



---

# 智能无人集群系统开发与实践

## 基于RflySim平台的全栈开发案例

### 第8讲 视觉感知与避障决策



# 大纲

1. 总体介绍

本讲的例程源码路径:

RflySimAPIs\8.RflySimVision

2. 基础接口使用

3. 视觉控制例子

本讲PPT公益课视频地址为:

B站: <https://www.bilibili.com/video/BV1t3411K7Nh>

4. 视觉AI进阶

5. 分布式视觉仿真



手机扫码观看

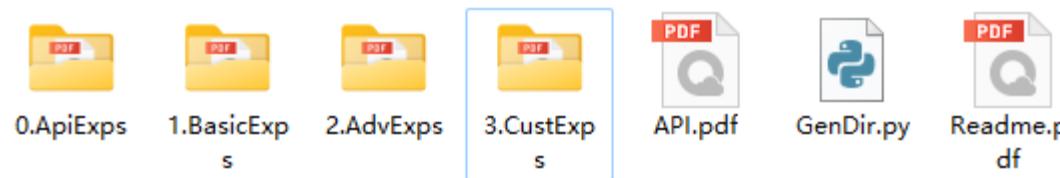


## 1. 总体介绍

### 1.1 视觉例程文件夹

- 目前平台的视觉控制例程在目录 **RflySimAPIs\8.RflySimVision** 中。
- 如右图所示，包含了四个文件夹，分别是：
  - **0.ApiExps**: 免费版例程
  - **1.BasicExps**: 免费版例程
  - **2.AdvExps**: 个人版包含例程
  - **3.CustExps**: 完整版包含例程
- **API.pdf**: 第八章各类接口的介绍以及部分基础知识介绍
- **Readme.pdf**: 各例程介绍目录

注：推荐使用平台自带的Python38环境来运行例程，如果使用其他的Python环境，请确保安装以下组件：  
**pip3 install pymavlink pyserial opencv**





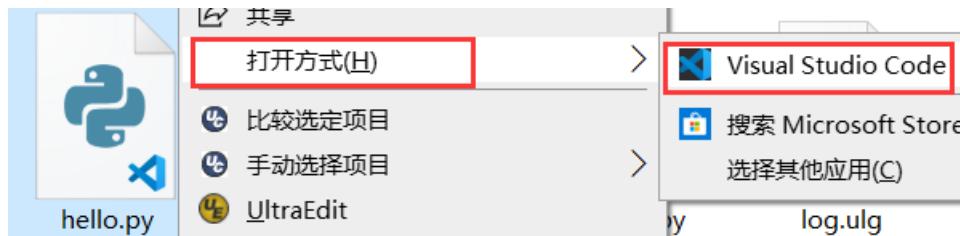
# 1.总体介绍

## 1.2 Python程序运行

- 在“RflySimAPIs\Python38Scripts”文件夹内，新建一个txt文件，并重命名为hello.py
- 右键该文件，用VS Code打开，在其中敲入如下代码

```
print("Hello, RflySim!")
```

- 在VS Code中运行，如右图查看结果。



- 同理，双击Python38Env桌面快捷方式或“RflySimAPIs\Python38Env.bat”，如下图查看运行结果。

1. 输入python脚本

2. 点击运行

3. 弹出终端

4. 查看结果

Python 3.8.1 64-bit ⚡ 0 △ 0 24 bytes 行 1, 列 25 空格: 4 UTF-8 CRLF Python ⚡

```
C:\WINDOWS\system32\cmd.exe
Python3.8 environment has been set with openCV+pymavlink+nu
You can use pip or pip3 command to install other libraries
Put your python scripts 'XXX.py' into the folder 'C:\PX4PSP
Use the command: 'python XXX.py' to run the script with Pyt
For example, try entering 'python ImgCVShow.py' below to us
You can also use pyulog (see https://github.com/PX4/pyulog)
For example, try entering 'ulog2csv log.ulg' to convert ulg
C:\PX4PSP\Python38Env\python hello.py
Hello, RflySim!
```

C:\PX4PSP\Python38Env>



# 1. 总体介绍

## 1.3 Python语法学习

- 访问：<https://runoob.com/python3> 了解学习Python3的基础知识与编程方法（与MATLAB语言相似）上手容易。
- 注：Python2目前已经停止更新，推荐大家直接学习Python3进行开发。
- Python3默认使用UTF-8编码，因此原生支持中文字符。
- Python采用缩进的方式来区分代码层级，因此编写代码是要注意缩进。（可在VS Code中安装Guides, indent-rainbow等插件来辅助）

RUNOOB.COM

首页 HTML CSS JAVASCRIPT JQUERY BOOTSTRAP PYTHON3 PYTHON2 JAVA C C++ C# SQL

Python 3 教程

Python3 教程

Python3 简介  
Python3 环境搭建  
Python3 基础语法  
Python3 基本数据类型  
Python3 解释器  
Python3 注释  
Python3 运算符  
Python3 数字(Number)  
Python3 字串串  
Python3 列表  
Python3 元组  
Python3 字典  
Python3 集合  
Python3 编程第一步  
Python3 条件控制  
Python3 循环语句  
Python3 迭代器与生成器

**Python 3 教程**

 Python powered  
print("Hello, world!")

Python 的 3.0 版本，常被称为 Python 3000，或简称 Py3k。相对于 Python 的早  
级。为了不带入过多的累赘，Python 3.0 在设计的时候没有考虑向下兼容。  
Python 介绍及安装教程我们在[Python 2.X 版本的教程](#)中已有介绍，这里就不再赘述  
你也可以点击[Python2.x与3.x版本区别](#)来查看两者的不同。

本教程主要针对 Python 3.x 版本的学习，如果你使用的是 Python 2.x 版本请移步至[Python 2.X 版本的教程](#)。  
官方宣布，2020 年 1 月 1 日，停止 Python 2 的更新。

**查看 Python 版本**

我们可以在命令窗口(Windows 使用 win+R 调出 cmd 运行框)使用以下命令查看我们使用的 Python 版本：

```
python -V
```

以上命令执行结果如下：

```
Python 3.3.2
```

你也可以进入 Python 的交互式编程模式，查看版本：



# 1. 总体介绍

## 1.4 OpenCV学习

- OpenCV是一个开源的跨平台计算机视觉和机器学习软件库，应用广泛。
- 官方教程网址：
- <https://docs.opencv.org/4.0.0/>
- 推荐中文翻译文档网址：
- <https://github.com/makelove/OpenCV-Python-Tutorial>
- 或直接打开“RflySimAPIs\PythonVisionAPI\OpenCV-Python-Tutorial-zh.pdf”来学习。

The screenshot shows the official OpenCV documentation website for version 4.0.0. The top navigation bar includes the OpenCV logo, a dropdown menu set to '4.0.0', and links for Main Page, Related Pages, Modules, Namespaces, Classes, Files, and Examples. Below the navigation is a search bar. The main content area is titled 'OpenCV modules' and lists various modules with their descriptions:

- Introduction
- OpenCV Tutorials
- OpenCV-Python Tutorials
- OpenCV.js Tutorials
- Tutorials for contrib modules
- Frequently Asked Questions
- Bibliography
- Main modules:
  - core. Core functionality
  - imgproc. Image Processing
  - imgcodecs. Image file reading and writing
  - videoio. Video I/O
  - highgui. High-level GUI
  - video. Video Analysis
  - calib3d. Camera Calibration and 3D Reconstruction
  - features2d. 2D Features Framework
  - objdetect. Object Detection
  - dnn. Deep Neural Network module
  - ml. Machine Learning



## 1. 总体介绍

### 1.5 MAVLink通信协议

- Pymavlink是MAVLink通信协议的Python版本，目前已经预装在平台Python环境中，通过它可以方便的通过串口、UDP、TCP等方式与Pixhawk真机通信，进行顶层控制。
- 官方文档网址如下：  
[https://mavlink.io/en/mavgen\\_python](https://mavlink.io/en/mavgen_python)
- 请自行学习其详细使用方法
- 注：Pymavlink只是一个方便使用的库函数，如果有更高的自定义需求，需要学习MAVLink协议的使用方法。
- MAVLink消息协议可阅读如下网址  
<https://mavlink.io/en/messages/common.html>

## Using Pymavlink Libraries (mavgen)

Pymavlink is a *low level* and *general purpose* MAVLink message processing library, written for many types of MAVLink systems, including a GCS (MAVProxy), Developer APIs (DroneKit-Python) and more.

The library can be used with Python 2.7+ (recommended) or Python 3.5+ and supports both C and Python interfaces.

This topic explains how to get and use the *Pymavlink* MAVLink Python libraries (generated from the MAVLink protocol definitions).



Pymavlink is developed in its own project, which includes the command line interface, unit tests, and other useful tools and utilities. MAVLink includes the [Pymavlink repository](#). The [Protocol documentation](#) explains how to work with *pymavlink* *using the MAVLink protocol definitions*.



If you're writing a MAVLink application to communicate with an autopilot you may want to consider using [DroneKit-Python](#). These implement a number of [MAVLink microservices](#).

### Getting Libraries

If you need a [standard dialect](#) then you can install these (for both MAVLink 1 and 2) with:



# 大纲

---

1. 总体介绍
  2. 基础接口使用
  3. 视觉控制例子
  4. 视觉AI进阶
  5. 分布式视觉仿真
-



## 2.基础接口使用

### 2.1 无人机控制接口—PX4MavCtrlV4.py

- 打开PX4PSP\RflySimSDK-master\rflysim sdk\ctrl目录下的PX4MavCtrlV4.py文件，可以看到右图所示的接口文件。
- 本接口定义了一个类PX4MavCtrler用于实现MAVLink消息的收发、RflySim3D场景控制、Pixhawk/PX4的Offboard控制等接口。
- 本接口可以发送消息给CopterSim/PX4进行飞机的控制，也可以发给RflySim3D进行场景控制或请求取图。
- 本接口文件通过UDP与RflySim3D通信，通过UDP-CopterSim-MAVLink-PX4控制飞机，或直接通过“串口-MAVLink-PX4”来控制飞机。

```
import socket
import threading
import time
from pymavlink import mavutil
from pymavlink.dialects.v20 import common as mavlink2
import struct
import math
import sys
import copy
import os
import cv2
import numpy as np
import EarthModel
import UE4CtrlAPI
ue = UE4CtrlAPI.UE4CtrlAPI()
# PX4 MAVLink listen and control API and RflySim3D control API
class PX4MavCtrler:

    """创建一个通信实例
    ID: 如果ID<10000则表示飞机的CopterID号。如果ID>10000, 例如20100这种，则表示通信端口号port
    ip: 数据向外发送的IP地址。默认是发往本机的127.0.0.1的IP，在分布式仿真时，也可以指定192.168.0.1
    Com: 与Pixhawk的连接模式。
        Com='udp': 表示使用默认的udp模式接收数据，这种模式下，是接收coptersim转发的PX4的MAVLINK消息，使用port+1端口收和port端口发（例如，一号飞机是20101端口收，20100端口发，与CopterID无关）
        Com='COM3': (Windows下) 或 Com='/dev/ttyUSB0' (Linux系统, 也可能是ttyS0, ttyAMA0等)
        Com='Direct': 表示UDP直连模式(对应旧版接口的真机模式)。这种模式下使用使用同一端口收发
        注意：COM模式和DIRECT模式下，ID只表示飞机的ID号，而不表示端口号
        Com='redis': 使用redis模式通信，服务器地址为ip，服务器端口为port
    port: UDP模式下默认情况下设为0，会自动根据IP填充。接平台规则，port=28100+CopterID*2-2。如
    COM模式下，Port默认表示波特率self.baud+port；如果port=0，则会设置self.baud+57600
    Direct模式下，Port默认表示收发端口号，使用相同端口
    redis模式下，Port对应服务端口号self.redisPort - port。如果port=0，则self.redisPort
    端口示例：
    UDP模式
    PX4MavCtrler(1) # 默认IP
    PX4MavCtrler(1,'192.168.31.24') # 指定IP，用于远程控制

    串口模式
    PX4MavCtrler(1,'127.0.0.1','com1',57600) # 指定IP，用于远程控制

    ...
    ...

    注意：下面这里是旧版接口，便于用户参考旧规则进行接口升级
    For hardware connection PX4MavCtrler('combaud','IP:CopterID') format
    Windows use format PX4MavCtrler('COM1') or PX4MavCtrler('COM3:115200') for Pixhawk USB
    Windows use format 'COM4:57600' for Pixhawk serial port connection
    Linux use format PX4MavCtrler('/dev/ttyUSB0') or PX4MavCtrler('/dev/ttyUSB0:115200') for
    PX4MavCtrler('COM3:115200:2'); the second input is the CopterID, in this case CopterID

    For real flight
    PX4MavCtrler(port,'IP:CopterID:Flag'), Flag set to 1 to enable real flight com mode
    for example PX4MavCtrler(15551,'192.168.1.123:1:1') to set IP=192.168.1.123, port=15551
    ...
```



## 2.基础接口使用

### 2.1 无人机控制接口—PX4MavCtrlV4内部原理

class PX4\_CUSTOM\_MAIN\_MODE: #PX4主模块枚举变量，用于设置模式

class PX4\_CUSTOM\_SUB\_MODE\_AUTO: #PX4子模块枚举变量

class PX4MavCtrler: # RflySim的主要通信接口类，可采用UDP或串口连接方式

def InitMavLoop: #启用MAVLINK接收线程，随时接收并更新MAVLINK消息

def sat: #一个饱和函数，用于控制变量的限幅

def SendMavCmdLong: #发送MAVLINK消息的COMMAND\_LONG消息

def sendMavOffboardCmd : # 发送Offboard命令给飞控，使其进入Offboard模式

def sendMavOffboardAPI : # 更新Offboard消息的数据（该数据会以一定频率发送）

def SendVelNED : # 发送地球坐标系速度指令

```
self.uavAngEular = [0, 0, 0]
self.uavAngRate = [0, 0, 0]
self.uavPosNED = [0, 0, 0]
self.uavVelNED = [0, 0, 0]
```

Pixhawk实时  
状态数据



## 2.基础接口使用

### 2.1 无人机控制接口—PX4MavCtrlV4内部原理

**def SendVelFRD:** #发送机体速度

**def SendPosNED:** #发送NED位置，让飞机飞到指定位置（相对解锁点）

**def initOffboard:** # 初始化Offboard模式

**def endOffboard:** # 结束Offboard模式

**def sendMavSetParam:** # 发送MAVLink消息改变Pixhawk参数

**def SendHILCtrlMsg :** #发送rfly\_msg消息到飞控（见第3讲4.3节内容）

**def SendMavArm :** #发送解锁命令

**def SendRcOverride :** #发送并模拟遥控器信号

**def sendMavManualCtrl :** #发送并模拟归一化遥控器信号

**def SendSetMode :** #发送并设置Pixhawk模式

**def stopRun :** #停止运行MAVLink数据接收线程

**def getMavMsg :** #更新MAVLink接收的数据

每个函数的详细定义，可以自行阅读PX4MavCtrlV4.py内部源码实现。



## 2.基础接口使用

### 2.1 无人机控制接口—基本使用例子

"PX4PSP\RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\

1.PX4MavCtrlAPITest\PX4MavCtrlAPITest.py"

是一个接口使用的Python例子，具体代码解析如下：

#新建一个MAVLink通信实例，CopterSim接口是20100

ma = PX4MavCtrl.PX4MavCtrler(1)

# RflySim3D样式调整API: sendUE4Cmd函数属于UE4CtrlAPI接口

# 样式ue.sendUE4Cmd(cmd,windowID=-1)，其中cmd应该输入字符串

# 向RflySim3D发送窗口调整命令，cmd为具体命令字符串，windowID为接收的窗口号(假设同时打开了多个RflySim3D窗口)，-1表示发送给所有窗口

# RflyChangeMapbyName命令表示切换地图，后面跟的字符串为地图名字，这里将所有打开窗口切换为草地地图Grasslands

ue.sendUE4Cmd(b'RflyChangeMapbyName Grasslands ')

```
import time  
import math  
import sys
```

导入库

```
import PX4MavCtrlV4 as PX4MavCtrl  
import UE4CtrlAPI  
ue = UE4CtrlAPI.UE4CtrlAPI()
```

创建UE接口实例

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

创建控制接口实例



## 2.基础接口使用

### 2.1无人机控制接口—基本使用例子

```
# RflySim3D生成三维物体并控制位姿姿态API: sendUE4Pos函数，隶属UE4CtrlAPI接口
# 样式ue.sendUE4Pos(CopterID, VehicleType, RotorSpeed, PosM, AngEulerRad, windowsID)
ue.sendUE4Pos(100,30,0,[2.5,0,-8.086],[0,0,math.pi])
# 向RflySim3D发送并生成一个三维物体，其中：物体ID为CopterID=100；
# 飞机类型VehicleType=30（人物）；旋翼转速RotorSpeed=0RPM；位置坐标PosM=[2.5,0,-8.086]m
# 飞机姿态角AngEulerRad=[0,0,math.pi]rad（旋转180度面对飞机），接收窗口号默认
windowsID=-1（发给所有打开的RflySim3D程序）
# VehicleType: 3四旋翼，5/6六旋翼，30人物，40标定棋盘格，50/51车，60球状发光灯，100
飞翼固定翼，150/152环形方形靶标
# RflyChange3DModel命令后跟飞机ID + 期望样式，其中期望12的样式是行走的人
ue.sendUE4Cmd(b'RflyChange3DModel 100 12')
#发送消息，使所有场景中的CopterID=100（刚创建的人物）变成行走的人的样式
```



