1.实验名称及目的

1.1.实验名称

无人机识别与路径规划比赛实验（仅限完整版及以上版本）

1.2.实验目的

无人机比赛仿真实验的主要内容：首先控制无人机起飞，随后无人机需实时识别环境物体，并规划路径穿过识别到的两个方框；接着，无人机需识别地面上转圈的小车，该小车携带有ArUco标记，无人机需准确输出检测到的标记的ID；最后，在完成所有任务后，无人机需识别降落位置并实现准确降落。整个比赛过程中，无人机的起飞、降落、穿越方框的准确性以及识别移动小车上图案编号的正确性，将作为评估其比赛表现的关键指标。

注意：此实验应在ROS1中运行。

1.3.关键知识点

关键知识点1：ReqCopterSim接口使用（自动获取ip接口）

在`\\wsl.localhost\RflySim-20.04\root\catkin_ws\src\Challege_ROS\sensor_pkg\main.py`中使用，本机制确保例程拷贝到NX或其他电脑上，也能请求并获取到图像和mavros消息。

```
req = ReqCopterSim.ReqCopterSim() \# 获取局域网内所有CopterSim程序电脑IP列表

TargetIP = req.getSimIpID(StartCopterID) #获取CopterSim的1号程序所在电脑的IP，作为目标IP

req.sendReSimIP(StartCopterID) \# 请求回传数据到本电脑
```

关键知识点2：视觉接口使用

在`\\wsl.localhost\RflySim-20.04\root\catkin_ws\src\Challege_ROS\sensor_pkg\main.py`中使用，向RflySim3D发送取图请求，并将图像转发到ROS空间。

```
vis = VisionCaptureApi.VisionCaptureApi(TargetIP) \#创建一个视觉传感器实例，这个实例对应的ip号为Targe

vis.jsonLoad(jsonPath ="/root/catkin_ws/src/Challege_ROS/sensor_pkg/Config.json") \#加载Config.

isSuss = vis.sendReqToUE4(0, TargetIP) \# 向 RflySim3D 发送取图请求

vis.startImgCap() \#开启取图循环，执行本语句后，已经可以通过vis.Img[i]读取到图片

vis.sendImuReqCopterSim(StartCopterID,TargetIP) \#发送请求，从目标飞机CopterSim读取IMU数据
```

关键知识点3：相机接收图像数据，将检测结果发布到ROS话题上

在 `\\wsl.localhost\RflySim-20.04\root\catkin_ws\src\Challege_ROS\object_det\scripts\det.py` 中使用，订阅图像，使用yolo识别，并将识别结果通过ROS发布出去。

```
rospy.init_node('det_node')\# 初始化ROS节点det_node

self.img1_sub = rospy.Subscriber(img1_topic, Image, partial(self.img_cb,idx=1))# 订阅前视相机的图

self.img2_sub = rospy.Subscriber(img2_topic, Image, partial(self.img_cb,idx=2))# 订阅下视相机的图

self.task_sub = rospy.Subscriber("/task_state",MissionState,self.task_cb)#订阅任务状态话题

self.ret_pub = rospy.Publisher("/objects",Objects,queue_size=10)#发布目标检测结果

img_yolo, det, dt = self.yolo_detector(self.color_img)#调用了Yolo_Detect类的实例，进行目标检测

self.ret_pub.publish(self.objs)# 将检测结果发布到ROS话题上
```

关键知识点4：检测ArUco标记，并将检测信息发布到ROS话题上

在 `\\wsl.localhost\RflySim-20.04\root\catkin_ws\src\Challege_ROS\recognize_aruco\image.py` 使用，循环读取下视摄像头数据，并识别二维码数值。

```
rospy.init_node("aruco")\# 初始化一个名为aruco的ROS节点

msg_pub = rospy.Publisher("/Aruco", Aruco, queue_size = 10) \# 创建一个发布者

img_sub = rospy.Subscriber("/rflysim/sensor2/img_rgb",Image,ImgCB) \#创建一个订阅者，订阅/rflysim/

image = np.frombuffer(msg.data, dtype=np.uint8).reshape(msg.height, msg.width,-1) \# 将ROS图像消

corners, ids, rejectedImgPoints = aruco.detectMarkers(image, aruco_dict,parameters=parameters)

msg_pub.publish(aruco_msg) \#将消息发布到/Aruco话题

cv2.putText(image, str(ids[0]), tuple([50,50]), font, font_scale, font_color,font_thickness)\#
```

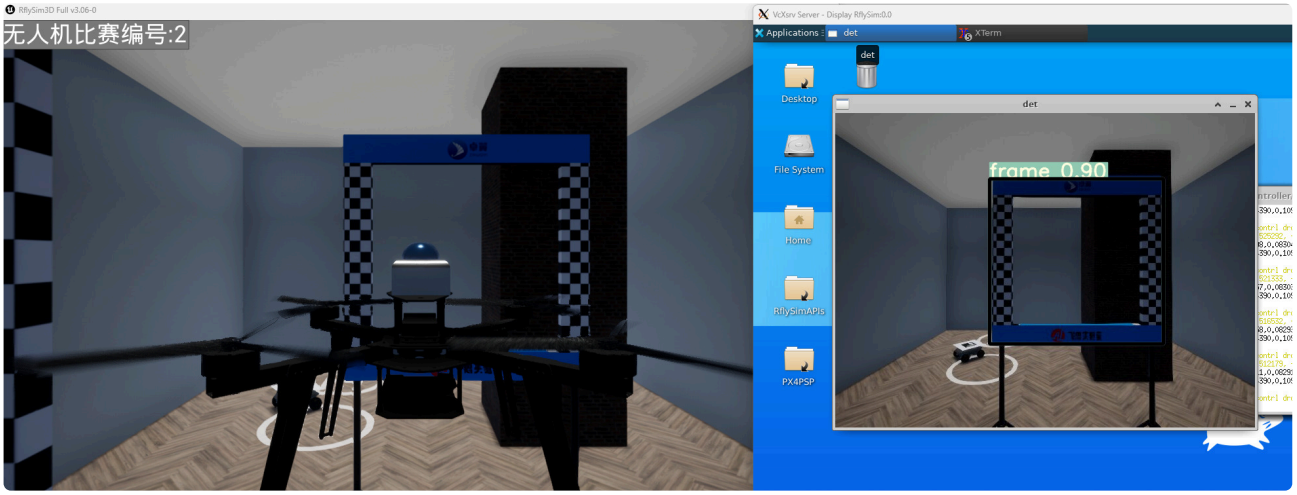
关键知识点5：UE控制

在`Windows\GameScore\maintestScore.py`中使用，创建小车，柱子等障碍物，控制一些动态效果，请求UE中物体数据，并进行实时评分。

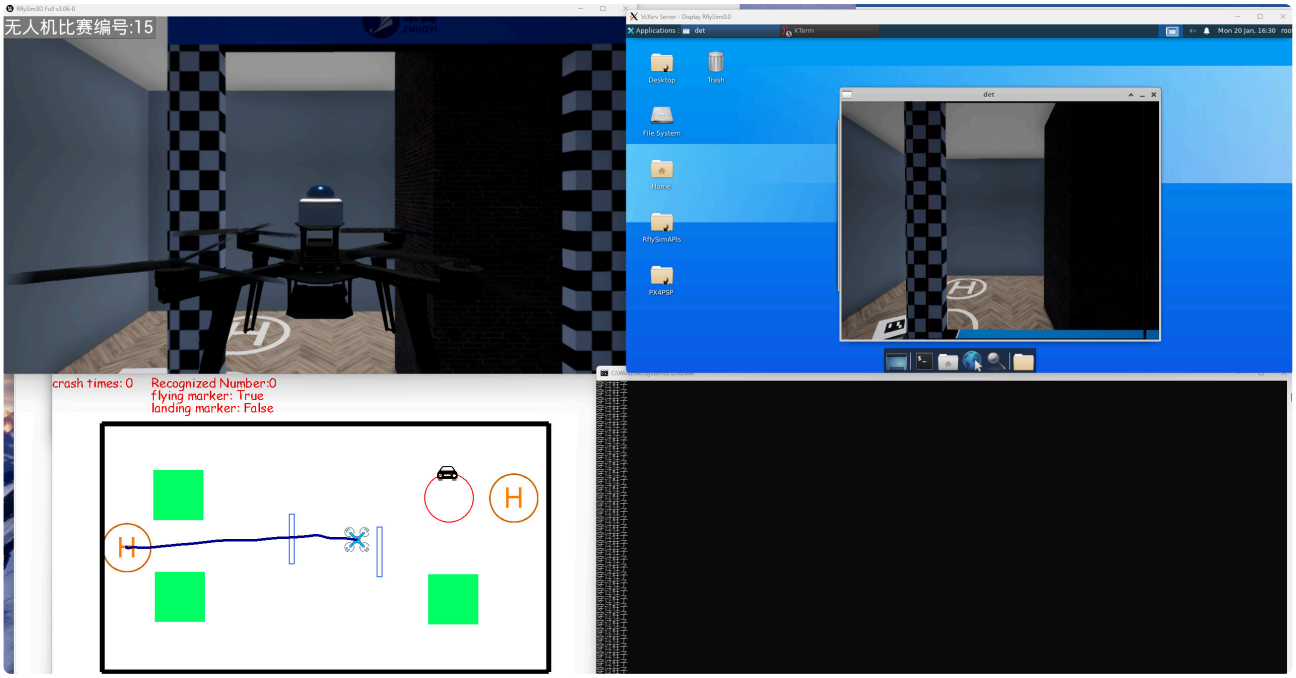
```
ue = UE4CtrlAPI.UE4CtrlAPI()#创建 UE 控制实例

ue.sendUE4Cmd('r.setres 720x405w',0) #设置UE4窗口分辨率，注意本窗口仅限于显示，取图分辨率在json中配置，本

ue.sendUE4Cmd('t.MaxFPS 20',0)# 设置UE4最大刷新频率，同时也是取图频率

ue.sendUE4PosNew(copterID=i+10002,vehicleType=pos[i][0], PosE=pos[i][1:4],AngEuler=ang[i],runn

ue.sendUE4ExtAct(10002,[0.5,0.5,0.5,1,0,0,0,0,0,0,0,0,0,0,0,0])#触发扩展蓝图接口
```

