

辽宁工业大学

D613 实验室设备使用手册

目录

一. 实验室简介.....	1
二. 310 无人机组装调试指南.....	2
1. 遥控器使用说明.....	2
1.1 遥控器对码.....	2
1.2 遥控器说明.....	2
2. 组装步骤详解.....	3
2.1 无人机零件.....	3
2.2 机架组装步骤.....	3
2.3 动力系统安装.....	3
3. 无人机调试.....	5
3.1 QGC 地面站使用	5
3.2 QGC 校准调试	6
3.3 无人机参数调试.....	8
4. 参数保存与加载.....	8
三. 设备介绍与使用.....	10
1. 动捕系统科研平台.....	10
1.1 室内光学定位系统的标定.....	10
1.2 程控启动流程.....	24
1.3 小车飞机单组天地协同.....	30
2. 自主导航无人机科研平台.....	33
2.1 一号四旋翼无人机.....	33
2.2 二号四旋翼无人机.....	36
3. 自主导航无人车科研平台.....	36
3.1 功能介绍.....	36
3.2 功能使用.....	37
四. 无人机电池管理与维护.....	40
1. 使用注意事项.....	40
1.1 充电注意事项.....	40
1.2 存放注意事项.....	41
2. 锂电池基础知识及维护保养:	41
2.1 新收到电池.....	41
2.2 满电和使用过程中的电压下降范围.....	42
2.3 长期保存注意事项.....	42

一. 实验室简介

本团队的无人系统智能控制实验室致力于动捕系统、集群控制、空地协同、轮式无人车、无人机定位导航等领域的研发创新。目前实验室具备动捕系统、无人车 9 台、四旋翼无人机 11 架。其中 8 台无人车及 8 架无人机用于集群控制与空地协同验证；一台无人车用于自主导航验证；两架四旋翼无人机用于深度学习验证；两架四旋翼无人机用于自主定位导航验证。

二. 310 无人机组装调试指南

参考链接: <https://cwkj-tech.yuque.com/bsge84/s2>

1. 遥控器使用说明

1.1 遥控器对码

一台新的遥控器和无人机需要对码, 或者是换其他的遥控的情况下需要重新对码。

对码步骤:

遥控器开机→点击 wfly(或者按键按菜单)→通信设置→对码→对码开始→无人机上电

当听到滴的声音后屏幕右上角显示信号说明对码成功



图 1.1 遥控器

1.2 遥控器说明



图 1.2 遥控器说明

- (1) kill 键: 当无人机失控时, 将 kill 往上拨, 使电机停止旋转
- (2) 模式选择: 从上到下分别为, 自稳、定高、定点。自稳模式为手飞, 没有定位原情况下使用自稳, 难度高, 非专业飞手不要直接上手操作。定点模式为启动程控后使用的模式, 此时定位原为激光雷达或深度相机, 使用遥控器操作会比较稳

- (3) Offboard: 全病程控时，程序运行后将 offboard 键拨到最下面，无人机按照程序自主起飞飞行
- (4) 油门，上下左右分别为升高、下降、左转、右转
- (5) 方向，上下左右分别为前进、后退、左飞、右飞
- (6) 菜单按键
- (7) 确认、旋转按键

2. 组装步骤详解

2.1 无人机零件

机架（包含机臂、机腿）、电机、电调、电池、分电板、桨叶、飞控、机载电脑、激光雷达、深度相机、各种铝柱、螺丝、打印件

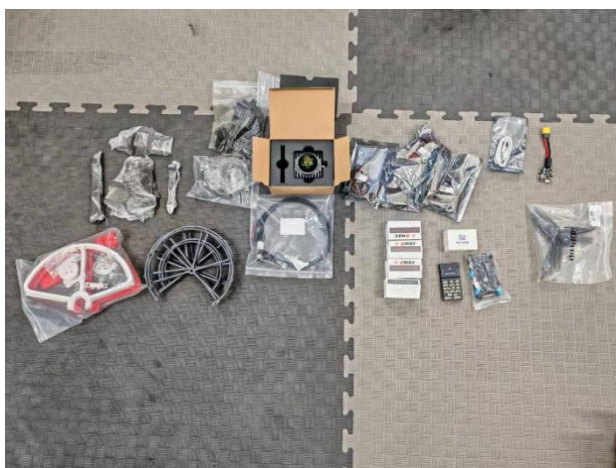


图 1.3 无人机零件

2.2 机架组装步骤

从下往上组装：

机架是无人机的骨架，它承载着所有电子设备。一个牢固、精确的组装是无人机稳定飞行的基础。

安装机腿：将机腿通过螺丝从下向上安装在机臂上。

安装机臂：将四条机臂对准上中心板的安装位置。

连接各中心板：下中心板与中间版之间用长支柱连接、中间板与上中心板之间用短支柱连接、上中心板与顶载雷达板之间用短支柱连接。

2.3 动力系统安装

mid360 激光雷达：将 mid360 激光雷达通过孔位使用螺丝和螺母安装在顶载雷达板上。网口插到机载电脑上，电源线使用降压电源模块，12v3a 输入，其余线不使用。

PX4 飞控板：飞控使用的为 pix 6c, px4 架构。飞控上接入 4 路电调，遥控器接收机、飞控电源、机载电脑连接线（mavlink）

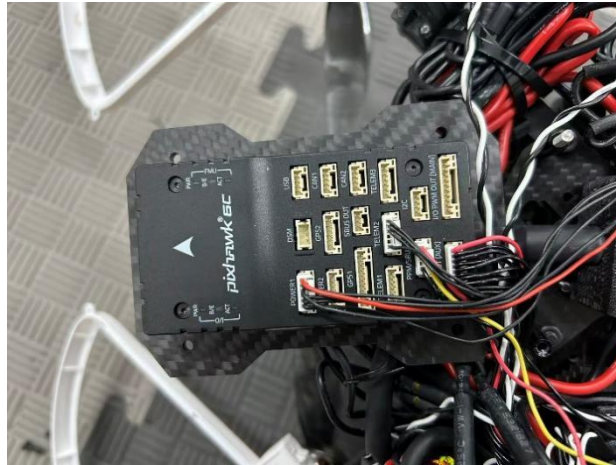


图 1.4 飞控连接

电调：电调使用的为 60A 的电调，这是电调最基本、最重要的功能是调速控制。它接收来自飞控或遥控器接收机的信号。电调解读这个信号后，精确地控制输出给电机的功率，从而实现电机转速的无级调节。将飞控板与电机电调一并安装在中间板上上面。

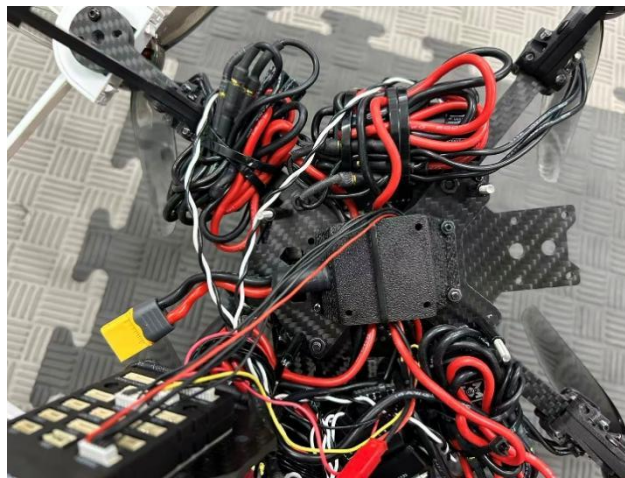


图 1.5 无人机底部

电机：电机使用的好盈 2807 电机，为无人机提供克服重力、实现前进、后退、横移并通过改变各个电机的转速，实现无人机的俯仰、横滚和偏航等功能。电机一般与螺旋桨连接安装在机臂上。

电池：使用 6S 电池，固定在下中心板与中间板之间。



图 1.6 无人机整体图

3. 无人机调试

3.1 QGC 地面站使用

- (1) 通过 type-c 数据线连接电脑 usb 和飞控 c 口
- (2) 开启遥控器，kill 键拨到上面，qgc 显示为：

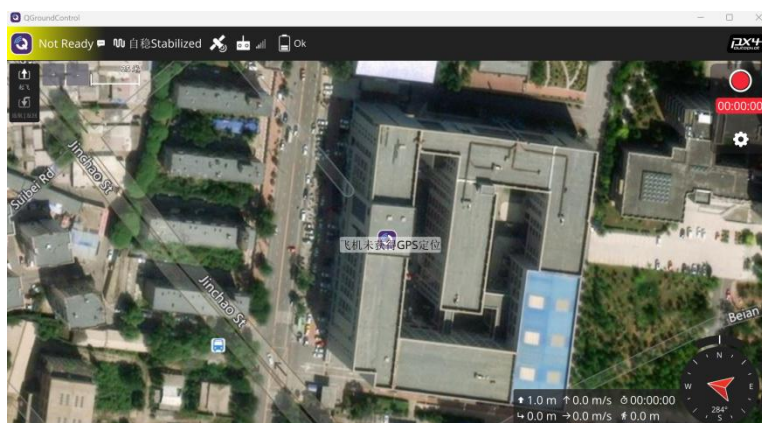


图 1.7 QGC 主页面

- (3) 点击左上角图标，选择设置进入设置页面



1.8 QGC 调试页面

在这个页面中 6 个模块，如果显示绿点说明全部已经设置好，如果是红点则对应的模块没有校准，左边菜单标红则对应没校准设置，不标红则全部校准设置好。

3.2 QGC 校准调试

(1) 传感器校准

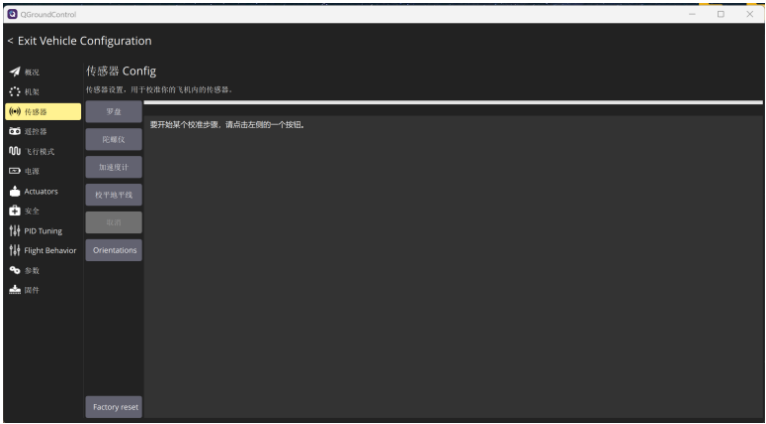


图 1.9 QGC 校准页面

依次校准罗盘、陀螺仪、加速度计、地平线。按照 qgc 中相应图示进行校准

(2) 校准遥控器

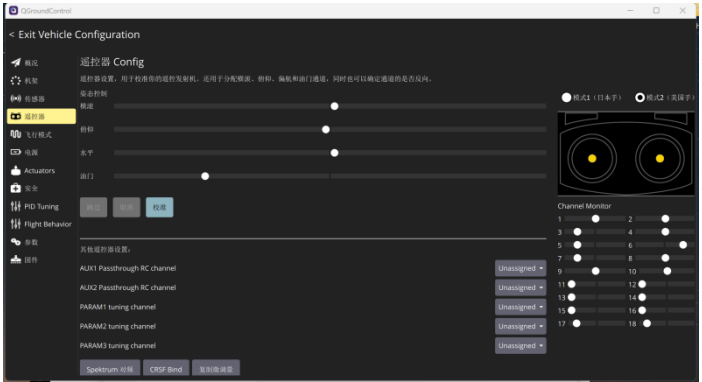


图 1.10 遥控器校准页面

按照右边图示进行校准遥控器

(3) 飞行模式

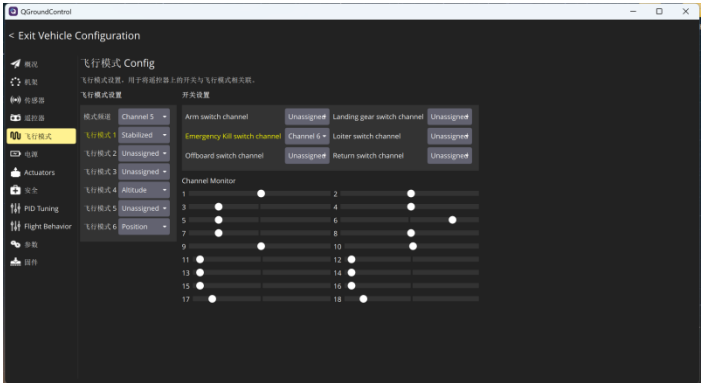


图 1.11 通道选择

根据使用哪个拨杆进行选择哪个通道，此遥控器模式选择的拨杆选择的为 SB 三段式拨杆，对应通道为 5 通道，飞行模式 1、4、6 分别设置为自稳、定高、定点。Kill 键对应的为通道 6，解锁为油门打到右下角保持 2s，桨叶转动，无人机解锁。

(4) 电池电调校准

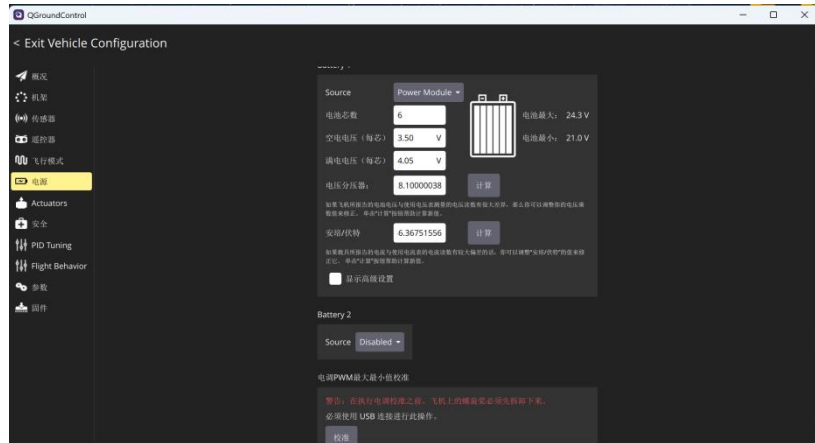


图 1.12 电调校准

无人机断电，点击校准，上电，校准完成

(5) 电机设置

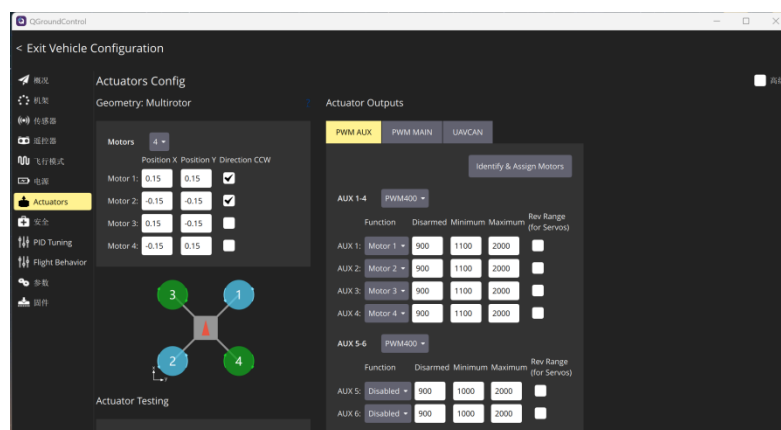


图 1.13 电机设置

选择 PWM AUX 模式，选择电机 1-4，电调校准时候，最小速度是 1000，这里改为 1100，为怠速，如果不改，无人机解锁后电机不转，其实已经解锁了，设置 1100 后，解锁电机会进行旋转，大概 5-10s 后停止。

