
1. 实验名称及目的

1.1. 实验名称

测距传感器实验

1.2. 实验目的

通过测距传感器配置文件来获取传感器测距。

1.3. 关键知识点

本实验主要是实现通过 Python 接口创建一个激光测距传感器，并实时获取测距数据。。

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）
obj_distance = vis.DistanceSensor.Distance # 获取距离传感器的距离 obj_
distance

cv2.line(img, (center_x - crosshair_length, center_y), (center_x + c
rosshair_length, center_y), (0, 0, 255), crosshair_thickness) # 使用 cv2.
line 绘制水平和垂直的十字准线

cv2.putText(img, distance_text, (center_x - 50, center_y + crosshair
_length + 20), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2) # 使用 cv2.
putText 在图像上绘制距离信息，位置在图像中心下方，颜色为红色 (0, 0, 255)
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 和 "TypeID":5: 传感器类型为 RGB 和测距
- "TargetCopter":1: 相机绑定在 1 号飞机上
- "SendProtocol":[1,0,0,0,0,0,0,0]: 传输模式为 1: UDP 网络传输模式（图片使用 jpeg 压缩，点云直传）

"SensorPosXYZ": [0.3, 0, 0] 和 "SensorPosXYZ": [0, 0, 0]: 相机的位置。

3) 飞机控制指令

```
mav = PX4MavCtrl.PX4MavCtrl(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, -2) # 发送 2 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
```

5) 测距代码说明

Config.json 中, 配置 TypeID 为 5, 这里 1 号传感器 (从 0 开始计数) 对应测距传感器

```
19      {
20          "SeqID": 0,
21          "TypeID": 5,
22          "TargetCopter": 0,
23          "TargetMountType": 4,
24          "DataWidth": 640,
25          "DataHeight": 480,
26          "DataCheckFreq": 30,
```

在 Python 程序中, 通过 vis.hasData[1] 确认已收到测距传感器数据, 并获取数据指针, 赋值给 obj_distance= vis.Img[1]。注: vis.hasData[0] 和 vis.Img[0] 对应 0 号传感器为 RGB 图像。

```

77
78 while True:
79     if vis.hasData[0] and vis.hasData[1]: # 是否成功取图和获取测距传感器
80
81         img = vis.Img[0] # 获取0号传感器，图像数据指针，格式为opencv图像格式
82         obj_distance = vis.Img[1] # 获取1号传感器，距离数据指针，格式见Visic
83

```

通过类成员引用，可以打印数据

```

94         cv2.line(img, (center_x, center_y - crosshair_length), (cent
95
96         # 绘制距离信息
97         distance_text = f"Distance: {obj_distance.Distance} m"
98         cv2.putText(img, distance_text, (center_x - 50, center_y + c
99

```

注：测距传感器数据结构体定义如下：

```

class DistanceSensor:
    ## @brief DistanceSensor 的构造函数
    # @param 初始化类属性
    def __init__(self):
        ## @var DistanceSensor.TimeStamp
        # @brief 这是当前消息的时间戳，初始化为 0
        self.TimeStamp = 0
        ## @var DistanceSensor.Distance
        # @brief 这是距离传感器测量到的距离，初始化为 0
        self.Distance = 0
        ## @var DistanceSensor.CopterID
        # @brief 用于标识直升机的 ID，初始化为 0
        self.CopterID = 0
        ## @var DistanceSensor.RayStart
        # @brief 这是射线起点的坐标，初始化为[0,0,0]
        self.RayStart = [0,0,0]
        ## @var DistanceSensor.AngEular
        # @brief 这是传感器的欧拉角（Euler Angles），初始化为[0,0,0]
        self.AngEular = [0,0,0]
        ## @var DistanceSensor.ImpactPoint
        # @brief 这是碰撞点的坐标，初始化为[0,0,0]
        self.ImpactPoint = [0,0,0]
        ## @var DistanceSensor.BoxOri
        # @brief 这是盒子的原点或参考点，初始化为[0,0,0]
        self.BoxOri = [0,0,0]

