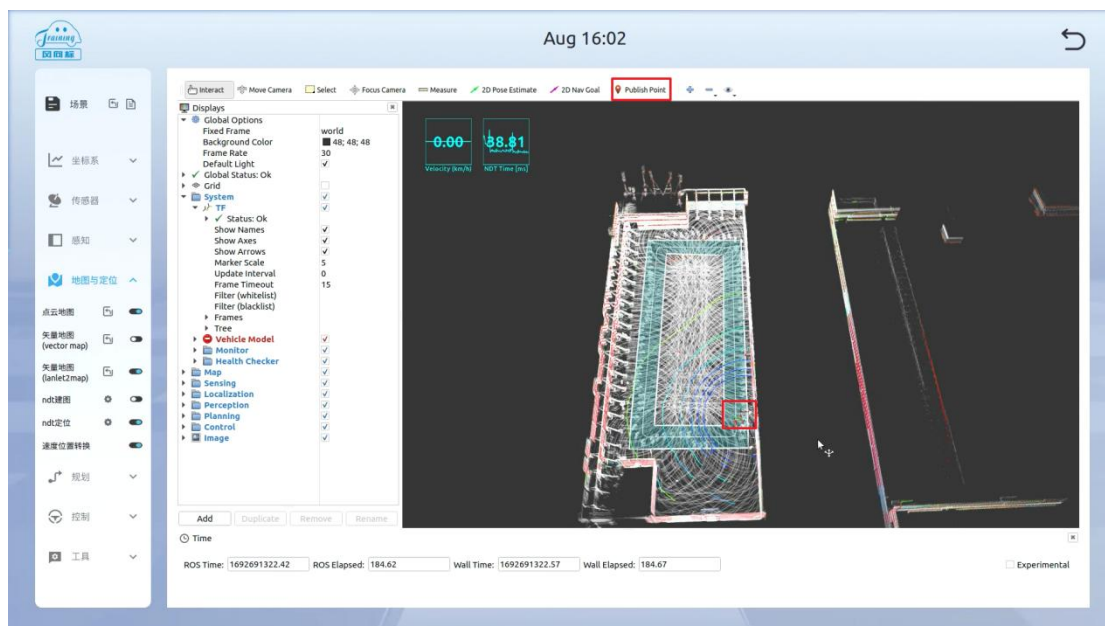
注意！可以通过点云与地图的匹配程度，来判断定位是否准确，如果定位不准，请先取消勾选 **ndt** 定位后，确认车辆移动到起始点位置后，再勾选启动 **ndt** 定位，也可以点击“**2D Pose Estimate**”按钮手动给车辆设置起始点（此方法难度较大）；



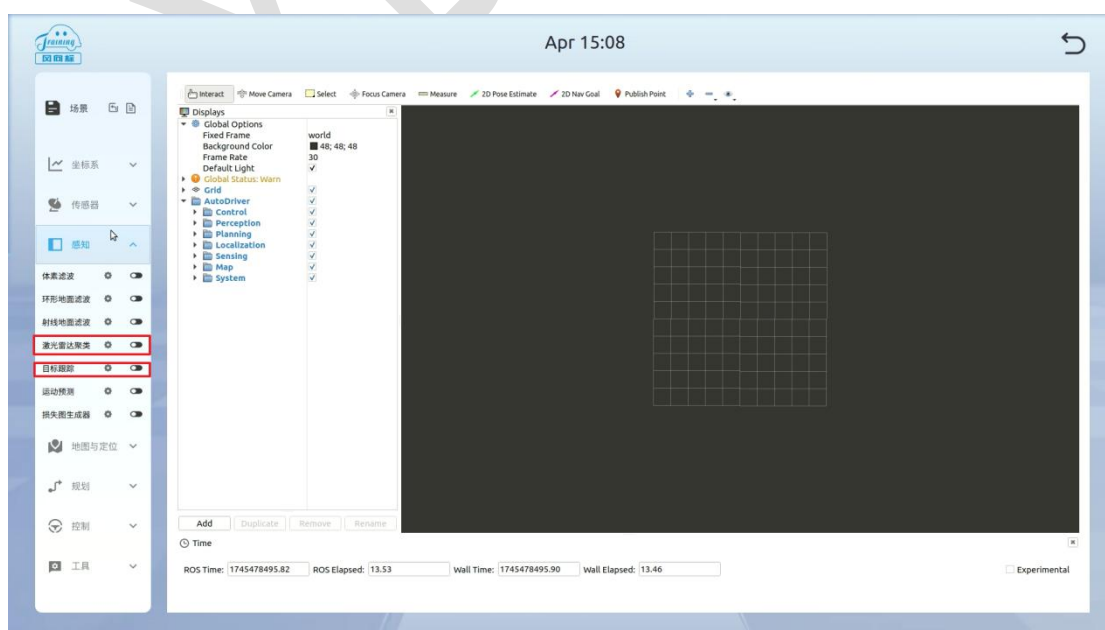
9) 使用快捷键 **ctrl+alt+t** 打开终端，输入命令 **rostopic echo /clicked_point** 后按下回车；

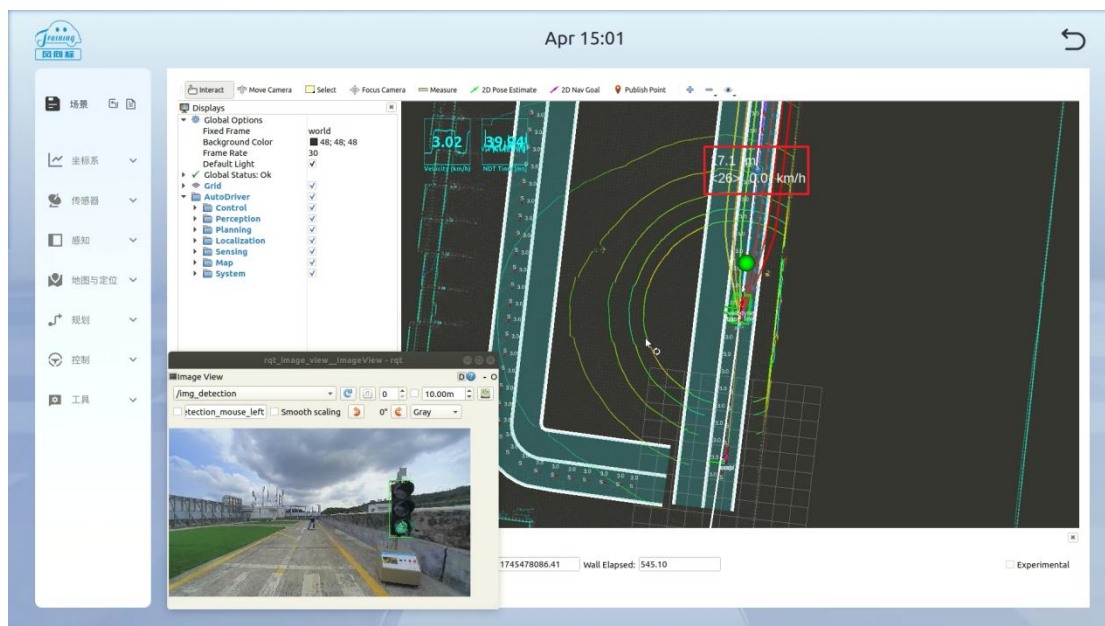


10) 点击“Publish Point”按钮后点击三色坐标，使用快捷键 **alt+tab** 切换到步骤 9 打开的终端，记录 xy 起始点坐标到报告单上；

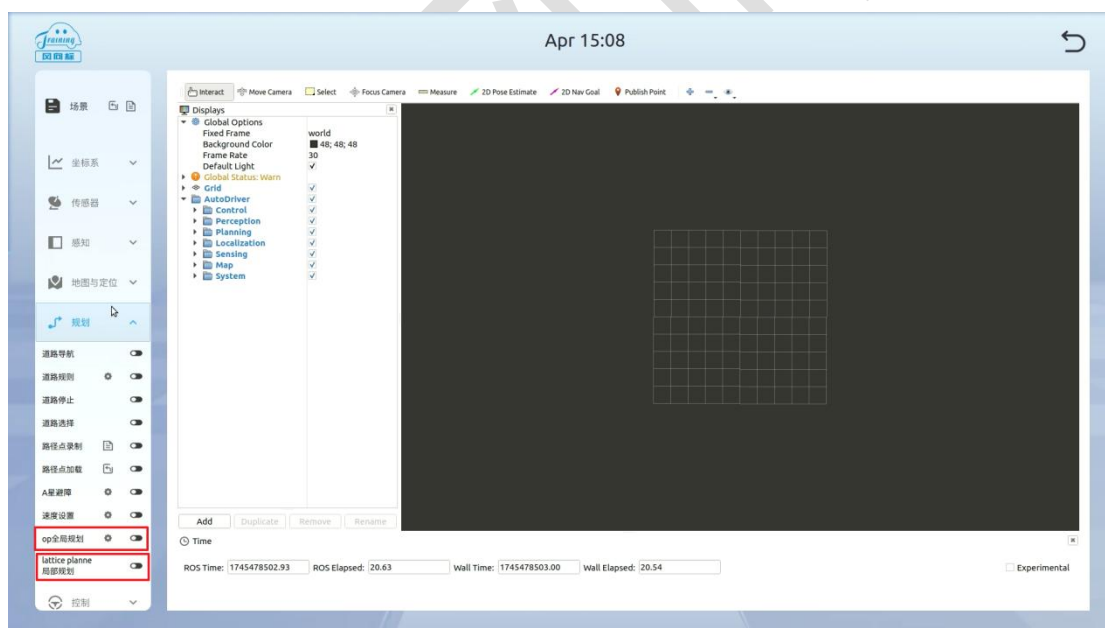


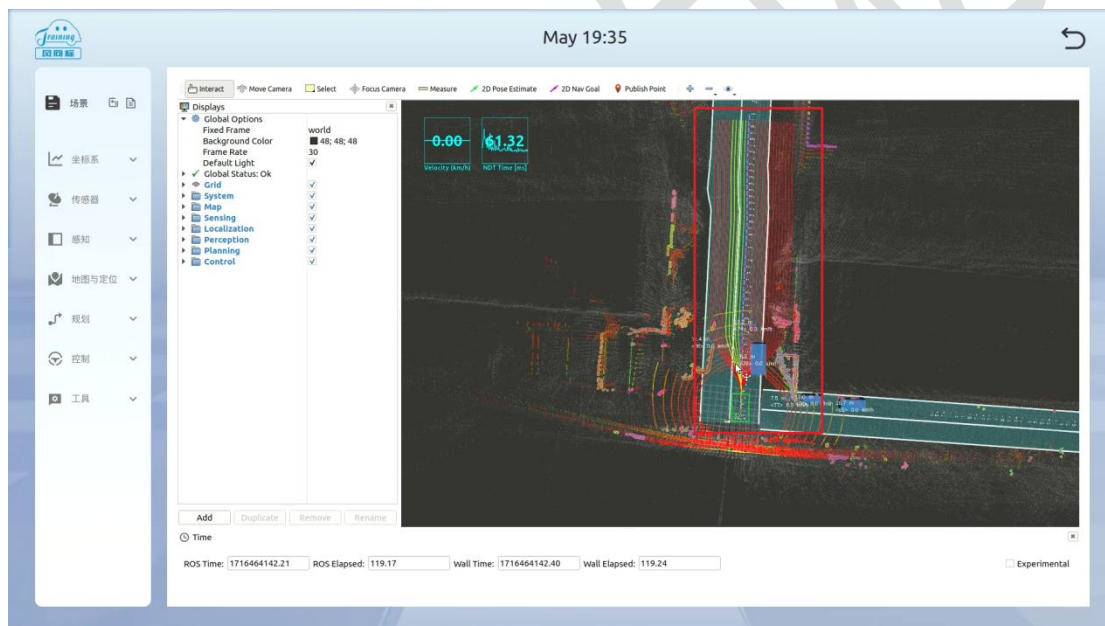
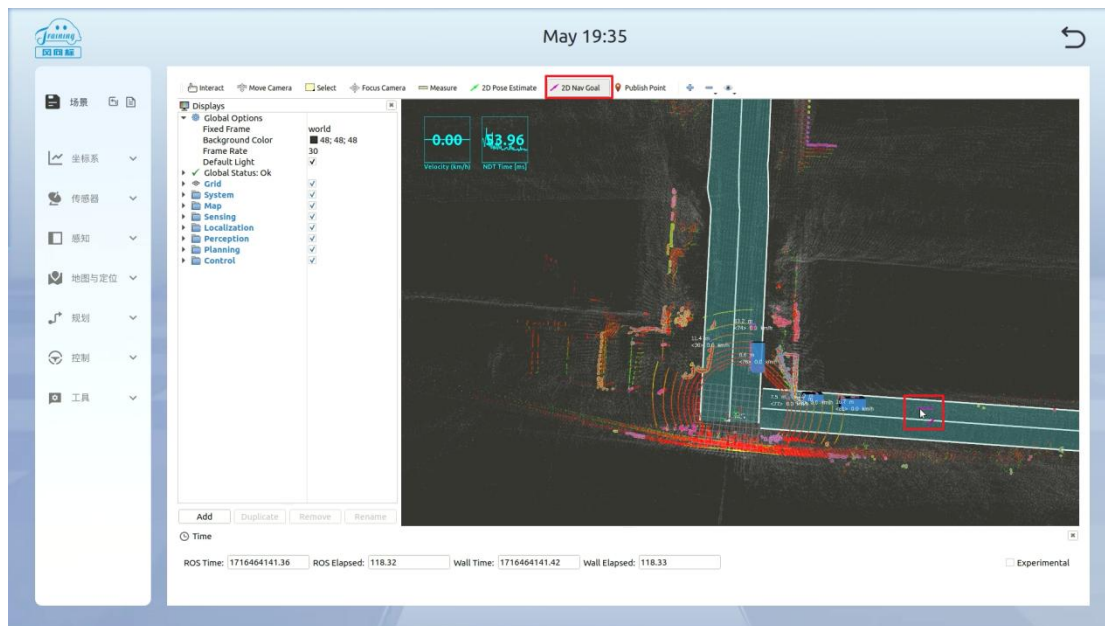
11) 找到“感知”下的“激光雷达聚类”和“目标跟踪”功能并勾选启动，等待几秒钟后，在高精地图上会看到物体检测跟踪的效果，说明后台算法启动成功；



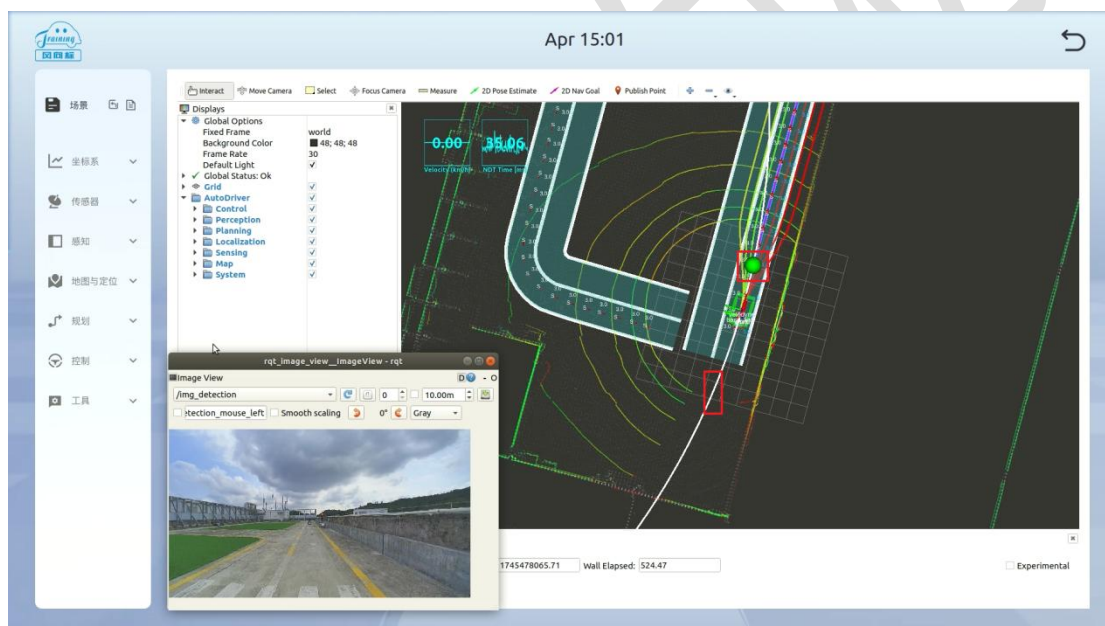
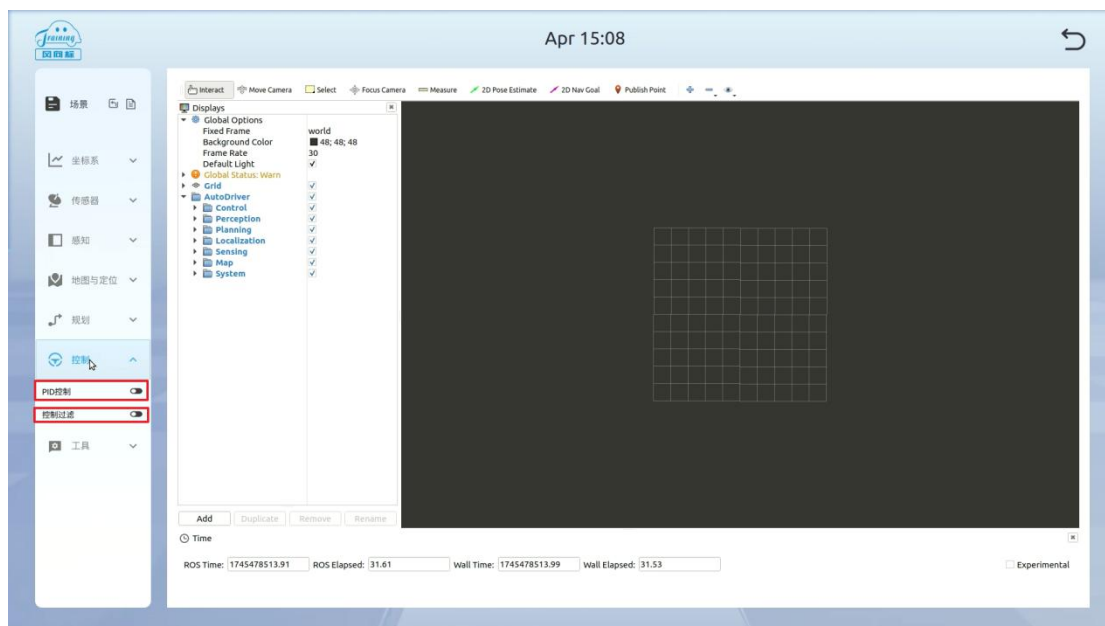


12) 找到“规划”下的“op 全局规划”和“lattice planne 局部规划”功能并勾选启动，等待几秒后，点击“2D Nav Goal”按钮，在高精地图同一个车道上，选择终点位置点击并拖动（左键点击后不松手，使用触摸板拖向车道方向后再松手）；





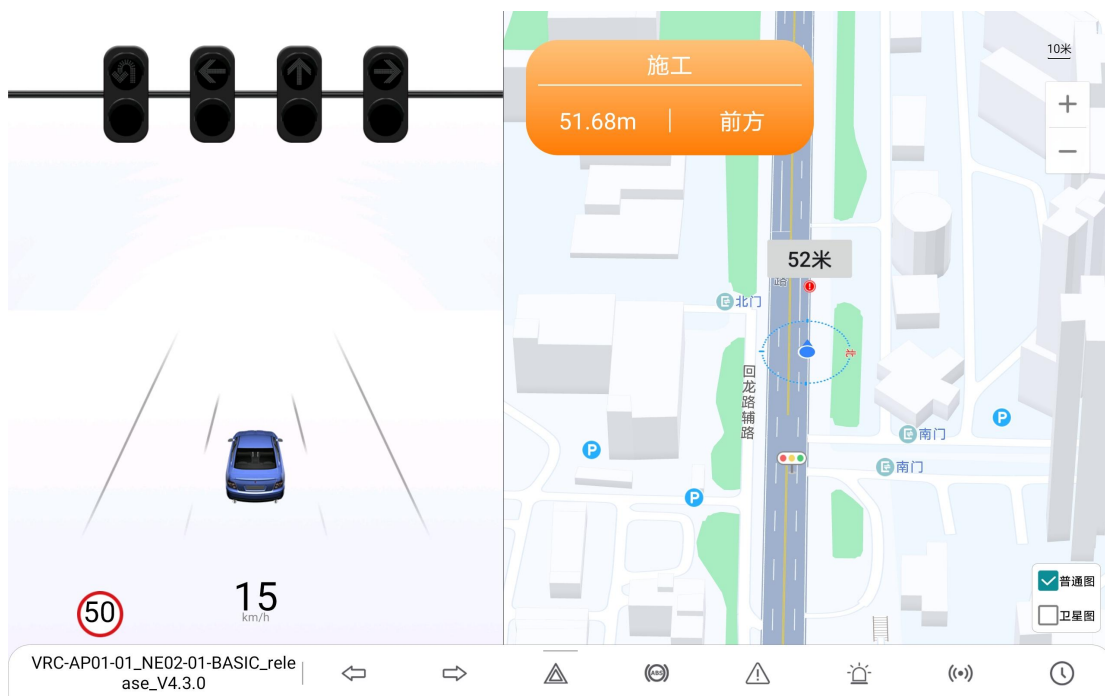
13) 找到“控制”下的“PID 控制”和“控制过滤”功能并勾选启动，当出现转向弧线和跟踪大绿点说明后台算法启动成功；

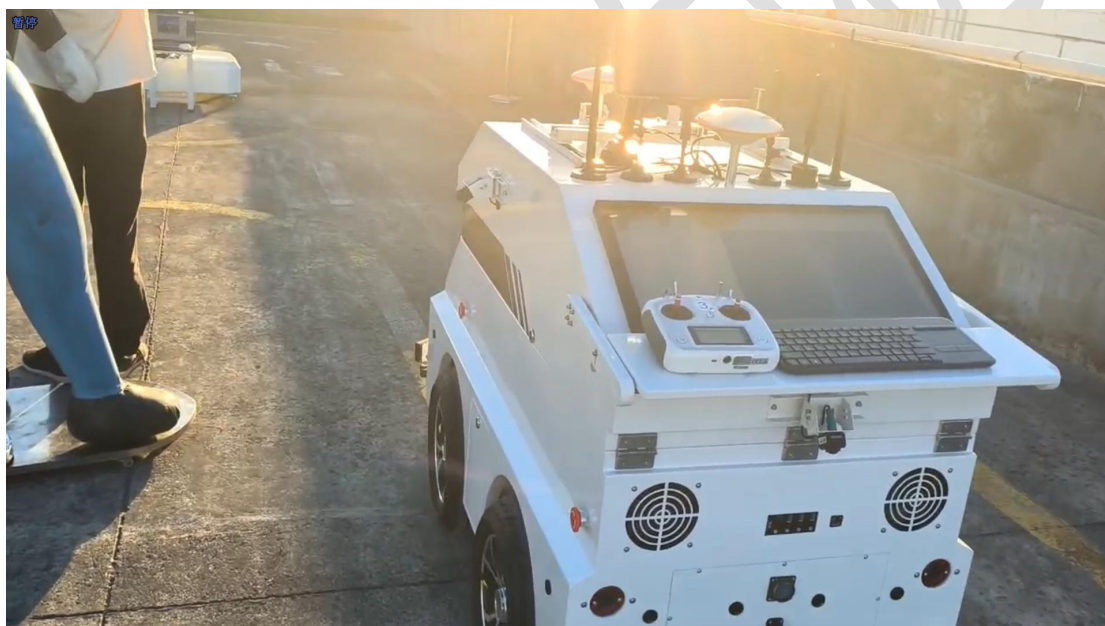
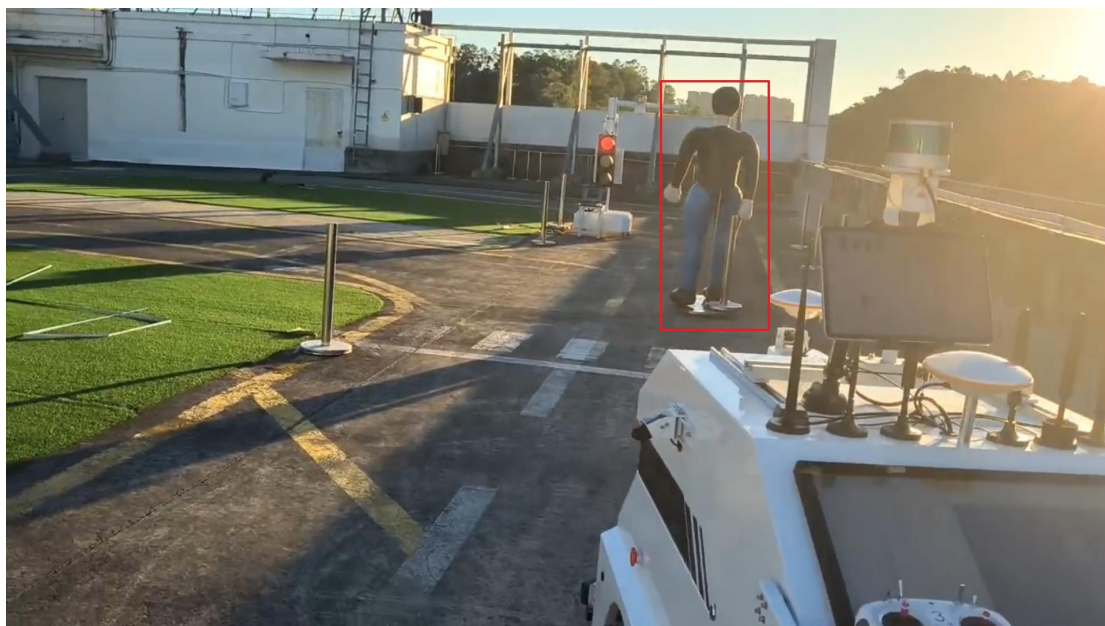


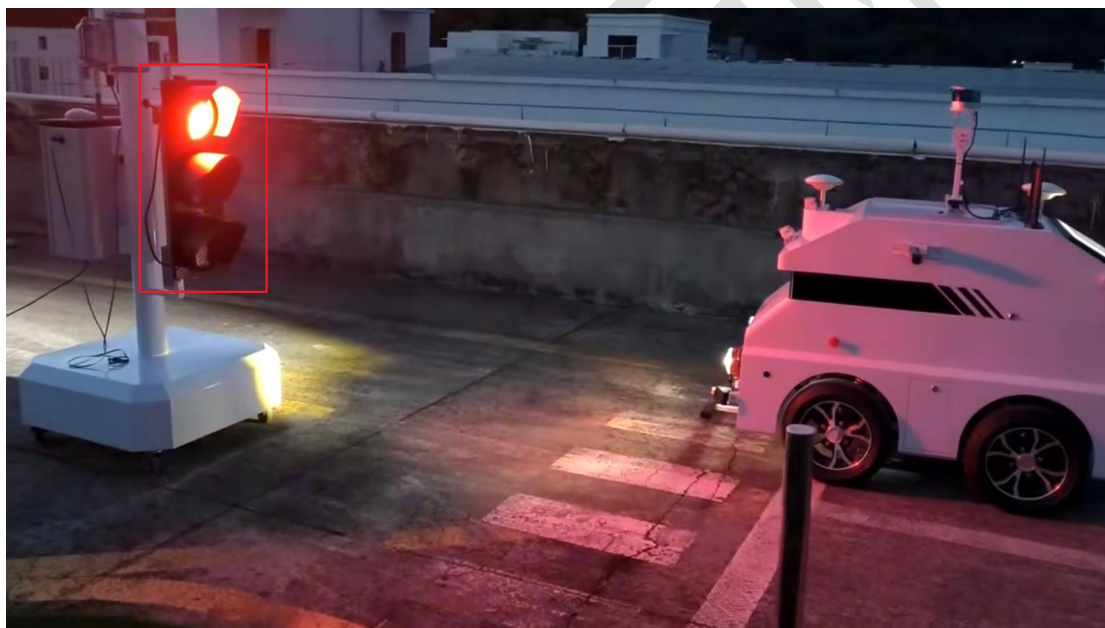
14) 将遥控器控制模式设置为自动驾驶模式，车辆将会按照指定路径行驶，遇到相关事件、标识、红绿灯时车辆将会执行对应操作同时 PAD 将播报事件名称；

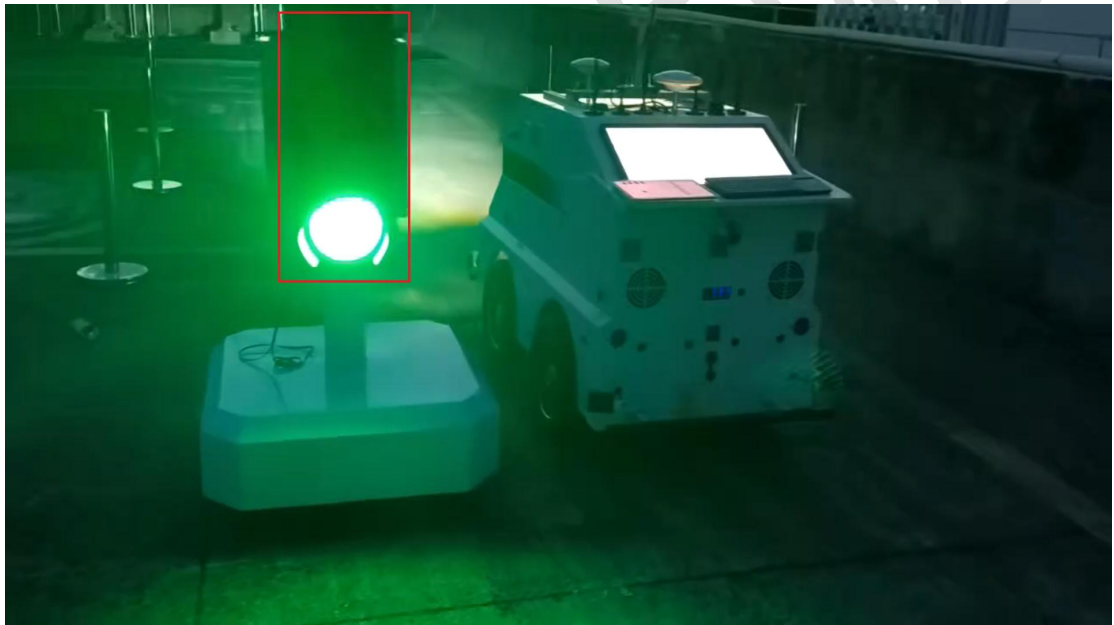
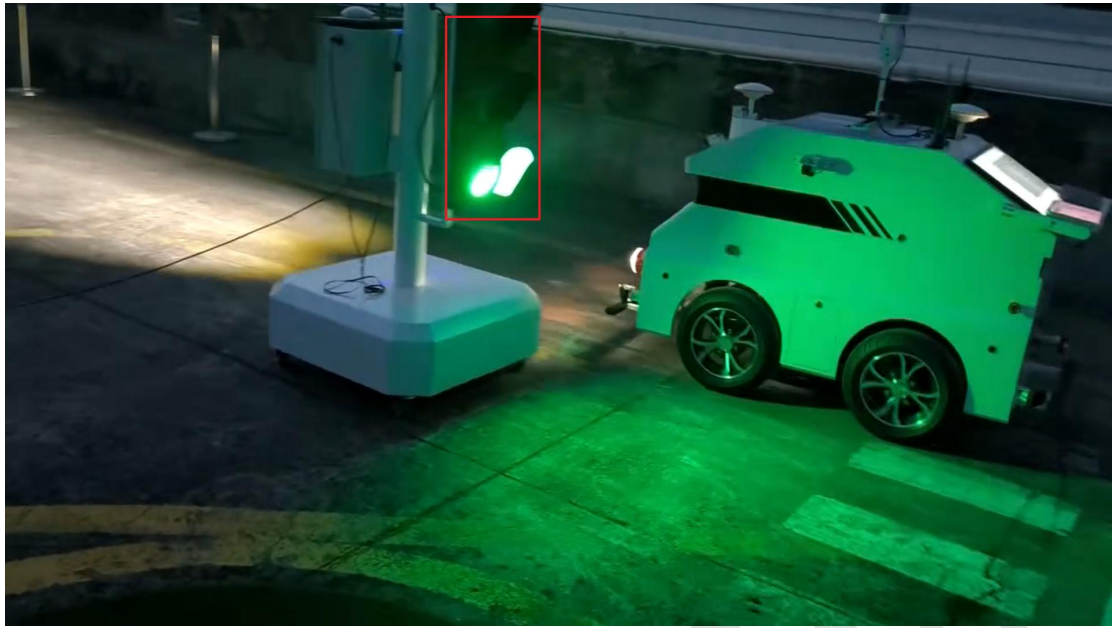


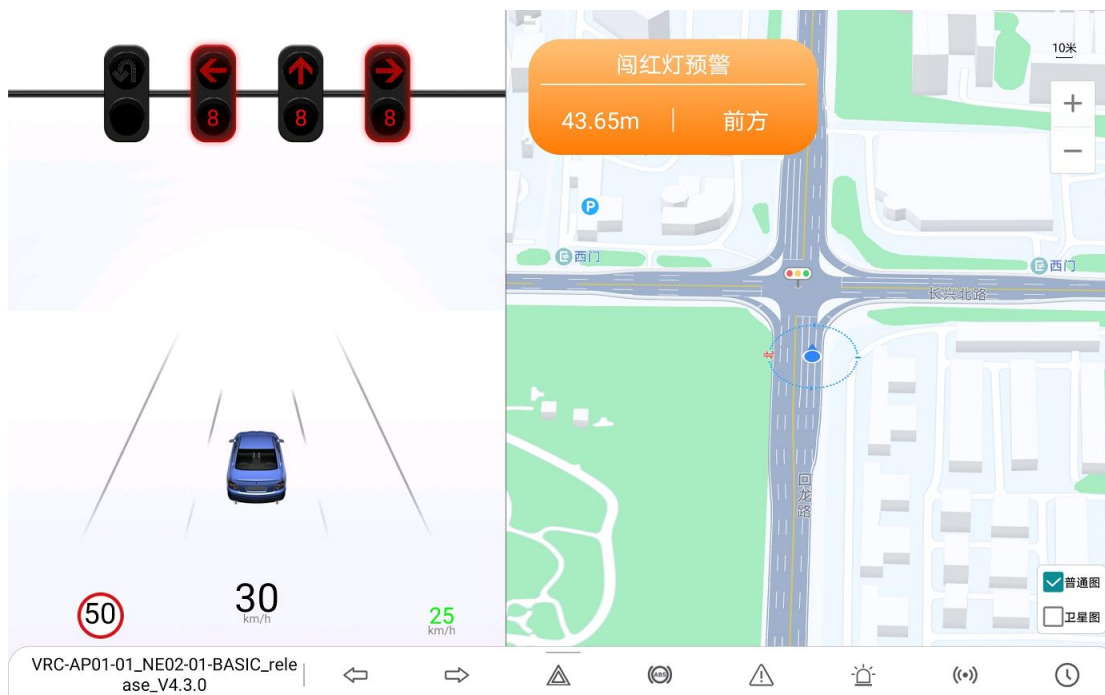




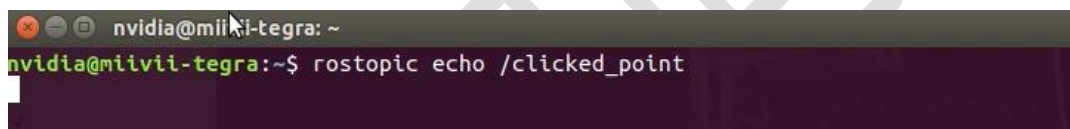




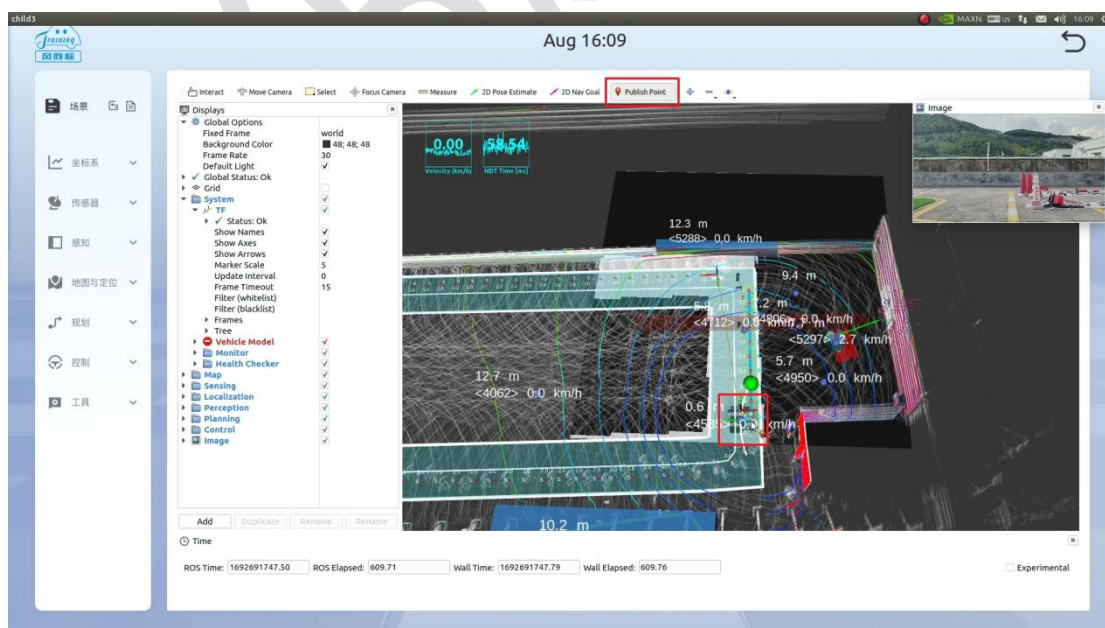




15) 当车辆行驶到终点时, 使用快捷键 `ctrl+alt+t` 打开终端, 输入命令 `rostopic echo /clicked_point` 后按下回车;



16) 点击“Publish Point”按钮后点击三色坐标, 使用快捷键 `alt+tab` 切换到步骤 15 打开的终端, 记录 `xy` 终点坐标到报告单上;



