

1. 实验名称及目的

1.1. 实验名称

无人机跟踪小球实验

1.2. 实验目的

通过平台接口进行图像的获取，然后通过运行“ShootBall3.py”程序。在前方生成一个红色球体，让飞机飞到靠左后方一段距离，并开启视觉跟踪，飞到小球面前停止。

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）。在前方生成一个红色球体，然后通过 opencv 库进行处理，找到红色区域轮廓，并计算红色区域的质心位置和红色区域的面积，并根据图像处理结果将速度命令发送给飞机，从而使得飞机能够飞行到红球前。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 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。

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.SendVelFRD(ctrl[0], ctrl[1], ctrl[2], ctrl[3]) # 设置飞机的速度,
分别是 x、y、z 以及 yaw 角的速度
```

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.sendUE4Pos(100,152,0,[3,0,-2],[0,0,0]) # 创建一个物体, id 号为 100,
类型为 152, 速度为 0, 位置为[3,0,-2], 欧拉角为[0,0,0]
```

5) 其余代码说明

```
def calc_centroid(img) # 用于计算图像中红色区域的质心和面积
def controller(p_i) # 根据检测到的红色球的位置生成控制 Pixhawk 的速度命令
def procssImage() # 处理图像以生成车辆的速度控制命令
def sat(inPwm,thres=1) # 用于限制速度命令的最大值
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
```

2. 实验效果

在前方生成一个红色球体, 让飞机飞到靠左后方一段距离, 并开启视觉跟踪, 飞到小球面前停止。

3. 文件目录

例程目录: [\[安装目录\]\RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e3](#)

ShootBall\

文件夹/文件名称	说明
ShootBall3HITL.bat	硬件在环一键启动脚本
ShootBall3SITL.bat	软件在环一键启动脚本
Config.json	视觉传感器配置文件
ShootBall3.py	无人机跟踪小球例程
Python38Run.bat	Python 实验代码

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		可选

①：推荐配置请见：<https://rflysim.com>

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“ShootBall3SITL.bat”文件开启软件在环仿真系统。也可插入飞控，并运行硬件在环仿真脚本“ShootBall3HITL.bat”，输入串口号来开启 HITL 仿真。



Step 2: 设置搜索路径

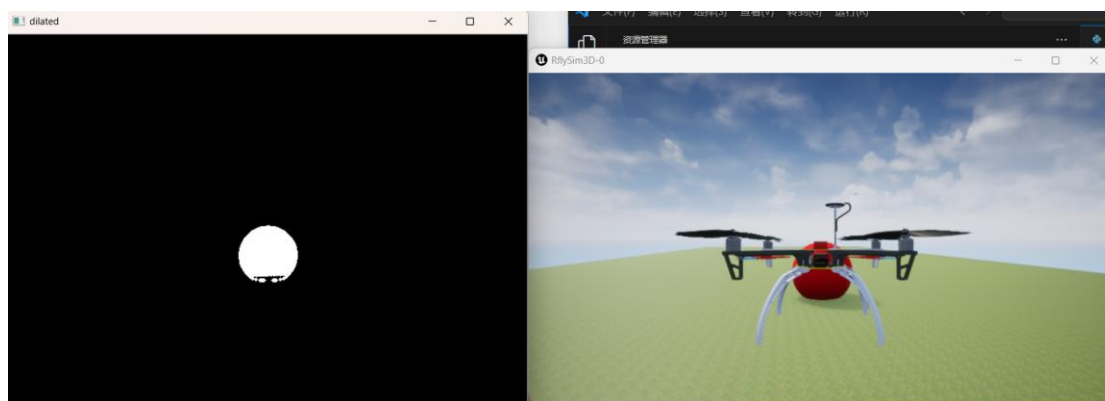
运行 PX4PSRfySimAPIs\RflySimSDK 目录下的 ReLabPath.py 文件。

Step 3: 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 python ShootBall3.py

Step 4: 观察结果

可以看到场景切换到草地场景，生成了一个多旋翼飞机，以及一个红球。飞机飞到靠左后方一段距离，并开启视觉跟踪，飞到小球面前停止。



Step 4: 结束仿真

在下图“ShootBall3SITL.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\3_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 ShootBall3.py 时，可使用 VS Code（或 Pycharm）

arm 等工具) 来打开 ShootBall3.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 ShootBall3.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 X
> RflySimAPIs > 8.RflySimVision > 0.ApiExps > 1-UsageAPI > 0.VisionSensorAPI > 1.Camera
8   ue = UE4CtrlAPI.UE4CtrlAPI()
9
10  #Create a new MAVLink communication instance, UDP sending
11  mav = PX4MavCtrl.PX4MavCtrl(1)
12
13  # The IP should be specified by the other computer
14  vis = VisionCaptureApi.VisionCaptureApi()
15
16  # Send command to UE4 Window 1 to change resolution
17  ue.sendUE4Cmd('r.setres 1280x720w',0) # 设置UE4窗口分辨率, 同时
18  ue.sendUE4Cmd('t.MaxFPS 30',0) # 设置UE4最大刷新频率, 同时也
19  time.sleep(2)
20
21  # VisionCaptureApi 中的配置函数
22  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

[1]. 无

7. 常见问题

Q1: 无

A1: 无