

1. 实验名称及目的

1.1 实验名称

MAVROS控制实验

1.2 实验目的

进行图像处理，控制飞机进行穿。

1.3 关键知识点

本实验主要是实现通过Python接口ReqCopterSim.py使用（见RflySimAPIs\RflySimSDK\ctrl目录）自动获取的IP，去建立远端电脑与本机RflySim3D（发送图片）与CopterSim（接收控制指令）的联机仿真，同时通过ROS订阅得到图像数据,使用mavros去控制飞机，C++实现穿环。关键代码解析如下：

本例子和其他分布式例子的区别，主要在于进行了更加复杂的控制，结合opencv库，实现飞机穿环功能，整个流程主要是启动mavros，开启图像读取，从图像中选点作为飞机穿环的位置，从而最终实现穿环。

关键知识点1：SendProtocol[0]决定了图像的传出模式。SendProtocol[0]=0：共享内存（仅限Windows下获取图像），1：UDP直传png压缩，2：UDP直传图片不压缩（只适用图片类传感器），3：UDP直传jpg压缩（只适用图片类传感器）。如果是激光雷达数据只有0或1（共享内存和UDP网络传输）。

关键知识点2：通过ReqCopterSim可以自动从局域网获取到仿真电脑的IP地址，从而自动建立连接，不再需要手动指定IP地址。不过，此种连接方式，可能在局域网中产生干扰（多台电脑同时打开多个CopterSim会产生误识别），不适合多个实验同时进行的场景。

1) 视觉接口使用

```
1 vis.jsonLoad(3) \# # 使用jpeg压缩方式加载Config.json中的传感器配置文件
2
3 isSuss = vis.sendReqToUE4() \# 向RflySim3D发送取图请求, 发给ip为TargetIP的地址
4
5 vis.startImgCap() \# 开启取图
6
7 vis.hasData[i] \# 图片i数据是否更新
8
9 vis.Img[i] \# 图片i数据 (像素矩阵)
10
11 cv2.imshow('Img'+str(i),vis.Img[i]) \# 显示图片i图像
```

2) ReqCopterSim接口使用 (自动获取ip接口)

```
1 req = ReqCopterSim.ReqCopterSim() \# 获取局域网内所有CopterSim程序的电脑IP列表
2
3 req.sendReSimIP(1) \# 请求mavlink数据到本电脑
4
5 req.sendReSimUdpMode(1,2) \# 强制切换MAVLINK_FULL
6
7 coptersim_ip = req.getSimIpID(1) \#自动获取CopterSim的1号程序所在电脑的IP, 作为目标IP。这里获
8
9 vis = VisionCaptureApi.VisionCaptureApi(coptersim_ip) \#创建一个视觉传感器实例, 这个实例对
```

3) 相机数量和参数配置

其中, 视觉传感器的初始状态由本文件夹下的Config.json决定, 主要包含以下配置项:

```
1 "SeqID":0: 使用自动更新ID的方式, 创建了SeqID为0的视觉传感器
2
3 "TypeID":1: 传感器类型为RGB彩色图像
4
5 "TargetCopter":1: 相机绑定在1号飞机上
6
7 "SendProtocol":[1,0,0,0,0,0,0,0]: 传输模式为1: UDP网络传输模式 (图片使用jpeg压缩, 点云直传)。
8
9 "SensorPosXYZ":[0.3,0,0]: 相机分布位置。
```

4) rospy接口使用

```
1 | rospy.init_node('OpenCVR0S',anonymous= True) \#创建节点，设置anonymous=True时，ROS会在节点
2 |
3 | rospy.Subscriber("/rflsim/sensor0/img_rgb",sensor.Image,ImgCallBack) \#订阅/rflsim/s
4 |
5 | rospy.spin() \# 进入一个循环，保持ROS节点运行，以便处理回调函数
```

