1.实验名称及目的

1.1 实验名称

RflySim平台基础视觉比赛接口实验

1.2 实验目的

- 验证平台中相机、深度相机、激光雷达等感知链路的端到端数据流（采集 -> ROS 发布 -> SLAM -> 飞控融合）。
- 理解并实践如何使用示例脚本完成感知到运动控制的闭环：使用 `main.py` 获取并转发传感器数据，使用 `StartSLAM.sh` 提供里程计，使用 `offboard.py` 或 `testRflyRosCtrl.py` 执行起飞/控制。
- 熟悉本例程的 ROS 话题、坐标系映射（ENU 与 NED 的转换）、以及基于 `PX4MavCtrlV4ROS.py` 的 MAVROS 控制接口调用方式。
- 为后续算法开发（视觉/激光 SLAM、视觉导航、避障）搭建可运行的样例环境并提供排查流程。

1.3 关键知识点（与本项目中 *.py 程序的解析）

下面以本例程中关键的 Python 脚本为中心，说明输入/输出、核心逻辑与常见失败模式（仅引用文件名）：

- `main.py`
  - 作用：连接仿真服务，读取相机、深度、IMU、点云等数据并将它们以 ROS 话题发布出来。
  - 输入：平台传感器数据（图像、点云、IMU）；输出：`/rflysim/imu`、`/rflysim/sensor*/img_rgb`、点云话题等。
  - 要点：确认 `Config.json` 中配置的传感器索引与话题对应关系；当话题不出现时，先运行 `main.py` 进行排查。
- `StartSLAM.sh`（由 SLAM 启动器调用对应 SLAM 节点）

    - 作用：启动激光 SLAM 节点，订阅点云和 IMU，产生里程计并发布到 `/mavros/odometry/out` 或 `/mavros/odometry`（依据实现）。
    - 要点：构建 SLAM 所需的驱动与工作空间需要先运行 `BuildSLAM.sh`（只需编译一次）。常见问题与修复见常见问题一节。
- `offboard.py`
    - 作用：示例位置控制程序，通过 MAVROS 接口把目标位置发送给飞控以进入 offboard 模式并起飞。
    - 输入：可选订阅的里程计/位姿；输出：位置目标到飞控。注意坐标系：MAVROS 使用 ENU，PX4 使用 NED，代码中会做坐标映射。
- `testRflyRosCtrl.py`
    - 作用：基于速度控制的示例，调用 `PX4MavCtrlV4ROS.py` 中封装的接口，展示如何读取飞控状态并发送速度命令。
    - 要点：适合快速验证控制链路（不涉及智能策略）。若飞机仅向前并偏转，说明脚本行为为预期，可在此基础上开发更复杂策略。
- `testRflyRosCV.py`
    - 作用：订阅前视/下视摄像头图像，在线程中进行图像处理，并在定时器回调中执行 10Hz 控制决策。
    - 输入/输出：订阅 `front_image_callback`、`down_image_callback` 接收图像；输出通过 MAVROS 或直接调用控制接口下发运动命令。
    - 要点：图像回调避免耗时阻塞主线程，使用线程安全队列或锁；状态机（模态切换）用于分阶段执行视觉任务。
- 与库文件的关系：`PX4MavCtrlV4ROS.py` 提供了和 MAVROS 的包装接口（信息获取、位置/速度控制），脚本通过导入该文件完成具体控制调用，建议阅读其实现理解失败场景（连接失败、话题未就绪等）。

输入/输出“契约”简述

- 输入：本例程的传感器数据（图像、点云、IMU）和飞控状态；
- 输出：ROS 话题（图像、点云、里程计、mavros 状态）以及对飞控的控制命令（起飞、位置/速度指令）；
- 错误模式：话题不出现、SLAM 不发布里程计、坐标系映射错误导致控制行为异常。

2.实验效果

完成所有步骤后，预期可以观察到的运行现象：

- 在运行 `main.py` 后，使用 `rostopic list` 能看到至少以下话题：`/rflysim/imu`、`/rflysim/sensor0/mid360_lidar`（或等效点云话题）、`/rflysim/sensor1/img_rgb`、`/rflysim/sensor2/img_rgb`。
- 在运行 `StartSLAM.sh` 并成功后，QGroundControl（QGC）中能看到视觉里程计（Odometry）和融合后的 Local_Position_NED；使用 `rostopic hz /mavros/odometry/out` 可看到约 10Hz 的里程计频率。
- 在运行 `offboard.py` 或 `testRflyRosCtrl.py` 后，飞行器应能响应解锁、起飞并执行示例命令（例如悬停或向前速度控制）。若飞行器只是向前并慢慢偏转，则是示例脚本的预期行为。
- 在运行 `StartRviz.sh` 或载入 `rflysim.rviz` 时，RViz 界面会显示点云与摄像头画面（可用于可视化调试）。

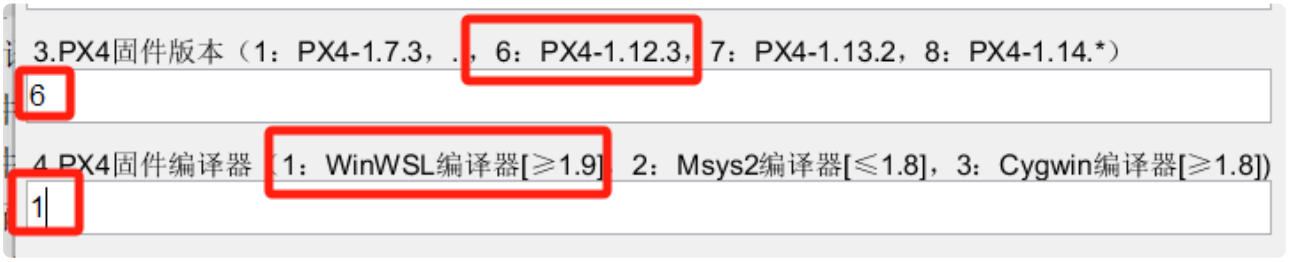
验证方法（快速检查清单）：

1. 运行 `main.py`，确认图像、IMU 与点云话题出现；
2. 运行 `StartSLAM.sh`，确认 `/mavros/odometry/out` 有数据；
3. 在 QGC 检查 Local_Position_NED；
4. 运行 `offboard.py` 或 `testRflyRosCtrl.py` 验证控制通路；
5. 若需要可视化，运行 `StartRviz.sh` 并加载 `rflysim.rviz`。

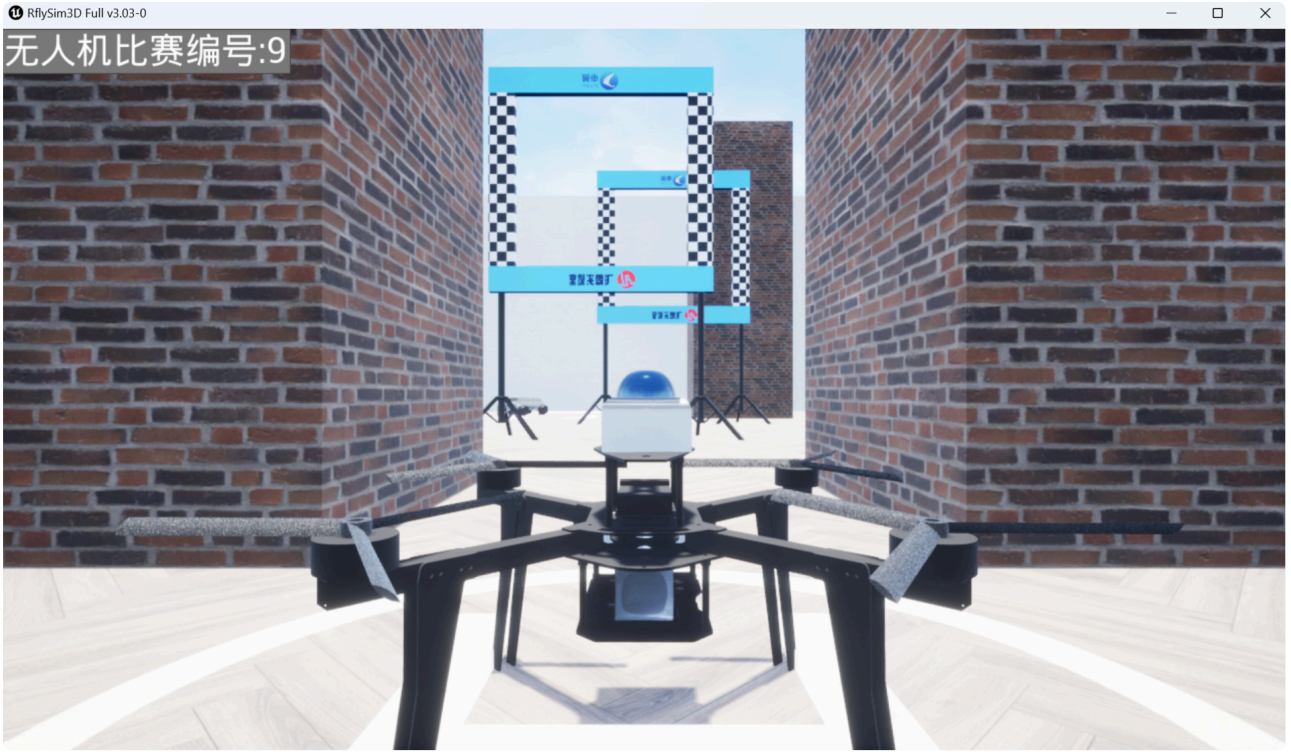
5.实验步骤

5.1 启动Windows 系统上的仿真平台

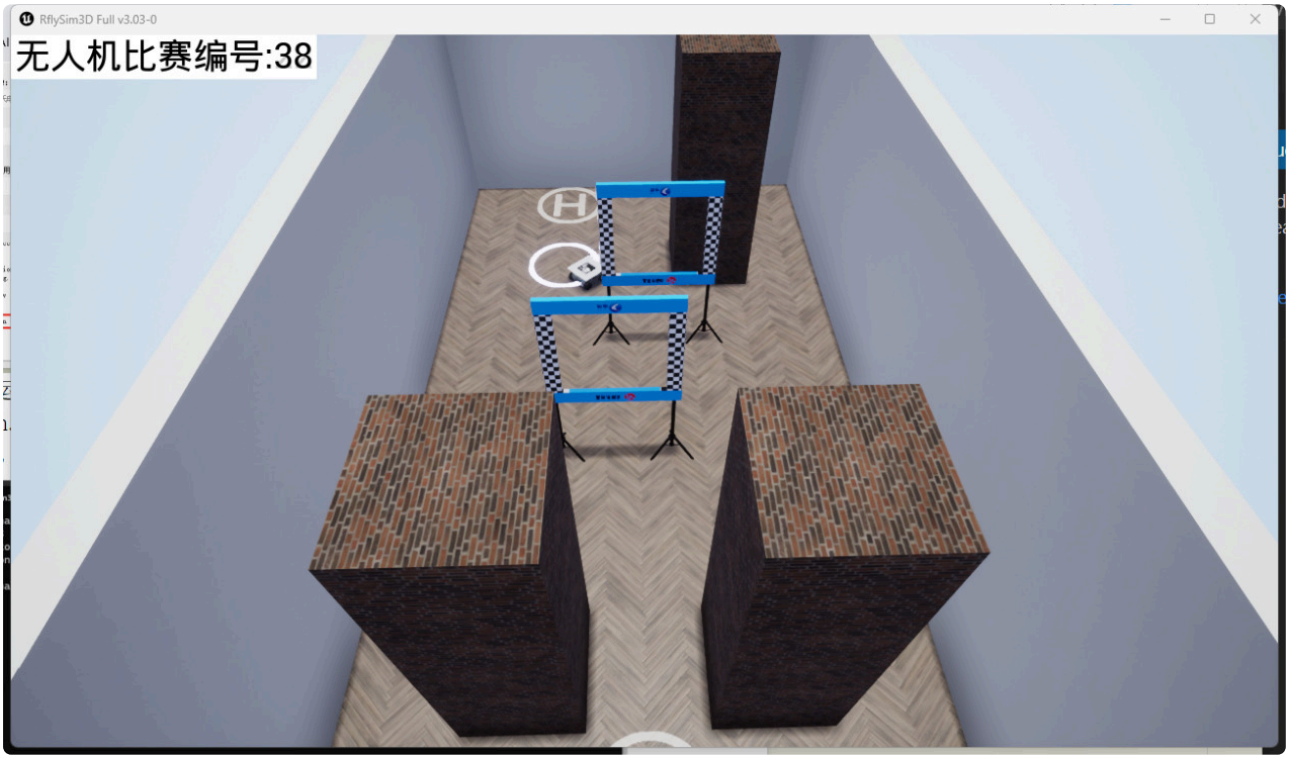
注意：本仿真例程推荐使用PX4 1.12版固件，经测试能够更稳定运行。



1. 双击运行“Windows\SITLPosStrOnekey.bat”即可完成所有配置。



三维仿真正视图

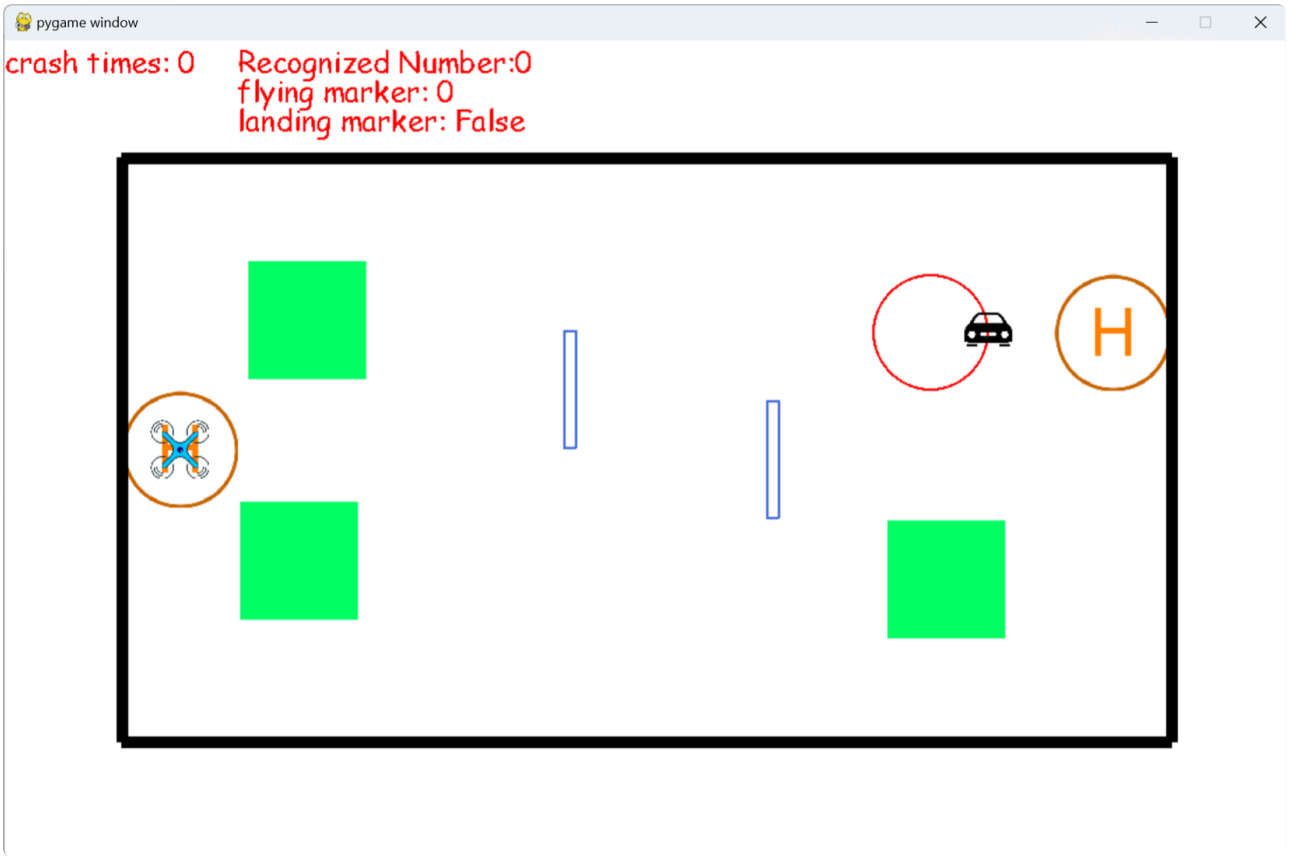


三维仿真俯视图

2. 高性能电脑用户，可开启评分程序。（选做）
   注意：对于电脑性能足够的用户（例如，台式机用户），也可以双击运行“Windows\SITLPosStrOnekeyWithScore.bat”，会启动如下图所示的一个实时轨迹显示和评分程序。