3.14 道路测试-初级教学-OBV

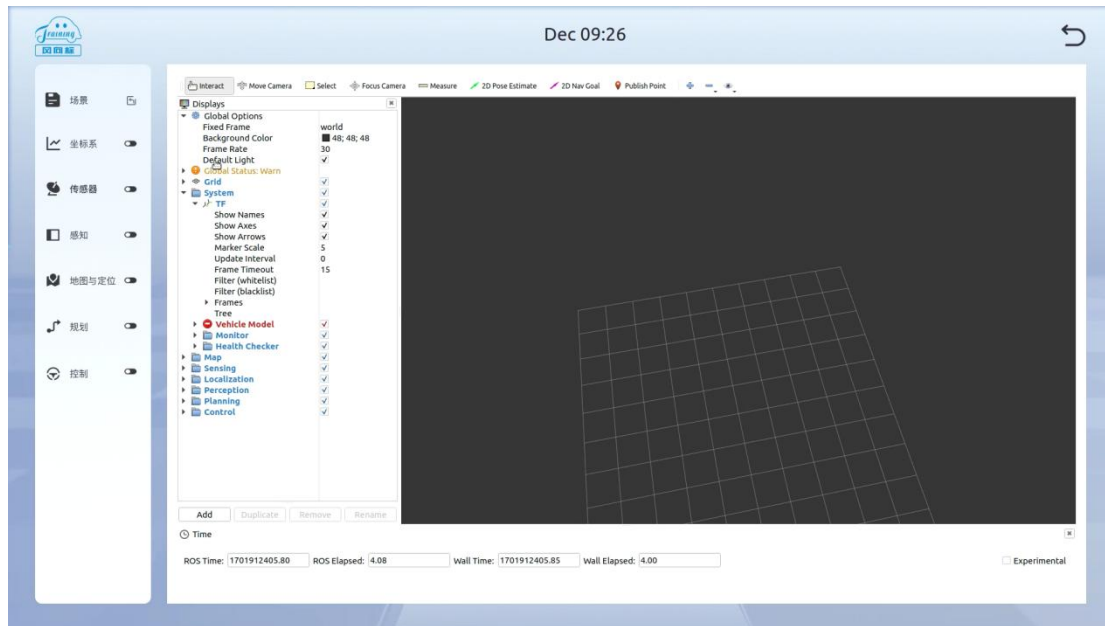
0) 在关闭所有软件和终端后再操作本小节；

1) 使用遥控器将车辆行驶到起始区域；

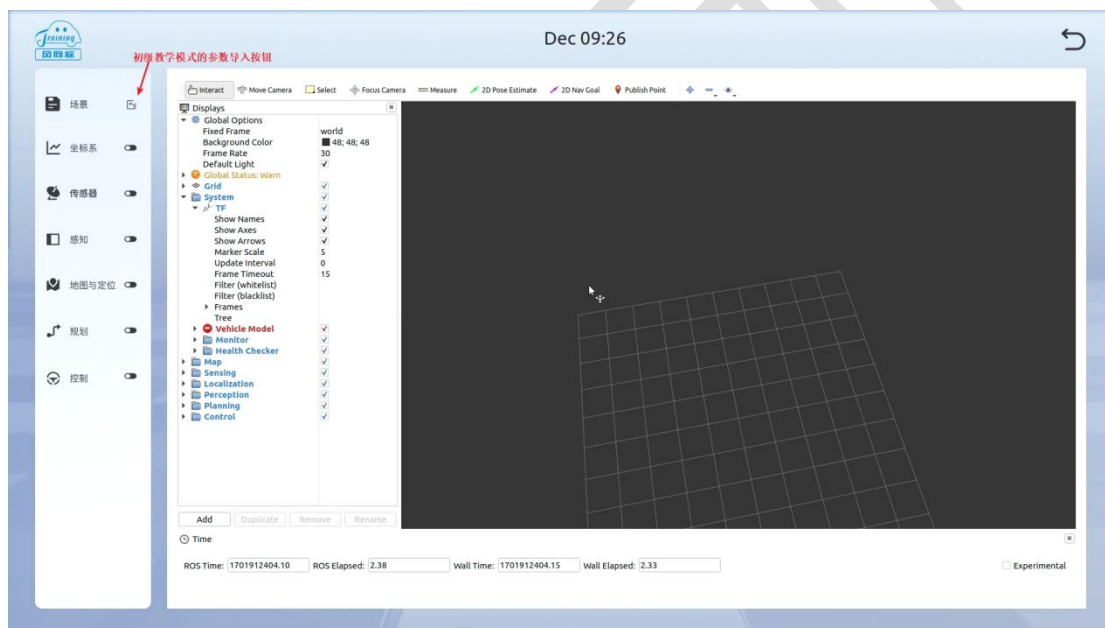


2) 打开自动驾驶教学软件，选择“初级教学”模式；

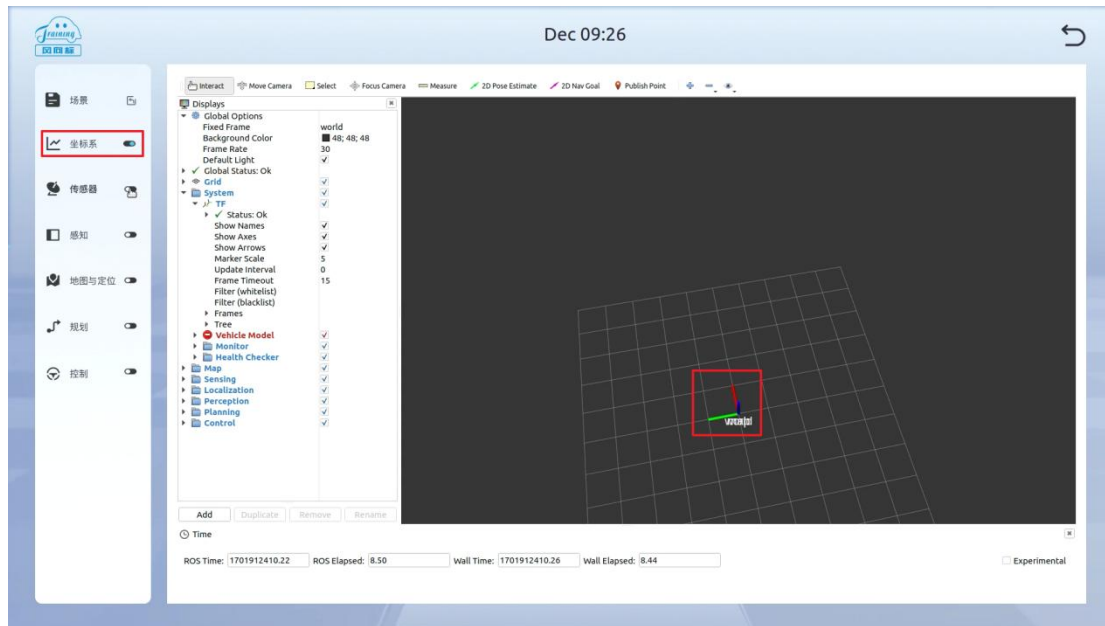




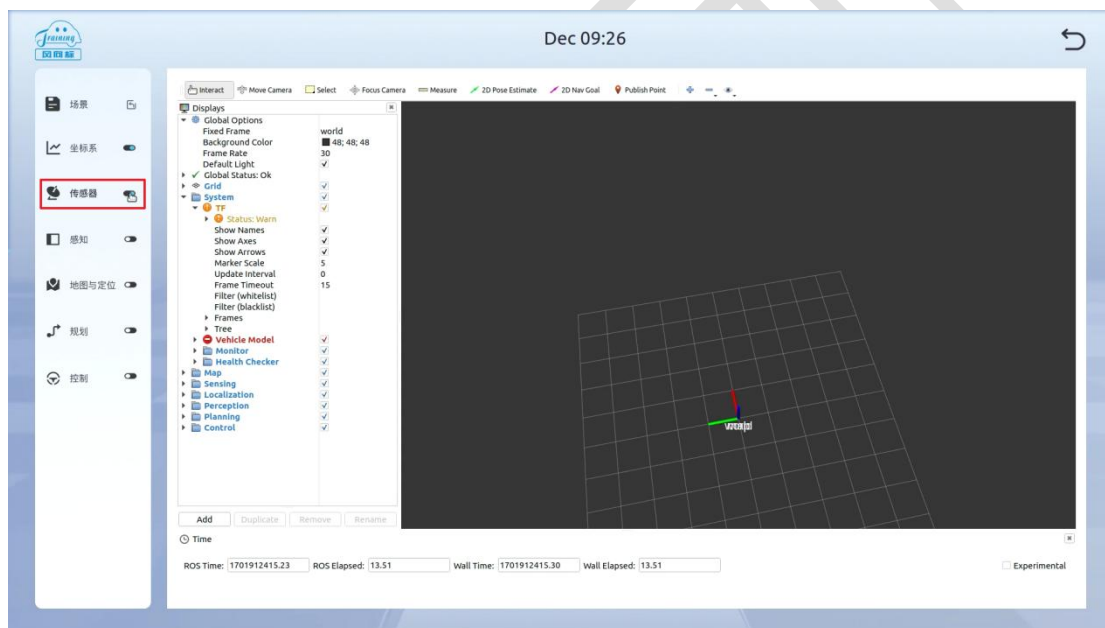
3) 导入从 3.6 小节导出的参数，选择方案二；



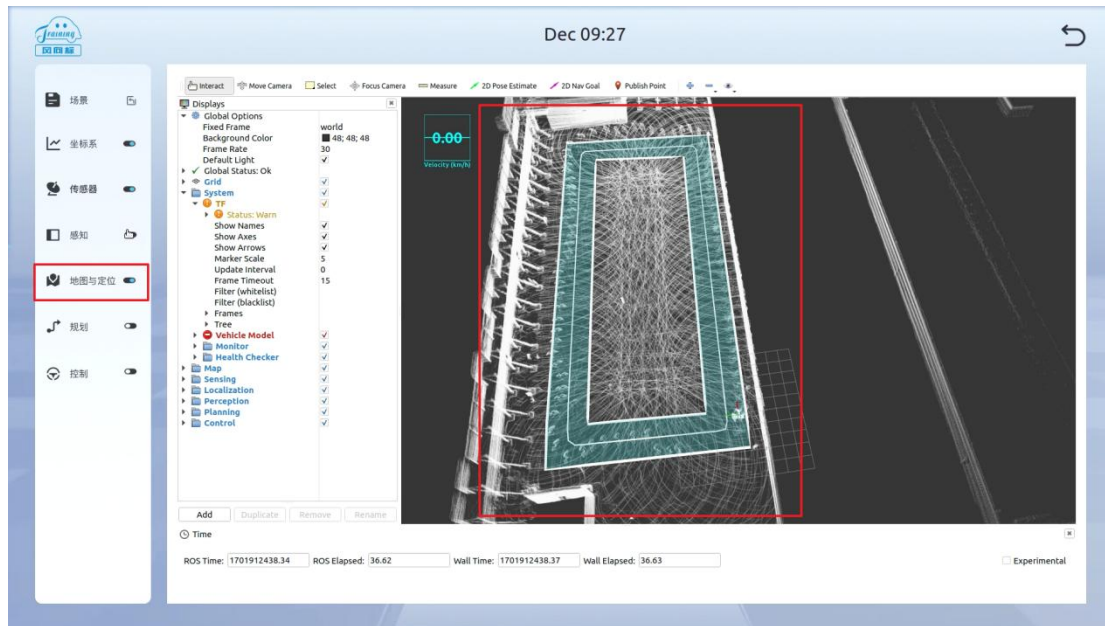
4) 勾选启动“坐标系”，启动成功后会出现“三色坐标系”；



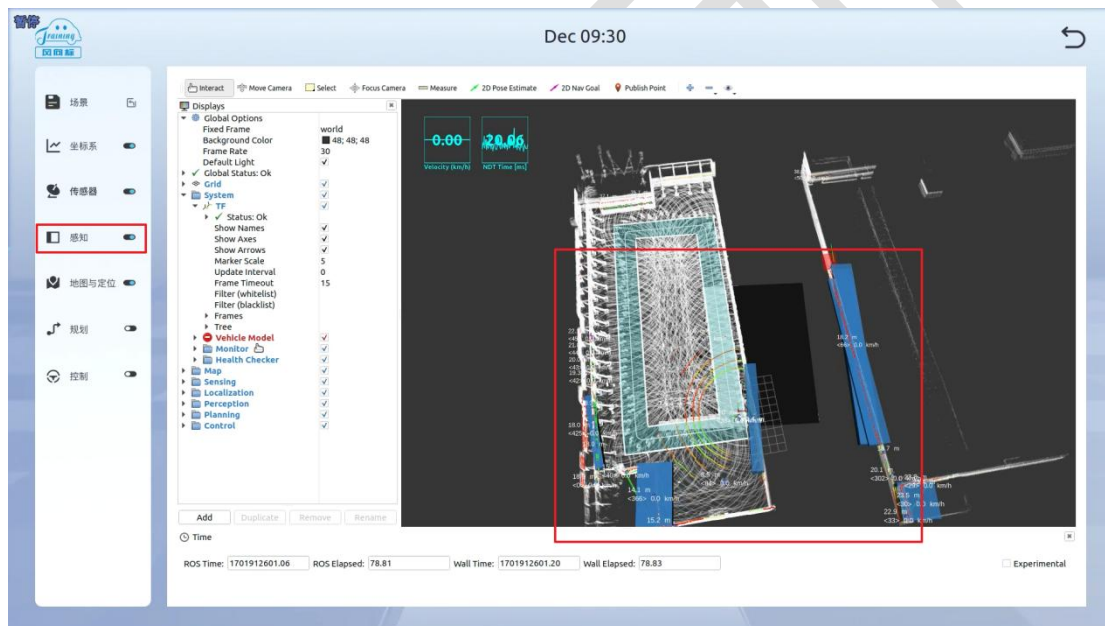
5) 勾选启动“传感器”；



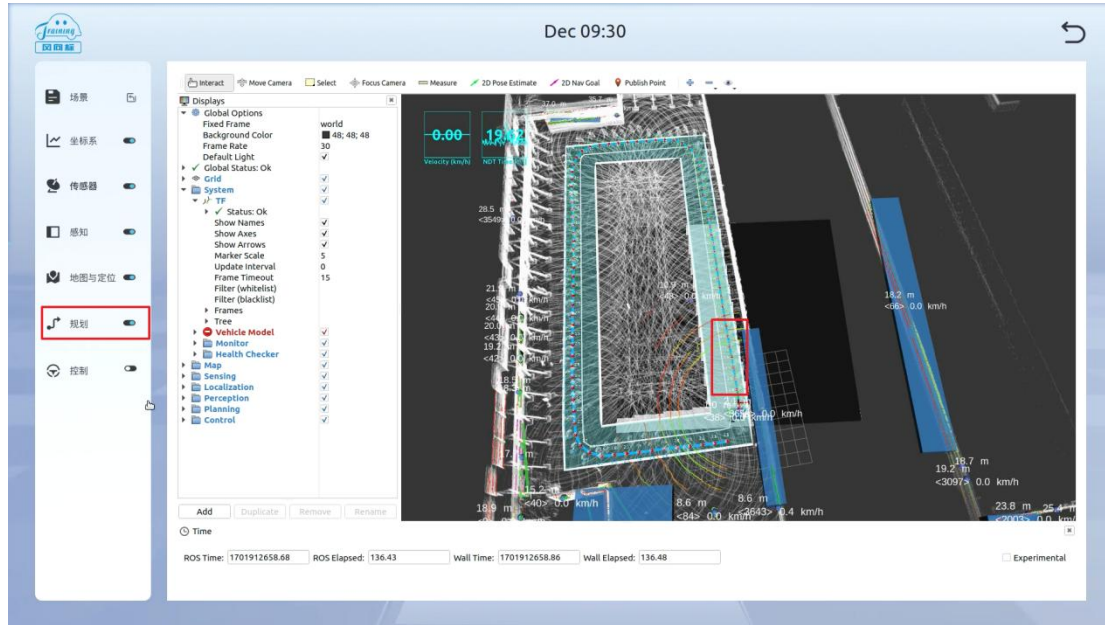
6) 勾选启动“地图与定位”，启动成功后会出现白色的 PCD 地图和青绿色的 HD 高精地图；



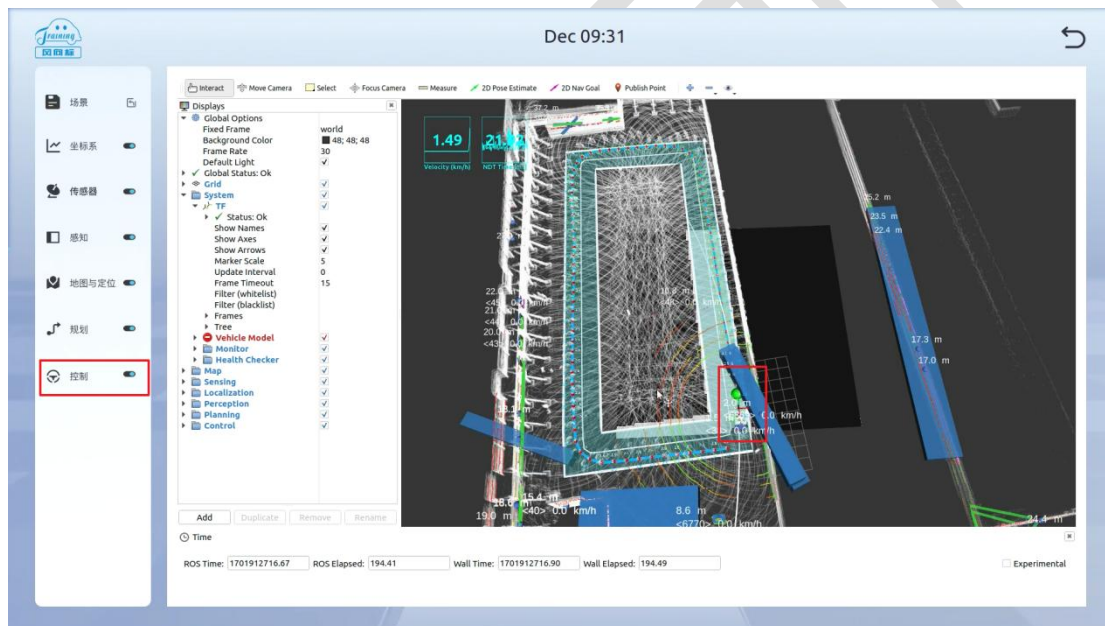
7) 勾选启动“感知”，启动成功后会出现蓝色矩形；



8) 勾选启动“规划”，点击“2D Nav Goal”按钮，在高精地图同一个车道上，选择终点位置点击并拖动（左键点击后不松手，使用触摸板拖向车道方向后再松手），启动成功后会出现多条不同颜色的候选路线；



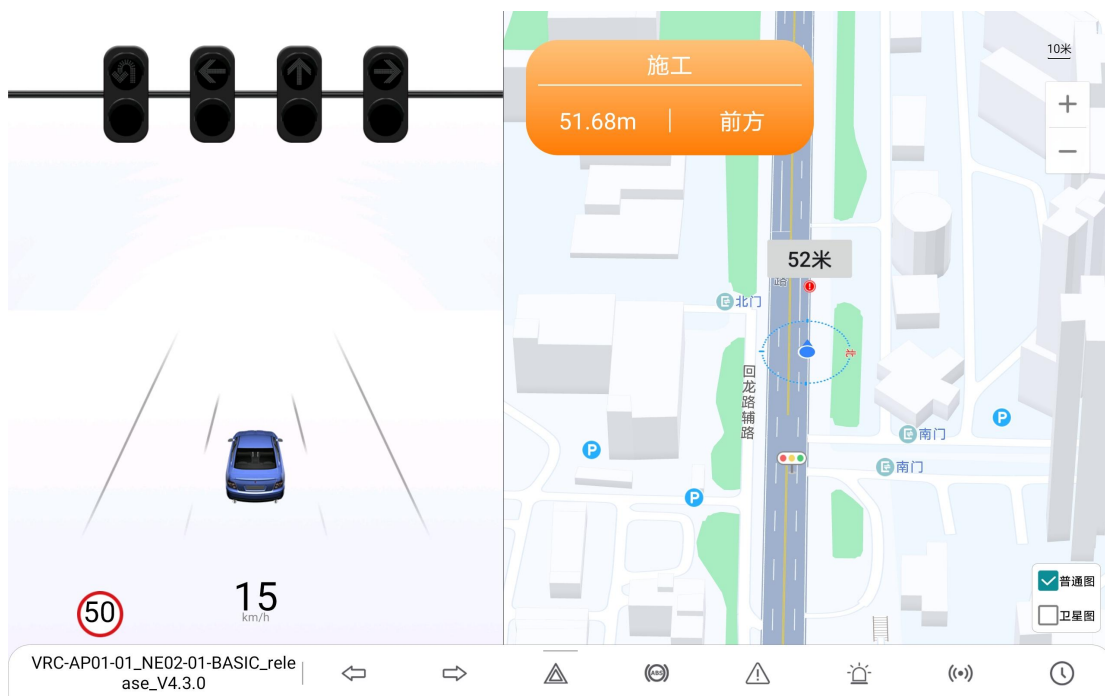
9) 勾选启动“控制”，启动成功后会出现大绿点和转向弧线；

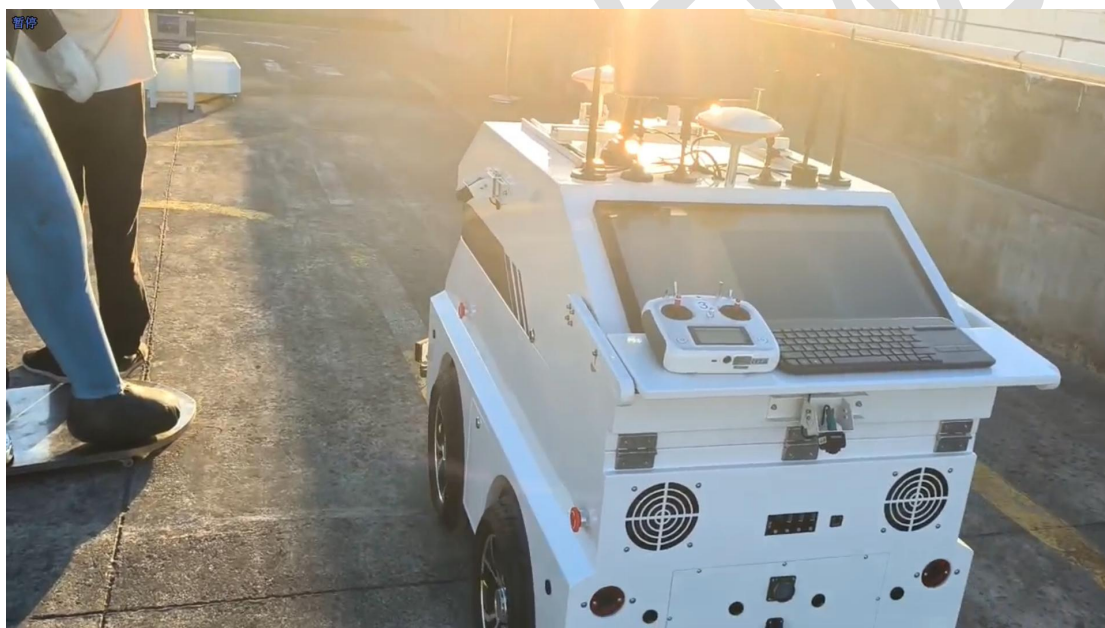
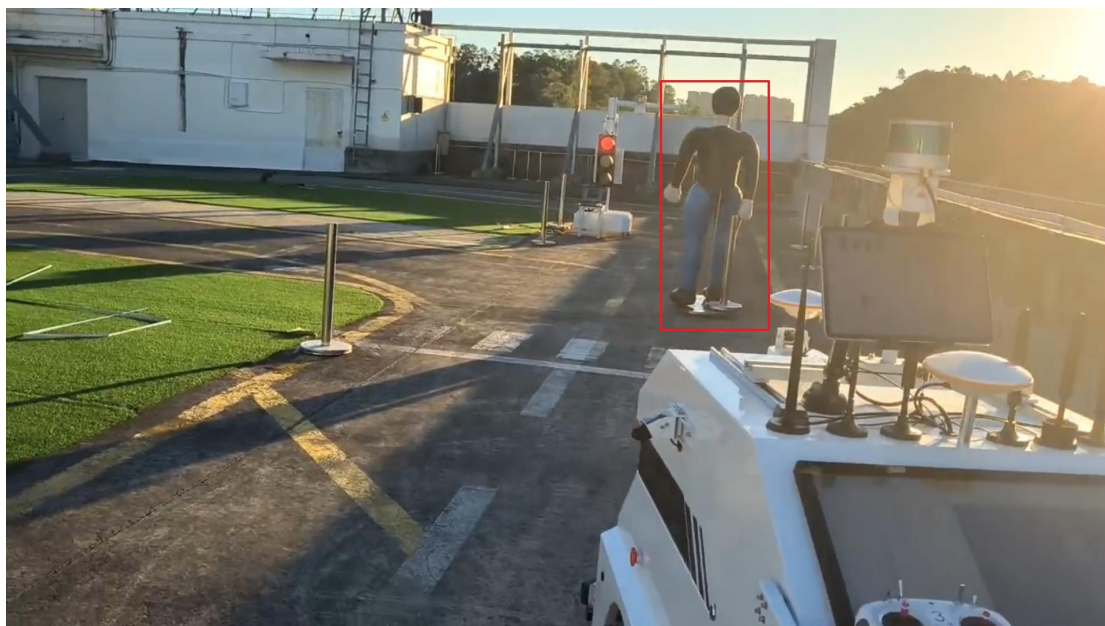


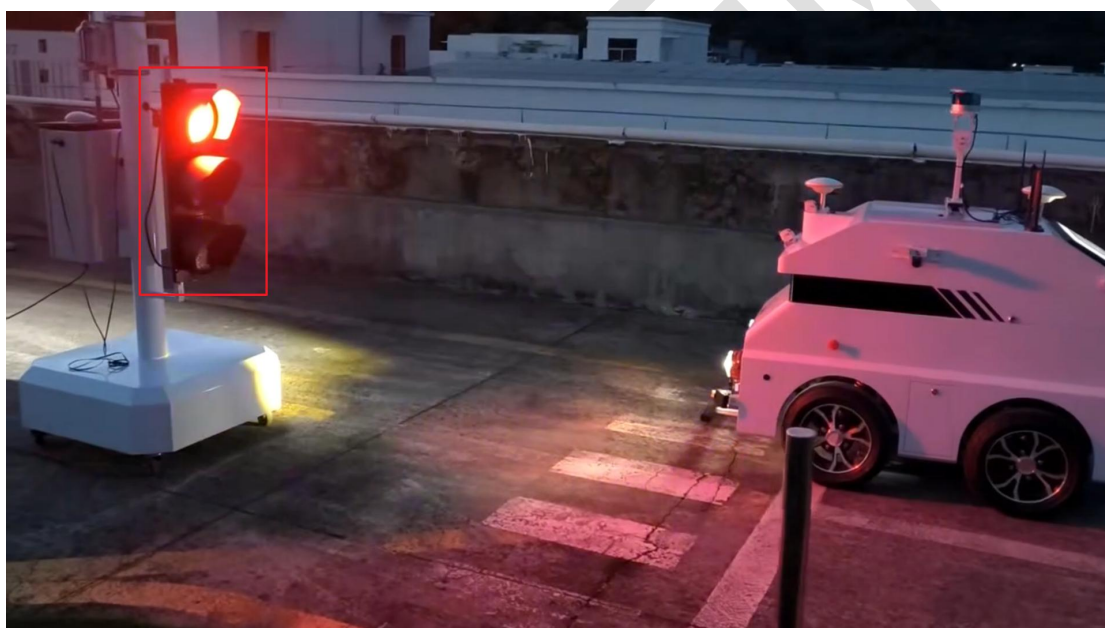
10) 将遥控器控制模式设置为自动驾驶模式，车辆将会按照指定路径行驶，遇到相关事件、标识、红绿灯时车辆将会执行对应操作同时 PAD 将播报事件名称；

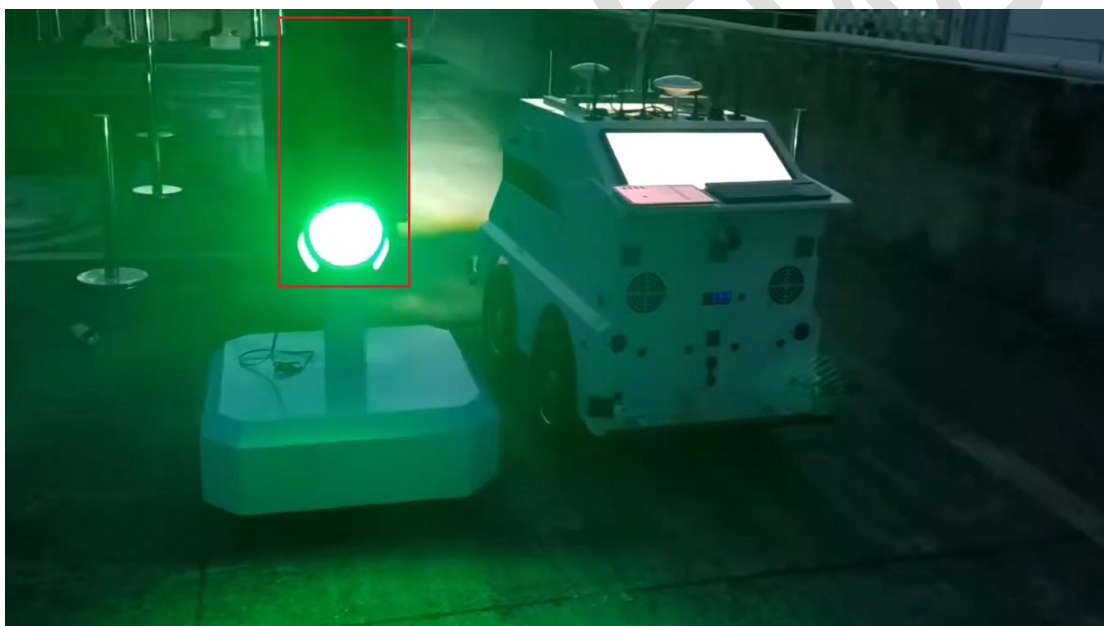
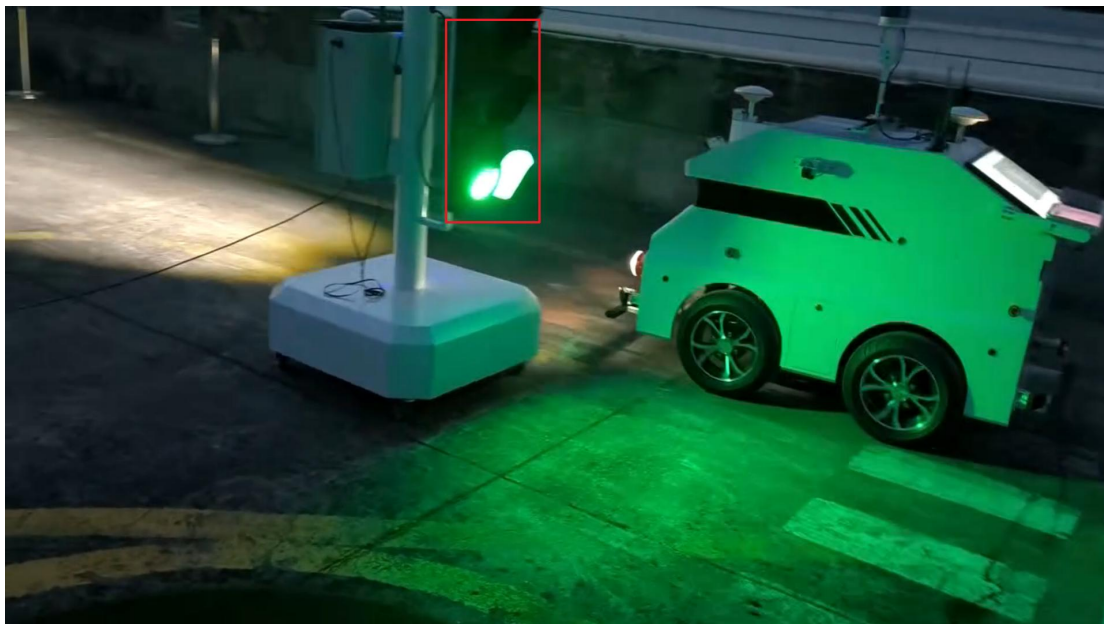


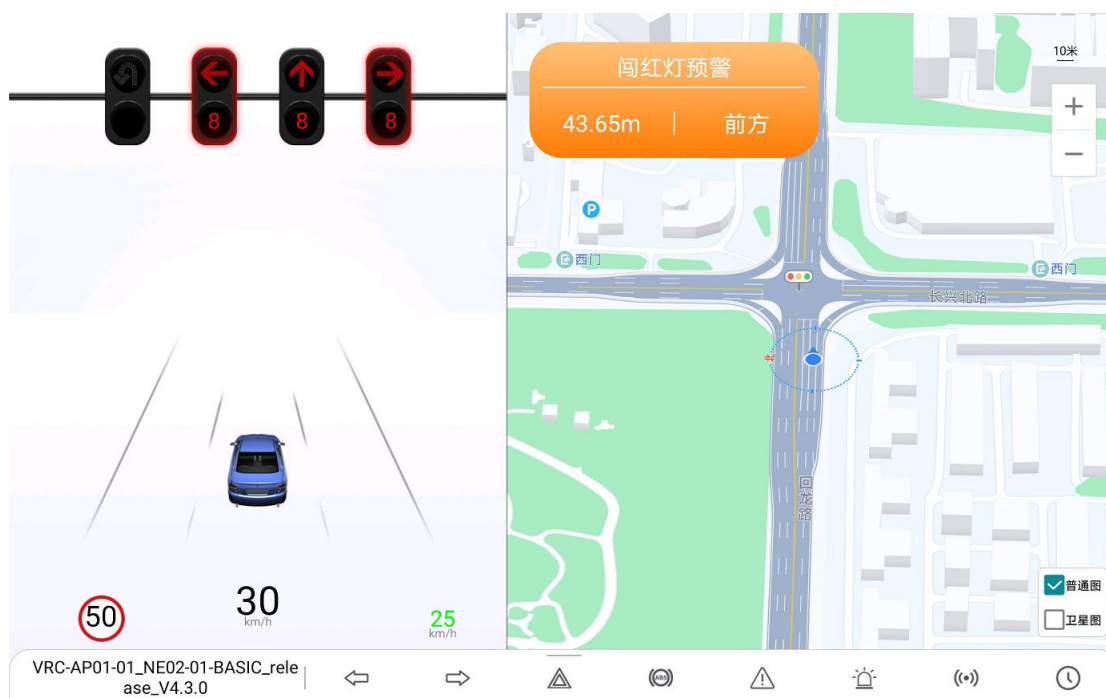










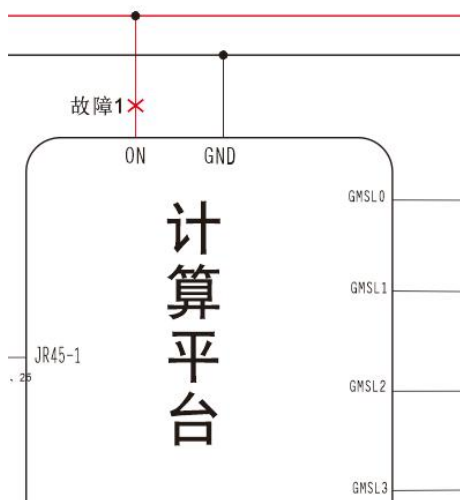


第 4 章 故障排查与考核实训

4.1 实验一

实验准备：能正常工作的万用表、专用诊断仪、故障设置设备。

实验对象：计算平台电源线断路故障。



实验目的：了解计算平台的工作原理，理解计算平台的线路走向。

实验现象：显示屏黑屏，计算平台风扇不转，无法正常开机。

故障分析：功能异常，不能正常使用的可能原因可能在于：

- (1) 总电源以及相关线路断路、虚接、短路
- (2) 计算平台电源以及本身元器件

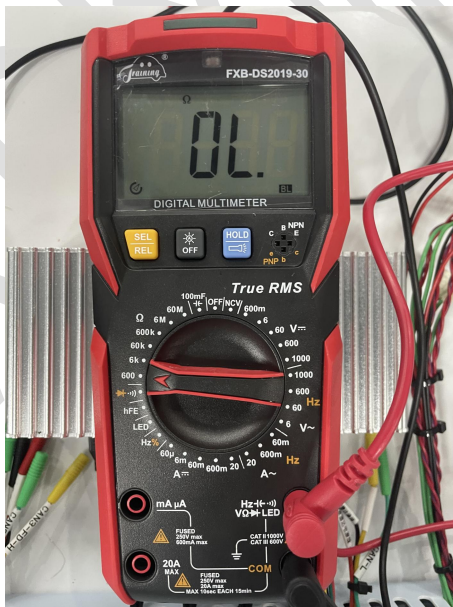
排查过程

- (1) 用万用表电压档测计算平台电源线（红表笔）对计算平台地线线（黑表笔）
实际测量电压为 0.19V，正常计算平台电压值应该为 24V 左右，判断计算平台电源可能存在故障。

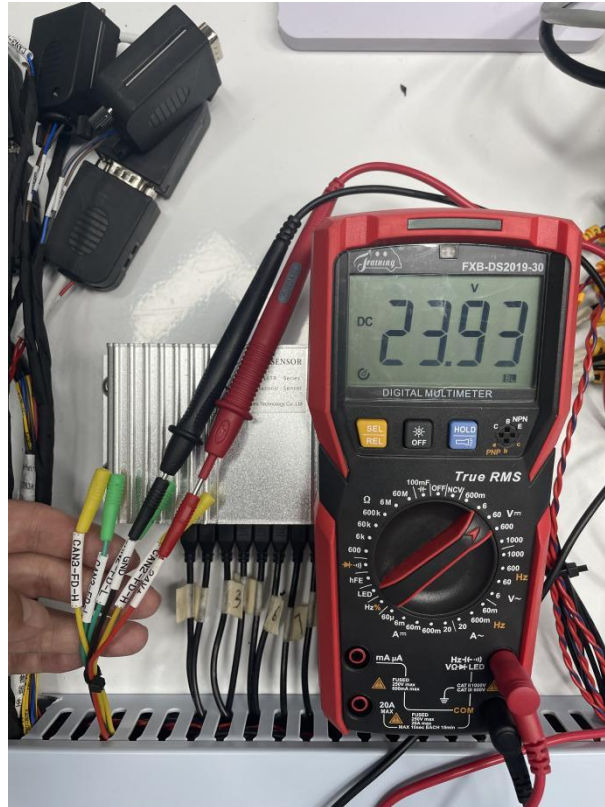


(2) 用万用表电压档测底盘给计算平台供电的 24V 电源线（红表笔）对地线（黑表笔）实际测量电压为 24V，说明总电源线路正常。可能总电源到计算平台之间供电的线路存在断路。

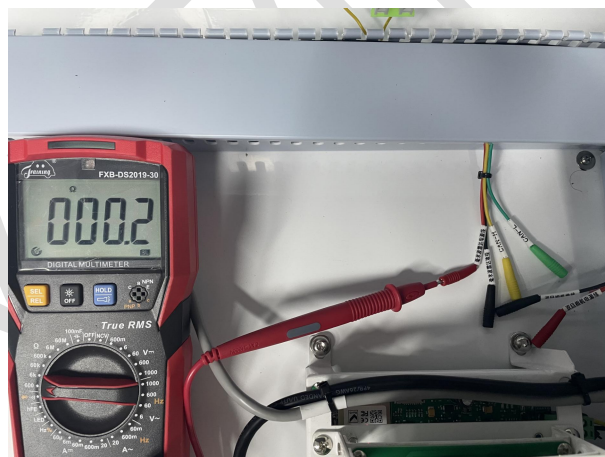
(3) 断开电源，用万用表电阻档测底盘 24V 电源端到计算平台电源线端，实际测量阻值无穷大，判断这节线路已断路。



(4) 故障恢复后再次测量电压值, 电压显示为 24V 左右正常。



(5) 再次用电阻挡，线路电阻为 0.2Ω ，电阻正常。



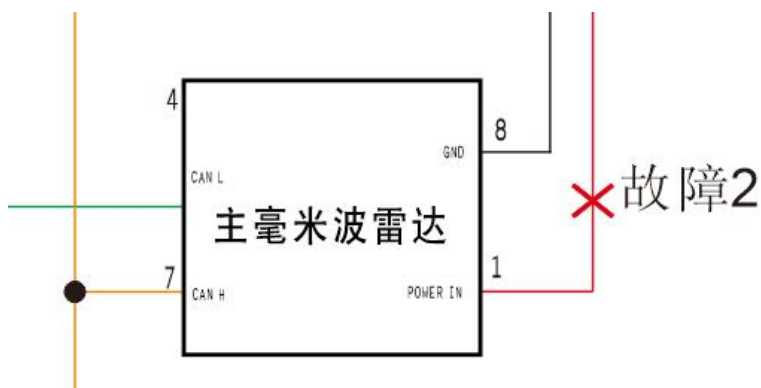
故障机理分析：由于计算平台的电源断路，导致显示屏黑屏，计算平台风扇不转，无法正常开机。

实验结果：计算平台电源线路断路

4.2 实验二

实验准备：能正常工作的万用表、专用诊断仪、故障设置设备。

实验对象：毫米波雷达电源线断路故障。



实验目的：了解毫米波的工作原理，理解毫米波的线路走向。

实验现象：使用上位软件检测实物无法成像，无数据显示，功能使用异常。

故障分析：功能异常，不能正常使用的可能原因可能在于：

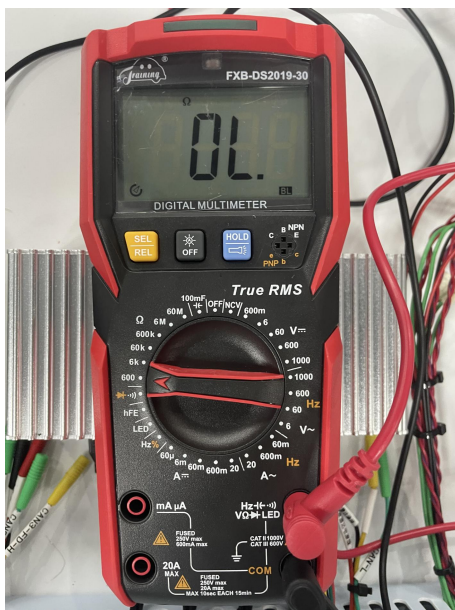
- (1) 电源以及相关线路
- (2) CAN 线断路、虚接、短路、反接
- (3) 毫米波电源以及本身元器件

排查过程

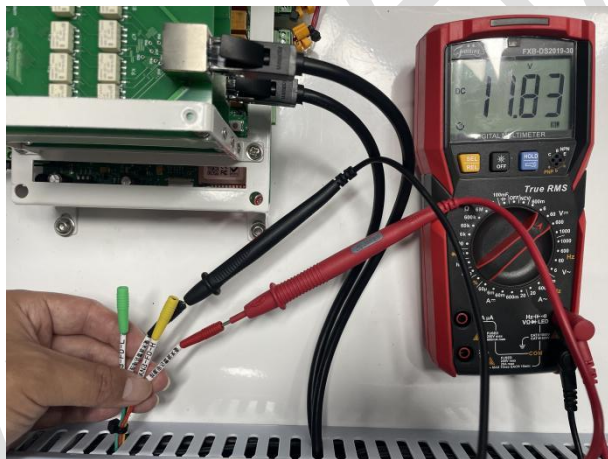
- (1) 用万用表电压档测毫米波雷达电源线（红表笔）对毫米波雷达地线线（黑表笔）实际测量电压为 0.64V，正常传感器电压值应该为 12V 左右，判断毫米波电源可能存在故障。



- (2) 用万用表电压档测底盘给传感器供电的 12V 电源线（红表笔）对地线（黑表笔）实际测量电压为 12V，说明总电源线路正常。可能总电源到毫米波雷达供电线路存在断路。
- (3) 断开电源，用万用表电阻档测底盘 12V 电源端到毫米波雷达电源线端，实际测量阻值无穷大，判断这节线路已断路。



(4) 故障恢复后再次测量电压值, 电压显示为 13.62V 正常。



(5) 再次用电阻挡, 线路电阻为 $0.2\ \Omega$, 电阻正常。



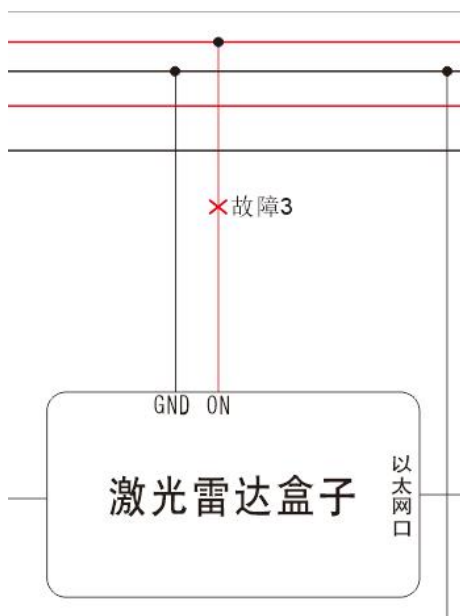
故障机理分析：由于毫米波电源断路，导致使用上位软件检测实物无法成像，无数据显示，功能使用异常。