2.实验效果

SITLPosStrOnekey.bat（不包含评分程序）：无人机起飞并完成相应的任务，最后降落到指定地点。



SITLPosStrOnekeyWithScore.bat（包含评分程序）：无人机起飞并完成相应的任务，最后降落到指定地点。同时会有一个可视化的界面可以实时看到无人机的飞行路径以及任务完成情况，并显示出得分。



3.文件目录

| 例程目录：[安装目录]\RflySimAPIs\8.RflySimVision\1.BasicExps\4_CompSlamNav
| ---------- | ---------- |
| 文件夹/文件名称 | 说明 |
| Ubuntu/Build_src.sh | 解压、创建工作空间并编译的Bash脚本文件 |
| Ubuntu/WinWSLRunBuildSrc.bat | 一键运行Build_src.sh，来解压编译源码 |
| Ubuntu/ run_temp_try.sh | 进入指定工作空间并执行相应文件的Bash文件 |
| Ubuntu/ src.zip | 实验运行文件夹压缩文件 |
| Ubuntu/ WinWSL.bat | 进入 WinWSL 的 Ubuntu 环境的程序运行脚本 |
| Ubuntu/ WinWSLRunTempTry.bat | WSL1/Ubuntu 22.04 环境程序运行脚本 |
| Ubuntu/ WslGUI.bat | WSL1/Ubuntu 22.04 可视化界面脚本 |
| Windows/GameScore | 无人机任务评分及可视化程序文件 |
| Windows/SITLPosStrOnekey.bat | 自动化启动和配置多旋翼无人机仿真环境程序运行脚本 |
| Windows/SITLPosStrOnekeyWithScore.bat | 自动化启动和配置多旋翼无人机仿真环境程序运行脚本(有得分) |

4.运行环境

4.1 软件要求

Windows 10及以上版本；RflySim工具链；Linux（Ubuntu 20.04/22.04）。

> ①：若使用Pixhawk 6X飞控，平台安装时的编译命令为：px4_fmu-v6x_default，推荐PX4固件版本为：1.12.3。其他配套飞控及编译命令请见：

