1.实验名称及目的

1.1 实验名称

飞机、物体、相机信息获取实验

1.2 实验目的

通过python接口获取飞机、物体和相机的信息。

注意：本实验只支持Windows下Python环境运行。不支持在WSL以及虚拟机下运行。

1.3 关键知识点

本实验主要是实现通过Python接口UE4CtrlAPI.py（见RflySimAPIs\RflySimSDK\ue目录）获取获取飞机、物体和相机的信息。关键代码解析如下：

1）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：

```
"SeqID":0：使用自动更新ID的方式，创建了SeqID为0和1的两个视觉传感器

"TypeID":1：传感器类型为RGB彩色图像

"TargetCopter":1：相机绑定在1号飞机上

"SendProtocol":[0,0,0,0,0,0,0,0]：传输模式为0共享内存机制，因此本例程只能运行在Windows环境下。

"SensorPosXYZ":[0.3,-0.15,0]和"SensorPosXYZ":[0.3,0.15,0]：两个RGB相机一左一右分布。
```

2）飞机控制指令

```
mav = PX4MavCtrl.PX4MavCtrler() \# 创建飞机控制实例
```

3）UE控制

接口详细使用方法见：UE4CtrlAPI.py

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

ue.sendUE4Pos(100,30,0,[2.5,0,-8.086],[0,0,math.pi]) \#创建障碍物，id号为100，类型为30（人

ue.sendUE4PosScale(101,2030,0,[10.5,0,-8.086],[0,0,math.pi],[10,10,10]) \#创建障碍物，i

ue.reqCamCoptObj(1,[TargetCopterID,100]) \#发送请求到RflySim3D，返回飞机数据，1和100号飞机。

ue.initUE4MsgRec() \# 开始飞机数据的监听，数据存储在inReqVect列表（碰撞数据）中

Copt = ue.getCamCoptObj(1,TargetCopterID) \#获取目标1号飞机的结构体引用，方便后面的数据输出
```

4）其余代码说明