实验结果：毫米波雷达电源线路断路

4.3 实验三

实验准备：能正常工作的万用表、专用诊断仪、故障设置设备。

实验对象：激光雷达电源线断路故障。



实验目的：了解激光雷达的工作原理，理解激光雷达的线路走向。

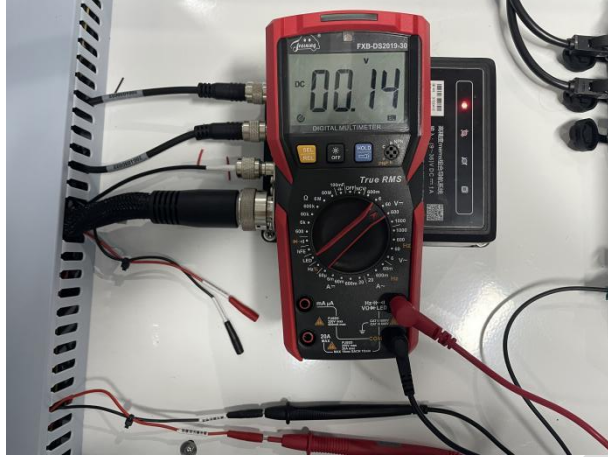
实验现象：使用上位软件无法成 3D 点云图像，无数据显示，功能使用异常。

故障分析：功能异常，不能正常使用的可能原因可能在于：

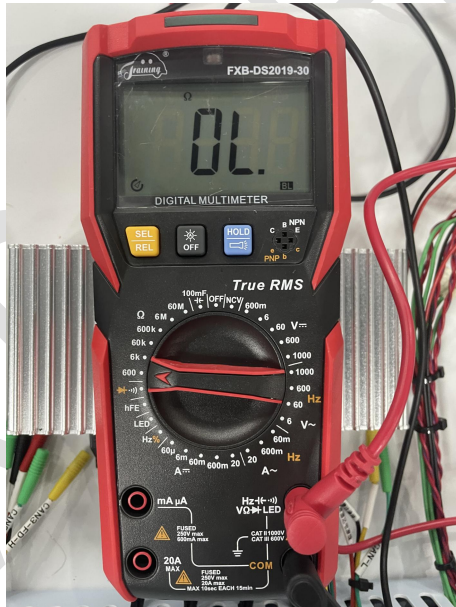
- (1) 电源以及相关线路
- (2) 网线断路、虚接
- (3) 激光雷达以及本身元器件

排查过程

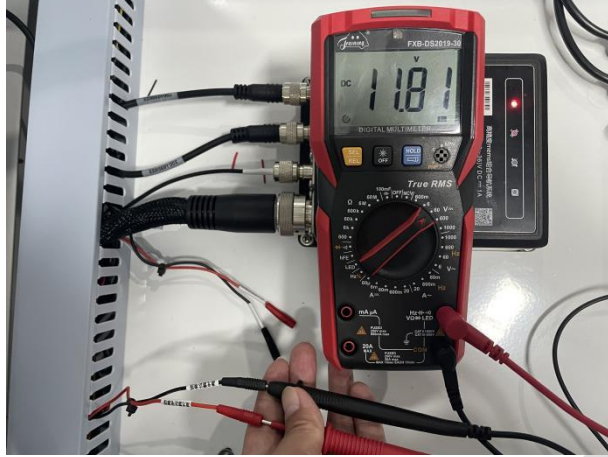
- (1) 用万用表电压档测激光雷达电源线（红表笔）对激光雷达地线线（黑表笔）
实际测量电压为 0.14V，正常传感器电压值应该为 12V 左右，判断激光雷达电源可能存在故障。



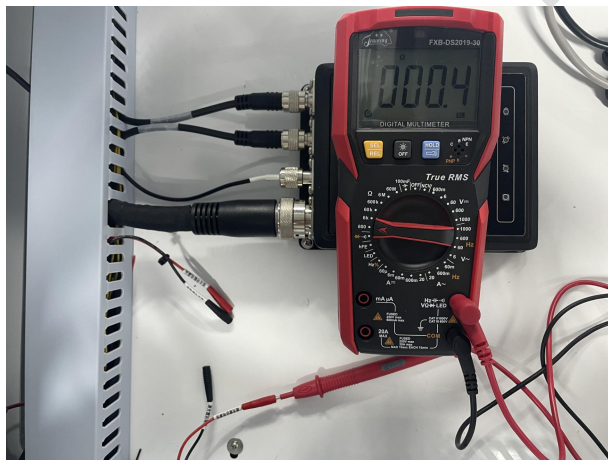
- (2) 用万用表电压档测底盘给传感器供电的 12V 电源线（红表笔）对地线（黑表笔）实际测量电压为 12V，说明总电源线路正常。可能总电源到激光雷达供电线路存在断路。
- (3) 断开电源，用万用表电阻档测底盘 12V 电源端到激光雷达电源线端，实际测量阻值无穷大，判断这节线路已断路。



- (4) 故障恢复后再次测量电压值，电压显示为 12V 左右正常。



(5) 再次用电阻挡，线路电阻为 $0.2\ \Omega$ ，电阻正常。



故障机理分析：由于激光雷达的电源断路，导致使用上位软件无法成 3D 点云图像，无数据显示，功能使用异常。

实验结果：激光雷达电源线路断路

4.4 实验四

实验准备：能正常工作的万用表、专用诊断仪、故障设置设备。

实验对象：单目相机电源线断路故障。



实验目的：了解单目相机的工作原理，理解单目相机的线路走向。

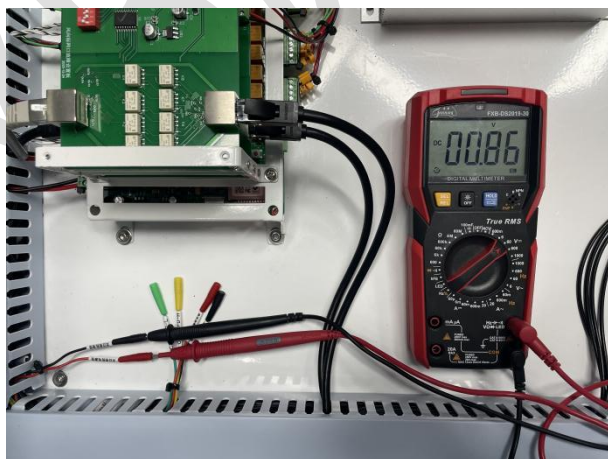
实验现象：使用上位软件无图像呈现，功能使用异常。

故障分析：功能异常，不能正常使用的可能原因可能在于：

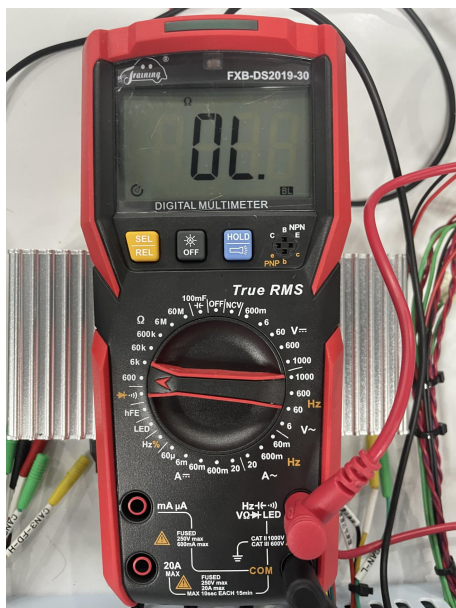
- (1) 电源以及相关线路
- (2) 网线断路、虚接
- (3) 单目相机以及本身元器件

排故过程

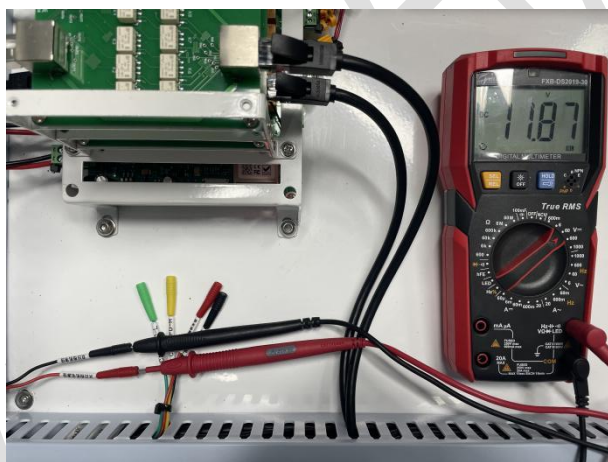
- (1) 用万用表电压档测单目相机电源线（红表笔）对单目相机地线线（黑表笔）实际测量电压为 0.86V，正常传感器电压值应该为 12V 左右，判断单目相机电源可能存在故障。



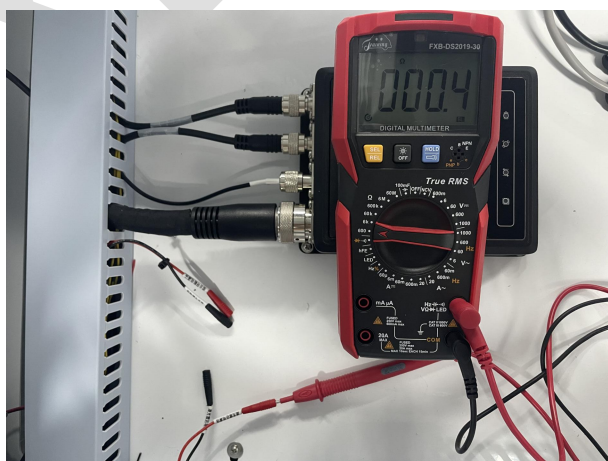
- (2) 用万用表电压档测底盘给传感器供电的 12V 电源线（红表笔）对地线（黑表笔）实际测量电压为 12V，说明总电源线路正常。可能总电源到激光雷达供电线路存在断路。
- (3) 断开电源，用万用表电阻档测底盘 12V 电源端到单目相机电源线端，实际测量阻值无穷大，判断这节线路已断路。



(4) 故障恢复后再次测量电压值, 电压显示为 12V 左右正常。



(5) 再次用电阻挡, 线路电阻为 $0.4\ \Omega$, 电阻正常。



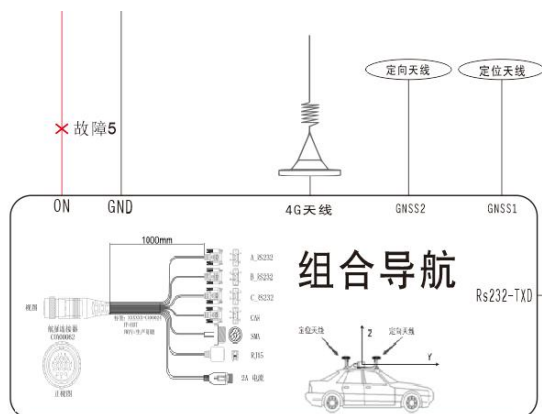
故障机理分析: 由于单目相机的电源断路, 导致使用上位软件无图像呈现, 功能使用异常。

实验结果: 单目相机电源线路断路

4.5 实验五

实验准备：能正常工作的万用表、专用诊断仪、故障设置设备。

实验对象：组合导航电源线断路故障。



实验目的：了解组合导航的工作原理，理解组合导航的线路走向。

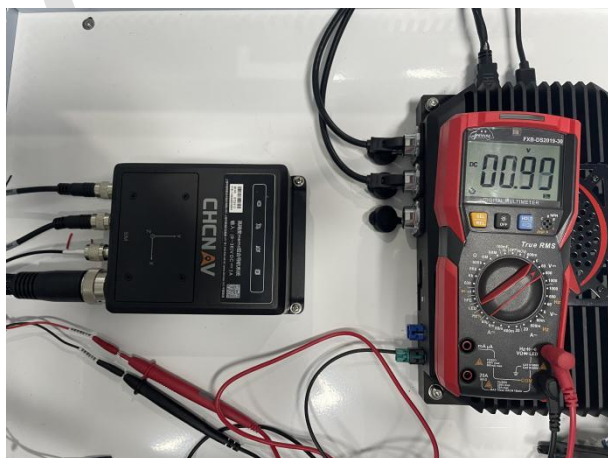
实验现象：使用上位软件无卫星信号，导航指示灯不亮，功能使用异常。

故障分析：功能异常，不能正常使用的可能原因可能在于：

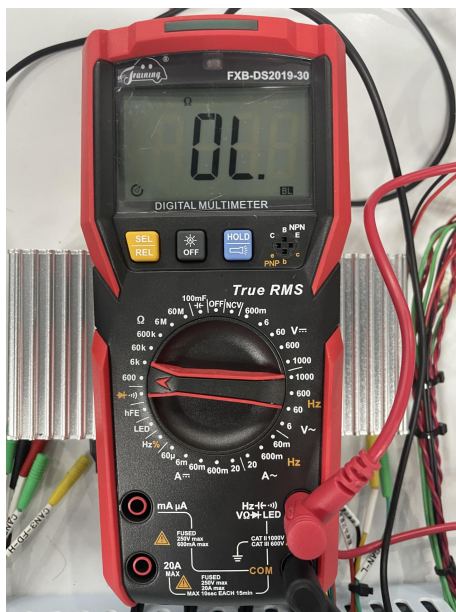
- (1) 电源以及相关线路
- (2) 232 传输线路断路、虚接
- (3) 组合导航以及本身元器件

排查过程

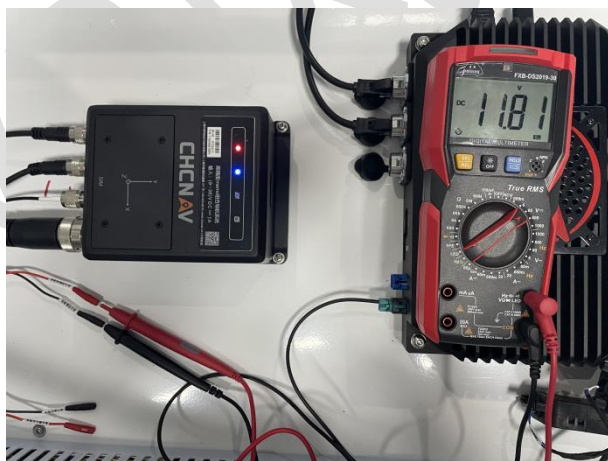
- (1) 用万用表电压档测组合导航电源线（红表笔）对组合导航地线线（黑表笔）实际测量电压为 0.99V，且组合导航电源指示灯不亮。正常传感器电压值应该为 12V 左右，判断组合导航电源可能存在故障。



- (2) 用万用表电压档测底盘给传感器供电的 12V 电源线（红表笔）对地线（黑表笔）实际测量电压为 12V，说明总电源线路正常。可能总电源到组合导航供电线路存在断路。
- (3) 断开电源，用万用表电阻档测底盘 12V 电源端到组合导航电源线端，实际测量阻值无穷大，判断这节线路已断路。



- (4) 故障恢复后再次测量电压值, 电压显示为 12V 左右正常。



- (5) 再次用电阻挡，线路电阻为 0.4Ω ，电阻正常。



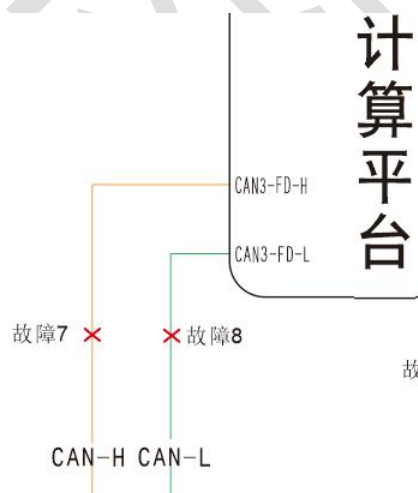
故障机理分析：由于组合导航的电源断路，导致使用上位软件无卫星信号，导航指示灯不亮，功能使用异常。

实验结果：组合导航电源线路断路

4.6 实验六

实验准备：能正常工作的万用表、专用诊断仪、故障设置设备。

实验对象：工控机 CAN3—H 线断路故障。



实验目的：了解线控底盘的工作原理，理解线控底盘的线路走向。

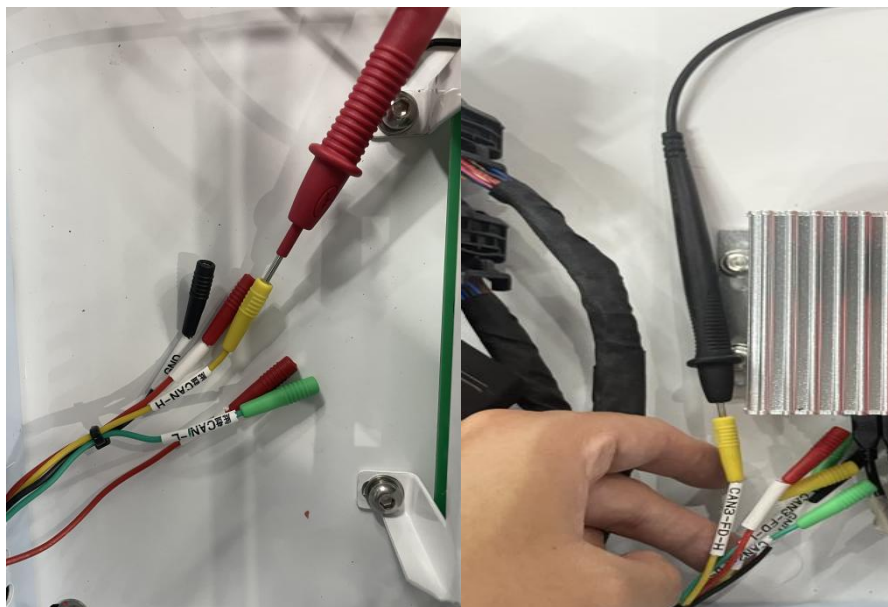
实验现象：无法通过软件实现线控控制，功能使用异常。

故障分析：功能异常，不能正常使用的可能原因可能在于：

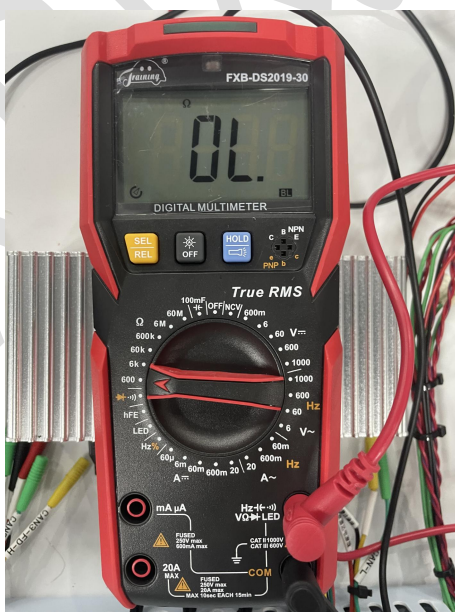
- (1) 电源以及相关线路
- (2) CAN 线断路、虚接、短路、反接
- (3) 线控底盘以及本身元器件

排故过程

- (1) 用万用电压档测底盘 CAN-H 和 CAN-L (红表笔) 对底盘 CAN 地线 (黑表笔) 实际测量, 高位线和地位线电压差存在异常, 判断底盘电源可能存在故障。



- (2) 断开电源用万用电压档测底盘 CAN-H (红表笔) 对计算单元 CAN3-H (黑表笔) 实际测量电阻为无穷大, 说明底盘 CAN-H 到计算单元 CAN3-H 线路之间存在断路。



- (3) 再次用电阻挡, 线路电阻为 $0.4\ \Omega$, 电阻正常。



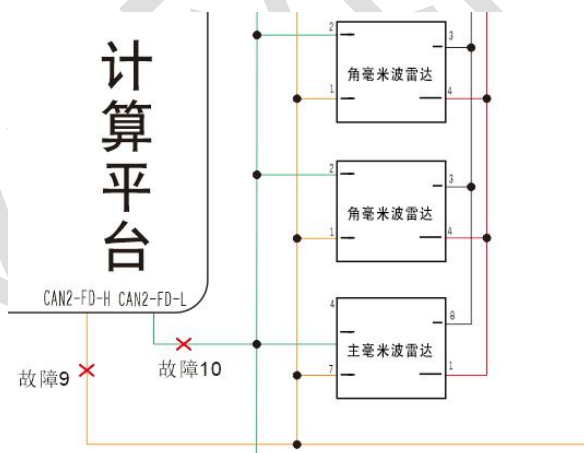
故障机理分析：由于底盘 CAN-H 断路，导致无法通过软件实现线控控制，功能使用异常。

实验结果：工控机 CAN3—H 线断路

4.7 实验七

实验准备：能正常工作的万用表、专用诊断仪、故障设置设备。

实验对象：工控机 CAN2—L 线断路故障。



实验目的：了解传感器的工作原理，理解传感器 CAN 线路的走向。

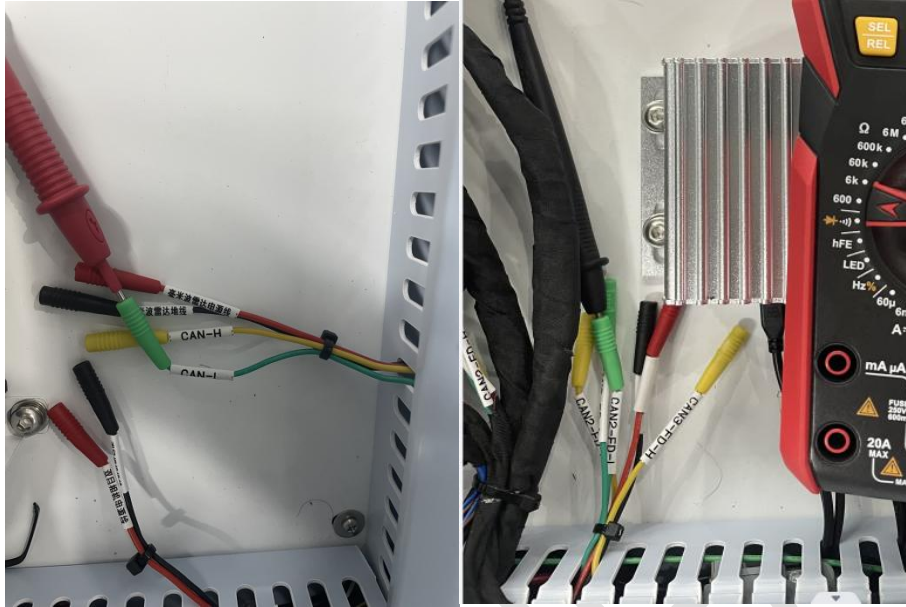
实验现象：通过软件检测实物无法成像，无数据显示，功能使用异常。

故障分析：功能异常，不能正常使用的可能原因可能在于：

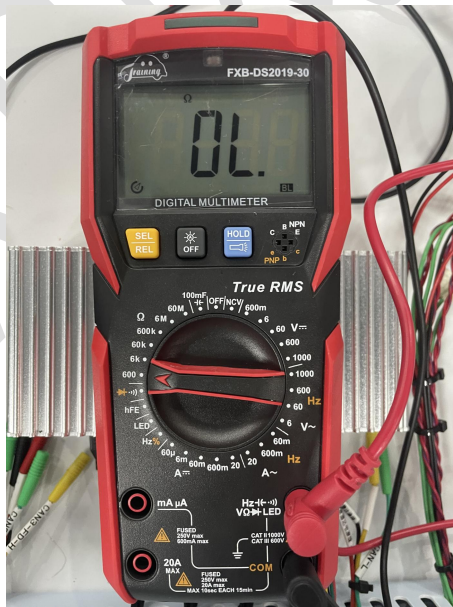
- (1) 电源以及相关线路
- (2) CAN 线断路、虚接、短路、反接
- (3) 计算单元以及本身元器件

排故过程

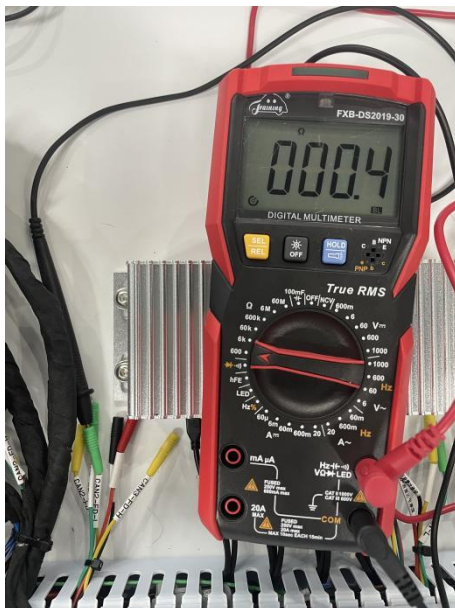
- (1) 用万用电压档测计算单元 CAN-H 和 CAN-L（红表笔）对计算单元 CAN 地线（黑表笔）实际测量，高位线和地位线电压差存在异常，判断计算单元电源可能存在故障。



- (2) 断开电源用万用电阻档测毫米波 CAN-L（红表笔）对计算单元 CAN2-L（黑表笔）实际测量电阻为无穷大，说明毫米波 CAN-L 到计算单元 CAN2-H 线路之间存在断路。



- (3) 再次用电阻挡，线路电阻为 $0.4\ \Omega$ ，电阻正常。



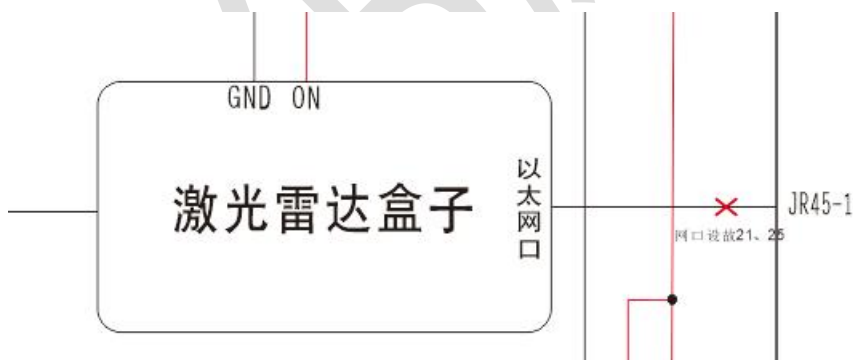
故障机理分析：由于传感器 CAN-H 断路，导致实物无法成像，无数据显示，功能使用异常。

实验结果：工控机 CAN2—L 线断路

4.8 实验八

实验准备：能正常工作的网线测试仪、专用诊断仪、故障设置设备。

实验对象：激光雷达以太网口 2 线断路故障。



实验目的：了解线激光雷达的工作原理，理解传感器如何通过网线传输信息。

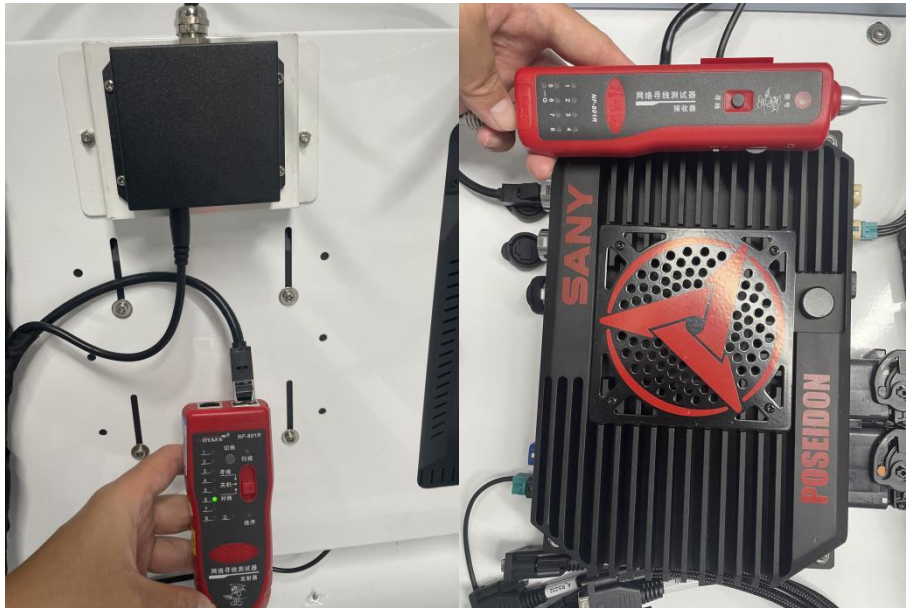
实验现象：通过软件检测实物无法成 3D 点云图像，功能使用异常。

故障分析：功能异常，不能正常使用的可能原因可能在于：

- (1) 电源以及相关线路
- (2) 网线断路、虚接
- (3) 激光雷达以及本身元器件

排查过程

(1) 把激光雷达控制盒上的网线插入网线测试仪的一头。



(2) 把计算单元上的网线插入网线测试仪另一头，按下对线开关。发现对线指示灯到 2 号灯的时候不亮，则判断网线口存在故障。

(3) 排除故障后，发现对线指示灯均正常闪烁。



故障机理分析：由于网线口 2 断路，导致实物无法成 3D 点云像，无数据显示，功能使用异常。

实验结果：激光雷达以太网口 2 线断路故障。

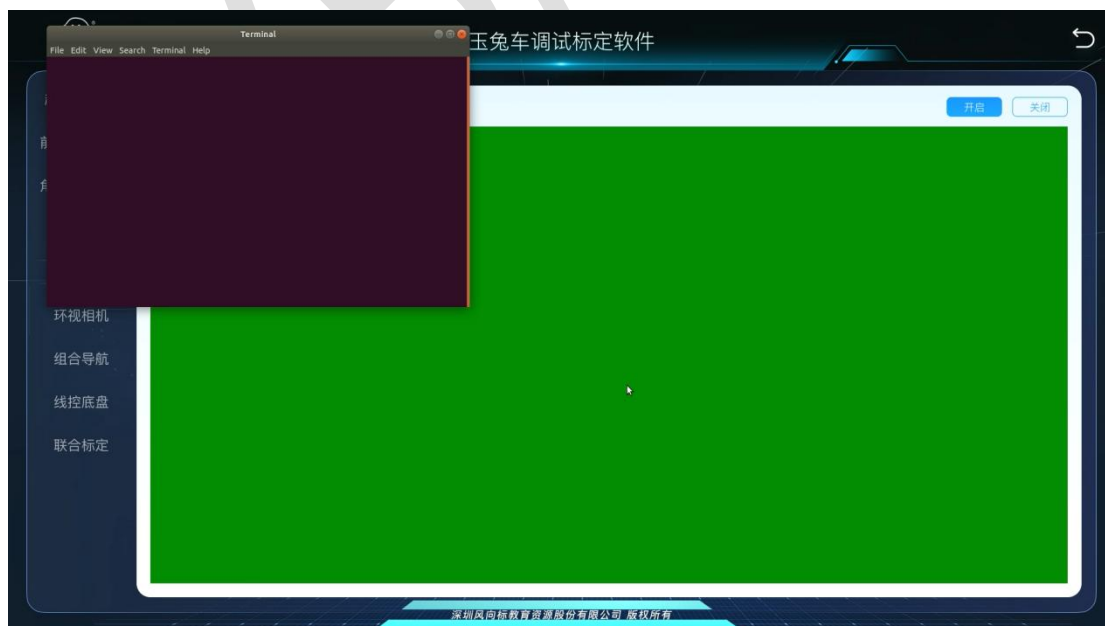
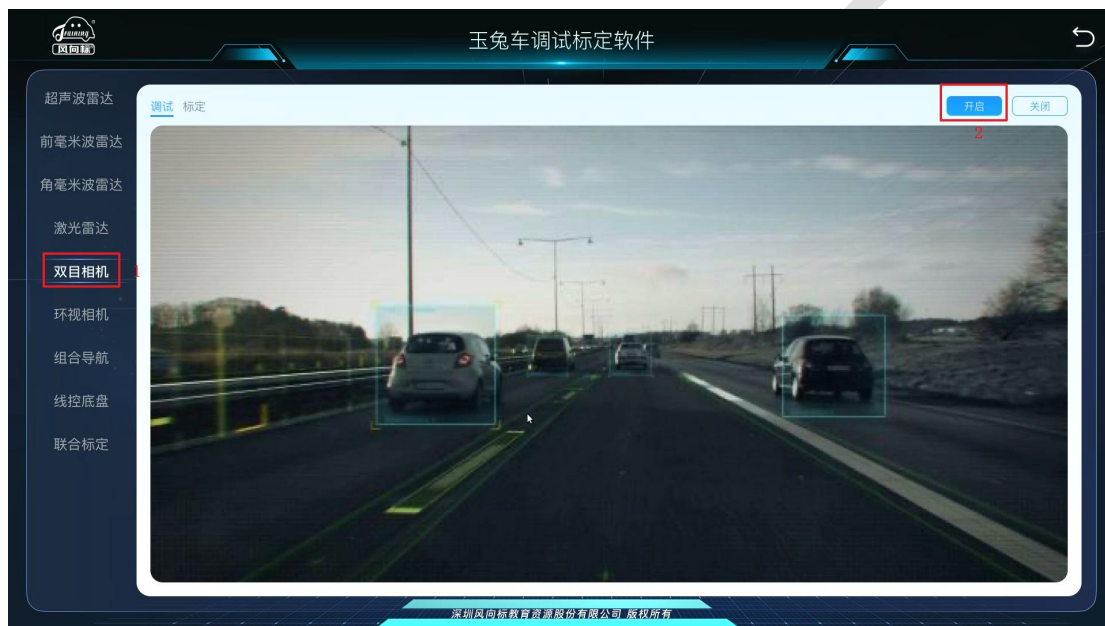
4.9 实验九

实验对象：单目相机 ROS 驱动的设备文件路径配置错误。

实验目的：学习传感器 ROS 驱动的启动文件配置。

实验现象

1) 打开“调试标定软件”，找到“双目相机”，点击“开启”按钮，发现没有相机画面，并且画面绿屏。



故障分析



没有相机画面，可能有以下几个原因

- 1) 没有启动相机节点
- 2) 启动文件配置错误
- 3) 单目相机线路问题
- 4) 单目相机本身部件问题

排查过程

- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rostopic list` 查询所有节点，发现存在相机节点/`v4l2_camera_node`

```
nvidia@oceanstar: ~  
File Edit View Search Terminal Help  
nvidia@oceanstar:~$ rostopic list  
/calibrate_camera_node  
/rosout  
/rviz_1747812166887559792  
/socket_can0  
/socket_can1  
/ui_node  
/v4l2_camera_node  
/verification_camera_node  
nvidia@oceanstar:~$
```

- 2) 输入命令 `rostopic info /v4l2_camera_node` 查询相机节点的发布话题，找到 `/center_cam/image_raw` 相机画面的话题名

```
nvidia@oceanstar:~$ rostopic info /v4l2_camera_node  
  
-----  
Node [/v4l2_camera_node]  
Publications:  
* /center_cam/image_raw [sensor_msgs/Image]  
* /center_cam/image_raw/compressed [sensor_msgs/CompressedImage]  
* /center_cam/image_raw/compressed/parameter_descriptions [dynamic_reconfigure/  
ConfigDescription]  
* /center_cam/image_raw/compressed/parameter_updates [dynamic_reconfigure/Conf  
ig]  
* /center_cam/image_raw/compressedDepth [sensor_msgs/CompressedImage]  
* /center_cam/image_raw/compressedDepth/parameter_descriptions [dynamic_reconfi  
gure/ConfigDescription]  
* /center_cam/image_raw/compressedDepth/parameter_updates [dynamic_reconfigure/  
Config]  
* /center_cam/image_raw/theora [theora_image_transport/Package]  
* /center_cam/image_raw/theora/parameter_descriptions [dynamic_reconfigure/Conf  
igDescription]  
* /center_cam/image_raw/theora/parameter_updates [dynamic_reconfigure/Config]
```

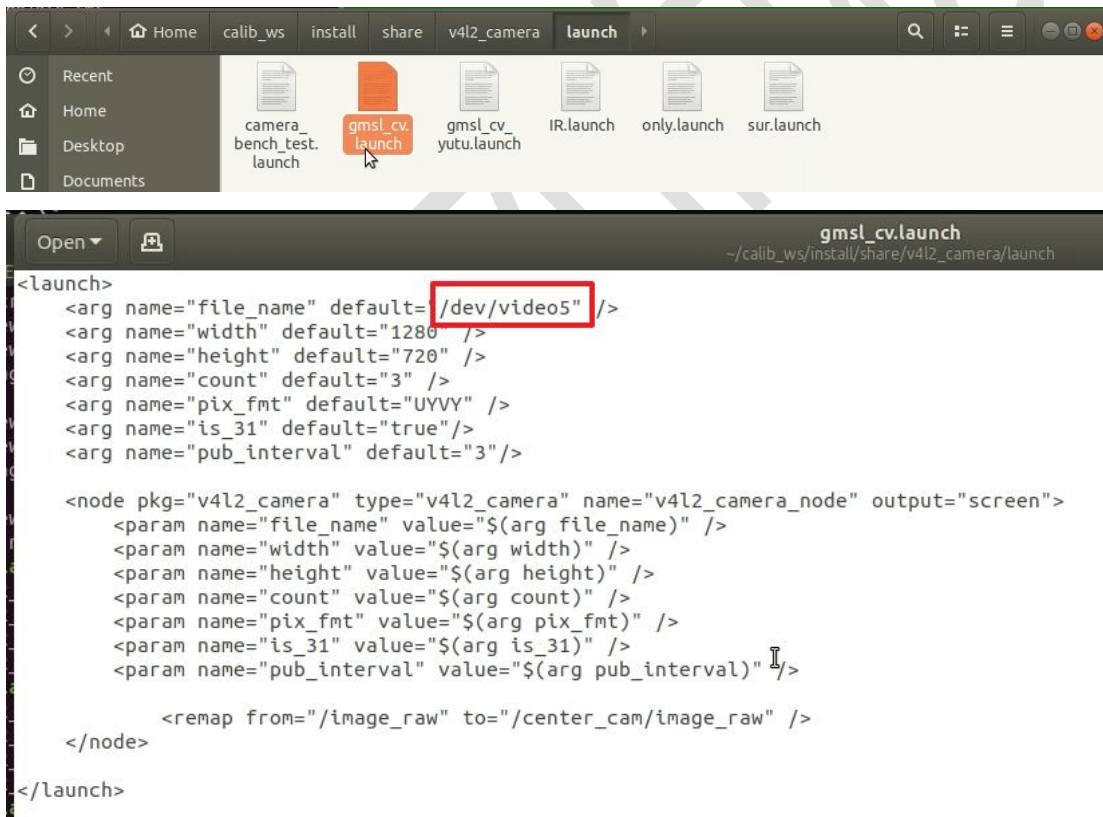
- 3) 输入命令 `rostopic hz /center_cam/image_raw` 查询该相机画面发布的帧率，发现几乎没有数据发布出来，确定没有相机画面的现象


```
nvidia@oceanstar:~$ rostopic hz /center_cam/image_raw
subscribed to [/center_cam/image_raw]
no new messages
no new messages
average rate: 0.392
min: 2.548s max: 2.548s std dev: 0.00000s window: 2
```

4) 输入命令 `rosparam get /v4l2_camera_node/file_name` 查询相机的设备文件路径配置，发现文件路径不正确，正确的单目相机为 `/dev/video4`

```
nvidia@oceanstar:~$ rosparam get /v4l2_camera_node/file_name
/dev/video5
```

5) 找到 `/home/nvidia/calib_ws/install/share/v4l2_camera/launch/gmsl_cv.launch` 文件，将 `video5` 改为 `video4` 后保存，重新启动调试标定软件后，发现相机正常



```
Open ▾  gmsl_cv.launch  
~/calib_ws/install/share/v4l2_camera/launch  
<launch>  
  <arg name="file_name" default="/dev/video4" />  
  <arg name="width" default="1280" />  
  <arg name="height" default="720" />  
  <arg name="count" default="3" />  
  <arg name="pix_fmt" default="UYVY" />  
  <arg name="is_31" default="true" />  
  <arg name="pub_interval" default="3" />  
  
  <node pkg="v4l2_camera" type="v4l2_camera" name="v4l2_camera_node" output="screen">  
    <param name="file_name" value="$(arg file_name)" />  
    <param name="width" value="$(arg width)" />  
    <param name="height" value="$(arg height)" />  
    <param name="count" value="$(arg count)" />  
    <param name="pix_fmt" value="$(arg pix_fmt)" />  
    <param name="is_31" value="$(arg is_31)" />  
    <param name="pub_interval" value="$(arg pub_interval)" />  
  
    <remap from="/image_raw" to="/center_cam/image_raw" />  
  </node>  
</launch>
```