> https://rflysim.com/doc/zh/1/Hardware.html

4.2 硬件要求

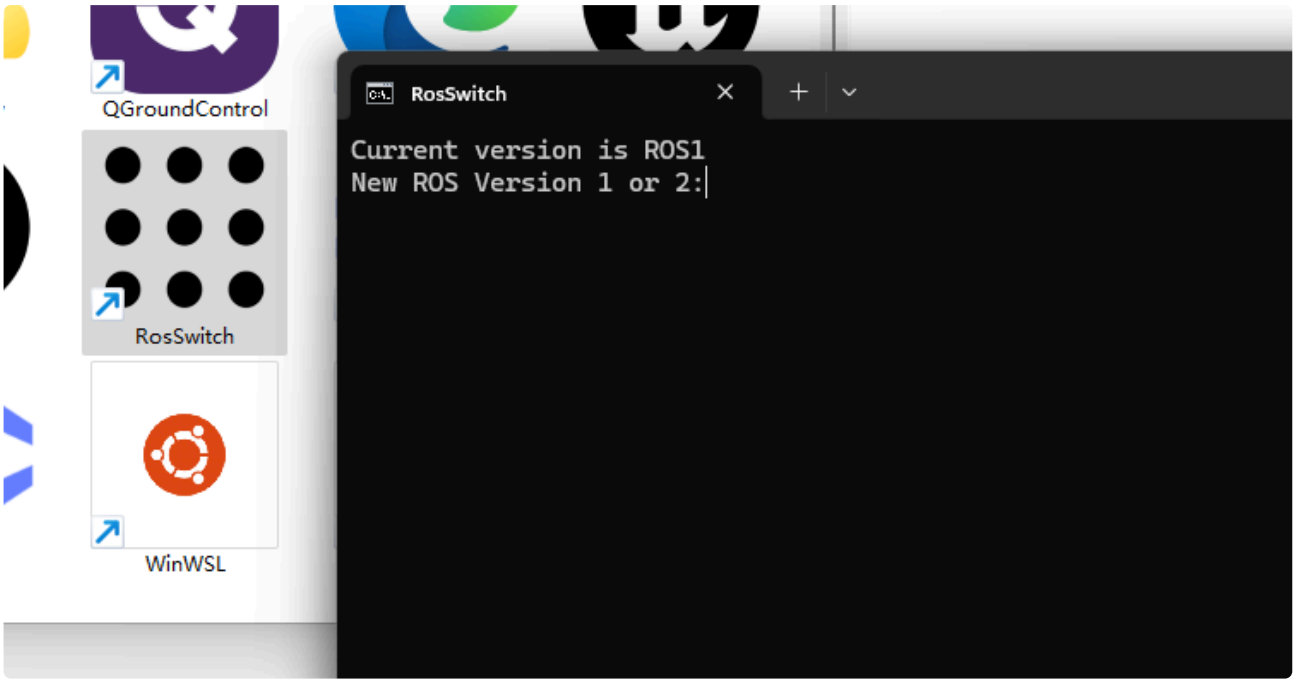
笔记本/台式电脑① 1台；WinWSL 1台。

> ①：推荐配置请见： https://rflysim.com/doc/zh/HowToInstall.pdf

5.实验步骤

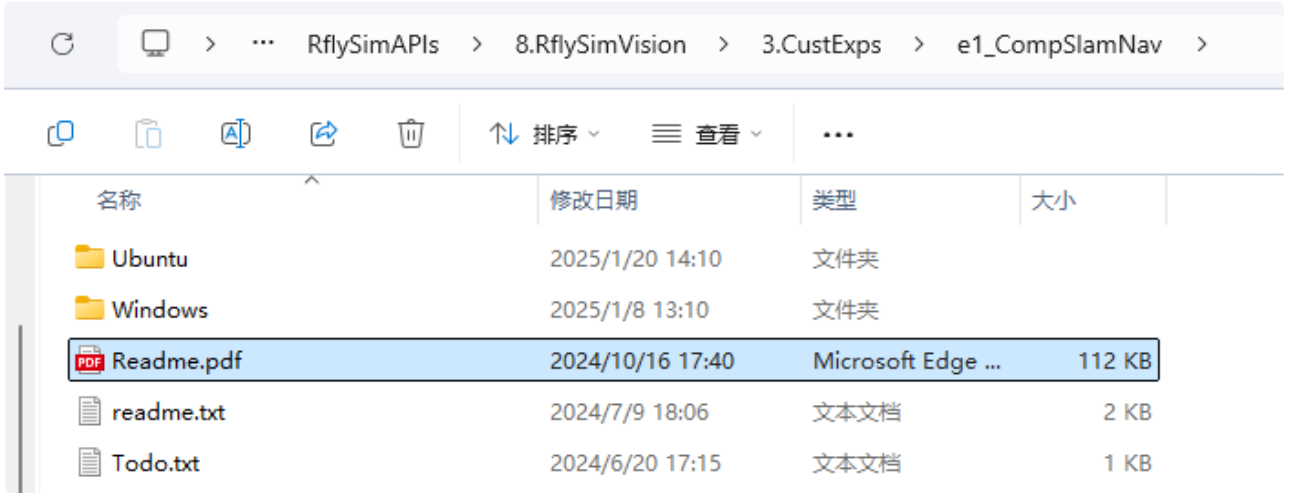
1.切换ROS版本

打开RflyTools文件夹中的RosSwitch，双击运行，将版本切换到ROS1。如果已经是ROS1，就直接退出即可。



2.进入指定文件夹

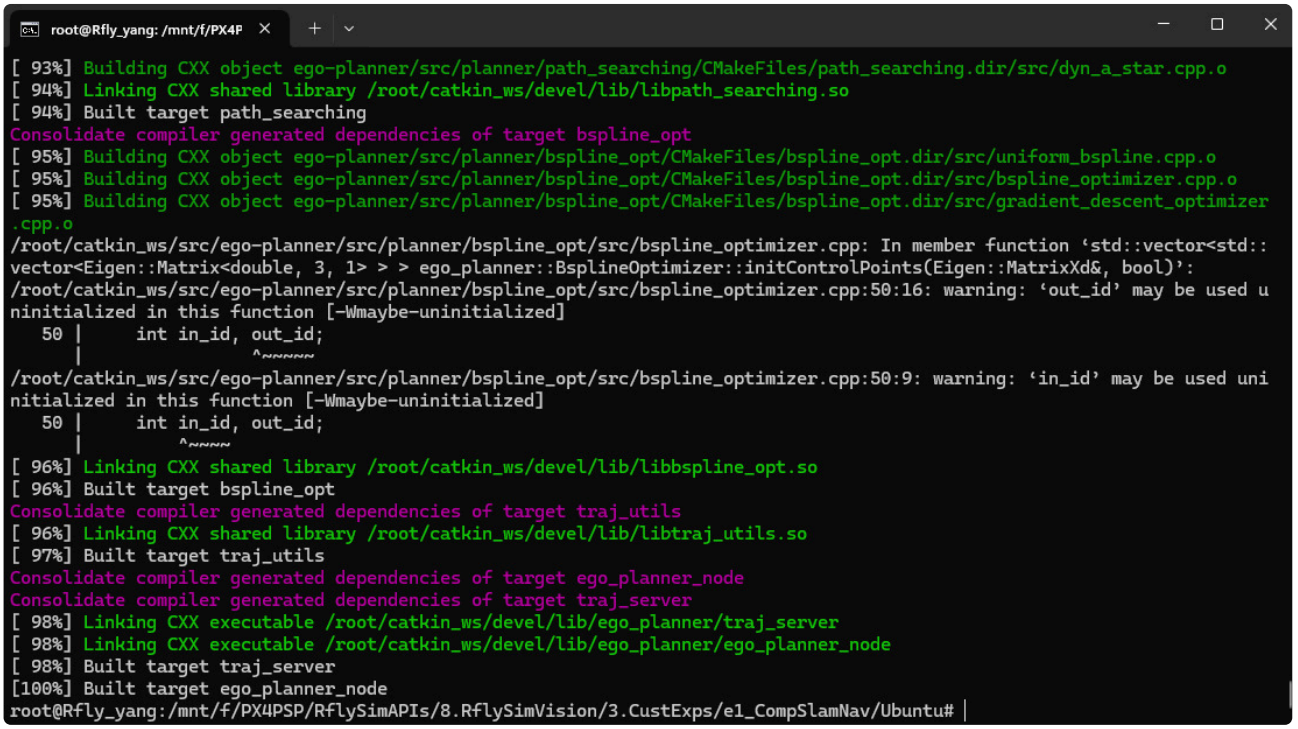
打开[安装目录]`\RflySimAPIs\\8.RflySimVision\\3.CustExps\\e1_CompSlamNav`文件夹。



3.自动编译例程源码(推荐)

进入Ubuntu文件夹，直接双击“WinWSLRunBuildSrc.bat”即可自动解压src.zip到~/catkin_ws目录，并进行ROS1编译。

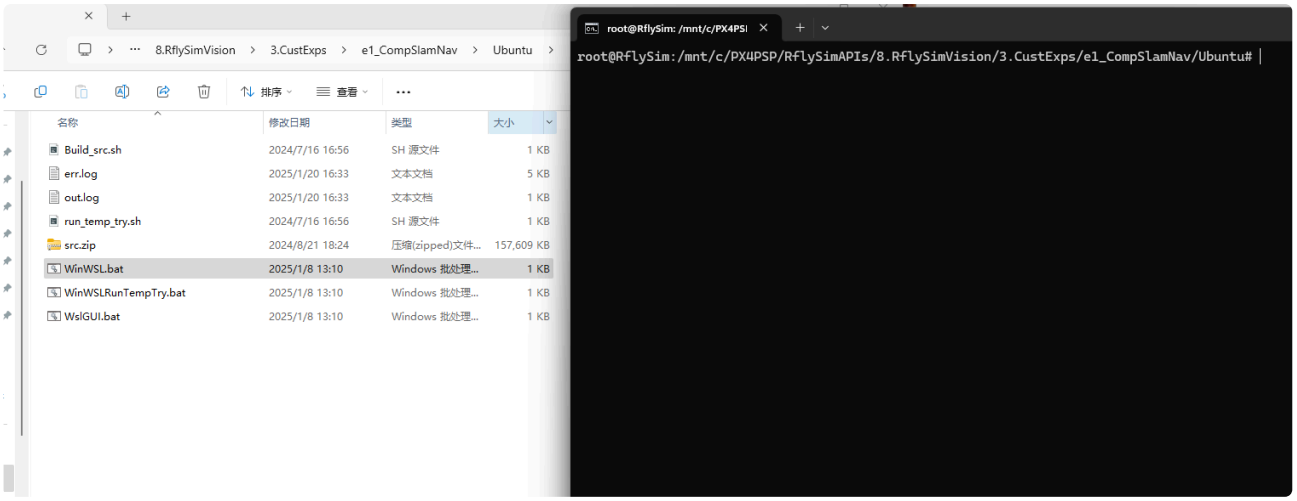
如果编译成功，说明没有问题，编译成功界面如下图所示：



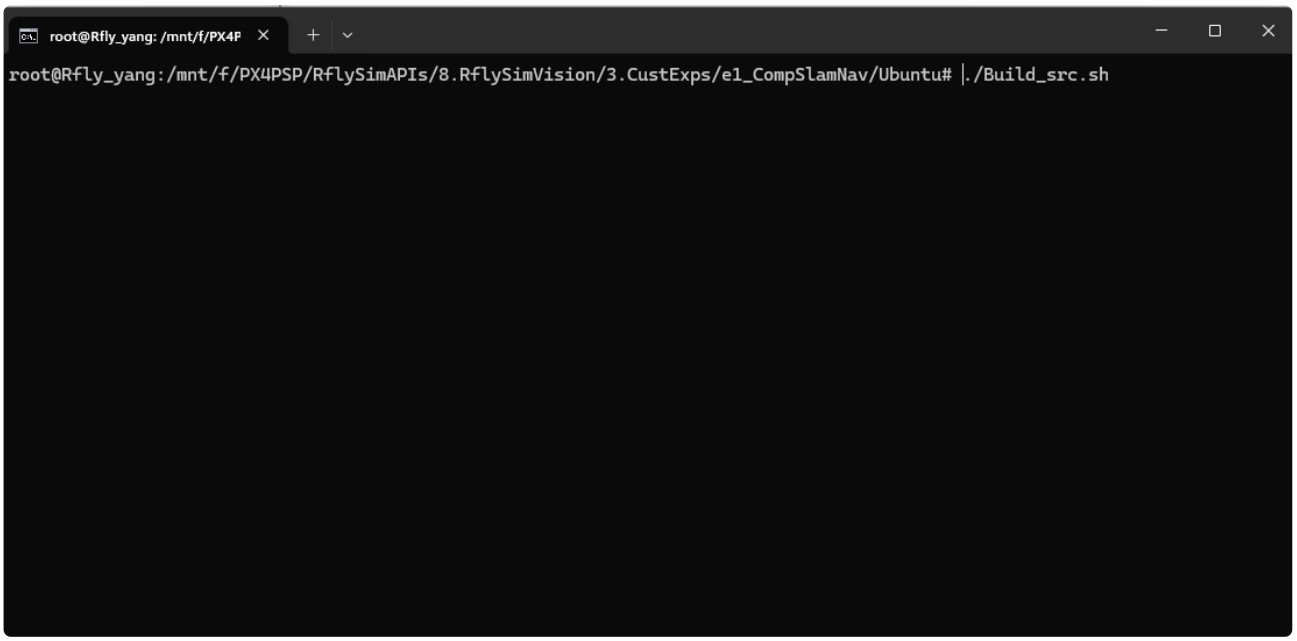
4. 手动编译方法（了解原理即可）

上述bat脚本，实际上包含了多个命令，即进入WSL环境，并执行Build_src.sh，进行拷贝与编译。用户也可以使用如下方法，利用WSL/22.04进行手动编译，并阅读每一步的脚本源码，熟悉基本的ROS程序编译流程：