3.3 无人机参数调试

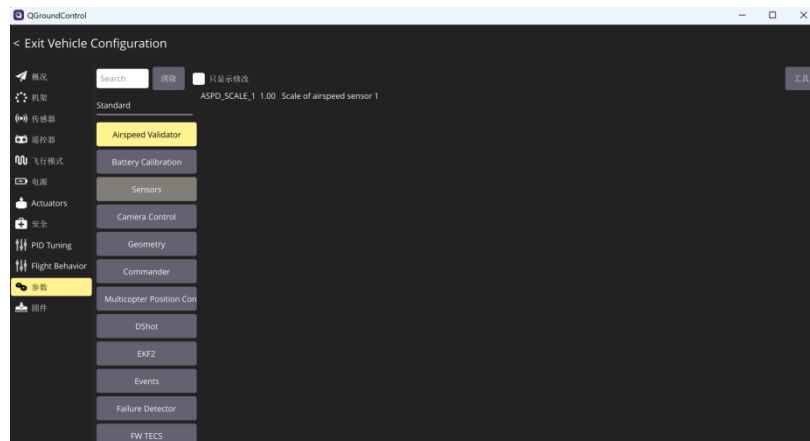


图 1.14 参数设置

主要参数设置：

1. **设置** `COM_ARM_SWISBTN = 1`
启用使用摇杆手势解锁。
2. **设置** `MAN_ARM_GESTURE = 1`
启用手势识别（包括内八）。
3. **确保** `RC_MAP_ARM_SW = 0`
表示不使用独立开关解锁，而是使用摇杆手势。
4. `MAV_1_CONFIG = TELEM 2`
5. `MAV_1_MODE = Onboard`
6. `SER_TEL2_BAUD = 921600 8N1`

4. 参数保存与加载

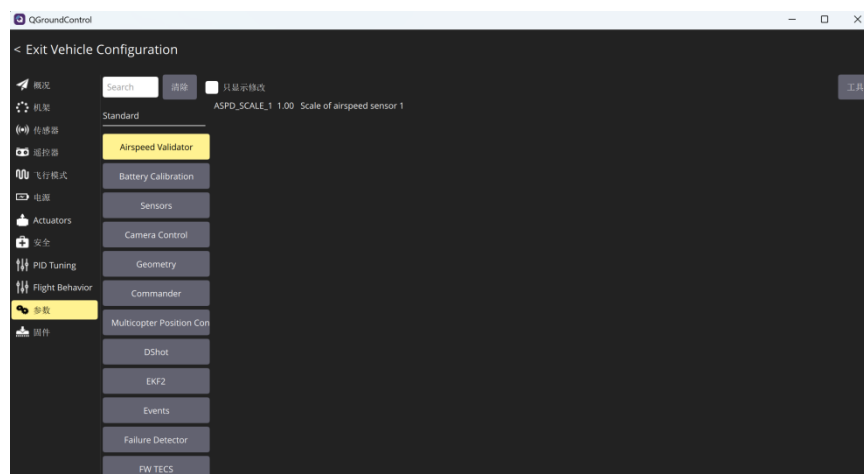


图 1.15 参数保存与加载

调试好的参数可以保存下来

工具→保存到文件

无人机参数有错误飞不稳的情况下可以加载以前保存的参数

工具→load from file.....

组装 310 参数，后缀 params 文件，调试好的下载链接：

<https://github.com/LNUT-AGCL/System.git>

三. 设备介绍与使用

1. 动捕系统科研平台

本系统由无人车集群与无人机集群构成一套完整的空地协同机器人平台。无人车采用差速驱动结构，配备麦克纳姆轮，具备全向移动能力；无人机为四旋翼构型。两类平台均采用高精度光学系统进行定位，定位精度达 0.8 毫米，支持手动遥控与自主集群两种运动模式，可实现复杂环境下的精准协同作业。

系统主要依赖光学运动捕捉技术，实现无人机、无人车的集群控制与空地协同，为无人机、无人车提供高精度毫米级跟踪定位能力；室内光学定位系统布置在无人机集群飞行测试区域四周，主要设备由运动捕捉摄像机、图像定位处理工作站、图像定位解算软件等 3 部分构成。如图 1.1、1.2 所示。



图 1.1 动捕系统



图 1.2 空地协同

该光学室内定位平台构建了一套高精度的室内空间感知网络。其核心在于利用布置在空间中的多个红外摄像机，以极高频率持续捕捉无人机机体上特定反光标识点的运动图像，从而为实时定位奠定数据基础。通过对这些标识点进行建模，系统为每架无人机创建一个虚拟的“刚体”，能精准跟踪其三维空间中的实时位置（X, Y, Z 坐标），更能完整复现其复杂的飞行姿态。最终，通过先进的计算机视觉算法对图像数据进行实时解算，系统能够输出无人机毫秒级延迟的六自由度位姿信息，即三个自由度的平移（位置）和三个自由度的旋转（俯仰、偏航、滚转姿态），实现对无人机运动状态的全方位、高动态捕捉。

1.1 室内光学定位系统的标定

1.1.1 标定前的准备工作

（1）在桌面或安装路径下找到软件可执行文件，双击直接打开。系统软件显示方式人性，操作窗口大小可灵活调节；

（2）打开软件，定位系统主界面如下图所示。菜单栏里有项目，视图，布

局，帮助四个选项，视图是常用的一项，点击视图，里面的工具栏中有设备，标定，平面浏览，刚体，数据流，点击其中的一项，该项就会在主界面上显示出来。显示界面模块化，显示区域与用户操作区域相互独立；显示方式多样化，支持多显示模式、多视角窗口，多视图显示方式如图 1.1.1 所示；

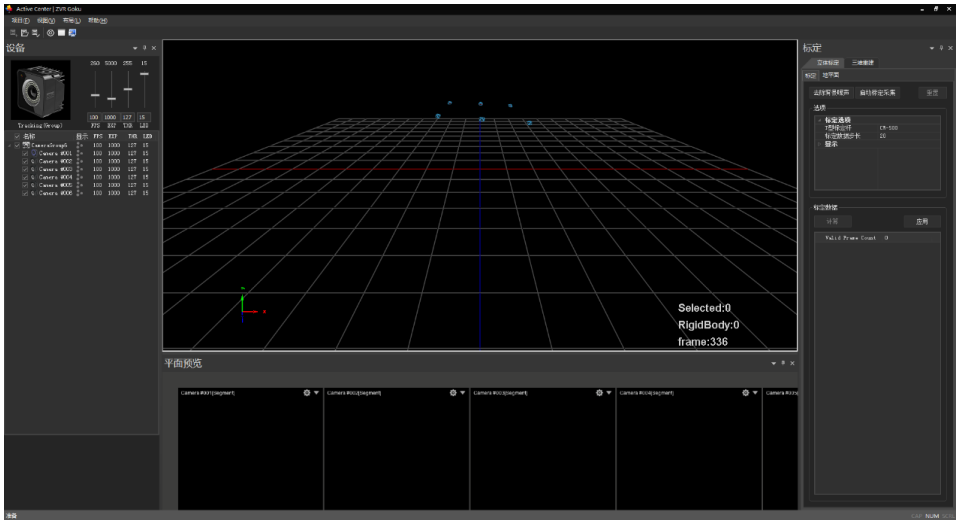


图 1.1.1 定位系统主界面

（3）相机角度和高度调整。在进行相机标定前，首先调节相机的角度，确保相机的拍照区域能够覆盖整块场地，并且能拍到足够或自己想要的高度（相机的安装角度和高度已经确定，如果没有物理改变，不需要再进行调节）。如果需要重新调节，点击相机参数图中红色框内的图标，将相机切换到自然拍照模式，可以更好的调节角度。

（4）相机参数设置。相机有 4 个关键参数，如帧率（FPS），曝光时间（EXP），二值化阈值（THR），红外光亮度（LED），一般情况下不需要调节，默认值即可。FPS:120, EXP:1000, THR:150, LED:主动光时选 0，被动光时 15。由于标定杆是主动发光灯，所以标定时相机 LED 灯参数设置为 0 即可。当无人机上是反光球时，LED 参数应该设置为 15，如图 1.1.2 所示。

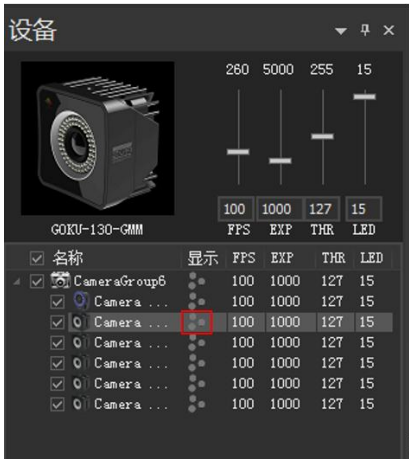


图 1.1.2 相机参数

1.1.2 定位系统标定的操作步骤

(5) 相机标定。在进行相机标定前，要确保平面浏览框中各个相机界面中没有干扰点，即不存在白点。如果存在，则有可能是阳光，反光点或对面相机的红外光形成的的干扰，所以要拉上窗帘，遮住或移走反光物，调节相机俯仰角度等，确保每个相机界面中没有白点，如下图所示。如有无法清除掉的干扰点可以进行屏蔽，点击标定栏中的“去除背景噪声”，图像栏里的干扰点会变成红色，如去除背景噪声图所示，然后开始标定，如图 1.1.3-4 所示。

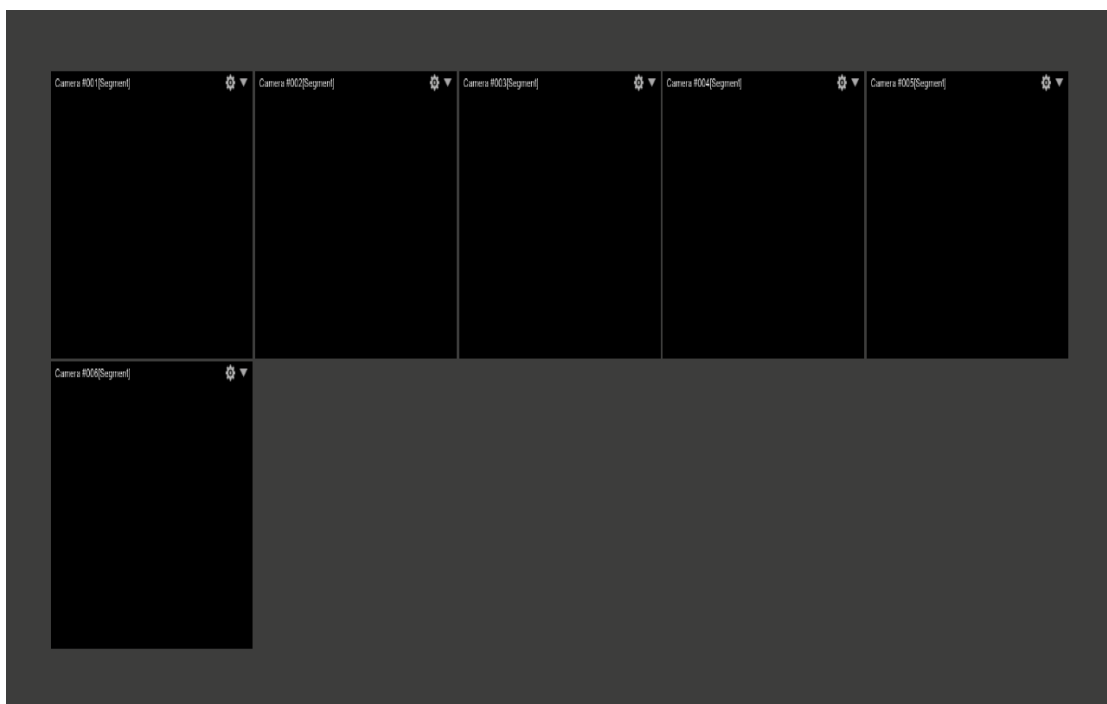


图 1.1.3 相机界面排除干扰点

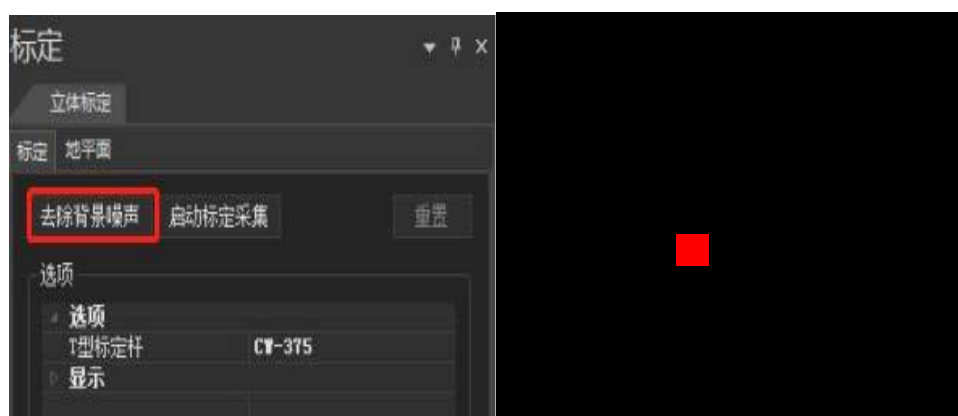


图 1.1.4 去除背景噪声

(6) 打开菜单栏里的视图—>工具栏—>标定，在主界面出现如下图 1.1.5 所示的标定框。



图 1.1.5 启动标定界面

(7) 在标定框中的立体标定一>标定下点击启动标定采集，出现下图 1.1.6 所示的界面。在平面预览中，每个相机窗口出现如同蜘蛛网的图片，右边的标定数据框显示每个相机的标定帧数。此时，代表标定开始。