注意：最终都是人工评分，此程序仅供调试时使用，以及比赛时作为评分参考。

注意：此程序可能占用电脑性能，如果电脑性能不足，飞机起飞不稳定，容易发生炸机，可以不运行本脚本。



比赛评分程序（仅作为人工评分依据，可能占用电脑资源）

3. 算法开发。
   接着可以直接选择WinWSL/Ubuntu环境下开发算法，也可选择在虚拟机、或者外接机载板卡上开发算法，去完成比赛任务。

5.2 平台自带WinWSL/Ubuntu环境运行起飞例子

一键控制脚本测试

1. 双击运行“WinWSLRunALL.bat”，即可一键在平台WinWSL（Ubuntu 22.04）环境下运行本平台的所有程序。
2. 可以看到，依次打开了几个程序，最后飞机起飞，并向前飞，同时慢慢向右偏转。

注意：如果本例程不能顺利起飞，可能是环境配置问题，请检查VPN杀毒软件等是否关闭状态。如果飞机起飞后飞行不稳定，则是电脑性能问题，请检查任务管理器当前性能消耗，需要更换台式机来确保程序稳定运行。

3. 右键用VS Code或记事本打开“WinWSLRunALL.bat”脚本，可以发现执行了以下步骤：

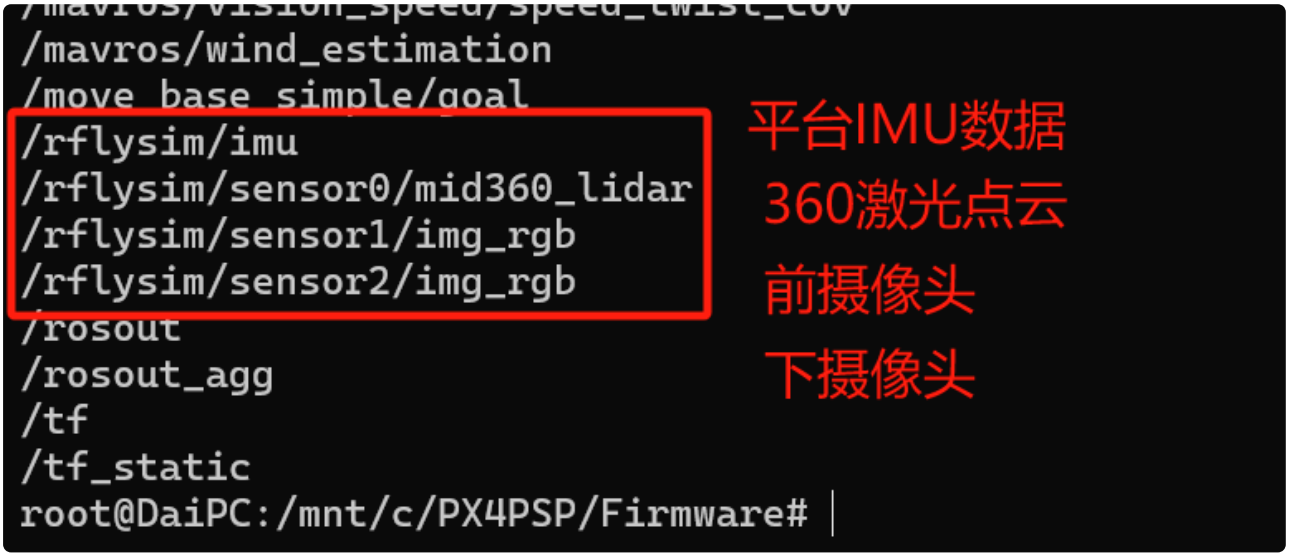
```
@echo off
echo python3 main.py
start wsl -d RflySim-20.04 -e bash -lic "python3 main.py"
choice /t 20 /d y /n >nul
echo ./StartSLAM.sh
start wsl -d RflySim-20.04 -e bash -lic "./StartSLAM.sh"
choice /t 5 /d y /n >nul
echo python3 testRflyRosCtrl.py
start wsl -d RflySim-20.04 -e bash -lic "python3 testRflyRosCtrl.py"
```

注：虽然图片脚本中名称为RflySim-20.04，实际上RflySim4.0版本开始，已启用22.04的系统，但是名称未变。

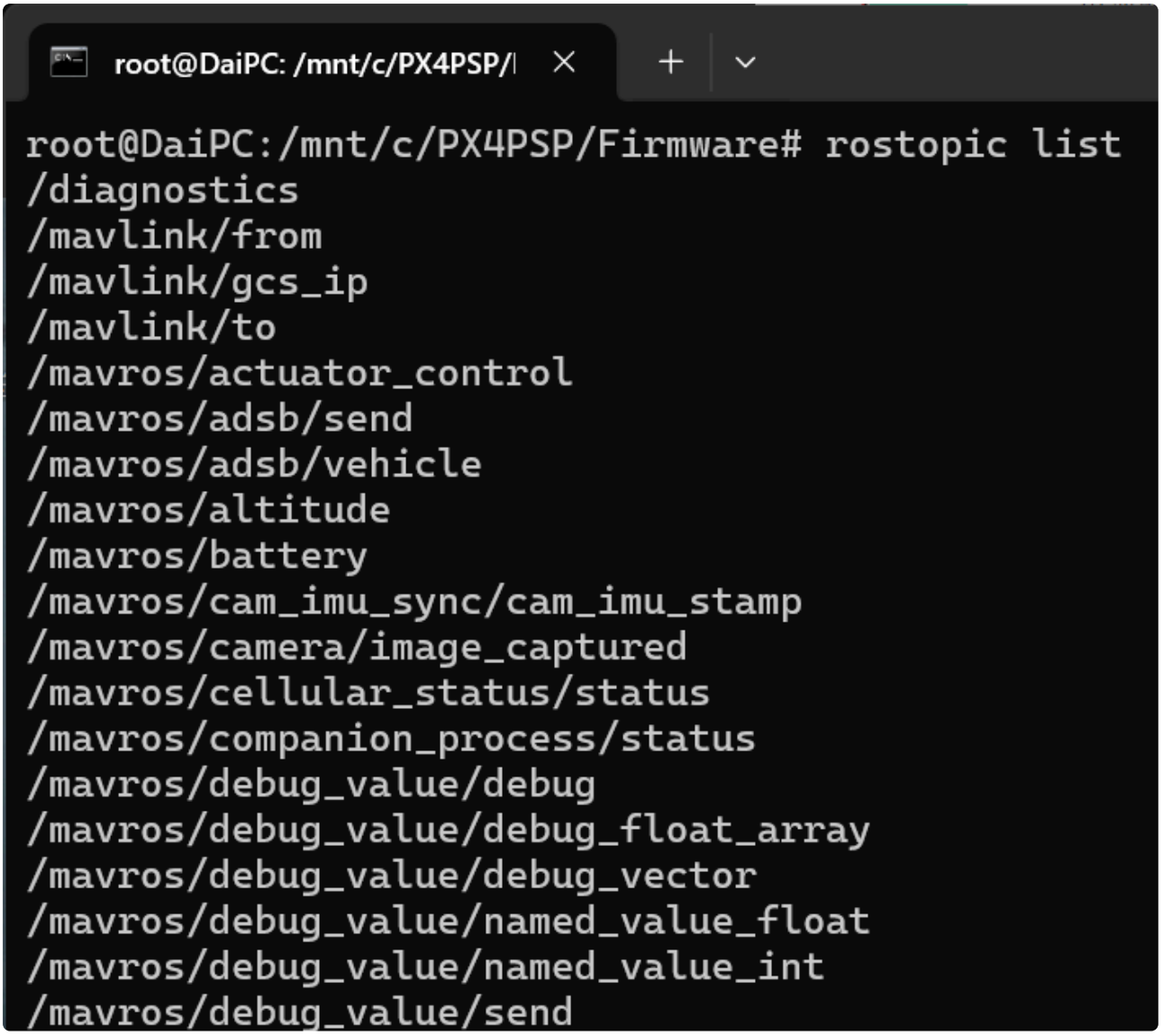
1）运行main.py程序，接收图像，转发到ROS消息。

此时双击“**WinWSL.bat**”进入平台Ubuntu环境，并输入“rostopic list”，得到如下消息列表

/rflysim/imu	IMU数据，来自CopterSim
/rflysim/sensor0/mid360_lidar	360激光点云数据，参数见Config.json
/rflysim/sensor1/img_rgb	前视RGB摄像头，参数见Config.json
/rflysim/sensor2/img_rgb	下视RGB摄像头，参数见Config.json
/mavros/odometry/out	激光SLAM发给PX4的里程计数据

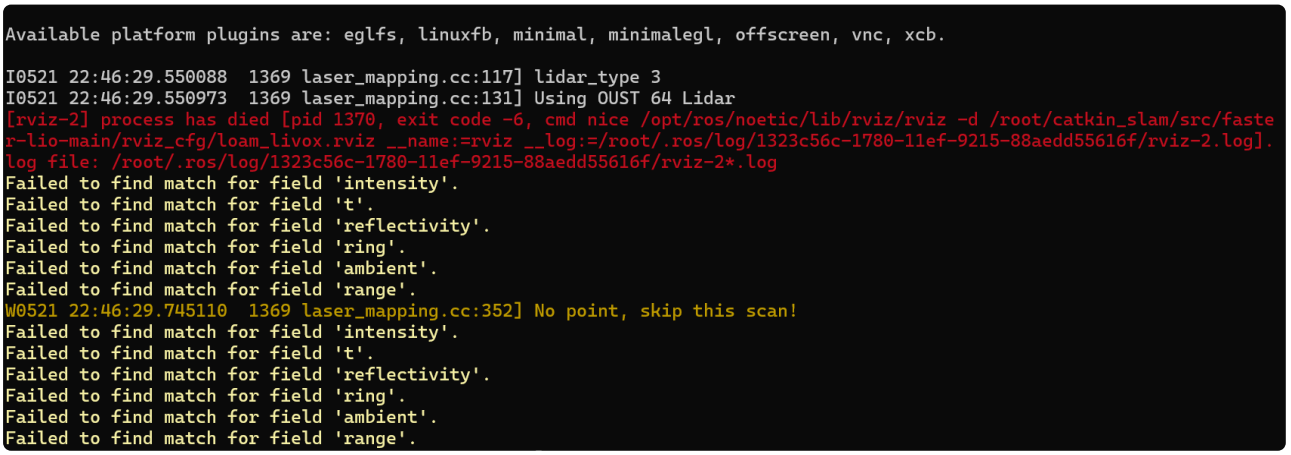


以及mavros消息



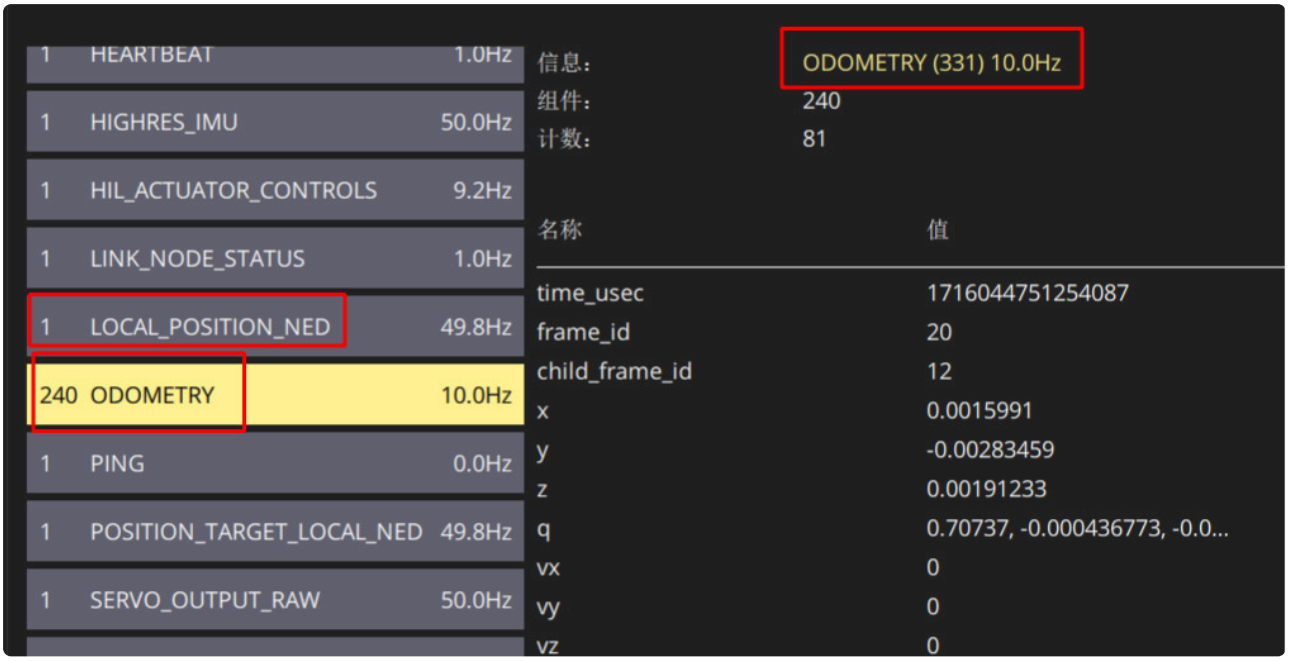
时，其他ROS节点，已经可以通过mavros订阅飞控状态，并发送起飞等控制指令了。具体的mavros控制方法，可以学习6.RflySimExtCtrl\0.ApiExps\e18_MavrosExps的例程。

2）运行StartSLAM.sh脚本，启动激光SLAM程序，得到定位数据，发给飞控。



注意：一些红色警告请忽略，可能是rviz可视化未弹出，不影响程序运行。

此时，可以在QGroudControl的
“MAVLink检测”页面中可以看到Odometry的视觉里程计消息，和Local_Position_NED的视觉融合定位数据。说明成功发送了SLAM数据，并且在飞控中进行了融合。



此时CopterSim的左下角也会显示视觉里程计的数据频率，大概10Hz。

注意：用视觉定位的方法，只能获得本地定位（Local_position），不能获得全局定位（GPS定位），因此如下图QGC提示GPS定位未获得也是正常的。

3）运行testRflyRosCtrl.py程序，订阅mavros飞控状态，通过ROS控制飞机起飞和前进

注：请手动打开testRflyRosCtrl.py了解使用原理，它的核心思想是调用了平台的ROS控制接口：RflySimAPIs\RflySimSDK\ctrl\PX4MavCtrlV4ROS.py来实现飞机起飞和速度控制。