进入Ubuntu文件夹，双击WinWSL.bat进入WinWSL的Ubuntu环境。



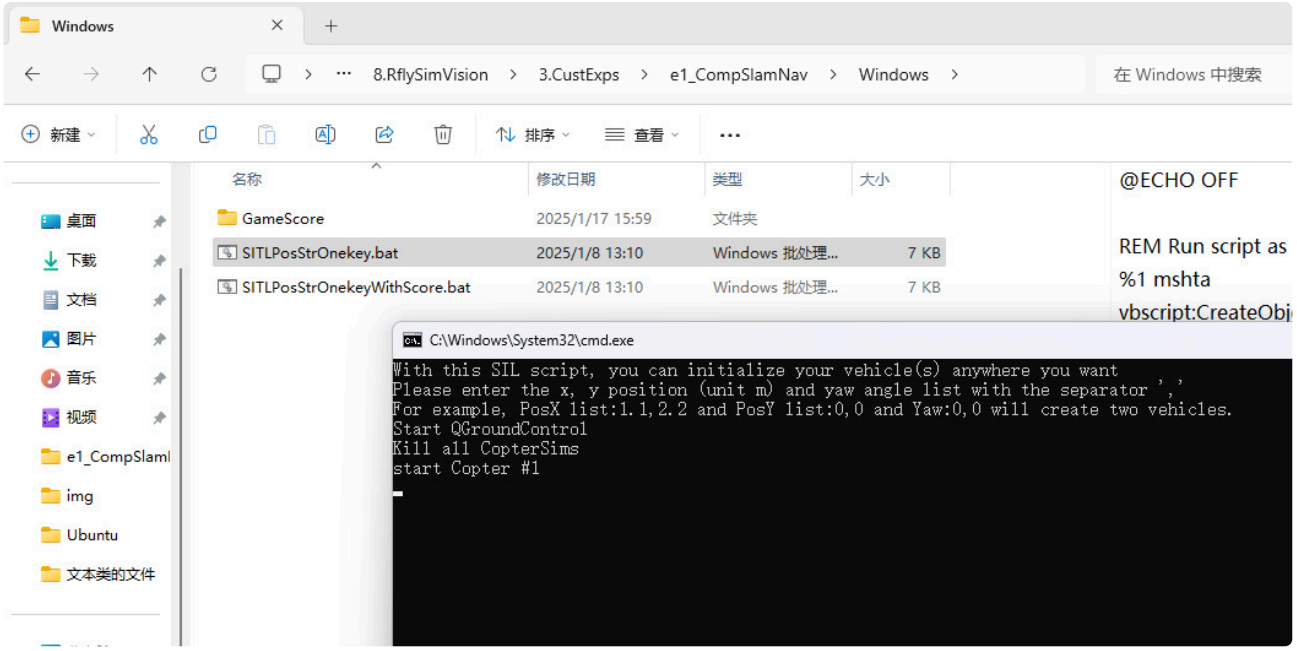
在Ubuntu环境中运行`./Build_src.sh`，运行此文件可以将src.zip拷贝到~/catkin_ws并编译。



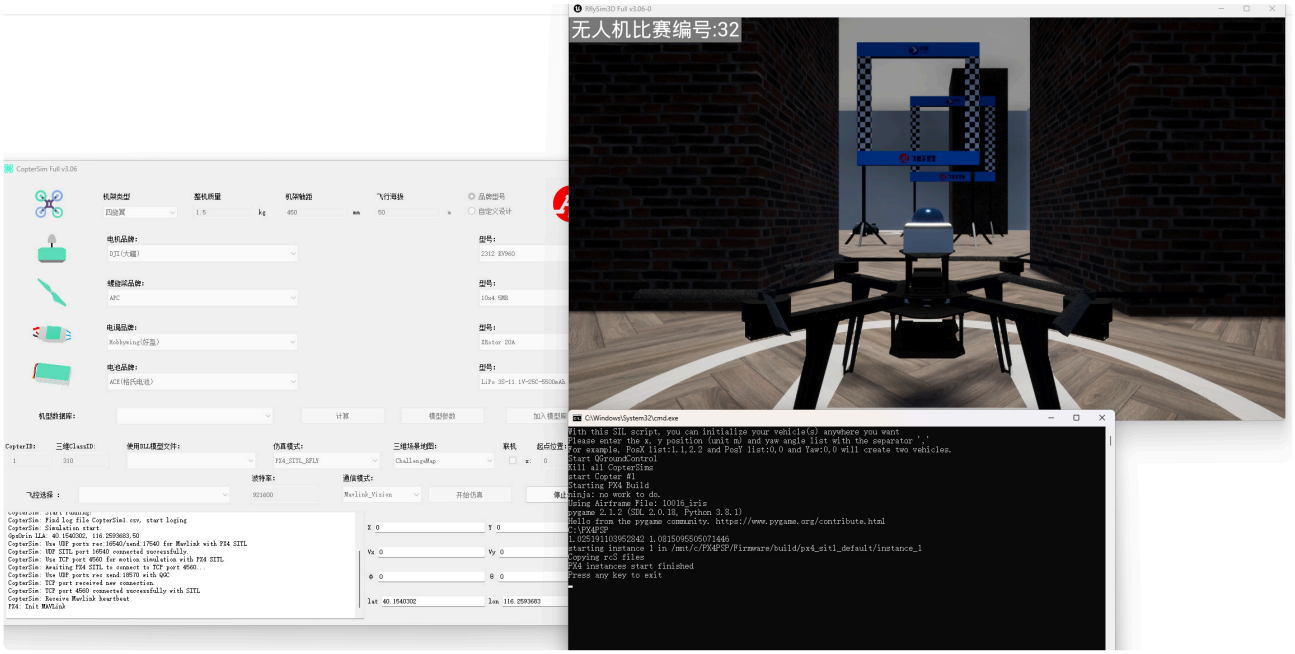
注意：如果在执行此步就有报错，显示没有xxx目录，说明生成的catkin_ws文件夹不是名字为catkin_ws的文件夹，可能包含有乱码后缀如catkin_ws#。（含有同名的文件夹，需要回收站也清理掉，或者想保留原来的catkin_ws文件夹）
把catkin_ws 文件夹删除，去回收站也删除，确保系统中不含有catkin_ws文件夹。