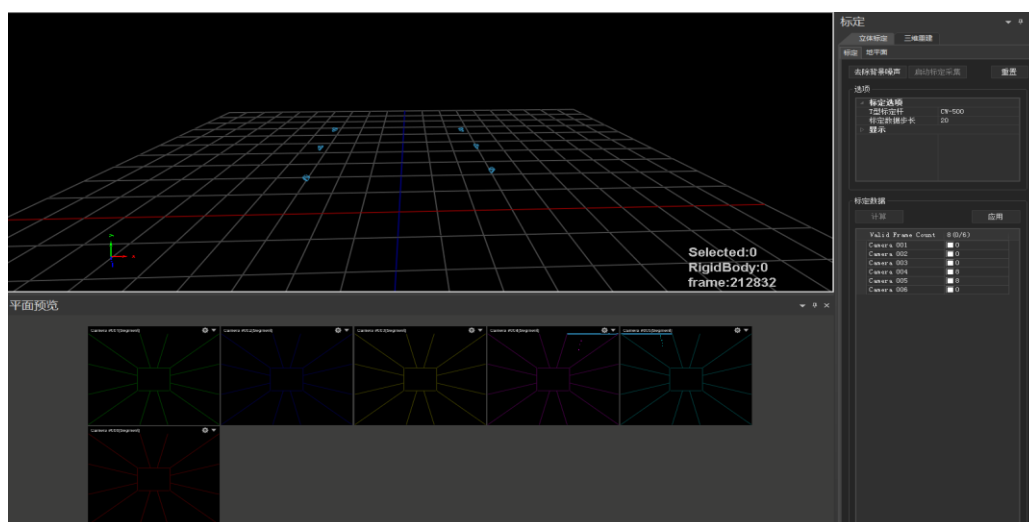


图 1.1.6 标定过程界面

(8) 拿出 T 型标定杆，如下图 1.1.7 所示，打开开关，如下图 (b) 所示位置，此时标定杆上的 3 个红外光源被点亮，如下图 (c) 所示，此时就可以在场地中挥杆标定了。



图 1.1.7 标定 T 形杆

(9) 挥杆时在平面预览框中每个相机界面有点存在。挥杆时尽量使红外光源均匀地 2 个及以上的相机捕捉到，持续挥杆，直到相机上的绿灯被点亮（有 1-2 个没被点亮也没关系），或者在标定数据框中每个相机显示蓝色。最好是相机的绿灯被点亮，这样效果会好点。达到以上效果后，停止挥杆，关闭标定杆电源，如图 1.1.8 所示

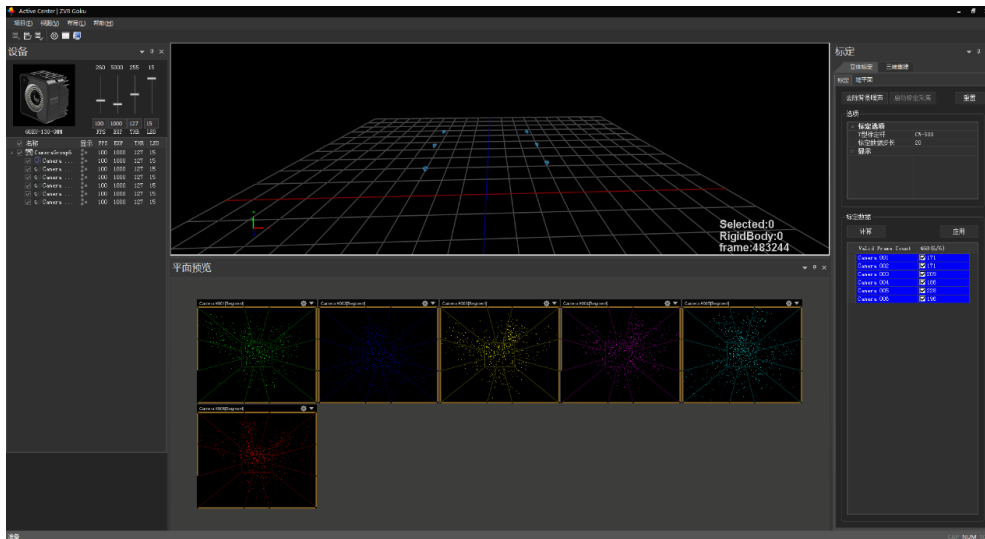


图 1.1.8 捕捉完成界面

(10) 点击计算，定位解算软件计算标定结果，支持多相机同时标定；如下图所示 1.1.9 所示。



图 1.1.9 点击计算

(11) 等待一会，会弹出下图所示的提示框。每个相机的重投影误差结果小于 0.5 为正常，点击确定。否则需要重新挥杆如下图所示 1.1.10 所示。

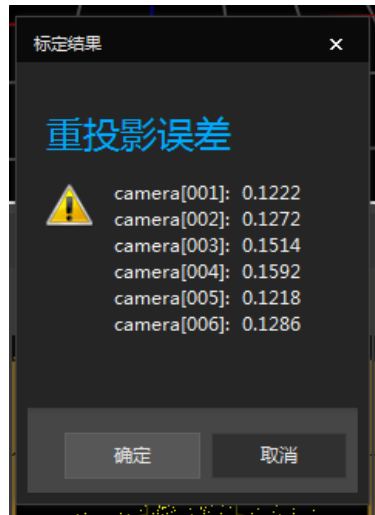


图 1.1.10 标定重投影误差

(12) 设置原点和坐标系。前面的标定只是确定了每个相机之间的相对位姿。但如果想形成一个精确的位置空间，还需要设置坐标原点和坐标系。可以在控制界面上指定 Yup 或 Zup 坐标系朝向；如下图所示，将直角标定器放在场地中央，方向可以自己设定，其中短边为 X 轴，长边为 Z 轴。拐点出为原点。本系统要求短边指向东，选择 Zup 坐标系如图 1.1.11 所示。

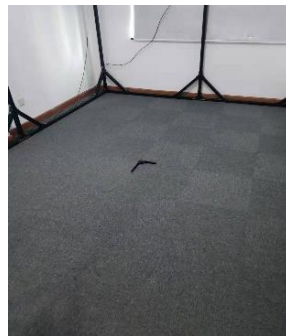


图 1.1.11 直角标定器

(13) 放置好直角标定器后，并打开开关，然后在标定框中的立体标定—>地平面下点击应用，如下图 1.1.12 所示。

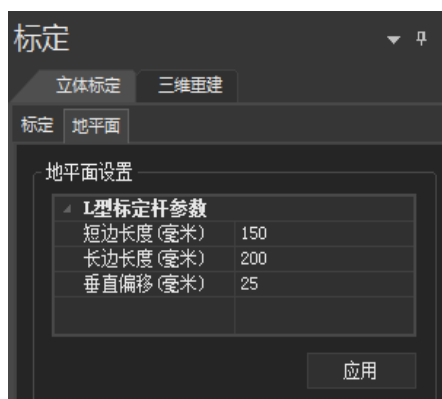


图 1.1.12 原点标定

(14) 标定数据可保存与加载，便于重复调用；如下图 1.1.13 所示，点击红框里的保存，将此次标定结果保存下来，下次打开可以重复使用。



图 1.1.13 标定保存界面

(15) 此时，相机标定完成，如下图 1.1.14 所示，界面中相机的位置跟实际的安装位置一致，并在中间形成了一个三维捕获空间。当场地中存在刚体时，会实时显示该刚体的位姿信息。

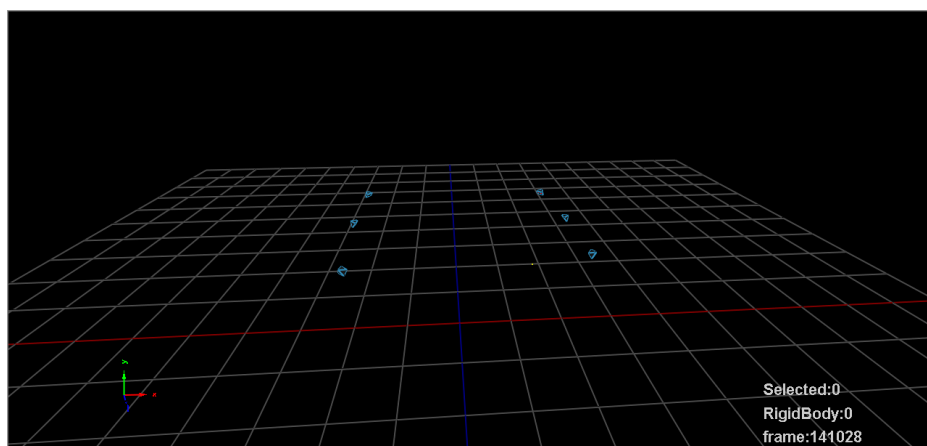


图 1.1.14 标定完成相机界面

创建刚体和命名：

定位系统刚体绑定灵活，组成刚体的光球数量可调，可选择使用模板或不使用模板创建刚体；

(1) 将带有反光球的无人机放入场地中，会显示每个小球的光点，如下图 1.1.15 所示。

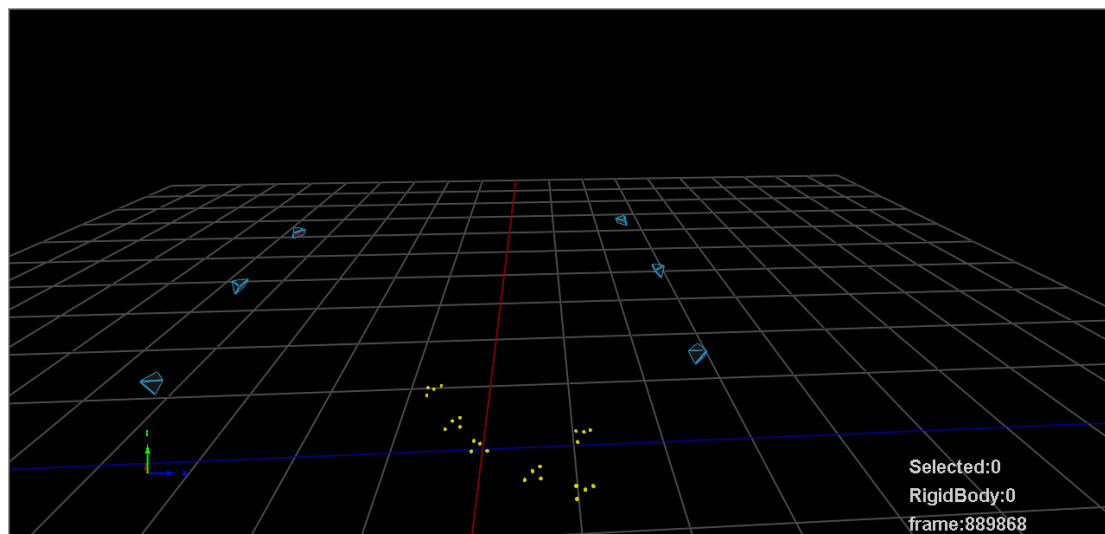


图 1.1.15 反光球显示界面

(2) 鼠标左键选中第一个无人机，如下图 1.1.16 所示。

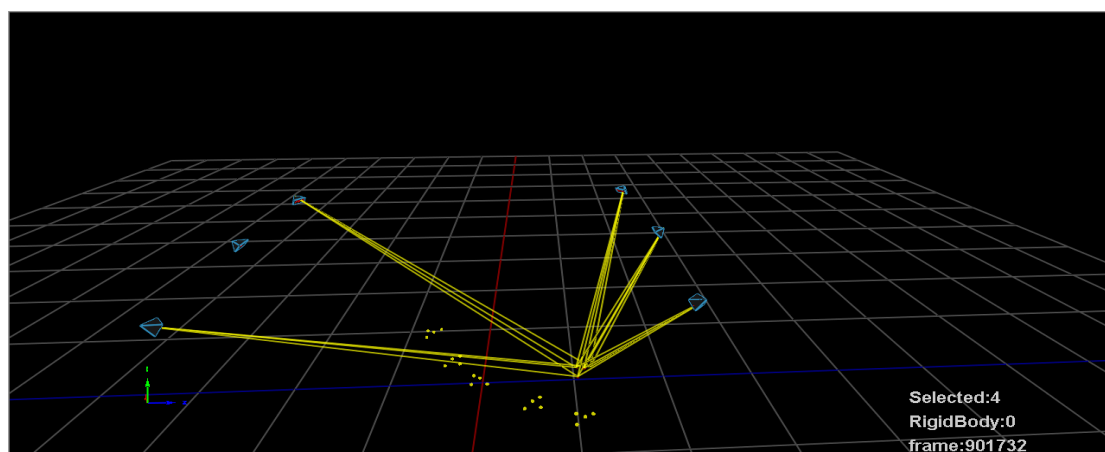


图 1.1.16 刚体选中界面

(3) 然后点击鼠标右键，创建刚体。如下图 1.1.17 所示，形成有颜色的立体图代表刚体创建成功。刚体调整方式便捷，可通过直接输入数值或拖动鼠标方式进行刚体调整。