5) vision_ctrl.cpp重要部分代码说明

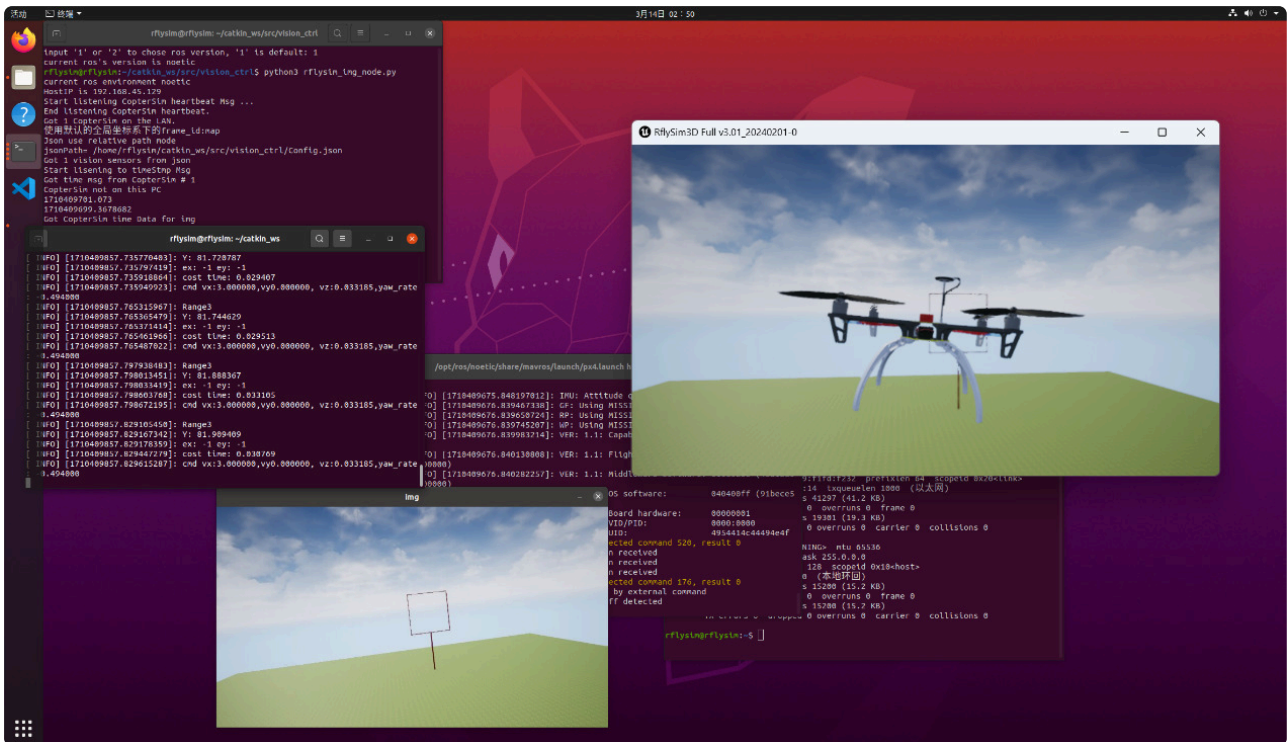
```
212 | }
213 |     if(ex != -1)
214 |     {
215 |         cmd.velocity.x = sat(ros::Time::now().toSec() - start_time.toSec(),3);
216 |         cmd.velocity.y =0 ;
217 |         cmd.velocity.z = -K_z * ey;
218 |         cmd.yaw_rate = -K_yawrate *ex;
219 |         cmd_pub.publish(cmd);
220 |         ros::spinOnce();
221 |     }
222 | }
```

这一部分就是当飞机检测到目标（环或者框），选择中心点，给予飞机速度控制的代码。通过修改系数 K_z 和 $K_{yawrate}$ 来改变飞机上升以及偏转的幅度，从而控制飞机成功穿环。

6) 其余代码说明

```
1 | encoding_ = "bgr8" \# 定义图像编码类型为BGR8
2 |
3 | cv_bidge = CvBridge() \# 创建CvBridge对象，用于图像消息与OpenCV图像之间的转换
4 |
5 | img = cv_bidge.imgmsg_to_cv2(data,encoding_) \#将ROS图像消息data转换为OpenCV格式的图像img
```

2. 实验效果



3. 文件目录

例程目录：

[安装目录]\RflySimAPI\8.RflySimVision\0.ApiExps\2-DistributedSimAPI\3.RosDistCtrl
\3.UavVisionRosCtrl