故障机理分析：由于单目相机 ROS 驱动的设备文件路径配置错误，所以无法获取相机图像，最终导致没有相机画面显示。

实验结果：单目相机 ROS 驱动的设备文件路径配置错误。

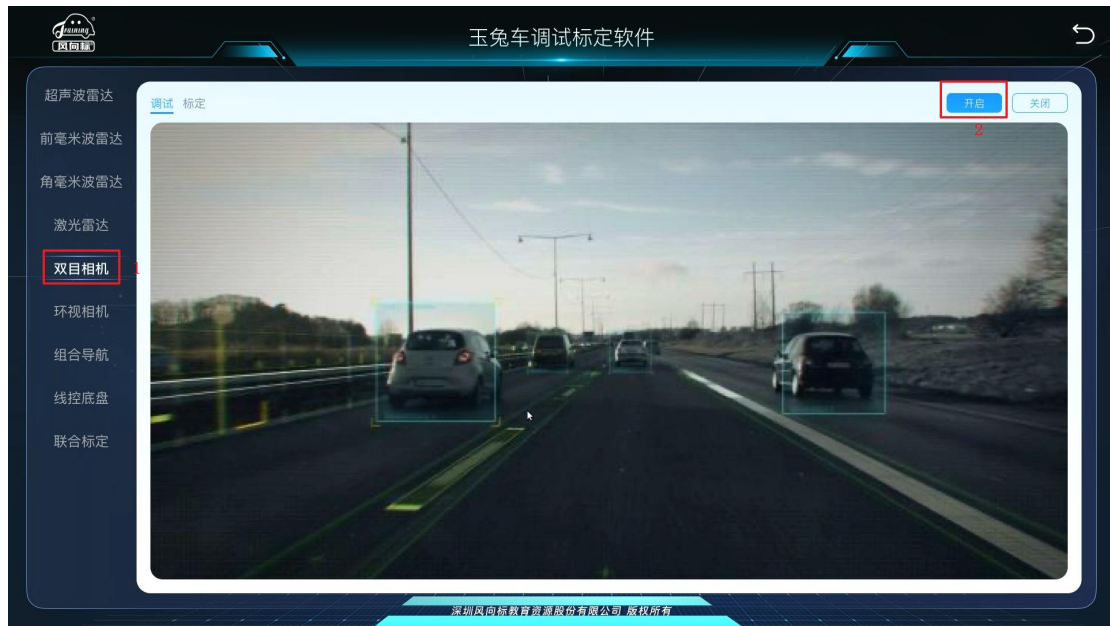
4.10 实验十

实验对象：单目相机 ROS 驱动的设备文件路径配置错误。

实验目的：学习传感器 ROS 驱动的启动文件配置。

实验现象

1) 打开调试标定软件，找到“双目相机”，点击“开启”按钮，发现相机画面正常，但是相机不是单目相机，而是环视相机。



故障分析

相机画面正常，但是相机不是单目相机的画面

- 1) 启动文件配置错误
- 2) 单目相机线路问题

排查过程

- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入命令

`rosparam get /v4l2_camera_node/file_name` 查询单目相机的设备文件路径配置，发现文件路径不正确，正确的单目相机为 `/dev/video4`

```
nvidia@oceanstar:~$ rosparam get /v4l2_camera_node/file_name
/dev/video0
```

2) 找到/home/nvidia/calib_ws/install/share/v4l2_camera/launch/gmsl_cv.launch 文件，将 video0 改为 video4 后保存，重新启动调试标定软件后，发现单目相机正常



```
<launch>
  <arg name="file_name" default="/dev/video0" />
  <arg name="width" default="1280" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="UYVY" />
  <arg name="is_31" default="true"/>
  <arg name="pub_interval" default="3"/>

  <node pkg="v4l2_camera" type="v4l2_camera" name="v4l2_camera_node" output="screen">
    <param name="file_name" value="$(arg file_name)" />
    <param name="width" value="$(arg width)" />
    <param name="height" value="$(arg height)" />
    <param name="count" value="$(arg count)" />
    <param name="pix_fmt" value="$(arg pix_fmt)" />
    <param name="is_31" value="$(arg is_31)" />
    <param name="pub_interval" value="$(arg pub_interval)" />

    <remap from="/image_raw" to="/center_cam/image_raw" />
  </node>
</launch>
```

```
<launch>
  <arg name="file_name" default="/dev/video4" />
  <arg name="width" default="1280" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="UYVY" />
  <arg name="is_31" default="true"/>
  <arg name="pub_interval" default="3"/>

  <node pkg="v4l2_camera" type="v4l2_camera" name="v4l2_camera_node" output="screen">
    <param name="file_name" value="$(arg file_name)" />
    <param name="width" value="$(arg width)" />
    <param name="height" value="$(arg height)" />
    <param name="count" value="$(arg count)" />
    <param name="pix_fmt" value="$(arg pix_fmt)" />
    <param name="is_31" value="$(arg is_31)" />
    <param name="pub_interval" value="$(arg pub_interval)" />

    <remap from="/image_raw" to="/center_cam/image_raw" />
  </node>
</launch>
```

故障机理分析：由于单目相机的 ROS 驱动的设备文件路径配置错误，所以获取相机画面错误。



实验结果：单目相机 ROS 驱动的设备文件路径配置错误。

4.11 实验十一

实验对象：单目相机 ROS 驱动的配置错误，导致无法启动 ROS 节点。

实验目的：学习传感器 ROS 驱动的启动文件配置。

实验现象

1) 打开调试标定软件，找到“双目相机”，点击“开启”按钮，发现没有相机画面。



故障分析

没有相机画面，可能有以下几个原因

- 1) 没有启动相机节点
- 2) 启动文件配置错误
- 3) 单目相机线路问题
- 4) 单目相机本身部件问题

排故过程

1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rostop list` 查询所有节点，发现不存在相机节点/`v4l2_camera_node`

```

nvidia@oceanstar: ~
File Edit View Search Terminal Help
nvidia@oceanstar:~$ roslaunch list
/calibrate_camera_node
/rosout
/rviz_1747812922583198608
/socket_can0
/socket_can1
/ui_node
/verification_camera_node
  
```

2) 找到/home/nvidia/calib_ws/install/share/v4l2_camera/launch/gmsl_cv.launch 文件，将!--和--删除后保存，重新启动调试标定软件后，发现单目相机正常



```

<launch>
  <arg name="file_name" default="/dev/video4" />
  <arg name="width" default="1280" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="UYVY" />
  <arg name="is_31" default="true"/>
  <arg name="pub_interval" default="3"/>

  <!-- node pkg="v4l2_camera" type="v4l2_camera" name="v4l2_camera_node" output="screen" -->
  <node pkg="v4l2_camera" type="v4l2_camera" name="v4l2_camera_node" output="screen" />
  <param name="file_name" value="$(arg file_name)" />
  <param name="width" value="$(arg width)" />
  <param name="height" value="$(arg height)" />
  <param name="count" value="$(arg count)" />
  <param name="pix_fmt" value="$(arg pix_fmt)" />
  <param name="is_31" value="$(arg is_31)" />
  <param name="pub_interval" value="$(arg pub_interval)" />

  <remap from="/image_raw" to="/center_cam/image_raw" />
</node>
</launch>
  
```

```

*gmsl_cv.launch
~/calib_ws/install/share/v4l2_camera/launch

<launch>
  <arg name="file_name" default="/dev/video4" />
  <arg name="width" default="1280" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="UYVY" />
  <arg name="is_31" default="true"/>
  <arg name="pub_interval" default="3"/>

  <node pkg="v4l2_camera" type="v4l2_camera" name="v4l2_camera_node" output="screen">
    <param name="file_name" value="$(arg file_name)" />
    <param name="width" value="$(arg width)" />
    <param name="height" value="$(arg height)" />
    <param name="count" value="$(arg count)" />
    <param name="pix_fmt" value="$(arg pix_fmt)" />
    <param name="is_31" value="$(arg is_31)" />
    <param name="pub_interval" value="$(arg pub_interval)" />

    <remap from="/image_raw" to="/center_cam/image_raw" />
  </node>
</launch>
  
```



故障机理分析：由于单目相机的 ROS 驱动配置错误，无法启动 ROS 节点，导致无法获取相机画面，最终无法显示相机画面。

实验结果：单目相机 ROS 驱动的配置错误，导致无法启动 ROS 节点。

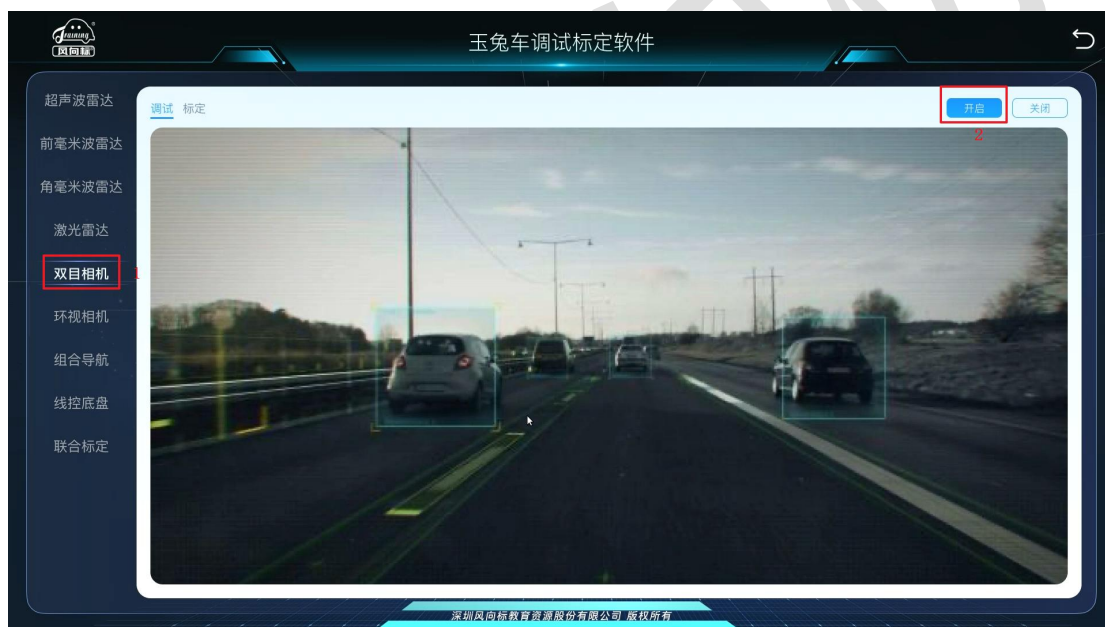
4.12 实验十二

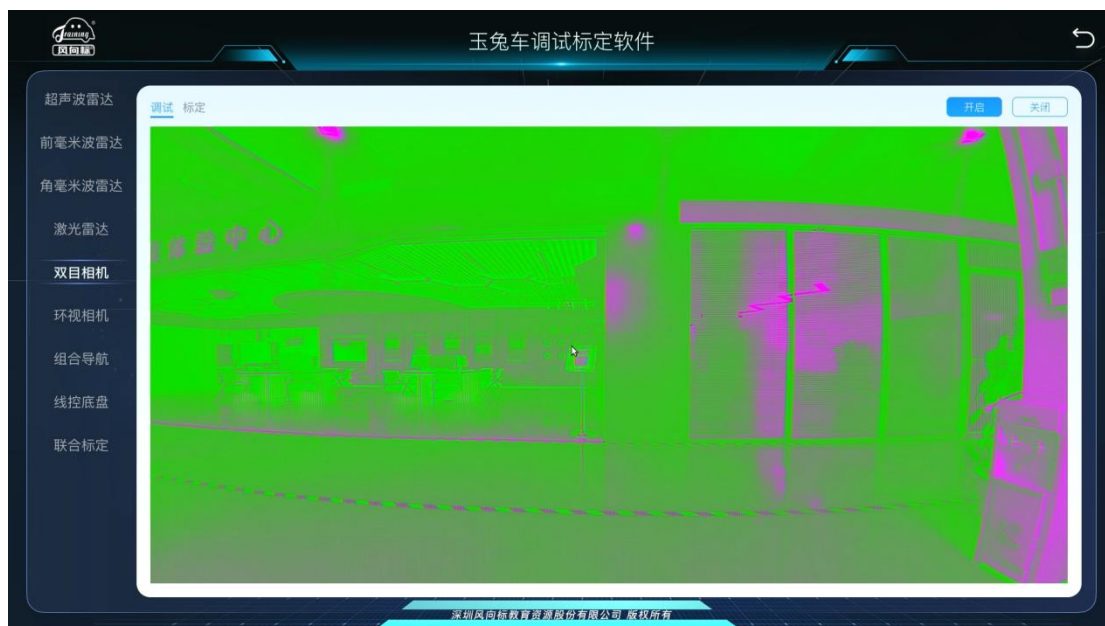
实验对象：单目相机 ROS 驱动的颜色编码配置错误

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象

1) 打开调试标定软件，找到“双目相机”，点击“开启”按钮，发现相机画面颜色异常。





故障分析

没有相机画面，可能有以下几个原因

- 1) 启动文件配置错误
- 2) 单目相机线路问题
- 3) 单目相机本身部件问题

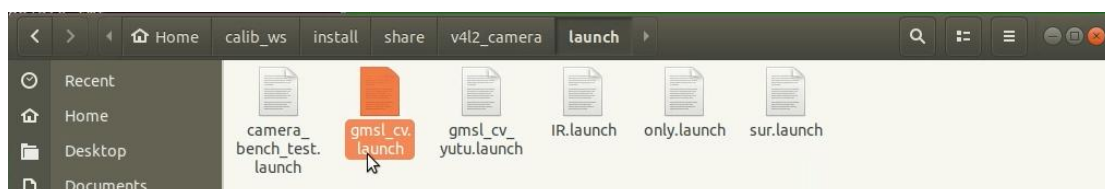
排查过程

- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入命令

`rosparam get /v4l2_camera_node/pix_fmt` 查询单目相机的颜色编码，发现颜色编码不正确，正确为 `UYVY`

```
nvidia@oceanstar:~$ rosparam get /v4l2_camera_node/pix_fmt
YUYV
```

- 2) 找到 `/home/nvidia/calib_ws/install/share/v4l2_camera/launch/gmsl_cv.launch` 文件，将 `YUYV` 改为 `UYVY`，重新启动调试标定软件后，发现单目相机正常




```

gmsl_cv.launch
~/calib_ws/install/share/v4l2_camera/launch

<launch>
  <arg name="file_name" default="/dev/video4" />
  <arg name="width" default="1280" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="YUYV" />
  <arg name="is_31" default="true" />
  <arg name="pub_interval" default="3" />

  <node pkg="v4l2_camera" type="v4l2_camera" name="v4l2_camera_node" output="screen">
    <param name="file_name" value="$(arg file_name)" />
    <param name="width" value="$(arg width)" />
    <param name="height" value="$(arg height)" />
    <param name="count" value="$(arg count)" />
    <param name="pix_fmt" value="$(arg pix_fmt)" />
    <param name="is_31" value="$(arg is_31)" />
    <param name="pub_interval" value="$(arg pub_interval)" />

    <remap from="/image_raw" to="/center_cam/image_raw" />
  </node>
</launch>

gmsl_cv.launch
~/calib_ws/install/share/v4l2_camera/launch

<launch>
  <arg name="file_name" default="/dev/video4" />
  <arg name="width" default="1280" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="UYVY" />
  <arg name="is_31" default="true" />
  <arg name="pub_interval" default="3" />

  <node pkg="v4l2_camera" type="v4l2_camera" name="v4l2_camera_node" output="screen">
    <param name="file_name" value="$(arg file_name)" />
    <param name="width" value="$(arg width)" />
    <param name="height" value="$(arg height)" />
    <param name="count" value="$(arg count)" />
    <param name="pix_fmt" value="$(arg pix_fmt)" />
    <param name="is_31" value="$(arg is_31)" />
    <param name="pub_interval" value="$(arg pub_interval)" />

    <remap from="/image_raw" to="/center_cam/image_raw" />
  </node>
</launch>

```

故障机理分析：由于单目相机的 ROS 驱动颜色编码配置错误，导致显示画面颜色异常。

实验结果：单目相机 ROS 驱动的颜色编码配置错误

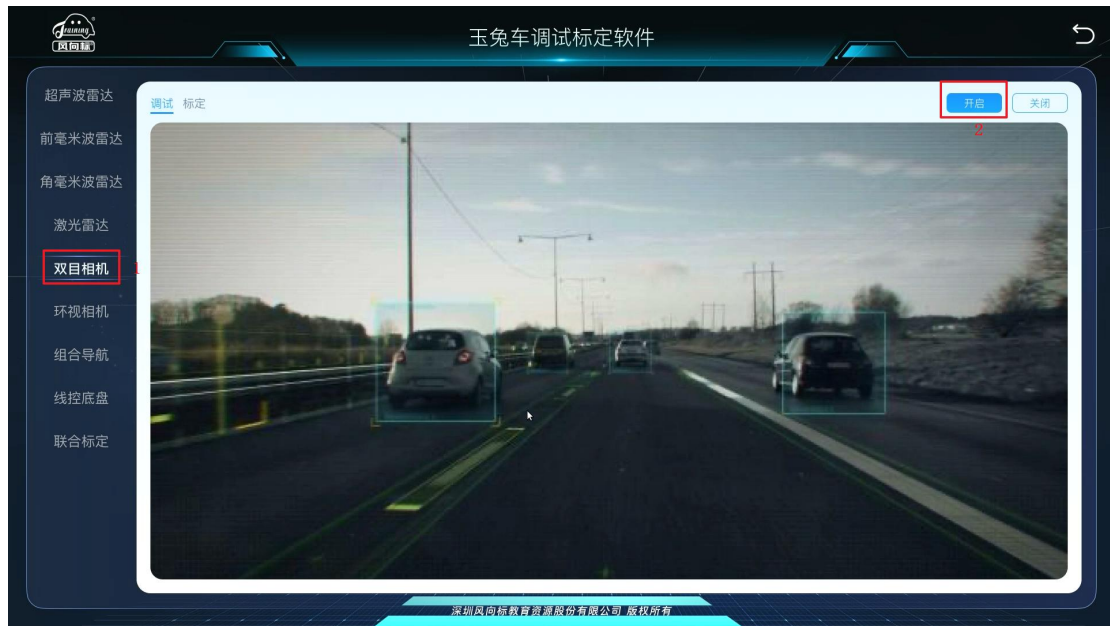
4.13 实验十三

实验对象：单目相机 ROS 驱动的分辨率配置错误

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象

1) 打开调试标定软件，找到“双目相机”，点击“开启”按钮，发现没有相机画面。



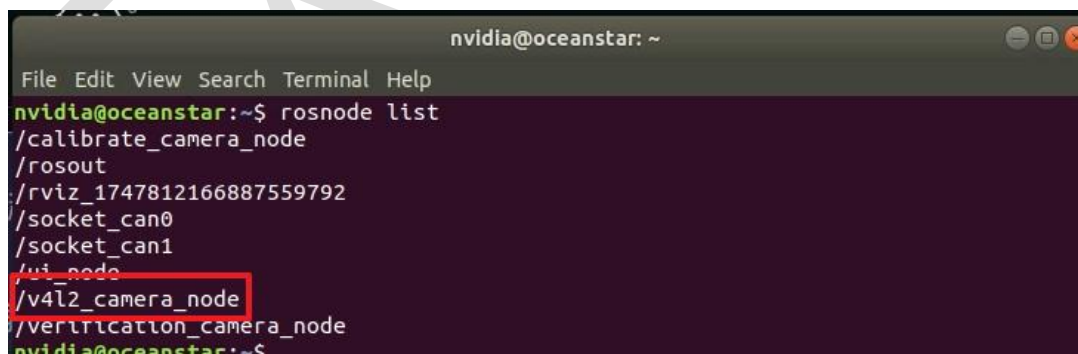
故障分析

没有相机画面，可能有以下几个原因

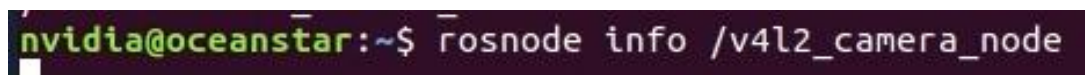
- 1) 没有启动相机节点
- 2) 启动文件配置错误
- 3) 单目相机线路问题
- 4) 单目相机本身部件问题

排故过程

1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rostopic list` 查询所有节点，发现存在相机节点 `/v4l2_camera_node`



2) 输入命令 `rostopic info /v4l2_camera_node` 查询相机节点的发布话题，找到 `/center_cam/image_raw` 相机画面的话题名



```
Node [/v4l2_camera_node]
Publications:
* /center_cam/image_raw [sensor_msgs/Image]
* /center_cam/image_raw/compressed [sensor_msgs/CompressedImage]
* /center_cam/image_raw/compressed/parameter_descriptions [dynamic_reconfigure/ConfigDescription]
* /center_cam/image_raw/compressed/parameter_updates [dynamic_reconfigure/Config]
* /center_cam/image_raw/compressedDepth [sensor_msgs/CompressedImage]
* /center_cam/image_raw/compressedDepth/parameter_descriptions [dynamic_reconfigure/ConfigDescription]
* /center_cam/image_raw/compressedDepth/parameter_updates [dynamic_reconfigure/Config]
* /center_cam/image_raw/theora [theora_image_transport/Packet]
* /center_cam/image_raw/theora/parameter_descriptions [dynamic_reconfigure/ConfigDescription]
* /center_cam/image_raw/theora/parameter_updates [dynamic_reconfigure/Config]
```

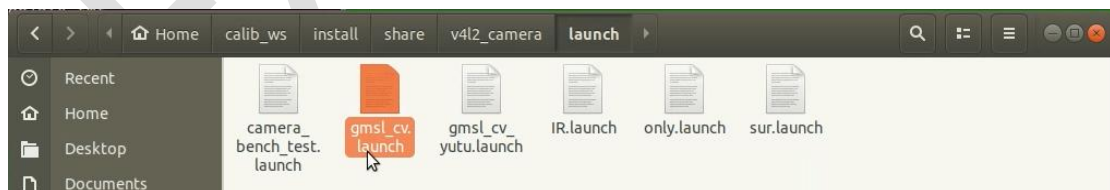
3) 输入命令 `rostopic hz /center_cam/image_raw` 查询该相机画面发布的帧率，发现几乎没有数据发布出来，确定没有相机画面的现象

```
nvidia@oceanstar:~$ rostopic hz /center_cam/image_raw
subscribed to [/center_cam/image_raw]
no new messages
no new messages
average rate: 0.392
min: 2.548s max: 2.548s std dev: 0.00000s window: 2
```

4) 输入命令 `rosparam get /v4l2_camera_node/width` 查询相机的分辨率宽度，发现分辨率宽度不正确，正确分辨率为 1280x720

```
nvidia@oceanstar:~$ rosparam get /v4l2_camera_node/width
128
```

5) 找到 `/home/nvidia/calib_ws/install/share/v4l2_camera/launch/gmsl_cv.launch` 文件，将 128 改为 1280 后保存，重新启动调试标定软件后，发现相机正常



```

Open  [icon] gmsl_cv.launch
~/calib_ws/install/share/v4l2_camera/launch

<launch>
  <arg name="file_name" default="/dev/video4" />
  <arg name="width" default="128" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="UYVY" />
  <arg name="is_31" default="true"/>
  <arg name="pub_interval" default="3"/>

  <node pkg="v4l2_camera" type="v4l2_camera" name="v4l2_camera_node" output="screen">
    <param name="file_name" value="$(arg file_name)" />
    <param name="width" value="$(arg width)" />
    <param name="height" value="$(arg height)" />
    <param name="count" value="$(arg count)" />
    <param name="pix_fmt" value="$(arg pix_fmt)" />
    <param name="is_31" value="$(arg is_31)" />
    <param name="pub_interval" value="$(arg pub_interval)" />

    <remap from="/image_raw" to="/center_cam/image_raw" />
  </node>
</launch>

```

```

Open  [icon] gmsl_cv.launch
~/calib_ws/install/share/v4l2_camera/launch

<launch>
  <arg name="file_name" default="/dev/video4" />
  <arg name="width" default="1280" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="UYVY" />
  <arg name="is_31" default="true"/>
  <arg name="pub_interval" default="3"/>

  <node pkg="v4l2_camera" type="v4l2_camera" name="v4l2_camera_node" output="screen">
    <param name="file_name" value="$(arg file_name)" />
    <param name="width" value="$(arg width)" />
    <param name="height" value="$(arg height)" />
    <param name="count" value="$(arg count)" />
    <param name="pix_fmt" value="$(arg pix_fmt)" />
    <param name="is_31" value="$(arg is_31)" />
    <param name="pub_interval" value="$(arg pub_interval)" />

    <remap from="/image_raw" to="/center_cam/image_raw" />
  </node>
</launch>

```

故障机理分析：由于单目相机的 ROS 驱动分辨率宽度配置错误，导致无法获取相机画面，最终没有相机画面显示。

实验结果：单目相机 ROS 驱动分辨率配置错误

4.14 实验十四

实验对象：线控底盘 ROS 驱动接收报文话题配置错误

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象

•



- 1) 举升车辆，将遥控器设置为自动驾驶模式，确认所有应急开关没有被按下。
- 2) 打开调试标定软件，找到“线控底盘”下的“线控转向”。



- 3) 释放制动力，发现车辆未执行目标转向角度。
- 4) 勾选制动力，找到“线控底盘”下的“线控驱动”。



- 5) 释放制动力，发现车辆未执行目标速度。

故障分析

线控底盘未执行目标控制，可能有以下几个原因

- 1) 没有启动线控底盘节点



- 2) 启动文件配置错误
- 3) 线控底盘线路问题
- 4) 线控底盘相关控制部件本身问题

排查过程

- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rostopic list`，发现存在 `/yhs_send_node` 线控发送节点

```
nvidia@oceanstar: ~  
File Edit View Search Terminal Help  
nvidia@oceanstar:~$ rostopic list  
/decision_multi_signals_node  
/rosout  
/rviz_1747814436569678352  
/socket_can0  
/socket_can1  
/ui_node  
/yhs_receive_node  
/yhs_send_node
```

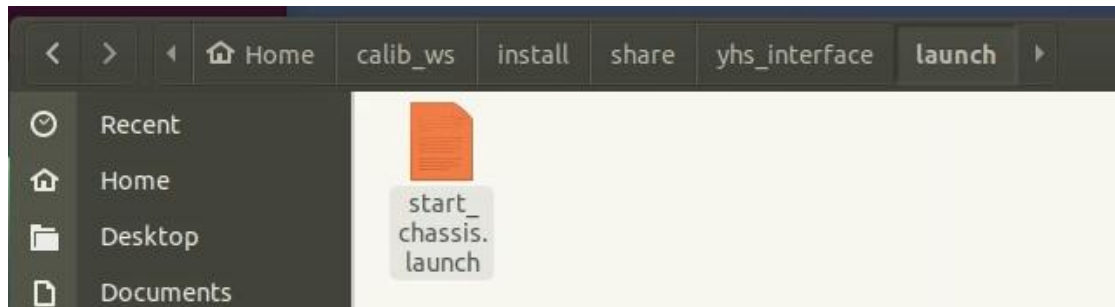
- 2) 输入 `rostopic info /yhs_send_node`，查询该节点信息，发现发布 CAN 的话题名错误，正确为 `/can0_sender`

```
nvidia@oceanstar:~$ rostopic info /yhs_send_node  
-----  
Node [/yhs_send_node]  
Publications:  
* /can1_sender [can_msgs/Frame]  
* /rosout [roscpp_msgs/Log]  
  
Subscriptions:  
* /cmd2vehicle [autoware_msgs/VehicleCmd]  
* /message_control [can_msgs/Frame]  
* /yhs_light_control [unknown type]  
* /yhs_speed_control [std_msgs/Float32]  
* /yhs_steer_calibration [std_msgs/Bool]  
* /yhs_steer_control [std_msgs/Float32]  
  
Services:  
* /yhs_send_node/get_loggers  
* /yhs_send_node/set_logger_level
```



3) 找到

/home/nvidia/calib_ws/install/share/yhs_interface/launch/start_chassis.launch 文件
将/can1_sender 改为/can0_sender，保存后，重启“调试标定软件”发现线控底盘正常。



```
start_chassis.launch
~/calib_ws/install/share/yhs_interface/launch
Save

<launch>

  <node pkg="yhs_interface" type="yhs_send_node" name="yhs_send_node" output="screen">
    <remap from="/can0_sender" to="/can1_sender" />
    <remap from="/cmd2vehicle" to="/cmd2vehicle" />
    <remap from="/yhs_light_control" to="/yhs_light_control" />
    <remap from="/yhs_speed_control" to="/yhs_speed_control" />
    <remap from="/yhs_steer_control" to="/yhs_steer_control" />
  </node>

  <node pkg="yhs_interface" type="yhs_receive_node" name="yhs_receive_node" output="screen">
    <remap from="/can_info" to="/can_info" />
    <remap from="/vehicle_info" to="/vehicle_info" />
    <remap from="/can0_receiver" to="/can0_receiver" />
  </node>

  <!-- 是否开启碰撞检测 -->
  <arg name="start_collision_detection" default="false"/>
  <group if="$(arg start_collision_detection)">
    <include file="$(find collision_detection)/launch/start_detection.launch" />
  </group>

  <!-- 速度限制km/h -->
  <arg name="limit_speed" default="3.0" />
  <arg name="start_decision_multi_signals" default="true"/>
  <group if="$(arg start_decision_multi_signals)">
    <node pkg="decision_multi_signals" type="decision_multi_signals_node"
name="decision_multi_signals_node" output="screen">
      <param name="limit_speed" value="$(arg limit_speed)" type="double" />
    </node>
  </group>
</launch>
```

```

start_chassis.launch
~/calib_ws/install/share/yhs_interface/launch

<launch>

  <node pkg="yhs_interface" type="yhs_send_node" name="yhs_send_node" output="screen">
    <remap from="/can0_sender" to="/can0_sender" />
    <remap from="/cmd2vehicle" to="/cmd2vehicle" />
    <remap from="/yhs_light_control" to="/yhs_light_control" />
    <remap from="/yhs_speed_control" to="/yhs_speed_control" />
    <remap from="/yhs_steer_control" to="/yhs_steer_control" />
  </node>

  <node pkg="yhs_interface" type="yhs_receive_node" name="yhs_receive_node" output="screen">
    <remap from="/can_info" to="/can_info" />
    <remap from="/vehicle_info" to="/vehicle_info" />
    <remap from="/can0_receiver" to="/can0_receiver" />
  </node>

  <!-- 是否开启碰撞检测 -->
  <arg name="start_collision_detection" default="false"/>
  <group if="$(arg start_collision_detection)">
    <include file="$(find collision_detection)/launch/start_detection.launch" />
  </group>

  <!-- 速度限制km/h -->
  <arg name="limit_speed" default="3.0" />
  <arg name="start_decision_multi_signals" default="true"/>
  <group if="$(arg start_decision_multi_signals)">
    <node pkg="decision_multi_signals" type="decision_multi_signals_node"
name="decision_multi_signals_node" output="screen">
      <param name="limit_speed" value="$(arg limit_speed)" type="double" />
    </node>
  </group>

</launch>

```

故障机理分析：由于线控底盘 ROS 驱动发送话题名配置错误，导致无法将控制命令发送到底盘中，最终出现线控底盘未响应控制的现象。

实验结果：线控底盘 ROS 驱动接收报文话题配置错误

4.15 实验十五

实验对象：线控底盘 ROS 驱动配置错误，导致无法启动 ROS 节点。

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象

- 1) 举升车辆，将遥控器设置为自动驾驶模式，确认所有应急开关没有被按下。
- 2) 打开调试标定软件，找到“线控底盘”下的“线控转向”。



3) 释放制动力，发现车辆未执行目标转向角度。

4) 勾选制动力，找到“线控底盘”下的“线控驱动”。



5) 释放制动力，发现车辆未执行目标速度。

故障分析

线控底盘未执行目标控制，可能有以下几个原因

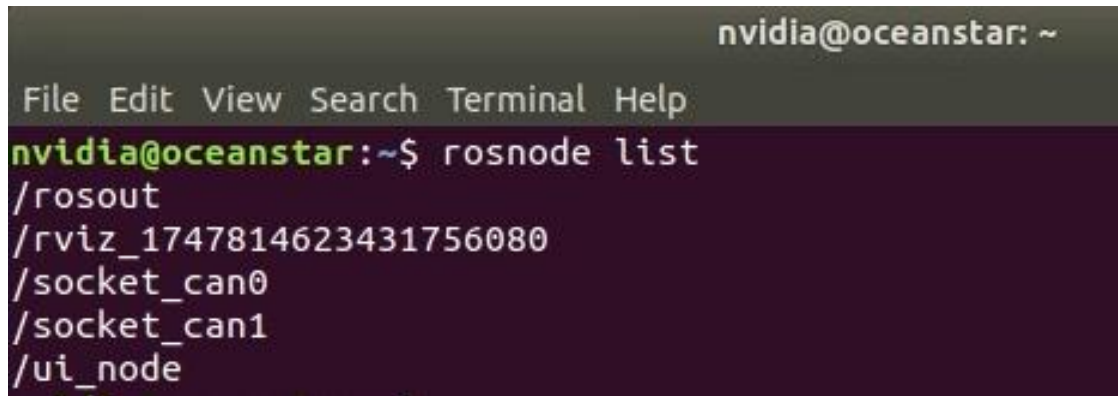
- 1) 没有启动线控底盘节点
- 2) 启动文件配置错误
- 3) 线控底盘线路问题



4) 线控底盘相关控制部件本身问题

排查过程

1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rostop list`，发现不存在 `/yhs_send_node` 线控发送节点

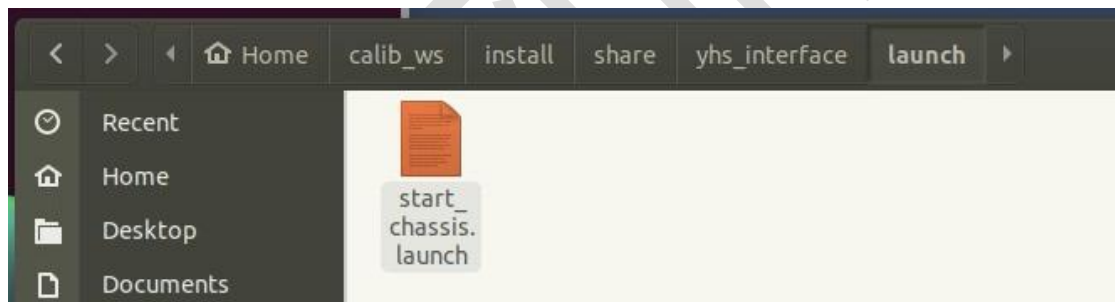


```
nvidia@oceanstar: ~  
File Edit View Search Terminal Help  
nvidia@oceanstar:~$ rostop list  
/rostop  
/rviz_1747814623431756080  
/socket_can0  
/socket_can1  
/ui_node
```

2) 找到

`/home/nvidia/calib_ws/install/share/yhs_interface/launch/start_chassis.launch`

将`!--`和`--`删除，保存后重启“调试标定软件”发现线控底盘正常。





```
Open start_chassis.launch Save
~/calib_ws/install/share/yhs_interface/launch

<launch>
<!-- node pkg="yhs_interface" type="yhs_send_node" name="yhs_send_node" output="screen" -->
  <remap from="/can0_sender" to="/can0_sender" />
  <remap from="/cmd2vehicle" to="/cmd2vehicle" />
  <remap from="/yhs_light_control" to="/yhs_light_control" />
  <remap from="/yhs_speed_control" to="/yhs_speed_control" />
  <remap from="/yhs_steer_control" to="/yhs_steer_control" />
</node>

<node pkg="yhs_interface" type="yhs_receive_node" name="yhs_receive_node" output="screen">
  <remap from="/can_info" to="/can_info" />
  <remap from="/vehicle_info" to="/vehicle_info" />
  <remap from="/can0_receiver" to="/can0_receiver" />
</node>

<!-- 是否开启碰撞检测 -->
<arg name="start_collision_detection" default="false"/>
<group if="$(arg start_collision_detection)">
  <include file="$(find collision_detection)/launch/start_detection.launch" />
</group>

<!-- 速度限制km/h -->
<arg name="limit_speed" default="3.0" />
<arg name="start_decision_multi_signals" default="true"/>
<group if="$(arg start_decision_multi_signals)">
  <node pkg="decision_multi_signals" type="decision_multi_signals_node"
name="decision_multi_signals_node" output="screen">
    <param name="limit_speed" value="$(arg limit_speed)" type="double" />
  </node>
</group>
</launch>
```

```
Open *start_chassis.launch Save
~/calib_ws/install/share/yhs_interface/launch

<launch>
<node pkg="yhs_interface" type="yhs_send_node" name="yhs_send_node" output="screen">
  <remap from="/can0_sender" to="/can0_sender" />
  <remap from="/cmd2vehicle" to="/cmd2vehicle" />
  <remap from="/yhs_light_control" to="/yhs_light_control" />
  <remap from="/yhs_speed_control" to="/yhs_speed_control" />
  <remap from="/yhs_steer_control" to="/yhs_steer_control" />
</node>

<node pkg="yhs_interface" type="yhs_receive_node" name="yhs_receive_node" output="screen">
  <remap from="/can_info" to="/can_info" />
  <remap from="/vehicle_info" to="/vehicle_info" />
  <remap from="/can0_receiver" to="/can0_receiver" />
</node>

<!-- 是否开启碰撞检测 -->
<arg name="start_collision_detection" default="false"/>
<group if="$(arg start_collision_detection)">
  <include file="$(find collision_detection)/launch/start_detection.launch" />
</group>

<!-- 速度限制km/h -->
<arg name="limit_speed" default="3.0" />
<arg name="start_decision_multi_signals" default="true"/>
<group if="$(arg start_decision_multi_signals)">
  <node pkg="decision_multi_signals" type="decision_multi_signals_node"
name="decision_multi_signals_node" output="screen">
    <param name="limit_speed" value="$(arg limit_speed)" type="double" />
  </node>
</group>
</launch>
```

故障机理分析：由于线控底盘 ROS 驱动配置错误，导致未能启动线控底盘节点，最终造成线控底盘无法响应控制命令。

实验结果：线控底盘 ROS 驱动配置错误，导致无法启动 ROS 节点。

4.16 实验十六

实验对象：环视相机 ROS 驱动配置错误，导致无法启动 ROS 节点。

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象

1) 打开“调试标定软件”，找到“环视相机”点击开启，发现无法显示 4 个环视相机的画面



故障分析

环视相机无法显示画面，可能有以下几个原因

- 1) 没有启动环视相机节点
- 2) 启动文件配置错误
- 3) 环视相机线路问题
- 4) 环视相机本身问题

排故过程

1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rostopic list` 发现没有四个环视相机节点


```
nvidia@oc

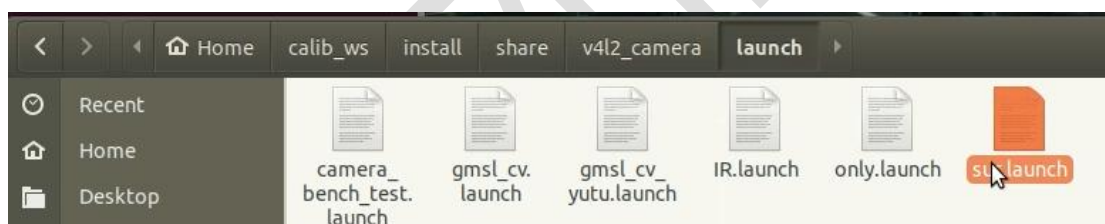
File Edit View Search Terminal Help

nvidia@oceanstar:~$ rosnodetool list
/projector_maps_node
/rosout
/rviz_1747814120224603248
/socket_can0
/socket_can1
/surround_view_node
/ui_node
```

2) 找到

/home/nvidia/calib_ws/install/share/v4l2_camera/launch/sur.launch

将每个相机对应的 false 改为 true，保存后，重启“调试标定软件”发现相机正常



```
sur.launch
~/calib_ws/install/share/v4l2_camera/launch

<launch>
  <arg name="enable_front" default="false" />
  <arg name="enable_back" default="false" />
  <arg name="enable_left" default="false" />
  <arg name="enable_right" default="false" />

  <arg name="width" default="1280" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="UYVY" />
  <arg name="is_31" default="true" />
  <arg name="pub_interval" default="3" />

  <group if="$(arg enable_front)">
    <node pkg="v4l2_camera" type="v4l2_camera" name="front_cam" output="screen">
      <param name="file_name" value="/dev/video0" />
      <param name="width" value="$(arg width)" />
      <param name="height" value="$(arg height)" />
      <param name="count" value="$(arg count)" />
      <param name="pix_fmt" value="$(arg pix_fmt)" />
      <param name="is_31" value="$(arg is_31)" />
      <param name="pub_interval" value="$(arg pub_interval)" />

      <remap from="/image_raw" to="/front_cam/image_raw" />
    </node>
  </group>

  <group if="$(arg enable_left)">
    <node pkg="v4l2_camera" type="v4l2_camera" name="left_cam" output="screen">
      <param name="file_name" value="/dev/video1" />
      <param name="width" value="$(arg width)" />
      <param name="height" value="$(arg height)" />
      <param name="count" value="$(arg count)" />
      <param name="pix_fmt" value="$(arg pix_fmt)" />
      <param name="is_31" value="$(arg is_31)" />
      <param name="pub_interval" value="$(arg pub_interval)" />

      <remap from="/image_raw" to="/left_cam/image_raw" />
    </node>
  </group>
</launch>

Loading file "/home/nvidia/calib_ws/install/share/v4l2_camera/launch/... Plain Text Tab Width: 8 Ln 5, Col 44 INS
```

```
*sur.launch
~/calib_ws/install/share/v4l2_camera/launch

<launch>
  <arg name="enable_front" default="true" />
  <arg name="enable_back" default="true" />
  <arg name="enable_left" default="true" />
  <arg name="enable_right" default="true" />

  <arg name="width" default="1280" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="UYVY" />
  <arg name="is_31" default="true" />
  <arg name="pub_interval" default="3" />

  <group if="$(arg enable_front)">
    <node pkg="v4l2_camera" type="v4l2_camera" name="front_cam" output="screen">
      <param name="file_name" value="/dev/video0" />
      <param name="width" value="$(arg width)" />
      <param name="height" value="$(arg height)" />
      <param name="count" value="$(arg count)" />
      <param name="pix_fmt" value="$(arg pix_fmt)" />
      <param name="is_31" value="$(arg is_31)" />
      <param name="pub_interval" value="$(arg pub_interval)" />

      <remap from="/image_raw" to="/front_cam/image_raw" />
    </node>
  </group>

  <group if="$(arg enable_left)">
    <node pkg="v4l2_camera" type="v4l2_camera" name="left_cam" output="screen">
      <param name="file_name" value="/dev/video1" />
      <param name="width" value="$(arg width)" />
      <param name="height" value="$(arg height)" />
      <param name="count" value="$(arg count)" />
      <param name="pix_fmt" value="$(arg pix_fmt)" />
      <param name="is_31" value="$(arg is_31)" />
      <param name="pub_interval" value="$(arg pub_interval)" />

      <remap from="/image_raw" to="/left_cam/image_raw" />
    </node>
  </group>
</launch>

Saving file "/home/nvidia/calib_ws/install/share/v4l2_camera/launch/s... Plain Text Tab Width: 8 Ln 5, Col 43 INS
```