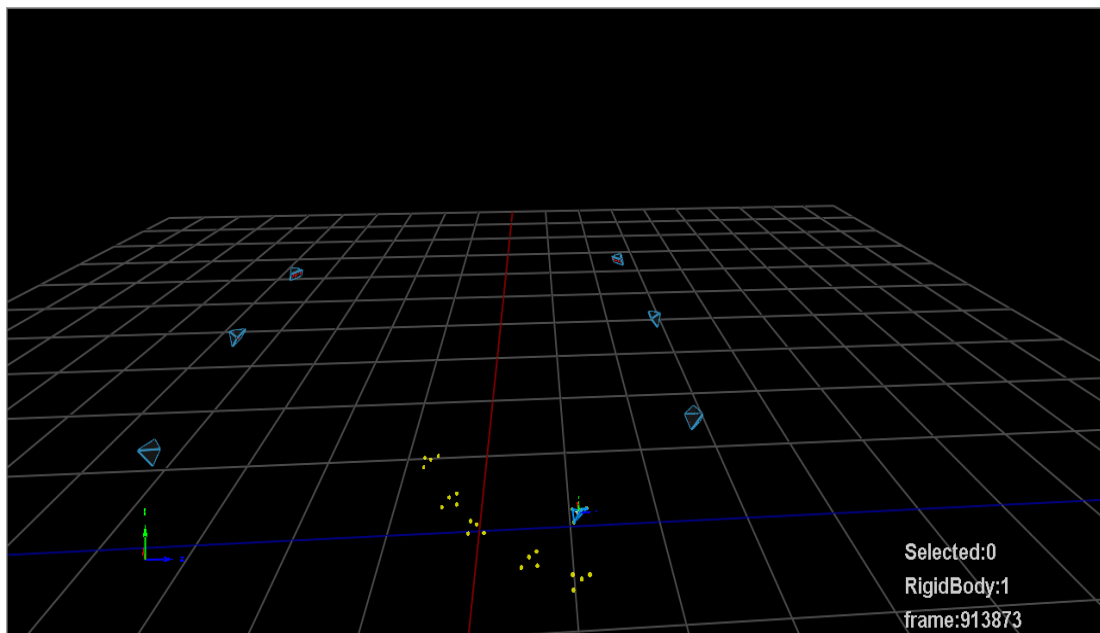


图 1.1.17 单个刚体创建完成界面

(4) 依次创建其他的刚体，如下图 1. 1. 18 所示

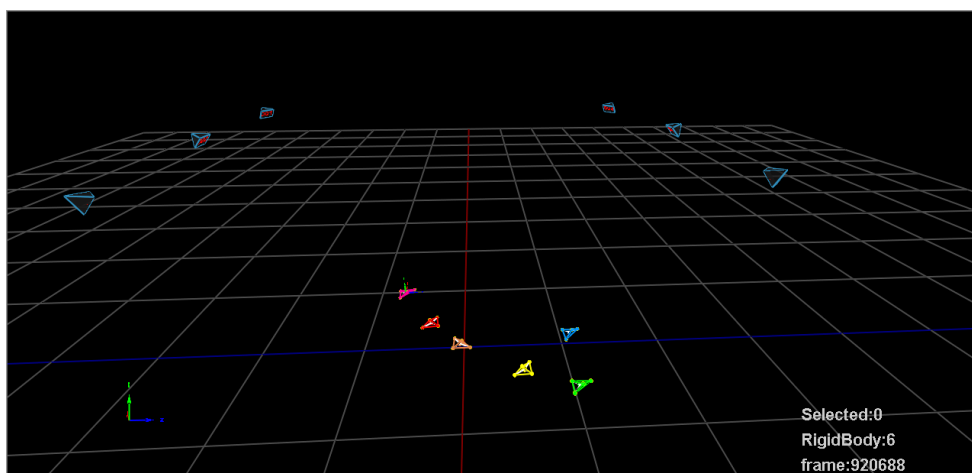


图 1.1.18 全部刚体创建完成界面

(5) 打开视图—>工具栏—>刚体，会出现刚体的信息栏。刚体数据处理多样，刚体数据可保存，可另存，可加载，可替换，如下图 1. 1. 19 所示。

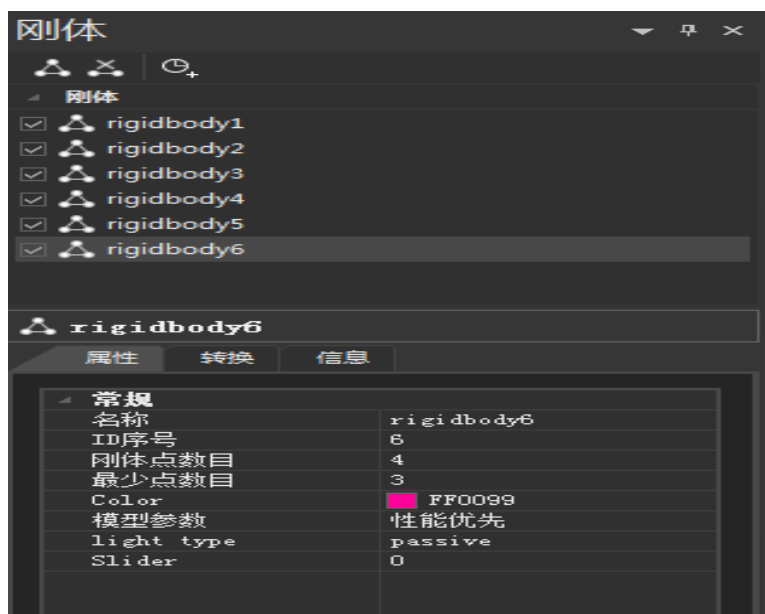


图 1.1.19 刚体信息界面

(6) 此时定位系统会追踪多个刚体，每个刚体有自己的刚体名称和序号。在信息一栏中会显示每个刚体的位姿信息。飞机的刚体的名称从“droneyee1”，开始，即 1 号飞机是 droneyee1，二号飞机是 droneyee2;小车的刚体名称从“droneyee11”开始，即一号小车是 droneyee11，二号小车是 droneyee12。