## 2.基础接口使用

---

### 2.1 无人机控制接口—基本使用例子

```
# 命令RflyChangeViewKeyCmd表示模拟RflySim3D的快捷键操作，B 1快捷键表示将焦点切换到CopterID=1的对象
```

```
# 这里设定向0号窗口发送，其他窗口不发送
```

```
ue.sendUE4Cmd('RflyChangeViewKeyCmd B 1',0)
```

```
# V 1快捷键表示将视角切换到第1号机载视角
```

```
ue.sendUE4Cmd('RflyChangeViewKeyCmd V 1',0)
```

```
# RflyCameraPosAng x y z roll pitch yaw 设置相机相对机体中心的位置方向，默认给0
```

```
# 这里设置前置相机的位置为[0.1 0 0]
```

```
ue.sendUE4Cmd('RflyCameraPosAng 0.1 0 0',0)
```

```
# r.setres 720x405w 是UE4内置的命令，表示切换分辨率到720x405
```

```
ue.sendUE4Cmd('r.setres 720x405w',0)
```



## 2.基础接口使用

### 2.1 无人机控制接口—基本使用例子

```
# 向1号窗口发送快捷键命令，将焦点切换到飞机1  
ue.sendUE4Cmd('RflyChangeViewKeyCmd B 1',1)
```

```
# 向1号窗口发送快捷键控制命令，N 1 快捷键表示将视角切换到地面固定视角1  
ue.sendUE4Cmd('RflyChangeViewKeyCmd N 1',1)
```

```
# 设置当前相机视场角为90度（RflySim3D中默认为90度），视场角范围是0到180度  
ue.sendUE4Cmd('RflyCameraFovDegrees 90',1)
```

```
# 这里设置当前相机的位置为[-2 0 -9.7]  
ue.sendUE4Cmd('RflyCameraPosAng -2 0 -9.7',1)
```

```
#开启MAVLink监听CopterSim数据，并实时更新，数据见右下图
```

```
mav.InitMavLoop()
```

```
#显示从CopterSim收到的位置信息  
print(mav.uavPosNED)
```

```
self.uavAngEular = [0, 0, 0]  
self.uavAngRate = [0, 0, 0]  
self.uavPosNED = [0, 0, 0]  
self.uavVelNED = [0, 0, 0]
```

Pixhawk实时  
状态数据



## 2.基础接口使用

### 2.1 无人机控制接口—基本使用例子

#开启 Offboard 模式

**mav.initOffboard()**

#发送期望位置信号，飞到目标点 0,0, -1.7 位置，偏航角为 0

**mav.SendPosNED(0, 0, -1.7, 0)**

#发送解锁命令

**mav.SendMavArm(True)**

#发送期望速度信号，0.2m/s向下降落，z轴向下为正

**mav.SendVelNED(0, 0, 0.2, 0)**

#退出 Offboard 控制模式

**mav.endOffboard()**

#退出 MAVLink 数据接收模式

**mav.stopRun()**



## 2.基础接口使用

### 2.1 无人机控制接口—实验1：接口调试实验

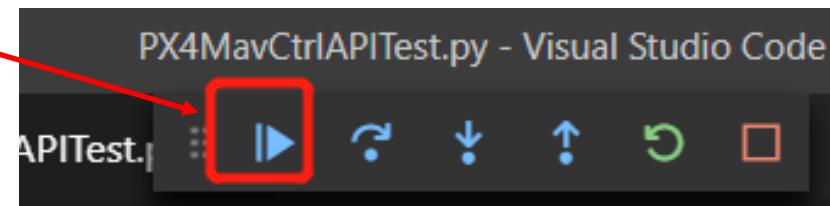
- 在Windows资源管理器中定位到“PX4PSP\RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\1.PX4MavCtrlAPITest”文件夹。
- 双击“PX4MavCtrlAPITest.bat”脚本来打开一个飞机的PX4 SITL仿真系统。
- 用VS Code打开“PX4MavCtrlAPITest.py”文件，如右图在每条关键语句前面点上断点（红点），按下图所示开启调试模式，点右下图箭头按钮，依次执行语句。



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

# sendUE4Cmd: RflySim3D API to modify scene display
# Format: ue.sendUE4Cmd(cmd,windowID=-1), where cmd
# Argument: RflyChangeMapByName command means to s
ue.sendUE4Cmd('RflyChangeMapByName Grasslands')
time.sleep(2)

# sendUE4Pos: RflySim3D API to generate 3D objects
# Formart: ue.sendUE4Pos(CopterID, VehicleType, Rot
ue.sendUE4Pos(100,30,0,[2.5,0,-8.086],[0,0,math.pi]
# Send and generate a 3D object to RflySim3D, where
# Vehicle type VehicleType=30 (a man); RotorSpeed=
```

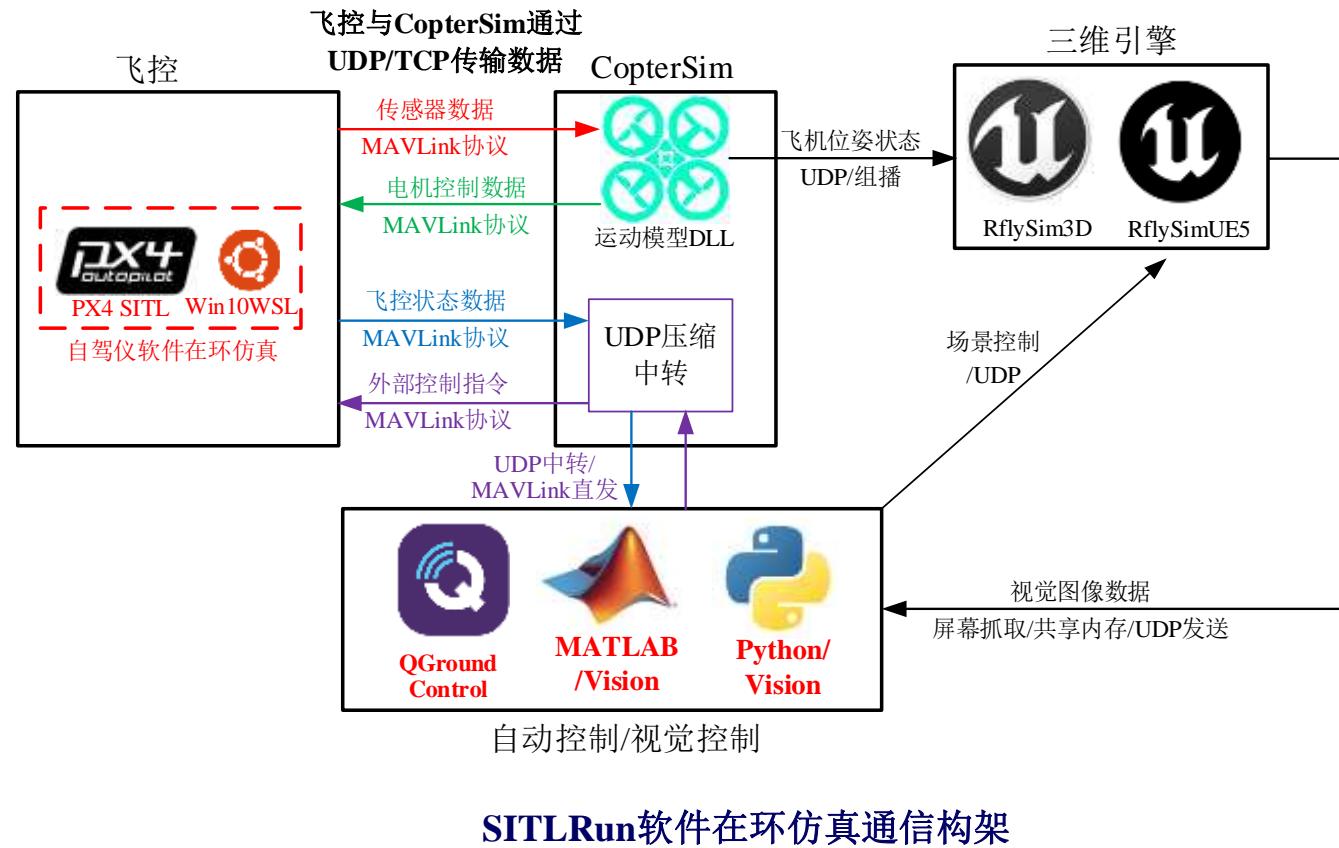




## 2.基础接口使用

### 2.1 无人机控制接口—实验1： SITLRun通信框架

- 在SITL软件在环仿真过程中，**PX4飞控完整运行于Win10WSL虚拟机中**，使用**px4\_sitl固件**。
- **PX4与CopterSim直接通过网络MAVLink协议通信**，然后**CopterSim通过20100/20101端口实现外部Python消息的收发**。
- **Python程序通过UDP直接与RflySim3D程序通信**，并通过**共享内存/UDP发送等方式获取获取视觉图像**。

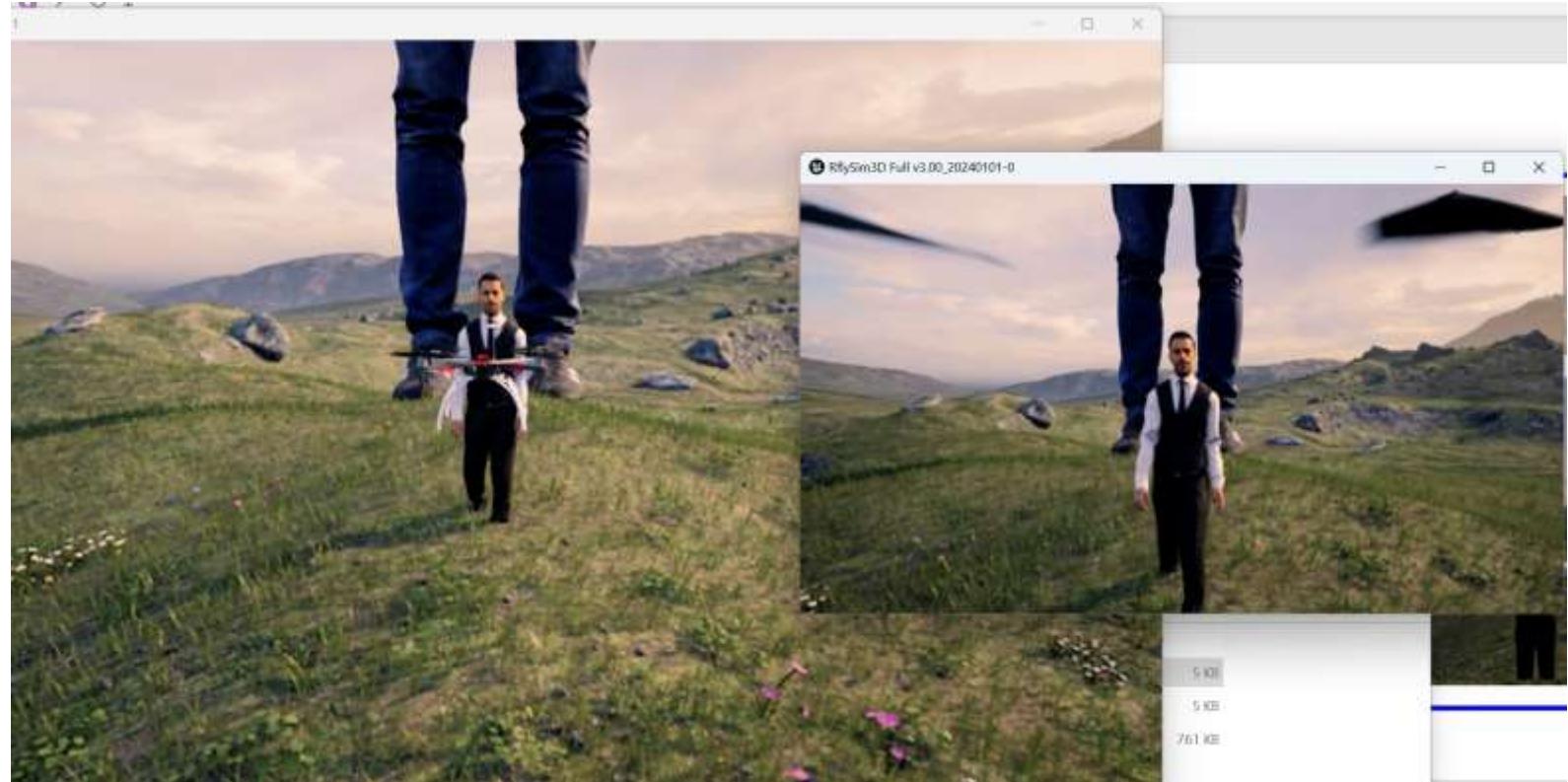




## 2.基础接口使用

### 2.1 无人机控制接口—实验1： 实验效果

- 本例程的现象是，  
`python`程序发送一系列  
指令，在RflySim3D程  
序中新建了一个走动的  
人的目标，设置了视角  
形式、尺寸、位置，向  
仿真的无人机发送控制  
指令使其起飞与降落。
- 如右图所示，本例子会  
打开两个RflySim3D窗  
口，一个是前置摄像头，  
一个是上帝视角观测。





## 2.基础接口使用

### 2.1 无人机控制接口—实验1：结束仿真

- 在下图所示“PX4MavCtrlAPITest.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。
- 如右下图，在VS Code中，点击“终止终端”，可以彻底退出脚本运行。

```
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





## 2.基础接口使用

### 2.1 无人机控制接口—实验2：数传连接Pixhawk硬件在环仿真

- 用MicroUSB线连接电脑和Pixhawk飞控，然后在“RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\2.PX4ComAPITest”中双击“PX4ComAPITest.bat”并输入飞控串口号，开启一个飞机的硬件在环仿真。
- 用数传或TTL串口线连接Pixhawk的TELEM1和电脑，并记录此时的串口号，例如COM14
- 用VS Code打开“PX4ComAPITest.py”，将下面代码的COM14修改为你自己的数传串口号：
- ```

```
mav = PX4MavCtrl.PX4MavCtrl(1,'127.0.0.1','COM9',57600)
```
- 在VS Code中运行PX4ComAPITest.py程序，可以观察到CopterSim中收到解锁的消息，同时Python能够输出uavPosGPS的全球定位数据。
- ```

```
print(mav.uavPosGPS)
```



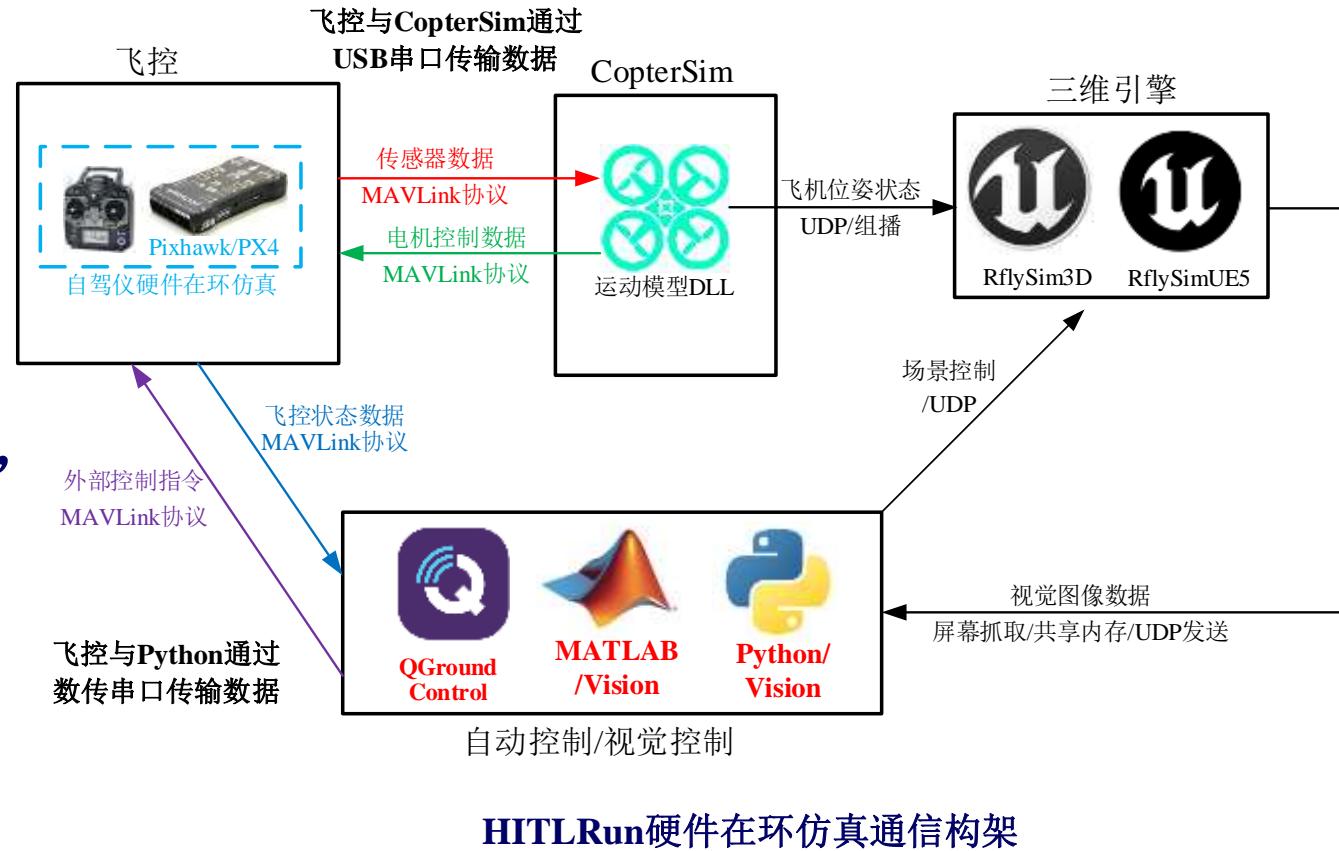
注：如果是Linux系统，串口号的格式为/dev/ttyUSB0或者/dev/ttyAMA0  
冒号的是波特率，PX4 的TELEM1默认波特率是57600