4. 右键用VS Code或记事本打开“testRflyRosCtrl.py”，主要包含以下步骤

```
14  # 飞机起飞到0.1米前，0.8米高
15  print('发送起飞命令')
16  mav.SendPosNED(0.1,0,-0.8)
17  time.sleep(10)
18
19  print('PosE',mav.uavPosNED)
20  print('VelE',mav.uavVelNED)
21  print('Euler',mav.uavAngEular)
22  print('Quaternion',mav.uavAngQuatern)
23  print('Rate',mav.uavAngRate)
24
25  time.sleep(1)
26  print('Start control.')
27
28  # 下面开始你的控制算法
29  # mav.SendPosNED(x,y,z,yaw) 发送期望位置点，NED地球坐标系
30  # mav.SendVelNED(vx,vy,vz,yawrate) 发送速度，NED地球坐标系
31  # mav.SendVelFRD(vx,vy,vz,yawrate) 发送速度，FRD机体坐标系，
32  # mav.SendPosVelNED(PosE=[0,0,0],VelE=[0,0,0],yaw,yawrate)
33
34
35  # 下面仅展示使用SendVelFRD控制的例子，会撞墙，为正常现象
36
37  print('开始速度控制')
38
39  # 向前0.1m/s飞，并缓慢调转方向
40  mav.SendVelFRD(0.1,0,0,0.05)
```

发送起飞指令

获取飞机状态

速度+姿态控制接口

发送机体速度控制指令，飞机向前飞+向右转

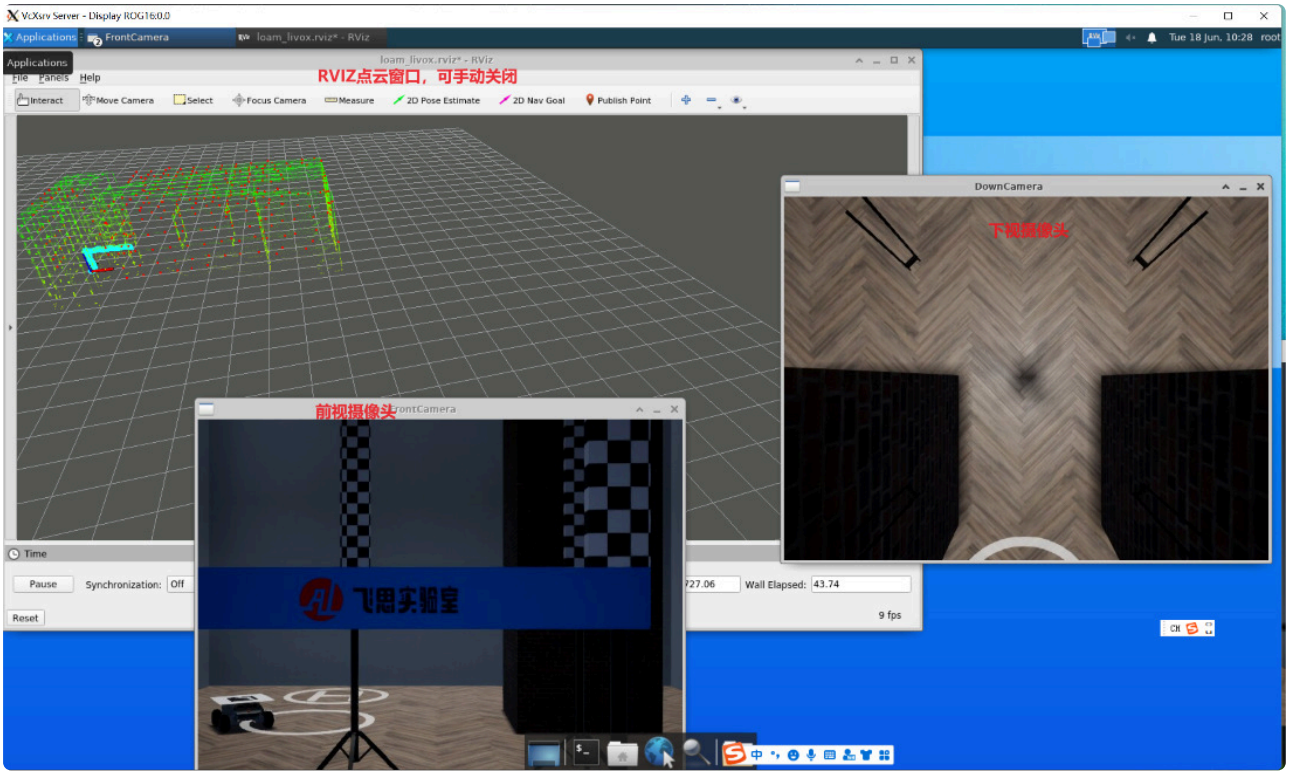
包含了获取飞机位置、姿态信息的接口展示，以及发送机体轴下的速度，进行前进飞行。用户可自行开发更复杂的控制算法。

一键视觉+控制脚本测试

关闭前面所有程序

1. 双击运行“Windows\SITLPosStrOnekey.bat”（或者，高性能台式机用户可以打开Windows\SITLPosStrOnekeyWithScore.bat，打开辅助评分程序）
2. 双击“WslGUI.bat”（仅限高性能台式机用户，笔记本用户在调试时可以打开，比赛时不要打开），打开WinWSL的图形界面，便于观察点云和图像。（注：RVIZ点云程序可能

占用较多电脑性能，如果只想观察前视和下视摄像头数据，可以手动关闭RVIZ点云窗口）。注：如果打开发现窗口白屏，没有桌面，则关了重开一两次。



注：点云和前下摄像头数据，需要运行后面程序才能出现。

3. 双击运行“WinWSLRunCV.bat”，即可一键在平台WinWSL（Ubuntu 22.04）环境下运行本平台的所有程序。右键用VS Code或记事本打开WinWSLRunCV.bat脚本，可以发现包括以下步骤：

```
echo python3 main.py
start wsl -d RflySim-20.04 -e bash -lic "python3 main.py"
choice /t 20 /d y /n >nul
echo ./StartSLAM.sh
start wsl -d RflySim-20.04 -e bash -lic "./StartSLAM.sh"
choice /t 5 /d y /n >nul
echo python3 testRflyRosCV.py
start /realtime wsl -d RflySim-20.04 -e bash -lic "python3 testRflyRosCV.py"
```

- 运行main.py程序，接收图像，转发到ROS消息。
  - 运行StartSLAM.sh脚本，启动激光SLAM程序，得到定位数据，发给飞控。
  - 运行testRflyRosCV.py程序，订阅ROS图像，通过ROS控制飞机起飞和前进

4. 右键打开“testRflyRosCV.py”可以看到关键代码如下
   - ROS订阅前置和下视摄像头数据

```
156  # 开启接收摄像头数据
157  rospy.Subscriber("/rflysim/sensor1/img_rgb", Image, front_image_callback) # 前置摄像头
158  time.sleep(1)
159  rospy.Subscriber("/rflysim/sensor2/img_rgb", Image, down_image_callback) # 下视摄像头，可以挪到后面开启
```

- 图像回调函数front_image_callback和down_image_callback，在线程中进行图像处理

```
89   # 前置摄像头订阅与处理函数
90   def front_image_callback(msg):
91       global bridge
92       global stateFlag
93       global Img1Result
94       global isShowImg
95       try:
96           cv_image = bridge.imgmsg_to_cv2(msg, desired_encoding="bgr8")
97
```

- 简单的定时器，实现10Hz的控制循环

```
163  # 下面开启一个循环，每0.1秒执行一次
164  timeInterval = 0.1
165  lastTime = time.time()
166  stateFlag =1 # 进入起飞等待悬停的模式
167  print('进入模式1')
168  while True:
169      # The above code will be executed 30Hz (0.033333s)
170      lastTime = lastTime + timeInterval
171      sleepTime = lastTime - time.time()
172      if sleepTime > 0:
173          time.sleep(sleepTime)
174      else:
175          lastTime = time.time()
176
177      # 下面的代码，每0.1秒执行一次，即10hz
```

- 简单的状态切换标志，用于实现任务阶段

```
72  stateFlag=0 # 一个状态标志位
73  # stateFlag==0 -> 飞机地面静止状态
74  # stateFlag==1 -> 飞机起飞等待到达目标高度
75  # stateFlag==2 -> 飞机起飞且到达目标高度悬停状态，开始调用
76  # stateFlag==3 -> 等待搜索到第一个框
77  # stateFlag==4 -> 搜索到第一个框，飞行靠近第一个框
78  # stateFlag==5 -> 足够接近第一个框，开始直线穿越
79  # stateFlag==6 -> 穿过第一个框，开始搜索第二个框
80  # stateFlag==7 -> 搜索到第二个框，开始靠近第一个框
81  # stateFlag==8 -> 足够接近第二个框，开始直线穿越
82  # stateFlag==9 -> 穿过第二个框，开始前往小车二维码区域
83  # stateFlag==10 -> 到达二维码区域，开始关闭前置摄像头，并
84  # ... 依次类推，完成任务链的设计
```

注：本状态切换逻辑仅供参考，需要自行设定完善

```
178
179     if stateFlag==1:
180         if abs(mav.uavPosNED[2]+0.8)<0.05: # 如果z到达-0.8米，即高度为0.8米
181             stateFlag=2 # 进入模式2，开始订阅前置摄像头数据
```

核心思想是，当前模态且满足预设条件时，转到下一个状态，用于启用下一阶段的控制或视觉处理算法。

- 起飞前简易检查程序，可以自行加入起飞阶段的校验逻辑，如果起飞不稳定，可以重启仿真。

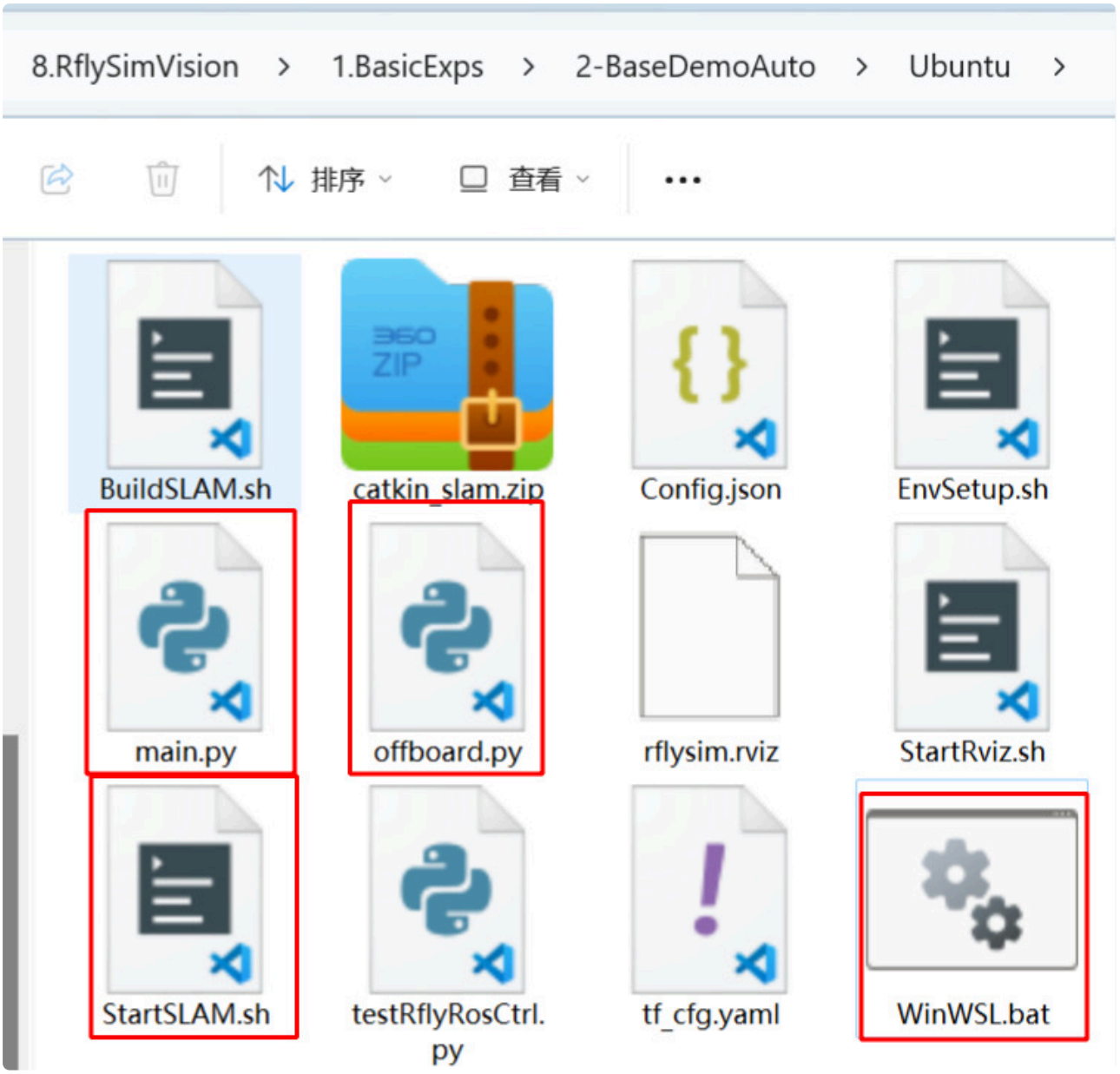
```
36  if abs(mav.uavPosNED[0])+abs(mav.uavPosNED[1])+abs(mav.uavPosNED[2]) >0.1:
37      print("当前滤波不稳定，可能导致起飞失败")
38      print("按下任意键继续，或者直接退出程序，重新仿真")
39      sys.stdin.readline()
40  else:
41      print("滤波器校验通过，可以起飞。")
42
43  # 飞机起飞到0.1米前，0.8米高
44  print('发送起飞命令')
45  mav.SendVelNED(0,0,-5)
```

手动运行程序方法

1.

在Windows下进入目录“2-BaseDemoAuto\Ubuntu”，其中：“WinWSL.bat”是进入平台WinWSL/Ubuntu环境的快捷方式，“main.py”是接收图像并转发ROS消息的中间程序，“

StartSLAM.sh”是开启激光SLAM并返回定位数据的脚本，“offboard.py”是一个基于位置控制的例子，“testRflyRosCtrl.py”是一个基于RflySIm平台接口的速度控制的例子。

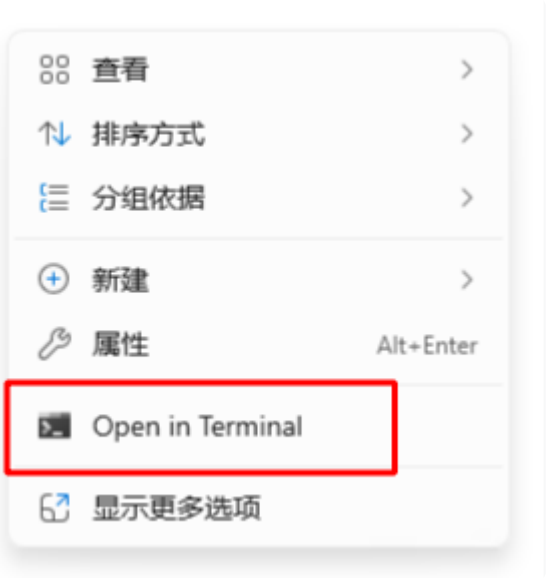


2.

