1.实验名称及目的

1.1实验名称

四旋翼分布式轨迹跟踪实验(VMware)：该实验基于领导跟随法的四旋翼无人机轨迹跟踪实验（仅限完整版及以上版本）

1.2实验目的

本实验控制算法从传统的操作系统移植到
Ubuntu，旨在解决不同平台之间的兼容性问题，确保算法能够在Ubuntu操作系统下正常运行。

该实验基于领导跟随法的四旋翼无人机编队，领导跟随法是一种常见的编队控制方法，其基本思想是选取一架无人机作为领导者，其他无人机作为跟随者，通过控制跟随者的位置和速度，使整个编队保持一定的队形和运动状态。在这种方法中，领导者无人机的位置和速度由任务需求确定并通过UDP组网发送给各个跟随者无人机，而跟随者无人机的控制则通过接收前一架无人机的信息并计算得到。

本实验中1号领导者旋翼机接收到起飞指令后先垂直起飞，2、3、4号跟随者旋翼机接收到前列旋翼机的起飞数据后，依次间隔3s起飞。1号领导者旋翼机垂直起飞至10m后实时检测所有跟随者旋翼机数据，并等待其他跟随者旋翼机垂直起飞至同等高度后，开始进入路线规划。

1.3关键知识点

实验中主要实现通过Python接口PX4MavCtrlV4、UE4CtrlAPI、
NetSimAPIV4等，控制飞行器进行任务规划飞行。具体实现思路如下：

下述介绍代码说明。

本段代码的主要功能是获取特定 ID 的 CopterSim 的计算机 IP
地址，并配置该计算机将模拟数据重定向到本机，以便进行进一步的控制和监测。这可以简化网络配置的过程，优化无人机模拟的操作。创建1号飞行器通信mav、ue，与
CopterSim、RflySim3D相连copterSimId
= 1；

ue = UE4CtrlAPI.UE4CtrlAPI()

#
创建一个CopterSim状态获取实例，并监听2s钟，获取局域网内所有CopterSim所在电脑的IP

req = ReqCopterSim.ReqCopterSim()

获取目标电脑IP，并且配置CopterSim回传数据到本电脑

获取到指定CopterID的CopterSim所在电脑的IP

#
注：Windows下运行本函数获取TargetIP=127.0.0.1，如果在其他电脑上运行，获取CopterID对应电脑的IP

TargetIP = req.getSimIpID(copterSimId)

请求目标CopterSim将数据返回到本电脑

通过本接口，可以不用再去bat脚本里面填写IP地址了

req.sendReSimIP(copterSimId)

本段代码的整体功能是初始化无人机控制器和网络模拟接口。mav
实例用于控制无人机的飞行，而 net
实例则用于管理与模拟环境之间的网络通信。这为后续的无人机控制和数据交互提供了基础；

mav = PX4MavCtrl.PX4MavCtrler(copterSimId,TargetIP)

net = NetSimAPIV4.NetSimAPI(mav)

本函数的主要作用是每隔 1
秒打印一次引导无人机和跟随者无人机的状态信息。通过这种方式，可以实时监控无人机的状态和行为，便于调试和分析飞行情况；

def func():

每隔1s打印指定飞机的数据

while True:

if copterSimId != 1:

print('跟随者状态信息：',net.UavData[0].CopterID,net.UavData[0].uavAngEular,net.UavData[0].uavVelNED,net.UavData[0].uavAngEular,net.UavData[0].uavPosNED,net.UavData[0].uavGlobalPos)

print('引航者状态信息:',mav.CopterID,mav.uavAngEular,mav.uavVelNED,mav.uavAngEular,mav.uavPosNED,mav.uavGlobalPos)

time.sleep(1)

本段代码的整体功能是初始化与 PX4 无人机的 MAVLink 通信，并将本机的数据根据指定的 IP 和端口转发给特定的无人机。这种设置允许在多架无人机之间进行同步和数据交换；

#mav.InitMavLoop(UDPMode), where UDPMode=0,1,2,3,4

Use UDP_Simple Mode to control PX4

In this mode, this script will send struct inHILCMDData to CopterSim

Then CopterSim convert it to MAVLinkOffboard message to PX4

In this mode, PX4 send MAVLink data to CopterSim, which convert Struct data to this script

Obviously, UDP Simple is faster (Little data, low delay) than UDP Simple.

这里使用MAVLink_Full模式来传输数据

mav.InitMavLoop()

将本机数据转发给'224.0.0.10' 下的 60002,60003,60004系列端口

这几个端口目前会被#2 #3 #4号飞机订阅，因此实际上是发给了#2 #3 #4号飞机

如果是组网仿真的话，先要发给组网程序的指定IP和端口，再转发到各飞机监听端口

#net.enNetForward([60002,60003,60004],'224.0.0.10')

net.enNetForward([60000 + copterSimId +1],'224.0.0.10')

本段代码的主要功能是检查无人机的状态，确保其准备好飞行后，启用 Offboard 控制。通过这种方式，用户可以在确认无人机状态良好后，开始对其进行控制和指令发送；

print('Check if CopterSim 3D Fixed...')

while True:

if mav.isPX4Ekf3DFixed:

print('CopterSim/PX4 3D Fixed, ready to fly.')

break

time.sleep(0.5)

time.sleep(1)

print(copterSimId)

print('开始起飞：Start Offboard Send!')

启用Offboard控制

mav.initOffboard()

time.sleep(1)

本行代码的主要作用是启动网络接收功能，使程序能够在计算机上监听从 IP 地址
224.0.0.10 发送到 60000 + copterSimId
端口的数据流。这对于实时接收无人机或者其他相关设备的信息至关重要，能够使程序获取并处理从网络传入的数据；

net.StartNetRec(60000 + copterSimId,'224.0.0.10')

本段代码根据无人机的 ID
分别处理引导无人机和跟随无人机的起飞过程。引导无人机直接解锁并起飞，而跟随无人机则等待接收引导无人机的数据后再进行相应的解锁和起飞操作。这种设计确保了多架无人机之间的协调和同步。如果为1号飞行器，则发送ue指令给RflySim3D进行飞行器数量显示和飞行器航迹显示，并解锁起飞，垂直起飞至十米后处于等待模式；对于其他
ID 的无人机，则需要等待接收到数据后再执行相同的起飞命令；

time.sleep(1)

if copterSimId == 1:

ue.sendUE4Cmd('RflyChangeViewKeyCmd S')

ue.sendUE4Cmd('RflyChangeViewKeyCmd T 4')

print('尝试重新解锁飞机')

```
mav.SendMavArm(True)

print('起飞到十米高')

mav.SendPosNED(0, 0, -10, 0)

else:

# 本例子期望接收领航者飞机的数据

while True:

if net.UavData[0].CopterID == copterSimId-1:

break

time.sleep(2)

print('开始执行航迹模式')

print('尝试重新解锁飞机')

mav.SendMavArm(True)

print('起飞到十米高')

mav.SendPosNED(0, 0, -10, 0)
```