文件夹/文件名称	说明
CrossRing3SITL.bat	启动仿真配置文件
rflsim_img_node.py	Python实验程序
Config.json	视觉传感器配置文件
vision_ctrl	控制功能包
Python38Run.bat	Windows下Python程序运行脚本
WinWSL.bat	WSL1/Ubuntu 20.04环境程序运行脚本
WslGUI.bat	WSL1/Ubuntu 20.04可视化界面脚本

4. 运行环境

4.1 软件要求

Windows 10及以上版本；RflySim工具链；Visual Studio Code；Linux（Ubuntu 20.04）；Linux（Ubuntu 20.04）。

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；虚拟机/视觉盒子/其他板卡 可选台。

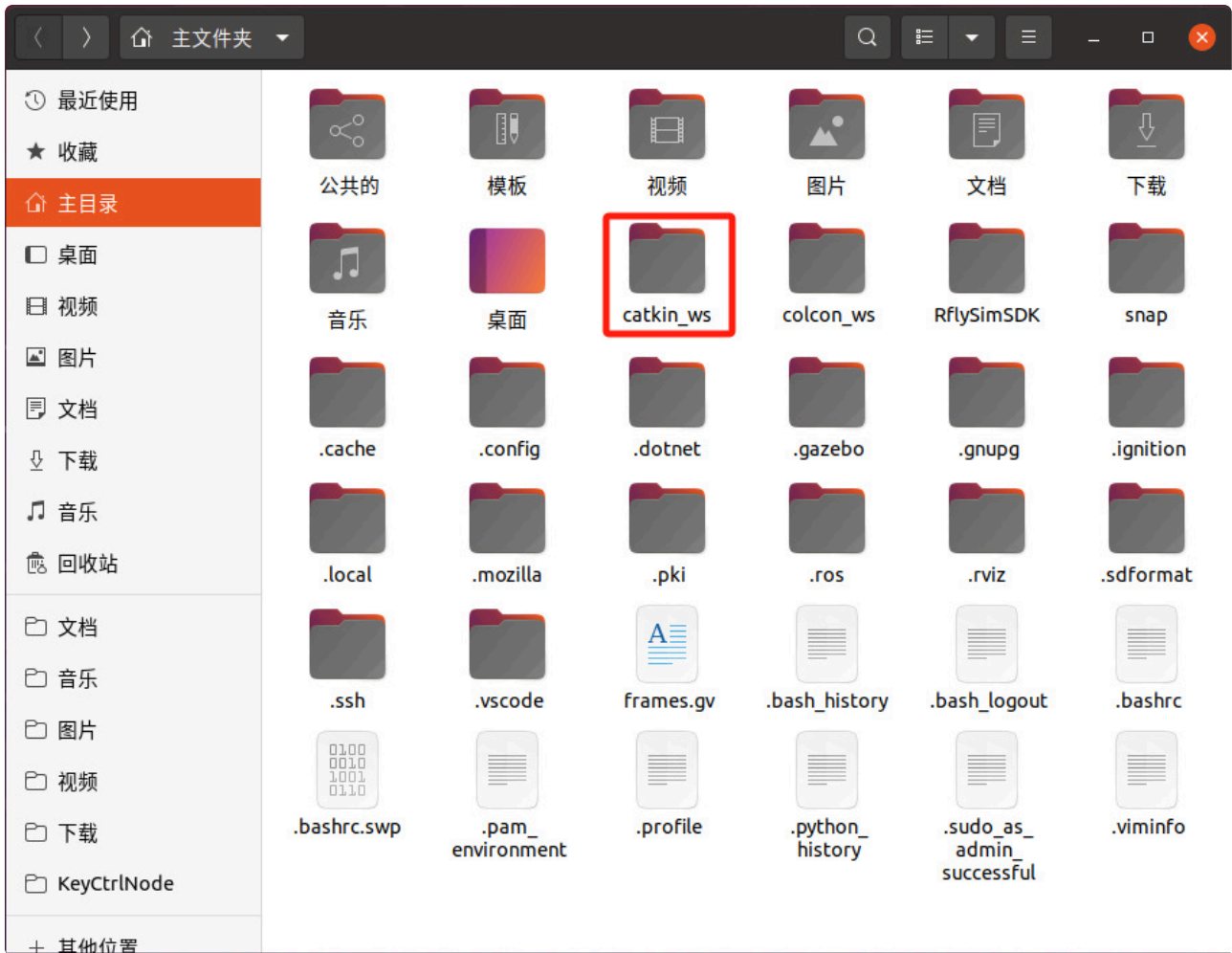
①：推荐配置请见：<https://rflysim.com/doc/zh/HowToInstall.pdf>