5.启动SITL程序

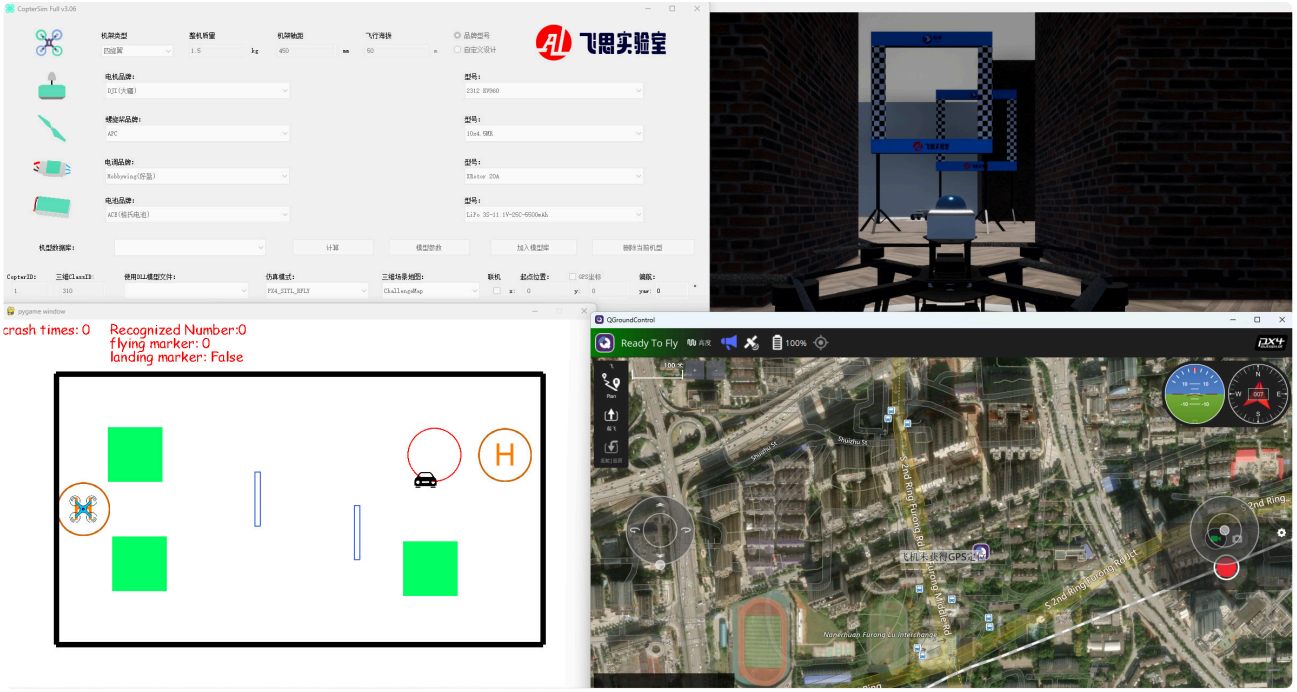
进入windows文件夹，双击SITLPosStrOnekey.bat启动SITL（不含评分程序），或双击SITLPosStrOnekeyWithScore.bat（含评分程序）。以SITLPosStrOnekey.bat为例：