本段代码的主要目的是启动一个新的线程，用于并行执行 func
函数的代码，而不影响主程序的其他操作。多线程可以用来执行耗时的任务而不冻结主界面或允许其他操作同时进行；

```
dance_thread = threading.Thread(target=func)

dance_thread.start()

time.sleep(0.5)
```

本段代码实现了一个基本的无人机控制逻辑。引导无人机在前 10
秒内向下飞行到指定高度，然后进入一个圆形轨迹飞行；而跟随无人机则根据领航无人机的位置动态调整自身速度，以保持跟随状态。整个过程通过简易定时器确保以
30Hz 的频率执行；

```
lastTime = time.time()

startTime = time.time()
```

```
timeInterval = 1/30.0 #here is 0.0333s (30Hz)

if copterSimId == 1:

Error2UE4Map=[]

Error2UE4Map =
Error2UE4Map+[-np.array([mav.uavGlobalPos[0]-
mav.uavPosNED[0],mav.uavGlobalPos[1]-mav.uavPosNED[1],mav.uavGlobalPos[2]-
mav.uavPosNED[2]])]

while True:

lastTime = lastTime + timeInterval

sleepTime = lastTime - time.time()

if sleepTime > 0:

time.sleep(sleepTime) # 未到期望时间，就休息

else:

lastTime = time.time()

# 上面的程序创建了一个简易定时器，保证代码以30Hz频率执行循环

# 在下面开发你的控制算法

t=time.time()-startTime

# target position in UE4 map global frame

if t<10:

targetPosE=np.array([-0,0,-20]) # fly to 15 meters high in 0 to 10s

else: #fly circle after 10s

targetPosE=np.array([10*math.sin(t/2+math.pi/2)-10,10*math.sin(t/2.0),-15])

# target position in vehilce takeoff frame

targetPosE=targetPosE+Error2UE4Map[0]

mav.SendPosNED(targetPosE[0],targetPosE[1],targetPosE[2],0)
```

else:

while True:

lastTime = lastTime + timeInterval

sleepTime = lastTime - time.time()

if sleepTime > 0:

time.sleep(sleepTime) # 未到期望时间，就休息

else:

lastTime = time.time()

上面的程序创建了一个简易定时器，保证代码以30Hz频率执行循环

mav.SendVelNED(net.UavData[0].uavGlobalPos[0]-mav.uavGlobalPos[0] ,net.UavData[0].uavGlobalPos[1]-mav.uavGlobalPos[1],net.UavData[0].uavGlobalPos[2]-mav.uavGlobalPos[2], 0) 1号飞行器解锁具备起飞状态，飞行器进入外部模式（offboard）；

2.实验效果

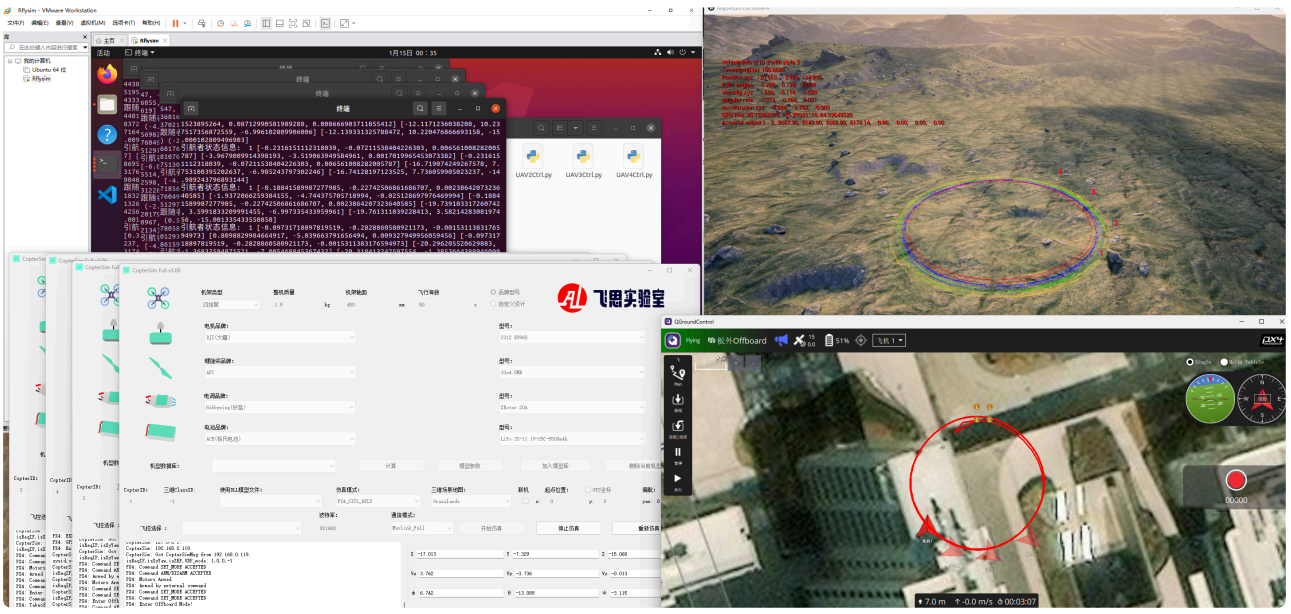

图 1 实验效果

3.文件目录

文件夹/文件名称	说明
Python38Run.bat	WSL启动脚本
UbuntuRullAll.sh	虚拟机运行脚本
SITLRun4MavlinkFull.bat	软件在环仿真脚本
UAV1Ctrl.py	领航者控制程序
UAV2Ctrl.py	跟随者1号控制程序
UAV3Ctrl.py	跟随者2号控制程序
UAV4Ctrl.py	跟随者3号控制程序
Readme.pdf	用户指南