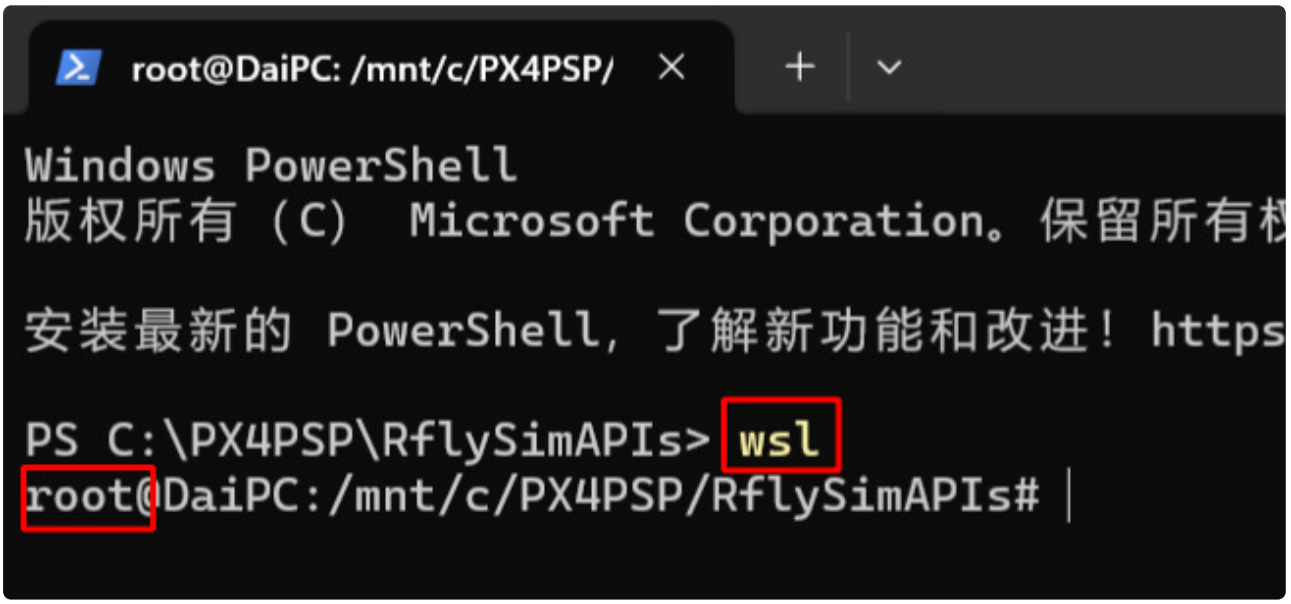
双击“**WinWSL.bat**”，可以进入平台的WSL/Ubuntu22.04环境，可以在其中模拟Ubuntu虚拟机执行各种Linux命令。

注意：平台自带的WinWSL环境，已经配置好了ROS1/ROS2、mavros、opencv、pytorch等所有环境，因此可以作为一个Ubuntu虚拟机直接运行平台所有Ubuntu下运行例程。

注意：进入任意例程文件夹，鼠标右键，点击“Open in Terminal”（在终端打开）



在弹出窗口中，输入**wsl**或**bash**，也能进入平台的WSL/Ubuntu22.04环境，特征是“root@用户名:文件目录”的命令行格式



3. 双击“**WinWSL.bat**”进入平台Ubuntu环境，并输入“**python3 main.py**”，可以开启mavors节点，并开启点云取图

```
/opt/ros/noetic/share/mavros
[ INFO] [1716215516.968883000]: CON: Got HEARTBEAT, connected. FCU: PX4 Autopilot
[ INFO] [1716215516.973546400]: IMU: Attitude quaternion IMU detected!
[ INFO] [1716215516.973679000]: IMU: High resolution IMU detected!
[ INFO] [1716215517.973958500]: GF: Using MISSION_ITEM_INT
[ INFO] [1716215517.974054900]: RP: Using MISSION_ITEM_INT
[ INFO] [1716215517.974130900]: WP: Using MISSION_ITEM_INT
[ INFO] [1716215517.974211800]: VER: 1.1: Capabilities         0x000000000000e4ef
[ INFO] [1716215517.974273900]: VER: 1.1: Flight software:     010c0300 (2e8918da66000000)
[ INFO] [1716215517.974328700]: VER: 1.1: Middleware software: 010c0300 (2e8918da66000000)
[ INFO] [1716215517.974375500]: VER: 1.1: OS software:         040400ff (bf660cba2af81f05)
[ INFO] [1716215517.974429500]: VER: 1.1: Board hardware:      00000001
[ INFO] [1716215517.974488500]: VER: 1.1: VID/PID:             0000:0000
[ INFO] [1716215517.974551000]: VER: 1.1: UID:                 4954414c44494e4f
[ WARN] [1716215517.977503000]: CMD: Unexpected command 520, result 0
Json use relative path mode
jsonPath= /mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/1.BasicExps/2-BaseDemoAuto/Ubuntu/Config.json
Got 3 vision sensors from json
Start lisening to timeStmp Msg
Got time msg from CopterSim # 1
CopterSim running on this PC
[ INFO] [1716215526.975720000]: HP: requesting home position
```

注意：如果发生了红色报错，请确认是否有之前开的python/mavros程序未关闭，关闭后，重新运行本命令。

```
Json use relative path mode
jsonPath= /mnt/e/RflySimGit/rflysimapis_new/RflySimAPIs/8.Rfl
moAuto/Ubuntu/Config.json
Got 3 vision sensors from json
Start lisening to timeStmp Msg
Got time msg from CopterSim # 1
CopterSim running on this PC
[ INFO] [1716302443.523768200]: HP: requesting home position
Got CopterSim time Data for img
Got start time for SeqID # 0
Got start time for SeqID # 1
Got start time for SeqID # 2
Start Image Reciver
Start lisening to IMU Msg
```

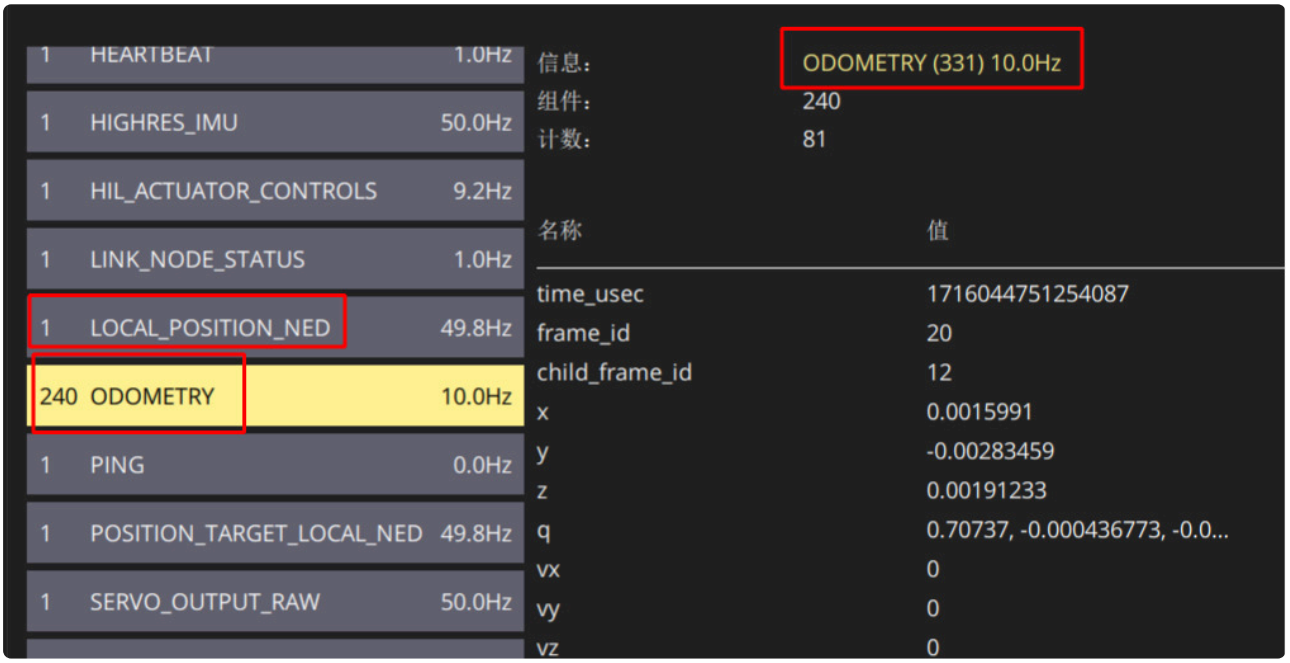
等待看到上图所示的图像和IMU发出消息，说明激光和IMU数据已经正常发送，可以执行下面的SLAM算法，解算飞机定位数据并回传飞控。

4.

再次双击“**WinWSL.bat**”进入平台Ubuntu环境，并输入“./StartSLAM.sh”，开启激光SLAM，并发送定位数据给PX4。注意：有如下黄色警告是正常的，另外还会有一条红色报错，那个也不影响程序运行。。

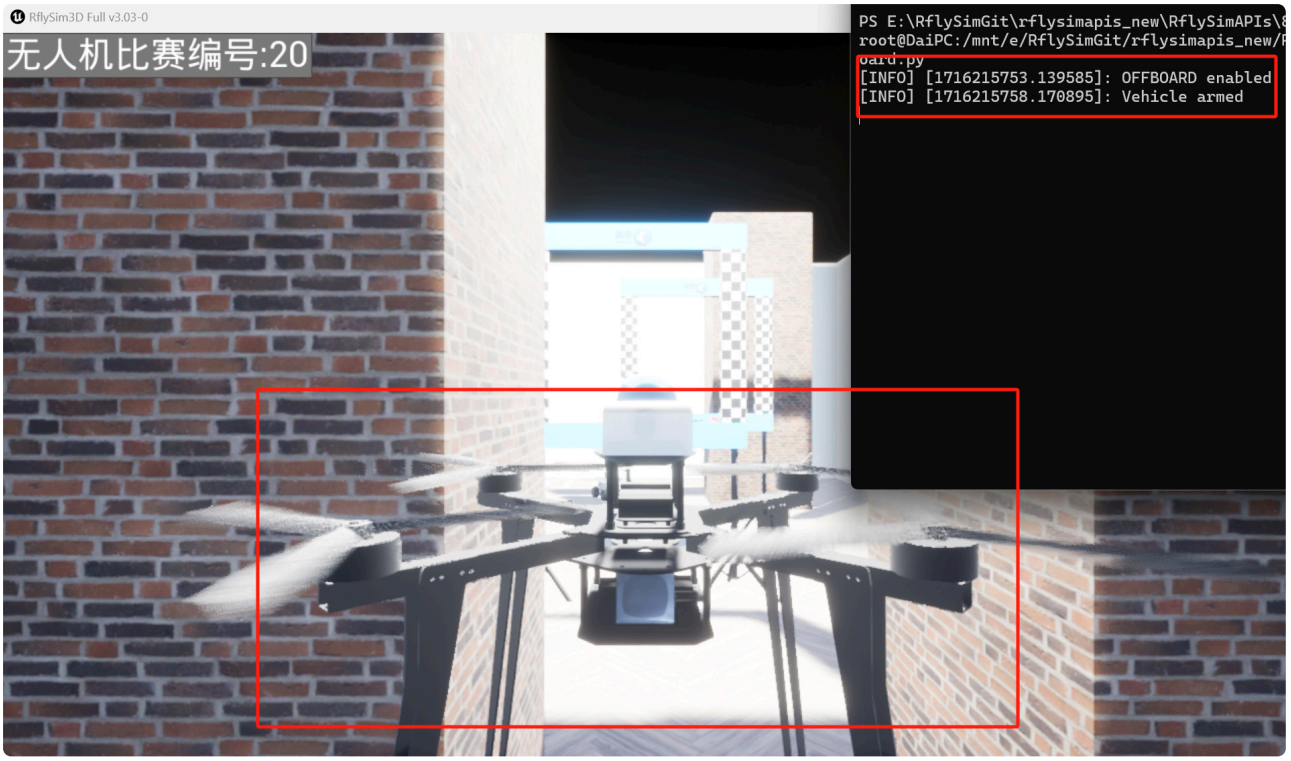
```
Available platform plugins are: eglfs, linuxfb, minimal, minimalegl, offscreen, vnc, xcb.

I0521 22:46:29.550088  1369 laser_mapping.cc:117] lidar_type 3
I0521 22:46:29.550973  1369 laser_mapping.cc:131] Using OUST 64 Lidar
[rviz-2] process has died [pid 1370, exit code -6, cmd nice /opt/ros/noetic/lib/rviz/rviz -d /root/catkin_slam/src/faste
r-lio-main/rviz_cfg/loam_livox.rviz __name:=rviz __log:=/root/.ros/log/1323c56c-1780-11ef-9215-88aedd55616f/rviz-2.log].
log file: /root/.ros/log/1323c56c-1780-11ef-9215-88aedd55616f/rviz-2*.log
Failed to find match for field 'intensity'.
Failed to find match for field 't'.
Failed to find match for field 'reflectivity'.
Failed to find match for field 'ring'.
Failed to find match for field 'ambient'.
Failed to find match for field 'range'.
W0521 22:46:29.745110  1369 laser_mapping.cc:352] No point, skip this scan!
Failed to find match for field 'intensity'.
Failed to find match for field 't'.
Failed to find match for field 'reflectivity'.
Failed to find match for field 'ring'.
Failed to find match for field 'ambient'.
Failed to find match for field 'range'.
```

此时，可以在QGroudControl的“MAVLink检测”页面中可以看到Odometry的视觉里程计消息，和Local_Position_NED的视觉融合定位数据。说明成功发送了SLAM数据，并且在飞控中进行了融合。



注意：用视觉定位的方法，只能获得本地定位（Local_position），不能获得全局定位（GPS定位），因此如下图QGC提示GPS定位未获得也是正常的。

5. 再次双击“**WinWSL.bat**”进入平台Ubuntu环境，并输入“python3 offboard.py”，可以提示进入offboard，并解锁，然后飞机起飞。



6.

基于“offboard.py”或自行编写Offboard的控制例子，并订阅图像信息，去编写算法，成功完成比赛任务即可。

注：更多mavlink、mavros或offboard控制的例子，可以参考“6.RflySimExtCtrl\0.ApiExps\e18_MavrosExps”、“8.RflySimVision\0.ApiExps\2-DistributedSimAPI\2.UavDistCtrl”或“8.RflySimVision\0.ApiExps\2-DistributedSimAPI\3.RosDistCtrl”。

7.