故障机理分析：由于环视相机配置错误，导致未能启动 4 个环视相机节点，最终无法显示 4 个环视相机画面。

实验结果：环视相机 ROS 驱动配置错误，导致无法启动 ROS 节点。

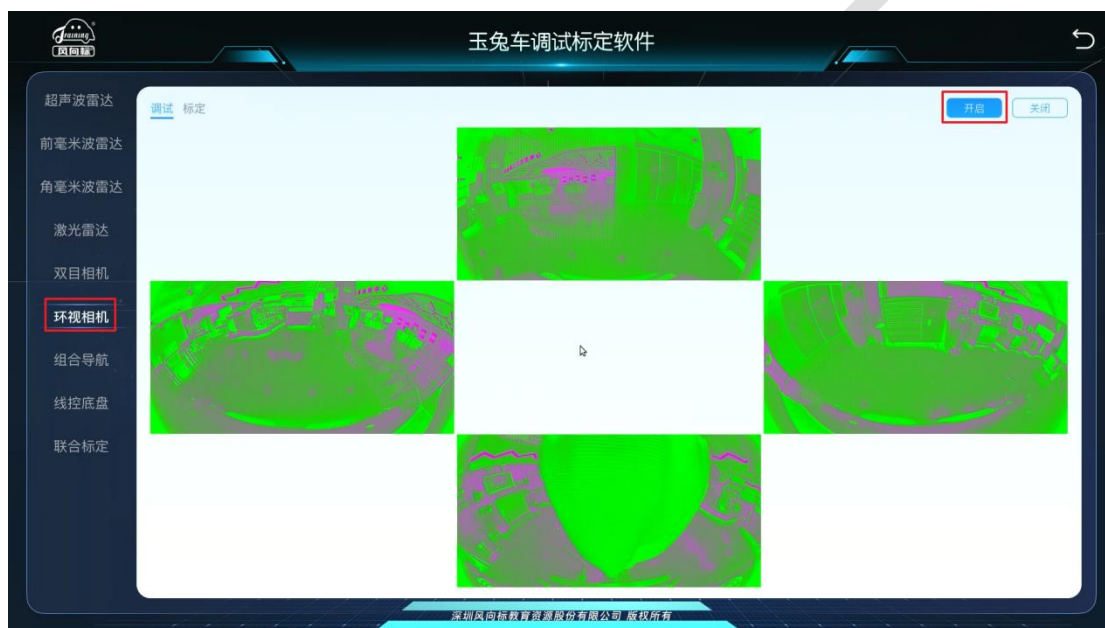
4.17 实验十七

实验对象：环视相机 ROS 驱动的颜色编码配置错误

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象

1) 打开“调试标定软件”，找到“环视相机”点击开启，发现 4 个环视相机的颜色异常。



故障分析

环视相机显示画面颜色异常，可能有以下几个原因

- 1) 启动文件配置错误
- 2) 环视相机线路问题
- 3) 环视相机本身问题

排故过程

1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rosparam get /back_cam/pix_fmt` 发现颜色编码错误，正确为 `UYVY`

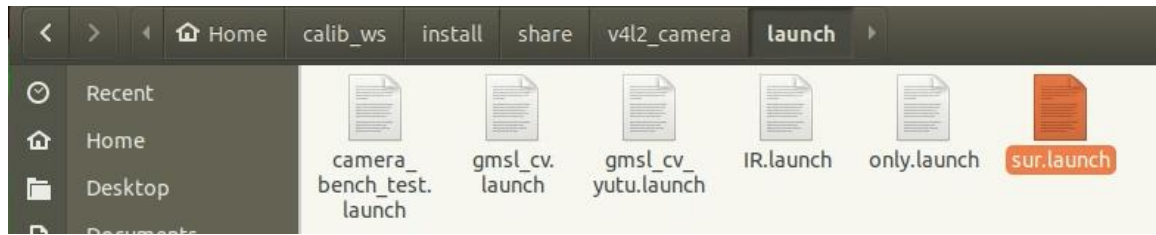
```
nvidia@oceanstar:~$ rosparam get /back_cam/pix_fmt
YUYV
```

2) 找到

`/home/nvidia/calib_ws/install/share/v4l2_camera/launch/sur.launch`



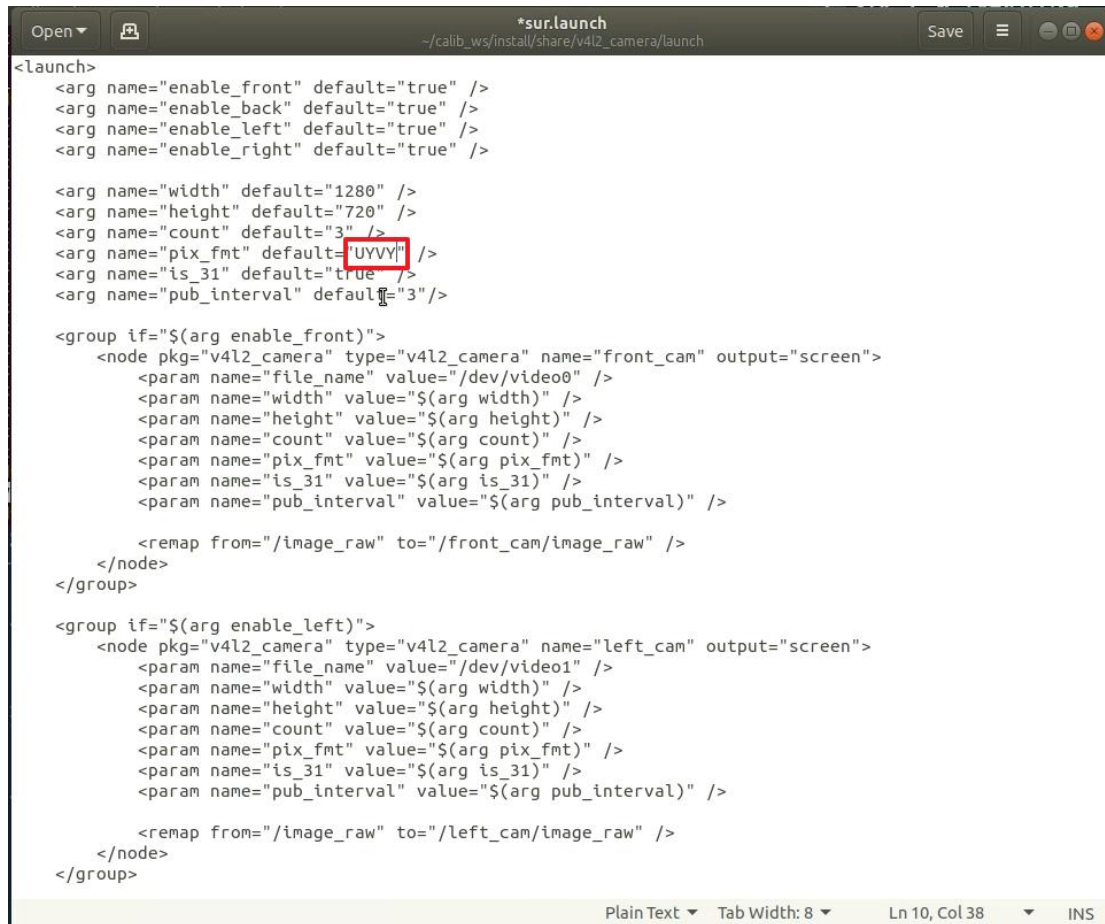
将 YUYV 改为 UYVY



```
Open ▾  sur.launch  
~/calib_ws/install/share/v4l2_camera/launch Save   
```

```
<launch>  
  <arg name="enable_front" default="true" />  
  <arg name="enable_back" default="true" />  
  <arg name="enable_left" default="true" />  
  <arg name="enable_right" default="true" />  
  
  <arg name="width" default="1280" />  
  <arg name="height" default="720" />  
  <arg name="count" default="3" />  
  <arg name="pix_fmt" default="YUYV" />  
  <arg name="is_31" default="true" />  
  <arg name="pub_interval" default="3" />  
  
  <group if="$(arg enable_front)">  
    <node pkg="v4l2_camera" type="v4l2_camera" name="front_cam" output="screen">  
      <param name="file_name" value="/dev/video0" />  
      <param name="width" value="$(arg width)" />  
      <param name="height" value="$(arg height)" />  
      <param name="count" value="$(arg count)" />  
      <param name="pix_fmt" value="$(arg pix_fmt)" />  
      <param name="is_31" value="$(arg is_31)" />  
      <param name="pub_interval" value="$(arg pub_interval)" />  
  
      <remap from="/image_raw" to="/front_cam/image_raw" />  
    </node>  
  </group>  
  
  <group if="$(arg enable_left)">  
    <node pkg="v4l2_camera" type="v4l2_camera" name="left_cam" output="screen">  
      <param name="file_name" value="/dev/video1" />  
      <param name="width" value="$(arg width)" />  
      <param name="height" value="$(arg height)" />  
      <param name="count" value="$(arg count)" />  
      <param name="pix_fmt" value="$(arg pix_fmt)" />  
      <param name="is_31" value="$(arg is_31)" />  
      <param name="pub_interval" value="$(arg pub_interval)" />  
  
      <remap from="/image_raw" to="/left_cam/image_raw" />  
    </node>  
  </group>  
</launch>
```

Loading file "/home/nvidia/calib_ws/install/share/v4l2_camera/launch/... Plain Text ▾ Tab Width: 8 ▾ Ln 10, Col 38 ▾ INS



```
<launch>
  <arg name="enable_front" default="true" />
  <arg name="enable_back" default="true" />
  <arg name="enable_left" default="true" />
  <arg name="enable_right" default="true" />

  <arg name="width" default="1280" />
  <arg name="height" default="720" />
  <arg name="count" default="3" />
  <arg name="pix_fmt" default="UYVY" />
  <arg name="is_31" default="true" />
  <arg name="pub_interval" default="3" />

  <group if="$(arg enable_front)">
    <node pkg="v4l2_camera" type="v4l2_camera" name="front_cam" output="screen">
      <param name="file_name" value="/dev/video0" />
      <param name="width" value="$(arg width)" />
      <param name="height" value="$(arg height)" />
      <param name="count" value="$(arg count)" />
      <param name="pix_fmt" value="$(arg pix_fmt)" />
      <param name="is_31" value="$(arg is_31)" />
      <param name="pub_interval" value="$(arg pub_interval)" />

      <remap from="/image_raw" to="/front_cam/image_raw" />
    </node>
  </group>

  <group if="$(arg enable_left)">
    <node pkg="v4l2_camera" type="v4l2_camera" name="left_cam" output="screen">
      <param name="file_name" value="/dev/video1" />
      <param name="width" value="$(arg width)" />
      <param name="height" value="$(arg height)" />
      <param name="count" value="$(arg count)" />
      <param name="pix_fmt" value="$(arg pix_fmt)" />
      <param name="is_31" value="$(arg is_31)" />
      <param name="pub_interval" value="$(arg pub_interval)" />

      <remap from="/image_raw" to="/left_cam/image_raw" />
    </node>
  </group>
</launch>
```

故障机理分析：由于环视相机 ROS 驱动颜色编码配置错误，导致显示相机画面异常。

实验结果：环视相机 ROS 驱动的颜色编码配置错误

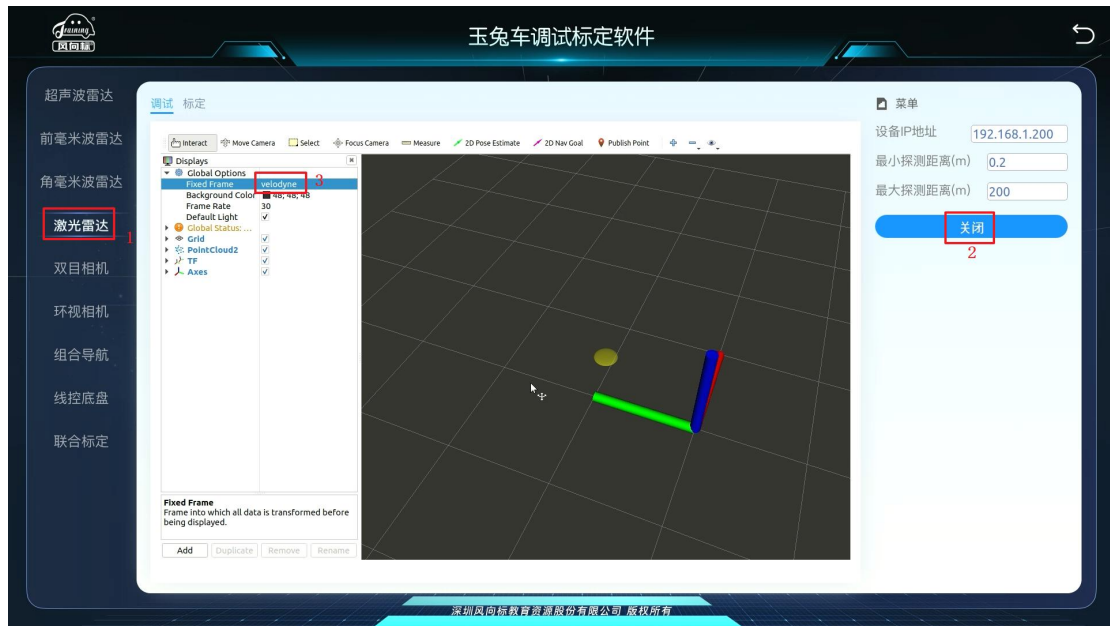
4.18 实验十八

实验对象：激光雷达 ROS 驱动 IP 配置错误

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象

1) 打开“调试标定软件”，找到“激光雷达”点击开启，修改 Fixed Frame 为 velodyne 发现没有点云出现。



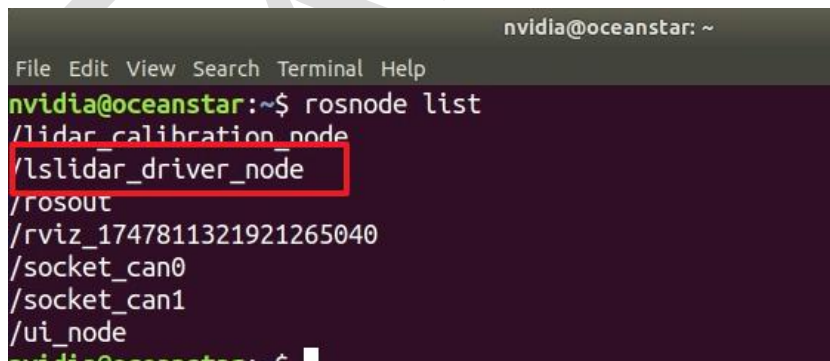
故障分析

激光雷达启动后没有点云出现，可能有以下几个原因

- 1) 没有启动激光雷达节点
- 2) 启动文件配置错误
- 3) 激光雷达线路问题
- 4) 激光雷达本身问题

排查过程

- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rostopic list`，发现存在激光雷达节点 `/lslidar_driver_node`



- 2) 输入 `rostopic info /lslidar_driver_node` 查询节点发布的话题，找到 `/points_raw` 点云话题名

```
nvidia@oceanstar:~$ rosnode info /lslidar_driver_node
```

```
Node [/lslidar_driver_node]
Publications:
* /points_raw [sensor_msgs/PointCloud2]
* /rosout [roscpp_msgs/Log]
* /scan [sensor_msgs/LaserScan]
Subscriptions: None
```

3) 输入 `rostopic hz /points_raw` 查询数据发布帧率，发现没有数据发布，确认没有点云现象。

```
nvidia@oceanstar:~$ rostopic hz /points_raw
subscribed to [/points_raw]
no new messages
no new messages
```

4) 输入 `rosparam get /lslidar_driver_node/device_ip` 查询目标激光雷达 IP 参数，发现异常，正确为 192.168.1.200

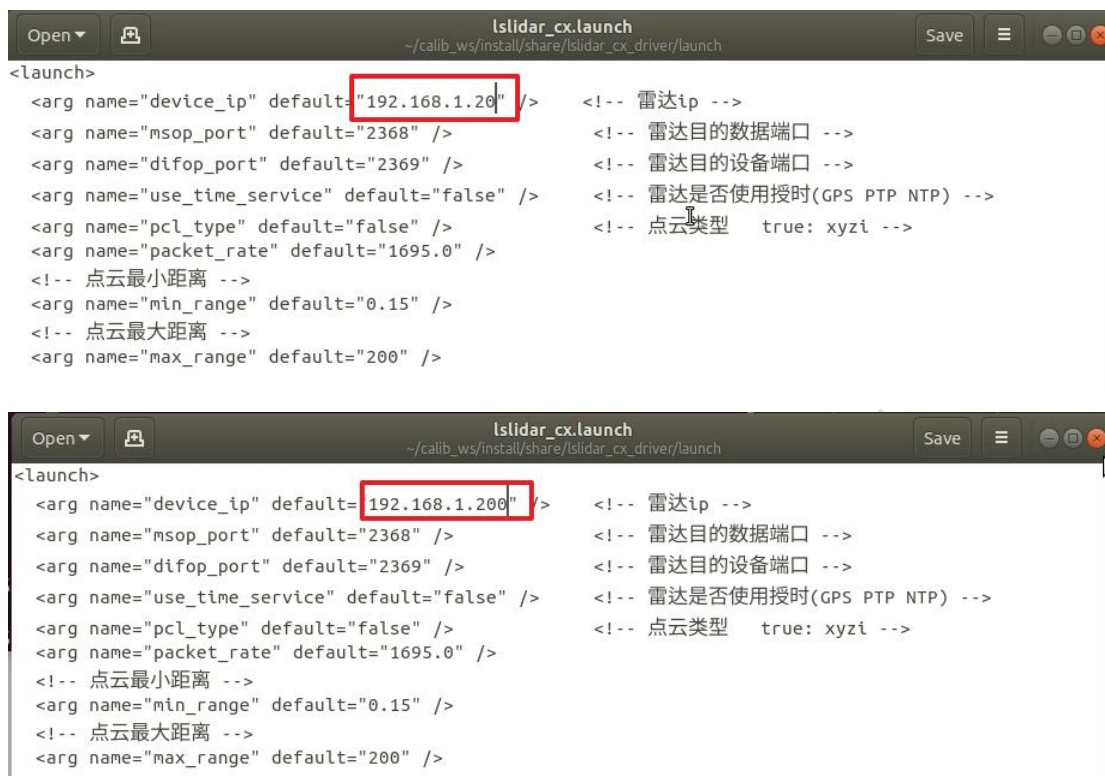
```
nvidia@oceanstar:~$ rosparam get /lslidar_driver_node/device_ip
192.168.1.20
```

5) 找到

`/home/nvidia/calib_ws/install/share/lslidar_cx_driver/launch/lslidar_cx.launch`

将 192.168.1.20 改为 192.168.1.200，保存后，重启“调试标定软件”发现雷达正常。





```

<launch>
  <arg name="device_ip" default="192.168.1.200" />    <!-- 雷达ip -->
  <arg name="msop_port" default="2368" />              <!-- 雷达目的数据端口 -->
  <arg name="difop_port" default="2369" />             <!-- 雷达目的设备端口 -->
  <arg name="use_time_service" default="false" />      <!-- 雷达是否使用授时(GPS PTP NTP) -->
  <arg name="pcl_type" default="false" />              <!-- 点云类型 true: xyzi -->
  <arg name="packet_rate" default="1695.0" />
  <!-- 点云最小距离 -->
  <arg name="min_range" default="0.15" />
  <!-- 点云最大距离 -->
  <arg name="max_range" default="200" />

```

故障机理分析：由于激光雷达 ROS 驱动 IP 参数配置错误，导致 ROS 驱动无法获取激光雷达的数据，最终导致没有点云出现。

实验结果：激光雷达 ROS 驱动 IP 配置错误。

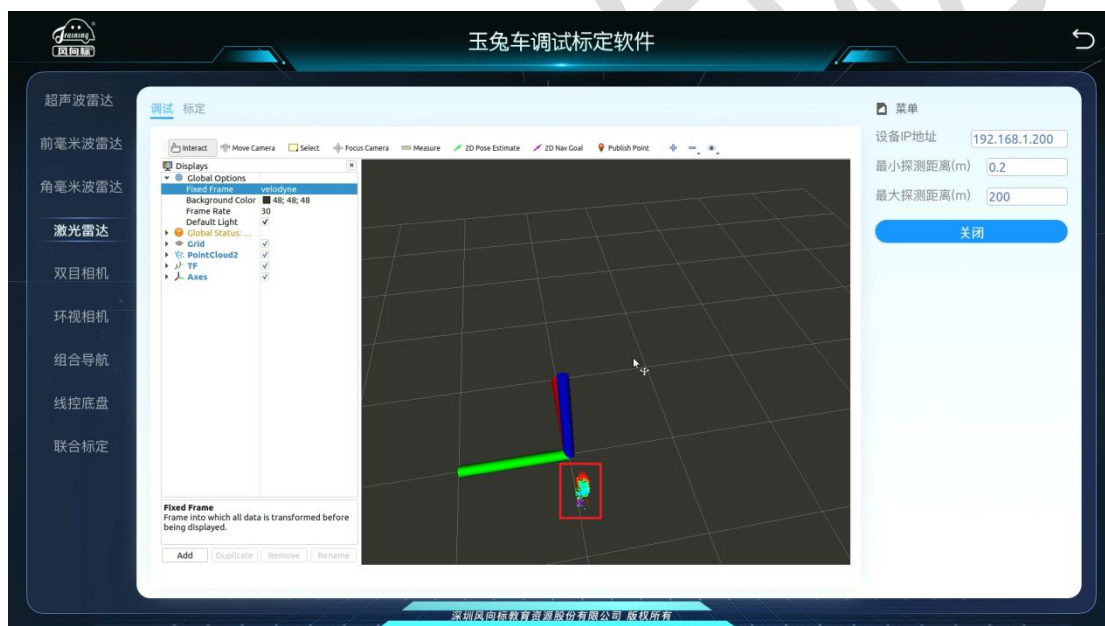
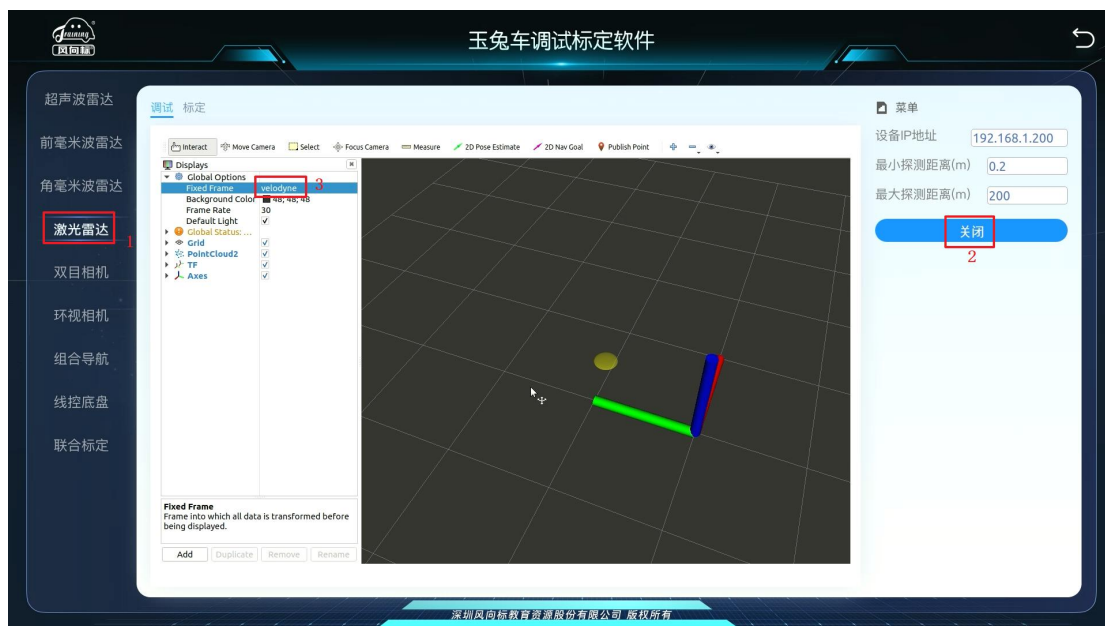
4.19 实验十九

实验对象：激光雷达 ROS 驱动最远距离配置错误。

实验目的：学习传感器 ROS 驱动的启动文件配置。

实验现象

1) 打开“调试标定软件”，找到“激光雷达”点击开启，修改 Fixed Frame 为 velodyne 发现几乎没有点云出现，一旦有物体靠近激光雷达则会有少量点云出现。



故障分析

激光雷达启动后几乎没有点云出现，可能有以下几个原因

- 1) 启动文件配置错误
- 2) 激光雷达线路问题
- 3) 激光雷达本身问题

排查过程

- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rosnode list`，发现存在激光雷达节点 `/lsidar_driver_node`

```
nvidia@oceanstar: ~  
File Edit View Search Terminal Help  
nvidia@oceanstar:~$ rosnodetop  
/lidar_calibration_node  
/lslidar_driver_node  
/rosout  
/rviz_1747811520562663920  
/socket_can0  
/socket_can1  
/ui_node  
nvidia@oceanstar:~$
```

2) 输入 `rostopic info /lslidar_driver_node` 查询节点发布的话题，找到 `/points_raw` 点云话题名

```
nvidia@oceanstar:~$ rostopic info /lslidar_driver_node  
-----  
Node [/lslidar_driver_node] I  
Publications:  
* /points_raw [sensor_msgs/PointCloud2]  
* /rosout [rosgraph_msgs/Log]  
* /scan [sensor_msgs/LaserScan]  
Subscriptions: None
```

3) 输入 `rostopic hz /points_raw` 查询数据发布帧率，发现一旦有物体靠近激光雷达则会有少量点云出现

```
nvidia@oceanstar:~$ rostopic hz /points_raw  
subscribed to [/points_raw]  
no new messages  
no new messages  
no new messages  
no new messages  
no new messages  
average rate: 10.034  
min: 0.099s max: 0.100s std dev: 0.00035s window: 3  
no new messages
```

4) 输入 `rostopic get /lslidar_driver_node/distance_max` 查询雷达最远距离参数，发现参数错误，正确为 200

```
nvidia@oceanstar:~$ rostopic get /lslidar_driver_node/distance_max  
0.5
```

5) 找到

`/home/nvidia/calib_ws/install/share/lslidar_cx_driver/launch/lslidar_cx.launch`



将 0.5 改为 200，保存后，重启“调试标定软件”发现雷达正常。



故障机理分析：由于激光雷达 ROS 驱动最远距离配置错误，导致只能检测出部分点云。

实验结果：激光雷达 ROS 驱动最远距离配置错误

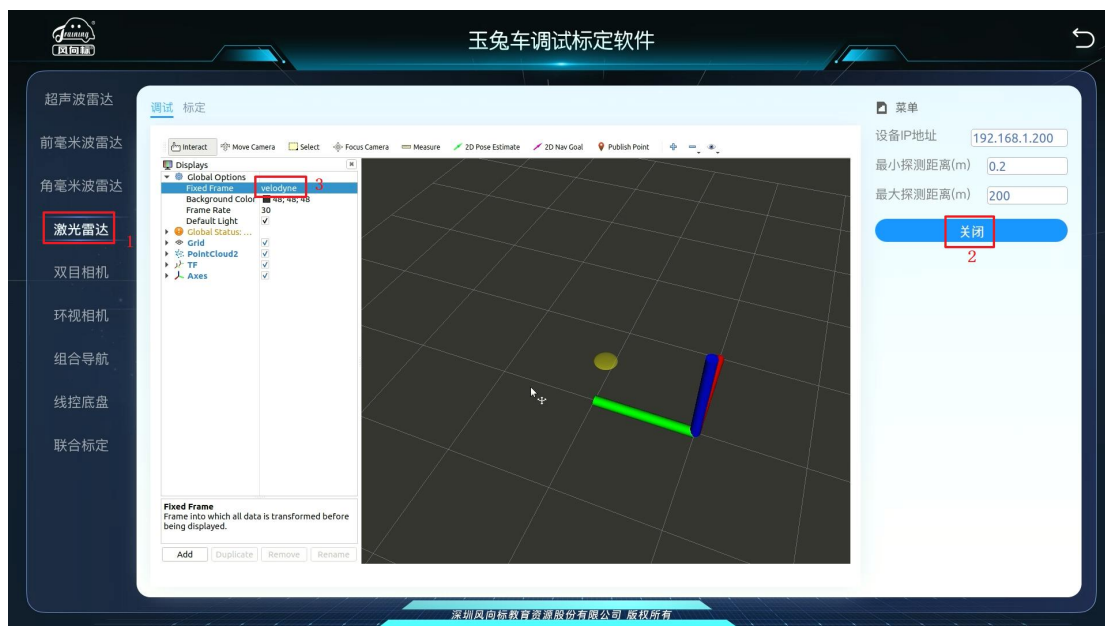
4.20 实验二十

实验对象：激光雷达 ROS 驱动的配置错误，导致无法启动 ROS 节点。

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象

1) 打开“调试标定软件”，找到“激光雷达”点击开启，修改 Fixed Frame 为 velodyne 发现没有点云出现。



故障分析

激光雷达启动后没有点云出现，可能有以下几个原因

- 1) 没有启动激光雷达节点
- 2) 启动文件配置错误
- 3) 激光雷达线路问题
- 4) 激光雷达本身问题

排查过程

- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rostopic list`，发现不存在激光雷达节点 `/lslidar_driver_node`

```
nvidia@oceanstar: ~
File Edit View Search Terminal Help
nvidia@oceanstar:~$ rostopic list
/lidar_calibration_node
/rosout
/rviz_1747811848617367344
/socket_can0
/socket_can1
/ui_node
```

- 2) 找到

`/home/nvidia/calib_ws/install/share/lslidar_cx_driver/launch/lslidar_cx.launch`

将`!--`和`--`删除，保存后，重启“调试标定软件”发现雷达正常。



```

<launch>
  <arg name="device_ip" default="192.168.1.200" />    <!-- 雷达ip -->
  <arg name="msop_port" default="2368" />             <!-- 雷达目的数据端口 -->
  <arg name="difop_port" default="2369" />            <!-- 雷达目的设备端口 -->
  <arg name="use_time_service" default="false" />     <!-- 雷达是否使用授时(GPS PTP NTP) -->
  <arg name="pcl_type" default="false" />             <!-- 点云类型 true: xyzi -->
  <arg name="packet_rate" default="1695.0" />
  <!-- 点云最小距离 -->
  <arg name="min_range" default="0.15" />
  <!-- 点云最大距离 -->
  <arg name="max_range" default="200" />

  <!-- node pkg="lslidar_cx_driver" type="lslidar_cx_driver_node" name="lslidar_driver_node"
  output="screen" -->
  <!--param
  name="pcap" value="$(find lslidar_cx_driver)/pcap/xxx.pcap" /-->
  <param name="use_time_service" value="$(arg use_time_service)" />
  <param name="packet_rate" value="$(arg packet_rate)" />
  <param name="device_ip" value="$(arg device_ip)" />
  <param name="msop_port" value="$(arg msop_port)" />
  <param name="difop port" value="$(arg difop port)" />
</launch>

```

故障机理分析：由于激光雷达 ROS 驱动的配置错误，导致无法启动 ROS 节点，最终没有点云显示。

实验结果：激光雷达 ROS 驱动的配置错误，导致无法启动 ROS 节点。

4.21 实验二十一

实验对象：毫米波雷达 ROS 驱动接收报文话题配置错误

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象



1) 打开“调试标定软件”，找到“前毫米波雷达”，点击“CAN 报文反馈”，发现反馈正常。



玉兔车调试标定软件

左侧菜单：超声波雷达、前毫米波雷达、角毫米波雷达、激光雷达、双目相机、环视相机、组合导航、线控底盘、联合标定。

主界面：调试 标定

清除 Can报文反馈

序号	时间	can通道	传输方向	ID号	帧类型	帧格式	长度	数据
0	15:49:02:71	can1	接收	0x60B	数据帧	标准帧	8	02 4E A3 FE 80 20 01 A0
1	15:49:02:71	can1	接收	0x60B	数据帧	标准帧	8	04 4F 3B F3 80 20 01 A8
2	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	3D 4F AD FD 80 20 01 AA
3	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	33 50 2B F6 80 20 01 B8
4	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	00 50 7B E9 80 1F E1 90
5	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	0A 50 73 D2 80 20 01 96
6	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	08 51 2B C0 80 20 01 B6
7	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	01 52 03 FB 80 20 05 8A
8	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	1A 53 04 0C 80 20 01 7C
9	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	2B 53 84 03 80 1F E1 78

深圳风向往标教育科技股份有限公司 版权所有

2) 点击“数据解析”，点击摘取按钮，发现一直无法摘取到报文。



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

调试 标定

 读取报文 ☒ 考核模式 ☐ 数据解析 ☐

帧ID(HEX) 0X60B 数据 01 A2 BF C3 D4 F5 A3 D1

	7	6	5	4	3	2	1	0
0	7	6	5	4	3	2	1	0
1	纵向距离	15	14	13	12	11	10	9
2	纵向距离	23	22	21	20	19	18	17
3	纵向距离	31	30	29	28	27	26	25
4	纵向速度	39	38	37	36	35	34	33
5	纵向速度	47	46	45	44	43	42	41
6	纵向速度	55	54	53	52	51	50	49
7	散射面积	63	62	61	60	59	58	57

2进制

00000000

总线值

0

解析后的数据

横向距离(m)

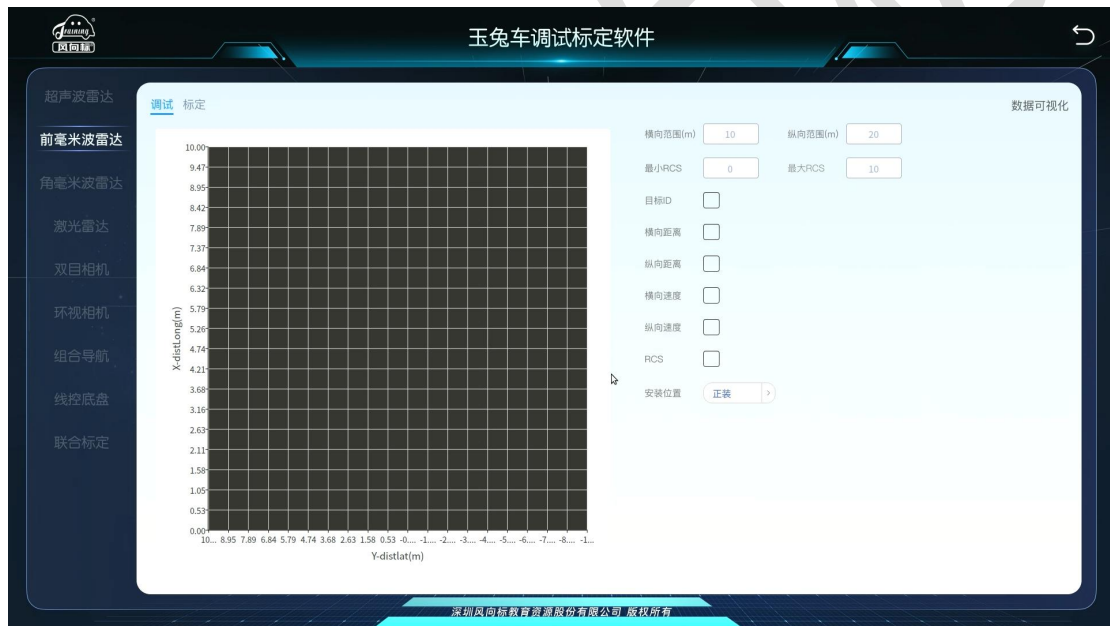
纵向距离(m)

横向速度(m/s)

纵向速度(m/s)

散射面积(dBm2)

3) 点击“数据可视化”，发现一直无法检测到目标显示出来。



故障分析

毫米波雷达反馈报文正常，但是无法摘取报文和无法数据可视化，可能有以下几个原因。

- 1) 没有启动毫米波雷达节点
- 2) 启动文件配置错误

排查过程

- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rosnode list`，发现存在毫米波雷达节点 `/ars_40X_node`


```
nvidia@oceanstar: ~  
File Edit View Search Terminal Help  
nvidia@oceanstar:~$ rosnodetop  
/ars_40X_node  
/rosout  
/rviz_1747813723342363632  
/socket_can0  
/socket_can1  
/ui_node
```

2) 输入 `rostopic info /ars_40X_node` 查询该节点发布和订阅情况，发现 CAN 订阅话题错误，正确为 `/can1_receiver`。

```
nvidia@oceanstar:~$ rostopic info /ars_40X_node  
-----  
Node [/ars_40X_node]  
Publications:  
* /can1_sender [can_msgs/Frame]  
* /radar0_detect_can [fxb_sensor_msgs_srvs/RadarArs408]  
* /rosout [roscpp_msgs/Log]  
Subscriptions:  
* /can0_receiver [can_msgs/Frame]
```

3) 输入 `rostopic param get /ars_40X_node/can_receiver` 查询订阅 CAN 话题名参数，发现配置错误，正确为 `can1_receiver`

```
nvidia@oceanstar:~$ rostopic param get /ars_40X_node/can_receiver  
can0_receiver
```

4) 找到

`/home/nvidia/calib_ws/install/share/ars_40x/launch/ars_40X.launch`

将 `can0_receiver` 改为 `can1_receiver`，保存后，重启“调试标定软件”发现毫米波雷达正常。



```
ars_40X.launch
~/calib_ws/install/share/ars_40x/launch

<launch>
  <arg name="can_receiver" default="can0_receiver"/>
  <arg name="can_sender" default="can1_sender"/>
  <arg name="is_pub_can" default="false"/>
  <arg name="error_dist" default="0.3"/>
  <arg name="error_angle" default="10"/>
  <arg name="no_response" default="false"/>

  <node pkg="ars_40x" type="ars_40X_node" name="ars_40X_node" output="screen">
    <param name="can_receiver" value="$(arg can_receiver)" type="str"/>
    <param name="can_sender" value="$(arg can_sender)" type="str"/>
    <param name="is_pub_can" value="$(arg is_pub_can)" type="bool"/>
    <param name="error_dist" value="$(arg error_dist)" type="double"/>
    <param name="error_angle" value="$(arg error_angle)" type="double"/>
    <param name="no_response" value="$(arg no_response)" type="bool"/>
  </node>
</launch>
```

```
*ars_40X.launch
~/calib_ws/install/share/ars_40x/launch

<launch>
  <arg name="can_receiver" default="can1_receiver"/>
  <arg name="can_sender" default="can1_sender"/>
  <arg name="is_pub_can" default="false"/>
  <arg name="error_dist" default="0.3"/>
  <arg name="error_angle" default="10"/>
  <arg name="no_response" default="false"/>

  <node pkg="ars_40x" type="ars_40X_node" name="ars_40X_node" output="screen">
    <param name="can_receiver" value="$(arg can_receiver)" type="str"/>
    <param name="can_sender" value="$(arg can_sender)" type="str"/>
    <param name="is_pub_can" value="$(arg is_pub_can)" type="bool"/>
    <param name="error_dist" value="$(arg error_dist)" type="double"/>
    <param name="error_angle" value="$(arg error_angle)" type="double"/>
    <param name="no_response" value="$(arg no_response)" type="bool"/>
  </node>
</launch>
```

故障机理分析：由于毫米波雷达 ROS 驱动接收报文话题配置错误，导致无法解析数据，最终造成“数据解析”和“数据可视化”功能异常。

实验结果：毫米波雷达 ROS 驱动接收报文话题配置错误。

4.22 实验二十二

实验对象：毫米波雷达 ROS 驱动的配置错误，导致无法启动 ROS 节点。

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象

1) 打开“调试标定软件”，找到“前毫米波雷达”，点击“CAN 报文反馈”，发现反馈正常。



玉兔车调试标定软件

调试 标定
清除 Can报文反馈

序号	时间	can通道	传输方向	ID号	帧类型	帧格式	长度	数据
0	15:49:02:71	can1	接收	0x60B	数据帧	标准帧	8	02 4E A3 FE 80 20 01 A0
1	15:49:02:71	can1	接收	0x60B	数据帧	标准帧	8	04 4F 3B F3 80 20 01 A8
2	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	3D 4F AD FD 80 20 01 AA
3	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	33 50 2B F6 80 20 01 B8
4	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	00 50 7B E9 80 1F E1 90
5	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	0A 50 73 D2 80 20 01 96
6	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	08 51 2B C0 80 20 01 86
7	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	01 52 03 FB 80 20 05 8A
8	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	1A 53 04 0C 80 20 01 7C
9	15:49:02:72	can1	接收	0x60B	数据帧	标准帧	8	2B 53 84 03 80 1F E1 78