4.运行环境

4.1 软件要求

Windows 10及以上版本；RflySim平台完整版及以上；MATLAB R2022B。

> 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

台式电脑/笔记本 1台。

> ①：推荐配置请见：https://rflysim.com/

5.实验步骤

RflySim平台初始化

以管理员身份运行SITLRun4MavlinkFull.bat，启动仿真脚本，等待旋翼机飞行器初始化完成，即CopterSim界面返回语句“GPS
3D fixed & EKF initialization finished ”和“Enter Auto Loiter Mode”即可。

图 2 CopterSim等待界面

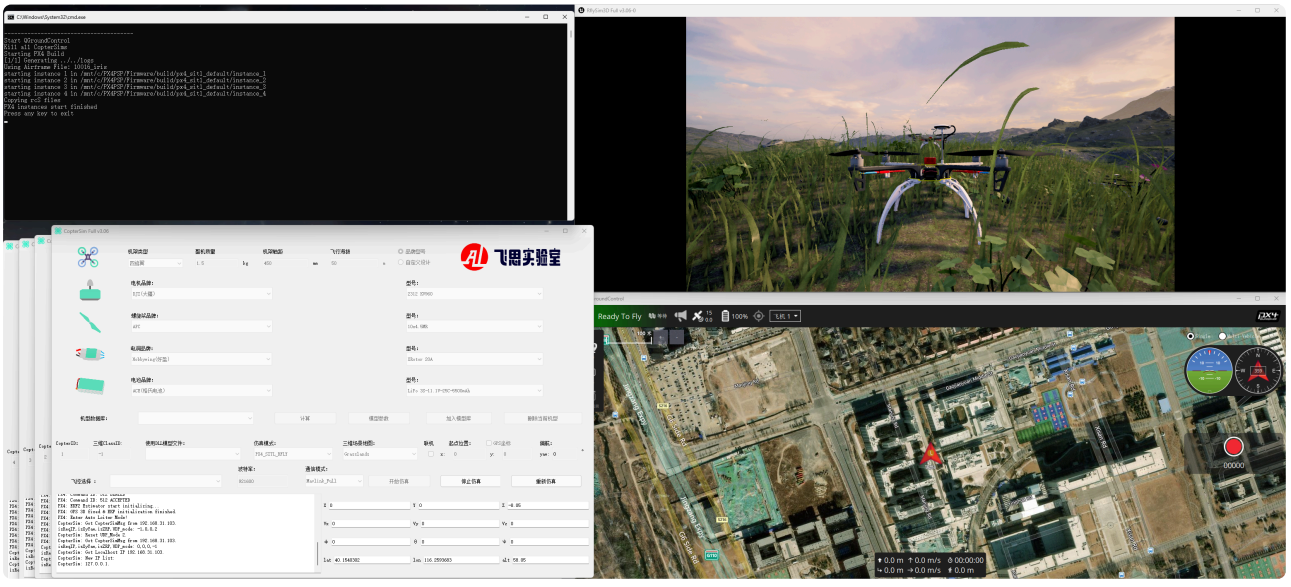

图 3 RflySim工具链初始化

安装虚拟机- rflysim

飞思实验室提供基于RflySim工具链开发的ubuntu-20.04镜像，用户可通过以下链接进行下载安装：

https://pan.baidu.com/s/10MDjINKG20k4mWUYz0Nm1A?pwd=78r7

具体注意事项请看本文件第7节常见问题。本章节不介绍虚拟机安装及操作，虚拟机部分详见：…/PX4PSP/RflySimAPIs/2.RflySimUsage/0.ApiExps/e5_Ubuntu/Readme.pdf。