除了上述位置控制的例子，平台还提供了一个速度控制的例子。关闭前面所有窗口，重来一次全部流程。将最后的“python3 offboard.py”命令换成 “**python3 testRflyRosCtrl.py**”即可观察到飞机起飞后，向前飞，并慢慢向右转向。

注意：**testRflyRosCtrl.py**引用了“RflySimSDK\ctrl\PX4MavCtrlV4ROS.py”库文件，是RflySim平台提供的一个基于mavros的控制接口（兼容ROS1和ROS2）。提供了基本的信息获取和位置+速度控制接口，请自行阅读源码，了解实现原理。

```
15    # 飞机起飞到0.1米前，0.8米高
16    print('发送起飞命令')
17    mav.SendPosNED(0.1,0,-0.8)
18    time.sleep(10)
19
20    print('PosE',mav.uavPosNED)
21    print('VelE',mav.uavVelNED)
22    print('Euler',mav.uavAngEular)
23    print('Quaternion',mav.uavAngQuatern)
24    print('Rate',mav.uavAngRate)
25
26    time.sleep(1)
27    print('Start control.')
28
29    # 下面开始你的控制算法
30    # mav.SendPosNED(x,y,z,yaw) 发送期望位置点，NED地球坐标系
31    # mav.SendVelNED(vx,vy,vz,yawrate) 发送速度，NED地球坐标系
32    # mav.SendVelFRD(vx,vy,vz,yawrate) 发送速度，FRD机体坐标系，逆
33    # mav.SendPosVelNED(PosE=[0,0,0],VelE=[0,0,0],yaw,yawrate)
34
35
36    # 下面仅展示使用SendVelFRD控制的例子，会撞墙，为正常现象
37
38    print('开始速度控制')
39
40    # 向前0.1m/s飞，并缓慢调转方向
41    mav.SendVelFRD(0.1,0,0,0.05)
```

注意：本例子没有智能控制算法在里面，飞机会慢慢向前，并向右偏转，最后会撞墙，这个是正常的。可以在此基础上，研发自己的算法。

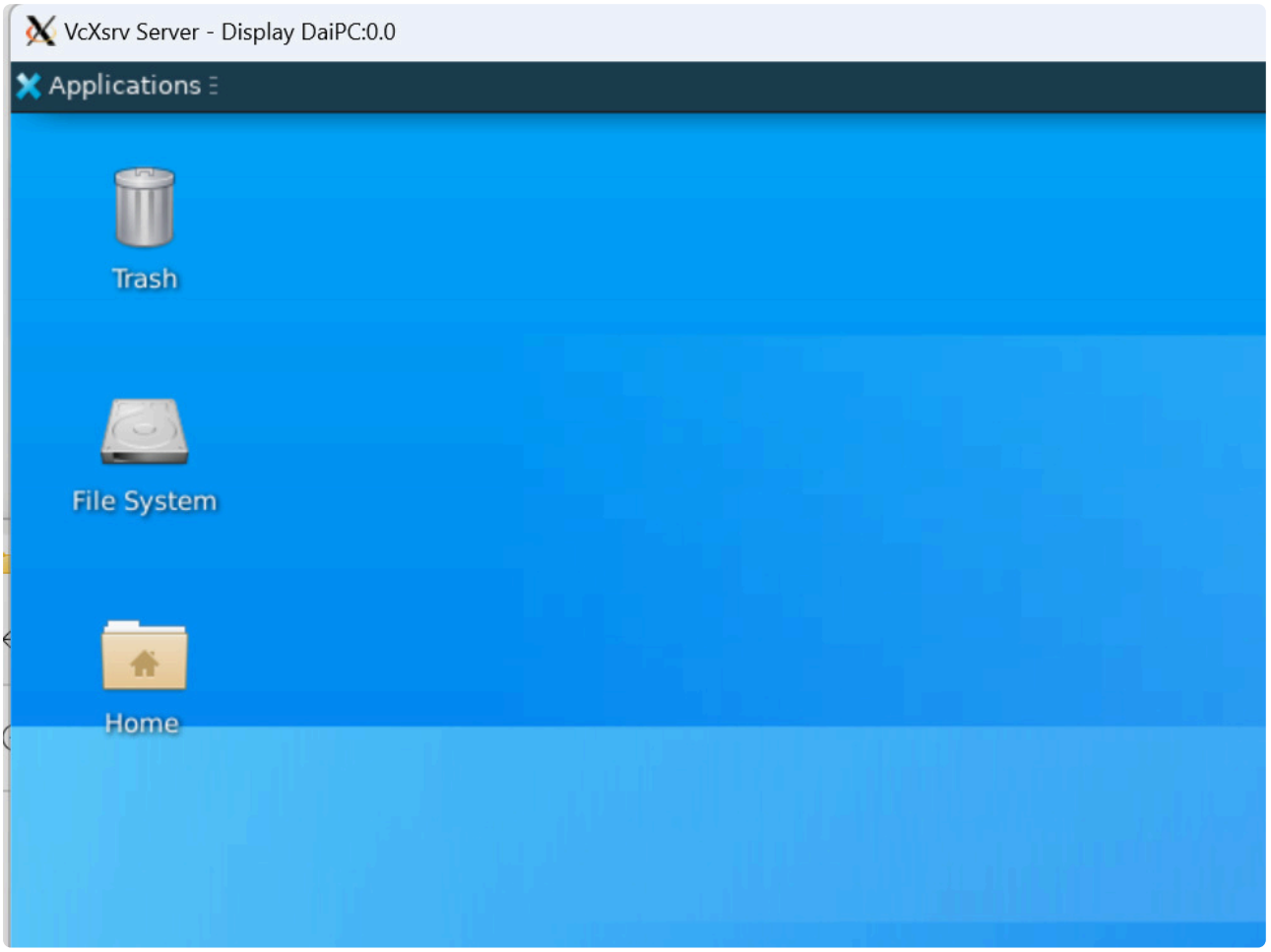
注意：也可以双击运行“WinWSLRunALL.bat”的脚本，来一键完成Ubuntu下的代码执行，它会自动依次执行“python3 main.py”、“./StartSLAM”、“python3 testRflyRosCtrl.py”，打开脚本，可见关键代码如下，可以参考自行设计自动化运行脚本：

```
start wsl -d RflySim-20.04 -e bash -lic "python3 main.py"

choice /t 20 /d y /n >nul

start wsl -d RflySim-20.04 -e bash -lic "./StartSLAM.sh"

choice /t 5 /d y /n >nul
```

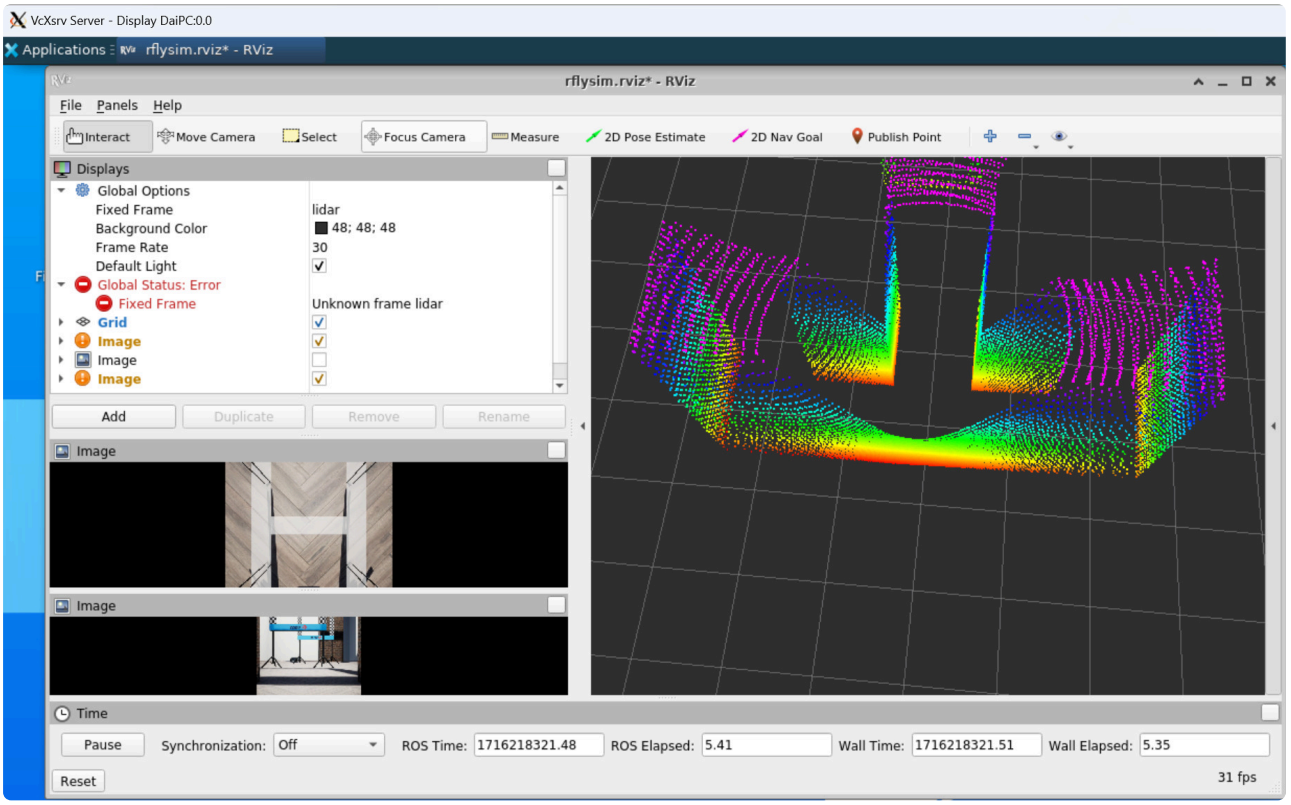
start wsl -d RflySim-20.04 -e bash -lic "python3 testRflyRosCtrl.py"

8. 注意：除了命令行的方式，平台还提供了一个图形界面，便于观察程序运行结果。

   1. 双击本文件下“**WslGUI.bat**”，或双击桌面快捷方式“RflyTools\WslGUI”，可以开启一个图形界面。

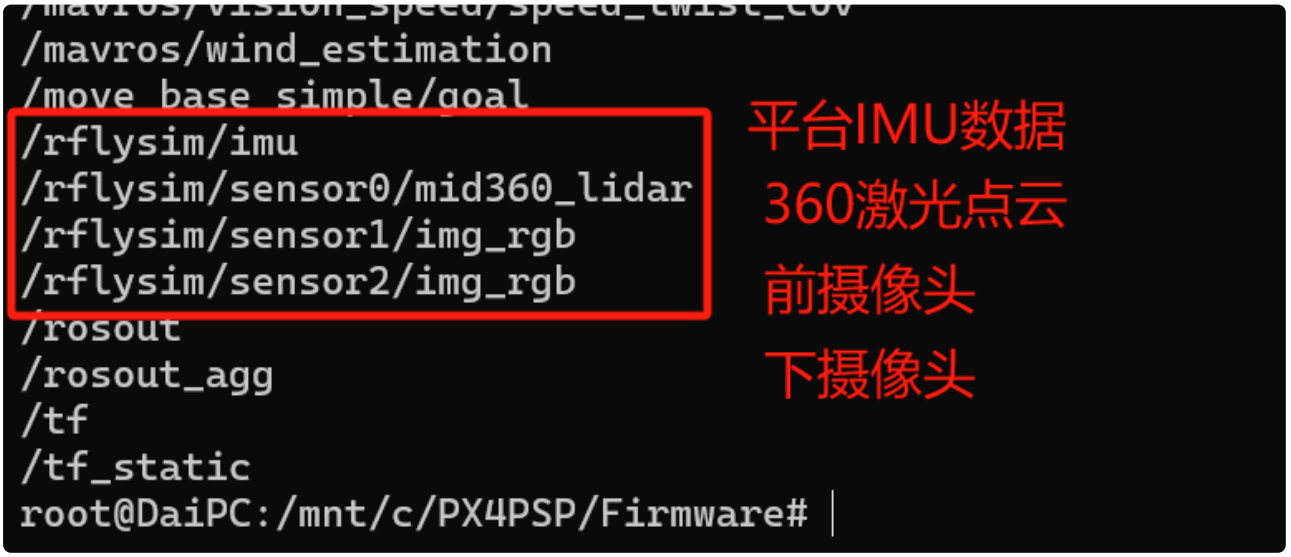


   2. 双击“**WinWSL.bat**”进入平台Ubuntu环境，并输入“./StartRviz.sh”，就可以在上面图像界面中，看到Rviz的界面，并看到点云数据。



3）双击“**WinWSL.bat**”进入平台Ubuntu环境，并输入“rostopic list”，得到如下消息列表

/rflysim/imu	**IMU数据，来自CopterSim**
/rflysim/sensor0/mid360_lidar	360激光点云数据，参数见Config.json
/rflysim/sensor1/img_rgb	前视RGB摄像头，参数见Config.json
/rflysim/sensor2/img_rgb	下视RGB摄像头，参数见Config.json
/mavros/odometry/out	激光SLAM发给PX4的里程计数据



4）更多关于WSL和GUI的功能说明和例程，可以参考“PX4PSP/RflySimAPIs/1.RflySimIntro/2.AdvExps/e7_WslGUI/Intro.pdf”

注意：上文的WinWSL环境已经能满足所有比赛开发需求，后续步骤仅限感兴趣用户。

5.3 使用RflySim GPU加速外挂环境运行实验(选做)

除了使用WinWSL编译器，还可使用RflySim基于WSL2/Ubuntu 22.04开发的外挂环境来运行本实验。本外挂环境，附带了docker、cuda、pytorch、tensorflow等额外功能，能够开发更复杂的无人系统感知决策算法。

1. 根据实验例程：`1.RflySimIntro\2.AdvExps\e13.WinWSL2-GPU\Readme.pdf` 来下载WinWSL2-GPU的外挂WSL2镜像（约50G），并部署到平台。
2. 运行`\Windows\SITLPosStrOnekey.bat`来启动无人机的SITL仿真。
3. 运行`\Ubuntu\WinWSLRunCV2.bat`来使用外挂环境运行本实验。

期望效果：和上文效果相同，飞机起飞，向前飞，并显示前下视摄像头数据。

5.4 Ubuntu 系统（包括虚拟机）程序运行（非必须）

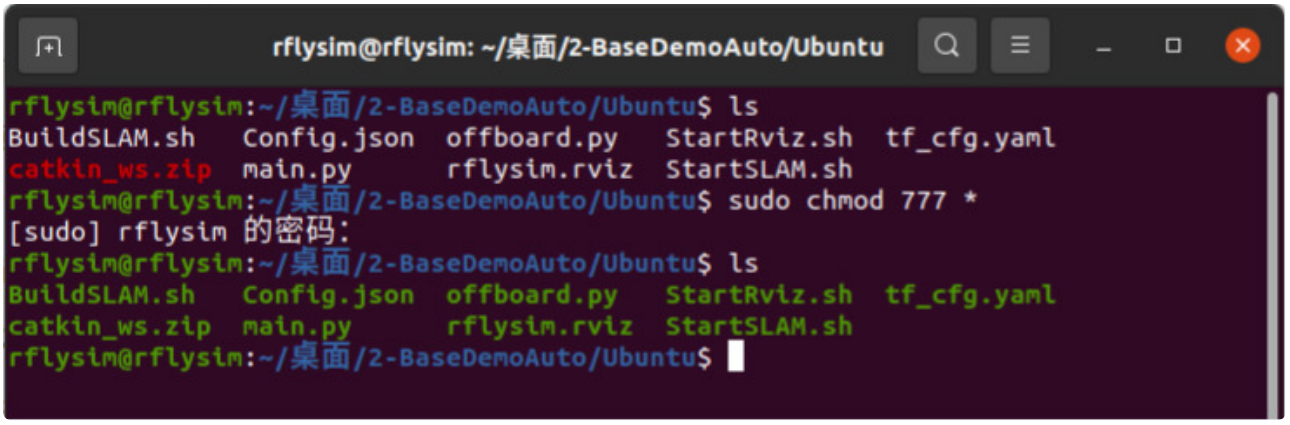
注意：如果使用官方提供的虚拟机ubuntu-20.04\rflysim.zip，已经预装好了平台运行所需要的所有环境，确保按《虚拟机使用文档.pdf》步骤正确配置虚拟机，并按后文步骤实验即可。

虚拟机下载链接：https://pan.baidu.com/s/10MDjINKG20k4mWUYz0Nm1A?pwd=78r7

实验前准备：**拷贝“PX4PSP\RflySimAPIs\RflySimSDK”到虚拟机主目录~，并在终端中运行“python3 ReLabPath.py”来导入或更新RflySim库文件**。请务必完成本步骤，再进行下面操作。