深圳风向往标教育股份有限公司 版权所有

2) 点击“数据解析”，点击摘取按钮，发现一直无法摘取到报文。



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

调试 标定

 读取报文 ☒ 考核模式 ☐ 数据解析 ☐

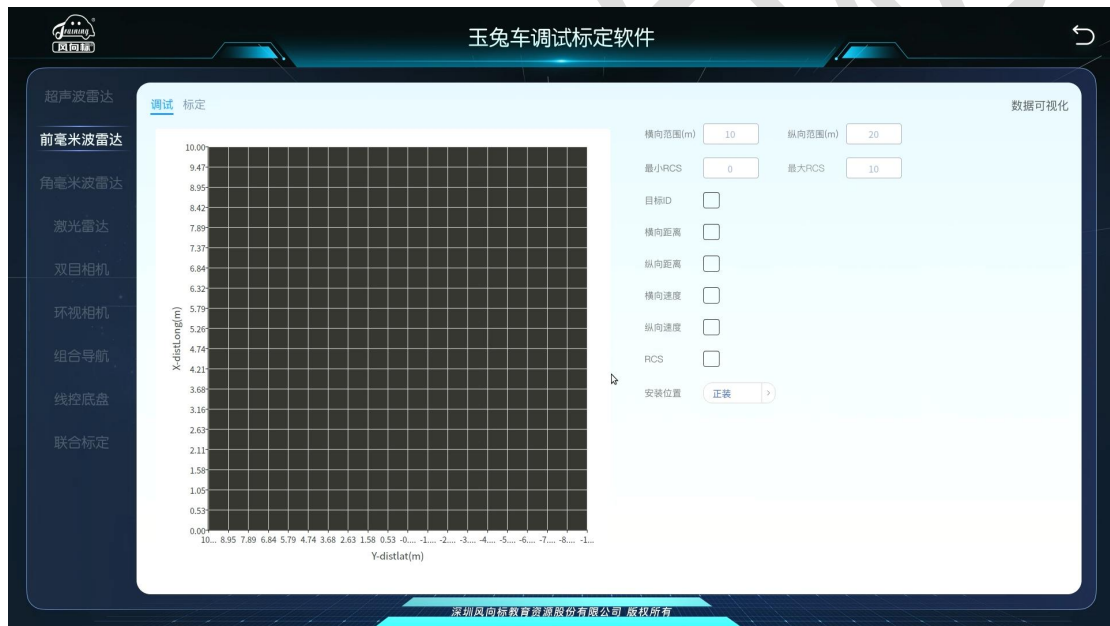
帧ID(HEX) 01 A2 BF C3 D4 F5 A3 D1

	7	6	5	4	3	2	1	0
0	7	6	5	4	3	2	1	0
1	纵向距离	15	14	13	12	11	10	9
2	纵向距离	23	22	21	20	19	18	17
3	纵向距离	31	30	29	28	27	26	25
4	纵向速度	39	38	37	36	35	34	33
5	纵向速度	47	46	45	44	43	42	41
6	纵向速度	55	54	53	52	51	50	49
7	散射面积	63	62	61	60	59	58	57

2进制	00000000
总数值	0

解析后的数据	横向距离(m)	0
	纵向距离(m)	0
	横向速度(m/s)	0
	纵向速度(m/s)	0
	散射面积(dBm2)	0

3) 点击“数据可视化”，发现一直无法检测到目标显示出来。



故障分析

毫米波雷达反馈报文正常，但是无法摘取报文和无法数据可视化，可能有以下几个原因。

- 1) 没有启动毫米波雷达节点
- 2) 启动文件配置错误

排查过程

- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rosnode list`，发现不存在毫米波雷达节点/`ars_40X_node`

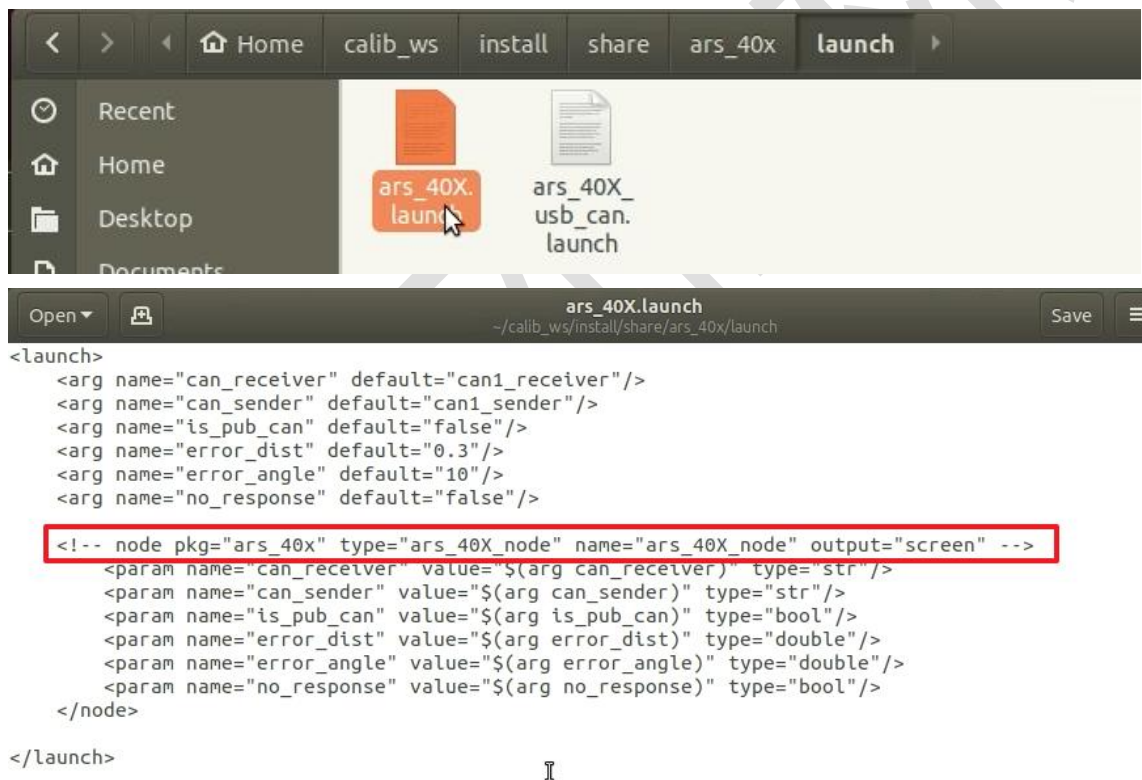
```

nvidia@oceanstar: ~
File Edit View Search Terminal Help
nvidia@oceanstar:~$ rosnodetool list
/roscout
/rviz_1747813936100051472
/socket_can0
/socket_can1
/ui_node
  
```

2) 找到

/home/nvidia/calib_ws/install/share/ars_40x/launch/ars_40X.launch

删除!--和--, 保存后, 重启“调试标定软件”发现毫米波雷达正常。



The image shows a file manager window with the path `calib_ws` / `install` / `share` / `ars_40x` / `launch`. It displays two files: `ars_40X.launch` (highlighted) and `ars_40X_usb_can.launch`. Below, a code editor shows the content of `ars_40X.launch`. The line `<!-- node pkg="ars_40x" type="ars_40X_node" name="ars_40X_node" output="screen" -->` is highlighted with a red box.

```

<launch>
  <arg name="can_receiver" default="can1_receiver"/>
  <arg name="can_sender" default="can1_sender"/>
  <arg name="is_pub_can" default="false"/>
  <arg name="error_dist" default="0.3"/>
  <arg name="error_angle" default="10"/>
  <arg name="no_response" default="false"/>

  <!-- node pkg="ars_40x" type="ars_40X_node" name="ars_40X_node" output="screen" -->
  <param name="can_receiver" value="$(arg can_receiver)" type="str"/>
  <param name="can_sender" value="$(arg can_sender)" type="str"/>
  <param name="is_pub_can" value="$(arg is_pub_can)" type="bool"/>
  <param name="error_dist" value="$(arg error_dist)" type="double"/>
  <param name="error_angle" value="$(arg error_angle)" type="double"/>
  <param name="no_response" value="$(arg no_response)" type="bool"/>
</node>
</launch>
  
```

```

ars_40X.launch
~/calib_ws/install/share/ars_40x/launch

<launch>
  <arg name="can_receiver" default="can1_receiver"/>
  <arg name="can_sender" default="can1_sender"/>
  <arg name="is_pub_can" default="false"/>
  <arg name="error_dist" default="0.3"/>
  <arg name="error_angle" default="10"/>
  <arg name="no_response" default="false"/>

  <node pkg="ars_40x" type="ars_40X_node" name="ars_40X_node" output="screen">
    <param name="can_receiver" value="$(arg can_receiver)" type="str"/>
    <param name="can_sender" value="$(arg can_sender)" type="str"/>
    <param name="is_pub_can" value="$(arg is_pub_can)" type="bool"/>
    <param name="error_dist" value="$(arg error_dist)" type="double"/>
    <param name="error_angle" value="$(arg error_angle)" type="double"/>
    <param name="no_response" value="$(arg no_response)" type="bool"/>
  </node>
</launch>
  
```

故障机理分析：由于毫米波雷达 ROS 驱动配置错误，导致无法启动 ROS 节点，最终造成“数据解析”和“数据可视化”功能异常。

实验结果：学习传感器 ROS 驱动的启动文件配置。

4.23 实验二十三

实验对象：超声波雷达 ROS 驱动接收报文话题配置错误。

实验目的：学习传感器 ROS 驱动的启动文件配置。

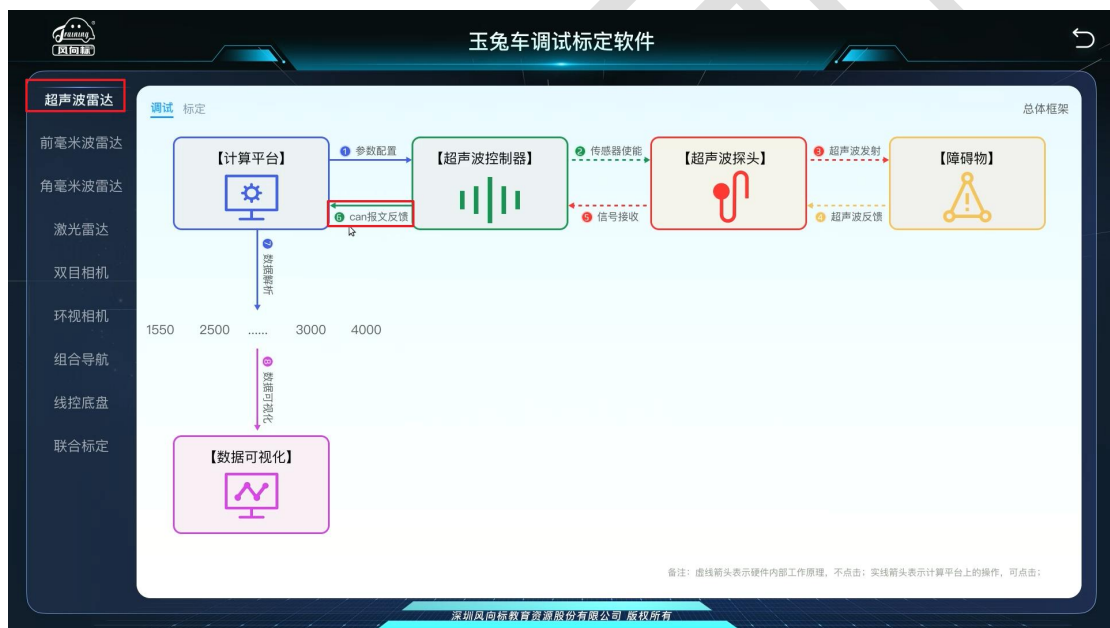
实验现象

- 1) 打开“调试标定软件”，找到“超声波雷达”，点击“参数配置”，发送 B3 1F FF（或者输入 B7 1F FF 两者只是量程不一样，B7 量程短，在短距离精度高）开启所有探头报文



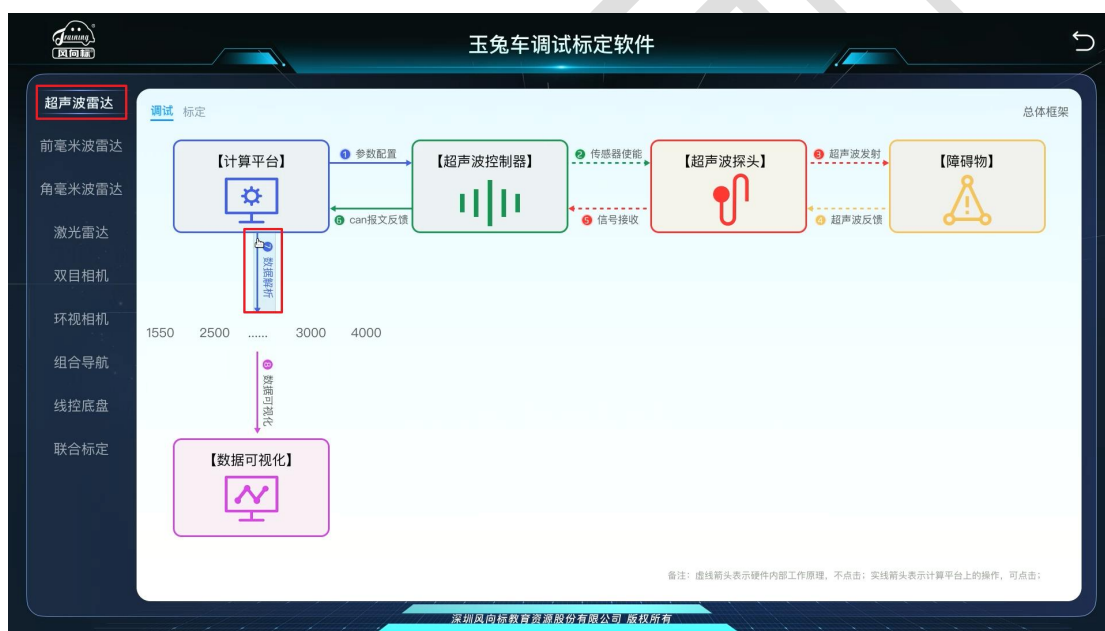


2) 点击“can 报文反馈”发现有反馈报文。





3) 点击“数据解析”，发现一直无法摘取报文



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

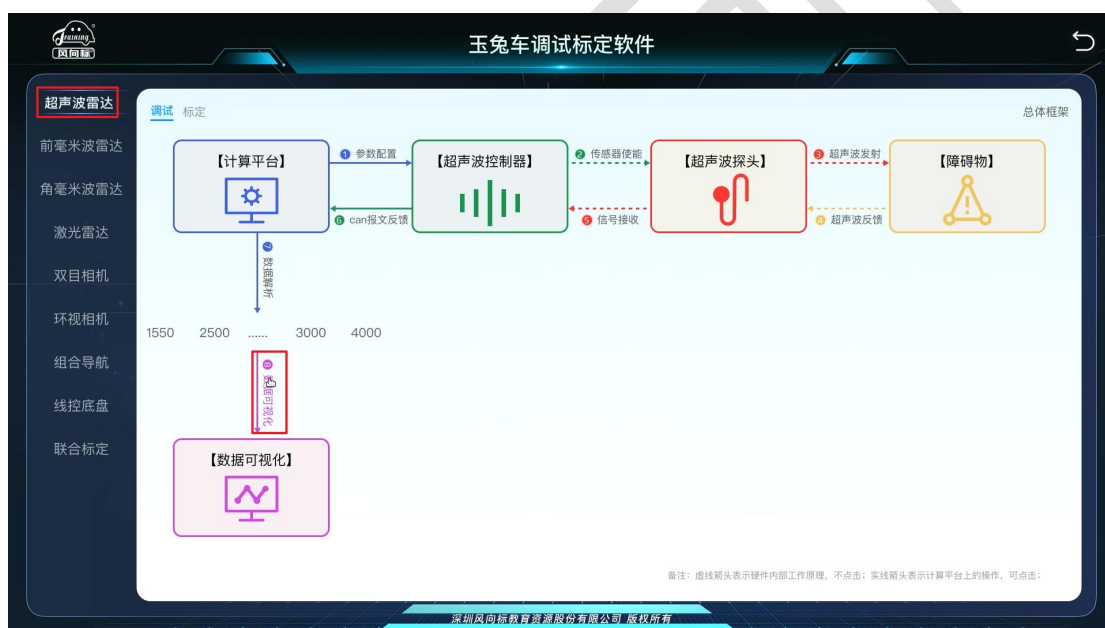
调试 标定
考核模式 ☐ 摘取报文 ☒ 数据解析 ☒

ID号	d1 0x0611								d2 0x0612								d3 0x0613								
字节序号	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
原始报文(16进制)	15	50	25	00	30	00	40	00	15	50	25	00	30	00	40	00	15	50	25	00	30	00	40	00	从can网络中获取id为0x0611、0x0612、0x0613的报文数据 每个报文包含8个字节，序号分别为0-7
转换关系	↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓								↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓								↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓								原始报文字节到超声波序号的对应关系
超声波序号	1路	2路	3路	4路					5路	6路	7路	8路					9路	10路	11路	12路					
解析后数据(10进制,mm)	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>					<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>					<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>					实际的12路超声波距离值

*换算公式: $(d1[i]/16*10+d1[i+1]\%16)*100+d1[i+1]*16*10+d1[i+2]\%16$ *备注: d指代ID号的报文, 取d1、d2、d3, i指代报文中的字节序号, 取0、2、4、6

深圳风向往标教育股份有限公司 版权所有

4) 点击“数据可视化”，发现无法可视化





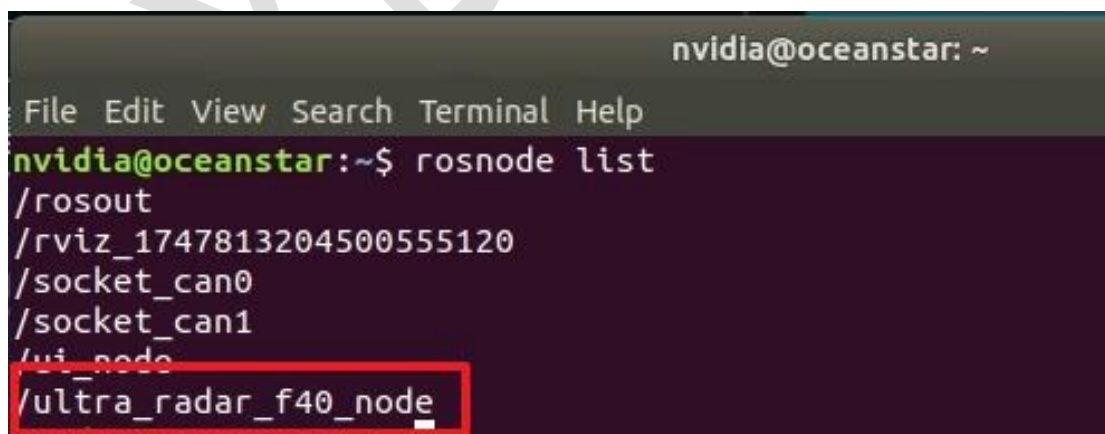
故障分析

超声波雷达反馈报文正常，但是无法摘取报文和无法数据可视化，可能有以下几个原因。

- 1) 没有启动超声波雷达节点
- 2) 启动文件配置错误

排查过程

- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rostopic list`，发现存在超声波雷达节点 `/ultra_radar_f40_node`



- 2) 输入 `rostopic info /ultra_radar_f40_node` 查询该节点的发布和订阅情况，发现订阅 CAN 话题名错误，正确为 `/can1_receiver`

```
nvidia@oceanstar:~$ roscore info /ultra_radar_f40_node
-----
Node [/ultra_radar_f40_node]
Publications:
* /can1_sender [can_msgs/Frame]
* /rosout [rosgraph_msgs/Log]
* /ultra_detect_can [fxb_sensor_msgs_srvs/UltraF40]

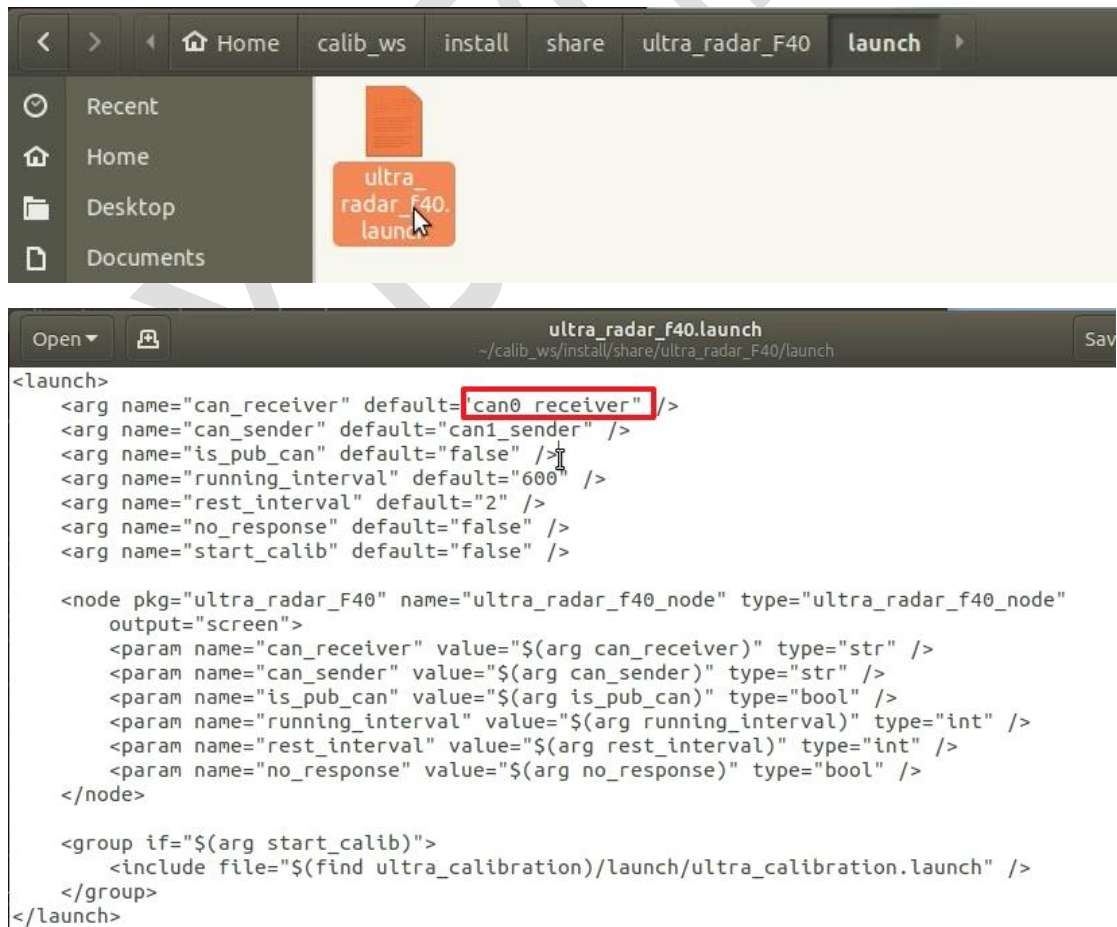
Subscriptions:
* /can0_receiver [can_msgs/Frame]
```

3) 输入 `rosparam get /ultra_radar_f40_node/can_receiver` 查询订阅 CAN 话题名参数，发现错误，正确为 `/can1_receiver`

```
nvidia@oceanstar:~$ rosparam get /ultra_radar_f40_node/can_receiver
can0_receiver
```

4) 找到

`/home/nvidia/calib_ws/install/share/ultra_radar_F40/launch/ultra_radar_f40.launch`
将 `can0_receiver` 改为 `can1_receiver`，保存后，重启“调试标定软件”发现超声波雷达正常。




```
ultra_radar_f40.launch
~/calib_ws/install/share/ultra_radar_f40/launch

<launch>
  <arg name="can_receiver" default="can1_receiver" />
  <arg name="can_sender" default="can1_sender" />
  <arg name="is_pub_can" default="false" />
  <arg name="running_interval" default="600" />
  <arg name="rest_interval" default="2" />
  <arg name="no_response" default="false" />
  <arg name="start_calib" default="false" />

  <node pkg="ultra_radar_f40" name="ultra_radar_f40_node" type="ultra_radar_f40_node"
    output="screen">
    <param name="can_receiver" value="$(arg can_receiver)" type="str" />
    <param name="can_sender" value="$(arg can_sender)" type="str" />
    <param name="is_pub_can" value="$(arg is_pub_can)" type="bool" />
    <param name="running_interval" value="$(arg running_interval)" type="int" />
    <param name="rest_interval" value="$(arg rest_interval)" type="int" />
    <param name="no_response" value="$(arg no_response)" type="bool" />
  </node>

  <group if="$(arg start_calib)">
    <include file="$(find ultra_calibration)/launch/ultra_calibration.launch" />
  </group>
</launch>
```

故障机理分析：由于超声波雷达 ROS 驱动接收报文话题配置错误，导致节点一直无法获取超声波报文，最终造成“数据解析”和“数据可视化”功能异常。

实验结果：超声波雷达 ROS 驱动接收报文话题配置错误

4.24 实验二十四

实验对象：超声波雷达 ROS 驱动的配置错误，导致无法启动 ROS 节点。

实验目的：学习传感器 ROS 驱动的启动文件配置

实验现象

1) 打开“调试标定软件”，找到“超声波雷达”，点击“参数配置”，发送 B3 1F FF（或者输入 B7 1F FF 两者只是量程不一样，B7 量程短，在短距离精度高）开启所有探头报文



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

调试 标定
考核模式 ☐ 配置参数

报文发送

波特率: 500k bps | 帧格式: 标准帧 | 帧类型: 数据帧

帧ID(Hex) OX: 601 | 数据长度DLC OX: 3

数据: B3 1F FF | 发送

报文编写

工作模式: 大于2m | 总线值: 177 | 2进制: 10110001

探头状态: 开启 | 4095 | 10110001

	7	6	5	4	3	2	1	0
0	35	14	13	12	11	10	9	8
1	35	25	21	20	18	17	16	15
2	31	30	29	28	27	26	25	24
3	30	38	37	36	35	34	33	32
4	47	46	45	44	43	42	41	40
5	55	54	53	52	51	50	49	48
6	63	62	61	60	59	58	57	56

2) 点击“can 报文反馈”发现有反馈报文。



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

调试 标定

清除 Can报文反馈

序号	时间	can通道	传输方向	ID号	帧类型	帧格式	长度	数据
0	15:40:36:21	can1	接收	0x611	数据帧	标准帧	8	50 00 50 00 32 35 02 55
1	15:40:36:21	can1	接收	0x612	数据帧	标准帧	8	13 20 19 00 02 50 02 50
2	15:40:36:21	can1	接收	0x613	数据帧	标准帧	8	02 50 02 75 50 00 50 00
3	15:40:36:21	can1	接收	0x611	数据帧	标准帧	8	50 00 50 00 32 35 02 65
4	15:40:36:21	can1	接收	0x612	数据帧	标准帧	8	07 90 03 00 02 50 02 65
5	15:40:36:21	can1	接收	0x613	数据帧	标准帧	8	02 50 02 75 50 00 50 00
6	15:40:36:38	can1	接收	0x611	数据帧	标准帧	8	50 00 50 00 32 35 02 65
7	15:40:36:38	can1	接收	0x612	数据帧	标准帧	8	13 20 03 20 02 50 02 60
8	15:40:36:38	can1	接收	0x613	数据帧	标准帧	8	02 60 03 00 50 00 50 00
9	15:40:36:38	can1	接收	0x611	数据帧	标准帧	8	50 00 50 00 32 35 02 65
10	15:40:36:39	can1	接收	0x612	数据帧	标准帧	8	13 20 19 05 02 50 02 50
11	15:40:36:39	can1	接收	0x613	数据帧	标准帧	8	02 50 02 90 50 00 50 00

深圳风向往标教育科技股份有限公司 版权所有

3) 点击“数据解析”，发现一直无法摘取报文



玉兔车调试标定软件

超声波雷达

前毫米波雷达

角毫米波雷达

激光雷达

双目相机

环视相机

组合导航

线控底盘

联合标定

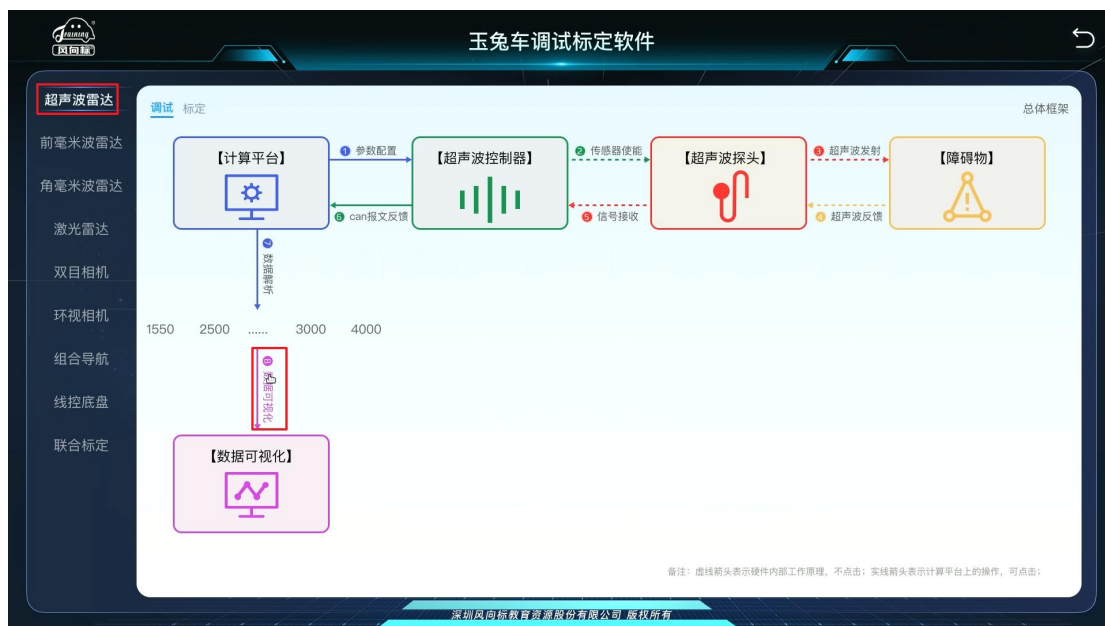
调试 标定
考核模式 ☐ 抽取报文 ☒ 数据解析

ID号	d1 0x0611								d2 0x0612								d3 0x0613								
字节序号	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
原始报文(16进制)	15	50	25	00	30	00	40	00	15	50	25	00	30	00	40	00	15	50	25	00	30	00	40	00	每个报文包含8个字节，序号分别为0-7
转换关系	↓ ↓ ↓ ↓								↓ ↓ ↓ ↓								↓ ↓ ↓ ↓								原始报文字节到超声波序号的对应关系
超声波序号	1路		2路		3路		4路		5路		6路		7路		8路		9路		10路		11路		12路		超声波ID号
解析后数据(10进制,mm)	<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		实际的12路超声波距离值

*换算公式：(d1[j]/16*10+d1[j+1]*16)*100+d1[j+1]*16+d1[j+1]*16
 *备注：d代表ID号的报文，取d1、d2、d3，j代表报文中的字节序号，取0、2、4、6

深圳风向往标教育股份有限公司 版权所有

4) 点击“数据可视化”，发现无法可视化



故障分析

超声波雷达反馈报文正常，但是无法摘取报文和无法数据可视化，可能有以下几个原因。

- 1) 没有启动超声波雷达节点
- 2) 启动文件配置错误

排查过程


- 1) 使用快捷键 `ctrl+alt+t` 打开终端，输入 `rostopic list`，发现不存在超声波雷达节点/`ultra_radar_f40_node`

```
nvidia@oceanstar: ~
File Edit View Search Terminal Help
nvidia@oceanstar:~$ rosnodetool list
/roscout
/rviz_1747813482404104720
/socket_can0
/socket_can1
/ui_node
```

2) 找到

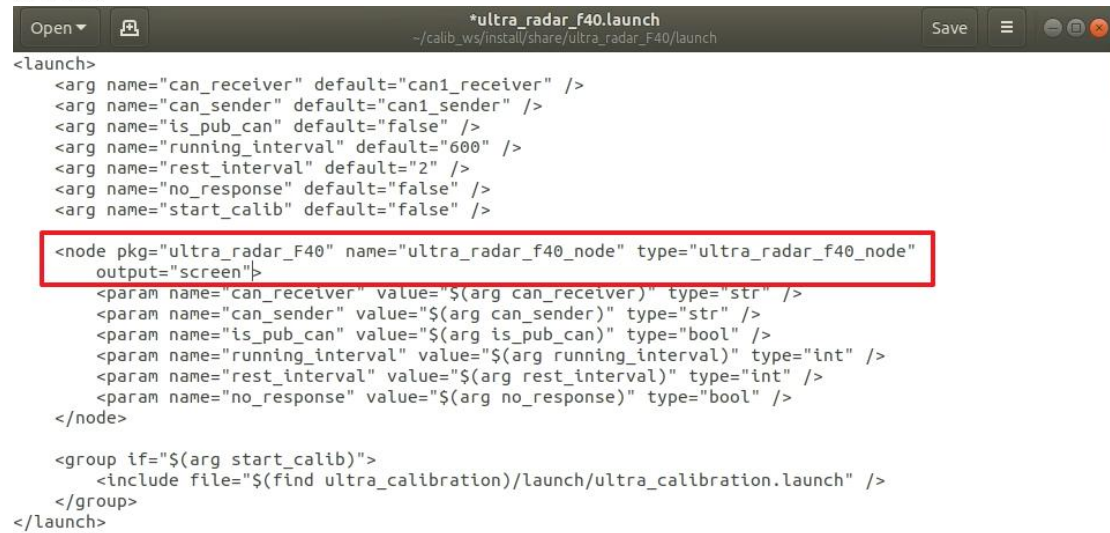
/home/nvidia/calib_ws/install/share/ultra_radar_F40/launch/ultra_radar_f40.launch

删除!--和--, 保存后, 重启“调试标定软件”发现超声波雷达正常。



```
ultra_radar_f40.launch
~/calib_ws/install/share/ultra_radar_F40/launch
Save
```

```
<!-- node pkg="ultra_radar_F40" name="ultra_radar_f40_node" type="ultra_radar_f40_node"
      output="screen" -->
<param name="can_receiver" value="$(arg can_receiver)" type="str" />
<param name="can_sender" value="$(arg can_sender)" type="str" />
<param name="is_pub_can" value="$(arg is_pub_can)" type="bool" />
<param name="running_interval" value="$(arg running_interval)" type="int" />
<param name="rest_interval" value="$(arg rest_interval)" type="int" />
<param name="no_response" value="$(arg no_response)" type="bool" />
</node>
<group if="$(arg start_calib)">
  <include file="$(find ultra_calibration)/launch/ultra_calibration.launch" />
</group>
</launch>
```



故障机理分析：由于超声波雷达 ROS 驱动配置错误，导致无法启动节点，最终造成“数据解析”和“数据可视化”功能异常。

实验结果：超声波雷达 ROS 驱动的配置错误，导致无法启动 ROS 节点。

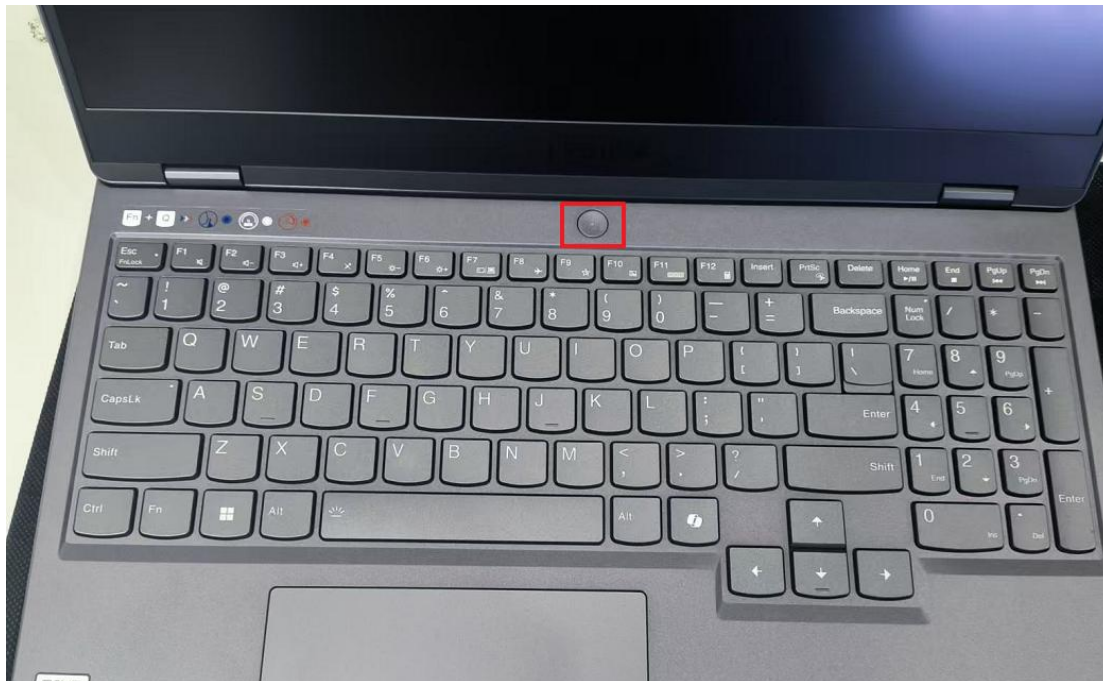
第 5 章 自动驾驶仿真测试系统

5.1 使用准备

- 1) 准备好车辆和仿真笔记本电脑以及网线和笔记本充电器。
- 2) 将网线连接到车辆和笔记本电脑（有些车辆无外置网口，可以打开车辆盖子，找到计算单元的网口连接）。



- 3) 开机，选择 Ubuntu 系统，进入系统后双击桌面上的“玉兔仿真测试软件”。

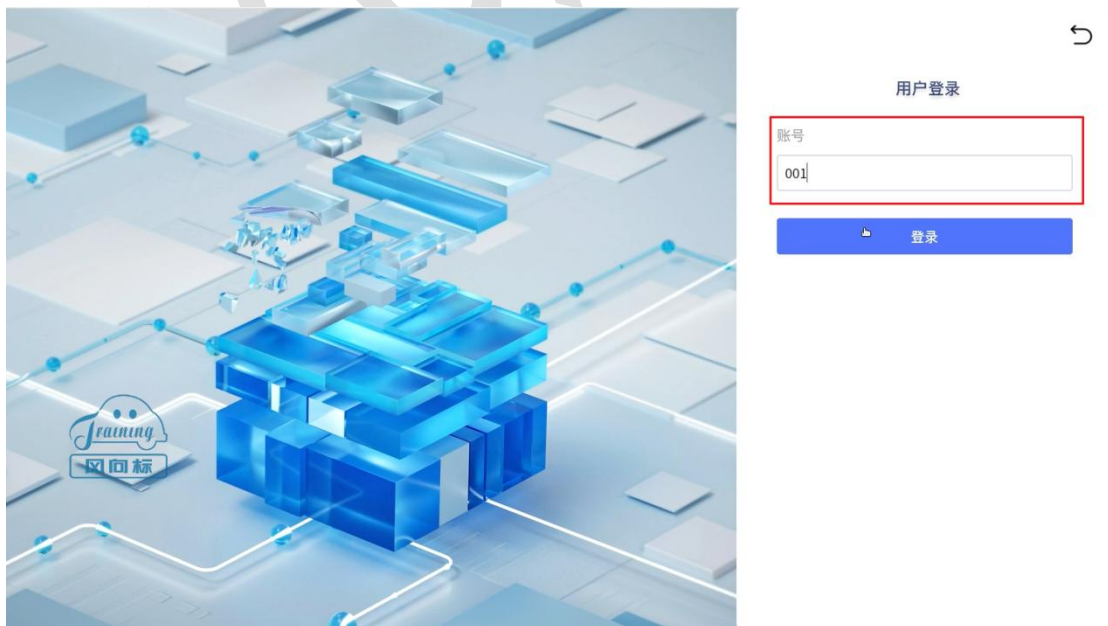




4) 为了保证软件运行流畅，使用软件时最好将笔记本的电源连接，让笔记本充着电使用。

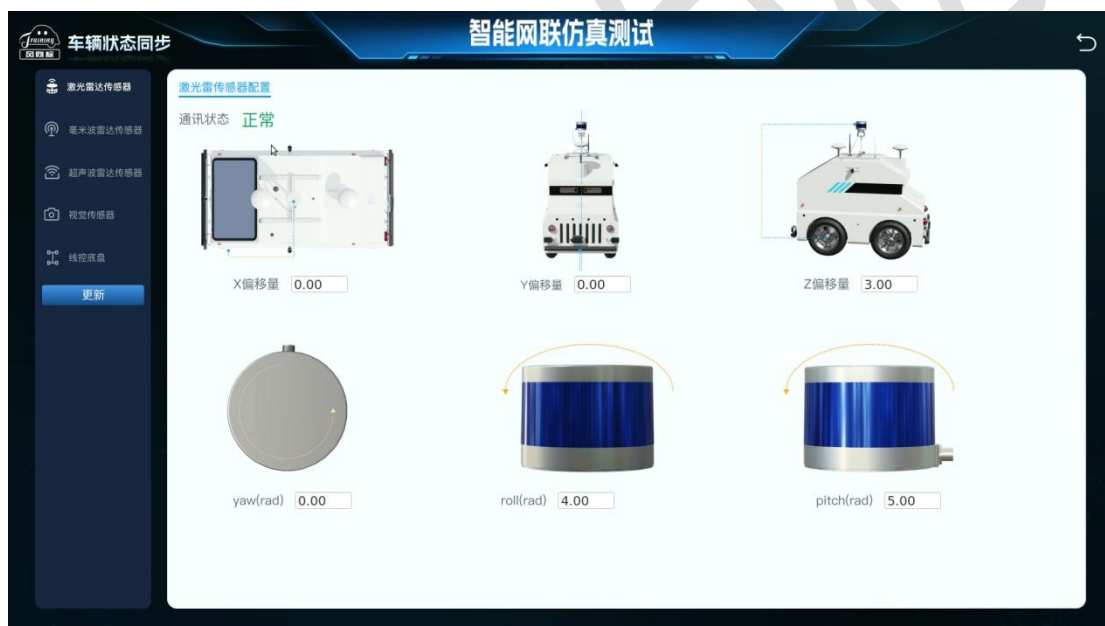
5.2 软件使用说明

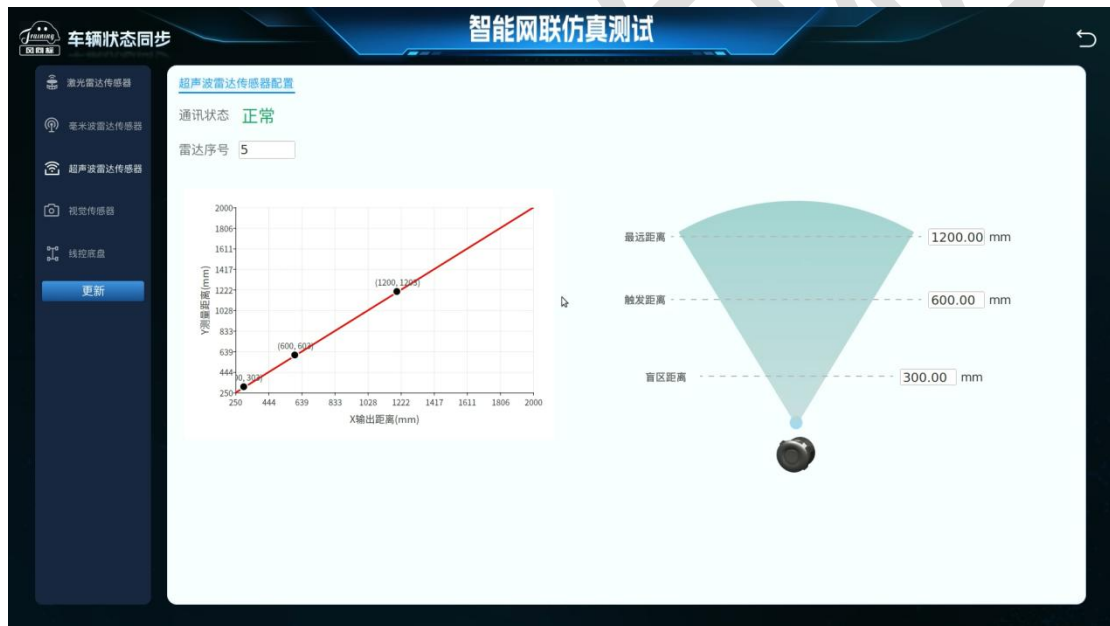
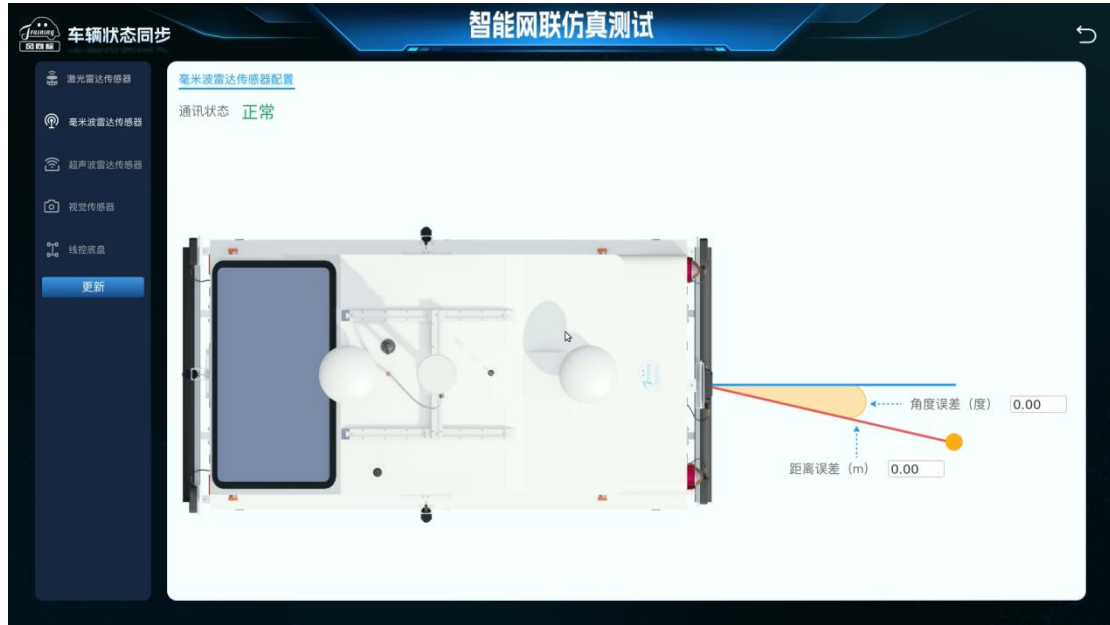
1) 输入账号，该账号为进入车辆的“装调标定软件”的账号，输入完成后点击登录，如果账号不存在会提示“用户数据不存在”，如果存在则会进入到功能页面。