三维有效区域显示

通过标定的定位系统会形成一个毫米级的动态捕获空间，并可以查看生成的有效捕获区域。

(1) 如下图 1.1.20 所示，点击相机->CameraGroup。

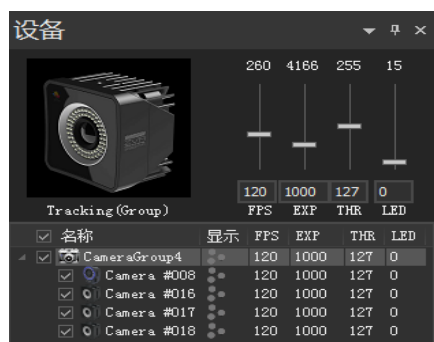


图 1.1.20 设备界面

(2) 出现如下图 1.1.21 所示的界面。



图 1.1.21 CameraGroup 界面

(3) 点击视域，选择 True。出现如下图的有效立体区域显示。包含区域的长、宽、高信息如下图 1.1.22 所示。

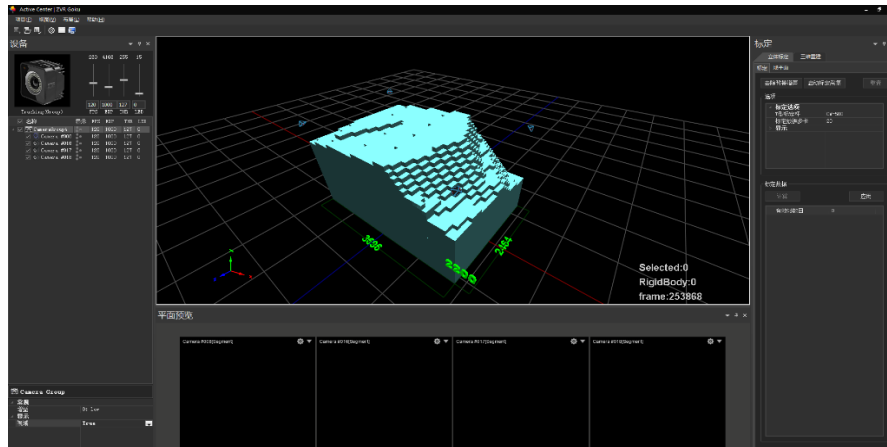


图 1.1.22 有效区域显示界面

运动轨迹显示

无人机在三维有效空间中运动时会形成运动轨迹，可以在软件界面实时显示无人机的运动轨迹。

(1) 创建刚体，如下图 1.1.23 所示。

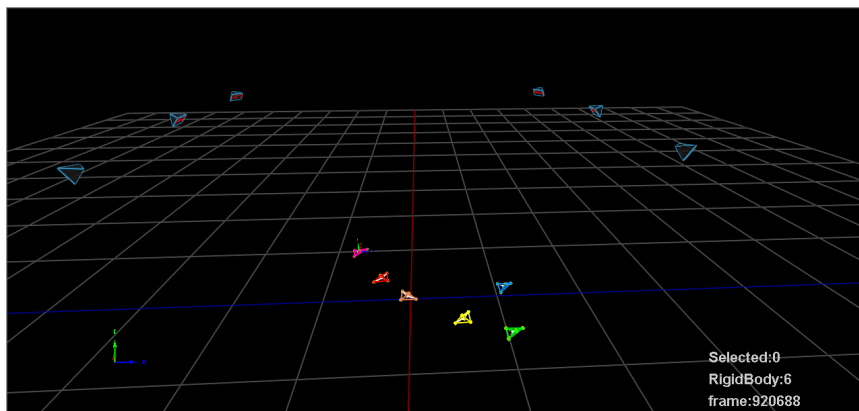


图 1.1.23 刚体创建界面

(3) 选择一个刚体，在运动轨迹（秒）里设置轨迹显示的时间，如下图图 1.1.24 所示，设置为 50 秒，即 50 秒后轨迹会消失。

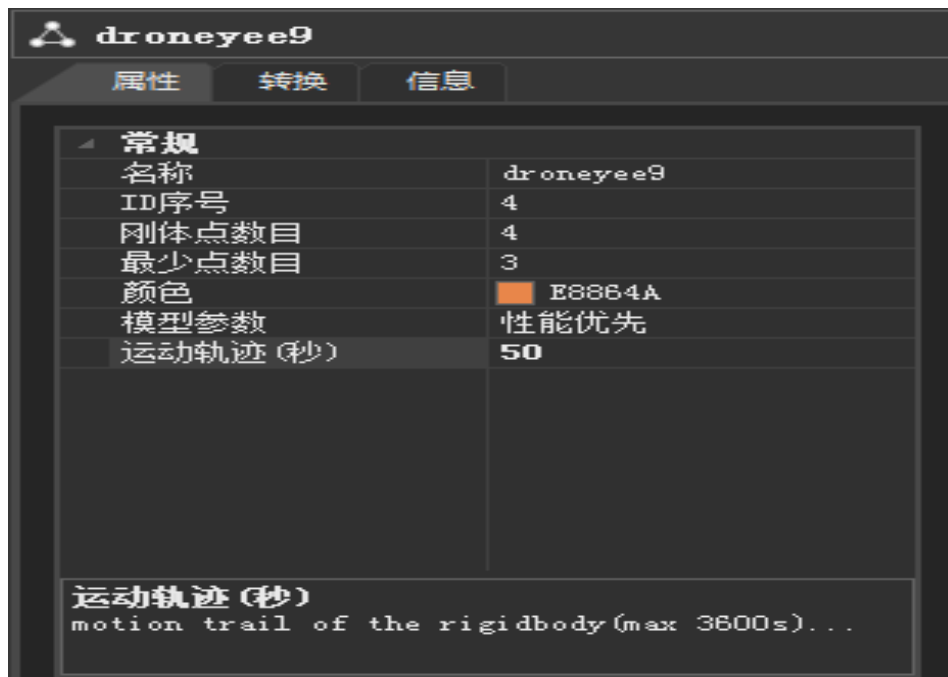


图 1.1.24 运动轨迹设置界面

(3.1) 当无人机运动时，会在界面显示运动轨迹，如下图 1.1.25 所示。

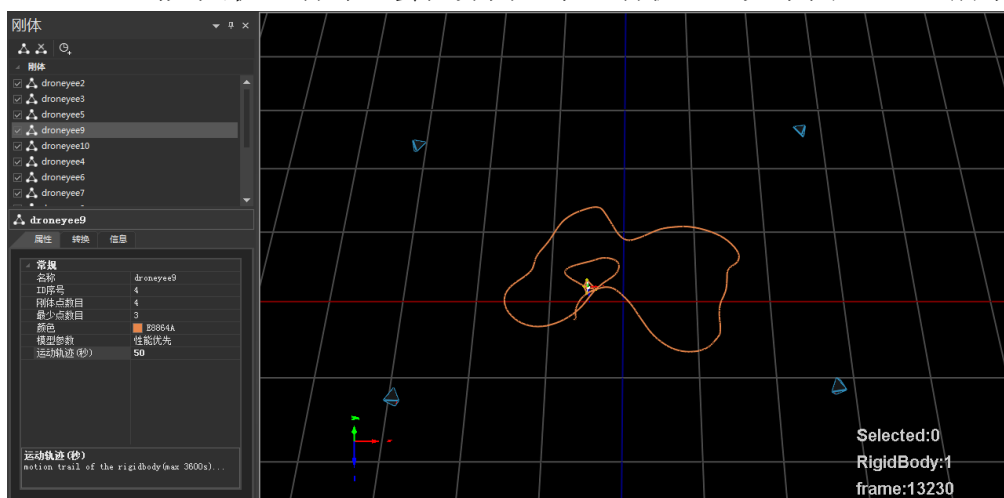


图 1.1.25 轨迹显示界面

(3.2) 从设置开始，50 秒后，无人机运动轨迹逐渐消失。如下图 1.1.26 所示。

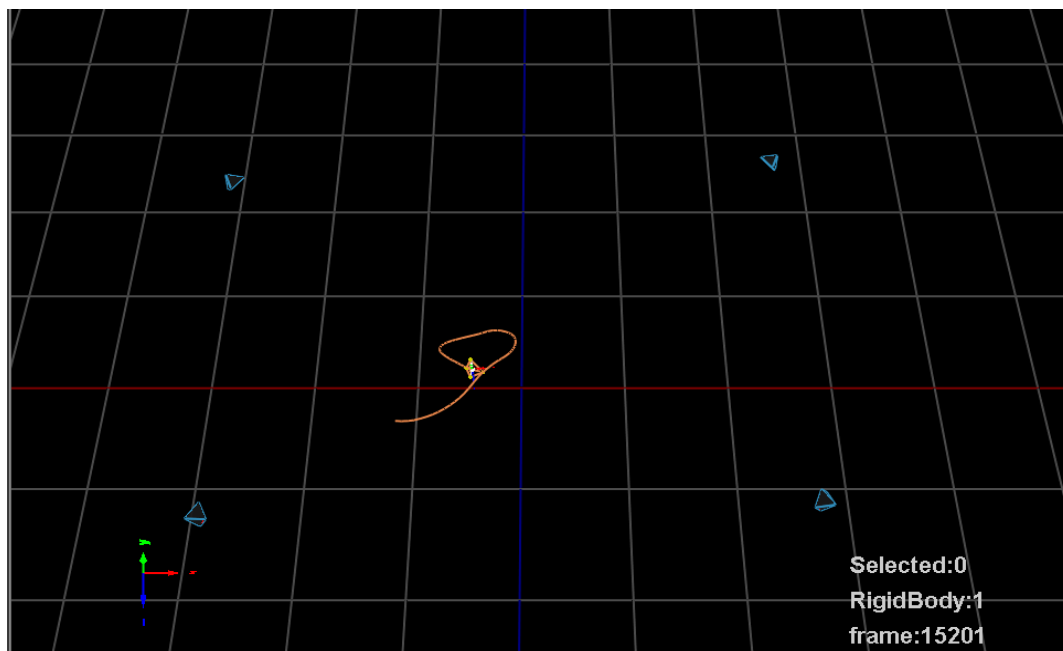


图 1.1.26 轨迹时效图片

无人机运动回放

(1) 打开菜单栏里的视图—>工具栏—>数据流。点击保存原始数据，此时会在软件安装路径下产生一个.raw 的文件，即无人机运动的轨迹记录文件。如下图所示。

(2) 在安装路径下打开 AC_View.exe 软件。如下图所示。

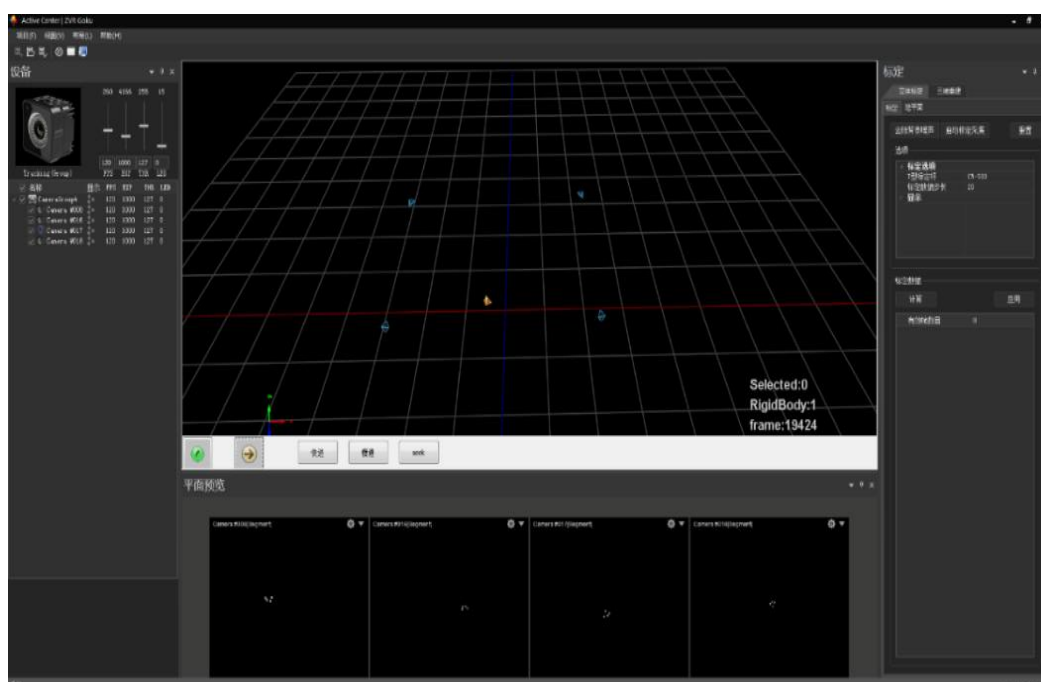
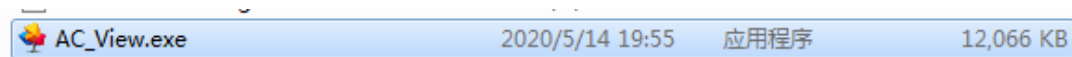


图 1.1.27 轨迹回放界面

(3) 点击左侧的图标，找到保存的.raw 文件并打开。如下图 1.1.28 所示

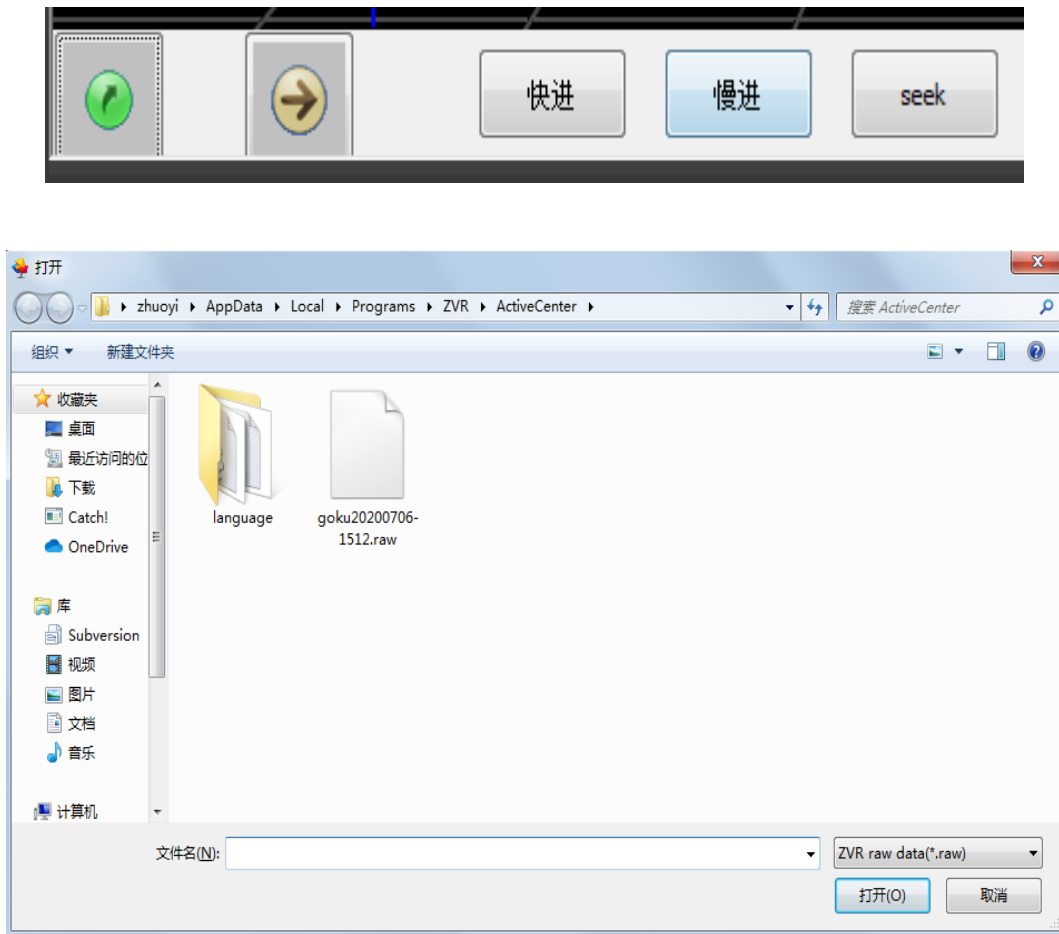



图 1.1.28 轨迹文件储存路径

(4) 点击  图标，会回放该刚体之前的运动状态。如下图 1.1.29 所示。

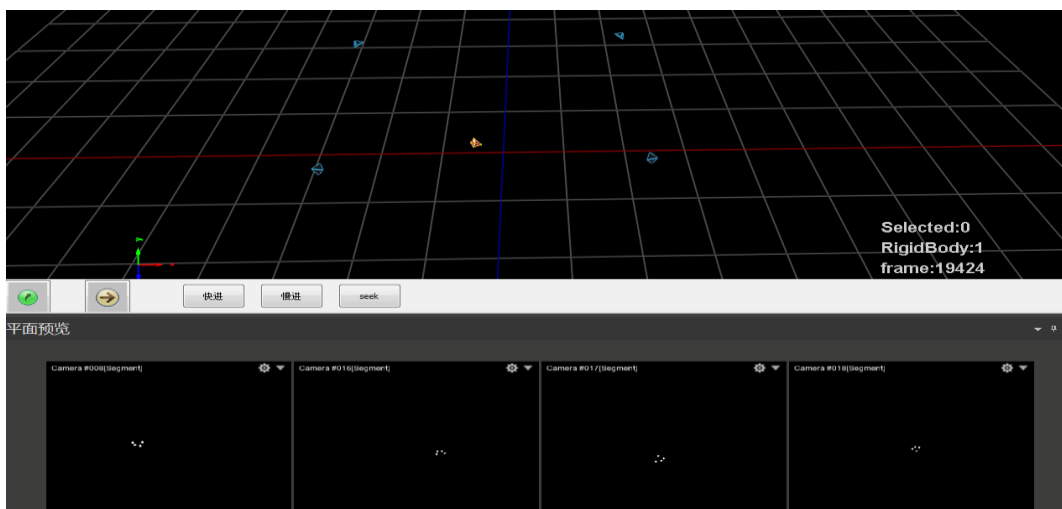


图 1.1.29 运动状态回放界面

1.2 程控启动流程

1.2.1 程控前的检查

(1) 无人机电池电量检查，满电电压单节 4.2V、最低电压单节 3.7V。

(2) 无人机状态检查。检查飞机桨叶是否受损需要更换；检查飞机机体是否完整；检查飞机各个线路连接是否稳定

(3) 检查飞机传感器是否正常。先使用遥控器解锁，如果能够解锁则说明传感器正常，如果不能解锁则说明传感器可能存在问题，此时如果连接地面站则会有传感器错误显示，此情况需要重新校准传感器。

(4) 检查飞机 ID 是否对应。飞机 ID 为唯一，参数内显示 sys_id 对应飞机的编号。必须保证该数值一致。

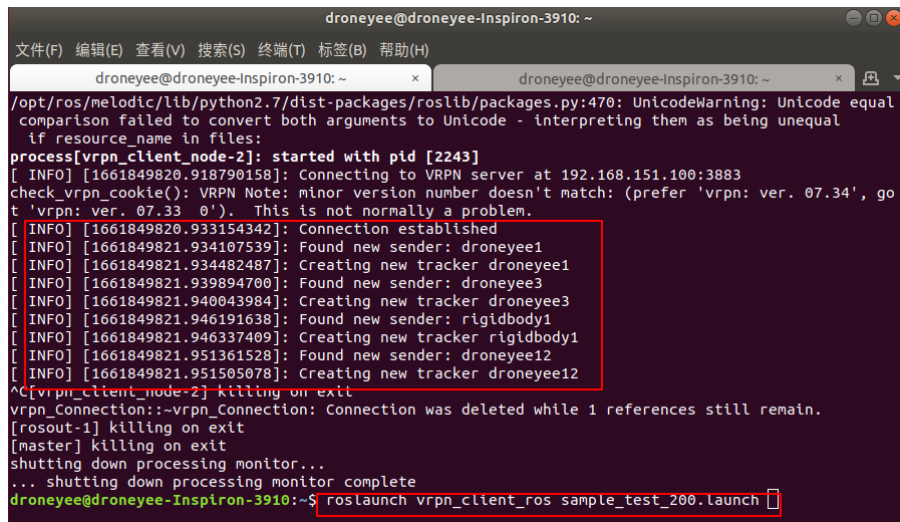
(5) 检查飞机通信数据是否正常；控制站 ping 至飞机 IP，是否有正常通信。控制站电脑连接路由器之后通过终端连接飞机的机载 WiFi 模块，终端内输入 ping 【飞机的 IP 地址】（飞机正常供电）。

(6) 定位数据是否有问题。在飞机启动完成之后查看飞机定位数据是否正常，通过 ROS 话题查看飞机内部数据的 local_position 是否和定位系统该处的 vrpn 数据一致。命令行如下数据查询：

```
rostopic list           //查看当前有哪些 ros 话题
rostopic echo +话题名字 //查看当前话题数据
例如：rostopic echo /vrpn_client_node/droneyeel/pose
rostopic echo /droneyeel/mavros/vision_pose/pose
```

1.2.2 飞机的状态和通信检定

(1) 启动定位系统：在桌面打开一个终端，输入指令：roslaunch vrpn_client_ros sample_test_200.launch，（此步是为了接收图形控制站通过 VRPN 协议转发过来的刚体数据以此来验证图形工作站是否和控制台的数据链路是否连接正常）如下图 1.2.1 所示：



```
droneeye@droneeye-Inspiron-3910: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 标签(B) 帮助(H)  
droneeye@droneeye-Inspiron-3910: ~  
/opt/ros/melodic/lib/python2.7/dist-packages/roslib/packages.py:470: UnicodeWarning: Unicode equal  
comparison failed to convert both arguments to Unicode - interpreting them as being unequal  
if resource_name in files:  
process[vrpn_client_node-2]: started with pid [2243]  
[ INFO] [1661849820.918790158]: Connecting to VRPN server at 192.168.151.100:3883  
check_vrpn_cookie(): VRPN Note: minor version number doesn't match: (prefer 'vrpn: ver. 07.34', go  
t 'vrpn: ver. 07.33 0'). This is not normally a problem.  
[ INFO] [1661849820.933154342]: Connection established  
[ INFO] [1661849821.934107539]: Found new sender: droneeye1  
[ INFO] [1661849821.934482487]: Creating new tracker droneeye1  
[ INFO] [1661849821.939894700]: Found new sender: droneeye3  
[ INFO] [1661849821.940043984]: Creating new tracker droneeye3  
[ INFO] [1661849821.946191638]: Found new sender: riglbody1  
[ INFO] [1661849821.946337409]: Creating new tracker riglbody1  
[ INFO] [1661849821.951361528]: Found new sender: droneeye12  
[ INFO] [1661849821.951505078]: Creating new tracker droneeye12  
^C[vrpn_client_node-2]: killing on exit  
vrpn_Connection::~vrpn_Connection: Connection was deleted while 1 references still remain.  
[rosout-1] killing on exit  
[master] killing on exit  
shutting down processing monitor...  
... shutting down processing monitor complete  
droneeye@droneeye-Inspiron-3910: ~$ roslaunch vrpn_client_ros sample_test_200.launch
```