```
timeInterval = 1/30.0 \# 以30hz的频率进行控制

lastTime = lastTime + timeInterval \# 设置每一帧的处理结束时间

sleepTime = lastTime - time.time() \#计算休息时间，从而保持按照设定的频率执行代码

targetPosE=targetPosE+Error2UE4Map[j] \# 设置飞机位置
```

2.实验效果

本实验通过python接口获取飞机、物体和相机的信息，详细原理见“求取标志点协议v3.txt”。

3.文件目录

例程目录：
[安装目录]\RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\1.ImgSenorAPI\4.GetCamObjDemo

文件夹/文件名称	说明
GetCamObjDemo.bat	一键仿真启动脚本

文件夹/文件名称	说明
GetCamObjDemo.py	Python实验代码
GetCamObjDemoWithCam.py	Python实验代码
Config.json	视觉传感器配置文件
Python38Run.bat	Python环境启动脚本

4.运行环境

4.1 软件要求

Windows 10及以上版本；RflySim工具链；VS Code。

> 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。

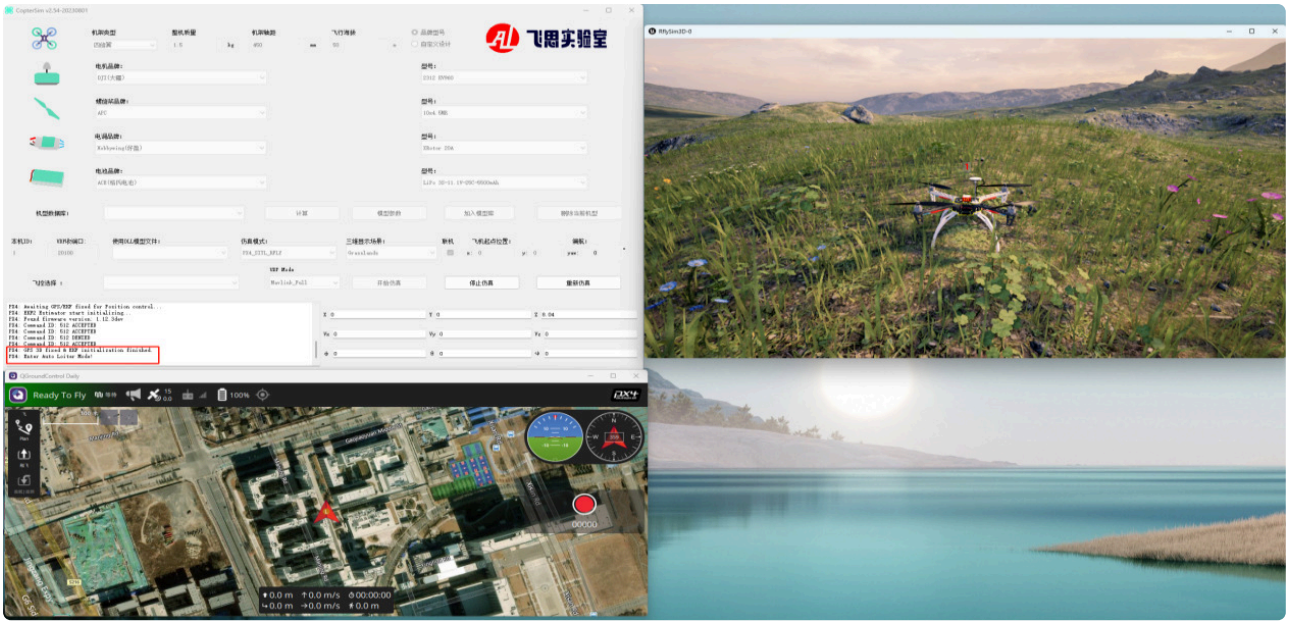
> ①：推荐配置请见：https://rflysim.com/doc/zh/HowToInstall.pdf

5.实验步骤

5.1 必做实验：Windows取图控制

Step 1：开启仿真

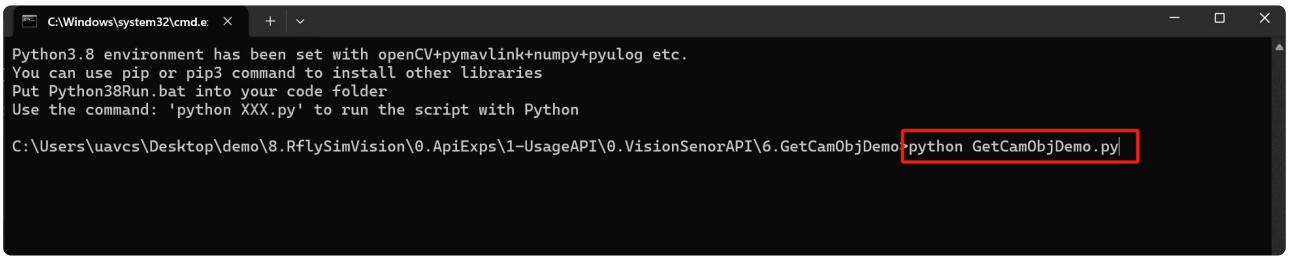
双击运行GetCamObjDemo.bat，开启一个飞机的软件在环仿真。将会启动1个QGC地面站，1个CopterSim软件且其软件下侧日志栏必须打印出GPS 3D fixed & EKF initialization finished字样代表初始化完成，并且RflySim3D软件内有1架无人机。



Step 2：运行控制程序

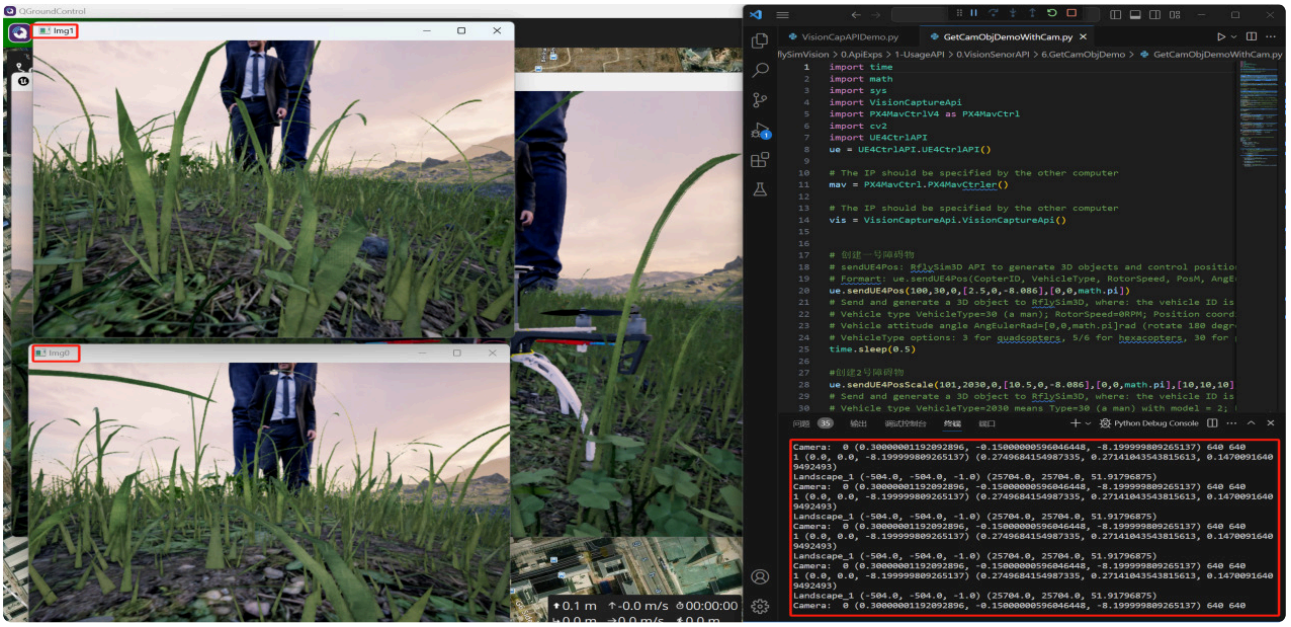
在文件夹下，双击Python38Run.bat，打开集成好的python环境，在该环境下运行GetCamObjDemoWithCam.py.py文件，输入