## 2.基础接口使用

### 2.1 无人机控制接口—实验2： HITL+数传通信框架

- 在HITL硬件在环仿真中，PX4飞控算法运行与Pixhawk硬件中，使用px4\_fmu-v5固件。
- PX4与CopterSim直接通过USB串口的MAVLink协议通信，然后CopterSim通过数传串口实现与外部Python程序的消息收发。
- Python程序通过UDP直接与RflySim3D程序通信，并通过共享内存/UDP发送等方式获取获取视觉图像。

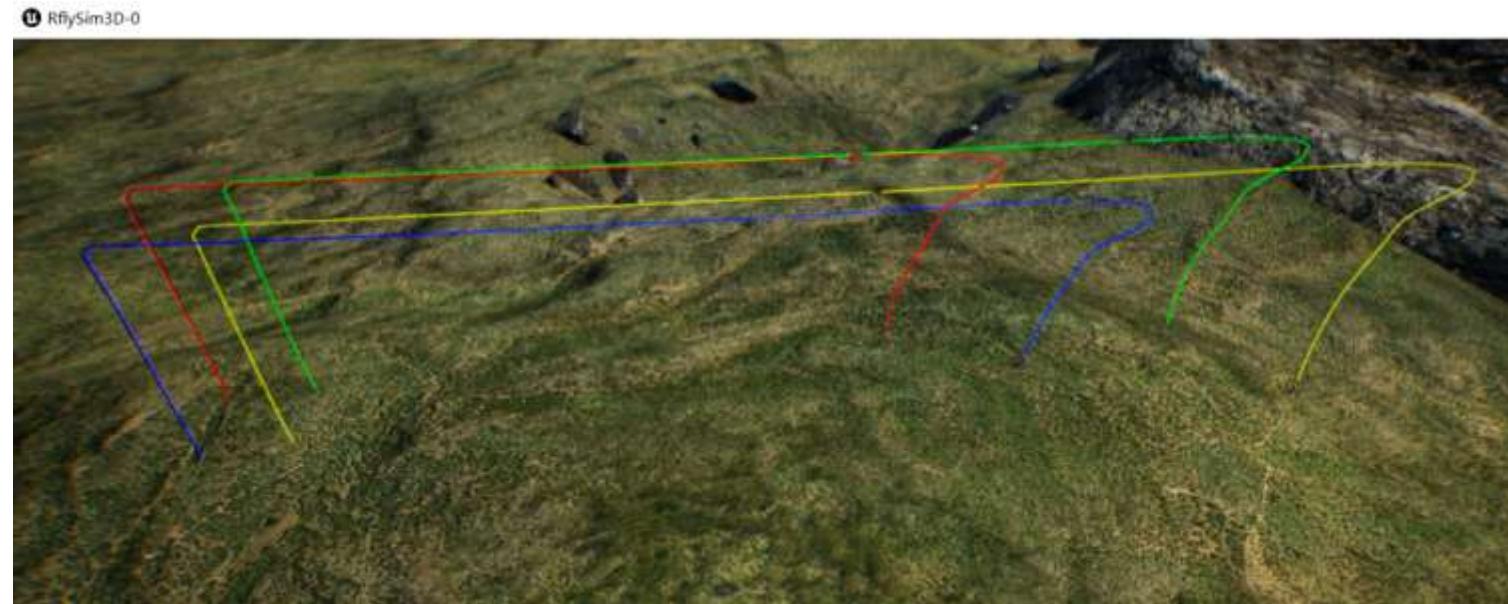




## 2.基础接口使用

### 2.1 无人机控制接口—实验3：多机SITL控制实验

- 在“RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\5.PX4MultiUavTest”文件夹内，双击“PX4MultiUavTest.bat”可以开启四个飞机的SITL仿真闭环。
- 用VS Code打开“PX4MultiUavTest.py”，可以看到代码中新建了四个PX4MavController实例，分别连接端口20100 / 20102/20104/20106对应1到4号飞机。
- 然后，调用了UE4的按键模拟指令，模拟了S键（显示飞机标号）和T键（显示飞机轨迹）
- 最后，依次控制飞机解锁，起飞，前飞，再下降。
- 实验效果如右图



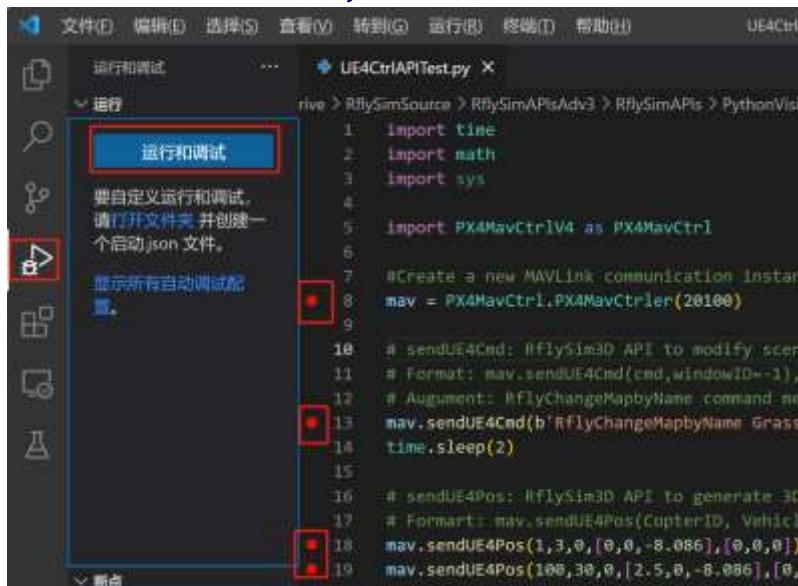


## 2.基础接口使用

注：要让生成物体贴合地面，可以通过UEMapServe类来解算地形高度，或使用sendUE4PosScale2Ground接口时创建的物体自动贴合地面，详见例程源码。

### 2.2 UE4场景控制接口—实验1：场景配置接口（导入障碍物等）

- “RflySimAPIs\3.RflySim3DUE\2.AdvExps\c0\_AdvApiExps\c1\_UEMapCtrl\12.DamageModel”文件夹，双击“UE4CtrlAPITest.bat”可以打开两个RflySim3D窗口。
- 下面要通过Python接口分别控制两窗口，导入障碍物（目标），并配置窗口显示。
- 用VS Code打开“UE4CtrlAPITest.py”，在关键语句设置断点，然后进入调试运行模式，逐句执行语句，查看执行效果。



```
文件(D) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H) UE4CtrlA
运行和调试
运行
...
UE4CtrlAPITest.py X
文件 > RflySimSource > RflySimAPIsAdv3 > RflySimAPIs > PythonVisio
1 import time
2 import math
3 import sys
4
5 import PX4MavCtrlV4 as PX4MavCtrl
6
7 #Create a new MAVLINK communication instance
8 mav = PX4MavCtrl.PX4MavCtrl(20100)
9
10 # sendUE4Cmd: RflySim3D API to modify scene
11 # Format: mav.sendUE4Cmd(cmd,windowID=-1),
12 # Argument: RflyChangeMapByName.command name
13 mav.sendUE4Cmd(b'RflyChangeMapByName:Grass1')
14 time.sleep(2)
15
16 # sendUE4Pos: RflySim3D API to generate 3D
17 # Format: mav.sendUE4Pos(CopterID, Vehicle
18 mav.sendUE4Pos(1,3,0,[0,0,-8.086],[0,0,0])
19 mav.sendUE4Pos(100,30,0,[2.5,0,-8.086],[0,0,0])
```

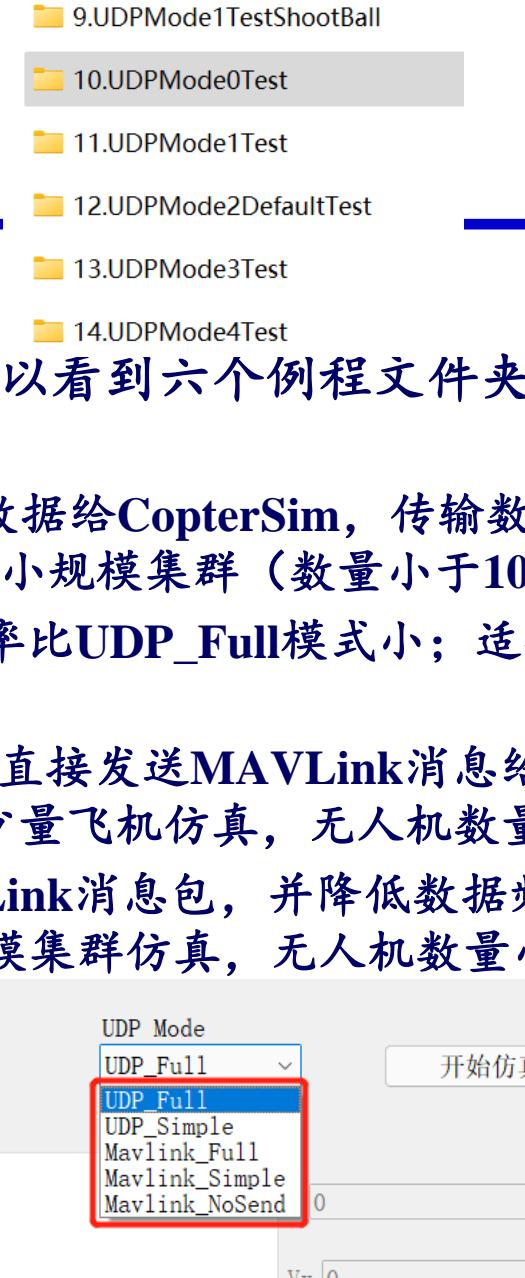




## 2.基础接口使用

### 2.3 Python/CopterSim数据模式（UDP\_Mode）

- 在“RflySimAPIs\6.RflySimExtCtrl\0.ApiExps”文件夹内，可以看到六个例程文件夹，可自行运行查看效果。主要区别在于语句InitMavLoop
- **mav.InitMavLoop(0)** # 对应**UDP\_Full**模式，Python传输完整的UDP数据给CopterSim，传输数据量小；CopterSim收到数据后，再转换为Mavlink后传输给PX4飞控；适合中小规模集群（数量小于10）仿真。
- **mav.InitMavLoop(1)** # 对应**UDP\_Simple**模式，数据包大小与发送频率比UDP\_Full模式小；适合大规模集群仿真，无人机数量小于100。
- **mav.InitMavLoop(2)** # 对应**Mavlink\_Full**模式（默认模式），Python直接发送MAVLink消息给CopterSim，再转发给PX4，数据量较大适合单机控制；适合单机或少量飞机仿真，无人机数量小于4；
- **mav.InitMavLoop(3)** # 对应**Mavlink\_Simple**模式，会屏蔽部分MAVLink消息包，并降低数据频率，发送数据量比MAVLink\_Full小很多，适合多机集群控制；适合小规模集群仿真，无人机数量小于8。
- **mav.InitMavLoop(4)** # 对应**Mavlink\_NoSend**模式，CopterSim不会向外发送MAVLink数据，此模式需要配合硬件在环仿真+数传串口通信，通过有线方式传输MAVLink，此模式局域网内数据量最小，适合分布式视觉硬件在环仿真，无人机数量不限制。





## 2.基础接口使用

注：免费版只支持2个RGB图像，  
以共享内存方式接收图像

### 2.4 RflySim3D取图接口—视觉传感器配置文件Config.json

- “RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\0.VisionSenorAPI\1.CameraImageGet” 可以打开Config.json文件，其中包含了两个视觉传感器结构体，定义如下
- SeqID;** //传感器序号ID，从**0**开始标号（免费版只支持2个图）
- TypeID;** //传感器类型ID，**1**:RGB图，**2**:深度图，**3**:灰度图，**4**:分割图，**5**:测距，**20-22**:激光雷达，**40**:红外灰度，**41**:热力图
- TargetCopter;** //相机装载的目标飞机的ID //可改变
- TargetMountType;** //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），**3**:固定飞机上，但相机姿态不随飞机变化（地面坐标系），**4**:将传感器附加到另外一个传感器上，当MountType == 4的时候，Config.json 中的 TargetCopter == SeqID （因为MountType == 4是将传感器附加到传感器上，所以TargetCopter本来是用于给定载具ID，这时候就没用了，就被用来设定传感器的ID了，也就是SeqID） //可变

注： TargetMountType决定了SensorPosXYZ的值是相对飞机中心，还是飞机底部中心，还是相对地面。另外，为保证物体能贴合地面，sendUE4\*\*命令发送的坐标都是物体底部中心坐标，而不是中心坐标，两者相隔物体高度（见XML定义）。

```
    "VisionSensors": [      {        "SeqID": 0,        "TypeID": 1,        "TargetCopter": 1,        "TargetMountType": 0,        "DataWidth": 640,        "DataHeight": 480,        "DataCheckFreq": 200,        "SendProtocol": [0, 127, 0, 0, 1, 9999, 0, 0],        "CameraFOV": 90,        "SensorPosXYZ": [0, 0, 0],        "SensorAngEular": [0, 0, 0],        "otherParams": [0, 0, 0, 0, 0, 0]      }    ]
```



## 2.基础接口使用

### 2.4 RflySim3D取图接口—视觉传感器配置文件Config.json

- **DataWidth:** //数据或图像宽度 (测距传感器无此参数)
- **DataHeight:** //数据或图像高度 (测距传感器无此参数)
- **DataCheckFreq:** //检查数据更新频率 (测距传感器无此参数)
- **SendProtocol[8]:** //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
- **EulerOrOuat:** 0表示使用欧拉角也就是**SensorAngEular**, 1表示使用四元数**SensorAngQuat**
- **CameraFOV:** //相机视场角 (仅限视觉类传感器), 单位度 //可改变
- **SensorPosXYZ[3]:** // 传感器安装位置, 单位米 //可改变
- **SensorAngEular[3]:** //传感器安装角度, 单位度。 //可改变
- **SensorAngQuat[4]:** //传感器安装角度, 用四元数表示。
- 对于测距传感器还有如下参数:

**Distance** :能探测到的最大距离

注: 免费版只支持2个RGB图像,  
以共享内存方式接收图像

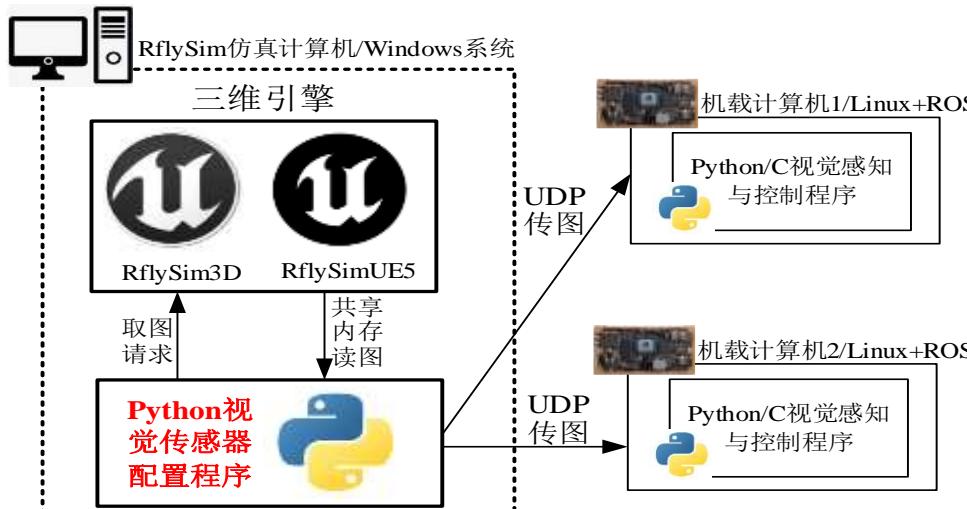
注: **TargetMountType**决定了**SensorPosXYZ**的值  
是相对飞机中心, 还是飞机底部中心, 还是相对地  
面。另外, 为保证物体能贴合地面, **sendUE4\*\***命  
令发送的坐标都是物体底部中心坐标, 而不是中心  
坐标, 两者相隔物体高度 (见XML定义)。