5. 实验步骤

5.1 必做实验：WinsWSL控制

Step 1：编译工作空间

1.在ubuntu里打开终端运行命令构建工作空间：`mkdir -p catkin_ws/src`。

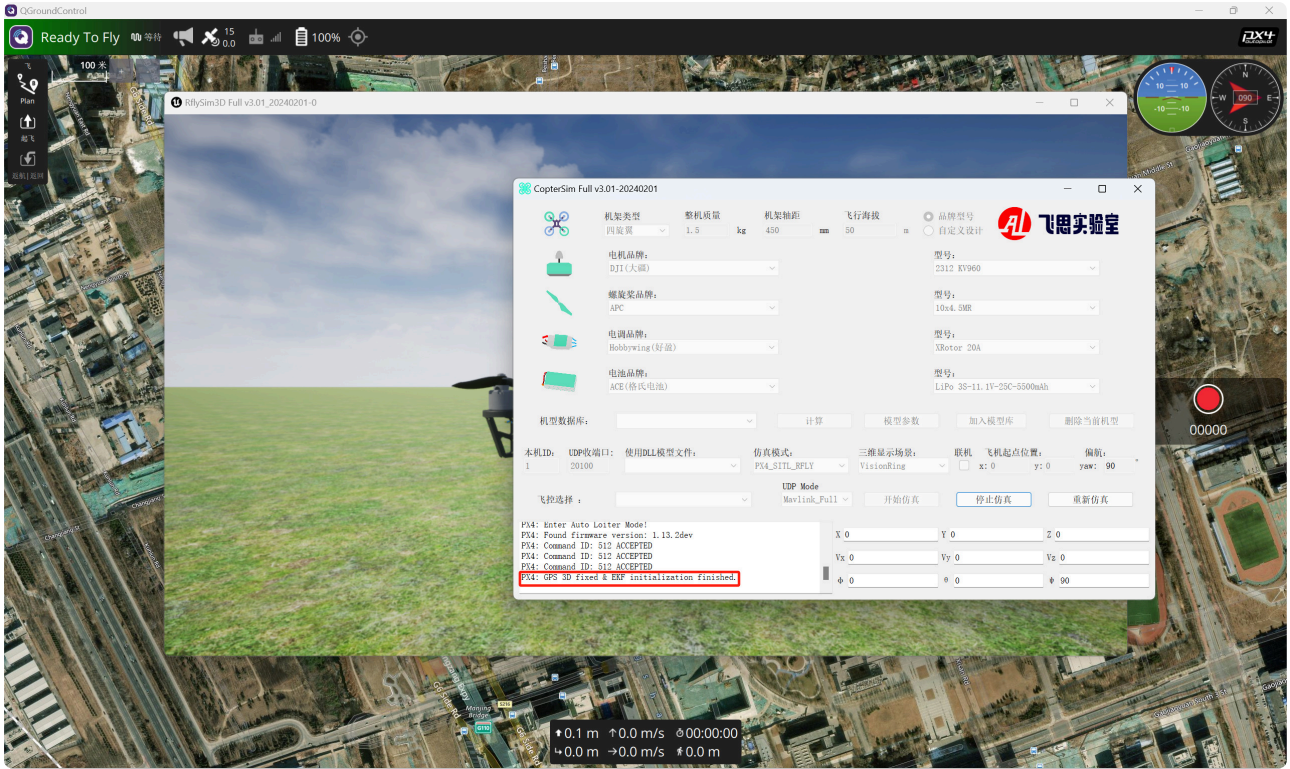


2.把文件vision_ctrl拷贝到 catkin_ws/src里面。

3.编译：编译之前请确保WSL中的ROS环境为ROS Noetic，可运行"*\桌面\RflyTools\RosSwitch.lnk"进行切换。打开终端进入到catkin_ws目录下，运行命令 catkin_make 进行编译。