2) 点击“车辆状态同步”，在这里会看到之前完成各种传感器标定的结果，以及该传感器在标定时状态，如果看到传感器异常，回去重新检测到正常后，需要点击更新按钮，以将车辆数据传输到仿真笔记本电脑上。







车辆状态同步

激光雷达传感器

毫米波雷达传感器

超声波雷达传感器

视觉传感器

线控底盘

更新

智能网联仿真测试

视觉传感器配置

通讯状态 正常

相机矩阵

畸变参数

标定前

标定后



车辆状态同步

激光雷达传感器

毫米波雷达传感器

超声波雷达传感器

视觉传感器

线控底盘

更新

智能网联仿真测试

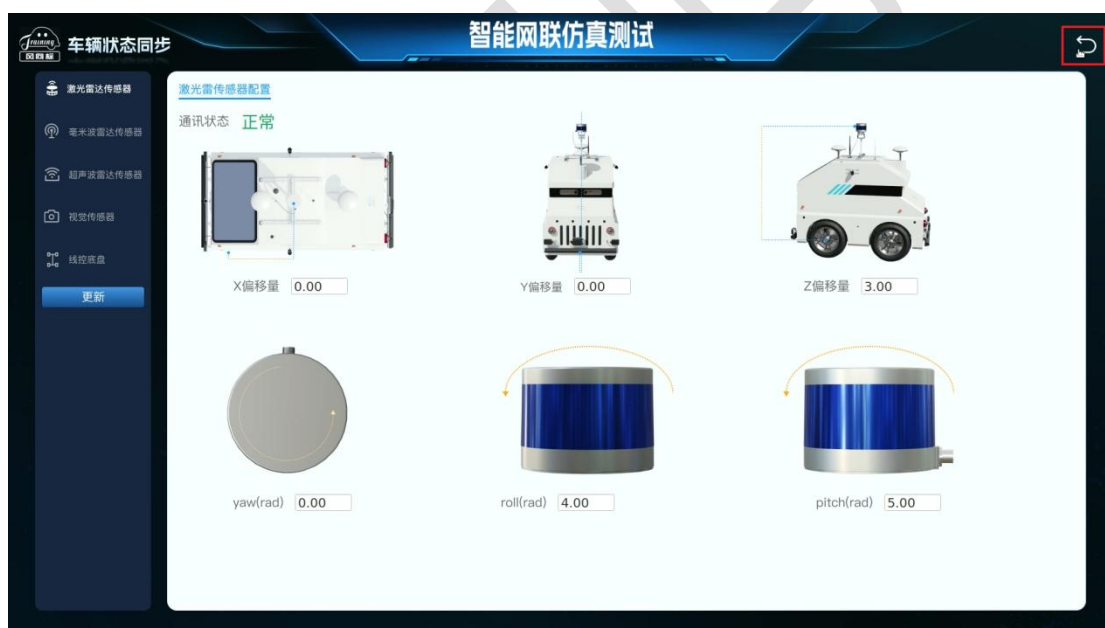
线控底盘

通讯状态 正常

转向中位标定结果 0.0°



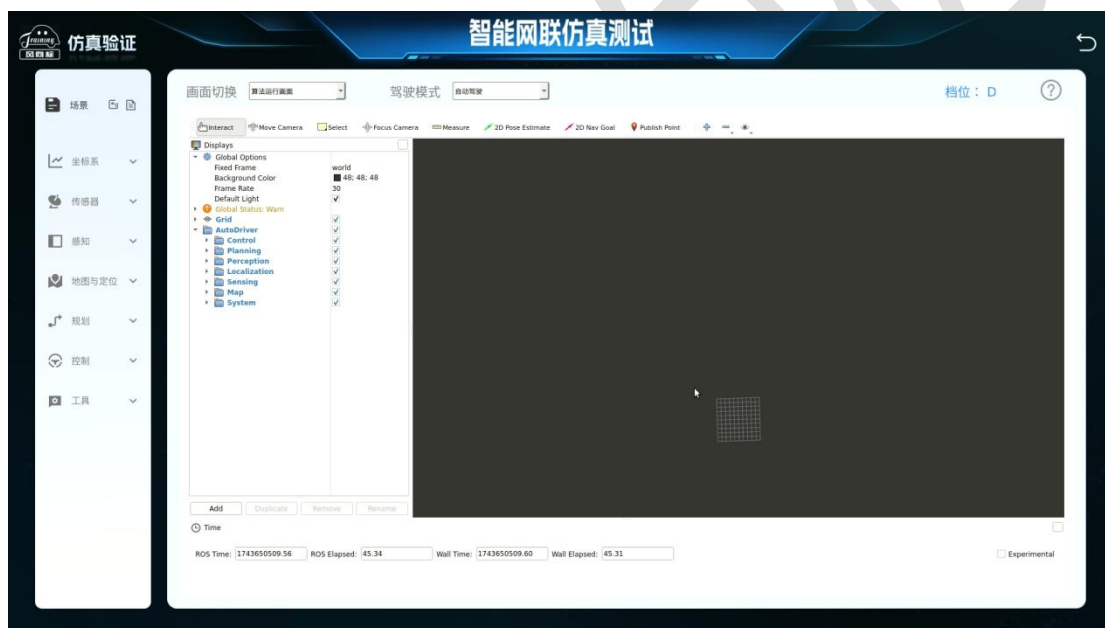
3) 当在“车辆状态同步”认为正常时，点击返回进入功能页面，点击“仿真验证”。



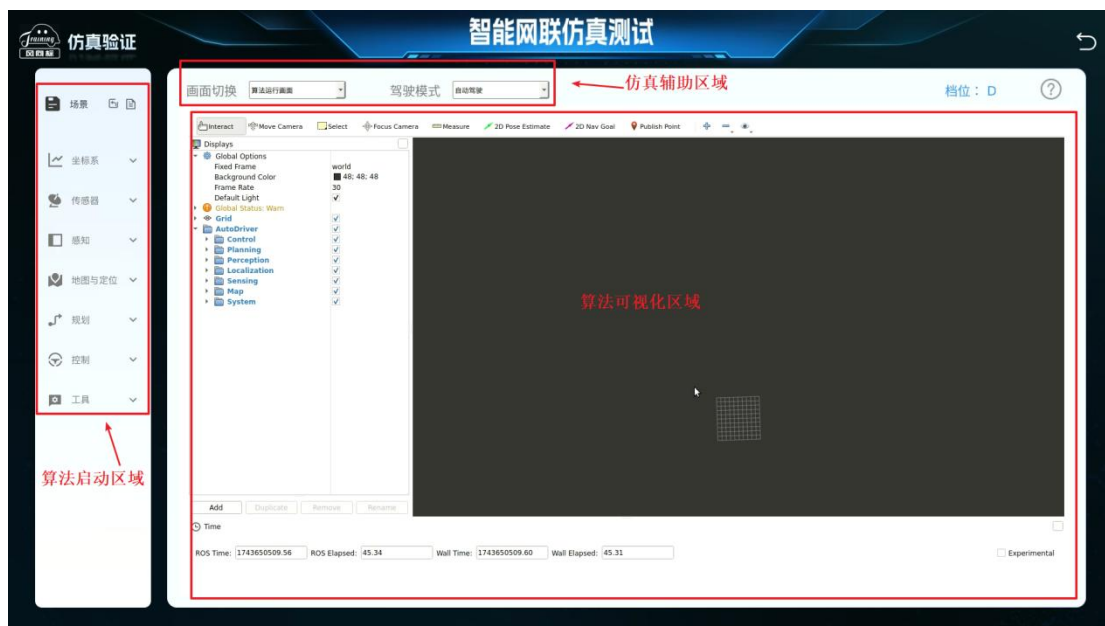


4) 选择任意一个仿真测试场景地图，等待进入仿真操作界面。



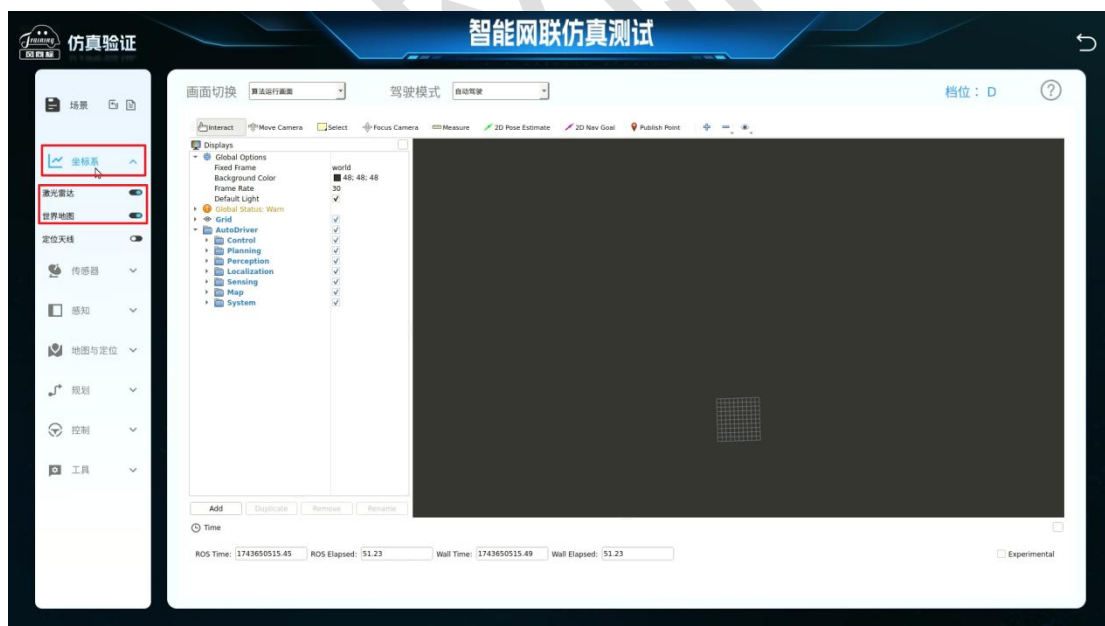


5) 仿真操作界面，由算法启动区域，算法可视化区域，仿真辅助区域，三大部分组成。

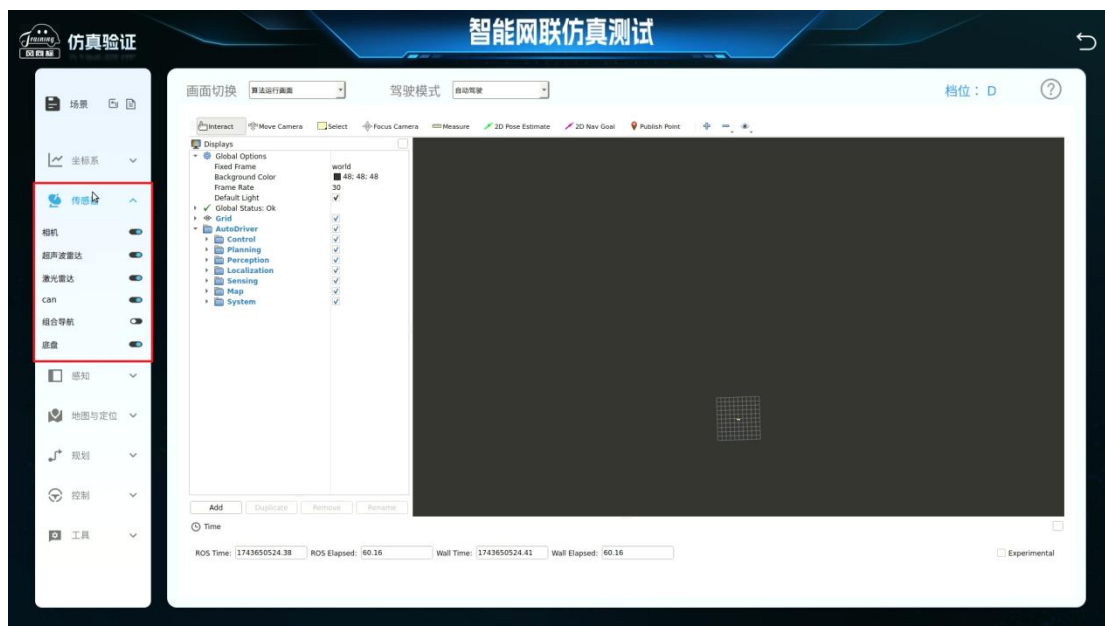


6) 按照实际车辆的自动驾驶启动流程（无需配置），即可利用仿真世界的的数据，测试基于调试标定的结果对应的自动驾驶运行效果，这样可以减少危险发生，也可以验证各种异常情况下的效果，下面是启动流程步骤说明。

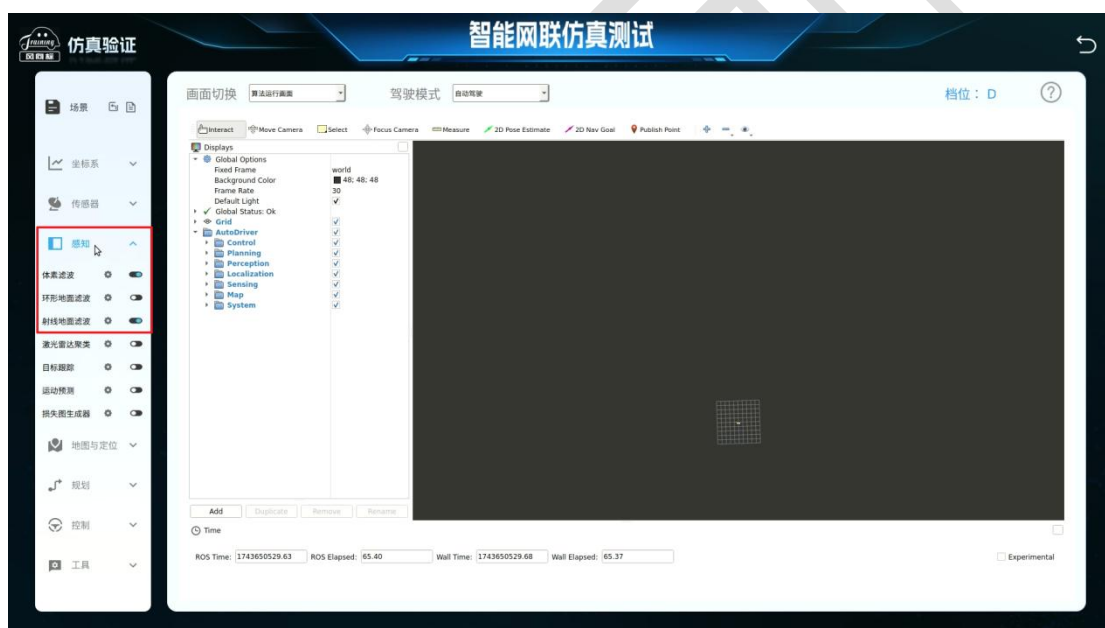
7) 找到“坐标系”下的“激光雷达”和“世界地图”并勾选启动。



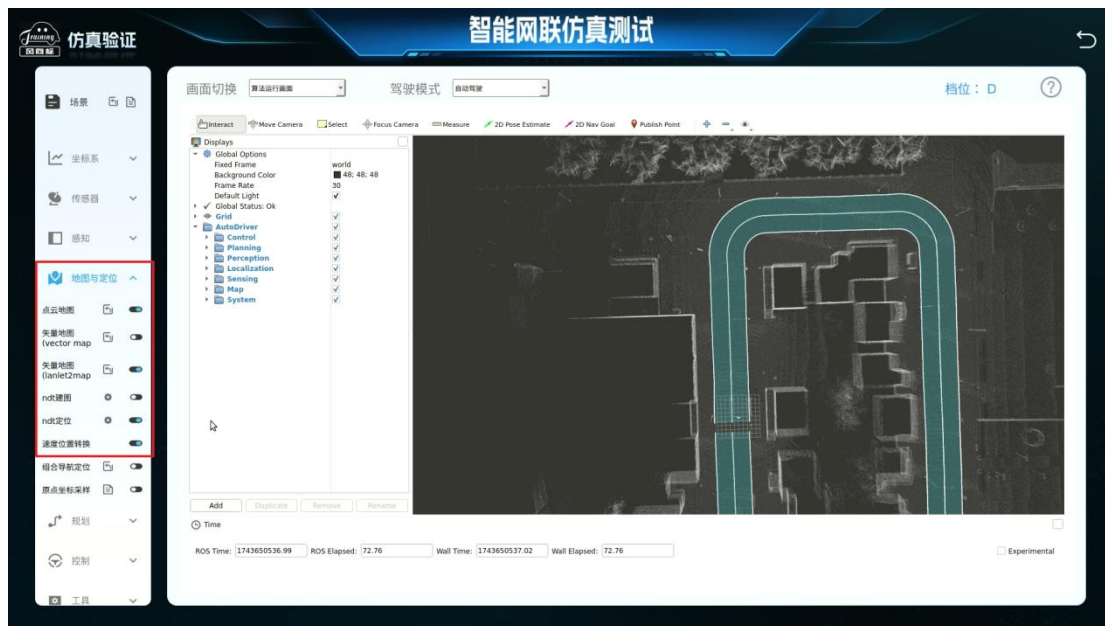
8) 找到“传感器”下的“相机”、“超声波雷达”、“激光雷达”、“can”和“底盘”并勾选启动。



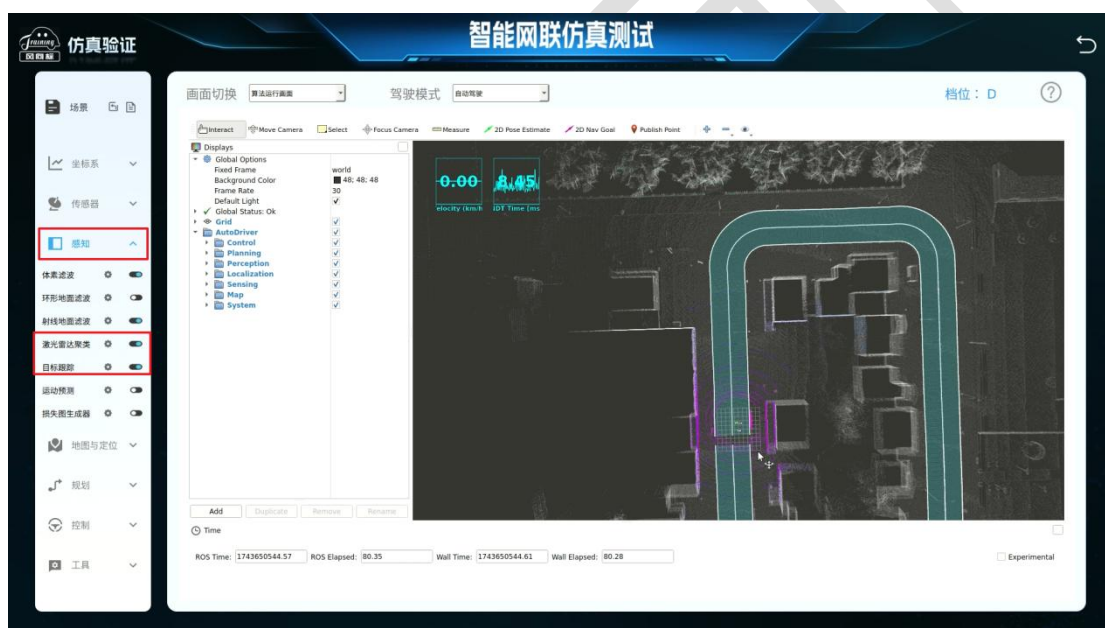
9) 找到“感知”下的“体素滤波”和“射线地面滤波”并勾选启动。



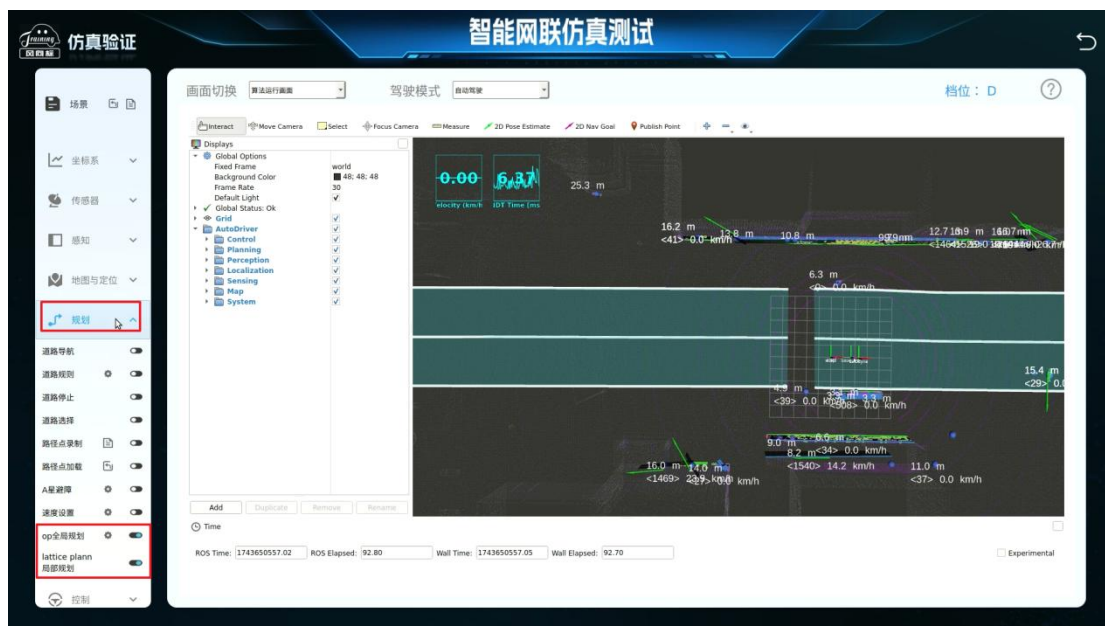
10) 找到“地图与定位”下“点云地图”、“矢量地图（lanelet2map）”、“ndt定位”、“速度位置转换”并勾选启动。



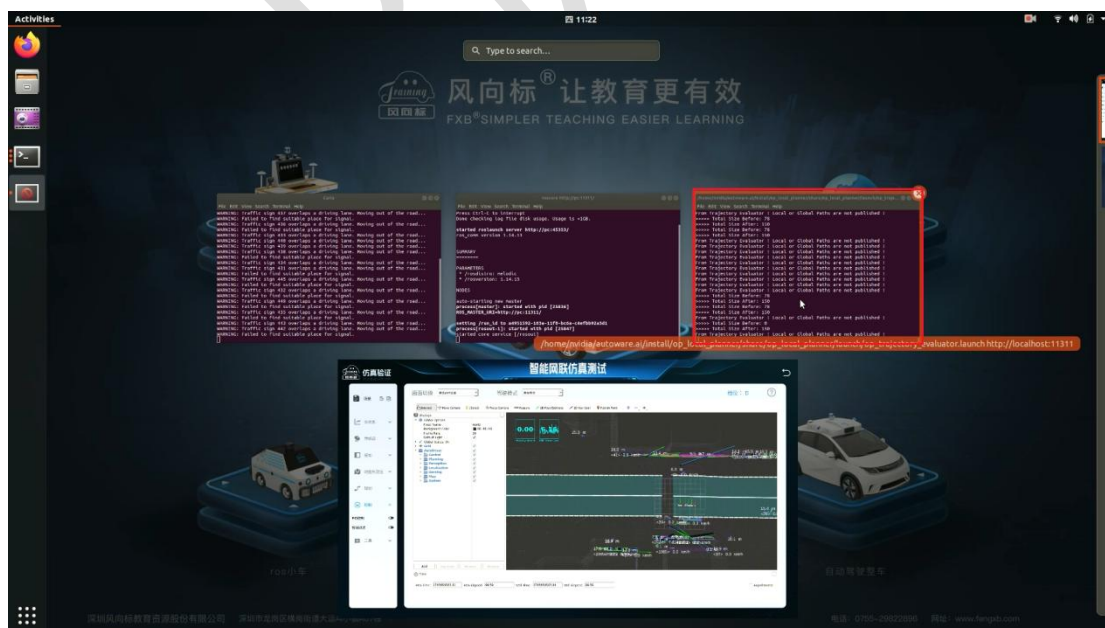
11) 找到“感知”下的“激光雷达聚类”和“目标跟踪”并勾选启动。

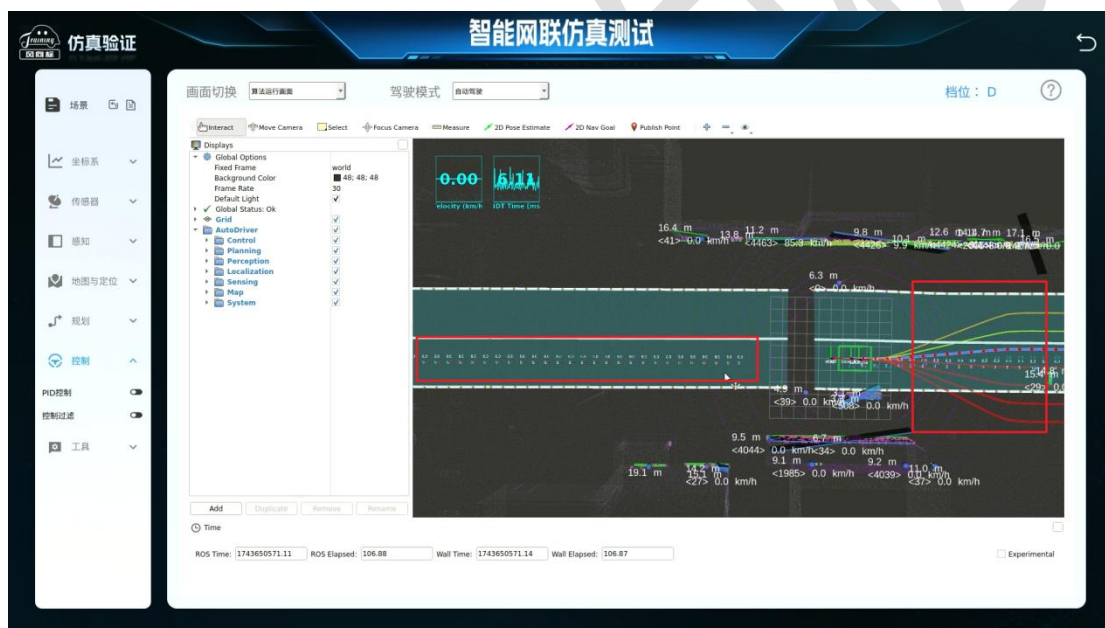
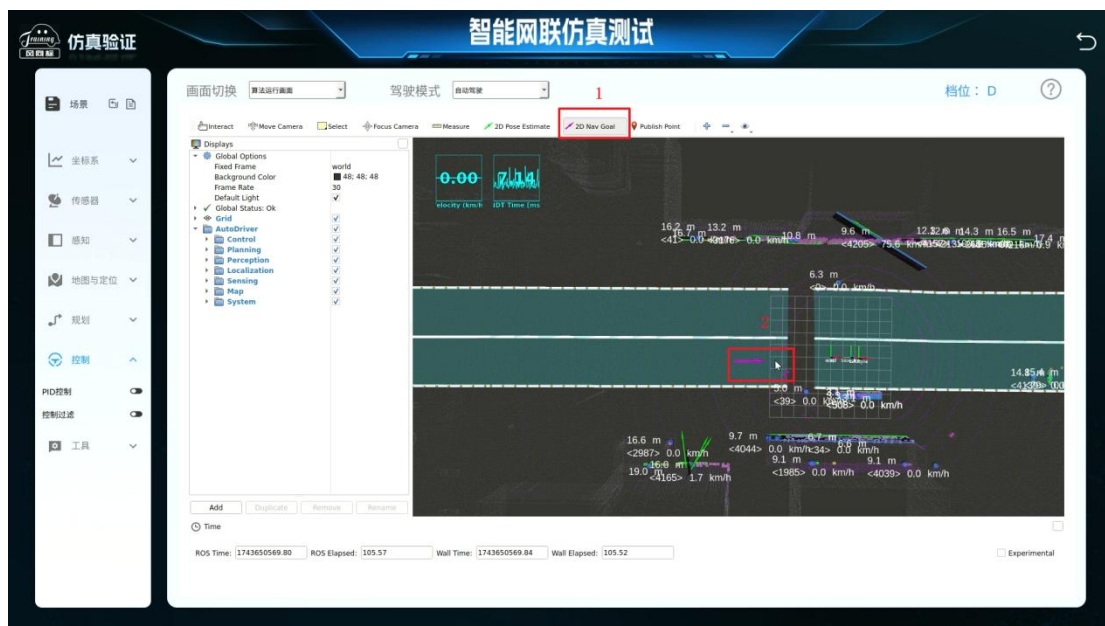


12) 找到“规划”下的“op 全局规划”和“lattice planner 局部规划”并勾选启动。

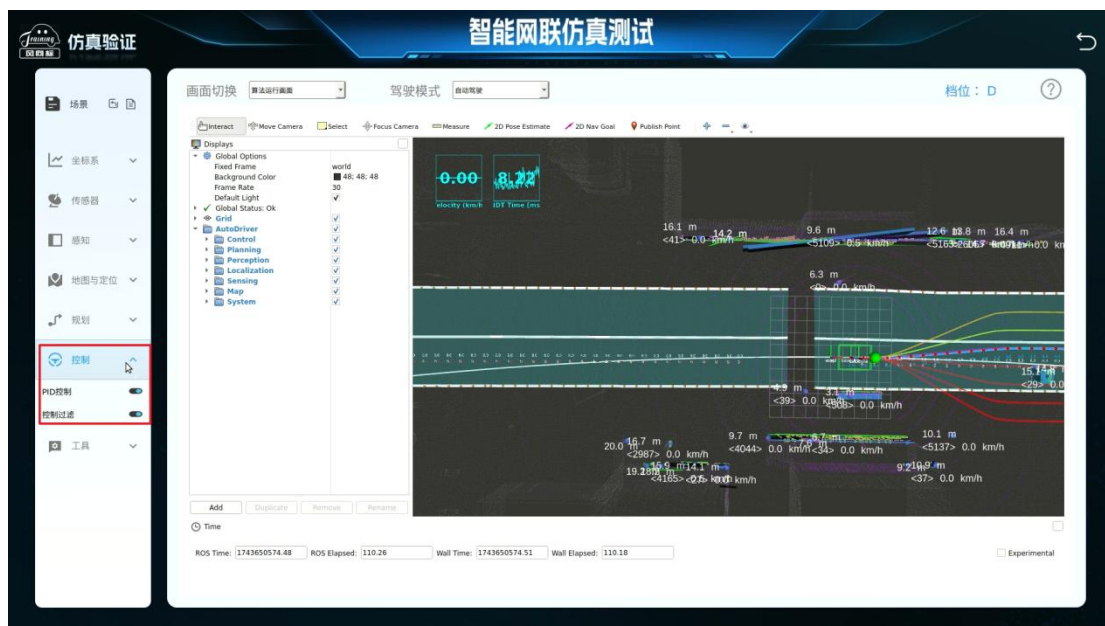


13) 按下键盘的 win 键切换到后台，当看到终端出现“Local or Global Paths are not published!”时，点击“2D Nav Goal”按钮，选择车辆到达终点（终点必须选择在道路结束位置，因为系统在统计分数时会检测车辆是否到达这个位置），选择完成后会出现多条彩色路线，注意！终点必须选择车辆当前位置的车道上，不可选到其他车道，如果选择终点后，没有生成多条彩色路线，则需要关闭“op 全局规划”和“lattice planner 局部规划”等待 5s 后，重新启动然后再选择终点。

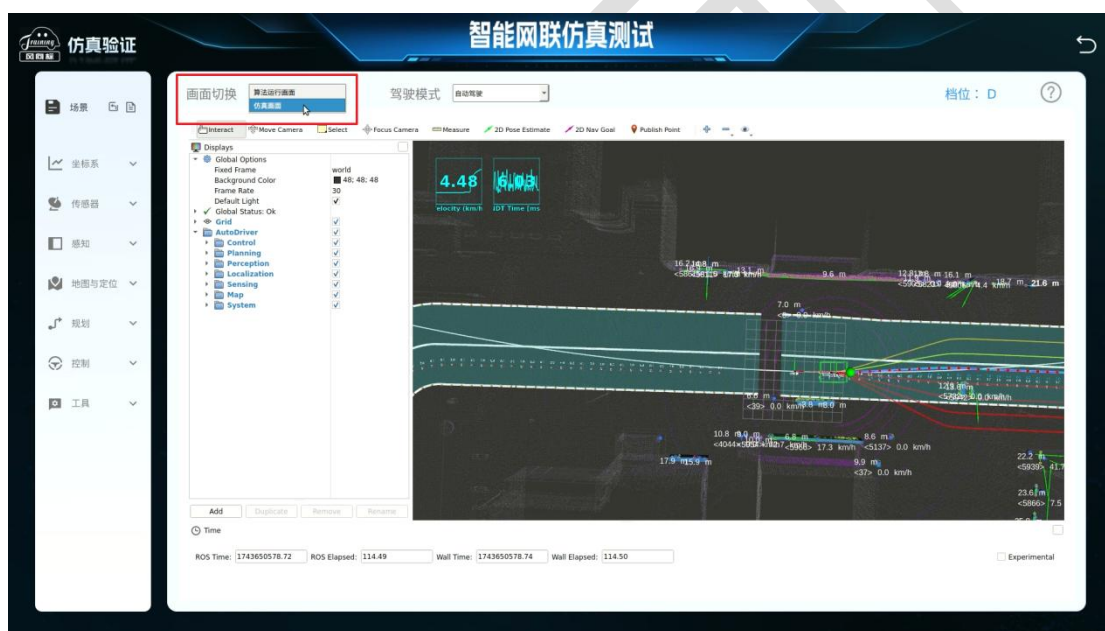




14) 找到“控制”下的“PID 控制”和“控制过滤”并勾选启动，很快车辆就会按照到达终点的路线自动行驶。



15) 找到“画面切换”将“算法运行画面”换为“仿真画面”。





16) 可以点击“第一人称”或“第三人称”切换视角。





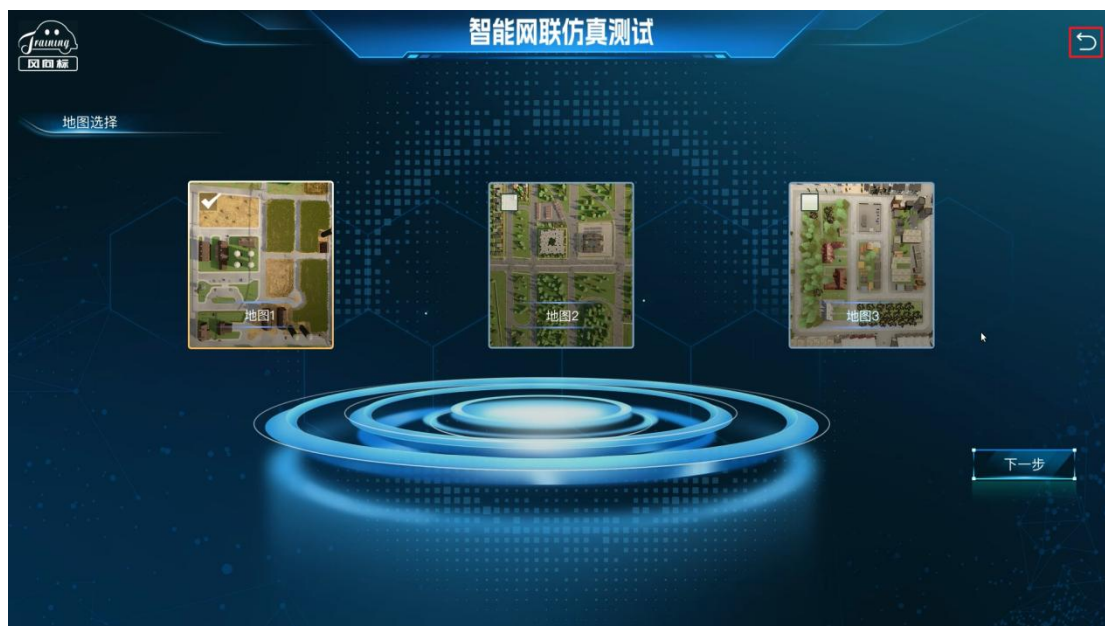
17) 当需要手动控制车辆时，找到“驾驶模式”，将“自动驾驶”换为“手动驾驶”，手动驾驶的控制键位可以将鼠标放到问号图标上会显示。





18) 当车辆到达终点或不能运行时，点击返回，在功能页面上点击“验证结果”，查看本轮测试分数。







验证结果

智能网联仿真测试

得分详情

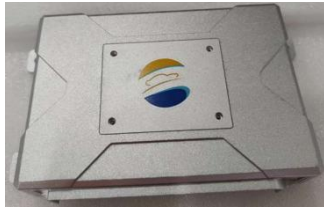
总得分: 100.0

评分项	子项总分
激光雷达传感器	100.0
毫米波雷达传感器	100.0
超声波雷达传感器	100.0
视觉传感器	100.0
线控底盘	100.0
是否红绿灯启停	100.0
是否到达终点	100.0
是否发生碰撞	100.0
是否发生压线	100.0