注意：由于RflySim 4.10开始，WSL2已经能完全覆盖虚拟机的需求，因此，虚拟机的方法不再维护。读者可参考其基本步骤进行配置，如果遇到问题，请联系我们。

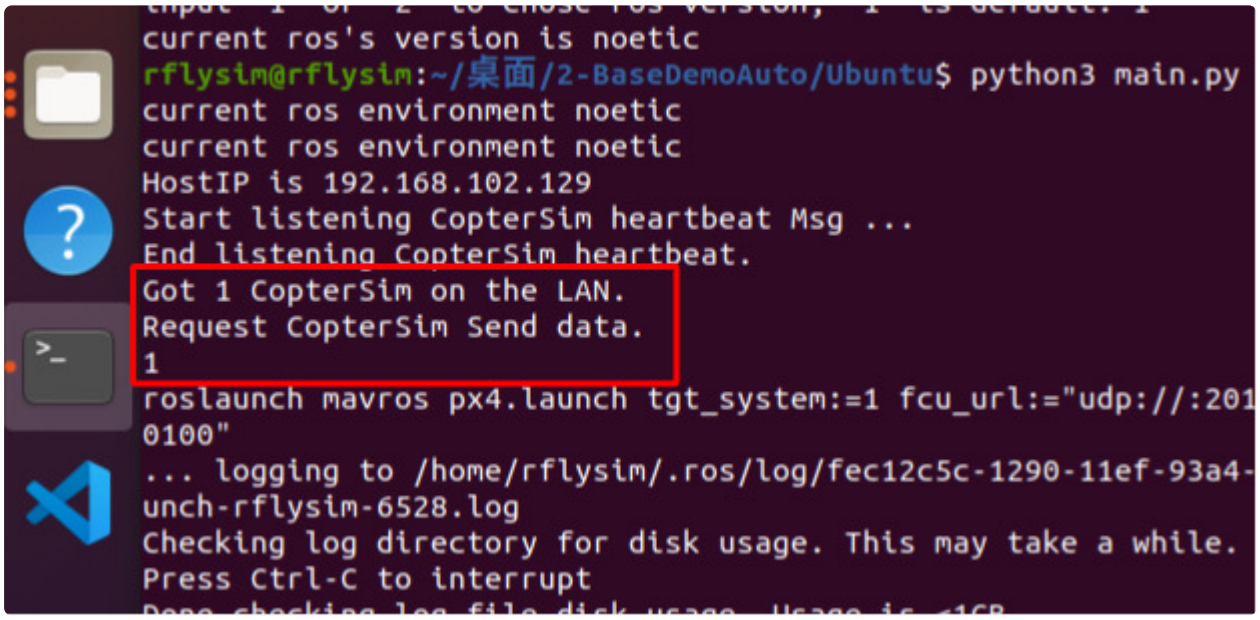
1. 将”2-BaseDemoAuto”整个拷贝到Ubuntu虚拟机中（例如桌面上），然后在Ubuntu中进入“2-BaseDemoAuto/Ubuntu”目录，鼠标右键单机并进入“终端”，然后输入如下指令“sudo chmod 777 *”，如下图所示，输入“ls”命令，可以看到文件都为绿色，说明权限正确。



注意：如果是自己的机载板卡或其他Ubuntu电脑，需要确保Ubuntu版本为20.04，然后运行“2-BaseDemoAuto/Ubuntu/EnvSetup.sh”，来安装mavros，opencv，pymavlink等一些额外包。

2. 在终端中，输入“python3 main.py”，即可完成所有配置。等待一段时间，终端中返回如下信息，说明运行正确。

1）成功获取CopterSim所在电脑的IP地址



2）成功连上飞控，并进入mavros模式，获取到了飞控版本信息（看到Board hardware，UID等信息）。

```
INFO] [1715759945.148114165]: WP: Using MISSION_ITEM_INT
INFO] [1715759945.148314800]: VER: 1.1: Capabilities         0x000000000000e4e

INFO] [1715759945.148621653]: VER: 1.1: Flight software:     010c0300 (2e8918d
66000000)
INFO] [1715759945.148859381]: VER: 1.1: Middleware software: 010c0300 (2e8918d
66000000)
INFO] [1715759945.149178396]: VER: 1.1: OS software:         040400ff (bf660cb
2af81f05)
INFO] [1715759945.149386040]: VER: 1.1: Board hardware:      00000001
INFO] [1715759945.149701974]: VER: 1.1: VID/PID:             0000:0000
INFO] [1715759945.149848701]: VER: 1.1: UID:                 4954414c44494e4f
```

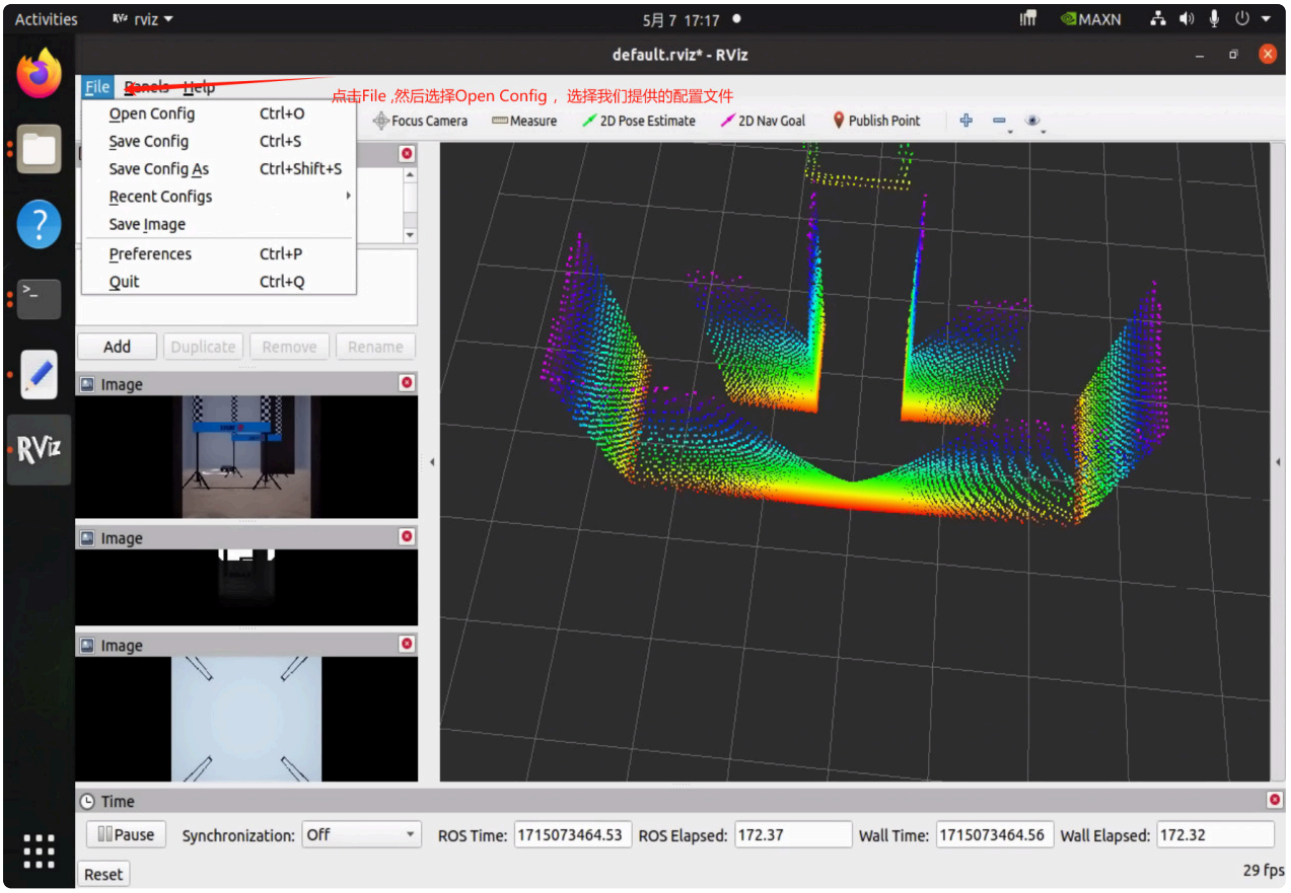
3）成功读取json并向RflySim3D请求图像，以及获取到了imu数据

```
Json use relative path mode
jsonPath= /home/rflysim/桌面/2-BaseDemoAuto/Ubuntu/Config.json
Got 3 vision sensors from json
Start lisening to timeStmp Msg
Got time msg from CopterSim # 1
CopterSim not on this PC
1715759953.552
1715759953.766975
[ INFO] [1715759954.145774317]: HP: requesting home position
Got CopterSim time Data for img
Got start time for SeqID # 0
Got start time for SeqID # 1
Got start time for SeqID # 2
Start Image Reciver
Start lisening to IMU Msg
Got CopterSim IMU Msg!
[ INFO] [1715759959.144647190]: GF: mission received
```

3. 在Ubuntu中，进入“2-BaseDemoAuto/Ubuntu”，鼠标右键启动终端，并输入命令“./StartRviz.sh”，**或者**直接输入命令“rosrun rviz rviz -d ./rflysim.rviz”
（注，两个命令二选一即可）

```
current ros's version is noetic
rflysim@rflysim:~/桌面/2-BaseDemoAuto/Ubuntu$ rosrun rviz rviz -d ./rflysim.rviz
[ INFO] [1715770602.015116105]: rviz version 1.14.20
[ INFO] [1715770602.015186333]: compiled against Qt version 5.12.8
[ INFO] [1715770602.015205211]: compiled against OGRE version 1.9.0 (Ghadamon)
[ INFO] [1715770602.082141765]: Forcing OpenGl version 0.
[ INFO] [1715770602.740202152]: Stereo is NOT SUPPORTED
```

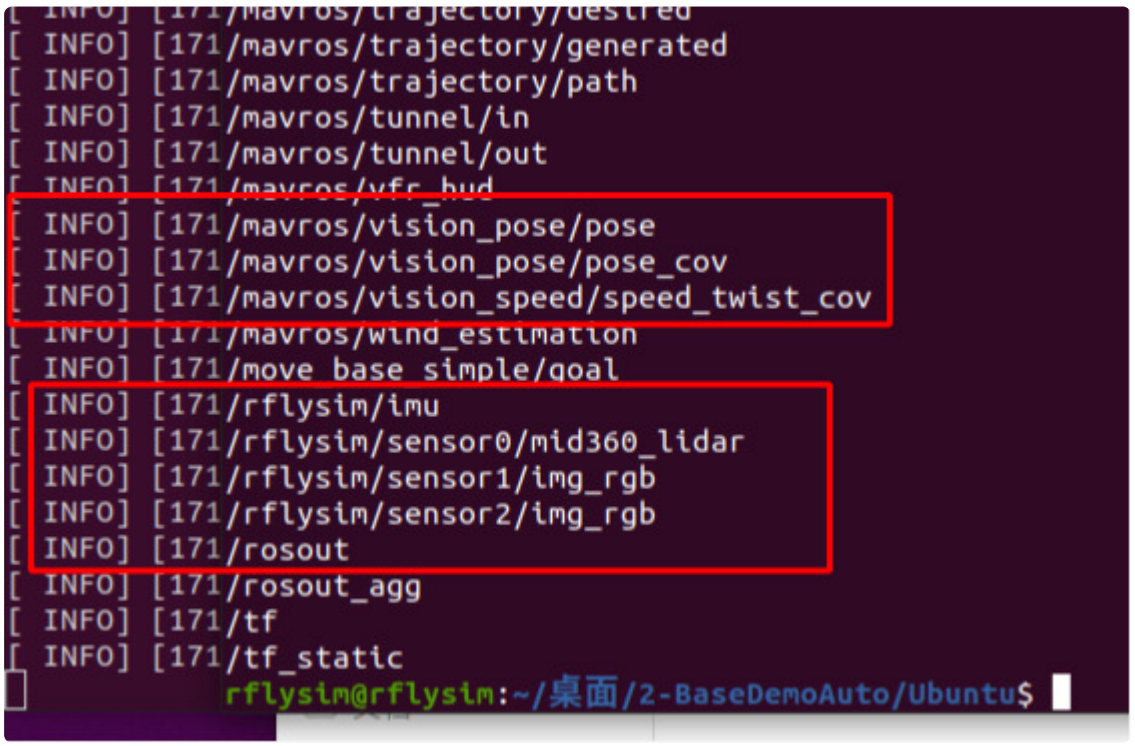
就能自动加载点云，前置摄像头，以及下视摄像头数据



能看到上面的数据，说明传感器数据获取正确。

注意：rviz的界面在某些情况下可能闪退，这个不影响程序运行。

4. ROS消息检查，打开一个终端，输入“rostopic list”命令，发现有rflysim打头的imu数据、激光点云数据，以及两个RGB话题，说明ROS配置正确。



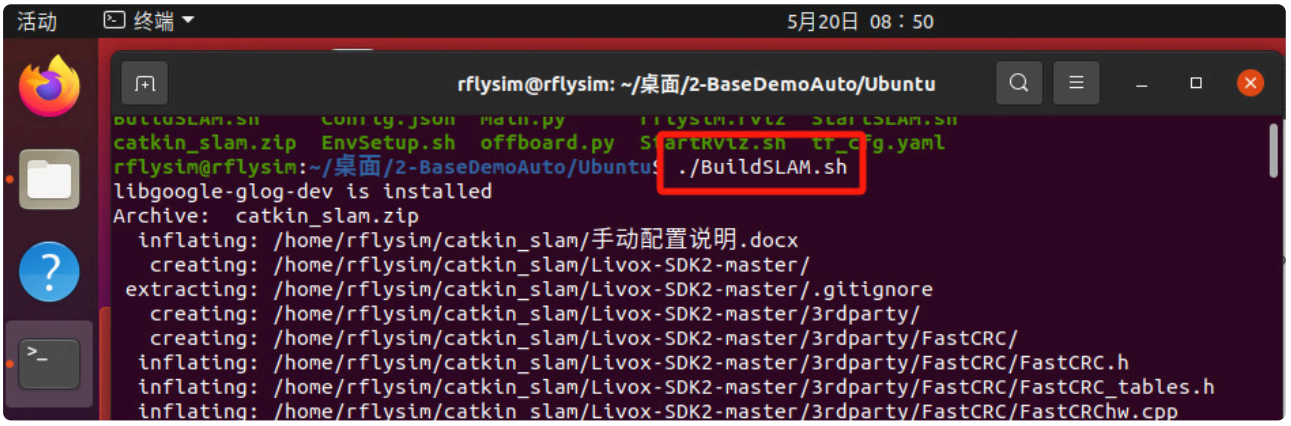
5.