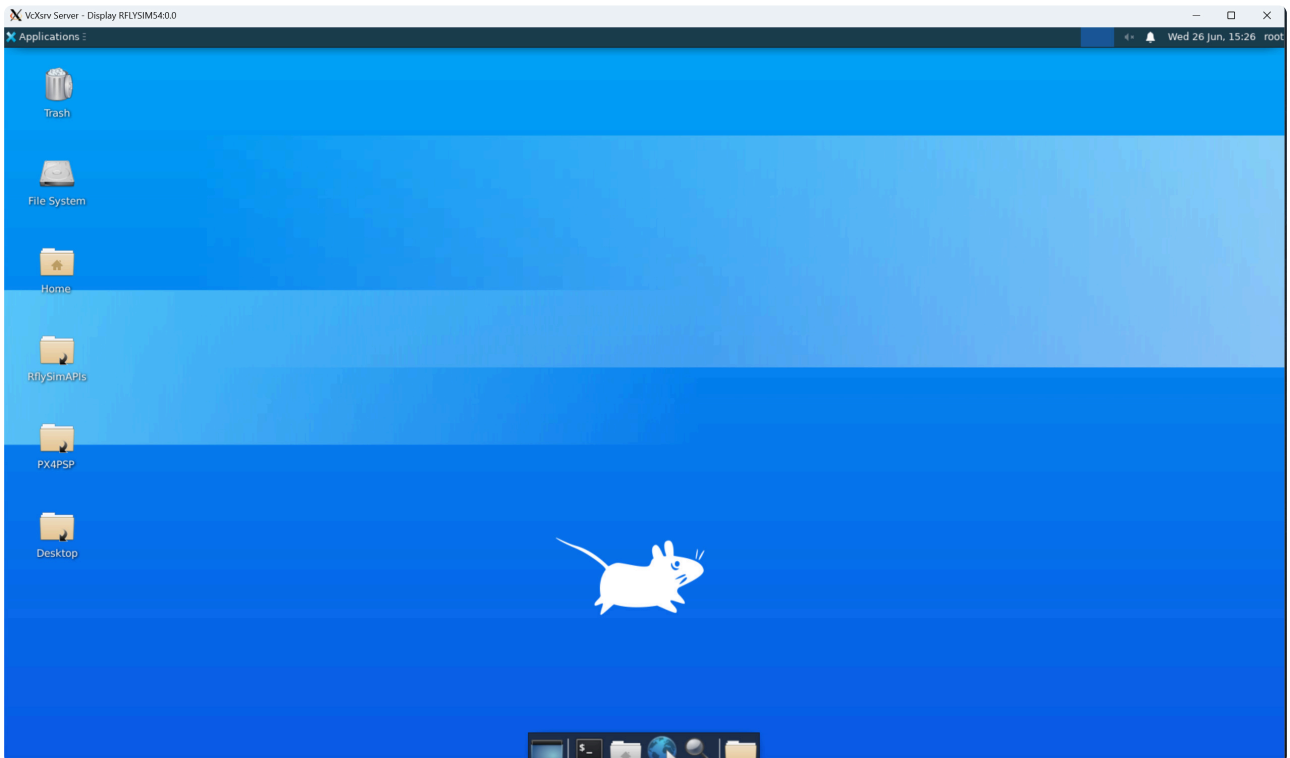
Step 2: 开启仿真

在windows双击运行 [CrossRing3SITL.bat](#) 开启一个软件在环仿真，等待CopterSim日志打印初始化完成。



Step 2: 开启WSL可视化界面

双击打开 [WslGUI.bat](#)，启动WSL可视化界面。（注：如果打开发现窗口白屏，没有桌面，则关了重开一两次。）



注意：参考 [\[安装目录\]\RflySimAPIs\1.RflySimIntro\2.AdvExps\e7_WslGUI\Intro.pdf](#)，了解WslGUI的功能与使用。