```
python GetCamObjDemoWithCam.py.py
```



Step3：观察结果

飞机成功起飞，终端可以获取到飞机和物体的信息，以及摄像头窗口，创建的几个物体，如下图所示。



Step 4：结束仿真

在下图“GetCamObjDemo.bat”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。

5.2 选作实验（VS Code调试运行）

准备工作

- 先确保已经按RflySimAPIs\1.RflySimIntro\2.AdvExps\e3.PythonConfig\Readme.pdf步骤，正确配置VS Code环境。或者配置了自己的Pycharm等自定义Python环境。
- 其他步骤与上文相同，在Step2运行GetCamObjDemo.py时，可使用VS Code（或Pycharm等工具）来打开GetCamObjDemo.py文件，并阅读代码，修改代码，调试执行等。

扩展实验

- 请自行使用VS Code阅读GetCamObjDemo.py源码，通过程序跳转，了解每条代码的执行原理；再通过调试工具，验证每条指令的执行效果。

VisionCapAPIDemo.py

> RflySimAPIs > 8.RflySimVision > 0.ApiExps > 1-UsageAPI > 0.VisionSenorAPI > 1.Came

```
ue = UE4CtrlAPI.UE4CtrlAPI()

#Create a new MAVLink communication instance, UDP sending
mav = PX4MavCtrl.PX4MavCtrler(1)

# The IP should be specified by the other computer
vis = VisionCaptureApi.VisionCaptureApi()

# Send command to UE4 Window 1 to change resolution
ue.sendUE4Cmd('r.setres 1280x720w',0) # 设置UE4窗口分辨率，
ue.sendUE4Cmd('t.MaxFPS 30',0) # 设置UE4最大刷新频率，同时也
time.sleep(2)