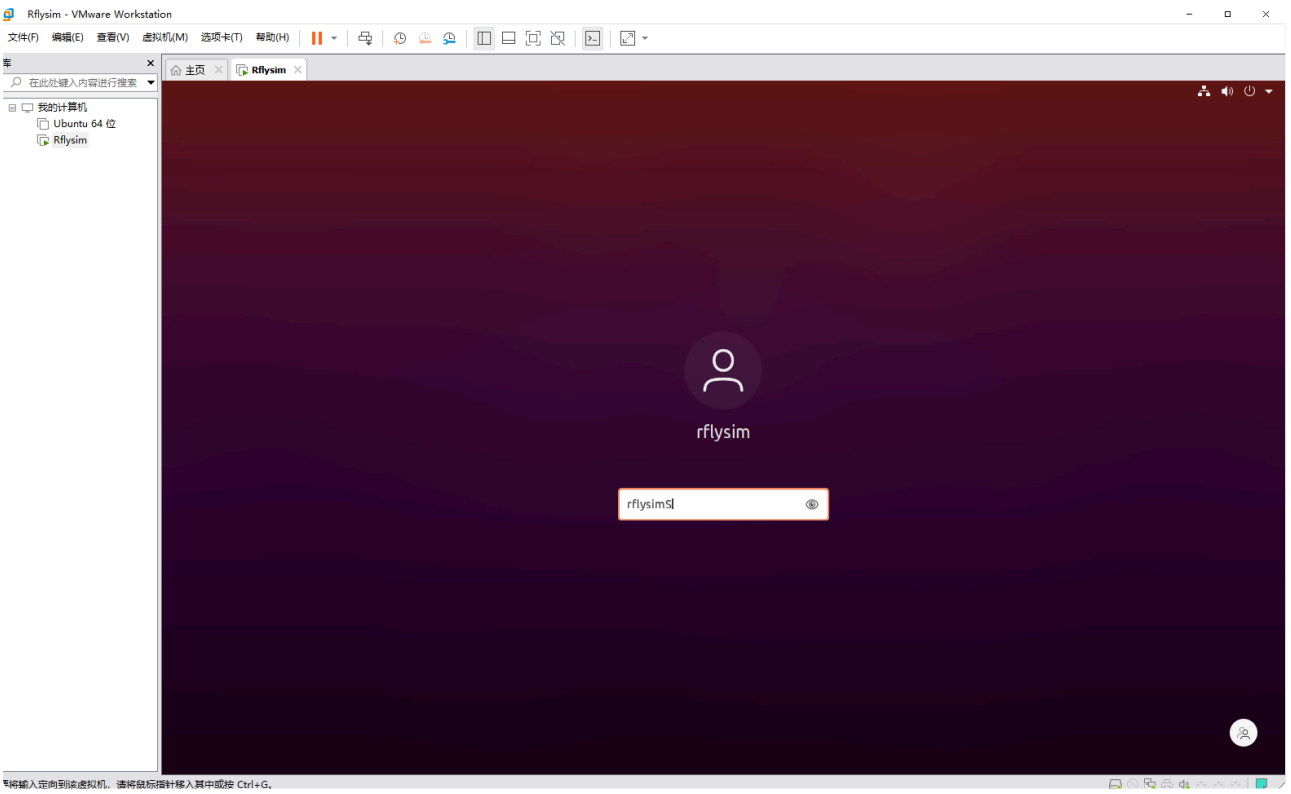

图 4 飞思rflysim成功安装界面

RflySimSDK部署

将Windows端安装的RflySimSDK文件复制。（注意：每次Windows平台RflySim更新后，都必须更新虚拟机里面的RflySimSDK文件）

1. 将安装目录下的“PX4PSP\RflySimAPIs\RflySimSDK”文件夹，复制，并在虚拟机主目录（如下图）上粘贴，将文件夹拷贝过去。

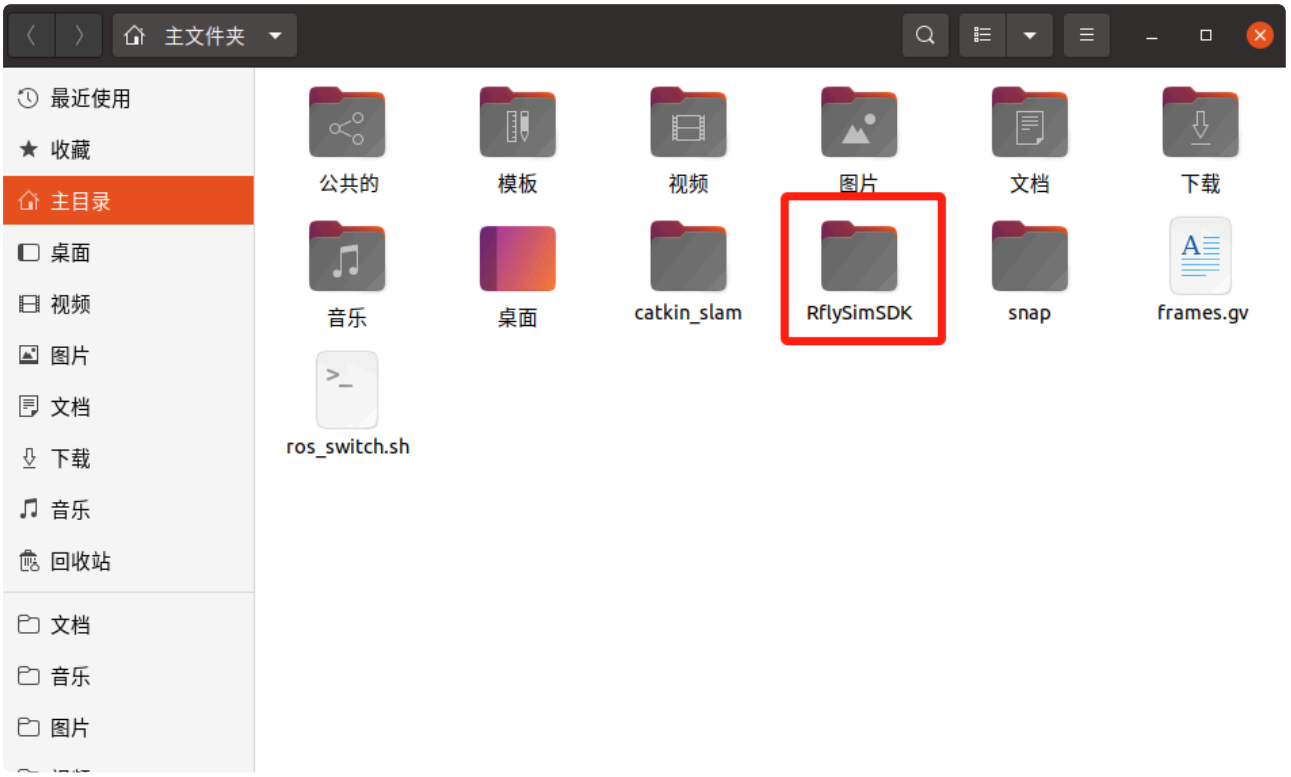

图 5 RflySimSDK安装位置

注意：如果平台之前导入过RflySimSDK，那么只需要复制新RflySimSDK文件夹并覆盖原文件夹即可。不需要再运行后文的ReLabPath.py。

2. 进入虚拟机内的RflySimSDK文件夹，空白处点击鼠标右键，选择“在终端打开”；

图 6 打开终端

3. 输入命令“python3 ReLabPath.py”将RflySimSDK的库加载进入Ubuntu环境，最后输入密码“rflysim”（注意linux系统输入密码时不会显示，输入rflysim直接回车即可），确认库的导入；