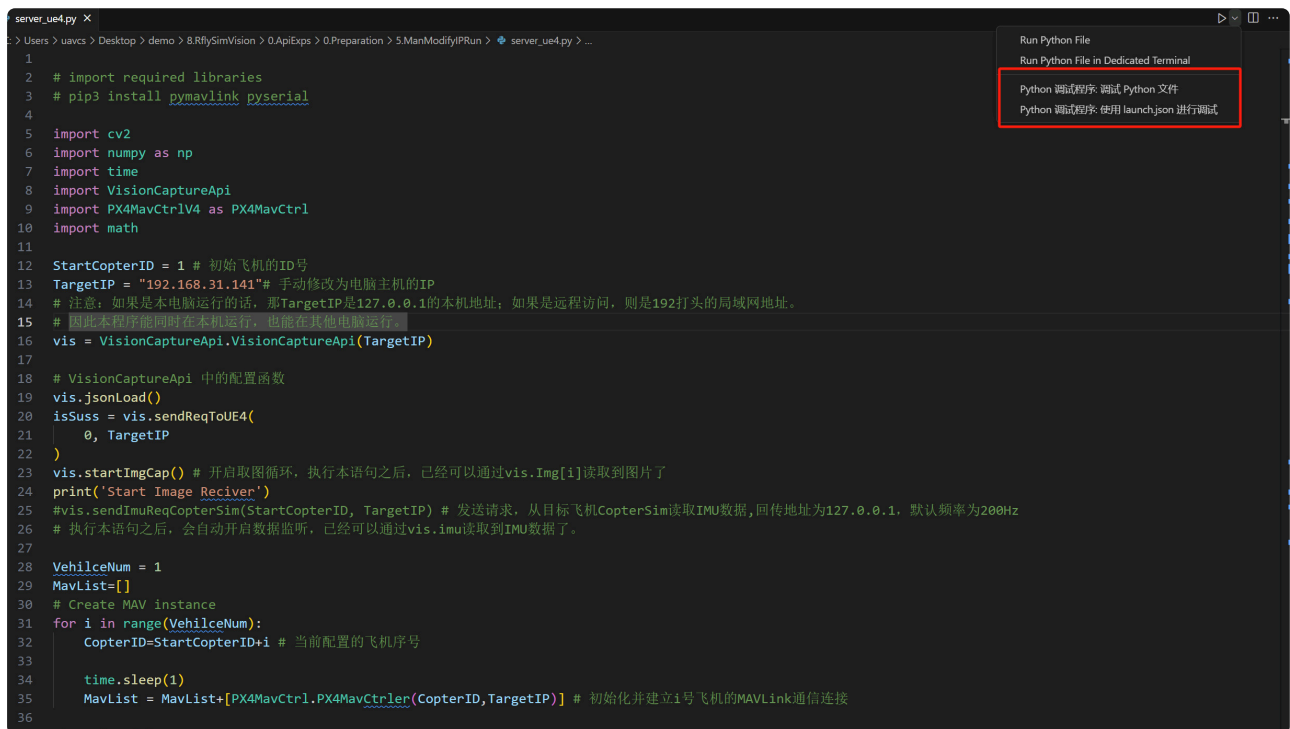
Step 3: 运行控制程序

1. 打开终端进入到catkin_ws/src/vision_ctrl目录下，运行 `python3 mavros.py` 程序启动 mavros。

2. 再此目录下开启新终端运行 `python3 rflysim_img_node.py`

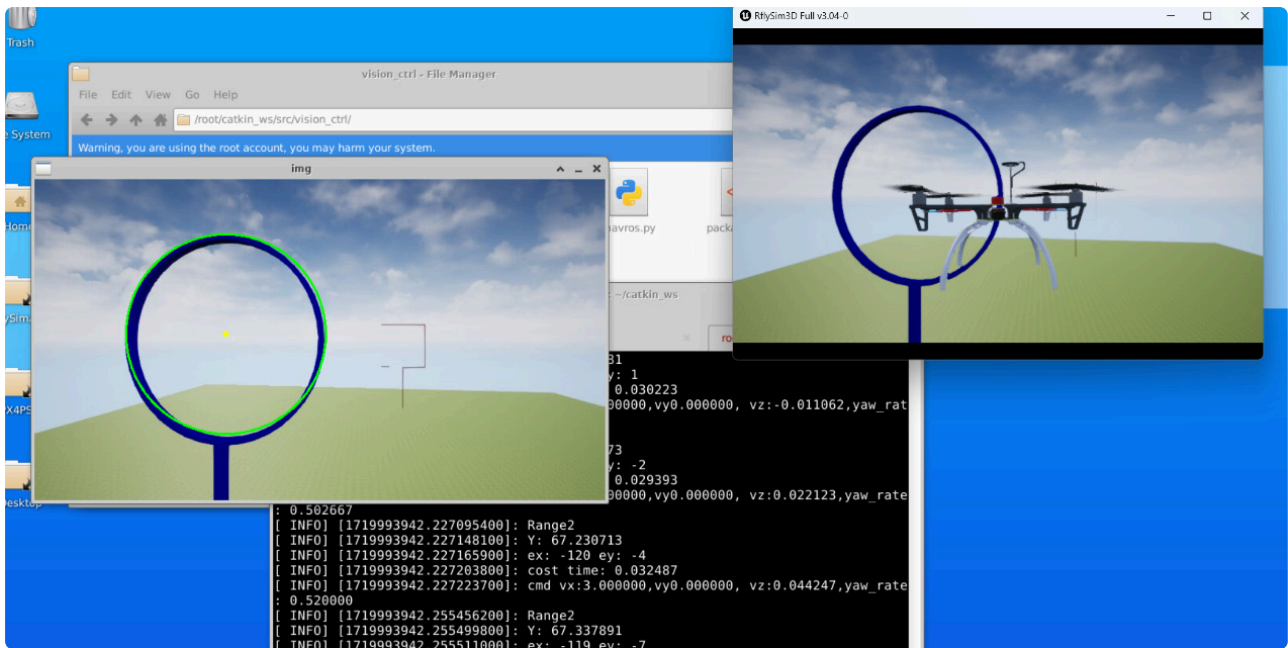
备注：可以参考

[\[安装目录\]\RflySimAPIs\1.RflySimIntro\2.AdvExps\e8.WslVsCode\Intro.pdf](#) 来使用VS Code开发并调试Ubuntu下python文件。



```
server_ue4.py X
> Users > Desktop > demo > 8.RflySimVision > 0.ApiExps > 0.Preparation > 5.ManModifyIPRun > server_ue4.py > ...
1
2 # import required libraries
3 # pip3 install pymavlink pyserial
4
5 import cv2
6 import numpy as np
7 import time
8 import VisionCaptureApi
9 import PX4MavCtrlV4 as PX4MavCtrl
10 import math
11
12 StartCopterID = 1 # 初始飞机的ID号
13 TargetIP = "192.168.31.141" # 手动修改为电脑主机的IP
14 # 注意：如果是本电脑运行的话，那TargetIP是127.0.0.1的本机地址；如果是远程访问，则是192打头的局域网地址。
15 # 因此本程序能同时在本机运行，也能在其他电脑运行。
16 vis = VisionCaptureApi.VisionCaptureApi(TargetIP)
17