此时会启动RflySim3D以及CopterSim：

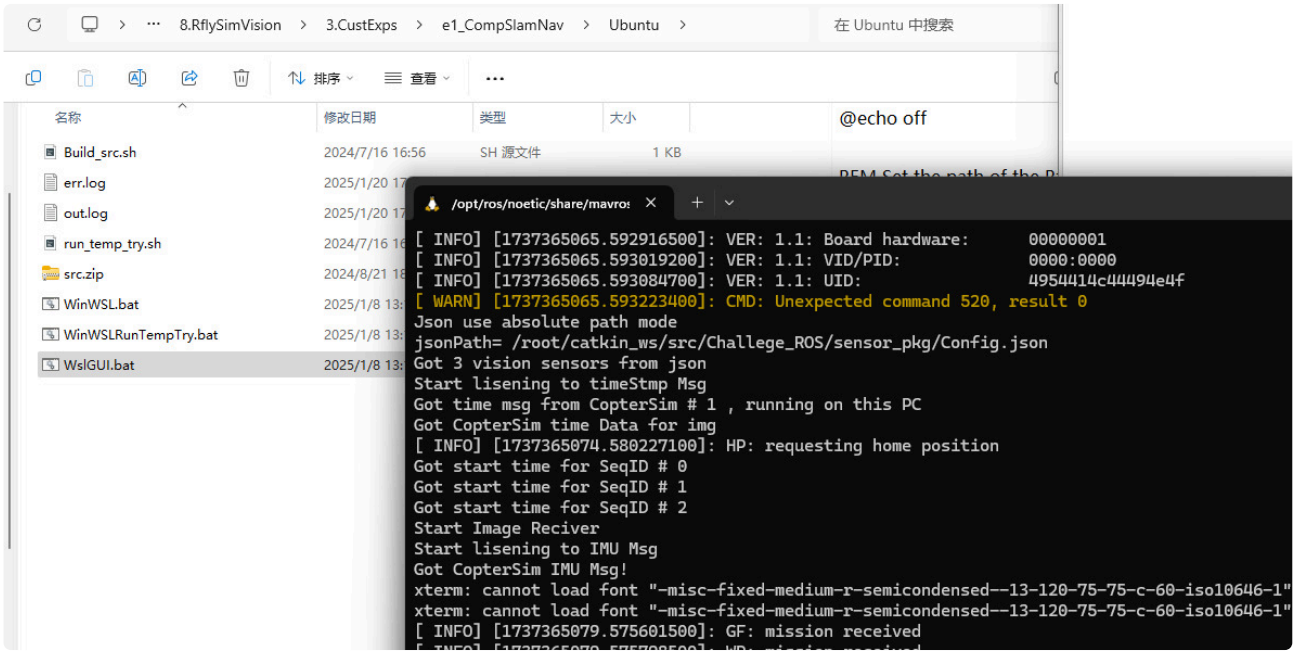


（如果运行SITLPosStrOnekeyWithScore.bat，会多出QGC以及一个可视化界面）

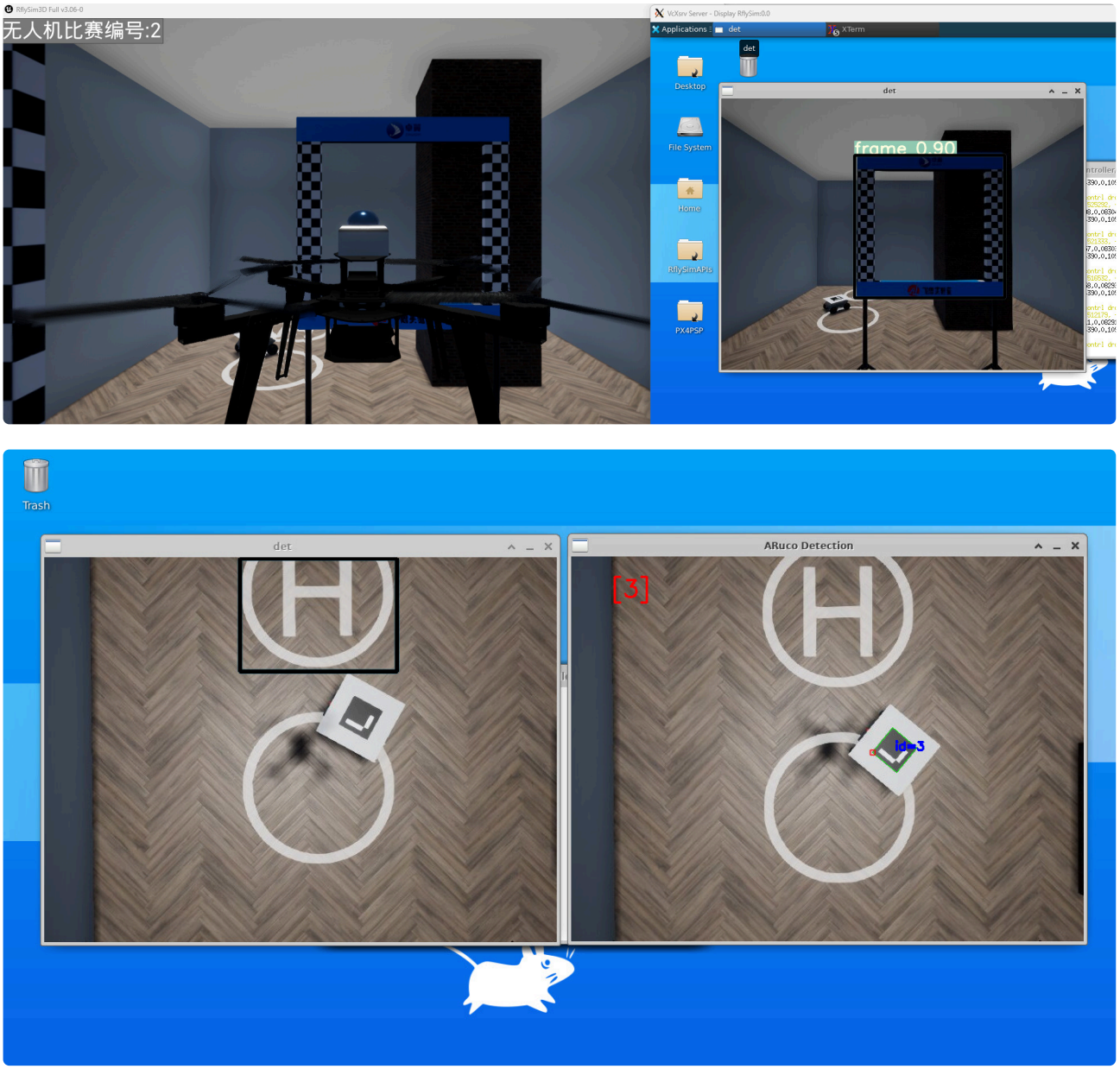


6.运行并观察结果

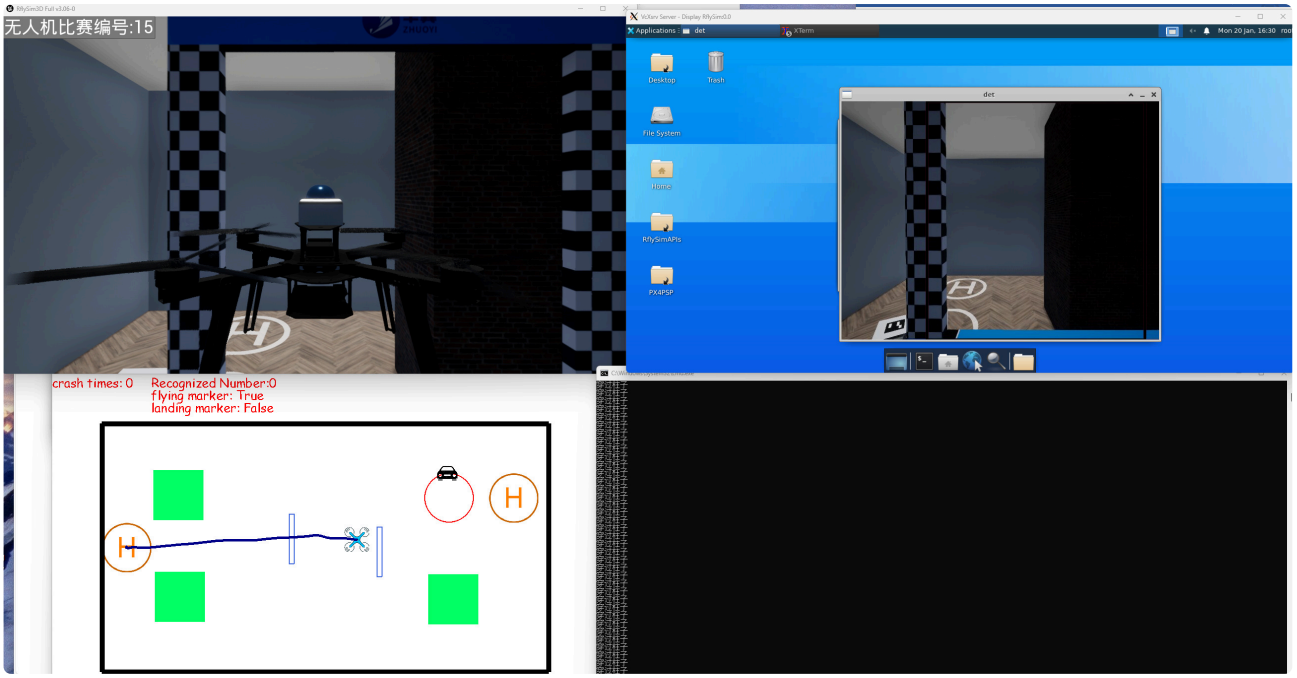
双击WinWSLRunTempTry.bat开始自动启动所有算法，观察运行结果。

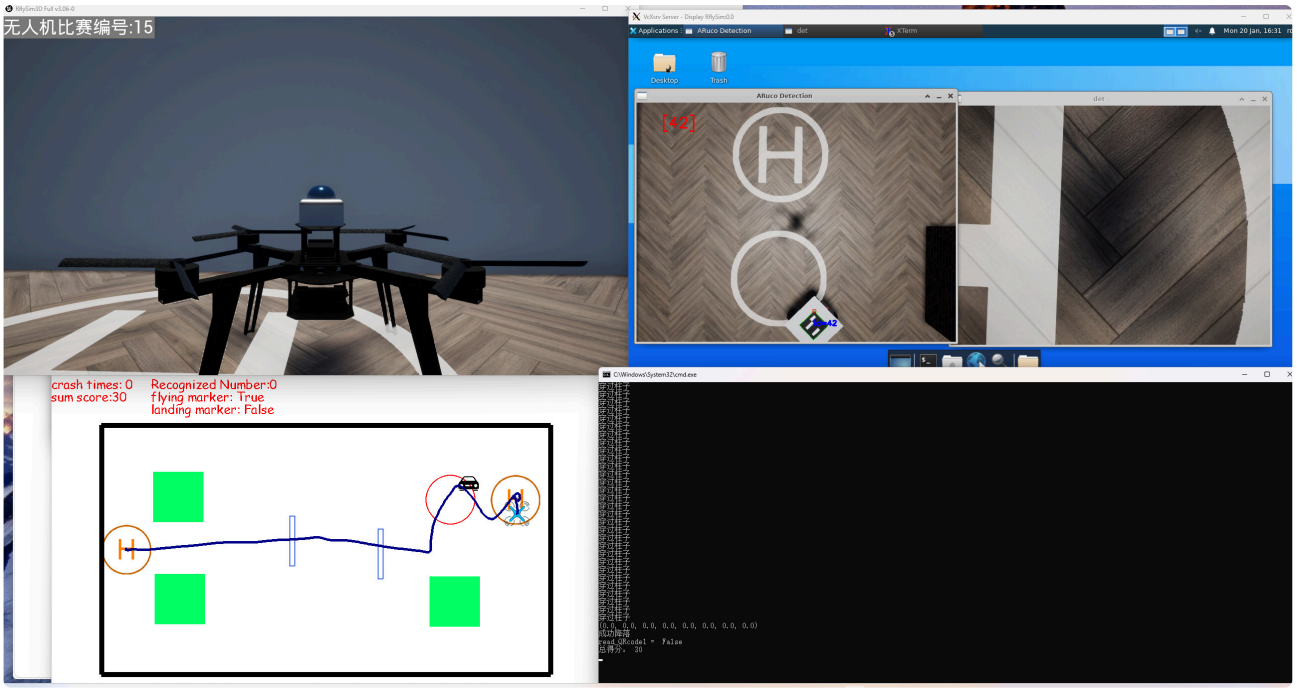


可以观察到无人机起飞并完成相应的任务，最后降落到指定地点。



（如果运行SITLPosStrOnekeyWithScore.bat，无人机起飞并完成相应的任务，最后降落到指定地点。同时会有一个可视化的界面可以实时看到无人机的飞行路径以及任务完成情况，并显示出得分。）





7.结束仿真

在脚本开启的命令提示符 CMD窗口中，按下回车键（任意键）就能快速关闭CopterSim、QGC、RflySim3D等所有程序。

8. 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。

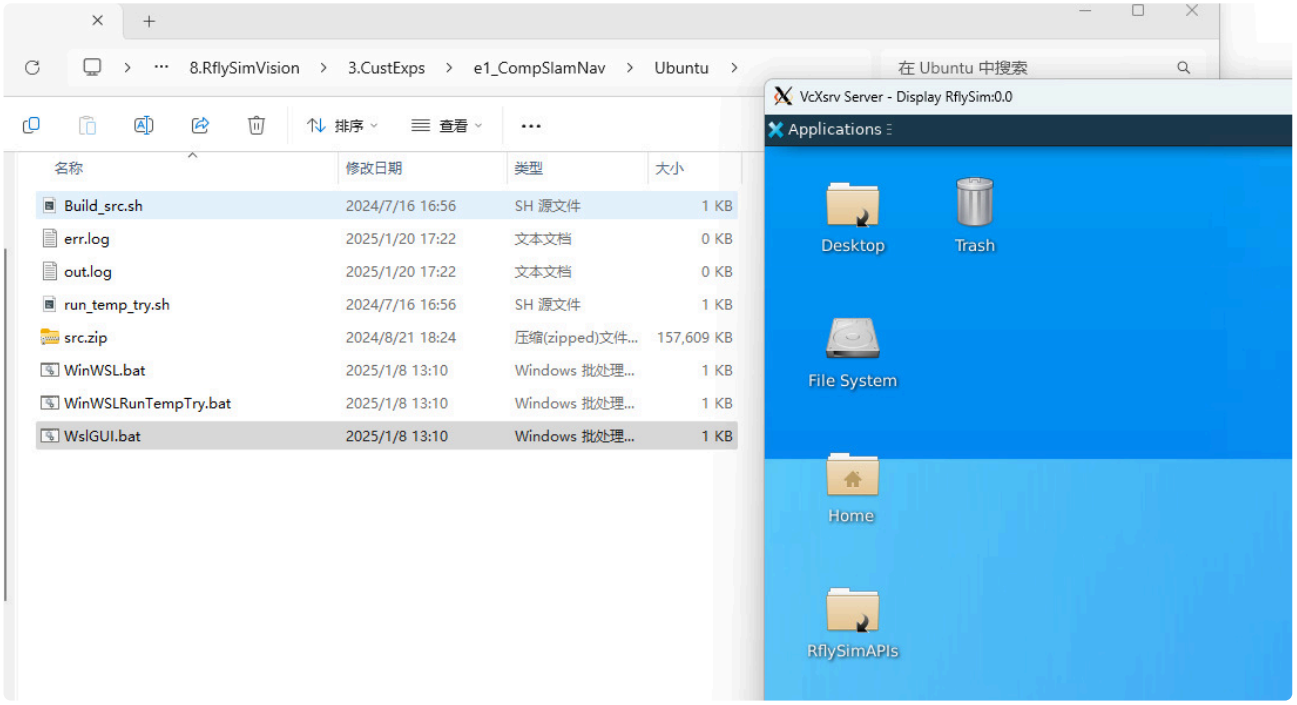
1. 根据实验例程：`1.RflySimIntro\2.AdvExps\e13.WinWSL2-GPU\Readme.pdf` 来下载WinWSL2-GPU的外挂WSL2镜像（约50G），并部署到平台。
2. 运行`\Ubuntu\WinWSLRunBuildSrc2.bat`来部署实验源码，并通过ROS编译算法节点。