图 1.2.1 定位系统启动界面

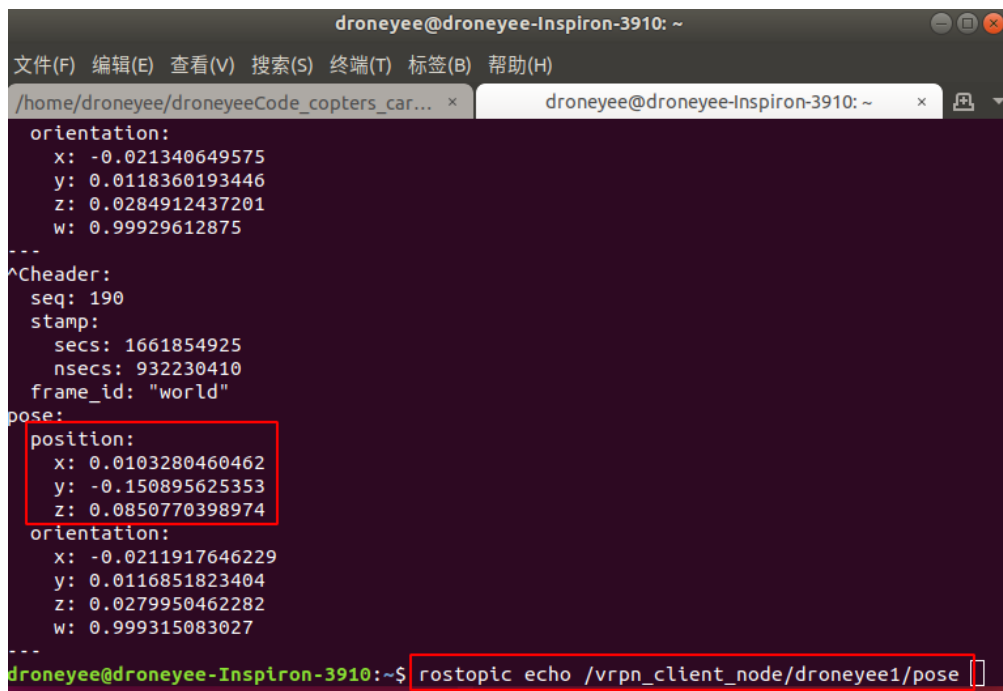
(2) 查看数据源：在桌面打开一个终端，输入指令：

1. rostopic list (此步是查询前面命令启动后打印的话题)
2. rostopic echo /vrpn_client_node/droneeye1/pose (此步是查询第一步显示刚体的位置数据否能和图形控制站中的坐标能否对应) 如下图 1.2.2, 1.2.3 所示:



```
droneeye@droneeye-Inspiron-3910: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 标签(B) 帮助(H)  
/home/droneeye/droneeyeCode_copters_car... x droneeye@droneeye-Inspiron-3910: ~ x  
droneeye@droneeye-Inspiron-3910: ~$ rostopic list  
/rosout  
/rosout_agg  
/tf  
/vrpn_client_node/droneeye1/pose  
droneeye@droneeye-Inspiron-3910: ~$
```

图 1.2.2 话题查看界面



```
droneyee@droneyee-Inspiron-3910: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 标签(B) 帮助(H)
/home/droneyee/droneyeeCode_copters_car... x droneyee@droneyee-Inspiron-3910: ~ x
orientation:
  x: -0.021340649575
  y: 0.0118360193446
  z: 0.0284912437201
  w: 0.99929612875
---
^Cheader:
  seq: 190
  stamp:
    secs: 1661854925
    nsecs: 932230410
  frame_id: "world"
pose:
  position:
    x: 0.0103280460462
    y: -0.150895625353
    z: 0.0850770398974
  orientation:
    x: -0.0211917646229
    y: 0.0116851823404
    z: 0.0279950462282
    w: 0.999315083027
---
droneyee@droneyee-Inspiron-3910:~$ rostopic echo /vrpn_client_node/droneyee1/pose
```

图 1.2.3 数据源查看界面

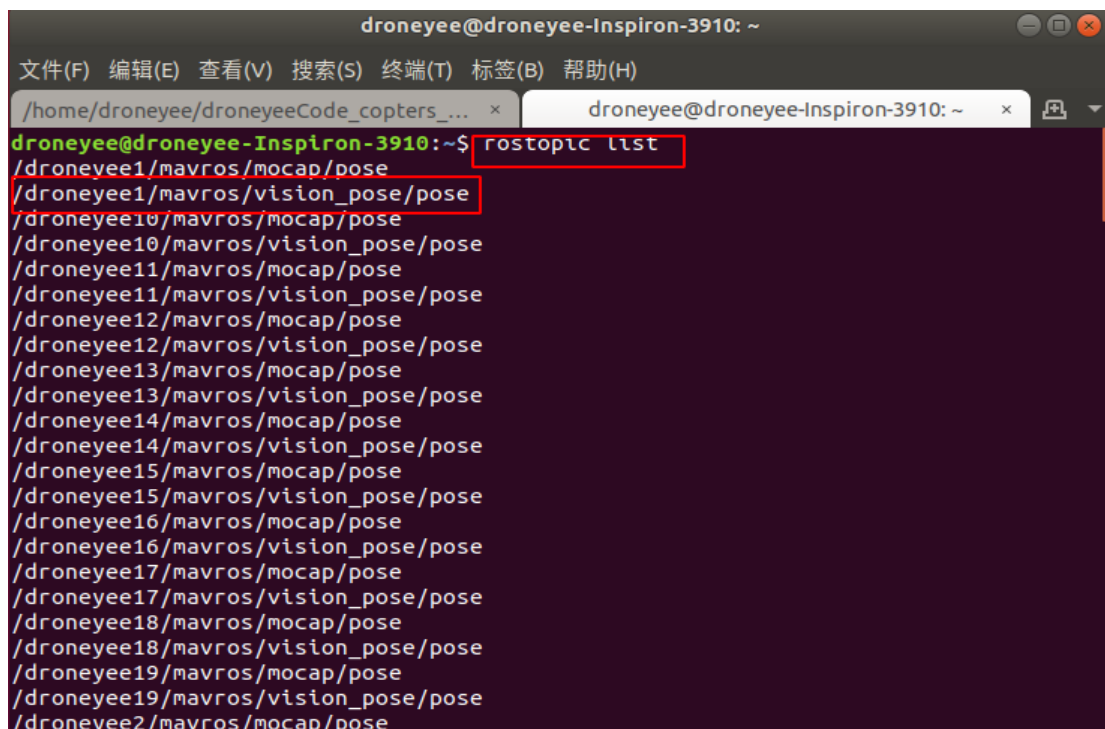
注:

1. 此坐标数据与 ZVR 软件中坐标数值所用的单位不一样故而只需对应数值即可
2. 此坐标数据与 ZVR 软件的坐标体系不一样故而只需对应数值无需对应坐标轴。

(3) 启动 matlab 接收端的节点, 输入指令 `roslaunch network_interface droneyee_vrpn_swarm_mocapCarsCopters.launch`. (因为此系统是基于 mavlink 的底层协议控制故而此步骤是为了将前面查询到的刚体位置数据基于 ros 做了封装转换。) 此步如果启动失败在 matlab 的模型中会没有飞机的数据信息, 当发现模型中没有数据信息时, 可以重新启动一次。

(4) 查看数据源: 在桌面打开一个终端, 输入指令:

1. `rostopic list` (此步是查询前面命令启动后打印的话题)
2. `rostopic echo /droneyee1/mavros/vision_pose/pose` (因为已经做了数据转换故而需要查询转换后的数据源与未转换之前的坐标能否对应) 如下图 1.2, 4 所示:

A terminal window titled 'droneyee@droneyee-Inspiron-3910: ~' with a menu bar (文件(F), 编辑(E), 查看(V), 搜索(S), 终端(T), 标签(B), 帮助(H)). The terminal shows the command 'rostopic list' being executed, with the output listing various topics like '/droneyee1/mavros/mocap/pose' and '/droneyee1/mavros/vision_pose/pose'. The command and the first two lines of output are highlighted with red boxes.

```
droneyee@droneyee-Inspiron-3910: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 标签(B) 帮助(H)  
/home/droneyee/droneyeeCode_copters_... x droneyee@droneyee-Inspiron-3910: ~ x  
droneyee@droneyee-Inspiron-3910:~$ rostopic list  
/droneyee1/mavros/mocap/pose  
/droneyee1/mavros/vision_pose/pose  
/droneyee10/mavros/mocap/pose  
/droneyee10/mavros/vision_pose/pose  
/droneyee11/mavros/mocap/pose  
/droneyee11/mavros/vision_pose/pose  
/droneyee12/mavros/mocap/pose  
/droneyee12/mavros/vision_pose/pose  
/droneyee13/mavros/mocap/pose  
/droneyee13/mavros/vision_pose/pose  
/droneyee14/mavros/mocap/pose  
/droneyee14/mavros/vision_pose/pose  
/droneyee15/mavros/mocap/pose  
/droneyee15/mavros/vision_pose/pose  
/droneyee16/mavros/mocap/pose  
/droneyee16/mavros/vision_pose/pose  
/droneyee17/mavros/mocap/pose  
/droneyee17/mavros/vision_pose/pose  
/droneyee18/mavros/mocap/pose  
/droneyee18/mavros/vision_pose/pose  
/droneyee19/mavros/mocap/pose  
/droneyee19/mavros/vision_pose/pose  
/droneyee2/mavros/mocap/pose
```

图 1.2.4 数据源查看界面

(5)135 文件检定：飞机初次使用或者使用时间间隔过长需进行此步操作。

首先通过命令打开 135_only.launch 文件(此文件的作用为建立飞机与地面站的纽带)，打开一个终端输入；

cd droneyee\ Optics/在此路径下用：gedit 135_only.launch 命令打开文件如图 1.2.5 所示：

Group up: 改成测试产品图形控制站对应刚体名称

ID: 改成产品地面站软件对应端口

Fcu_url: 改成产品对应的端口号和 ip

Gcs_url: ip 改为地面站软件所在电脑的 IP，端口号定义为 14579。

改完以后保存退出。


```
droneyee@droneyee-Inspiron-3910: ~/droneyee_Optics
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
droneyee@droneyee-Inspiron-3910:~$ cd droneyee\ Optics/
droneyee@droneyee-Inspiron-3910:~/droneyee_Optics$ ls
'135_only(1).launch'  droneyee_track      mav.conf
droneyee_follow      droneyee_uac_ufc.zip  MavRouter_standard_car.py
droneyee_formation   mav_car.conf         MavRouter_standard.py
droneyee@droneyee-Inspiron-3910:~/droneyee_Optics$ gedit 135_only\1\1.launch
```

图 1.2.5 文件打开路径

在上个终端中打印 135 文件建立通信连接命令如下：roslaunch 135_only.launch 如图 1.2.6, 1.2.7 所示。

```
[ INFO] [1655269835.327674057]: VER: 3.1: Capabilities      0x00000000000000e4
f
[ INFO] [1655269835.327742314]: VER: 3.1: Flight software:   010a0000 (152118
0ed000000)
[ INFO] [1655269835.327803567]: VER: 3.1: Middleware software: 010a0000 (152118
0ed000000)
[ INFO] [1655269835.327857339]: VER: 3.1: OS software:      071d00ff (427238
33be2b0ec)
[ INFO] [1655269835.327906540]: VER: 3.1: Board hardware:    00000012
[ INFO] [1655269835.327959805]: VER: 3.1: VID/PID:          26ac:0012
[ INFO] [1655269835.328017823]: VER: 3.1: UID:              3535510533383932
[ WARN] [1655269835.329878230]: CMD: Unexpected command 520, result 0
[ WARN] [1655269837.217643596]: TM : RTT too high for timesync: 33.26 ms.
[ WARN] [1655269838.915896625]: TM : RTT too high for timesync: 31.58 ms.
```

图 1.2.6 心跳信息显示界面

```
process[droneyee3/mavros-1]: started with pid [12640]
[ INFO] [1655269834.104992265]: FCU URL: udp://:18553@192.168.151.153:18553
[ INFO] [1655269834.105932328]: udp0: Bind address: 0.0.0.0:18553
[ INFO] [1655269834.105960634]: udp0: Remote address: 192.168.151.153:18553
[ INFO] [1655269834.105987100]: GCS URL: udp://:14509@192.168.151.100:14579
[ INFO] [1655269834.106018251]: udp1: Bind address: 0.0.0.0:14509
[ INFO] [1655269834.106039452]: udp1: Remote address: 192.168.151.100:14579
[ INFO] [1655269834.113131313]: Plugin 3dr radio loaded
```

图 1.2.7 135 文件更改显示

在通信连接建立完成后需查看心跳信息如图片一中所示信息注意: 图片所示信息 VER 为 1.1 是因为此设备刚体为“droneyee1”例: 刚体为“droneyee*”则 VER 信息为“*.1”

通信连接建立后通过 14579 端口连接地面站软件如下图 1.2.8,1.2.9 所示:

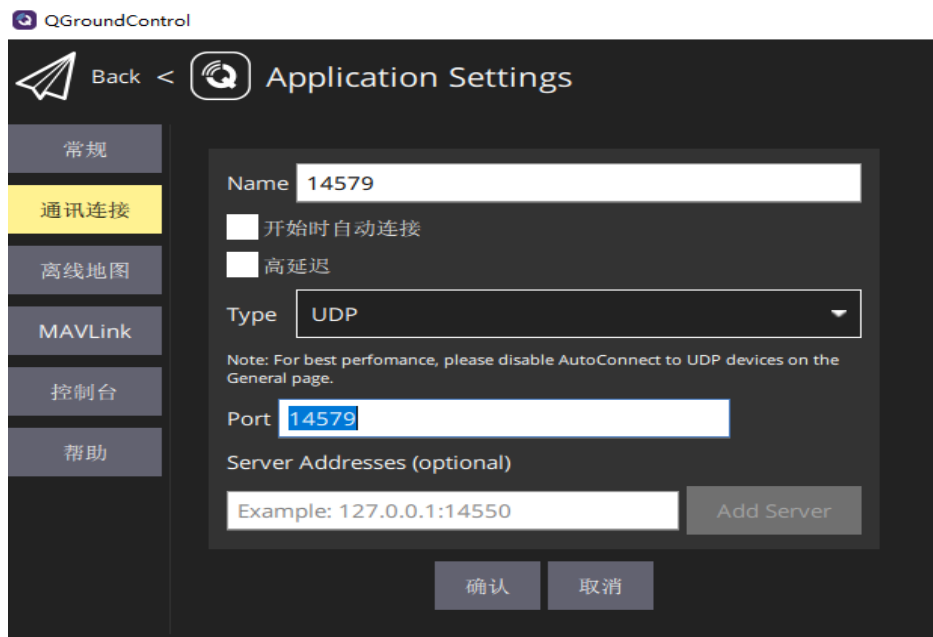


图 1.2.8 通讯连接端口建立

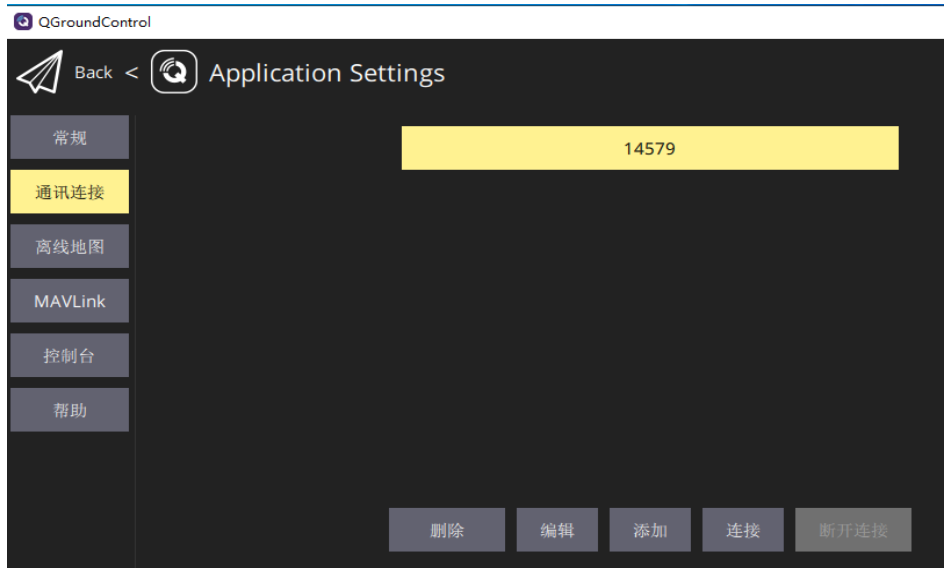


图 1.2.9 通讯接口连接界面

连接成功后如下图 1.2.10 显示:

利用遥控器切换飞机状态看与地面站是否对应。后查看 local potion 数据与前两部的数据是否能对应。若数据能对应上则需要飞手遥控切换至定点模式，进行手飞定点实时观测飞机状态（能否定住判定标准：手飞定点飞机起飞后飞机摆动幅度在 $\pm 5\text{cm}$ 以内即可）及地面站数据，定点成功后若要程控飞行需将 135 文件对应终端关闭。

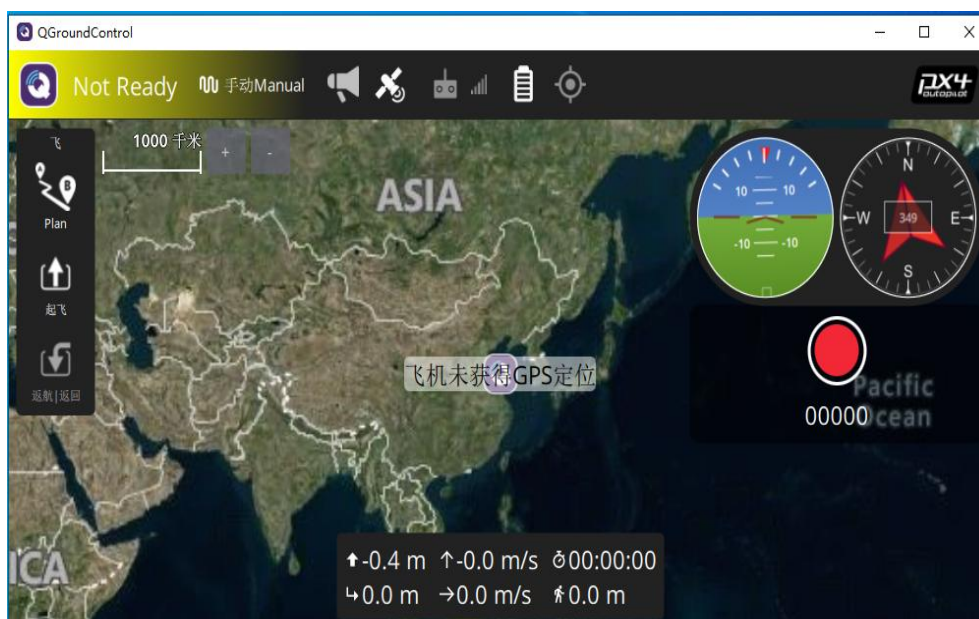


图 1.2.10 连接完成显现界面

1.3 小车飞机单组天地协同

建刚体 (Windows)

通过 Windows 图形站中的 ZVR 软件建立 1 号 200 飞机和 1 号小车的刚体，如 droneyeel 和 droneyeel1，注意：1 号小车刚体要建立编号 11，程序通过分辨 1 或 11 来判断飞机还是小车，droneyeel 为 1 号飞机，droneyeel1 为 1 号小车。判断小车刚体的 x、y、z 坐标轴是否与实际一致。单击刚体，选择属性，选择运行轨迹，输入 555，可以绘制实时轨迹，再输入 0，可以清除轨迹。单击信息可以查看该刚体的位置信息。

在场地内分别建好刚体后，然后将飞机放在小车上面的碳板上。1 号飞机对应 1 号小车，2 号飞机对应 2 号小车，3 号飞机对应 3 号小车。

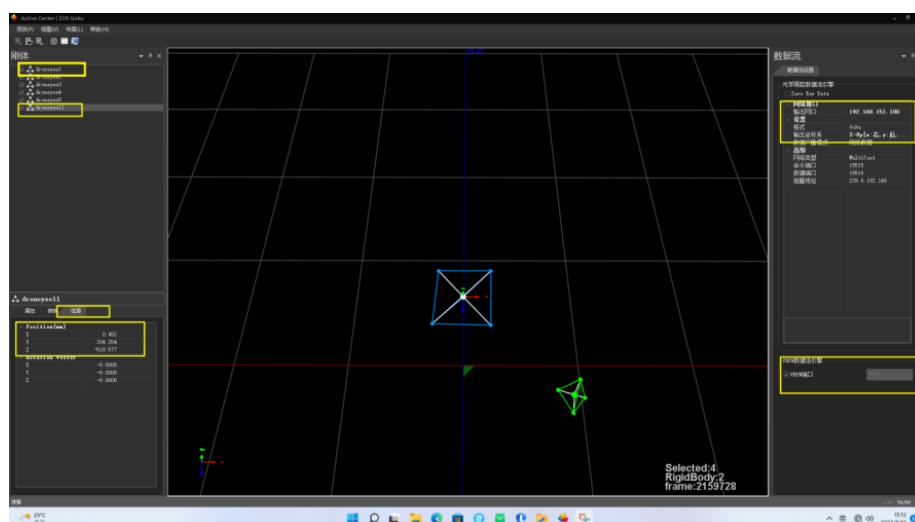


图 1.3.1 上位机软件界面

启动 simulink 程序

启动之前，可以先 ping 一下无人机（无人车），查看控制站与无人车是否通信正常，查看网络是否有延迟，通信时间为 3-5ms 时，说明通信正常，若通信延迟在 50ms 以上，甚至 100-200ms，或 1000ms，建议不要执行程序。对无人车进行重新上电，断电与上电之间间隔大约 30 秒左右（断电之后等一会儿再上电，给设备一个放电时间）。

（注意：当执行 python 脚本，打印信息刷新频率慢时，可以查看 ping 的延迟时间，也可以在 python 的打印信息中观察延迟时间，python 中延迟时间正常范围在 50-60ms，当延迟时间在 100-200ms 以上时，通信异常，无人机需要重新上电）。

1) 获取图形站刚体位置信息：`roslaunch vrpn_client_ros sample_test_200.launch`

`ctrl+alt+t` 打开一个终端，执行 `roslaunch vrpn_client_ros sample_test_200.launch` 命令，其中 3883 为 vrpn 端口号，192.168.151.100 为图形站 IP，`droneyeel` 和 `droneyeel1` 为图形站所建的刚体。

2) 更换话题名称：`roslaunch network_interface droneeye_vrpn_swarm_mocapCarsCopters.launch`

`ctrl+shift+t` 打开一个标签页，执行 `roslaunch network_interface droneeye_vrpn_swarm_mocapCarsCopters.launch` 命令，其中 `zvr` 为相机名称，192.168.151.168 为控制站 IP

3) 查看坐标信息

`rostopic list`

`rostopic echo /droneyeel/mavros/vision_pose/pose`

4) 查看小车和飞机状态

与查看飞机状态相同

①执行 `roscd droneeyecopters/launch` 命令

执行 `gedit 135_only.launch` 命令

修改 135_only 文件参数，与飞机对应

②执行 `roslaunch droneeyecopters 135_only.launch` 命令。

飞机检查完成后，再根据相同操作检查小车状态。

③连接 QGC

选择端口号进行连接，查看飞机姿态和位置信息是否正确，切换模式是否正常，且没有报错，若以上正常，手飞一下定点，若能定住，说明飞机状态正常。

选择端口号进行连接，查看小车姿态和位置信息是否正确，切换模式是否正

常，且没有报错，若以上正常，说明小车状态正常

5) 程控 (matlab+python)

①打开一个终端，输入 matlab 命令，启动 matlab

②点击“运行”，点击“更换文件夹”

③选择 trjufc.m 文件，查看运行轨迹是否为主从跟随轨迹（即飞机跟随小车，天地协同）

④选择 SIMULINK_U_f_C.slx 文件，双击打开，进入 simulink 界面

⑤在 python 脚本路径下，鼠标右键打开一个终端，执行 python MavRouter_standard_car.py 命令，python MavRouter_standard.py 命令（注：天地协同时，小车和飞机的 python 脚本都要执行），等待 local pose 的值与图形站数值相似。

⑥再次进入 simulink 界面，左侧旋钮置于 none，右侧旋钮置于 land，然后点击“运行”，缓存结束后，依次点击左侧旋钮的 off board--arm，再点击右侧旋钮的 hover，飞机悬停，悬停稳定后，点击 follow，飞机跟随小车运行小车的轨迹，运行轨迹两周后，点击右侧旋钮 hover 悬停，悬停稳定后，点击 land 降落，再点击左侧旋钮 none 停止，再点击菜单栏中的停止，结束运行。

然后进入 python 运行终端，按 ctrlz+c 结束 python 运行

2. 自主导航无人机科研平台

基于激光定位技术，实现无人机自主定位导航；应用：电力巡检、应急响应、目标检测、教学科研平台等。如图 1.1，1.2 所示。



图 1.1 1 号无人机 1 号



图 1.2 2 号无人机

2.1 一号四旋翼无人机

2.1.1 功能介绍

轴距 310mm，具备防护设计，最大飞行时间约 15 分钟。系统集成激光雷达（Mid360）与 SLAM 算法，实现了在无 GPS 环境下的实时定位与地图构建；借助 Nvidia Xavier NX 机载计算机，一方面运行 A* 与 Ego-Planner 等路径规划算法，完成自主导航任务，另一方面部署 YOLO v8 目标检测模型，赋予无人机对环境目标的实时识别能力。此外，平台还配备了 Intel D435i 深度相机以增强环境感知，并由 6S 5300mAh 电池系统供电，支持手动遥控与自主导航两种工作模式。

无人机的自主导航功能基于激光雷达实现。通过激光 SLAM 技术，系统首先完成在无 GPS 环境下的实时定位与地图构建。所构建的地图经由 OctoMap 算法进行处理，该算法利用八叉树结构高效地表达三维空间的占据、自由与未知区域，形成适用于导航与避障的语义化环境模型。在此基础之上，全局路径规划由 A*算法负责，它通过启发式搜索在离散化的网格地图中计算出全局最优路径；而 Ego-Planner 则作为局部规划器，通过以无人机为中心的优化策略，在飞行过程中实时处理未知障碍物，生成平滑、安全的局部轨迹，从而确保在复杂环境中的高动态避障能力。

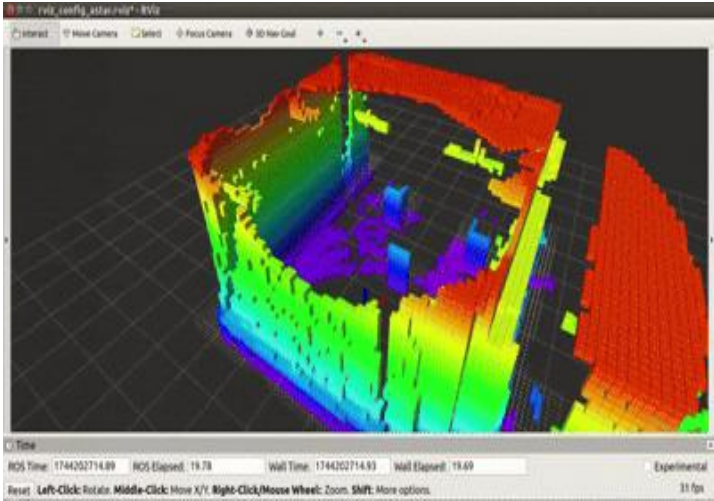


图 2.3 Slam 建图

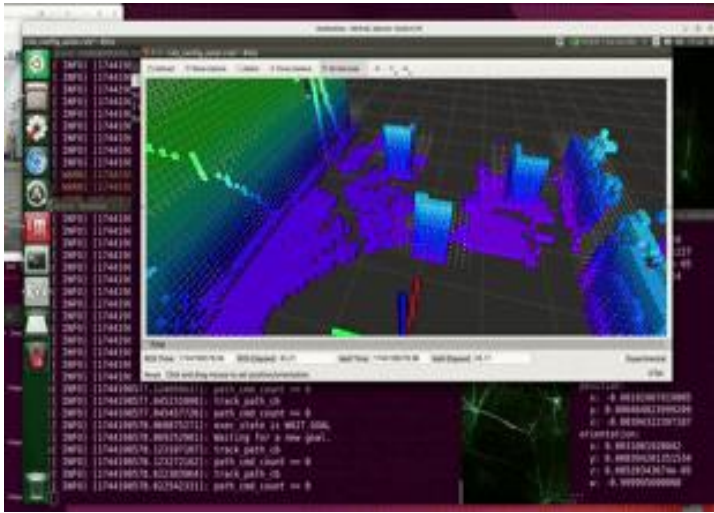


图 2.4 A*路径规划

无人机利用搭载的深度相机，基于 YOLO（You Only Look Once）算法实现高效的目标检测功能。该算法采用单阶段检测架构，将检测任务转化为回归问题，通过单次神经网络前向传播即可快速、精准地识别图像中的物体并定位，兼具高速度与高精度的特点。在具体应用中，此项能力服务于两大核心功能：在任务执行层面，无人机可实时分析地面图像，用以识别搜救任务中的受困人员、农业监

测中的病虫害区域或安防监控中的特定目标；在飞行安全层面，YOLO 能够实时检测飞行路径上的障碍物，为自主导航与避障系统提供关键的环境感知信息。

2.2.2 功能使用

（1）电池使用

连接电源之前确保电源单芯电压在 3.7V 以上（6s 电源输出电压=3.7×6），连接 BB 响，设置 BB 响电压阈值为 3.7V，如 BB 响报警说明电压降低到 3.7V 以下，需等待电源恢复常温后充电。