18 # VisionCaptureApi 中的配置函数
19 vis.jsonLoad()
20 isSuss = vis.sendReqToUE4(
21     0, TargetIP
22 )
23 vis.startImgCap() # 开启取图循环，执行本语句之后，已经可以通过vis.Img[i]读取到图片了
24 print('Start Image Receiver')
25 #vis.sendImuReqCopterSim(StartCopterID, TargetIP) # 发送请求，从目标飞机CopterSim读取IMU数据，回传地址为127.0.0.1，默认频率为200Hz
26 # 执行本语句之后，会自动开启数据监听，已经可以通过vis.imu读取到IMU数据了。
27
28 VehilceNum = 1
29 MavList=[]
30 # Create MAV instance
31 for i in range(VehilceNum):
32     CopterID=StartCopterID+i # 当前配置的飞机序号
33
34     time.sleep(1)
35     MavList = MavList+[PX4MavCtrl.PX4MavCtrlIer(CopterID,TargetIP)] # 初始化并建立i号飞机的MAVLink通信连接
36
37
```

3. 打开新终端，切换到catkin_ws目录下，运行 `source devel/setup.bash`，再运行运行节点 `roslaunch vision_ctrl visino_ctrl`，可以看到如下图所示的效果。



5.2. 选作实验

准备工作：

虚拟机或NX的配置方法是相同的。

1) Ubuntu虚拟机环境下，进行分布式联机实验。先参考[\[安装目录\]\RflySimAPIs\8.RflySimVision\0.ApiExps\0.Preparation\1.VMwareUbuntu\Readme.pdf](#)，完成虚拟机的下载与配置。