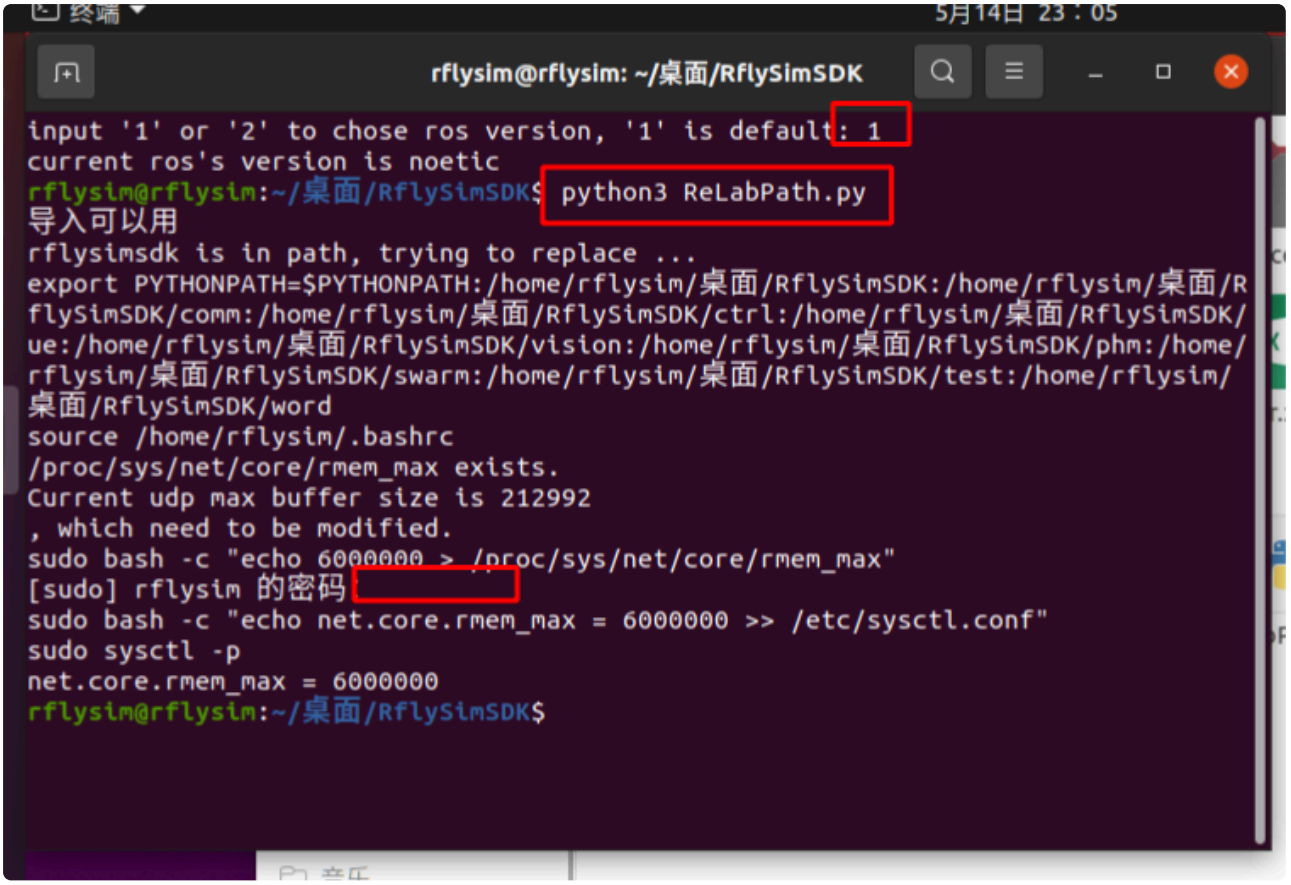

图 7 RflySimSDK的库加载进入Ubuntu环境

4. 然后，输入命令cat ~/.bashrc ，
   如果最后能看到RflySimSDK的目录注册，说明环境配置正确。

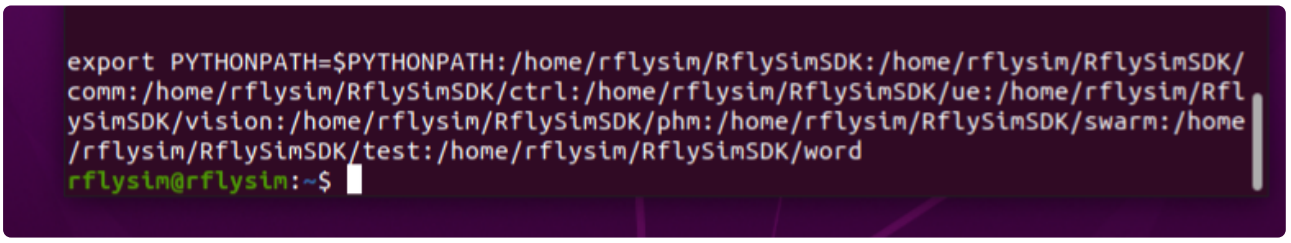

图 8 检验目录注册表

案例测试

将Windows下本实验的所有文件复制到虚拟机桌面上，双击打开实验文件夹，并右键打开终端。

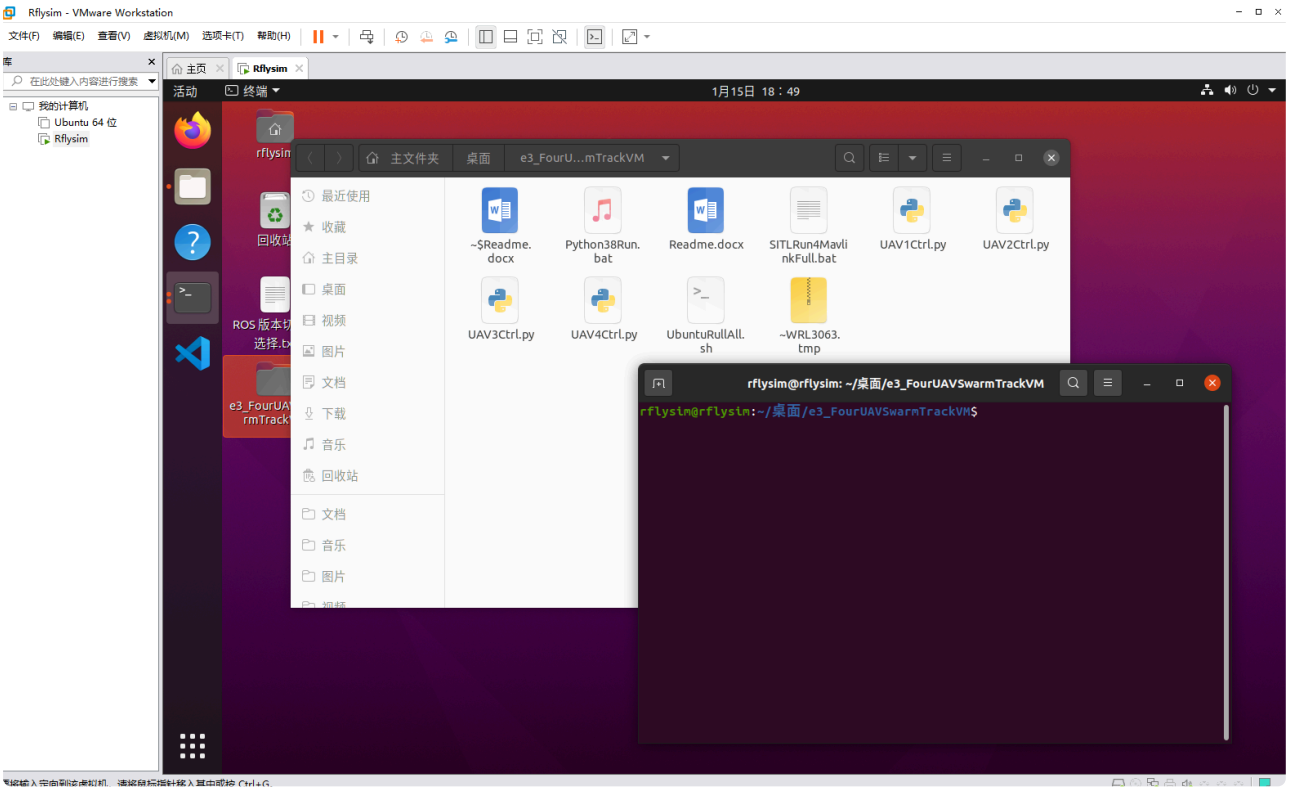