图 2.5 电源连接方式

（2）拔掉飞控与板卡间的连接排线。

注：板卡与飞控通过串口连接，板卡启动时串口不可以有数据输入。上电之前先将飞控连接线拔掉再进行上电。

（3）连接电源使飞控与板卡启动。

（4）使用 NoMachine/SSH 进入板卡。

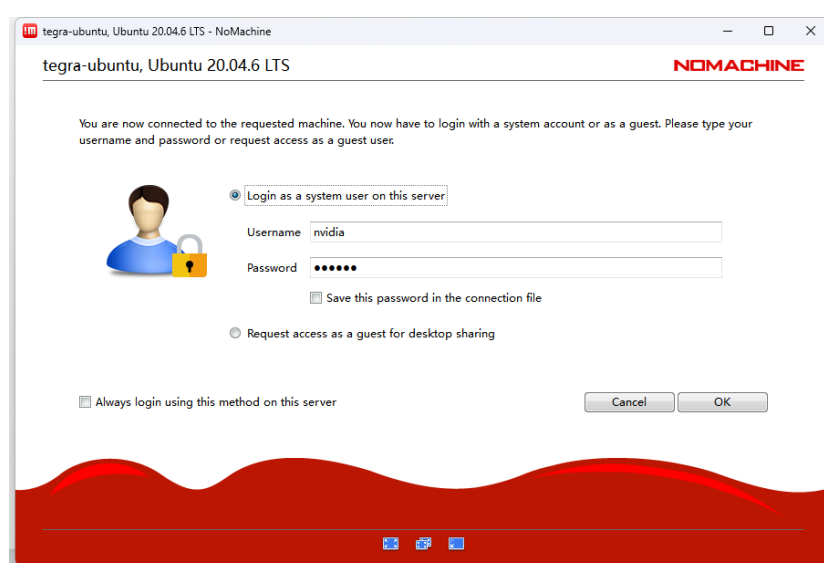
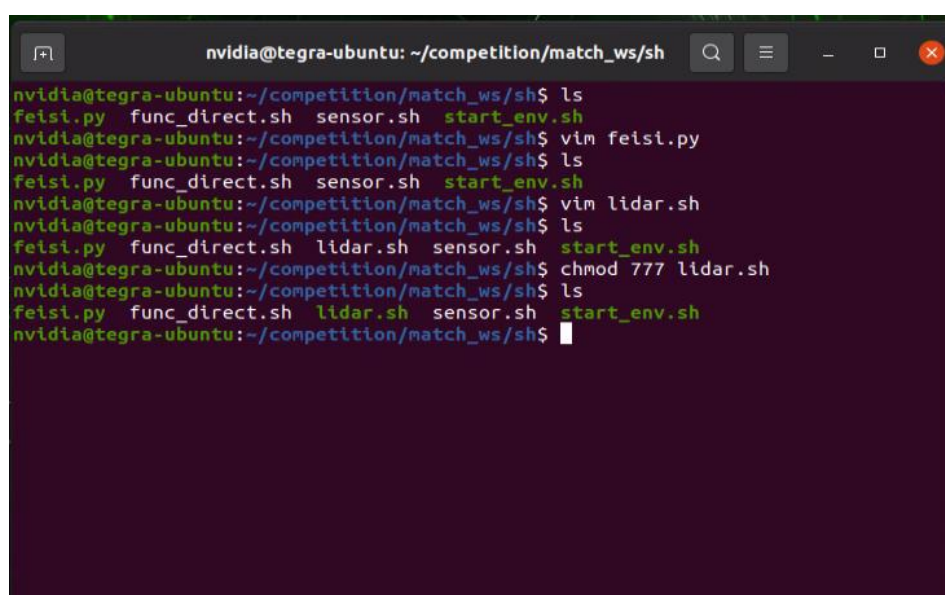


图 2.6 进入 NoMachine

（5）进入 competition/match_ws/sh 文件夹下运行 `chmod 777 lidar.sh` 启动雷

达，再运行 start_env.sh 启动所有传感器。



```
nvidia@tegra-ubuntu: ~/competition/match_ws/sh
nvidia@tegra-ubuntu:~/competition/match_ws/sh$ ls
feisi.py  func_direct.sh  sensor.sh  start_env.sh
nvidia@tegra-ubuntu:~/competition/match_ws/sh$ vim feisi.py
nvidia@tegra-ubuntu:~/competition/match_ws/sh$ ls
feisi.py  func_direct.sh  sensor.sh  start_env.sh
nvidia@tegra-ubuntu:~/competition/match_ws/sh$ vim lidar.sh
nvidia@tegra-ubuntu:~/competition/match_ws/sh$ ls
feisi.py  func_direct.sh  lidar.sh  sensor.sh  start_env.sh
nvidia@tegra-ubuntu:~/competition/match_ws/sh$ chmod 777 lidar.sh
nvidia@tegra-ubuntu:~/competition/match_ws/sh$ ls
feisi.py  func_direct.sh  lidar.sh  sensor.sh  start_env.sh
nvidia@tegra-ubuntu:~/competition/match_ws/sh$
```

图 2.7 运行程序

(6) 运行 feisi.py，启动飞思比赛功能模块。

2.2 二号四旋翼无人机

2.2.1 功能介绍

(参考一号无人机)

2.2.2 功能使用

无人机解锁与一号不同，一号无人机为内八解锁，二号无人机为油门打到右下角解锁。

(参考一号无人机)

3. 自主导航无人车科研平台

3.1 功能介绍

基于激光定位技术，实现无人车自主定位导航；应用：物流搬运、应急响应、教学科研平台等。

本自主导航无人车科研平台是一款非全向差速移动机器人，采用后轮麦克纳姆轮结构，整车尺寸为 425×230×200mm，支持手动遥控与自主导航两种运动模式。平台搭载二维激光雷达，基于 SLAM 算法实现无 GPS 环境下的实时定位与地图构建，并集成路径规划算法实现自主导航。计算核心采用 Nvidia Nano B01 机载计算机，感知系统配备奥比中光 335L 深度相机，由 10000mAh 电池供电，用户可通过专用 APP 实现对无人车的远程控制与建图过程实时监测。



图 3.1 无人车

基于激光雷达的 SLAM 技术，本平台实现了智能小车在无 GPS 环境下的高精度自主定位与导航。系统通过二维激光雷达对周围环境进行实时扫描，生成点云数据以构建环境模型；同时借助里程计连续估计小车的相对位移和转向角度，为定位提供稳定的运动先验。通过 SLAM 算法的深度融合处理，智能小车能够在移动过程中同步完成增量式地图构建和自身位姿估算，形成"感知-建图-定位"的闭环系统，最终实现完全自主的导航能力。该方案兼具激光雷达的环境感知优势与里程计的运动推算稳定性，有效保障了在复杂场景下的定位可靠性。



图 3.2 障碍物

3.2 功能使用

(1) 打开 NoMachine，连接同局域网。

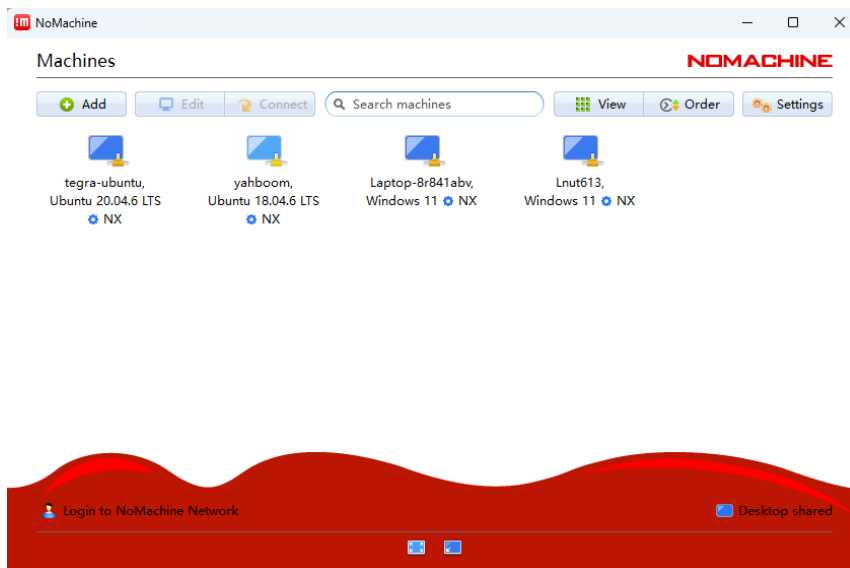


图 3.3 NoMachine

- (2) 选择对应设备，输入账号密码 jetson，yahboom，点击 OK 进入。
- (3) 输入 `cd ydlidar_ws →→ roslaunch lingao_bringup auto_nav.launch`，启动完毕。

```

/home/jetson/ydlidar_ws/src/lingao_bringup/launch/auto_nav.launch http://192.168.99.17:11
jetson@yahboom:~$ cd ydlidar_ws
jetson@yahboom:~/ydlidar_ws$ ls
build  devel  drone_follower.py  feisl.sh  scripts  src  trick.py
jetson@yahboom:~/ydlidar_ws$ cd src
jetson@yahboom:~/ydlidar_ws/src$ ls
CMakeLists.txt      robot_pose_publisher
laser_scan_to_point_publisher  rrt_exploration
LICENSE              scripts
lingao_base          yahboom_app_save_map
lingao_bringup       yahboomcar_bringup
lingao_description   yahboomcar_laser
lingao_msgs          yahboomcar_msgs
navigation           yahboomcar_nav
navigation.zip       yahboom_web_savemap_interfaces
README.md            ydlidar_ros_driver-master
r12p_laser_adminstr

jetson@yahboom:~/ydlidar_ws/src$ roslaunch lingao_bringup auto_nav.launch
... logging to /home/jetson/.ros/log/49c8ae9e-9da2-11f0-ab1f-488f4cfedb3a/roslau
nch-yahboom-10206.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://192.168.99.17:44787/

```

图 3.4 启动程序

- (4) 打开终端输入 `rviz` 添加 (Add) 相应插件，工作空间下进入 `cd scripts` 里面有 Python 脚本 `Loop_nav.py`

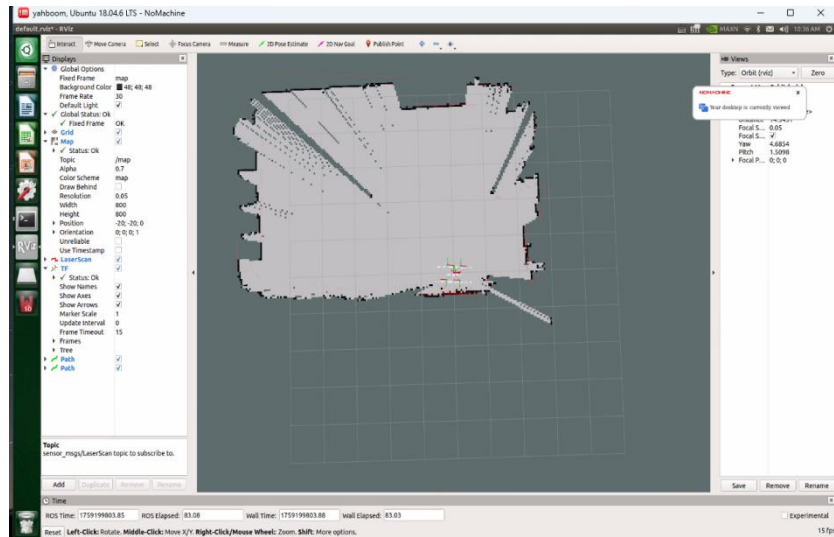


图 3.5 建图效果

(5) 点击 RVIZ 中的 2D NAV Goal 在地图中随机选择一个目标点，算法将规划出一条路径，无人车将按照路径进行运动。

四. 无人机电池管理与维护

1. 使用注意事项

(1) 在使用电池飞无人机时，先使用 BB 响测量电池电压。

单 电芯 电压值	4.2 v	3.9 5v	3.8 5v	3.7 3v	3. 5v	2.7 5v
电 量 估 计	100 %	75 %	50 %	25 %	5 %	0%

注：以上为单片电芯电压值，本实验室使用的电池规格为 4s 电池，相应电压值*4。

(2) 电池通常严禁高电压进行长期存放，因此一般使用时电压都会低于 $4.2\text{v} \times 4 = 16.8\text{v}$ 。使用前先进行充电，充满后再进行使用，禁止低电压使用。

(3) 在飞无人机时，将 BB 响挂载到电池上面，时刻关注电池电压值，BB 响警报电压值已全部设置好，不要重复设置。BB 响报警说明电池电量不足，不要继续飞行，要立即停止飞行，等待电池冷却后再进行充电。

(4) 锂电池最低电压警报值为 $3.6\text{v} \times 4 = 14.4\text{v}$ 。禁止将电池电压使用到 14.4v 以下。在电压值 $3.7\text{v} \times 4 = 14.8\text{v}$ 时应进行充电。

(5) 充电完后禁止立即使用电池，需将电池冷却后再使用。一般至少放置 10 分钟后再进行使用。

1.1 充电注意事项

(1) 电池使用完毕后，禁止立即充电，需等电池冷却后再进

行充电。一般至少放置 10 分钟。

(2) 充电时需有相关人员在现场，人走断电。禁止无人时进行充电

(3) 充电完成后，应立即拔掉电池，防止过充。

1.2 存放注意事项

(1) 电池在存放时，4s 电池电压应保持在 15.2v。低于 15.2v 应进行充电后进行存放。高于 15.2v 应进行放电进行存放。

(2) 禁止长期将电池放在无人机里面。不使用时待电池冷却后放进防爆箱内进行存放。

(3) 存放的电池电压低于 15.2v 时及时进行充电，防止过放。过充过放会导致电池鼓包、漏液、爆炸。



2. 锂电池基础知识及维护保养：

2.1 新收到电池

新电池出厂仅有少量电，不要使用，务必使用标准平衡充电器充满后再使用。平时如不急，推荐 0.5C 左右充电，例如 4000mAh 电池用 2A 左右电流充电，慢一点有利电池使用寿命。

2.2 满电和使用过程中的电压下降范围

电池充满时电压为单片 4.2v 左右（高压版为 4.35V）。用电时开始逐渐电压缓慢降低，极端用尽情况下大约为单片 3.0V，但单片电芯建议一般不能用到低于 3.6v，特殊情况不得低于 3.3v，请在类似 BB 响的电压监控设备上设置好报警阈值(总电压下限计算：2s 电池由两片电芯叠加,总电压不得低于 6.6v；3s 三片叠加，总电压不得低于 9.9v，以此类推)。过放电会导致电池损坏、鼓胀，充电器会拒绝充电，也不属质保范围。切忌放电到设备跑不动才断电，更禁止未断开设备连接、直接搁置！

2.3 长期保存注意事项

超过 24 小时不用请断开和航模及 BB 响连接，保持“半电”存放！单芯电压维持在 3.75-3.85v 较适当（如电池满电和无电就长期存放，极易鼓胀），一般充电器都有存储模式。约 45 到 60 天的周期，对电池充放激活一次，切勿长期搁置不管！