```

2. 实验效果

本实验通过测距传感器配置文件来获取传感器测距的距离。

3. 文件目录

例程目录：[\[安装目录\]\RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\0.VisionSensorAPI\12.RangingImageDemo\](#)

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

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

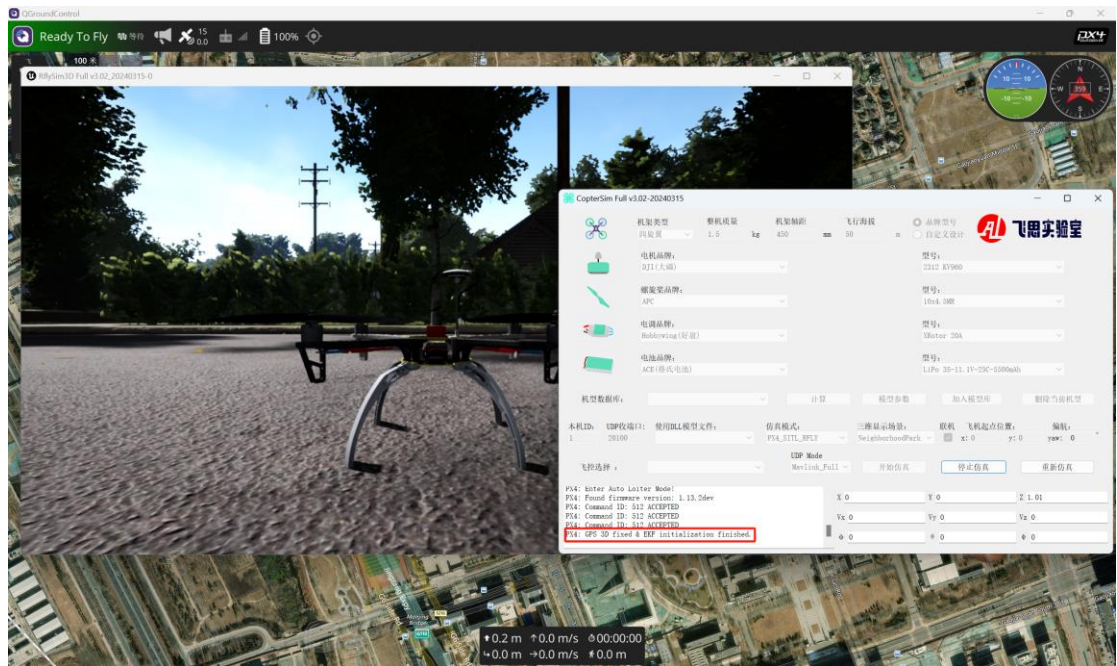
①：推荐配置请见：<https://rflysim.com>

5. 实验步骤

5.1 必做实验：Windows 取图控制

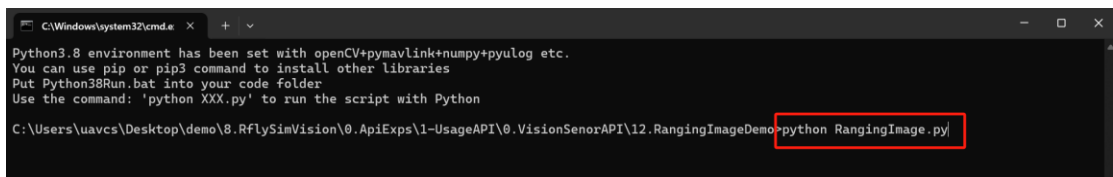
Step 1：开启仿真

双击运行 RangingImage.bat 开启一个软件在环仿真。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 RangingImage.py 文件，输入 python RangingImage.py



Step3: 观察结果

可以看到一个测距的图像窗口。



Step 4: 结束仿真

在下图“RangingImage.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 RangingImage.py 时，可使用 VS Code（或 Pycharm 等工具）来打开 RangingImage.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 RangingImage.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.CameraAPI
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，
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