图 9 实验准备阶段

在终端中输入“./UbuntuRullAll.sh”，回车。

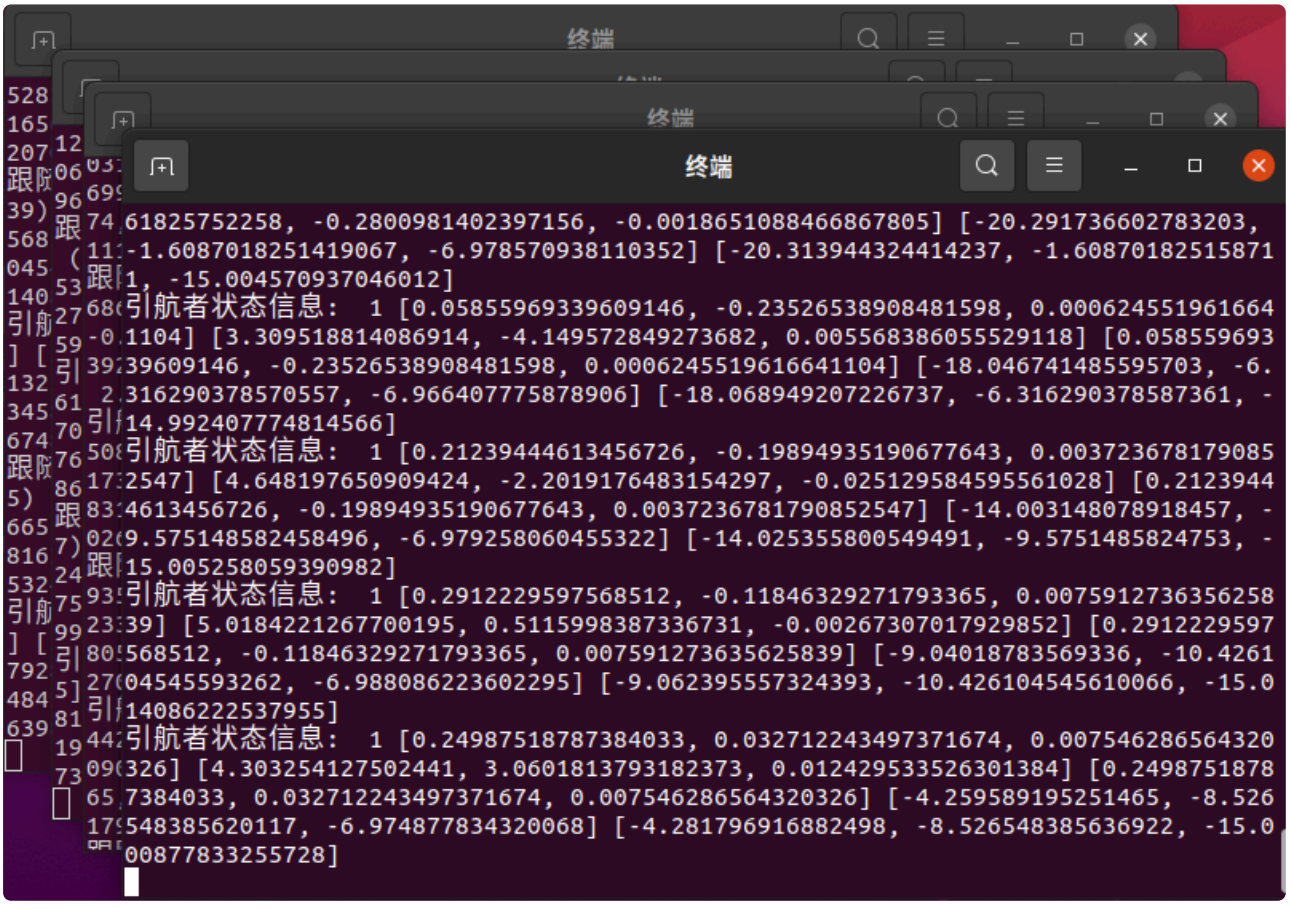

图 10 控制程序临近感知信息

方法二：

打开当前实验目录下，打开四个终端分别运行命令python3 UAV1Ctrl.py、UAV2Ctrl.py、UAV3Ctrl.py、UAV4Ctrl.py；

自动启动python脚本，并依次自动运行四架旋翼机的控制程序，可看到各个旋翼机临近感知的信息。