EXB 风向标

第 6 章 V2X 车路协同 OBU 端

6.1 部件描述



OBU 主机



5G 天线



V2X 天线



GNSS 天线



WIFI 天线



主线束



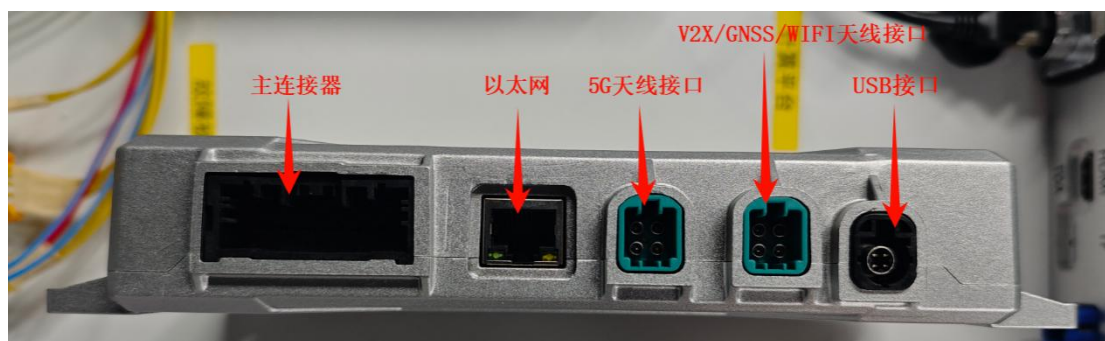
以太网线束



MINI FAKRA

- 1) 车载设备的电气连接包括天线连接（5G 天线、V2X 天线、GNSS 天线、WIFI 天线）、主线束连接、以太网连接。

6.2 设备接口描述



6.3 OBU 调试

6.3.1 连接 OBU

- 1) 连接 OBU 我们使用 ssh 命令进行连接，使用快捷键 `ctrl+alt+t` 打开终端，输入"`ssh root@192.168.1.214`"命令连接 OBU 设备，回车后会提示输入密码，密码为"`crv654321`"，输入完密码后回车即可看见如下界面说明连接 OBU 成功。

```
nvidia@oceanstar: ~
File Edit View Search Terminal Help
nvidia@oceanstar:~$ ssh root@192.168.1.214
root@192.168.1.214's password:
Last login: Sat Sep 27 10:02:13 2025 from 192.168.1.102
root@vrc_imx8qxp:~#
```

6.3.2 配置 OBU

- 1) 在连接上 OBU 的终端内输入“vi /home/root/appr_v2x/config/visibleRealtimeCommon.conf”命令打开配置文件，在打开的界面中输入“:506”（按住 shift + : 进行输入，输入内容会在左下角显示）按回车后进行跳转到配置行中，可以看到刚开默认“value”值是空的，这时候我们就需要配置对应路端设备的 SN 号，例如路端设备的 SN 号是 ENR22516180D，那就填写为：value:22516180（只需要填写数字部分即可）。

```
{
  "description": "OBU匹配的特定RSU设备ID配置",
  "items": [
    {
      "description": "RSU设备的SN号，如ENR22516180D, value:2",
      "name": "rsuId",
      "remark": "上位机采集数据使用",
      "type": 4,
      "value": ""
    },
    {
      "name": "RSUESN",
      "type": 5
    }
  ]
}
```

- /home/root/appr_v2x/config/visibleRealtimeCommon.conf 506/514 98%

vi 命令输入说明：1、键盘上下左右按键是控制光标的位置；2、在编辑前需要先按键盘 i 按键进入编辑模式才可以输入内容；3、编辑完成后先按 esc 按钮退出编辑模式，然后输入“:wq”指令后回车保存退出。

注意：当存在多台路测单元发出相同消息时，才需要进行配置接收单台路测单元设备。

6.3.3 启动 V2X 程序

- 1) 在连接上 OBU 的终端内输入“systemctl start start_obu_all.service”命令来启动 V2X 程序，如下图所示：

```
nvidia@oceanstar: ~  
File Edit View Search Terminal Help  
root@vrc_imx8qxp:~# systemctl start start_obu_all.service  
root@vrc_imx8qxp:~#
```

- 2) 检查程序是否启动成功：在连接上 OBU 的终端内输入“systemctl status start_obu_all.service”命令查看 V2X 程序服务是否启动成功，输出信息中存在“active (running)”并且进程有./V2xApps、obu_pub_v2xMsg.py 字样说明服务正常启动，如下图所示：

```
nvidia@oceanstar: ~  
File Edit View Search Terminal Help  
[0;1;32m[0m start_obu_all.service - start_rsu_all  
Loaded: loaded (/etc/systemd/system/start_obu_all.service; enabled; vendor pr  
set: enabled)  
Active: [0;1;32mactive (running)[0m since Tue 2022-07-12 17:14:21 CST; 3 ye  
ars 2 months ago  
Main PID: 3394 (start_obu_all.s)  
Tasks: 22 (limit: 2085)  
CGroup: /system.slice/start_obu_all.service  
├─3394 /bin/sh /home/root/start_obu_all.sh  
├─3587 ./V2xApps  
├─3686 ./JsonToPb  
└─3788 python3 ./obu_pub_v2x_msg.py
```

- 3) OBU 程序其他命令详解：

systemctl start start_obu_all.service: 启动 V2X 程序服务

systemctl stop start_obu_all.service: 停止 V2X 程序服务

systemctl status start_obu_all.service: 查看 V2X 程序状态服务

systemctl restart start_obu_all.service: 重启 V2X 程序服务

注意：手动修改了“config”文件夹中的文件需要重启 V2X 程序才会生效！

6.3.4 检查 GPS 信号

- 1) 在连接上 OBU 的终端内输入“gpsmon”命令查看 OBU 是否正常接收到 GPS 信号，输出信息内“Quality”的值大于或者等于 1，说明 OBU 正常接收 GPS 信号，如下如所示：

6.3.6 检查 OBU 与路测设备通讯

- 1) 在连接上 OBU 的终端内输入“tail -n 0 -F /home/root/appr_v2x/logs/v2x.log | grep -a MAP”命令来检查 OBU 端是否接收到路测单元所发出的“MAP”消息，如有消息输出说明接收正常，如下图所示：

```
root@vrc_imx8qxp:~# tail -n 0 -F /home/root/appr_v2x/logs/v2x.log | grep -a MAP
[09/27/25][14:25:08:718344 +08:00] [D] [3679][VMLS][recvRemoteMap][968]recvRemot
eMap:{"MAP":{"msgCnt":30,"nodes":[{"id":{"id":2,"region":0},"inLinks":[{"lanes":
[{"connectsTo":[{"connectingLane":{"lane":2,"maneuver":3},"phaseId":1,"remoteInt
ersection":{"id":3,"region":0}},{"connectingLane":{"lane":7,"maneuver":3},"phase
Id":1,"remoteIntersection":{"id":5,"region":0}},{"laneID":1,"lanewidth":3.5,"poi
nts":[{"lat":22.7250785,"long":114.2186453},{"lat":22.7238178,"long":114.2186834
}], "speedLimits":[{"speed":8.34,"type":5}]}],"linkWidth":3.5,"name":"node1_node2
_links","upstreamNodeId":{"id":1,"region":0}},{"lanes":[{"connectsTo":[{"connect
ingLane":{"lane":2,"maneuver":4},"phaseId":2,"remoteIntersection":{"id":3,"regio
n":0}},{"connectingLane":{"lane":1,"maneuver":4},"phaseId":2,"remoteIntersection
":{"id":1,"region":0}},{"laneID":1,"lanewidth":3.5,"points":[{"lat":22.7238383,"
long":114.2231535},{"lat":22.723784,"long":114.2188046}], "speedLimits":[{"speed
":8.34,"type":5}]}],"linkWidth":3.5,"name":"node5_node2_links","upstreamNodeId":{"
id":5,"region":0}},{"lanes":[{"connectsTo":[{"connectingLane":{"lane":1,"maneuv
er":1},"phaseId":1,"remoteIntersection":{"id":1,"region":0}},{"connectingLane":{"
lane":7,"maneuver":4},"phaseId":1,"remoteIntersection":{"id":5,"region":0}},{"l
```

注意：如果存在多台路测单元发出相同消息时，可以过滤查看，使用以下命令“tail -n 0 -F /home/root/appr_v2x/logs/v2x.log | grep -a MAP | grep -a "name": "22083011" ”；22083011 是路测单元的 SN 码。

- 2) 使用相同的方法进行检查“RSI”、“SPAT”消息接收是否正常，如下图所示：

```
root@vrc_imx8qxp:~# tail -n 0 -F /home/root/appr_v2x/logs/v2x.log | grep -a RSI
[09/27/25][14:29:11:952334 +08:00] [D] [3679][VMLS][recvRemoteRsi][1002]recvRemo
teRsi:{"RSI":{"id":"3232303833303131","msgCnt":33,"refPos":{"elevation":30.0,"la
t":22.7221708,"long":114.2228023},"rtcs":[{"description":"shigutixing","eventCon
fidence":0,"eventPos":{"elevation":0.0,"lat":0.0,"long":0.0},"eventSource":0,"ev
entType":0,"priority":0,"referencePaths":[{"activePath":[{"lat":0.0,"long":0.0},
{"lat":0.0,"long":0.0}], "pathRadius":0.0}], "rteId":1},"rtss":[{"description":"j
izhuanwan","priority":63,"referencePaths":[{"activePath":[{"lat":22.7221757,"lon
g":114.223366}], "lat":22.7221817,"long":114.2240425}], "pathRadius":6.6}], "rtsId"
:1,"signPos":{"elevation":30.0,"lat":22.7221787,"long":114.2237042},"signType":2
}, {"description":"zhixing","priority":63,"referencePaths":[{"activePath":[{"lat"
:22.7230346,"long":114.2243182}], "lat":22.7238,"long":114.2243091}], "pathRadius"
:6.6}], "rtsId":1,"signPos":{"elevation":30.0,"lat":22.7235056,"long":114.2243126
}, {"signType":96}, {"description":"renxinghengdao","priority":63,"referencePaths":
[{"activePath":[{"lat":22.7238071,"long":114.2206529}, {"lat":22.7237908,"long":1
14.2193482}], "pathRadius":6.6}], "rtsId":1,"signPos":{"elevation":30.0,"lat":22.7
237989,"long":114.2200005},"signType":114}], "utcTime":1751435460.0}
^C
```



```
root@vrc_lmx8qxp:~# tail -n 0 -F /home/root/appr_v2x/logs/v2x.log | grep -a SPAT
[09/27/25][14:30:21:833235 +08:00] [D] [3679][VMLS][recvRemoteSpat][990]recvRemo
teSpat:{"SPAT":{"intersections":[{"intersectionId":{"id":2,"region":0},"phases":
[{"id":1,"phaseStates":[{"light":3,"timing":{"counting":{"likelyEndTime":8.0,"ne
xtDuration":50.0,"nextStartTime":31.0,"startTime":0.0}}},{light":5,"timing":{"c
ounting":{"likelyEndTime":16.0,"nextDuration":8.0,"nextStartTime":81.0,"startTim
e":8.0}}},{light":7,"timing":{"counting":{"likelyEndTime":31.0,"nextDuration":1
5.0,"nextStartTime":89.0,"startTime":16.0}}}]}}],"status":32},"msgCnt":33,"name"
:"22083011","utcTime":1758954621.825}}
^C
```

注意：存在多台路测单元发出相同消息时，过滤查看是需要注意的
是“MAP”，“SPAT”消息时使用 SN 码进行过滤，“RSI”消息是使用 ID 进行过滤的。

6.4 OBU 与 PAD 通讯

6.4.1 PAD 连接 OBU

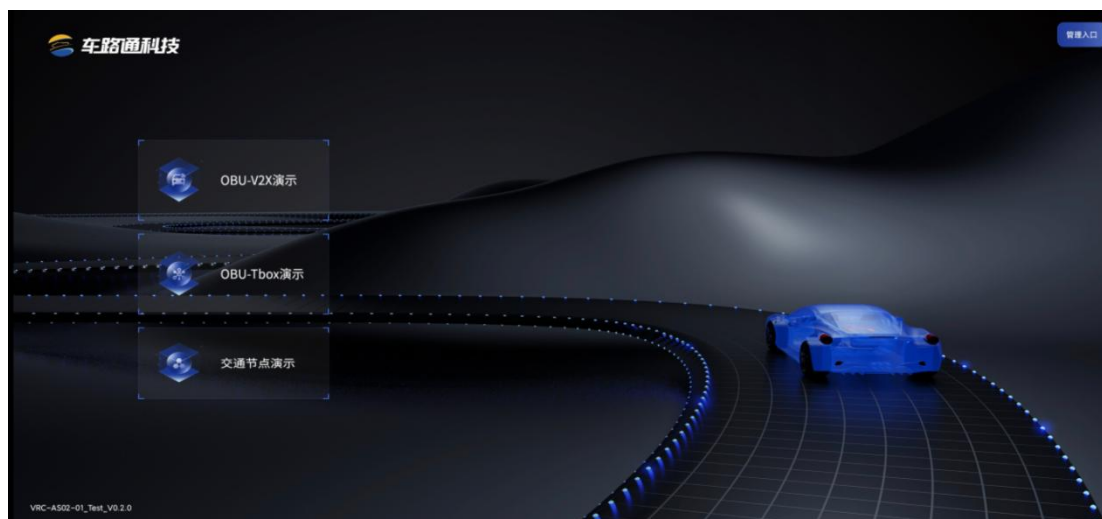
- 1) 打开 PAD，下拉状态栏长按 WIFI 图标打开 WIFI 设置界面，找到对应车辆的
OBU 发出的 WIFI，点击进行连接（WIFI 密码为：crvpublic），如下图所示连
接成功：



- 2) 然后在 PAD 桌面上找到如下图所示的 APP，点击进行打开



- 3) 进入 APP 后的主界面：



6.4.2 PAD 配置

- 1) 首次使用 APP 时需要设置 OBU 的 IP 地址, 如果不设置 APP 无法与 OBU 进行通讯; 首先打开 APP, 点击右上角的“管理入口”, 在弹出的登录界面中直接点击登录, 进入设置界面。



- 2) 点击登录按钮后进入的界面, 在此点击进入“OBU 管理”



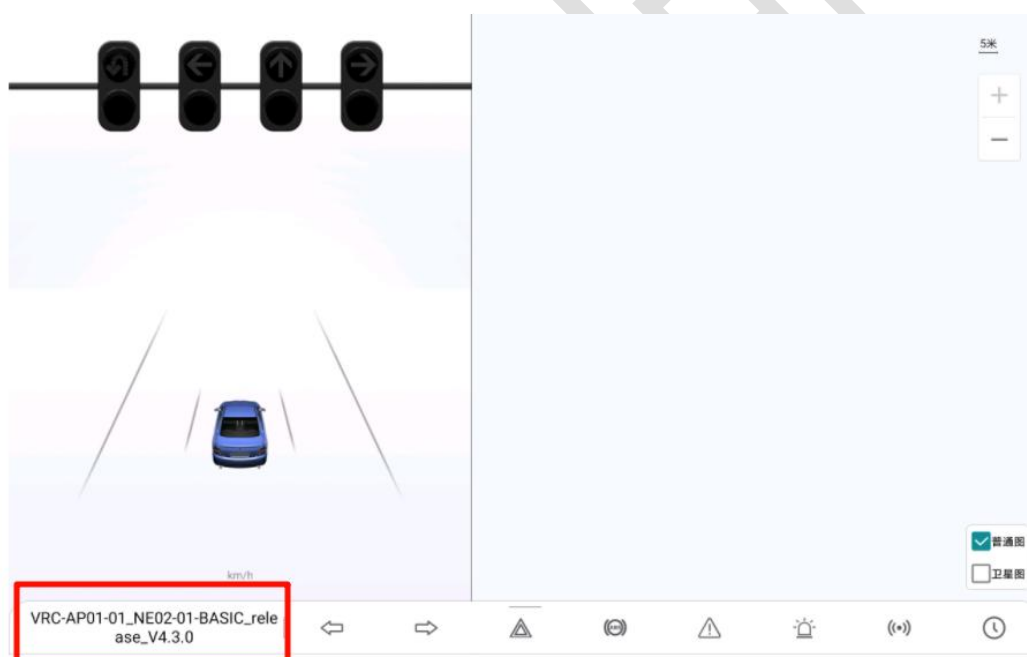
- 3) 在“OBU 管理”界面中，首先在左侧栏中选择“设置”，然后在右边出现的配置中的第一项 IP 地址中输入 OBU 的 IP 地址，默认 OBU 的 WIFI IP 地址是：192.168.110.1，设置完成之后，点击顶部的保存按钮将配置进行保持



- 4) 点击保存后沿着屏幕左侧边缘向右滑动可以返回，返回到主界面后点击“OBU-V2X 演示”，如下图所示：



- 5) 进入“OBU-V2X 演示”界面后，查看左下角是否显示出版本信息，如有说明已经与 OBU 连接成功，如下图所示：



6.4.3 检查 OBU 与 PAD 的通讯

- 1) PAD 连接上 OBU 后需要检查 OBU 与 PAD 之间的通讯是否正常；检查 OBU 与 PAD 之间的通讯是否正常可以向 OBU 端发送一个固定的虚拟定位信息来查看通讯是否正常，首先在车辆上打终端输入“roscore”，如下图所示：



```
roscore http://Ubuntu-18:11311/
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
nvidia@Ubuntu-18:~$ roscore
... logging to /home/nvidia/.ros/log/dfb85992-a4f3-11f0-a22d-000c29168dea/roslaunch-Ubuntu-18-3762.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://Ubuntu-18:40121/
ros_comm version 1.14.13

SUMMARY
=====

PARAMETERS
* /rostdistro: melodic
* /rosversion: 1.14.13

NODES

auto-starting new master
process[master]: started with pid [3773]
ROS_MASTER_URI=http://Ubuntu-18:11311/

setting /run_id to dfb85992-a4f3-11f0-a22d-000c29168dea
process[rosout-1]: started with pid [3784]
```

注意：如果已经打开了自动驾驶实训软件就不需要再启动 roscore 了！

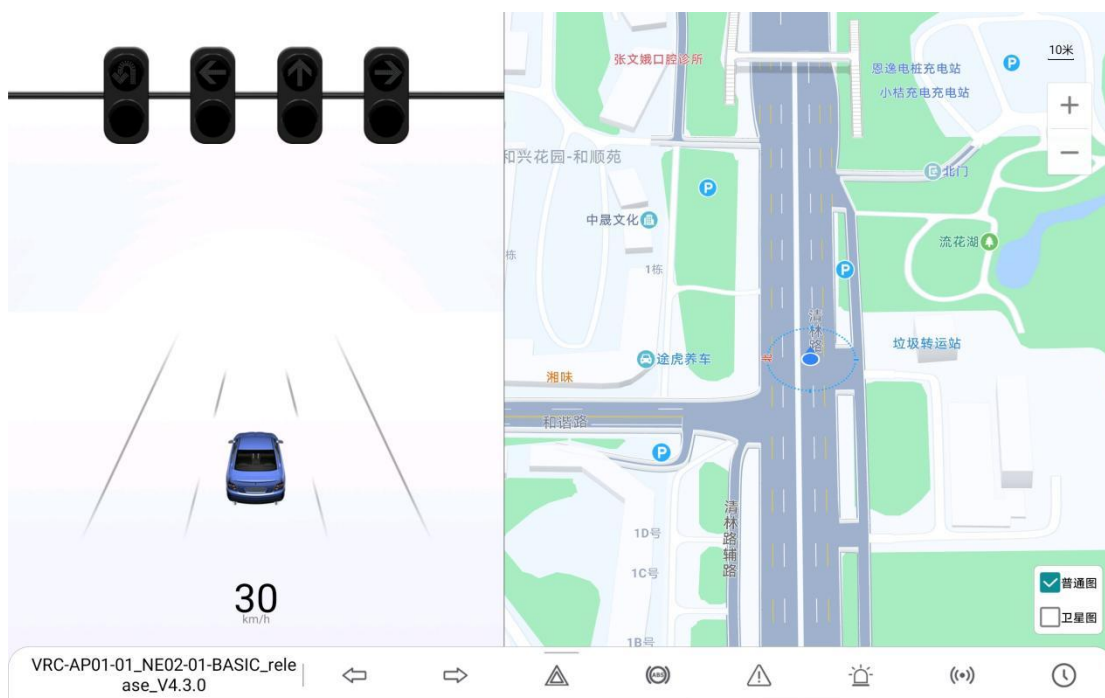
- 2) 然后再打开一个终端输入“cd ~/yutu_v2x_ws”进入存放程序的目录下，再输入“source install/setup.bash”来刷新系统环境，如下图所示：

```
终端
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
nvidia@Ubuntu-18:~$ cd ~/yutu_v2x_ws
nvidia@Ubuntu-18:yutu_v2x_ws$ source install/setup.bash
nvidia@Ubuntu-18:yutu_v2x_ws$ |
```

- 3) 环境准备好后启动发送虚拟定位的程序，继续在终端中输入“roslaunch start_function obu_gps_test.py”启动后如下图所示说明程序已经启动成功。

```
终端
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
nvidia@Ubuntu-18:yutu_v2x_ws$ cd ~/yutu_v2x_ws/
nvidia@Ubuntu-18:yutu_v2x_ws$ source install/setup.bash
nvidia@Ubuntu-18:yutu_v2x_ws$ roslaunch start_function obu_gps_test.py
[INFO] [1759996210.091540]: OBU_GPS_Test node start.
[INFO] [1759996210.092309]: The virtual positioning information is being sent to the OBU.
```

- 4) 此时查看 PAD 上的地图会自动定位在已经位置上，并且会有速度信息显示（如下图所示），滑动右侧的地图会自动回到定位的位置上，说明车辆发送的虚拟位置，车辆到 OBU 再到 PAD 上显示，这个通讯链路是正常的。

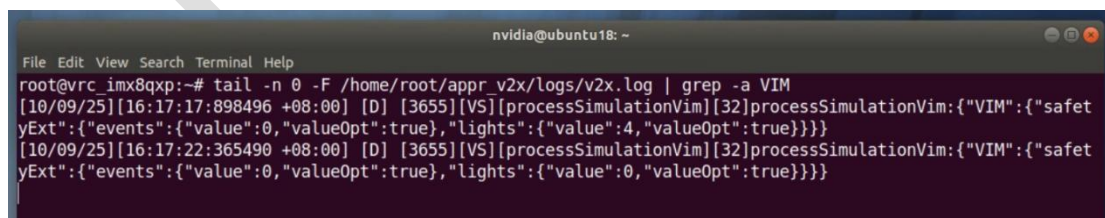


6.4.4 检查 PAD 与 OBU 的通讯

- 1) PAD 连接上 OBU 后需要检查 PAD 与 OBU 之间的通讯是否正常；首先 PAD 需要连接 OBU 的 WIFI，打开软件进入到主界面上，在车辆上打开终端使用 SSH 方式进入 OBU；在连接 OBU 的终端上输入“`tail -n 0 -F /home/root/appr_v2x/logs/v2x.log | grep -a VIM`”。
- 2) 此时可以在 PAD 上按下虚拟状态按键，按下后会向 OBU 端发送对应数据。



- 3) 例如这里第一次打开了“左转向灯”然后再关闭了“左转向灯”，如下图 OBU 的终端中就会输出：



在输出的信息中，主要查看的是 `"events":{"value":0,"valueOpt":true},"lights":{"value":0,"valueOpt":true}` 这两个 JSON 数据内的“value”值，例如这里打开转向的数据是 `"events":{"value":0,"valueOpt":true},"lights":{"value":4,"valueOpt":true}`



4) 各个虚拟按钮输出的值对应参考如下:

"safetyExt":{"events":0,"lights":0 无状态

"safetyExt":{"events":0,"lights":4 左转向

"safetyExt":{"events":0,"lights":8 右转向

"safetyExt":{"events":1,"lights":16 双闪

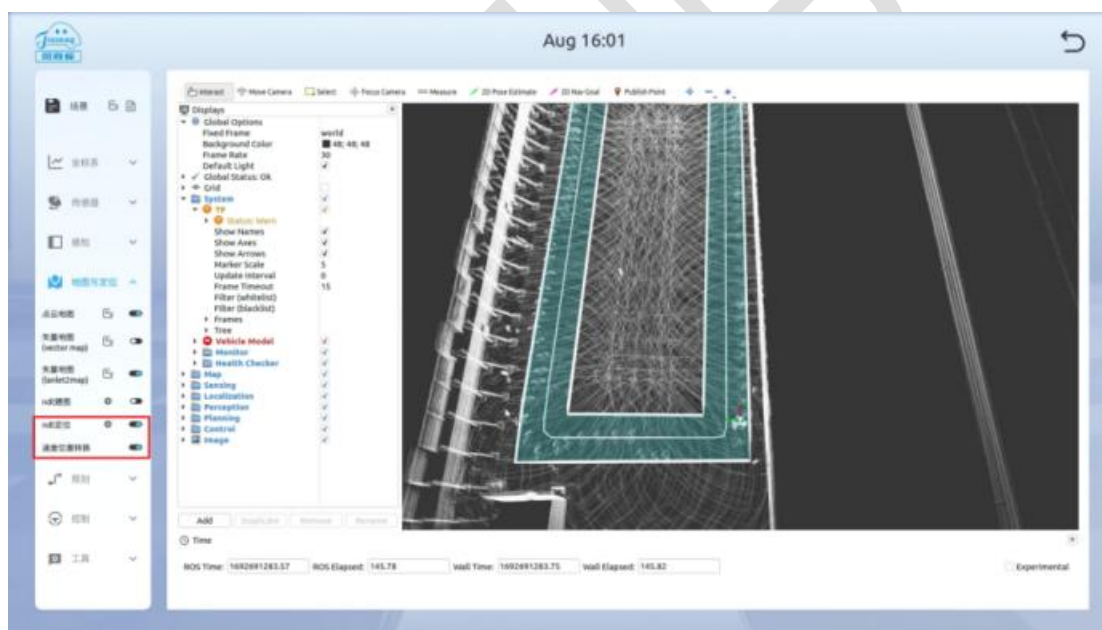
"safetyExt":{"events":128,"lights":0 紧急制动

"safetyExt":{"events":4,"lights":0 车辆失控

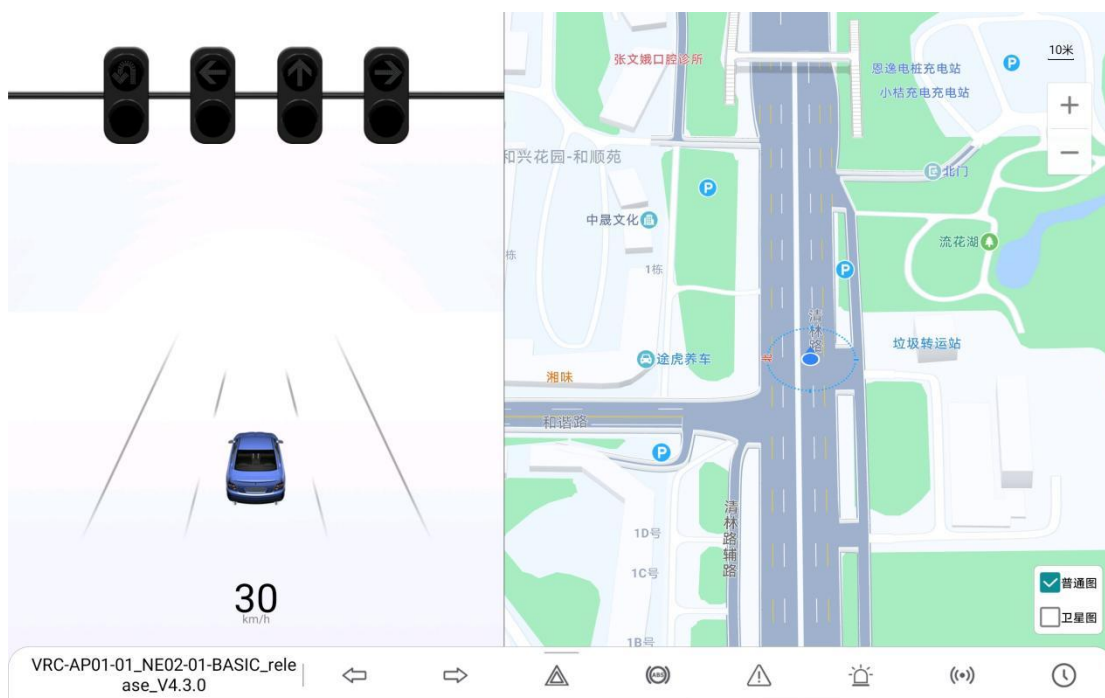
6.5 V2X 预警事件触发

6.5.1 准备工作

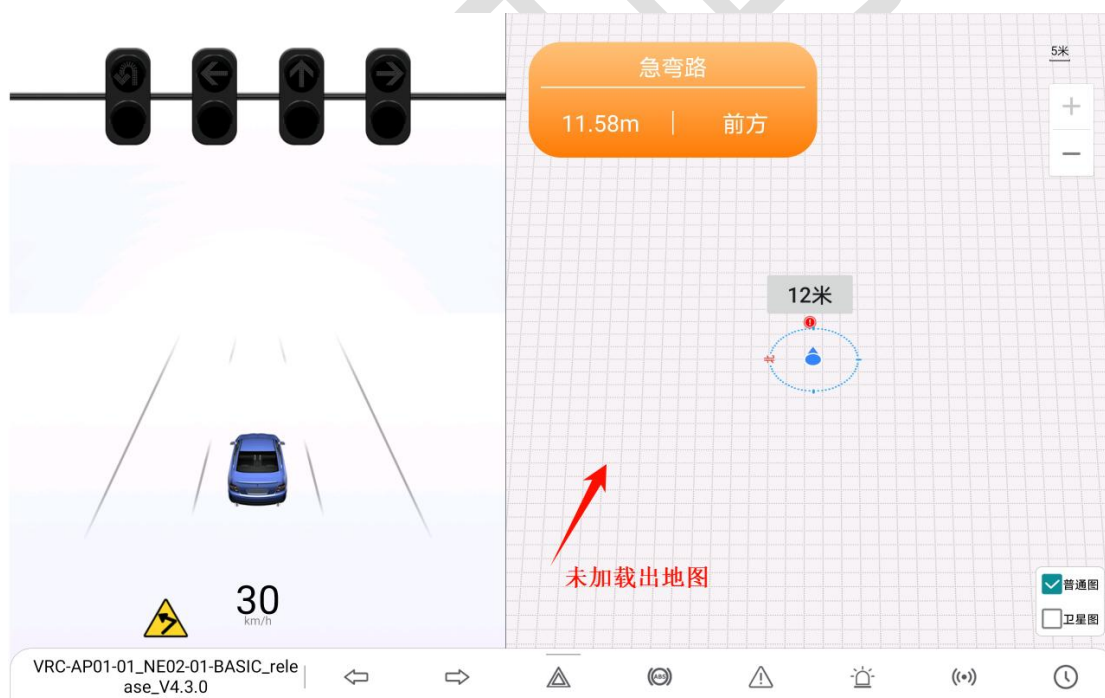
1) 打开自动驾驶软件中的高级教学，启动定位导航，保证功能运行正常



2) PAD 连接 OBU，打开 APP 确保 PAD 能与 OBU 进行通讯，界面上显示速度信息和定位信息，如下图所示:

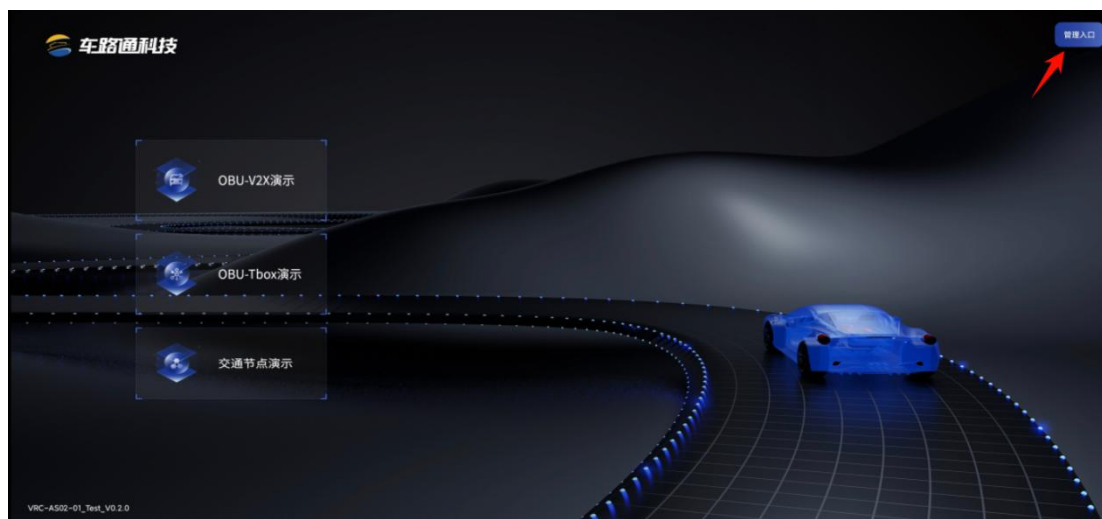


注意：PAD 连接上 OBU 后进入“OBU-V2X 演示”主界面中可能会加载离线地图失败导致没有地图底图只显示了方格子如下图所示：



这个时候需要我们手动的进行加载离线地图显示，如下步骤：

1. 回到主界面中点击右上角的“管理入口”，如下图所示：

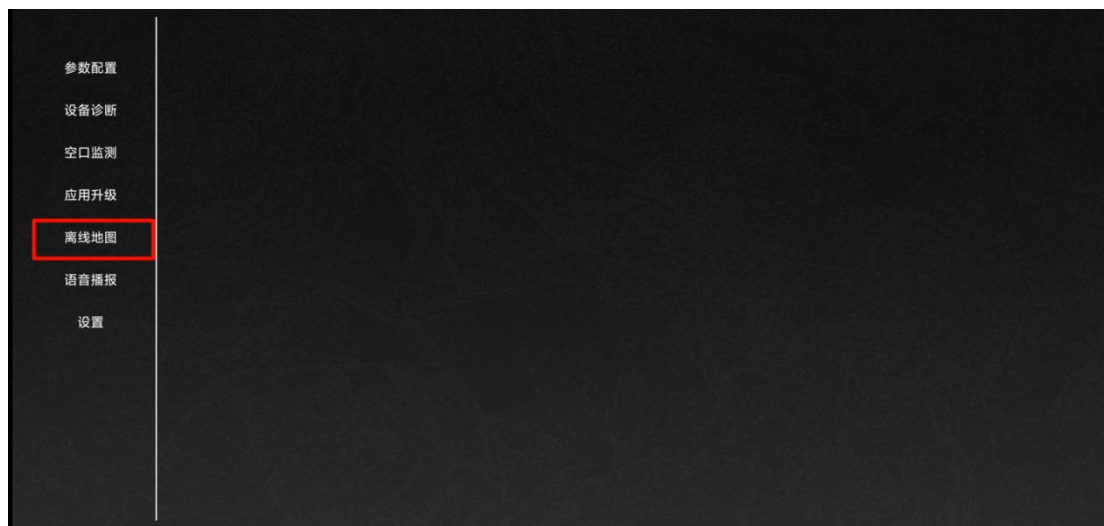


2. 在弹出的“登录窗口”中直接点击“登录”，如下图所示：



3. 进入管理界面中找到“OBU 管理”，点击进入；然后找到“离线地图”点击进入，如下图所示：





已下载:--

北京

开始

参数配置

设备诊断

空口监测

应用升级

离线地图

语音播报

设置

城市列表

下载管理

全国

全国基础包(1) --8.4M

北京市(131) --90.6M

上海市(289) --88.8M

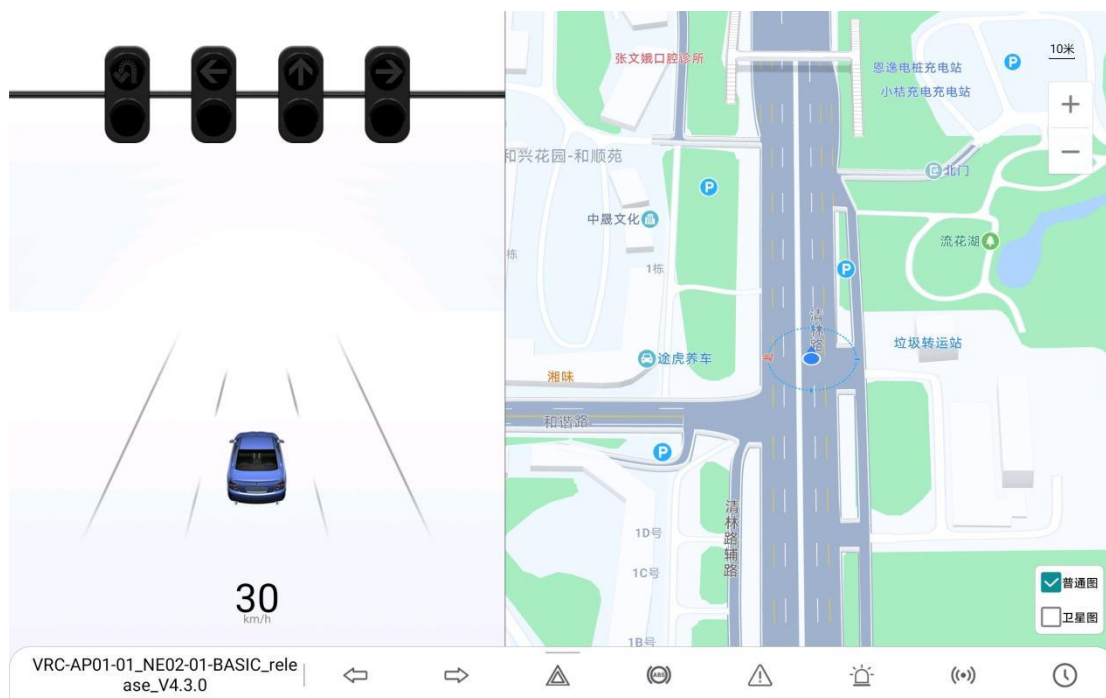
天津市(332) --48.5M

重庆市(132) --129.9M

The screenshot displays the '已下载城市' (Downloaded Cities) section of the '离线地图' (Offline Map) application. The interface includes a sidebar with navigation options: '参数配置' (Parameter Configuration), '设备诊断' (Device Diagnosis), '空口监测' (Air Interface Monitoring), '应用升级' (Application Upgrade), '离线地图' (Offline Map), '语音播报' (Voice Broadcast), and '设置' (Settings). The main content area shows a table of downloaded cities. The first row lists '全国基础包' (National Basic Package) with a status of '最新' (Latest) and a progress of '100%'. To the right of the progress bar are two buttons: '查看' (View) and '删除' (Delete). The '查看' button is highlighted with a red rectangle.

城市列表	下载管理
已下载城市	
全国基础包	最新 100%
	查看 删除

点击“查看”后会自动跳转到主界面当中，现在就可以显示出地图界面了。

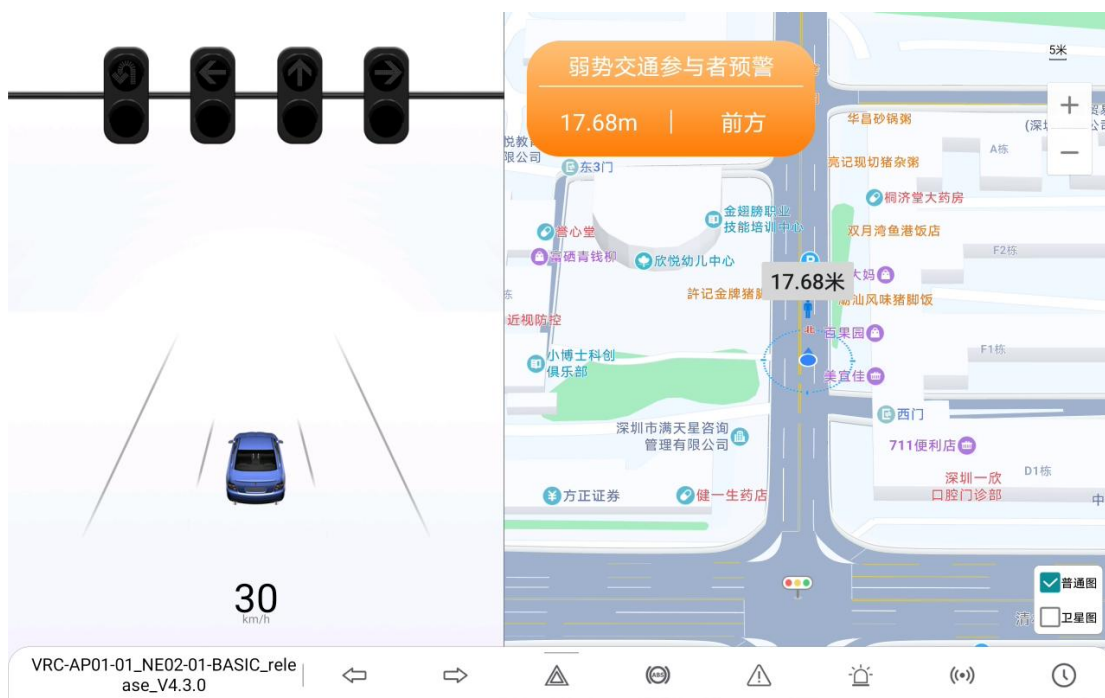


6.5.1 预警事件触发

- 1) 将车辆切换到自动驾驶模式，让其行驶在已经规划好的预警区域内



- 2) 当车辆行驶到预警区域是就会接收到预警消息，并在 PAD 上显示预警事件



● 公司总部

总部电话：0755-29822896
国内销售中心电话：0755-29189185
海外销售中心电话：0755-28077519
地址：深圳市龙岗区横岗街道大运AI小镇A07栋
网址：www.fengxb.com 邮编：518115

● 公司总部售后客服

电话：400-0755-408
0755-29822921
手机：13510391665



官方微信公众号