3. 运行 `\Windows\SITLPosStrOnekey.bat` 来启动无人机的SITL仿真。
4. 运行 `\Ubuntu\WinWSLRunTempTry2.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，并完成穿环任务，最后降落。

9.打开WSL可视化界面（选做）

在运行WinWSLRunTempTry.bat前，也可以双击WslGUI.bat打开WSL可视化界面，进行一些ROS开发操作。



6.参考资料

一些接口相关细节，请参考例程8.RflySimVision\1.BasicExps\2-BaseDemoAuto

7.开发说明

在本例中，WinWSLRunTempTry.bat，实际上主要是调用了run_temp_try.sh脚本，来在Ubuntu环境中执行命令。

```
13    cd /d "%~dp0"
14
15    echo run_temp_try.sh
16    start wsl -d RflySim-20.04 -e bash -lic "./run_temp_try.sh"
17    REM choice /t 10 /d y /n >nul
```

注意：实际上的Ubuntu端的例程源码在`\\wsl.localhost\RflySim-20.04\root\catkin_ws\src`目录下，

包含了比赛代码和规划器等模块。下面分别介绍。

1. 程序入口。run_temp_try.sh位于Challege_ROS\sh\temp_try.sh位置，是所有程序的起点。

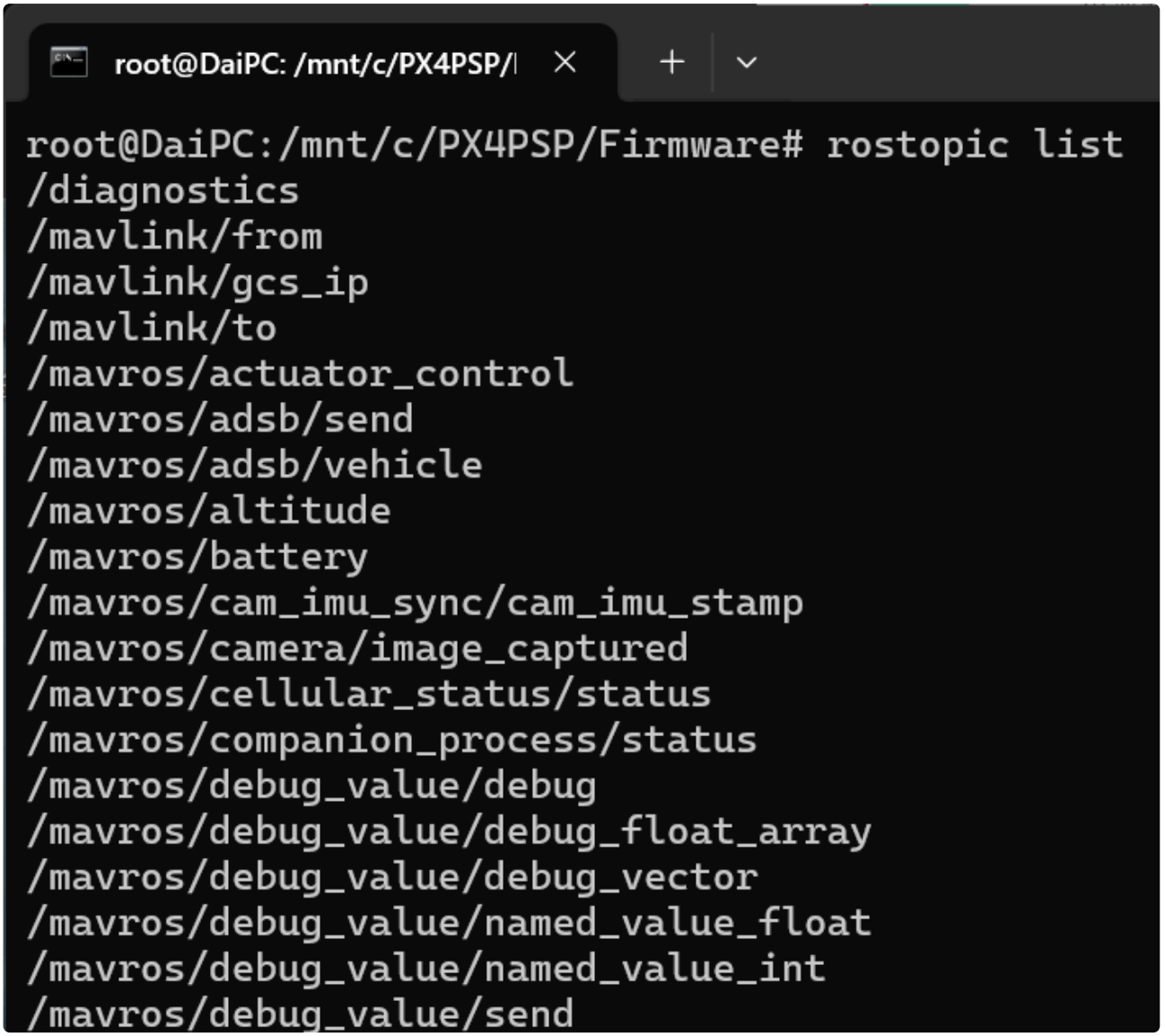
temp_try.sh代码截图以及执行的核心思路是这样的：

```
6   cd /root/catkin_ws/src/Challege_ROS/sensor_pkg && python3 main.py &
7
8   # 在新终端窗口中运行第一个命令
9   xterm -hold -e "cd /root/catkin_ws/src/Challege_ROS/object_det/scripts && python3 det.py" &
10
11  # 在新终端窗口中运行第二个命令
12  xterm -hold -e "cd /root/catkin_ws/src/Challege_ROS/recognize_aruco && python3 image.py" &
13
14  # 在新终端窗口中运行第三个命令
15  xterm -hold -e "roslaunch faster_lio rflysim.launch" &
16
17  # 在新终端窗口中运行第四个命令
18  xterm -hold -e "roslaunch ego_planner rflysim.launch" &
19
20  # 在新终端窗口中运行第五个命令
21  xterm -hold -e "roslaunch controller sim.launch"
```