观察状态- RflySim3D

观察RflySim3D查看各个旋翼机飞行状态。

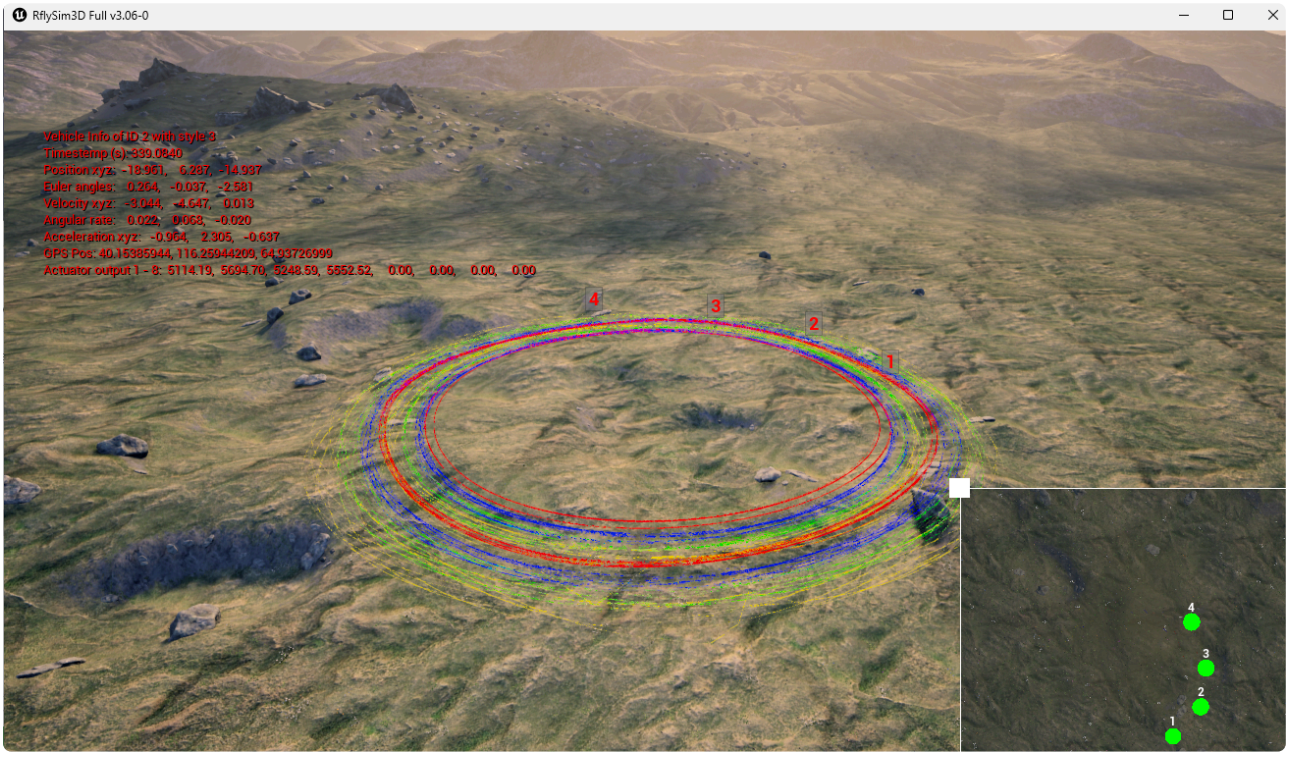

图 11 RflySim3D展示效果

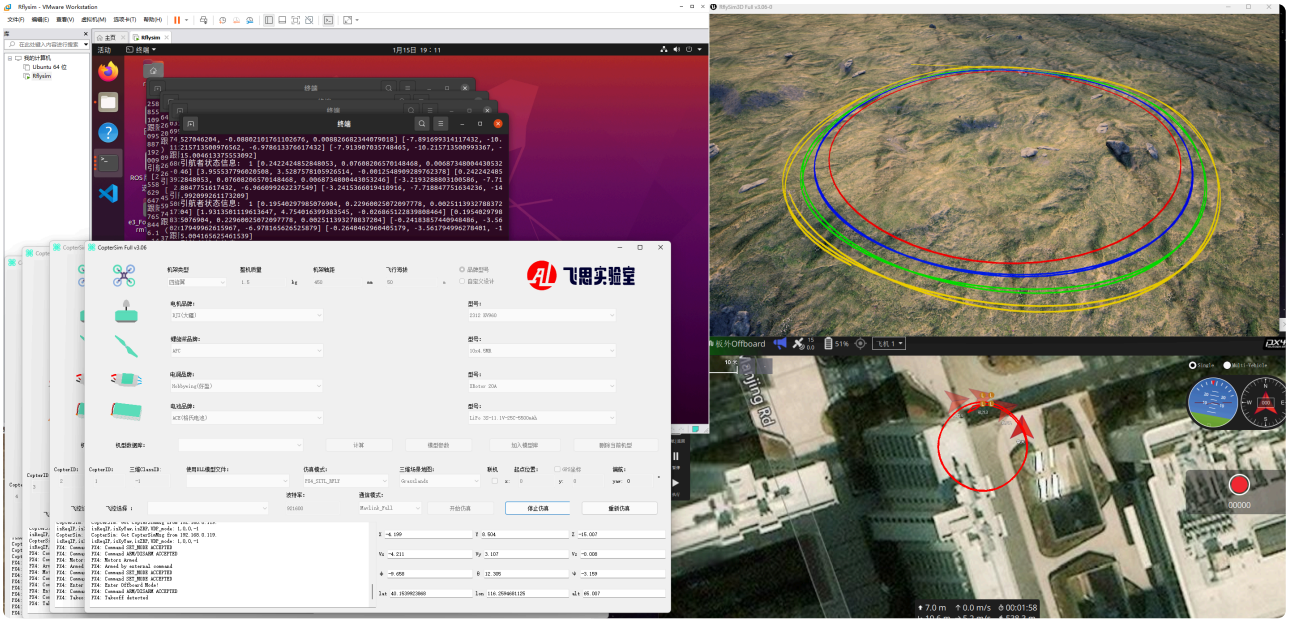

图 12 RflySim实验展示

6.参考资料

1. 无。

7.常见问题

Q1：在虚拟机Ubuntu运行UAV1Ctrl.py文件时，出现“ModuleNotFoundError : No Momodu
named ‘psutil’”。