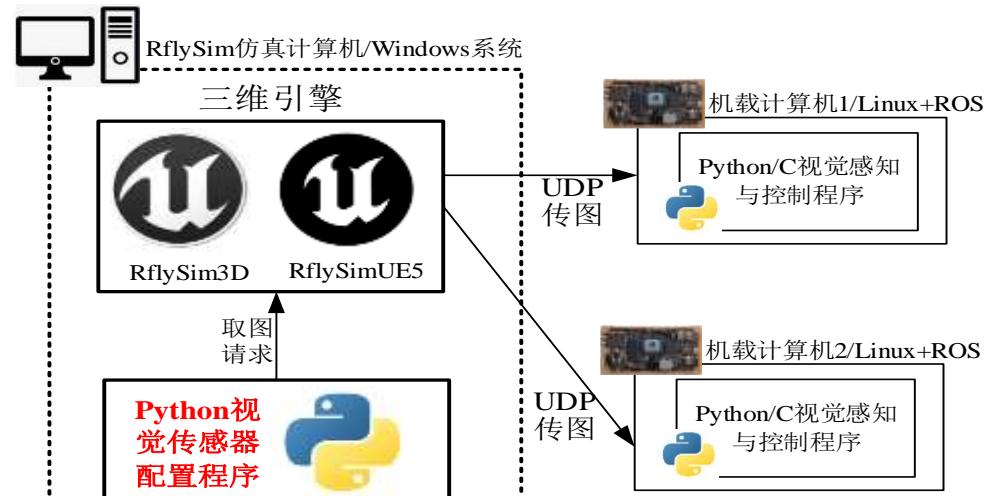
## 2.基础接口使用

### 2.4 RflySim3D取图接口—取图与分发原理

- RflySim平台必须运行于Windows电脑上，但是其图像流可以通过共享内存、网络通信等方式传输给本机视觉程序、其他电脑（Windows或Linux）、其他嵌入式计算机（Linux+ROS），用于视觉算法的开发与仿真验证。
- 将Config.json文件中的配置信息发送给RflySim3D可请求发送图片。
- SendProtocol[0]可选择传输模式，默认选择共享内存接收，完整版可选UDP直传。



(a) 共享内存取图，中转发送模式（适用免费版，存在延迟）



(b) RflySim3D图像直发（适用完整版，效率高延迟低）



## 2.基础接口使用

### 2.4 RflySim3D取图接口—取图接口 VisionCaptureApi.py

- VisionCaptureApi.py是本平台的取图接口文件，包含了json加载，图像请求，图像转发等
- class VisionSensorReq: # 数据结构体，发送给RflySim3D的取图数据包
- class imuDataCopter: # 数据结构体，CopterSim回传的IMU数据包
- class SensorReqCopterSim: # 数据结构包，发送给CopterSim请求传感器数据包
- class VisionCaptureApi: # 主接口类，实现了取图请求与接收
- addVisSensor(vs=VisionSensorReq()): # 类函数，增加一个视觉传感器
- sendReqToCopterSim(srcs=SensorReqCopterSim(),copterID=1): # 类函数，发送数据包给CopterSim，可以指定响应请求的CopterSim序号
- sendImuReqCopterSim(copterID=1,IP='127.0.0.1',port=31000,freq=200): # 类函数，发送数据包给CopterSim请求发送IMU数据（IP和端口频率），并开始监听数据
- sendUpdateUEImage(vs=VisionSensorReq(),windID=0): # 发送请求给RflySim3D，更新指定视觉传感器的参数、位置等，可指定接收的RflySim3D窗口号windID
- sendReqToUE4(windID=0): # 将存储的视觉传感器列表发送给RflySim3D，创建传感器，并检测是否创建成功，可指定接收的RflySim3D窗口号windID
- startImgCap(isRemoteSend=False): # 开始接收图片并存储到Img列表，isRemoteSend可配置共享内存图片是否向外通过UDP转发到其他系统中。
- jsonLoad(ChangeMode=-1,jsonPath=''): # 加载本地的Json文件，存储到视觉传感器列表中，ChangeMode可覆盖Json中的SendProtocol[0]传输模式，jsonPath可指定Json文件地址。



## 2.基础接口使用

### 2.4 RflySim3D取图接口—取图接口例程文件

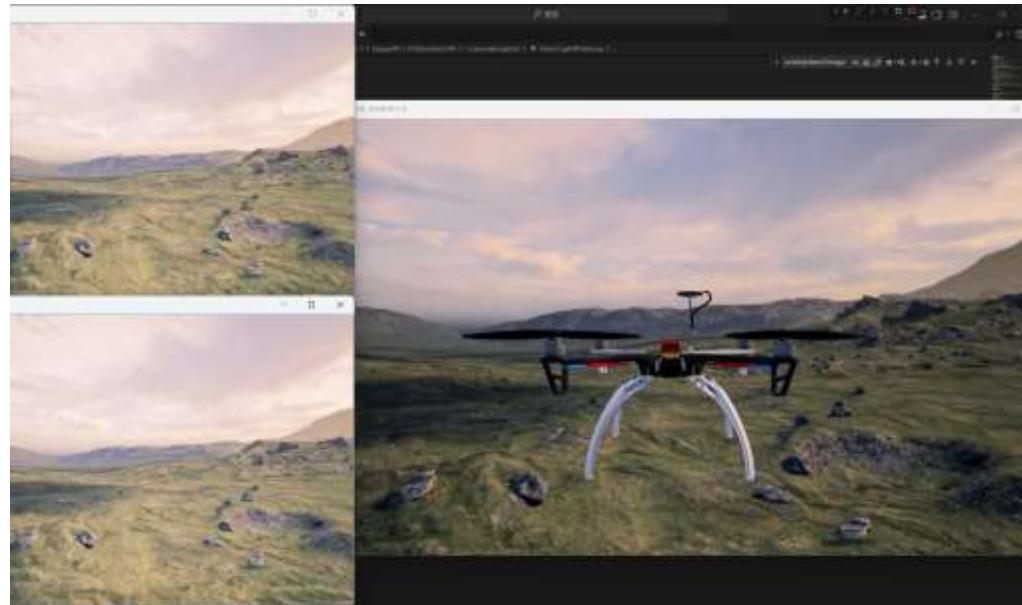
- 用VS Code打开“VisionCapAPIDemo.py”文件，可以看到例程实现原理
- vis = VisionCaptureApi.VisionCaptureApi() # 创建取图接口类实例
- ue.sendUE4Cmd('r.setres 1280x720w',0) # 设置UE4窗口分辨率，注意本窗口仅用于显示，取图分辨率在json中配置，本窗口设置越小，资源需求越少。
- ue.sendUE4Cmd('t.MaxFPS 30',0) # 设置UE4最大刷新频率，同时也是取图频率
- vis.jsonLoad() # 加载Config.json中的传感器配置文件
- isSuss = vis.sendReqToUE4() # 向RflySim3D发送取图请求，并验证
- vis.startImgCap() # 开启取图，并启用共享内存图像转发，转发到填写的目录
- if vis.hasData[i]: # 判断图片是否已经接收到
- cv2.imshow('Img'+str(i),vis.Img[i]) # OpenCV显示当前图像
- # 这里可以添加图像处理算法
- vis.sendUpdateUEImage(vs) # 发送请求，更新视觉传感器参数



## 2.基础接口使用

### 2.4 RflySim3D取图接口—实验验证

- 打开“RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\0.VisionSenorAPI\1.CameraImageGet\VisionCapAPIDemo.bat”以管理员身份运行，会开启。
- 本实验中，Json定义了两个左右两个前置RGB相机，并实时显示。

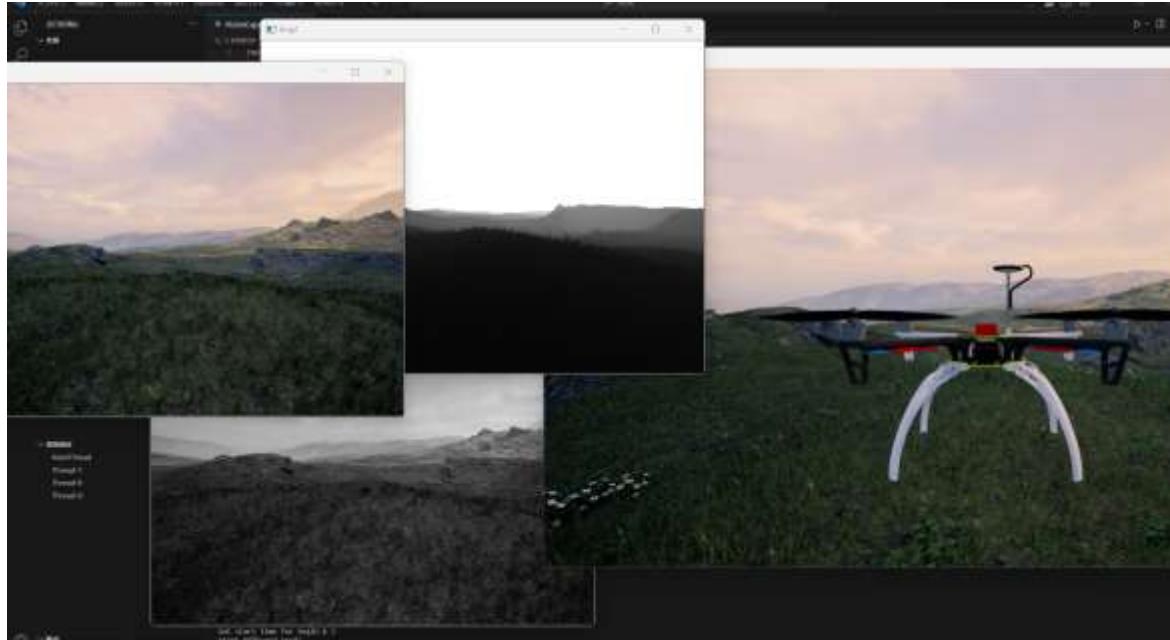




## 2.基础接口使用

### 2.4 RflySim3D取图接口—UE5多目相机实验验证（限完整版）

- 编辑“RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\0.VisionSenorAPI\2.MutCameraImageGet\VisionCapAPIDemo.bat”，可以看到“%PSP\_PATH%\RflySimUE5”说明使用UE5引擎；打开“Config.json”可以看到包含RGB、灰度、深度三个相机；
- 双击“VisionCapAPIDemo.bat”，再用VS Code运行“VisionCapAPIDemo.py”，可见如下效果





## 2.基础接口使用

---

### 2.5 PX4无人机位置+速度控制三种模式—原理讲解

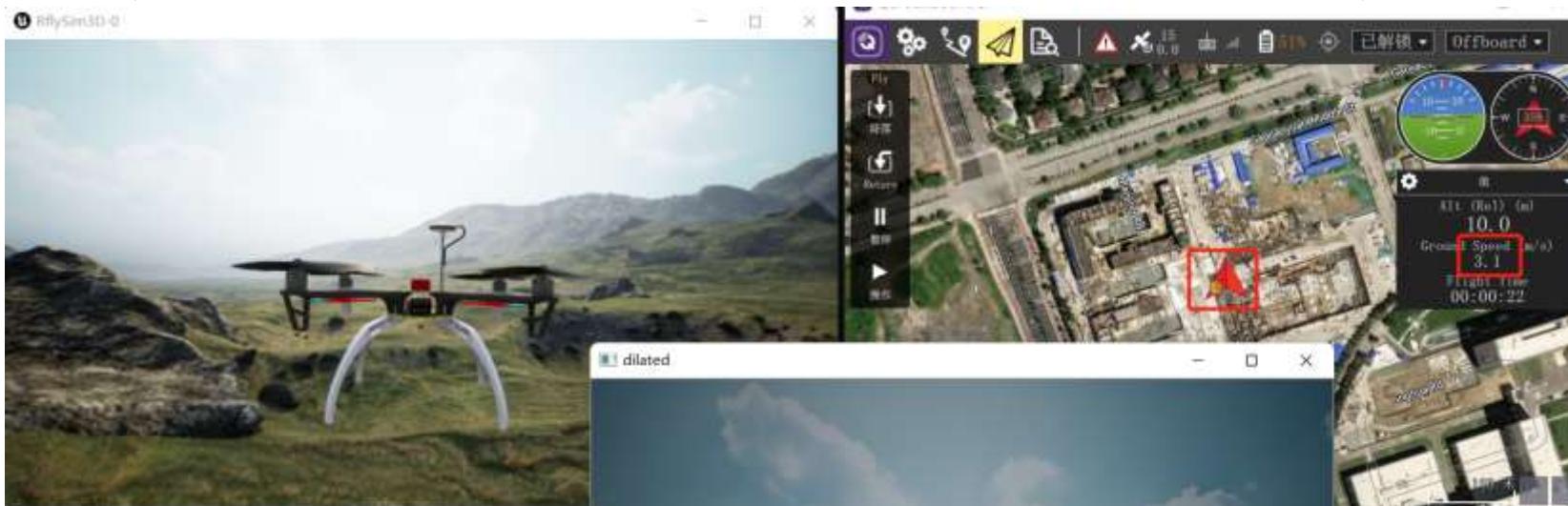
- 在视觉控制时，我们常常需要在控制飞机飞往指定目标位置的同时，控制飞机的前飞速度，达到好的跟踪效果。
- 进入“RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\2.ThreeCtrlModes”目录，可看到三种方法的控制例子。
- 方法一，例程见“ThreeCtrlModes\_PosCtrl.py”，使用Offboard位置接口
  - `mav.SendPosNED(12,13,-10,0)`# 发送期望位置
  - `mav.SendCopterSpeed(3)`#调用函数来设定飞机最大速度
- 方法二，例程见“ThreeCtrlModes\_VelCtrlEarth.py”，使用SendVelNED速度控制接口，在此基础上实现位置控制。
- 方法三，例程见“ThreeCtrlModes\_VelCtrlBody.py”，使用SendVelFRD机体速度控制接口，在此基础上实现位置控制，本模式机头始终朝向目标方向。



## 2.基础接口使用

### 2.5 PX4无人机位置+速度控制三种模式—实验效果

- 运行“ThreeCtrlModesSITL.bat”或“ThreeCtrlModesHITL.bat”，开启一个飞机的软件在环或硬件在环仿真。
- 运行“ThreeCtrlModes\_PosCtrl.py”，可以看到飞机飞往指定位置[120,130,-10]，在QGC上可以看到速度为设定值3m/s，同时机头始终朝北和速度方向不一致。
- 依次运行“\*\_PosCtrlFRD.py”、“\*\_VelCtrlBody.py”、“\*\_VelCtrlEarth.py”，请自行阅读代码，并确认实验效果。注：三种模式都可以控制飞机速度，并始终指向目标。





## 2.基础接口使用

注：本质点模型的初始高度求取方法，以及多机仿真的详细方法，可以阅读下一章相关内容。

### 2.6 轻量级无人机模型下的控制接口—实验原理

- 在上面的例子中，运行bat脚本都会开启飞机的软件在环或硬件在环仿真，需要CopterSim+飞控+QGC参与，占用资源较多，在多机视觉仿真时可能受到性能限制。
- 在“RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\1.UAVCtrlNoPX4Demo\1.UAVCtrlNoPX4Demo”目录中，我们在Python中开发了一个基于质点的无人机控制模型，能够提供软硬件在环仿真相近的无人机动态效果，但是极大降低对电脑性能的占用和提升飞机平稳性。
- 用VS Code打开“UAVCtrlNoPX4Demo.py”可以看到，本模式和SITL或HITL的基于MAVLink的控制接口完全相同，区别在于下面语句。

