

1. 实验名称及目的

1.1. 实验名称

无人机跟随圆形案板移动实验

1.2. 实验目的

该例程通过生成一块圆形案板并用按键控制圆形案板移动方向。通过使用平台接口进行图像的获取，并通过视觉处理控制无人机跟随圆形案板移动。

1.3. 关键知识点

本实验主要是实现通过 Python 接口 VisionCaptureApi.py（见 RflySimAPIs\RflySimSDK\vision 目录）获取 RflySim3D 图像并实时更新相机参数（姿态、位置、FOV 等）。然后通过共享内存的方式进行传输图像数据。并且通过调用 python 中的 keyboard 库进行键盘控制圆形案板的位置。键盘控制的程序流程可见 CameraCtrlApi 文件，最后通过图像处理和计算获得无人机控制指令。关键代码解析如下：

1) 视觉接口使用

```
vis = VisionCaptureApi.VisionCaptureApi() # 创建一个视觉传感器实例
vis.jsonLoad() # 加载 Config.json 中的传感器配置文件
isSuss = vis.sendReqToUE4() # 向 RflySim3D 发送取图请求
vis.startImgCap() # 开启取图
vis.hasData[i] # 图片 i 数据是否更新
vis.Img[i] # 图片 i 数据（像素矩阵）
cv2.imshow('Img'+str(i),vis.Img[i]) # 显示图片 i 图像
```

2) 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 的视觉传感器
- "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":[0.3,0,0]: RGB 相机分布。

3) 飞机控制指令

```
mav = PX4MavCtrl.PX4MavCtrler(1) # 创建飞机控制实例
mav.InitMavLoop() # 初始化 Mavlink 监听程序，读取飞机数据
mav.initOffboard() # 进入 Offboard 模式
mav.SendMavArm(True) # 解锁飞控
mav.SendPosNED(0, 0, -10, 0) # 发送 10 米高的位置控制指令
```

4) 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.sendUE4Cmd('r.setres 1280x720w',0) # 发送指令，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，本窗口设置越小，资源需求越少。
ue.sendUE4Cmd('t.MaxFPS 30',0) # 发送指令，设置 UE4 最大刷新频率 30Hz，
同时也是取图频率
ue.sendUE4PosNew(3, vehicleType=809, PosE=[2, 0, -2]) # 创建一个物
体，id 号为 3，类型为 809，位置为[2, 0, -2]
ue.sendUE4ExtAct(3, ActExt=[1, 0.8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]) # 外部动作信息，用于指定飞行器执行的外部动作。这是一个包含 16 个元素的
列表，每个元素表示一个外部动作的值。
```

5) 其余代码说明