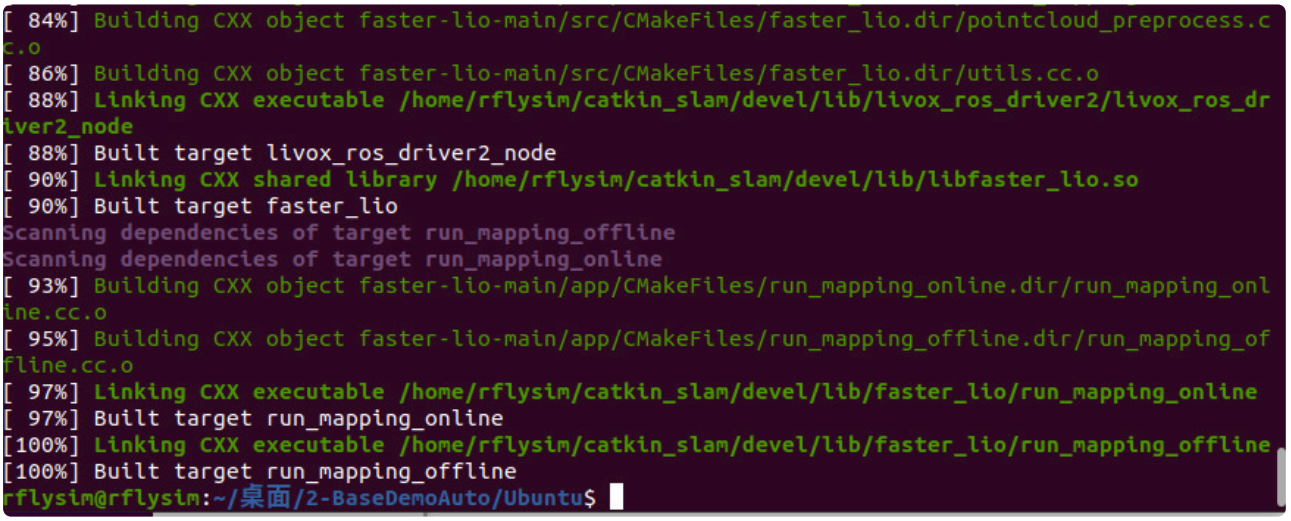
基于订阅到的点云、图像、以及IMU数据，可以做感知，然后基于mavros可以控制飞机的起飞，前进，在此基础上完成比赛任务即可。

5.5 Ubuntu 系统（包括虚拟机）SLAM起飞例子

1. 编译mid360驱动和激光SLAM源码并部署（只需要编译一次，之后就不需要再运行了）。

在Ubuntu中，进入“2-BaseDemoAuto/Ubuntu”，鼠标右键启动终端，并输入命令“./BuildSLAM.sh”即可完成所有编译工作。





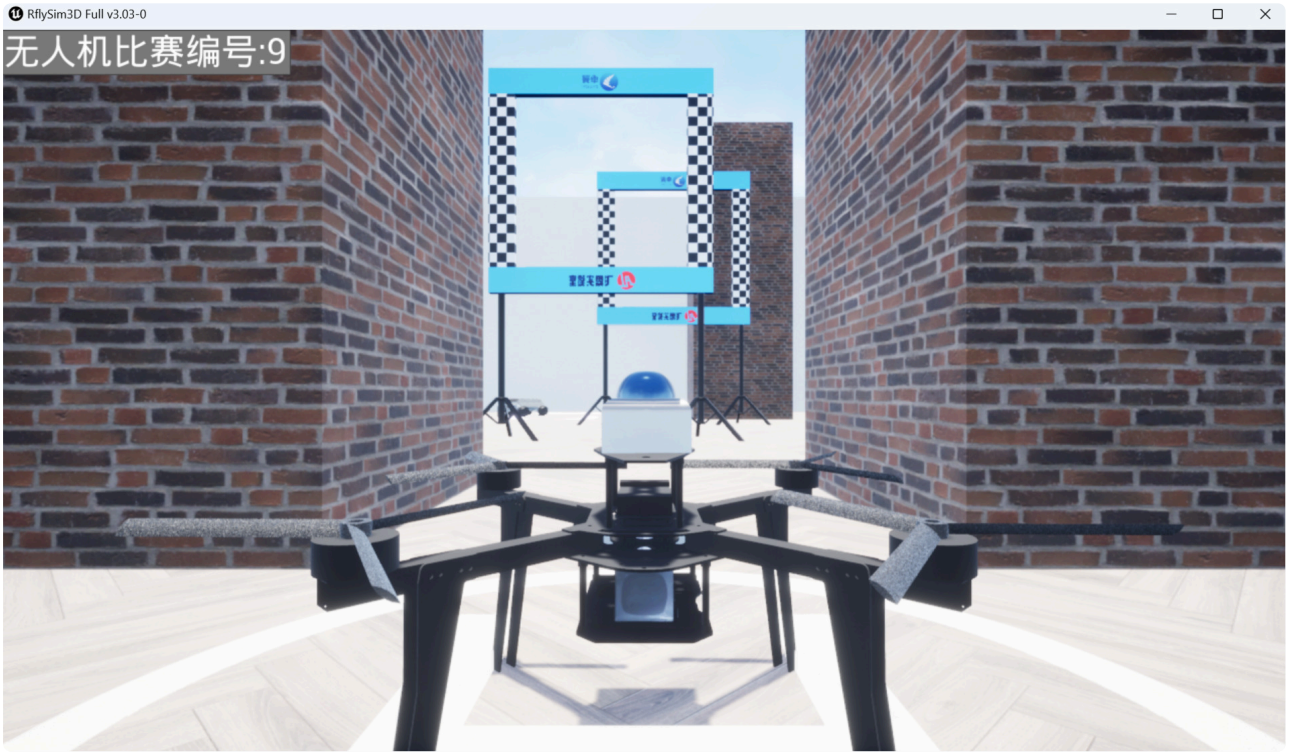
注意：请务必确保编译过程没有任何报错，且正常编译到100%的进度。

注意：本编译工作，会自动将catkin_ slam.zip解压到“主目录”下，并进行编译与部署。

注意：如果要手动配置，请参考“catkin_slam.zip\手动配置说明.docx”文档。

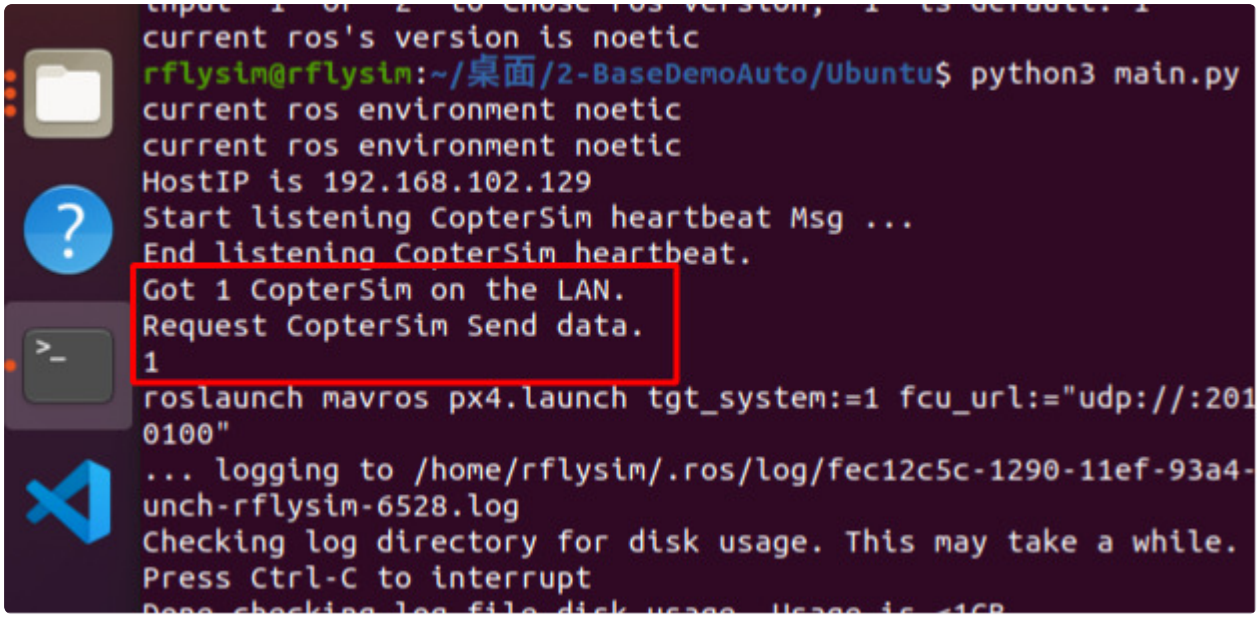
2. 实验顺序：

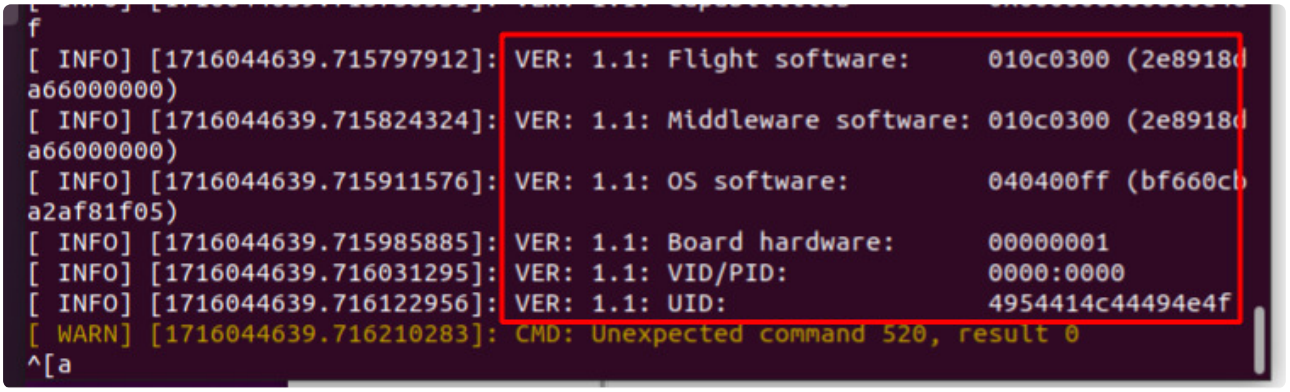
1）在Windows端运行“2-BaseDemoAuto\Windows\SITLPosStrOnekey.bat”。得到如下比赛场景界面。



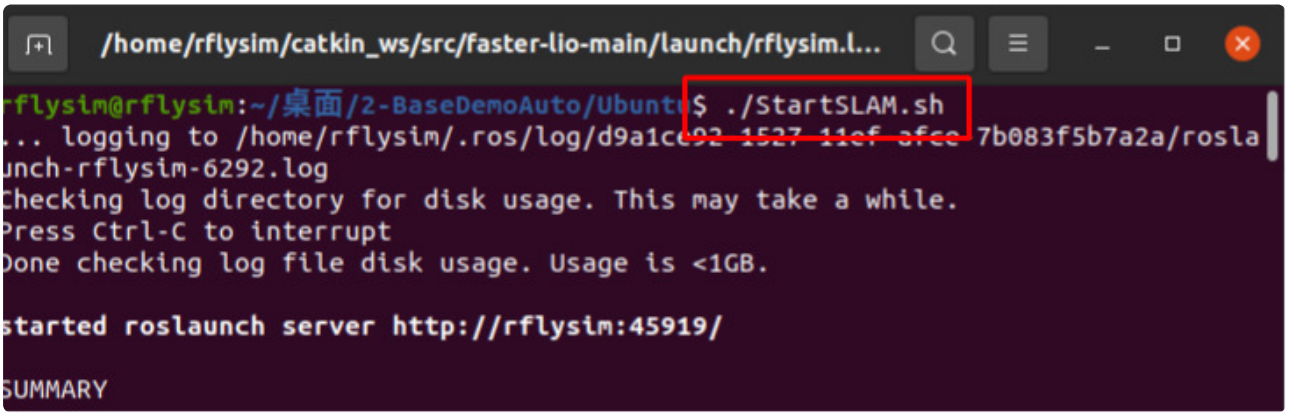
2）在Ubuntu端，进入 “2-BaseDemoAuto\Ubuntu”目录。

3）右键启动一个终端，输入命令“python3 main.py”，启动图像ROS转发，以及mavros节点，连上Windows下的RflySim平台。

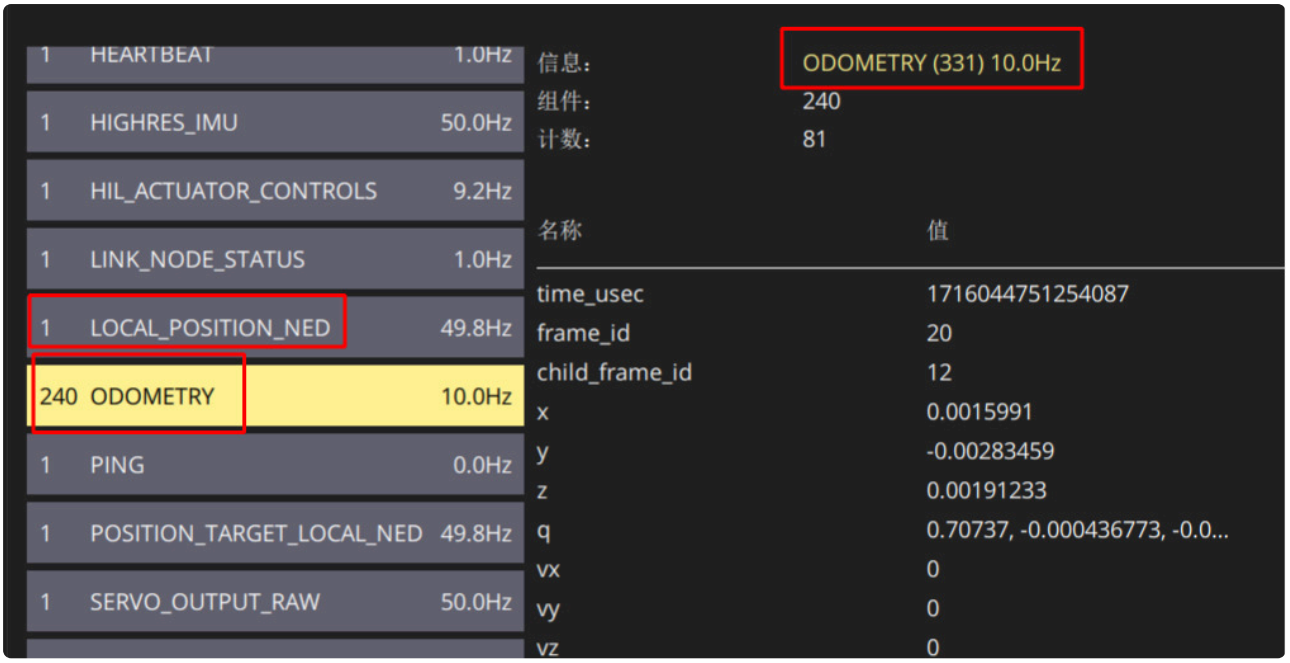




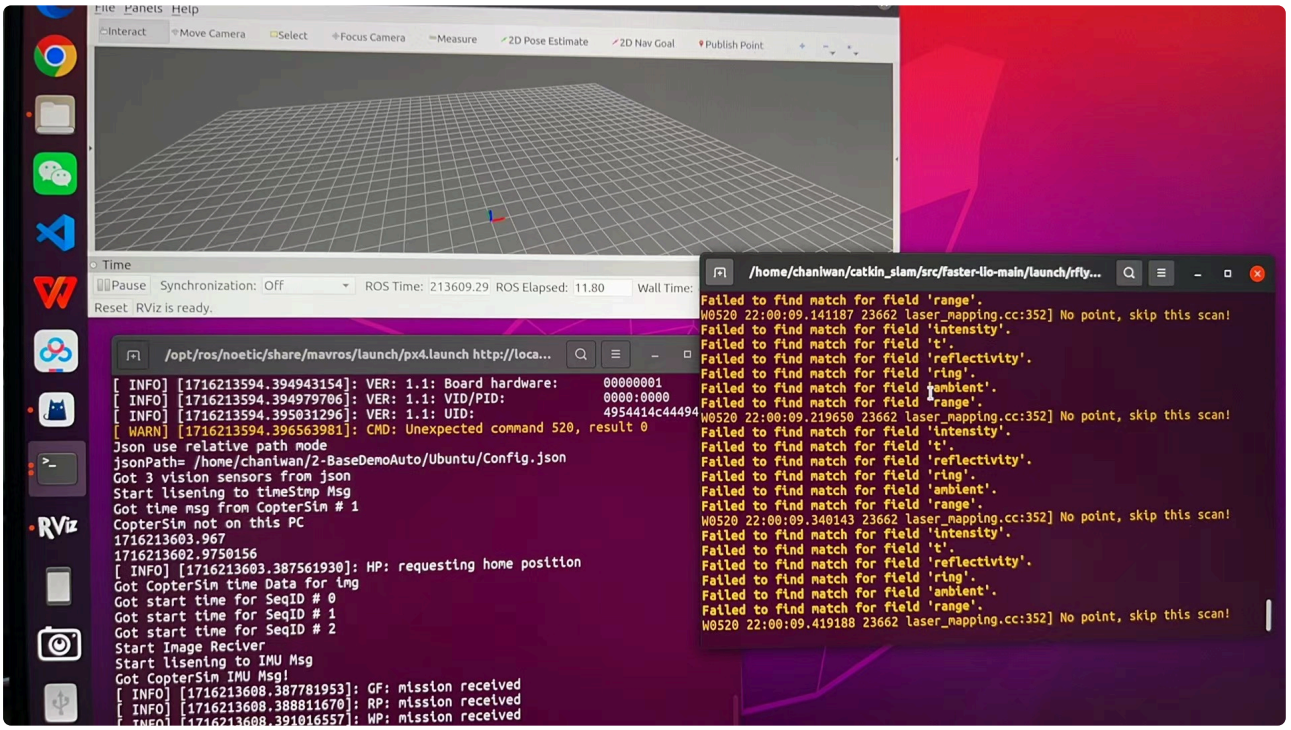
4）右键新启动一个终端，输入命令“./StartSLAM.sh”，来启动SLAM节点，它订阅激光点云和IMU数据做了一个SLAM融合，并将视觉例程计数据返回给飞控。



