

# 1、实验名称及目的

飞机、物体、相机信息获取实验：通过 python 接口获取飞机、物体和相机的信息。

# 2、实验原理

首先进行 Config.json 文件进行相机传感器的配置，配置参数如下解释：

“SeqID”代表第几个传感器。此处 0、1、分别表示第 1、2 个传感器（免费版只支持 2 个图）。

“TypeID”代表传感器类型 ID，1:RGB 图（免费版只支持 RGB 图），2:深度图，3:灰度图。

“TargetCopter”传感器装载的目标飞机的 ID，可改变。

“TargetMountType”代表坐标类型，0：固定飞机上（相对几何中心），1：固定飞机上（相对底部中心），2：固定地面上（监控）也可变。

“DataWidth”为数据或图像宽度此处为 640，“DataHeight”为数据或图像高度此处为 480。

“DataCheckFreq”检查数据更新频率此处为 30HZ。

“SendProtocol[8]”为传输方式与地址，SendProtocol[0]取值 0：共享内存（免费版只支持共享内存），1：UDP 直传 png 压缩，2：UDP 直传图片不压缩，3：UDP 直传 jpg 压缩；SendProtocol[1-4]：IP 地址；SendProtocol[5]端口号。

“CameraFOV”为相机视场角（仅限视觉类传感器），单位度也可改变。

“SensorPosXYZ[3]”为传感器安装位置，单位米也可改变。

“SensorAngEular[3]”为传感器安装角度，单位度°也可改变。

然后通过 ue.initUE4MsgRec 开启飞机数据的监听，通过 ue.getCamCoptObj 获取当前 1 号飞机的属性。

# 3、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”。

# 4、文件目录

文件夹/文件名称	说明
GetCamObjDemo.bat	软件在环仿真实验脚本
GetCamObjDemo.py	Python 实验脚本
GetCamObjDemoWithCam.py	Python 实验脚本
Config.json	取图配置文件

# 5、运行环境

序号	软件要求	硬件要求	
		名称	数量(个)

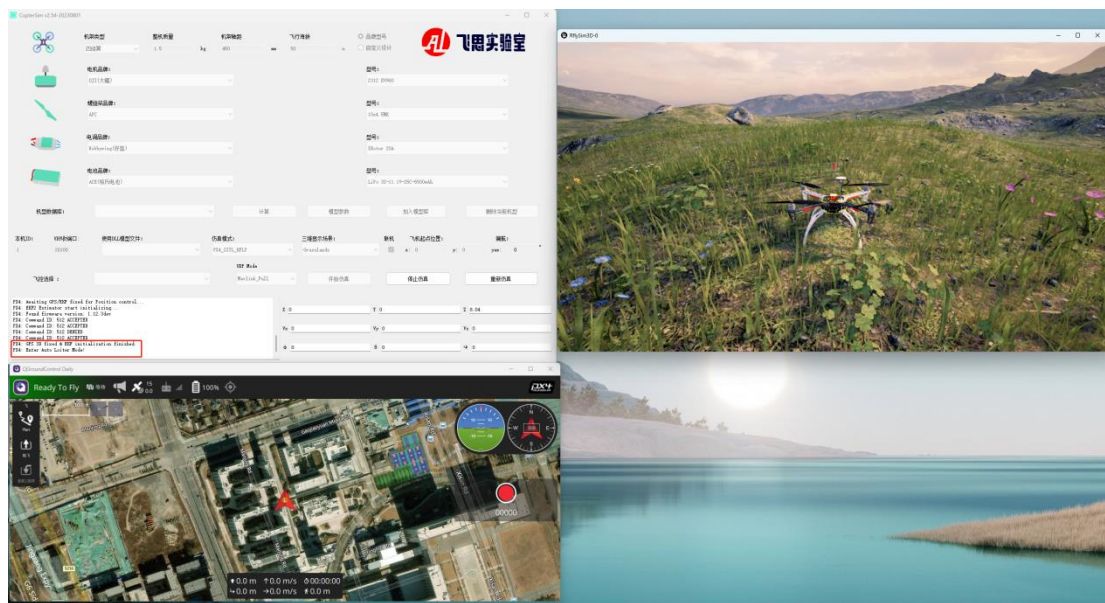
1	Windows 10 及以上版本	笔记本/台式电脑 <sup>①</sup>	1
2	RflySim 平台免费版及以上		
3	Visual Studio Code		

①：推荐配置请见：<https://doc.rflysim.com/1.1InstallMethod.html>

## 6、实验步骤

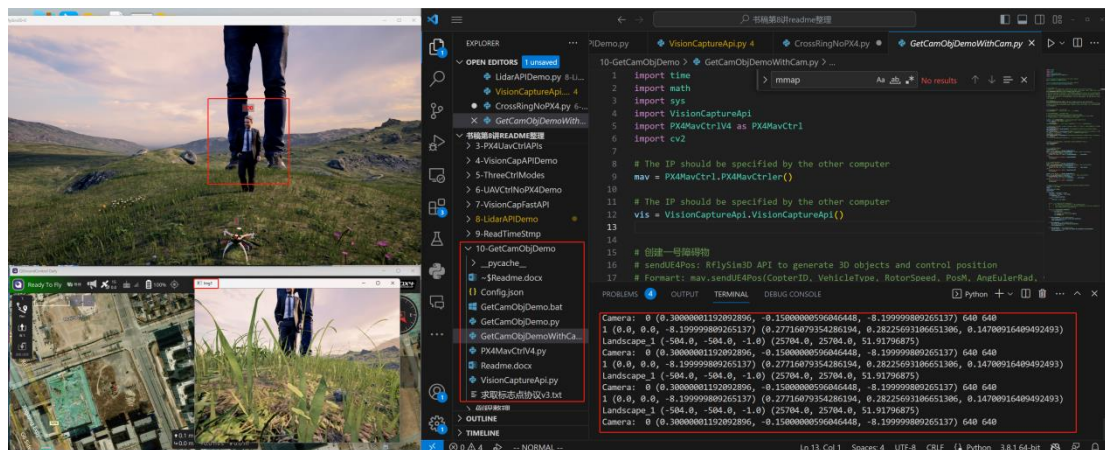
### Step 1:

以管理员方式运行 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:

用 VScode 运行 GetCamObjDemo.py（或 GetCamObjDemoWithCam.py，包含相机位置等信息）文件，可以获取到飞机和物体的信息，以及一个摄像头窗口，创建的几个物体，如下图所示。请查看注释理解 python 实现原理。详细原理见“求取标志点协议 v3.txt”。



### Step 3:

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。



```
C:\WINDOWS\system32\cmd.exe

-----
Start QGroundControl
Kill all CopterSims
Starting PX4 Build
[1/1] Generating ../../logs
killing running instances
starting instance 1 in /mnt/c/PX4PSPFull/Firmware/build/px4_sitl_default/instance_1
PX4 instances start finished
Press any key to exit

按下回车键，快速关闭所有仿真窗口
```

### Step 4:

在下图 VS Code 中，点击“终止终端”，可以彻底退出脚本运行。



## 7、参考文献

[1]. 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. 相机发布的数据结构为 **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

## 8、常见问题

Q1: 无

A1: 无