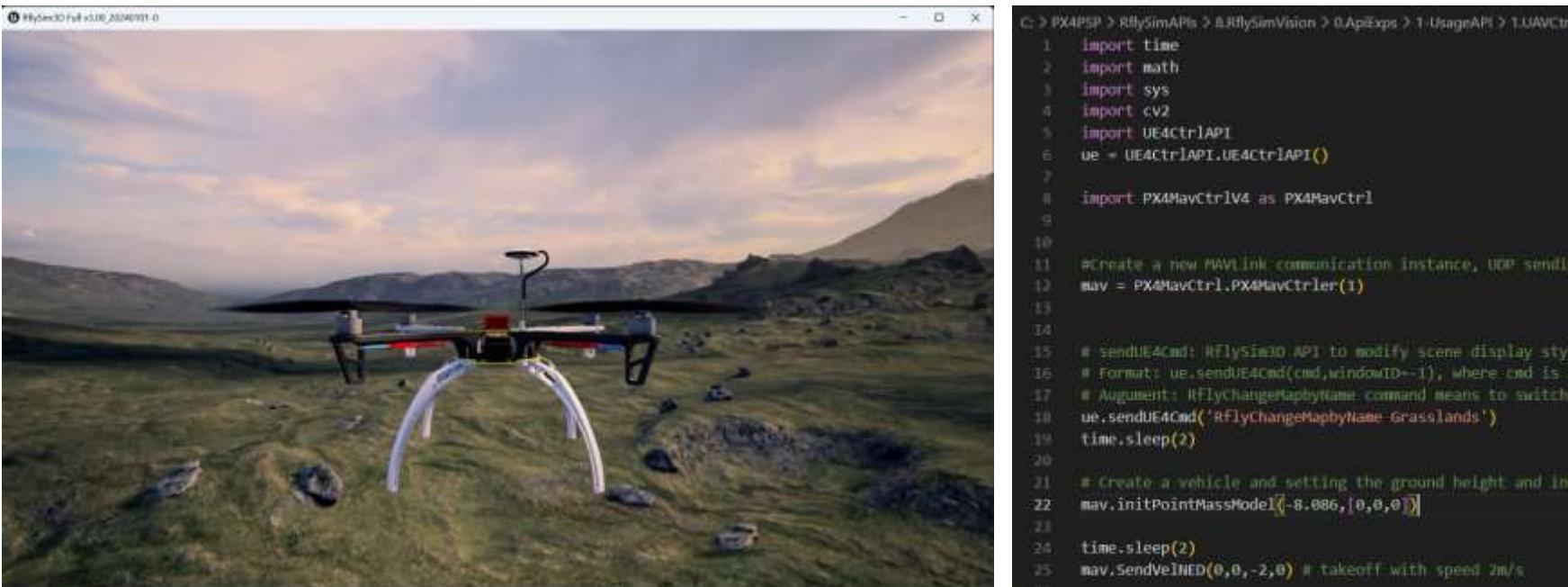
```
mav.initPointMassModel(-8.086,[0,0,0]) # 替换原initOffboard语句
```
- 上述语句执行后，会自动新建一个质点无人机模型（设定初始地面高度、XY位置、偏航角度），并监听位置和速度指令。（和原来的控制方法完全一致）
- 注：用VS Code打开“UAVCtrlNoPX4Demo.bat”可以看到我们只打开了一个RflySim3D程序，不需要开启CopterSim等其他程序。



## 2.基础接口使用

### 2.6 轻量级无人机模型下的控制接口—实验效果

- 双击运行“UAVCtrlNoPX4Demo.bat”可以看到打开一个RflySim3D窗口，没有其他程序打开。
- 用VS Code打开“UAVCtrlNoPX4Demo.py”并运行，可以看到飞机起飞，并按照发送的位置或速度命令飞行，飞行控制效果与软/硬件在环相近，但更平稳。





## 2.基础接口使用

### 2.6 轻量级无人机模型下的控制接口—视觉穿环实验

- 打开“RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e1\_CrossRingNoPX4”目录，可以看到基于质点模型的穿环实验例程。（具体视觉控制原理，可参考下一节穿环实验内容）
- 双击“CrossRingNoPX4.bat”可打开一个RflySim3D窗口
- 用VS Code打开“CrossRingNoPX4.py”并运行，可以看到场景切换到草地穿环场景，生成了一个多旋翼飞机，起飞后依次穿越三个环。





## 2.基础接口使用

注：激光雷达传感器目前仅限高级完整版，免费体验版无此功能。

### 2.7 激光雷达——源码说明

目前数据交互使用两种方式： UDP直传与内存共享方式，以下对源码目录说明

RflySimAPIs\8.RflySimVision\2.AdvExps\e0\_AdvApiExps\7.LidarAPIDemo 目录下

- **1.SharedMemory10Hz:** 共享内存方式，传输数据频率10hz;
- **2.SharedMemoryClientServer:** 共享内存方式， Client客户端共享内存接收后， UDP发出， Server接收点云
- **3.UDPDIRECT30Hz:** UDP直传30hz频率， Python发送取图请求给RflySim3D， 后者直接通过UDP传出点云；
- **4.UDPDIRECTClientServer:** UDP直传客户端与服务端传输方式， Client发送请求， Server接收点云； 注： server支持虚拟机或其他电脑
- **5.UDPDIRECTClientServerType5:** UDP直传客户端与服务端传输方式， 使用地图坐标系(默认使用激光雷达坐标系);

- 1.SharedMemory10Hz
- 2.SharedMemoryClientServer
- 3.UDPDIRECT30Hz
- 4.UDPDIRECTClientServer
- 5.UDPDIRECTClientServerType5



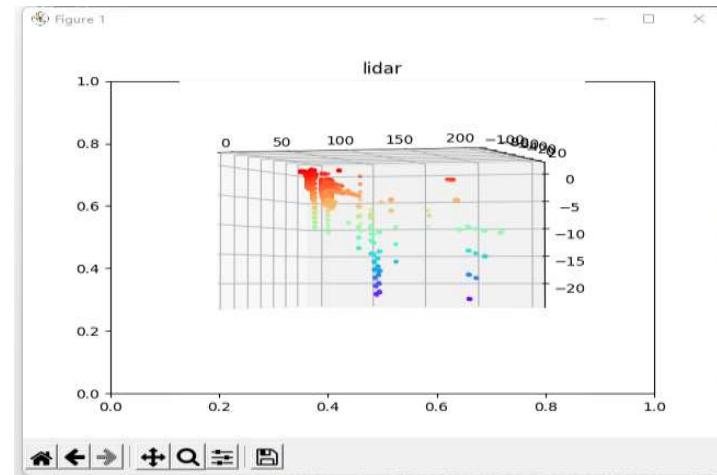
## 2.基础接口使用

### 2.7 激光雷达——源码说明2

- 虽有工程目录下有5个项目，但是传输方式就只有两种，现在分别对这两种方式的代码运行做下说明，以1.SharedMemory10Hz 和 4.UDPDIRECTClientServer为例：
- **1.SharedMemory10Hz**: 运行LidarAPIDemo.bat脚本之后，直接运行LidarAPIDemo.py脚本即可看到可视化点云
- **4.UDPDIRECTClientServer**: 远程通信需要配置ip，首先查看自己虚拟机ubuntu的ip：  
按Ctrl+Alt+T 打开终端，输入命令“ifconfig”(命令没有引号)如图：

注：共享内存还是UDP直传，由SendProtocol[0]的数值决定。在Python中调用jsonLoad函数时，附带参数可以强行设定SendProtocol[0]的取值，更改传输模式。

```
    "DataCheckFreq":10,  
    "SendProtocol":1,127,0,0  
    "CommandFOV":90  
    # VisionCaptureApi  
vis.jsonLoad(1) #
```





## 2.基础接口使用

### 2.7 激光雷达——源码说明3

- 如果要传输点云到远端的Linux\ROS系统中，需要修改json配置文件或client\_ue4.py 设定好IP地址。
- 如我的虚拟机或Linux电脑的ip为192.168.31.88，那么json配置文件如下图：

```

X4PSP > RilySimAPIs > 8.RilySimVision > 2.AdvExps > eu_AdvApiExps > 7.LidarAPIDemo >
{
    "VisionSensors": [
        {
            "SeqID":0,
            "TypeID":20,
            "TargetCopter":1,
            "TargetMountType":0,
            "DataWidth":900,
            "DataHeight":32,
            "DataCheckFreq":10,
            "SendProtocol": [1,192,168,31,88,9999,0,0],          替换成虚拟机IP地址
            "CameraFOV":90,                                     ←
            "SensorPosXYZ": [0,0,-0.3], 1表示UDP直传
            "SensorAngEular": [0,0,0],
            "otherParams": [200,0.05,-45,45,-20,20,0,0]
        }
    ]
}

```

```

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
→ - ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.31.88 netmask 255.255.255.0 broadcast 192.168.31.255
inet6 fe80::1179:12ff:fe1a:5a:f74f prefixlen 64 scopeid 0x20<link>
inet6 2001:250:4400:400:f:ae0:5b66:696f prefixlen 64 scopeid 0x0<
global>
inet6 2001:250:4400:400:f:5fae:694c:c36c:362a prefixlen 64 scopeid 0x0<
global>
inet6 2001:250:4400:400:f:b09e:76c1:93e2:587e prefixlen 64 scopeid 0x0<
global>
ether 00:0c:29:b6:51:a7 txqueuelen 1000 (以太网)
RX packets 174312209 bytes 135409277765 (135.4 GB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 7482414 bytes 813732686 (813.7 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: Flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (本地环回)
RX packets 3993480 bytes 142917850805 (142.9 GB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 3993480 bytes 142917850805 (142.9 GB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

注：也可以让json中SendProtocol中IP地址保持不变，使用127.0.0.1，但是在client\_ue4.py中取消注释vis.RemotSendIP并设置为IP地址

```

19 vis.RemotSendIP = '192.168.31.88'
20 # 注意，手动修改RemotSendIP的值，可以将

```

注：本方法会修改不json中定义的所有传感器的目标IP地址。



## 2.基础接口使用

### 2.7 激光雷达——源码说明3

- 另外还需要修改脚本client\_ue4\_SITL.bat中的参数，来让其他电脑可以收到CopterSim的飞机数据，飞机的实时控制。
- IS\_BROADCAST =1 对应广播模式，所有电脑都能收到数据，效率低，但是方便
- IS\_BROADCAST = 192.168.31.88(根据自己虚拟机来配置)，效率高。

```
40
41
42 REM Set broadcast to other computer; 0: only this computer, 1
43 REM e.g., IS_BROADCAST=0 equals to IS_BROADCAST=127.0.0.1, IS
44 SET IS_BROADCAST=192.168.31.88
45
46 REM Set UDP data mode; 0: UDP_FULL, 1:UDP_Simple, 2: Mavlink_
47 REM e.g., UDPSIMMODE=1 equals to UDPSIMMODE=UDP_Simple
48 SET UDPSIMMODE=2
49
50 REM Set the path of the RflySim tools
51 SET PSP_PATH=C:\PX4PSP
52 SET PSP_PATH_LINUX=/mnt/c/PX4PSP
53
```



## 2.基础接口使用

### 2.7 激光雷达——源码测试总结

- **1.SharedMemory10Hz**: 运行LidarAPIDemo.bat脚本，等RflySim3D提示Fixed后，用VS Code打开并运行LidarAPIDemo.py脚本查看点云，通过QGC控制飞机运动，观察点云变化。
- **2.SharedMemoryClientServer**: 运行client\_ue4\_SITL.bat脚本，待Fixed后双击运行Python38Run.bat，在其中输入python client\_ue4.py，从共享内存读点云并UDP发出，用VS Code打开server\_ue4.py并运行。
- **注意1**: 之所以client会将点云通过UDP传出，是因为“vis.startImgCap(True)”语句的作用（相对于例程1，多加了True的参数数据），会触发点云转发功能。
- **注意2**: 将json文件中SendProtocol的IP地址改成远端地址，就能将点云传输到Linux电脑；更简单的方式是不改SendProtocol，直接将client中下面语句取消注释，并填入目标IP地址，也会自动将点云转发到本IP地址上。