```
def MoveCircle(val) # 控制圆板目标的移动
def CalCirle(img: np.array) -> tuple() # 从输入图像中检测出红色区域，并返回该区域
的边界和中心坐标
def CtrlDrone() # 控制无人机的运动，根据视觉系统的反馈调整无人机的位置
ctrl_drone_th = threading.Thread(target=CtrlDrone) # 启动一个新的线程来运行 CtrlD
rone 函数
keyboard.add_hotkey("ctrl+up", MoveCircle, args=("front",)) # 使用键盘快捷键来移动
圆圈的位置
```

2. 实验效果

使用键盘上的 (Up(↑): 表示案板向上移动;

Down(↓): 表示案板向下移动;

Left(←): 表示案板向左右移动;

Right(→): 表示案板向右移动;

Ctrl+Up: 表示案板向飞机前方移动, (远离飞机);

Ctrl+Down:表示案板向飞机后移动, (靠近飞机);

飞机能够跟随运动。

3. 文件目录

例程目录: [\[安装目录\]\RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\c6-Circle-follow\](#)

文件夹/文件名称	说明
circle_follow.bat	一键启动脚本
circle_follow.py	无人机跟随圆形案板移动例程
Python38Run.bat	Python 环境启动脚本
Config.json	视觉传感器配置文件

4. 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 ^①	1
2	RflySim 工具链		
3	VS Code		1

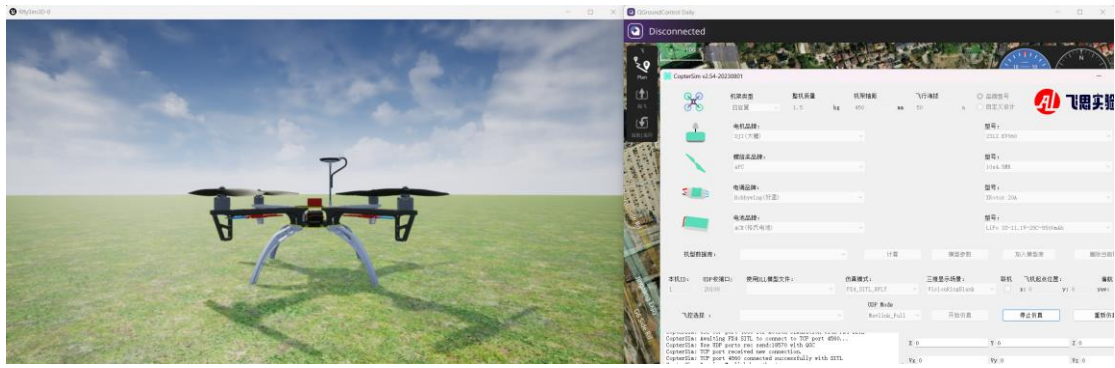
① : 推荐配置请见: <https://rflysim.com>

5. 实验步骤

5.1 必做实验: Windows 控制

Step 1: 开启仿真

双击运行 circle_follow.bat。

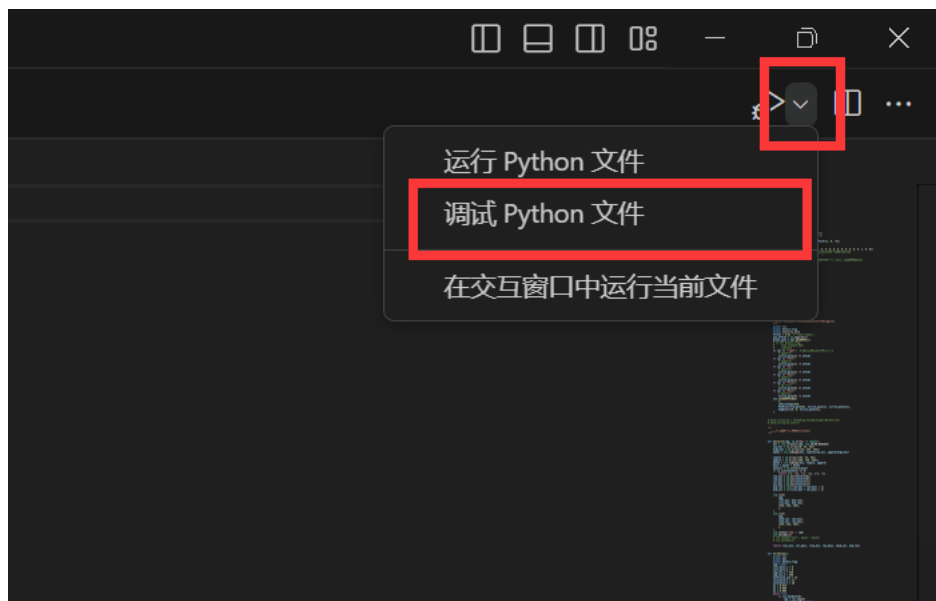


Step 2: 设置搜索路径

运行 PX4PSPRfySimAPIs\RflySimSDK 目录下的 ReLabPath.py 文件。

Step 3: 调试控制程序

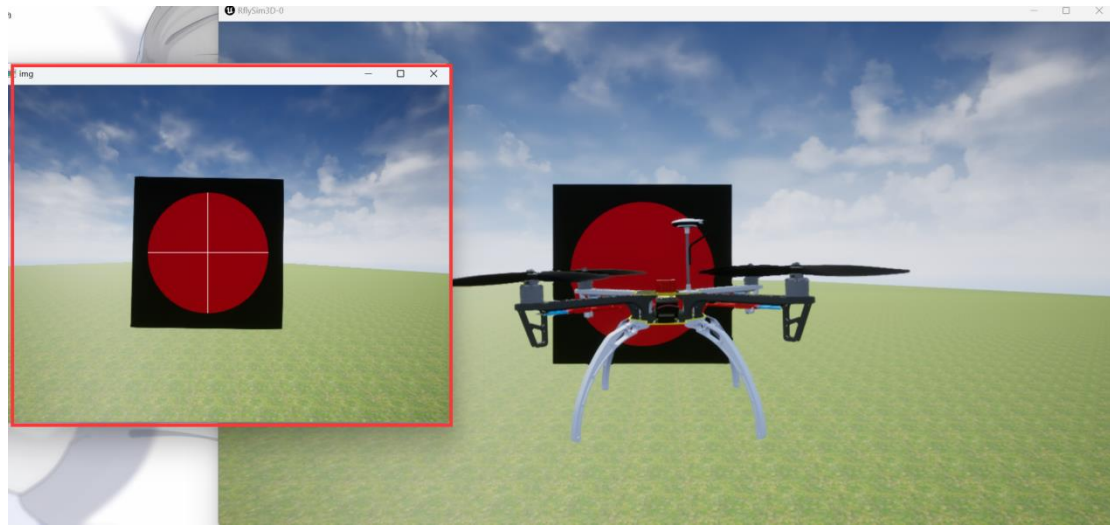
用 Visual Studio Code 打开 circle_follow.py，并点击调试。



Step 4: 运行控制程序

使用键盘上的 Up(↑): 表示案板向上移动;
 Down(↓): 表示案板向下移动;
 Left(←): 表示案板向左右移动;
 Right(→): 表示案板向右移动;
 Ctrl+Up: 表示案板向飞机前方移动, (远离飞机);

Ctrl+Down:表示案板向飞机后移动, (靠近飞机);
无人机跟随移动。



Step 5: 结束仿真

在下图“circle_follow.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。

6. 参考资料

[1]. 无

7. 常见问题

Q1: 无

A1: 无