2. 启动mavros节点。python3 main.py主要是让ROS环境能通过mavros连接上CopterSim和PX4。

`cd /root/catkin_ws/src/Challege_ROS/sensor_pkg && python3 main.py &`

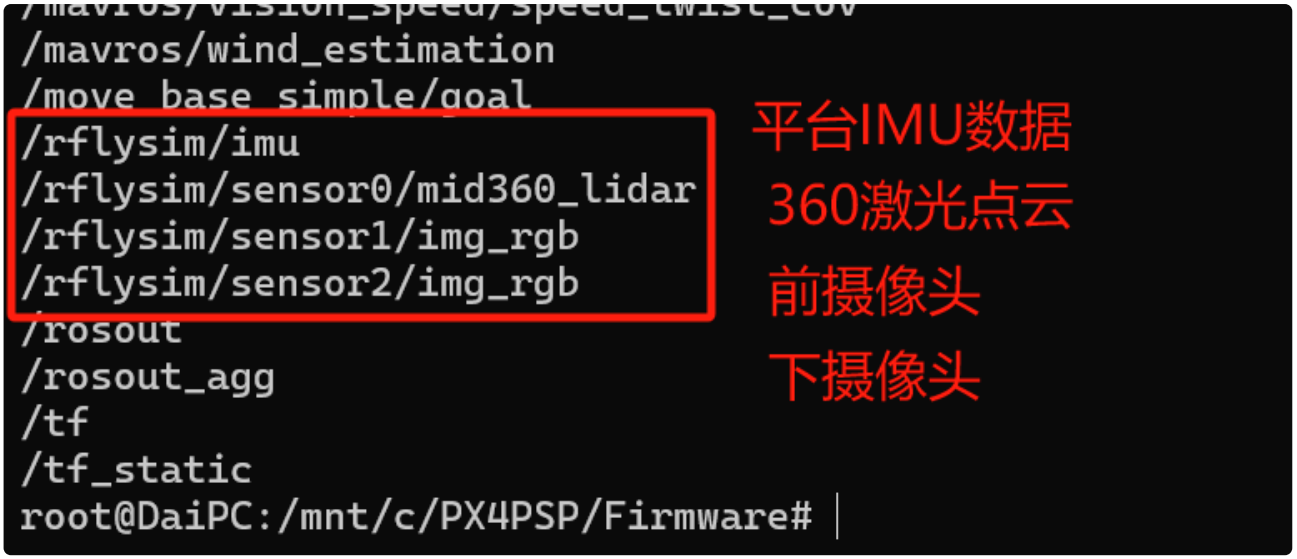
本python命令运行完毕后，mavros已经成功启动了，可以打开WinWSL.bat，输入`rostopic list`来观察mavros消息列表



此时，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。

注意：本程序还会通过`Challege_ROS/sensor_pkg/Config.json`配置文件来创建视觉传感器，并发送到ROS空间中。这次主要是用了如下传感器

/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



3. 运行核心检测程序。

```
cd /root/catkin_ws/src/Challege_ROS/object_det/scripts && python3 det.py
```

这是一个目标检测程序。主要功能是获取前视摄像头数据，并进行YOLO框检测。具体的实现原理，可仔细阅读det.py源码。

4. 二维码识别程序。

```
cd /root/catkin_ws/src/Challege_ROS/recognize_aruco && python3 image.py
```

上述命令会运行image.py，主要是为了获取下视摄像头数据，并识别其中的二维码数值。这个是本次比赛的一个得分点，要求飞机成功飞到小车二维码上方，并识别出二维码数字。

5. 运行激光SLAM定位程序。

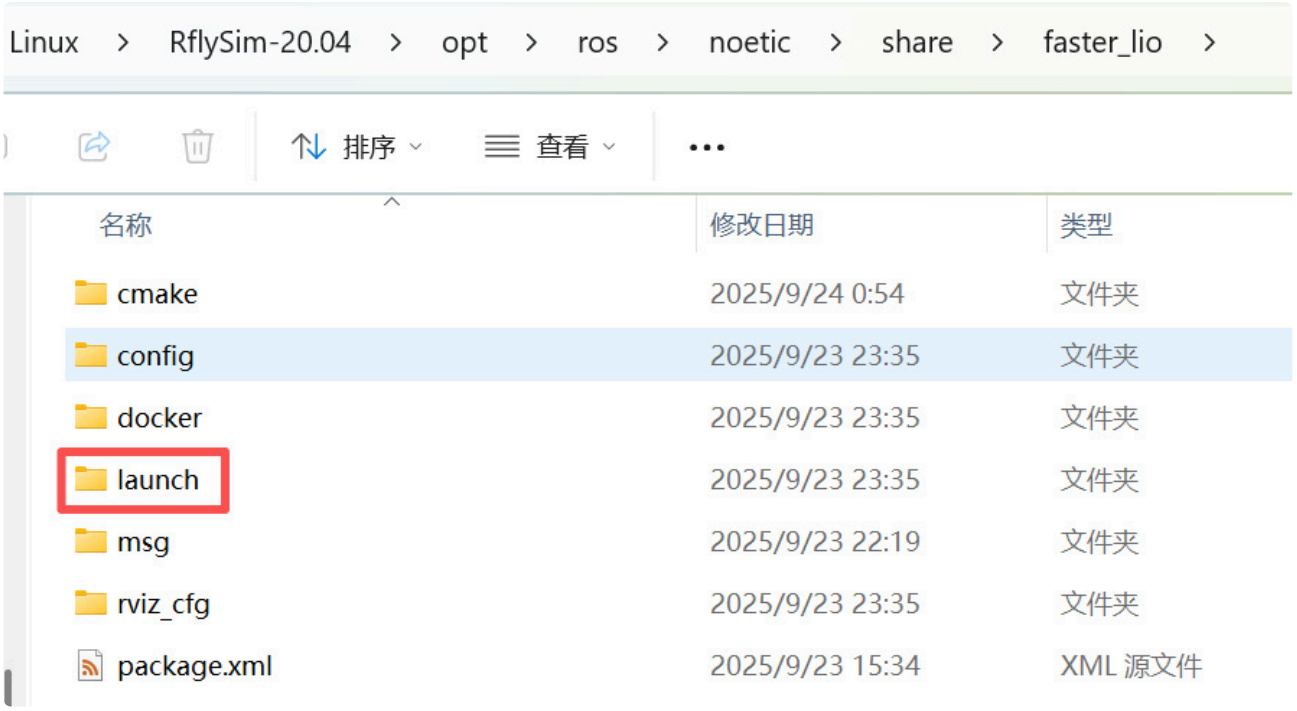
```
roslaunch faster_lio rflysim.launch
```

命令，主要是启动了faster_lio的激光SLAM定位程序，读取mid-360激光点云和飞控的IMU数据，进行SLAM定位，得到飞机的当前位置，并通过/mavros/odometry/out发送给飞控，使得飞控在没有GPS定位的情况下，能够通过视觉进行自我定位。有了定位信息后，我们就可以发送起飞、速度控制指令了。否则，飞机会拒绝接收Offboard消息。

1）fsater_lio程序，已经提前安装到平台`\\wsl.localhost\RflySim-20.04\opt\ros\noetic\share\faster_lio`中。

2）其源码和安装方法，可参考`8.RflySimVision\0.ApiExps\0.Preparation\13.faster-lio\install-faster-lio.sh`脚本。读者也可以修改src.zip中的源码，并重新安装，来获取更好的定位效果。

3）rflysim.launch文件，位于如下目录中，包含了订阅rflysim mid360的数据，和IMU的数据源，并进行SLAM算法的配置信息。



5. 实时规划器

```
roslaunch ego_planner rflysim.launch
```

核心思想是运行了ego_planner程序来进行在线轨迹规划，源码路劲为`\\wsl.localhost\RflySim-20.04\root\catkin_ws\src\ego-planner`，请自行阅读其中的原理。

其中，rflysim.launch文件位于`catkin_ws\src\ego-planner\src\planner\plan_manage\launch`，是规划器的配置信息。

6. 核心任务模块

最后运行如下脚本

```
roslaunch controller sim.launch
```

来开始整个比赛过程的流程控制和任务切换。sim.launch位于`catkin_ws\src\Challege_ROS\controller\launch`目录，本质上的最核心的算法文件在

`catkin_ws\src\Challege_ROS\controller\src\controller.cpp`，这个文件可以看作是C++版本的ROS任务控制器（与Python版本对应`8.RflySimVision\1.BasicExps\2-BaseDemoAuto\Ubuntu\testRflyRosCV.py`）

C++相比于Python具备更高的处理效率和实时性，但是编程会困难一些，用户可以自行研究两者源码。

任务模块的关键功能是区分目前比赛的进程，例如是否可以解锁起飞，是否开始穿环，是否完成穿环，是否需要开始识别二维码等。这个过程也可以基于行为树实现，详见`8.RflySimVision\3.CustExps\e7_BT_control_v2`。

7. 二次开发与运行。

用户可以修改`\\wsl.localhost\RflySim-20.04\root\catkin_ws\src`下的源码，然后通过WinWSL.bat执行下面指令，来重新编译

```
cd ~/catkin_ws/src
chmod 777 -R *
cd ~/catkin_ws
catkin_make
```

然后，在此运行WinWSLRunTempTry.bat即可。

也可以自行修改WinWSLRunTempTry.bat、run_temp_try.sh、或`/root/catkin_ws/src/Challege_ROS/sh/temp_try.sh`源文件，来控制启动的程序的时间和顺序。

8.常见问题

若无法降落，可能是电脑或网络性能问题，确认关闭VPN软件，确认关闭Windows实时防护，确认关闭其他杀毒软件或电脑管家，确认自己电脑显卡性能足够，确认切换到WSL1环境，确认切换到PX4 1.12.3，然后依次尝试。