2) 用第二台Ubuntu电脑或NX板卡，实现联机实验。其他Ubuntu电脑的配置，先看[\[安装目录\]\RflySimAPIs\8.RflySimVision\0.ApiExps\0.Preparation\2.GeneralUbuntuConfig\Readme.pdf](#)；NX板卡的配置方法，先看[\[安装目录\]\RflySimAPIs\8.RflySimVision\0.ApiExps\0.Preparation\3.NXwithPX4Config\Readme.pdf](#)

。

扩展实验：

5.2.1在虚拟机/视觉板卡/另一台Ubuntu上接收图像实验

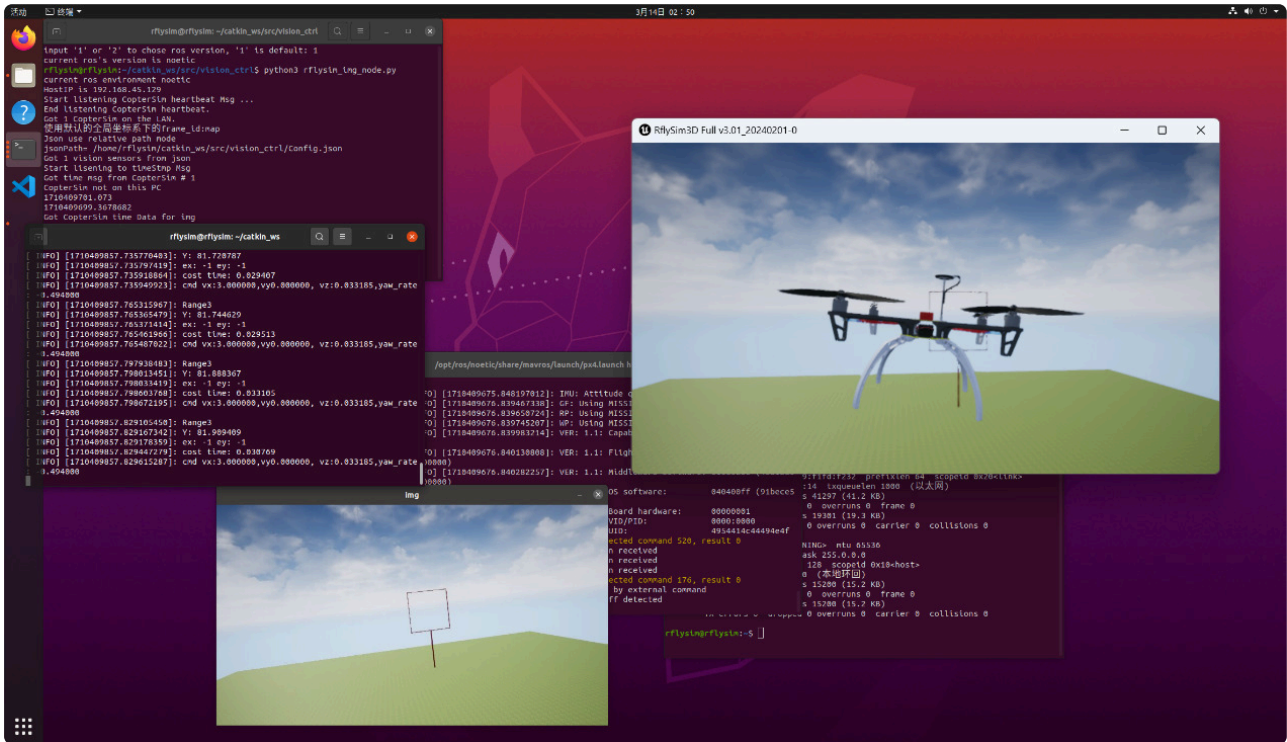
Step 1: 开启仿真

步骤1同上面的Step1步骤。

Step 2: 运行控制程序

1. 打开终端进入到vision_ctrl目录下，运行 `python3 mavros.py` 程序启动mavros。再运行 `rflsysim_img_node.py`

2. 打开终端，切换到catkin_ws目录下，运行 `source devel/setup.bash` ,再运行运行节点 `roslaunch vision_ctrl visino_ctrl` ,可以看到如下图所示的效果。



6.参考资料

无

7.常见问题

Q1: 无

A1: 无