```
18  # vis.RemotSendIP = '192.168.3.80'  
19  # 注意，手动修改RemotSendIP的值，可以将图  
20  # 如果不修改这个值，那么发送的IP地址为js
```



## 2.基础接口使用

---

### 2.7 激光雷达——源码测试总结

- **3. UDPDirect30Hz:** 测试方法同1.SharedMemory10Hz，先运行bat（可使用管理员方式），再用VS Code运行python脚本，通过QGC控制飞机，观察点云变化。  
本例程的特点：RflySim3D以90帧运行，点云获取速度30帧，UDP直传本机IP。
- **4. UDPDirectClientServer:** 先运行client\_ue4\_SITL.bat（管理员），再运行Python38Run.bat，命令行运行Python client\_ue4.py，VS Code运行server\_ue4.py。
- 注：如果要传点云到其他电脑，需要修改.bat 的IS\_BROADCAST 值和client\_ue4.py 的vis.RemotSendIP对应的IP地址，如果是ROS系统可运行server\_ue4ROS.py来发布点云ROS包，并进行预览。
- **5. UDPDirectClientServerType5:** 和上一个例程的测试方式相同，只不过Type5对应的坐标系轴向是相对地球的，叠加上ImgData中的雷达位置就能得到地图点云。如果是Type4模式，点云是相对雷达姿态的，需要映射到地面坐标系得到地图点云。



## 2.基础接口使用

---

### 2.7 激光雷达——数据接口：

- **vis.VisSensor[i]**: 第*i*个传感器（对应json文件的SeqID，传感器序号）的json数据结构体，可以获取当前传感器的参数配置信息。
- **vis.hasData[i]** : 第*i*个传感器有没有更新数据，有数据更新会变为True。
- **vis.Img[i]**: 第*i*个传感器的图像/点云数据，N\*3维向量数组（N的值等于vis.ImgData[i][6]），具体数据获取方法见例程。点云数据单位米（前-左-上坐标系）。
- **vis.timeStamp[i]**: 第*i*个传感器的时间戳，单位毫秒
- **vis.ImgData[i]**: 7维向量，vis.ImgData[i][0~2]第*i*个传感器的位置xyz（单位米，前-左-上坐标系）、vis.ImgData[i][3~5]姿态欧拉角roll,pitch,yaw（单位弧度）、vis.ImgData[i][6]点云总数。
- 注：本例中只有一个传感器，因此*i*的值取0。
- 注：控制飞机和获取飞控数据的接口是mav，见飞机控制的接口例程。



## 2.基础接口使用

### 2.7 激光雷达——参数配置说明

- 配置参数在8-LidarAPIDemo 目录下的Config.json文件里
- 这里说明激光雷达配置特有的参数,其他参数见2.6 Config.json配置文件说明:

- TypeID: 取值20-22; 20:代表输出点云为激光雷达坐标系, 21:代表输出点云为世界坐标系;22:代表livox激光雷达;
- DataWidth: 激光雷达一个ring内的点云个数;
- DataHeight: 为激光雷达线束数量。 DataCheckFreq: 点云发布频率(hz)
- DataHeight: 激光雷达线束数量;
- SendProtocol: 传输模式和IP, 其中SendProtocol[0]表示共享内存输出模式, SendProtocol[1]表示UDP直发模式。
- otherParams: [激光最远距离(m),精度(m),水平扫描角度下限值(度),水平扫描角度上限值(度),垂直扫描角度下限值(度),垂直扫描角度上限值(度),预留,预留];

注: 激光雷达水平分辨通过DataWidth和水平扫描角度范围体现, 垂直分辨率通过处置扫描角度体现(如图中的水平分辨率=90/900), 垂直分辨率=40/32), 以上角度值都是用degree表示。

注: 本取点云的接口存在一个UE帧率的延迟, 因此通过Python中“vis.sendUE4Cmd(b't.MaxFPS 30', 0)”语句可以加快UE4发送频率, 减小数据发送延迟。例如: UE帧率设置为100帧, 则输出延迟仅为0.01s。

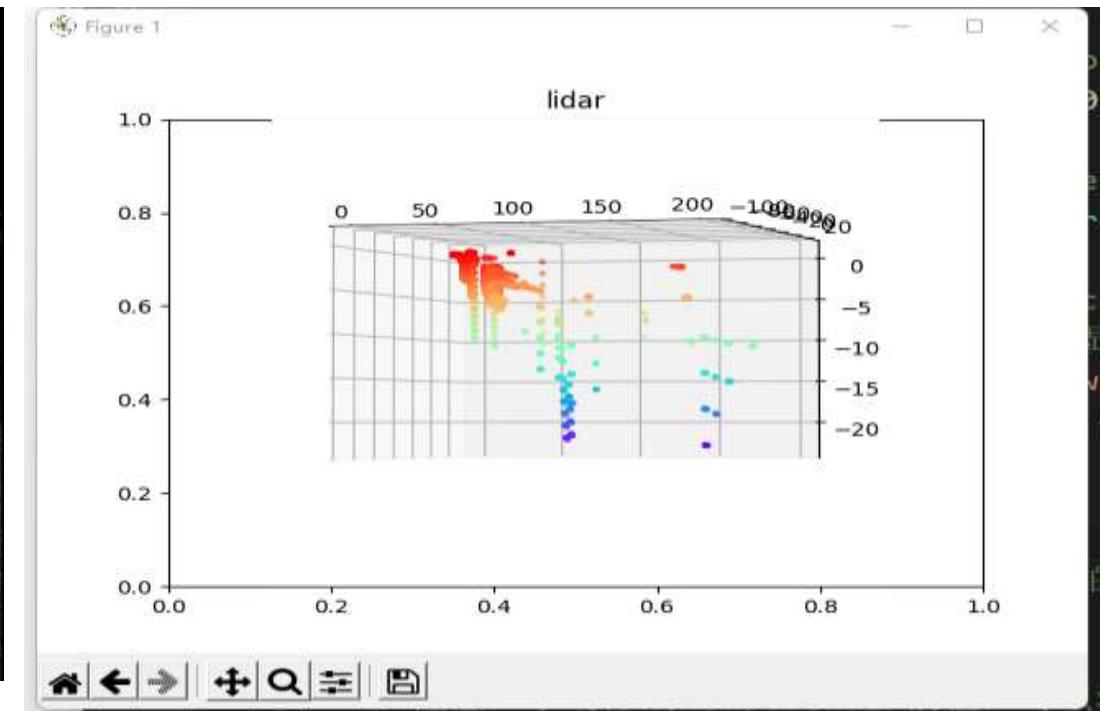
```
{  
    "VisionSensors": [  
        {  
            "SeqID": 0,  
            "TypeID": 20,  
            "TargetCopter": 1,  
            "TargetMountType": 0,  
            "Datawidth": 900,  
            "DataHeight": 32,  
            "DataCheckFreq": 10,  
            "SendProtocol": [0, 127, 0, 0, 1, 9999, 0, 0],  
            "CameraFOV": 90,  
            "SensorPosXYZ": [0, 0, -0.3],  
            "SensorAngEular": [0, 0, 0],  
            "otherParams": [200, 0.05, -45, 45, -20, 20, 0, 0]  
        }  
    ]  
}
```



## 2.基础接口使用

见1.SharedMemory10Hz的例程，关键点json中SendProtocol[0]设置为0。

- 2.7 激光雷达——共享内存方式运行
- 先启动RFlySim3D，即运行LdiarAPIDemo.bat文件，然后运行LidarAPIDemo.py脚本，就可以看到下面的运行
- 注意目前点云输出是北东坐标系，也就是Z轴朝下，所以点云是倒置的。



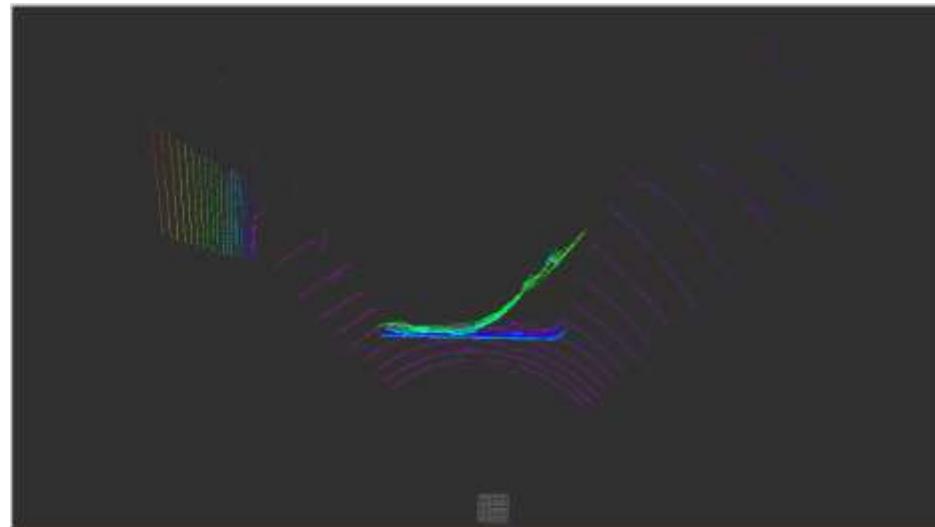
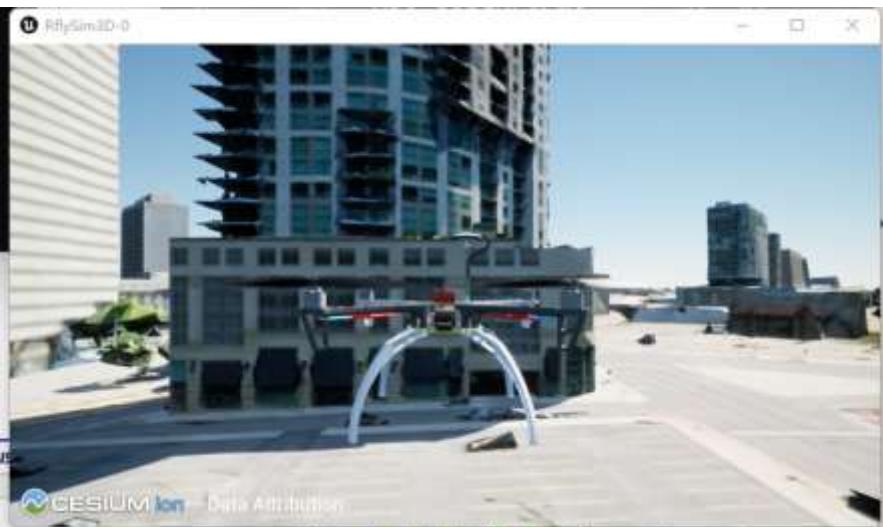


## 2.基础接口使用

### 2.7 激光雷达——UDP直传远端电脑方式运行

- 运行前先配置远程通信的参数，详见2.6参数配置说明。
- 运行RflySim3D，即client\_ue4\_SITL.bat脚本（管理员方式），然后在本机用VS Code或Python38Run.bat命令框，运行数据转换脚本client\_ue4.py，在远程端（虚拟机或Linux电脑）运行server\_ue4.py就可以看到和共享内存方式一样的运行结果，不过推荐使用ROS去显示点云，所以不运行远程端的server\_ue4.py，先运行roscore，然后使用python server\_ueROS.py运行程序，打开rviz就能看到下面运行结果。

见4.UDPDirectClientServer的例程，  
关键点json中SendProtocol[0]设置为1，  
并填入远端Linux电脑IP地址。





## 2.基础接口使用

### 2.7 激光雷达——UDP直传方式运行(ROS通信说明)

- 运行server\_ue4ROS.py后会有两个话题发布

➤ /rflysim/vehicle0\_pose: 里面的数字0是的载体编号;  
➤ /rflysim/vhicle\_0/lidar\_0: 后面的0是激光雷达编号，注意这个雷达  
编号是整个RFlySim里面的雷达统一编号，而非相对某个载体的雷达编号；

通过rostopic echo topic\_name查看数据，如：

可以看到位姿和点云数据，另外数据的  
frame\_id的值可以修改，根据自己的tf树需要  
自行修改代码。点云坐标可以是相对激光雷达  
坐标系，也可以是相对地面，具体看自己配  
置。

```
---  
header:  
  seq: 68  
  stamp:  
    secs: 0  
    nsecs:          0  
    frame_id: "vehicle0_pose"  
pose:  
  position:  
    x: 52.8860588074  
    y: -1.28458702564  
    z: 0.0112491119653  
  orientation:  
    x: -0.0033150415185  
    y: -0.00428877121906  
    z: 0.517733180595  
    w: 0.855524967872
```

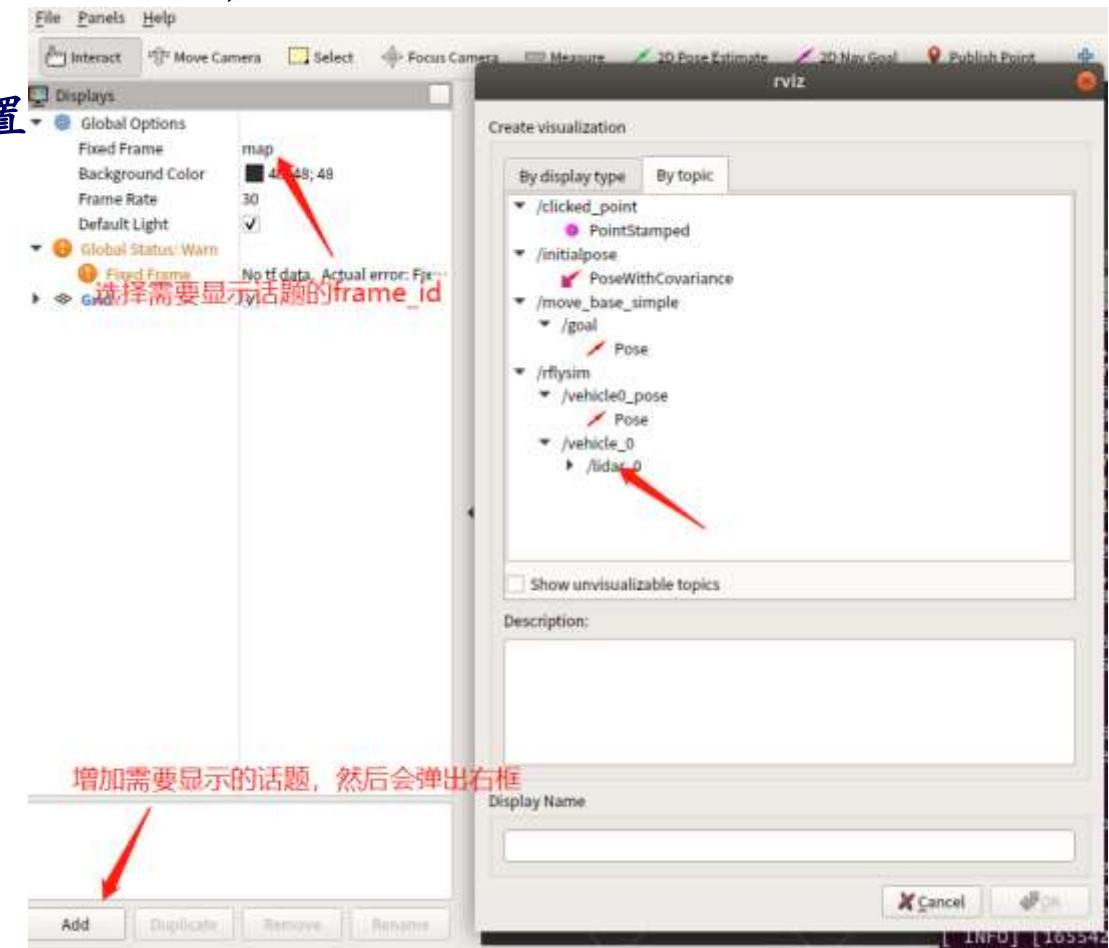
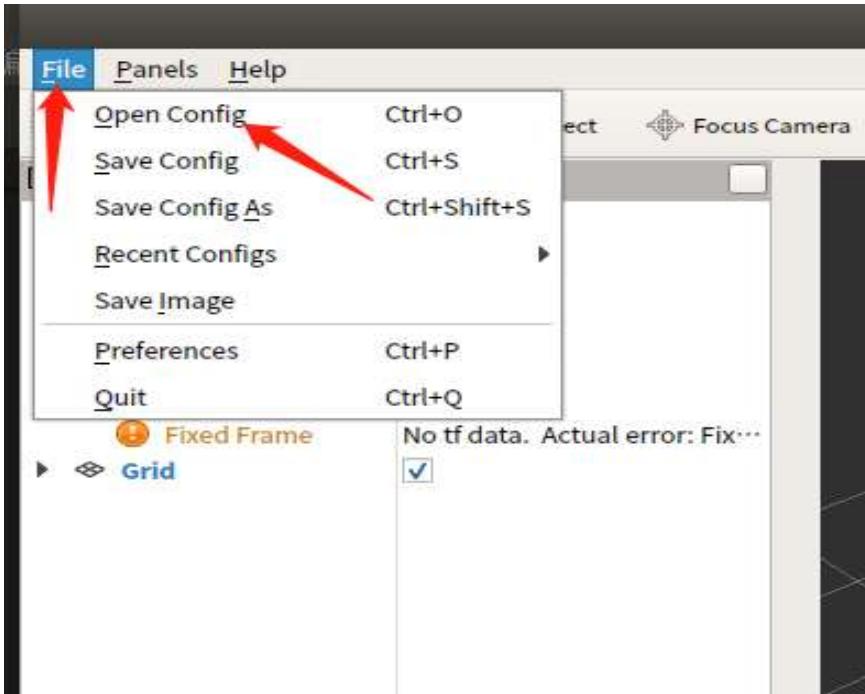
```
→ ~ rostopic list  
/rflysim/vehicle0_pose 发布载体位姿  
/rflysim/vehicle_0/lidar_0 发布点云  
/rosout  
/rosout_agg  
→ ~ [REDACTED]  
  
seq: 609950  
stamp:  
  secs: 1655367176  
  nsecs: 38000106  
  frame_id: "rflysim_vehicle0"  
height: 1  
width: 22818  
fields:  
  -  
    name: "x"  
    offset: 0  
    datatype: 7  
    count: 1  
  -  
    name: "y"  
    offset: 4  
    datatype: 7  
    count: 1  
  -  
    name: "z"  
    offset: 8  
    datatype: 7  
    count: 1  
is_bigendian: False  
point_step: 12  
row_step: 273816  
data: [127, 105, 191, 64, 126, 55, 191, 192  
  45, 64, 126, 55, 191, 64, 126, 5, 191, 192  
, 64, 127, 105, 191, 64, 126, 55, 191, 192,  
  253, 63, 127, 155, 191, 64, 127, 105, 191,  
  127, 105, 191, 64, 126, 55, 191, 192, 253, 63, 127, 155, 191, 64, 127, 105, 191,
```



## 2.基础接口使用

### 2.7 激光雷达——UDP传输方式运行( ROS, rviz可视化)

Rviz工具可以方便的查看环境点云数据，需要配置，如果是默认代码则可以直接加载源文件目录下的已经配置好了的lidar.rviz文件。





## 2.基础接口使用

### 2.8 深度相机—Json配置文件

- 深度相机的json文件配置方法与RGB与灰度相机的配置方法基本相同。
- 深度相机的TypeID需要设置为2，其他参数见2.3节，包括分辨率DataWidth和DataHeight、绑定飞机TargetCopter、绑定类型TargetMountType、刷新率DataCheckFrequently、视角CameraFOV、安装位置SensorPosXYZ（单位米）和安装姿态SensorPosEular。（单位度）
- 其他参数存储在OtherParams向量中，包括最小范围、最大范围、像素精度。