# VisionCaptureApi 中的配置函数
vis.jsonLoad() # 加载Config.json中的传感器配置文件
```

- 请尝试修改代码，实现飞机位置改变、相机姿态角改变、相机参数改变等功能。

6.参考资料

求取标志点协议v3：

一、接口介绍：

1.请求相机、飞机、物体数据返回的接口用UE命令行RflyReqObjData来实现，可以通过UDP发送命令，或直接在RflySim3D中输入命令。

2. 相机、飞机和物体的类型ID分别是0 1 2。

3.发送请求数据后，在RflySIM3D中创建待发送相机+飞机+物体列表，以后每个Tick会将相机和物体结构体发布出去（注，只有位置移动时才会每帧更新，否则每1s更新一次）。

4.飞机+物体发布的结构体为CoptReqData和ObjReqData，包含了校验位，物体位置PosUE，物体姿态angEuler，物体立方体原点boxOrigin，物体立方体长宽高BoxExtent（见后文注释）。

5.相机发布的结构体为CameraData，包含了校验位，相机ID号（json中定义的），相机位置，相机姿态等。

6.如果只需要计算某一个物体在某一个相机的像素坐标，那么就只需要请求一个相机+一个物体，并进行如下处理：

1）确定物体几何中心的位置（通常是boxOrigin，需要与物体坐标中心区别）。很多情况下，PosUE是以底部中心为坐标中心原点（人为选定），boxOrigin才是物体的几何中心（随着一些舵面偏转，中心会移动，可能有所偏移）。

2）以boxOrigin+BoxExtent+angEuler，确定八个顶点的三维坐标。

3）求八个顶点+中心点三维坐标，映射到相机二维平面的像素坐标。

4）在图片中画一个框，将物体框出来。

5）在点云中画一个长方体的框，将点云分割出来。

7. 如果要计算多个物体，相对多（单）个相机的相对关系与像素坐标，可重复上述步骤，方法类似。

二、发送请求数据，在RflySIM3D中创建待发送相机+物体+飞机列表，以后每个Tick会将相机和物体结构体发布出去

```
void RflyReqObjData(int opFlag, FString objName, FString colorflag);
```

对应命令行

```
RflyReqObjData opFlag objName colorflag
```

标志位说明如下：

Opflag操作标识符:

0-> 创建一个相机，1->创建一个飞机，2->创建一个物体

10-> 删除一个相机，11->删除一个飞机，12->删除一个物体

20->清除所有相机，21->清除所有飞机，22->清除所有物体，23->清除所有物体+飞机，24->清除所有物体、飞机+相机

ObjName物体名字: 飞机->
CopterID号的数字（1、2、3格式），场景物体->双击物体确定名字（Landscape_1格式），相机->相机的ID的序号（0、1、2、3格式）

colorflag颜色标志（将来用于生成分割图用，目前未启用）：用于设置飞机显示颜色；可以red/white等字符串，对应了一个颜色字符串；也可以是int型的纯数字，对应了颜色列表的序号；也可以是255:0:255这种格式的RGB数值。

注：colorflag对应了分割图中显示的颜色。

三、物体和飞机信息发布协议：

```
struct CoptReqData { //64，发送飞机数据

int checksum = 0; //接收端需确认1234567891作为校验

int CopterID;//飞机ID

float PosUE[3];
//物体中心位置（人为三维建模时指定，姿态坐标轴，不一定在几何中心）

float angEuler[3];//物体欧拉角

float boxOrigin[3];//物体几何中心坐标（相对于PosUE）

float BoxExtent[3];//物体外框长宽高的一半

double timestmp;//时间戳

};

struct ObjReqData { //96，发送物体数据

int checksum = 0; //接收端需确认1234567891作为校验

int seqID = 0;

float PosUE[3];
//物体中心位置（人为三维建模时指定，姿态坐标轴，不一定在几何中心）

float angEuler[3];//物体欧拉角

float boxOrigin[3];//物体几何中心坐标（相对于PosUE）

float BoxExtent[3];//物体外框长宽高的一半

double timestmp;//时间戳

char ObjName[32] = { 0 };//碰物体的名字

};

PosUE 物体的位置：以设定的物体中心。

boxOrigin Set to the center of the actor in world space；物体的几何中心所在位置

BoxExtent Set to half the actor's size in 3d space；物体的半直径。
```

四、相机信息发布协议（主体和json相同）：

```
struct CameraData { //56

int checksum = 0;//接收端需确认1234567891作为校验

int SeqID; //相机序号

int TypeID;//相机类型

int DataHeight;//像素高

int DataWidth;//像素宽

float CameraFOV;//相机视场角

float PosUE[3]; //相机中心位置

float angEuler[3];//相机欧拉角

double timestmp;//时间戳

};
```

注意：物体和相机的数据结构体，会议组播形式，传输给

五、所有数据包会发送到组播IP地址

组播地址224.0.0.10的20006端口

7.常见问题

Q1：无

A1：无