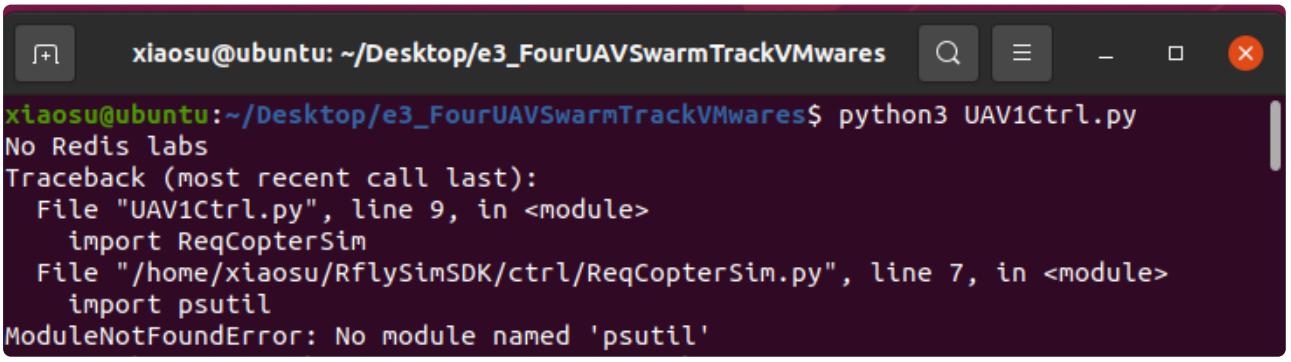

图 13 脚本运行异常

A1：在命令行中输入：sudo pip install psutil

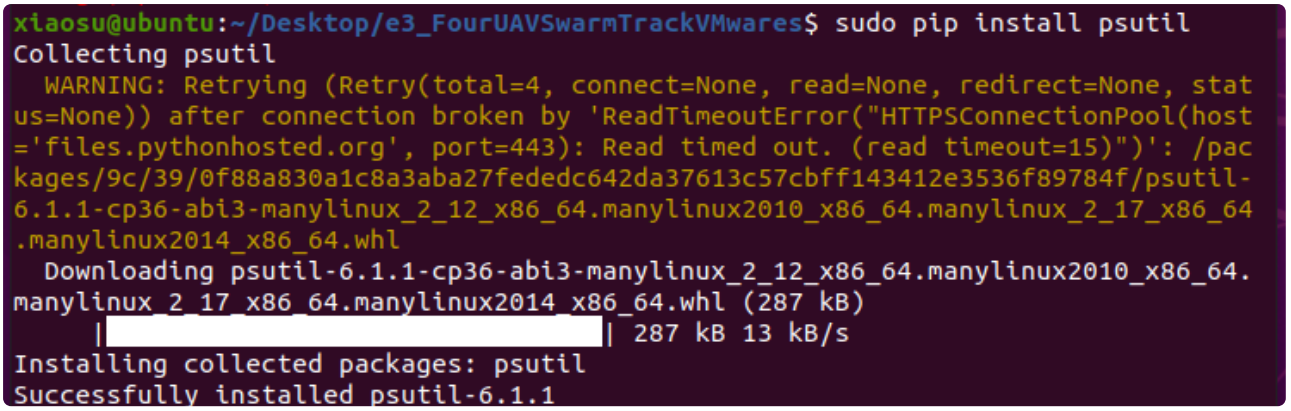

图 14 安装psutil库

Q2:无人机出现1号飞行器正常起飞，2、3、4号飞行器起飞异常?

A2:请更改虚拟机配置，虚拟机界面下点击—虚拟机—设置，打开如下界面，选择桥接模式后确认重启虚拟机。

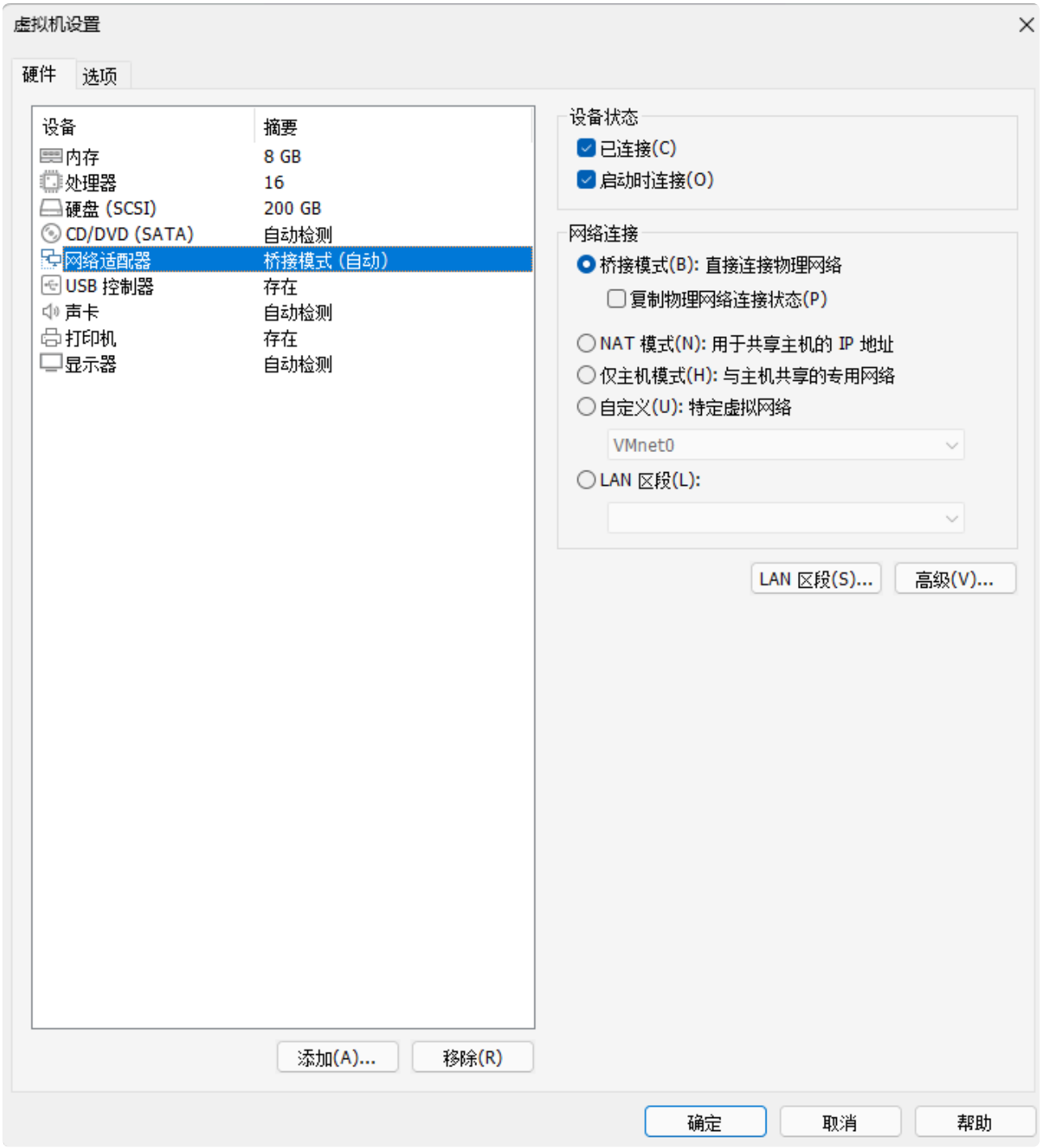

图 15 虚拟机桥接模式

Q3:在实验中飞行器出现姿态紊乱情况?

A3:此现象为数据干扰，请关机重启或电脑单独使用路由器连接。

Q3:在虚拟机中运行UbuntuRullAll.sh时，提示权限不足?

A3:在终端输入：sudo chmod +x UbuntuRullAll.sh。