```
{  
    "SeqID":1,  
    "TypeID":2, 类型ID为2  
    "TargetCopter":1,  
    "TargetMountType":0,  
    "DataWidth":640,  
    "DataHeight":480, 分辨率和刷新率  
    "DataCheckFreq":30,  
    "SendProtocol": [0,127,0,0,1,10000,0,0],  
    "CameraFOV":90,  
    "SensorPosXYZ": [0.3,0,0], 位置姿态  
    "SensorAngEular": [0,0,0],  
    "otherParams": [0.3,12,0.001,0,0,0,0,0]  
}
```

**其他参数 (视觉范围和进度)**



## 2.基础接口使用

---

### 2.8 深度相机—特有参数

- 注意：深度相机输出的数据是以uint16存储和传输的，它的数据范围是0~65535。默认情况下，一个单位表示1mm（由otherParams[2]控制），也就是说最大范围是0到65.535米。但是，数据范围并不代表相机的实际探测距离，还需要otherParams[0]设置最小探测距离和otherParams[1]设置最大探测距离。
- otherParams[0]: 深度相机的最小识别距离（单位米），如果深度距离小于本值，那么输出NaN对应65535。
- otherParams[1]: 深度相机的最大识别距离（单位米），如果深度距离大于本值，那么输出NaN对应65535。
- otherParams[2]: 深度相机uint16输出值的刻度单位（单位米），默认情况下深度值以毫秒为单位，因此需要填0.001。注，默认值填0的话，会被替换为otherParams[2]=0.001。
- 实际深度值（单位米） = 深度图片值（uint16范围） \* otherParams[2]



## 2.基础接口使用

---

### 2.8 深度相机—实验流程

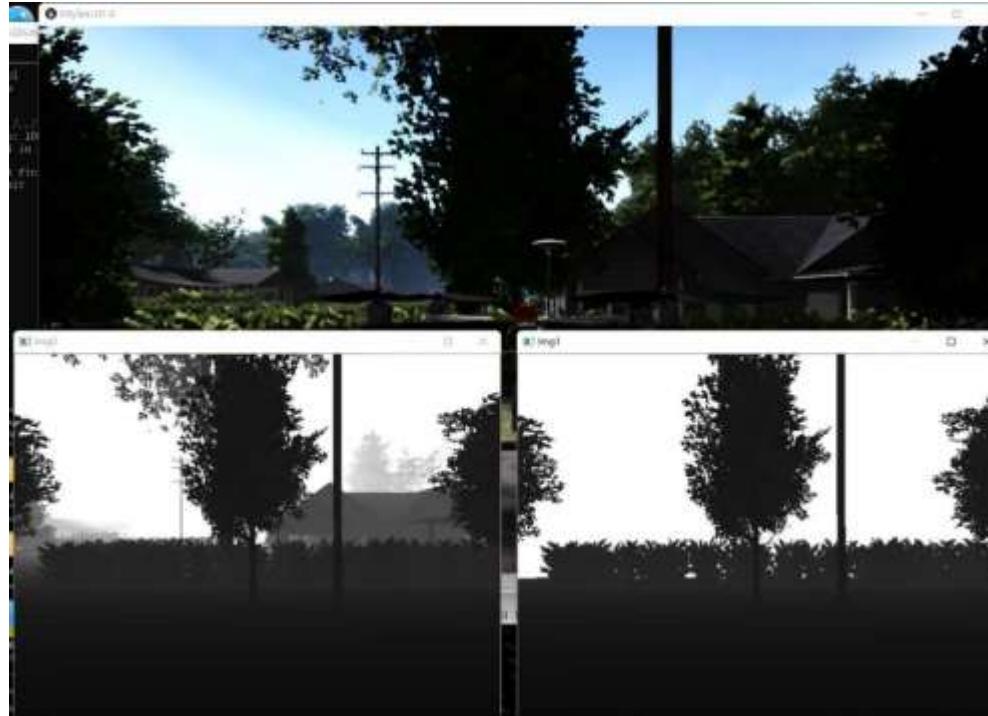
- 深度相机的例子见RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\0.VisionSenorAPI\5.DepthCameraDemo，先运行DepthCameraDemo.bat脚本开启仿真，待CopterSim显示3DFixed后，在运行DepthCameraDemo.py开启控制和视觉识别。
- 本例子包含了两个深度相机，都包含在了Config.json中。
- 在Config.json的第一个视觉结构体中，otherParams全0，表示使用默认配置。单位mm，最小距离0米，最大距离65.535米。
- 在Config.json的第2个视觉结构体中，otherParams[0]=0.3，otherParams[1]=12，otherParams[0]=0.001。表示最近识别距离0.3米，最远距离为12米，刻度单位为毫米。
- 在Python例程中，可以从vis.Img[i]可以直接读取深度图像矩阵。vis.hasData[i]来判断数据是否更新。
- 在运动过程中，可以通过vis.sendUpdateUEImage(vs)来动态调整相机参数。



## 2.基础接口使用

### 2.8 深度相机—测试结果

- 测试结果如下图，飞机从地面起飞，然后开启视觉探测窗口，可以输出两幅深度图片。左边的图使用默认配置，无范围约束可以看到远程建筑；右边深度图开启最大探测距离12米，因此无法看到远处建筑。





## 2.基础接口使用

### 2.9 时间戳—结构体定义

- 时间戳的结构体定义代码如右图所示。
- 接收的时候，需要从20005的UDP端口接收数据，然后校验包长是否等于32字节，再校验checksum是否为预设的123456789，还需要判断copterID是否为自己飞机的ID，Python的例程代码如右图所示。
- 具体代码可见RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\6.ReadTimeStmp的ReadTimeStmp.py

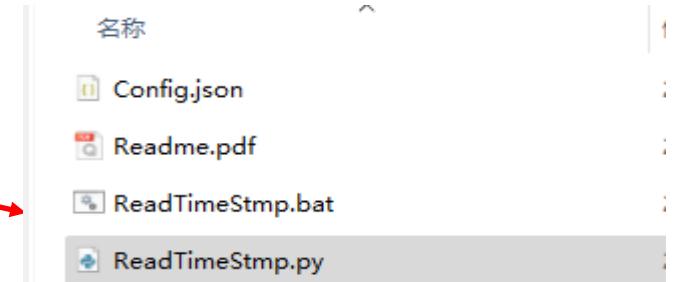
```
struct RflyTimeStmp{  
    int checksum; //校验位, 取123456789  
    int copterID; //当前飞机的ID号  
    long long SysStartTime; //Windows下的开始仿真时的时间戳 (单位毫秒, 格林尼治标准起点)  
    long long SysCurrentTime; //Windows下的当前时间戳 (单位毫秒, 格林尼治标准起点)  
    long long HeartCount; //心跳包的计数器  
    RflyTimeStmp(){  
        reset();  
    }  
    void reset(){  
        checksum=123456789;  
        copterID=-1;  
        SysStartTime=-1;  
        SysCurrentTime=-1;  
        HeartCount=0;  
    }  
};  
  
def StartTimeStmplisten(self,cpID=0):  
    """Start to listen to 20005 port to get RflyTimeStmp of CopterID  
    if cpID == 0 then only current CopterID will be listened.  
    if cpID >0 then timestamp of desired CopterID will be listened.  
    """  
    self.udp_time = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # 创建套接字  
    self.udp_time.bind(('0.0.0.0',20005)) # 绑定20005端口  
    self.tTimeStmp = threading.Thread(target=self.TimeStmloop, args=0) //启动一个监听线程  
    self.cpID=cpID  
    if cpID==0:  
        self.cpID=self.CopterID  
    self.tTimeStmp.start() # 开启线程  
  
def TimeStmloop(self):  
    print("Start lisening to timeStmp Msg")  
    while True: # 死循环  
        try:  
            buf,addr = self.udp_time.recvfrom(65500) # 从20005口读一个包  
            if len(buf)==32: # 判断是否等于32长度  
                #print(len(buf[0:12]))  
                TimeData=struct.unpack('2i3q',buf) # 解码为结构体  
                if TimeData[0]==123456789: # 第0位checksum是否为123456789  
                    cpIDTmp = TimeData[1]  
                    if cpIDTmp == self.cpID: # 第1位copterID是否为期望飞机的  
                        self.RflyTime.checksum=TimeData[0] #给结构体赋值  
                        self.RflyTime.copterID=TimeData[1]  
                        self.RflyTime.SysStartTime=TimeData[2]  
                        self.RflyTime.SysCurrentTime=TimeData[3]  
                        self.RflyTime.HeartCount=TimeData[4]  
  
                except:  
                    print("Error to listen to Time Msg!")  
                    sys.exit(0)
```



## 2.基础接口使用

### 2.9 时间戳—实验流程

- 时间戳的例子见RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\6.ReadTimeStmp的 ReadTimeStmp.py，先运行 ReadTimeStmp.bat脚本开启仿真，待CopterSim显示3DFixed后，在运行ReadTimeStmpVip.py或者 ReadTimeStmpMav.py都可以订阅得到时间戳。
- 关键代码：PX4MavCtrlV4和VisionCaptureApi接口都实现了时间戳监听函数，先是在jsonLoad中调用 StartTimeStmplisten()开启时间戳监听，vis.rflyStartStmp 是CopterSim启（TargetCopter对应的飞机）时，py所在电脑的时间（系统时间或ROS时间），vis.timeStmp是从 CopterSim启动到当前数据生成的时间，vis.imgStmp是图像真实时间戳





## 2.基础接口使用

### 2.9 时间戳—实验效果

测试结果如下图，打开ReadTimeStmp.bat，开启一个飞机的软件在环仿真，Python运行ReadTimeStmp.py订阅得到时间戳。

A screenshot of a terminal window titled '终端'. The window shows the output of a Python script named 'ReadTimeStmp.py'. The output text is:

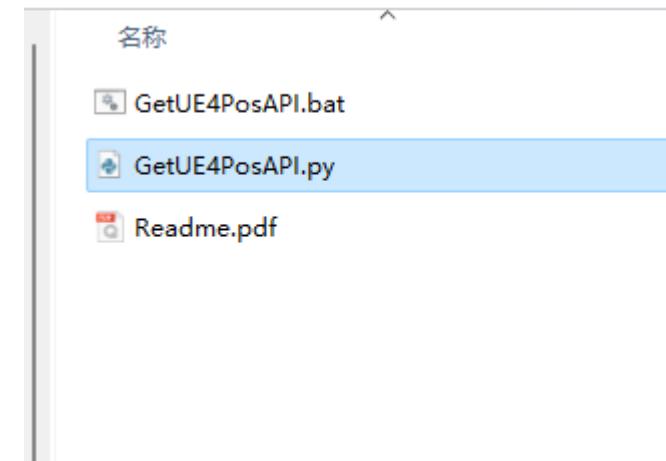
```
62  
问题 输出 编译控制台 终端 端口  
  
Got 1 vision sensors from json  
Start lisening to timeStmp Msg  
Got time msg from CopterSim # 1  
CopterSim running on this PC  
Got CopterSim time Data for img  
Sensor req success from UE4.  
Got start time for SeqID #Start lisening to IMU Msg 0  
  
Start Image Reciver  
Got CopterSim IMU Msg!  
Data for # 0  
rflyStartStamp 1703507811.055  
timeStamp [104.36]  
imgStamp [1.70350792e+09]  
imu.rflyStartStamp 1703507811.055  
imu.timestamp 104.47
```



## 2.基础接口使用

### 2.10 获取UE接口—接口介绍

- 本接口文件为Python获取UE内所有动态创建物体位置、碰撞数据的接口。
- ue.sendUE4Cmd('RflyReqVehicleData 1')这个接口用于请求UE4返回收到的所有飞机（障碍物）的数据。注意：为了降低宽带消耗，只有飞机数据发生变化时，才会有数据回传。这意味着，本命令之前发送的障碍物创建的物体，不会被传出来。因此，本命令需要在创建飞机之前发送。
- ue.initUE4MsgRec() 是python监听UE4发送的所有飞机状态数据的函数，调用本函数后，可以收到飞机数据。





## 2.基础接口使用

### 2.10 获取UE接口—接口介绍

- 飞机的数据会存储在列表inReqUpdateVect（布尔型，是否更新标志）和inReqVect列表（reqVeCrashData结构体，存储碰撞数据）中。这个列表的长度是可变的，每一位数据为一个结构体reqVeCrashData，定义如右图所示：
- ue.getUE4Pos(TargetCopterID)接口，会搜索列表中是否有copterID的飞机，并将位置输出。输出格式为4维向量，前三维是飞机的位置xyz向量，最后一维是是否有飞机在列表中的bool变量。
- ue.getUE4Data(TargetCopterID)可以获取当前飞机的数据结构体reqVeCrashData。

```
# struct reqVeCrashData {  
#     int checksum; //数据包校验码1234567897  
#     int copterID; //当前飞机的ID号  
#     int vehicleType; //当前飞机的样式  
#     int CrashType; //碰撞物体类型, -2表示地面, -1表示场景静态物体, 0表示无碰撞, 1以上表示被碰飞机的ID号  
#     double runnedTime; //当前飞机的时间戳  
#     float VelE[3]; //当前飞机的速度  
#     float PosE[3]; //当前飞机的位置  
#     float CrashPos[3]; //碰撞点的坐标  
#     float targetPos[3]; //被碰物体的中心坐标  
#     float AngEuler[3]; //当前飞机的欧拉角  
#     float MotorRPMS[8]; //当前飞机的电机转速  
#     float ray[6]; //飞机的前后左右上下扫描线  
#     char CrashedName[16]; //被碰物体的名字  
# } 41d29f20s
```



## 2. 基础接口使用

## 2.10 获得UE接口—实验流程与效果

- 获得UE接口的例子见 RflySimAPIs\8.RflySimVision\0.ApiExps\1-UsageAPI\4.RflySim3DAPI\1.RflySim3DPosGet，先运行 GetUE4PosAPI.bat 脚本开启仿真，待 CopterSim 显示 3DFixed 后，再用 VSCold 通过调试单步执行的方式运行 GetUE4PosAPI.py
  - 运行后效果如右图所示，各步骤可以获取不同的接口，包括所有动态创建物体位置，碰撞数据的接口。





# 大纲

---

1. 总体介绍
  2. 基础接口使用
  3. 视觉控制例子
  4. 视觉AI进阶
  5. 分布式视觉仿真
-



### 3.视觉控制例子

#### 3.1 无人机撞击小球实验—例程介绍

- 在Windows资源管理器中，打开并进入“RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e3\_ShootBall”文件夹，其中的内容如下图。
- “ShootBall3.py”是本例程的主Python程序，“ShootBall3HITL.bat”是快速启动硬件在环仿真的脚本，“ShootBall3SITL.bat”是快速启动软件在环仿真的脚本。如右下图所示，后两者相对于桌面的S/HITLRun快捷方式的区别在于：“UE4\_MAP”地图场景变量选择了用于视觉的平坦草地场景“VisionRingBlank”；其次，“UDPSIMMODE”通信UDP模式选择了“Mavlink\_Full”模式，便于与Python通信；最后打开1个RflySim3D窗口。

```
REM Set the map, use index or name of the map on CopterSim
REM e.g., UE4_MAP=1 equals to UE4_MAP=Grasslands
SET UE4_MAP=VisionRingBlank

REM Set UDP data mode; 0: UDP_FULL, 1: UDP_Simple, 2: Mavlink
REM e.g., UDPSIMMODE=0 equals to UDPSIMMODE=UDP_Simple
SET UDPSIMMODE=2
```



### 3.视觉控制例子

#### 3.1 无人机撞击小球实验—代码解析

- 用VS Code打开“ShootBall3.py”文件，查看源码和注释如下。

```
1 # import required libraries
2 import time
3 import numpy as np
4 import cv2
5 import sys
6
7 # import RflySim APIs
8 import PX4MavCtrlV4 as PX4MavCtrl 1.导入依赖库
9 import VisionCaptureApi
10 import UE4CtrlAPI
11 ue = UE4CtrlAPI.UE4CtrlAPI()
12
13 vis = VisionCaptureApi.VisionCaptureApi() 3.新建取图接口实例
14
15 # VisionCaptureApi 中的配置函数
16 vis.jsonLoad() # 加载Config.json中的传感器配置文件 4.加载json传感器参数列表
17
18 isSuss = vis.sendReqToUE4() # 向RflySim3D发送取图请求，并验证 5.向UE4发送取图请求
19 if not isSuss: # 如果请求取图失败，则退出
20     sys.exit(0) 6.开始读取图片，并存储在Img列表
21 vis.startImgCap() # 开启共享内存取图
22
23 # Send command to UE4 Window 1 to change resolution
24 ue.sendUE4Cmd('r.setres 720x405w',0) # 设置UE4窗口分辨率，注意本窗口仅限于显示
25 ue.sendUE4Cmd('t.MaxFPS 30',0) # 设置UE4最大刷新频率，同时也是取图频率
26 time.sleep(2) 7.修改UE4的窗口显示分辨率
27
28 # Create MAVLink control API instance 8.创建1号飞机控制实例
29 mav = PX4MavCtrl.PX4MavCtrler(1)
30 # Init MAVLink data receiving loop
31 mav.InitMavLoop() 9.开启PX4数据监听
32
33 # create a ball, set its position and altitude, use the default red color
34 ue.sendUE4Pos(100,152,0,[3,0,-2],[0,0,0]) 10.创建一个红色球
35 time.sleep(0.5)
36
37 # Function to calculate the location and radius of red ball
38 def calc_centroid(img): 11.计算球的中心
39     """Get the centroid and area of Red in the image"""
40     low_range = np.array([0,0,80])
41     high_range = np.array([100,100,255])
42     th = cv2.inRange(img, low_range, high_range)
43     dilated = cv2.dilate(th, cv2.getStructuringElement(
44         cv2.MORPH_ELLIPSE, (3, 3)), iterations=2)
45     cv2.imshow("dilated", dilated)
46     cv2.waitKey(1)
```





### 3.视觉控制例子

#### 3.1 无人机撞击小球实验—代码解析

```
56 # Function to obtain velocity commands for Pixhawk
57 # according to the image processing results
58 def controller(p_i):
59     # if the object is not in the image, search in clockwise
60     if p_i[0] < 0 or p_i[1] < 0:
61         return [0, 0, 0, 1]
62
63     # found
64     ex = p_i[0] - width / 2
65     ey = p_i[1] - height / 2
66
67     vx = 2 if p_i[2] < max_prop*width*height else 0
68     vy = 0
69     vz = K_z * ey
70     yawrate = K_yawrate * ex
71
72     # return forward, rightward, downward, and rightward-yaw
73     # velocity control signals
74     return [vx, vy, vz, yawrate]
```

12.控制器函数，用于将图像误差  
转变为速度控制误差

```
76 # Process image to obtain vehicle velocity control command
77 def processImage():
78     global ctrl_last
79     if vis.hasData[0]:
80         img_bgr=vis.Img[0]
81         p_i = calc_centroid(img_bgr)
82         ctrl = controller(p_i)
83     else:
84         ctrl=[0,0,0,0]
85     return ctrl
86
87 # saturation function to limit the maximum velocity
88 def sat(inPwm,thres=1):
89     outPwm= inPwm
90     for i in range(len(inPwm)):    14.饱和函数
91         if inPwm[i]>thres:
92             outPwm[i] = thres
93         elif inPwm[i]<-thres:
94             outPwm[i] = -thres
95     return outPwm
```

13.图像处理函数，用于根  
据图像求解Pixhawk控制速  
度量



### 3.视觉控制例子

#### 3.1 无人机撞击小球实验—代码解析