其次，在QGroudControl中可以看到Odometry的视觉里程计消息，和Local_Position_NED的视觉融合定位数据。说明成功发送了SLAM数据，并且在飞控中进行了融合。



注意：如果有如下黄色警告，为正常现象，不影响程序运行，请直接忽视。最终，QGC中能观测到Odometry的数据为准。



注：其他问题排查。

- 如果QGC看不到上面两条消息，说明配置未完全正确。输入“rostopic hz /mavros/odometry/out”，确认下是否已发送数据，且频率为10Hz。下图为正确现象。

```
/mavros/odometry/out
subscribed to [/mavros/odometry/out]
average rate: 10.234
        min: 0.079s max: 0.120s std dev: 0.01976s window: 10
average rate: 10.111
        min: 0.079s max: 0.121s std dev: 0.01979s window: 20
```

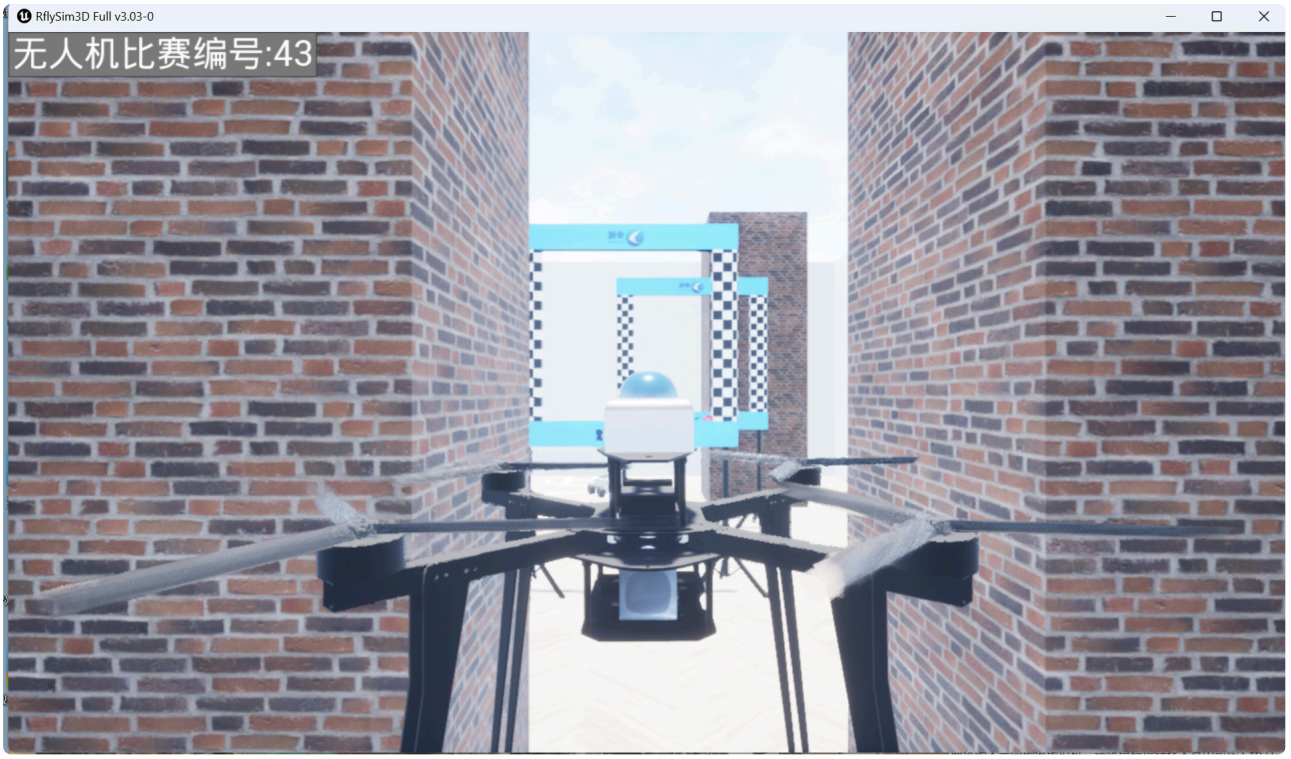
- 如果提示没有发出过数据（见下图），说明“BuildSLAM.sh”脚本未正确运行，需要重新手动检查并完成环境配置。

```
subscribed to [/mavros/odometry/out]
no new messages
no new messages
no new messages
```

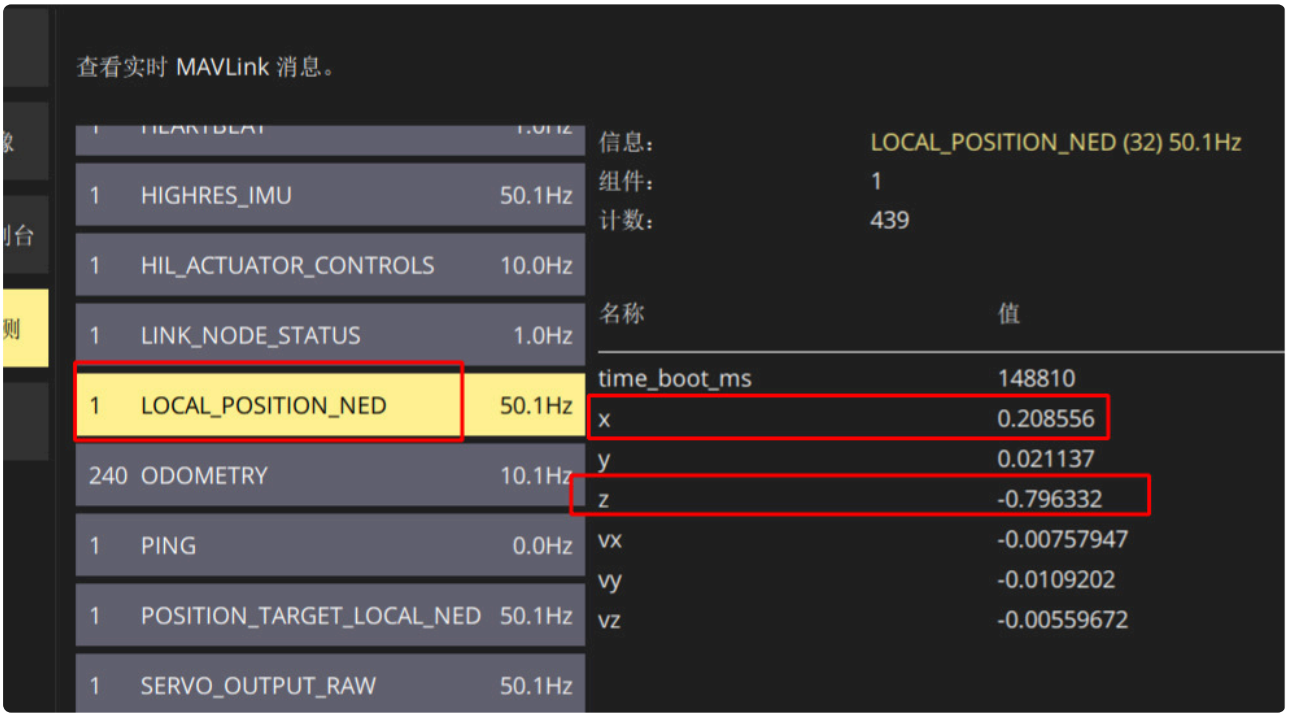
- 如果还是无法发布本消息，请尝试运行“./EnvSetup.sh”来补充安装环境包，再输入“StartRviz.sh”确认点云是否正确收到。如果点云收到了，但是odometry未发出，那可以确定为SLAM环境问题，需要自行检查系统解决。

5）最后，新打开一个终端，输入“python3 offboard.py”，可以看到飞机进入Offboard并起飞

解锁，起飞并悬停



从QGC可以看到，飞机悬停在 NED 坐标系的 0.2 0 -0.8



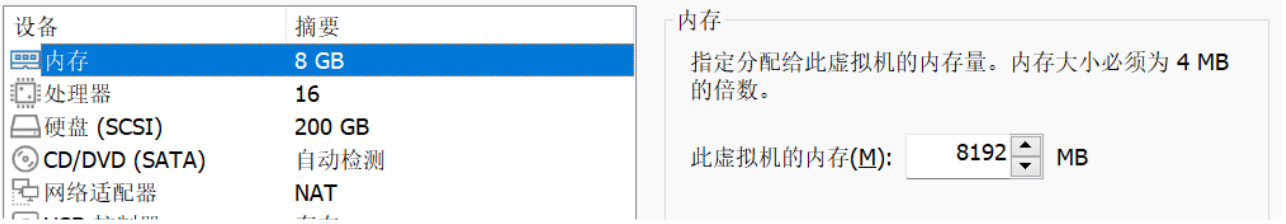
而offboard发送的位置是：ENU坐标系的 0 0.2 0.8

```
42
43      pose.pose.position.x = 0
44      pose.pose.position.y = 0.2
45      pose.pose.position.z = 0.8
46
```

上述现象说明，飞机正确地响应了信息。

6）重要注意：mavros使用的是ENU（东、北、天对应xyz）坐标系，px4内部（包括local_position等）使用的是NED（北、东、地对应xyz）坐标系。因此，mavros的x轴，对应PX4内y轴数据，y轴对应x轴数据，z轴对应-z轴数据。需要做一个映射才行。

7）注意：如果起飞后抖动，飞行不稳定。可以尝试增加虚拟机的资源配置，例如，分配8G内存（总内存32G），分配16核心给虚拟机。



6.参考资料

下面列出的资源包含本例程运行与源码理解所需的主要文件与建议阅读材料（仅列文件名或资料名称）：

- 本例程关键文件：`main.py`、`StartSLAM.sh`、`StartRviz.sh`、`offboard.py`、`testRflyRosCtrl.py`、`testRflyRosCV.py`、`BuildSLAM.sh`、`EnvSetup.sh`、`rflysim.rviz`、`Config.json`、`PX4MavCtrlV4ROS.py`。
- 一键运行脚本：`WinWSL.bat`、`WinWSLRunALL.bat`、`WinWSLRunCV.bat`、`SITLPosStrOnekey.bat`、`SITLPosStrOnekeyWithScore.bat`。
- 建议阅读（外部文档/资料名）：PX4 官方文档、ROS 官方文档、mavros 文档、RflySim API（API.pdf）。

使用建议：优先阅读示例脚本中的注释与 `PX4MavCtrlV4ROS.py` 实现，理解话题名称与坐标系映射规则后再进行算法开发。

7.常见问题

下面按故障现象给出排查要点，操作中只引用本例程文件名或命令提示：

1. 话题没有出现或图像/点云为空
   - 检查是否已运行 `main.py`。在 Ubuntu 环境或 WSL 终端中运行：

```
python3 main.py
```

```
- 确认 `Config.json` 中传感器索引正确，且在控制台没有网络/权限错误。
```

2. SLAM 无法发布里程计（QGC 中看不到 Odometry）
   - 确认已完成编译步骤（仅第一次需要）：运行 `BuildSLAM.sh`。
   - 启动 SLAM：运行 `StartSLAM.sh`，并检查终端日志中对点云和 IMU 的订阅是否正常。
   - 用 `rostopic hz /mavros/odometry/out` 检查频率（预期约 10Hz）。
3. RViz 界面白屏或不显示点云
   - 先确认 `StartRviz.sh` 已运行或手动载入 `rflysim.rviz`。
   - 如果白屏，尝试关闭后重启 `StartRviz.sh`；在虚拟化环境下，需确保图形加速/桌面会话正常。
4. 飞行器起飞后抖动或不稳定
   - 检查主机性能与资源分配（在虚拟机场景下，建议增加内存与 CPU 分配），可参考 `EnvSetup.sh` 帮助确认环境依赖。
   - 在使用 `testRflyRosCtrl.py` 或 `offboard.py` 前，先确认 `main.py` 与 SLAM 已稳定输出话题并且飞控已连接。
5. 坐标系与控制方向不一致（控制命令与实际方向错位）
   - 请注意：MAVROS（脚本中多数接口）使用 ENU，而 PX4 的 local_position 使用 NED。查看 `offboard.py` 中的坐标映射实现并按需修正。
6. 依赖或库缺失导致脚本错误
   - 若报错提示缺少 ROS、mavros、opencv 等包，运行 `EnvSetup.sh`（或参照该脚本内容手动安装所需包），然后重启相关节点。
7. 开发调试建议
   - 先按顺序执行：`main.py` -> `StartSLAM.sh` -> `offboard.py` / `testRflyRosCtrl.py`。
   - 在每步遇到问题时，只回滚到最近一步并检查对应日志，避免同时重启过多进程。

- 若需要快速复现整套流程，可使用 `WinWSLRunALL.bat` 或 `WinWSLRunCV.bat` 来在 WSL 环境中一键启动示例流程。
- 阅读 `PX4MavCtrlV4ROS.py` 源码以理解控制侧如何封装 MAVROS 接口，便于在出问题时定位是控制端还是消息链路的问题。
- 遇到无法解决的编译/驱动问题，可先参考 `BuildSLAM.sh` 中的日志输出定位编译失败点。
- 若仍无法解决，请把遇到的关键日志（保留文件名和报错信息）整理后反馈以便进一步诊断。

使用以上流程与文件名定位问题，能最大程度避免路径混淆与误引用。
使用平台免费版即可使用RGB相机，深度相机，灰度相机，以及激光雷达（包括传统机械式扫描雷达，以及固态雷达）。

- 仿真可以使用视觉定位方案，也可以使用激光雷达定位方案。避障可以用激光雷达，也可以用深度相机等。
- Demo
  仅展示配置mid360激光雷达，深度相机，前视以及下视单目RGB，如果需要其他传感器，参考 [RflySim安装目录]/RflySimAPIs/8.RflySimVision/API.pdf）7.3部分内容，如果配置需要的传感器，免费版仅支持3个传感器。Demo程序分为两部分，一部分在windows上运行，另一部在ubuntu上运行（默认系统里包含了ROS环境）。
- 本例程支持用平台的WSL/Ubuntu环境运行、或者使用Ubuntu虚拟机运行，或者使用第二台Ubuntu电脑（或NUC或NX等机载电脑）来联机运行。

注意：本例程只提供接口展示，没有实际的无人机任务控制，更完整的比赛代码请参考8.RflySimVision\1.BasicExps\4_CompSlamNav