```
97     lastTime = time.time()      15.记录开始仿真时间
98     startTime = time.time()
99     # time interval of the timer
100    timeInterval = 1/30.0 #here is 0.0333s (30Hz)
101    flag = 0                  16.设置更新频率间隔
102
103    # parameters
104    width = 640
105    height = 480
106    channel = 4                17.设置控制器参数
107    min_prop = 0.000001
108    max_prop = 0.3
109    K_z = 0.003 * 640 / height
110    K_yawrate = 0.005 * 480 / width
111
112    num=0
113    lastClock=time.time()      18.设置帧率统计参数
114
115    # Start a endless loop with 30Hz, timeInterval=1/30.0
116    ctrlLast = [0,0,0,0]
117    while True:
118        lastTime = lastTime + timeInterval
119        sleepTime = lastTime - time.time()    19.死循环，使程序一直能运行下去
120        if sleepTime > 0:
121            time.sleep(sleepTime) # sleep until the desired clock
122        else:                      20.通过休眠函数，确保每隔1/30s执行后续
123            lastTime = time.time()  代码
124            # The following code will be executed 30Hz (0.0333s)
125
126            num=num+1
127            if num%100==0:          21.统计并显示程序执行帧率
128                tiem=time.time()
129                print('MainThreadFPS: '+str(100/(tiem-lastClock)))
130                lastClock=tiem
131
132            if time.time() - startTime > 5 and flag == 0:  22.在第5秒进入模式1
133                # The following code will be executed at 5s
134                print("5s, Arm the drone")                 23.初始化外部控制Offboard模式
135                mav.initOffboard()                         解锁飞机，发送期望位置5米高
136                flag = 1
137                mav.SendMavArm(True) # Arm the drone
138                print("Arm the drone!, and fly to NED 0,0,-5")
139                mav.SendPosNED(0, 0, -5, 0) # Fly to target position [0, 0, -5], i.e
```



### 3.视觉控制例子

#### • 3.1 无人机撞击小球实验—代码解析

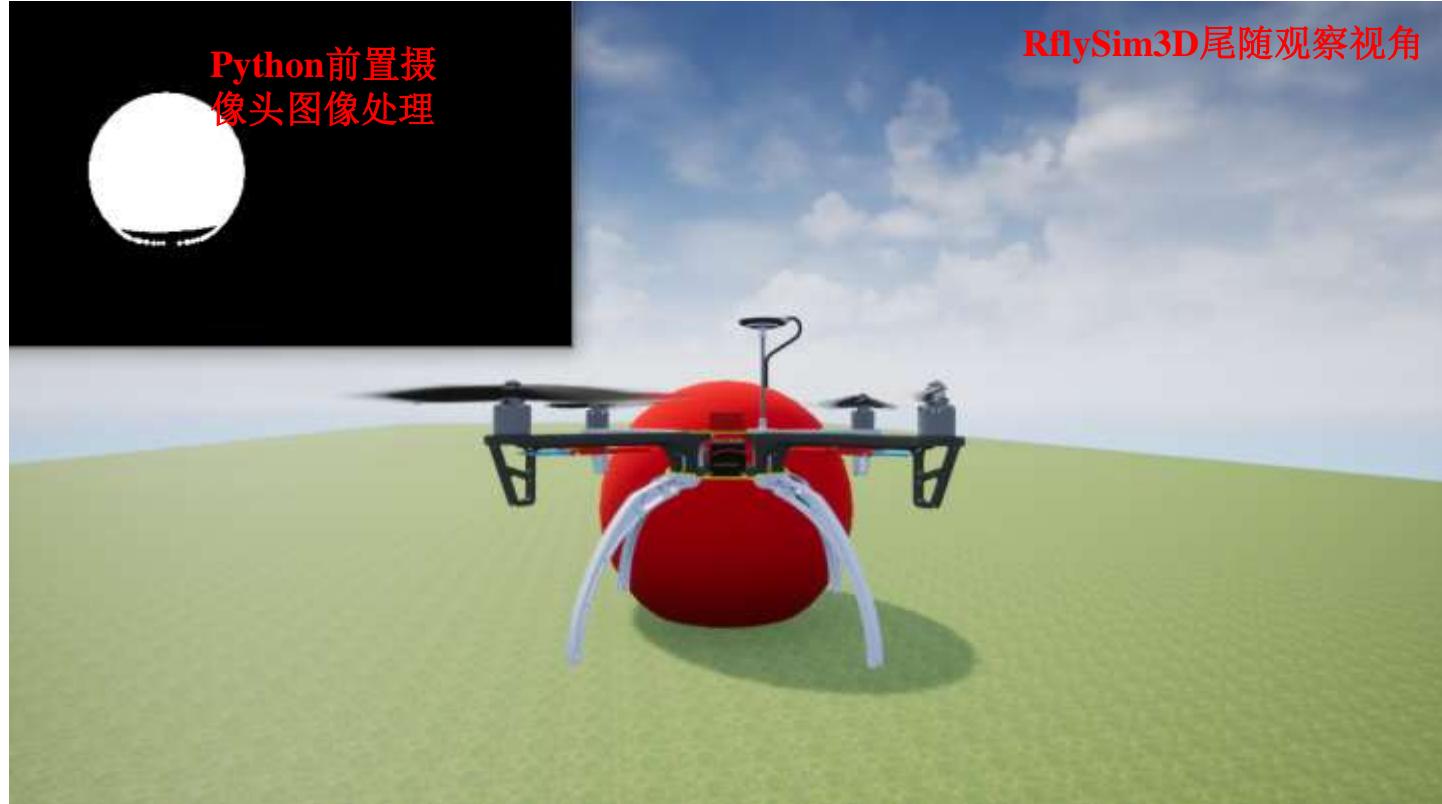
```
141 if time.time() - startTime > 15 and flag == 1:           158     if time.time() - startTime > 25 and flag == 3:
142     flag = 2                                         24.第15s进入模态2          159         ctrlNow = processImage()
143     # The following code will be executed at 15s          160         ctrl = sat(ctrlNow,5)      28.从25开始进入模态4
144     mav.SendPosNED(-30, -5, -5, 0) # Fly to target position [-30, 161
145     print("15s, fly to pos: -30,-5, -5")   25.飞机向后飞到30m后       162
146
147 if time.time() - startTime > 25 and flag == 2:           163
148     flag = 3                                         26.第25s进入模态3          164
149     # Show CV image and set the position          165
150     if vis.hasData[0]:                           166
151         img_bgr=vis.Img[0]                      167
152         cv2.imshow("dilated", img_bgr)        27.显示接收图片          168
153         cv2.waitKey(1)                         169
154         #time.sleep(0.5)                      170
155
156     print("25s, start to shoot the ball.")    171
172
173                                     # if control signals is obtained, send to Pixhawk
174                                     30.将速度控制指令发给PX4
175                                     mav.SendVelFRD(ctrl[0], ctrl[1], ctrl[2], ctrl[3])
```



### 3.视觉控制例子

#### 3.1 无人机撞击小球实验—实验效果

- 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。
- 再运行“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。



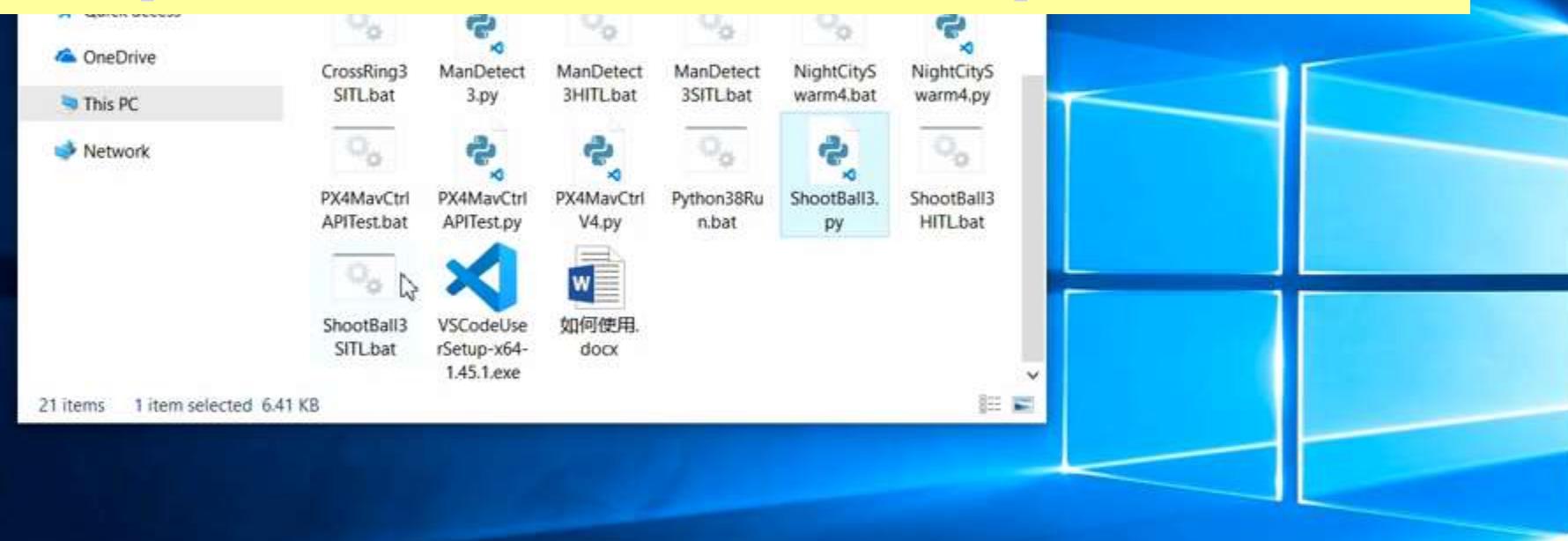
## RflySim. 多旋翼无人机进行视觉导航控制 —— 撞击小球实验

本视频观看地址：

优酷：[https://v.youku.com/v\\_show/id\\_XNDcwNjA4NTYwNA==.html](https://v.youku.com/v_show/id_XNDcwNjA4NTYwNA==.html)

YouTube：<https://youtu.be/PvxEfY7oMq4>

B站：<https://www.bilibili.com/video/BV13a411i7sH?p=13>



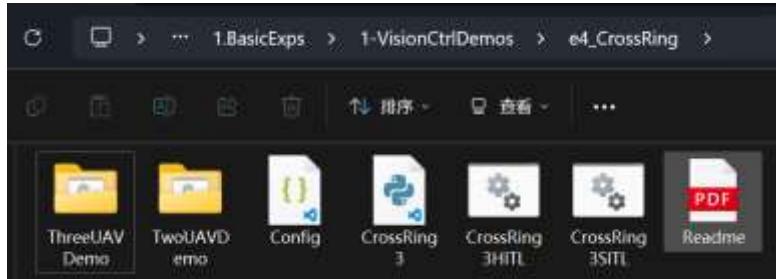
双击例程提供的**bat**脚本可以快速启动软/硬在环软件和期望数量的三维程序



### 3.视觉控制例子

#### 3.2 无人机穿环实验—例程介绍

- 在Windows资源管理器中，打开并进入“RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e4\_CrossRing”文件夹，其中的内容如下图。
- 其中，“CrossRing3.py”是本例程的主Python程序；“ShootBall3HITL.bat”和“ShootBall3SITL.bat”相对于上一个撞击小球的例子的区别在于：“UE4\_MAP”地图场景变量选择了用于视觉穿环场景“VisionRing”。注：其“UDPSIMMODE”通信UDP模式也选择了“Mavlink\_Full”模式。
- 子目录“TwoUAVDemo”和“ThreeUAVDemo”是两个分布式视觉控制的例子，生成两个或三个无人机独立处理各自视觉，在同一场景中完成独立穿环任务。





### 3.视觉控制例子

#### 3.2 无人机穿环实验—关键代码解析

```
42     def saturationYawRate(yaw_rate):
43         yr_bound = 20.0
44         if yaw_rate > yr_bound:
45             yaw_rate = yr_bound
46         if yaw_rate < -yr_bound:
47             yaw_rate = -yr_bound
48         return yaw_rate    1.偏航角速率控制
49
50     def taskChange(pos_x):
51         if pos_x < 40:
52             task = "range1"
53         elif pos_x < 70:
54             task = "range2"
55         elif pos_x < 130:    2.基于飞行距
56             task = "range3" 离的任务切换
57         elif pos_x < 140:    逻辑
58             task = "land"
59         else:
60             task = "finish"
61         return task
```

```
71     # object detect function
72     def objectDetect(task):
73         """According task to detect objects"""
74         if vis.hasData[0]:
75             img_bgr=vis.Img[0]  3.目标检测函数
76         else:
77             return -1,-1,-1
78         b,g,r = cv2.split(img_bgr)
79         img_edge = cv2.Canny(b, 50, 100)
80         if task == "range1" or task == "range2":
81             return circleDetect(img_bgr, img_edge, b)
82         else:
83             return squareDetect(img_bgr, img_edge)
84
85     # square detect for object detect function
86     def squareDetect(img_bgr, img_edge):
87         """Detect Square with PolyDP and diagonal length"""
88         # find contours
89         squares = []
90         cnts, Hierarchy = cv2.findContours(img_edge, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
91         for cnt in cnts:
92             cnt_len = cv2.arcLength(cnt, True)
93             cnt = cv2.approxPolyDP(cnt, 0.05 * cnt_len, True)
94             if len(cnt) == 4 and cv2.contourArea(cnt) > 2000 and cv2.isContourConvex(cnt):
95                 cnt = cnt.reshape(-1, 2)
96                 diag_delta = diagonal_check(cnt)
97                 if diag_delta < 0.2:
98                     squares.append(cnt)
99         cv2.drawContours( img_bgr, squares, -1, (0, 255, 0), 3)
100        cv2.imshow("img_bgr", img_bgr)    4.方形靶标检测函数
101
102        cv2.waitKey(1)
103        height, width, channel = img_bgr.shape
104        if squares:
105            return (squares[0][0][0]+squares[0][2][0])/2 - width/2, (squares[0][0][1]+squares
106            [0][2][1])/2
107        else:
108            return -1,-1,-1
```





### 3.视觉控制例子

#### 3.2 无人机穿环实验—关键代码解析

```
110 # circle detect for object detect function
111 def circleDetect(img_bgr, img_edge, img_b):
112     """Hough Circle detect"""
113     circles = cv2.HoughCircles(img_b, cv2.HOUGH_GRADIENT, 1, 20, pa
114     if circles is not None:
115         circles = np.uint16(np.around(circles))
116         obj = circles[0, 0]
117         cv2.circle(img_bgr, (obj[0], obj[1]), obj[2], (0,255,0), 2)
118         cv2.circle(img_bgr, (obj[0], obj[1]), 2, (0,255,255), 3)
119         cv2.imshow("img_bgr", img_bgr)      5.圆形靶标检测函数
120         cv2.waitKey(1)
121         height, width, channel = img_bgr.shape
122         return obj[0]-width/2, obj[1]-height/2, obj[2]
123     else:
124         return -1,-1,-1
```

```
127     # approaching Objective/ crossing rings algorithm
128     def approachObjective():
129         # 0. parameters
130         # some parameters that work:(0.03, -0.03, 1, 5.0); (0.06, -0.04, 1, 1
131         K_z = 0.004 * 640 / height
132         K_yawrate = 0.005 * 480 / width
133         task = "range1"
134         # 1. start
135         startAppTime= time.time()
136         lastTime = time.time()          6.目标接近控制函数
137         # time interval of the timer
138         timeInterval = 1/30.0 #here is 0.0333s (30Hz)
139
140         num=0
141         lastClock=time.time()
142         while (task != "finish") & (task != "land"):
143             lastTime = lastTime + timeInterval
144             sleepTime = lastTime - time.time()
145             if sleepTime > 0:
146                 time.sleep(sleepTime) # sleep until the desired clock
147             else:
148                 lastTime = time.time()
149             # The following code will be executed 30Hz (0.0333s)
```



### 3.视觉控制例子

#### 3.2 无人机穿环实验—关键代码解析

```
185 # main function
186 if __name__ == '__main__':
187     mav = PX4MavCtrl.PX4MavCtrl(1)
188     mav.InitMavLoop()
189
190     print("Simulation Start.")
191
192     print("Enter Offboard mode.")
193     time.sleep(5)
194     mav.initOffboard()
195     time.sleep(0.5)
196     mav.SendMavArm(True) # Arm the drone
197     mav.SendPosNED(0, 0, -5, 0) # Fly to target position 0,0, -5
198
199     # Show CV image and set the position
200     if vis.hasData[0]:
201         img_bgr=vis.Img[0]
202         cv2.imshow("img_bgr", img_bgr)
203         cv2.waitKey(1)
204         #time.sleep(0.5)
205
206         time.sleep(5)
207
208     # start crossing ring task
209     approachObjective()
```

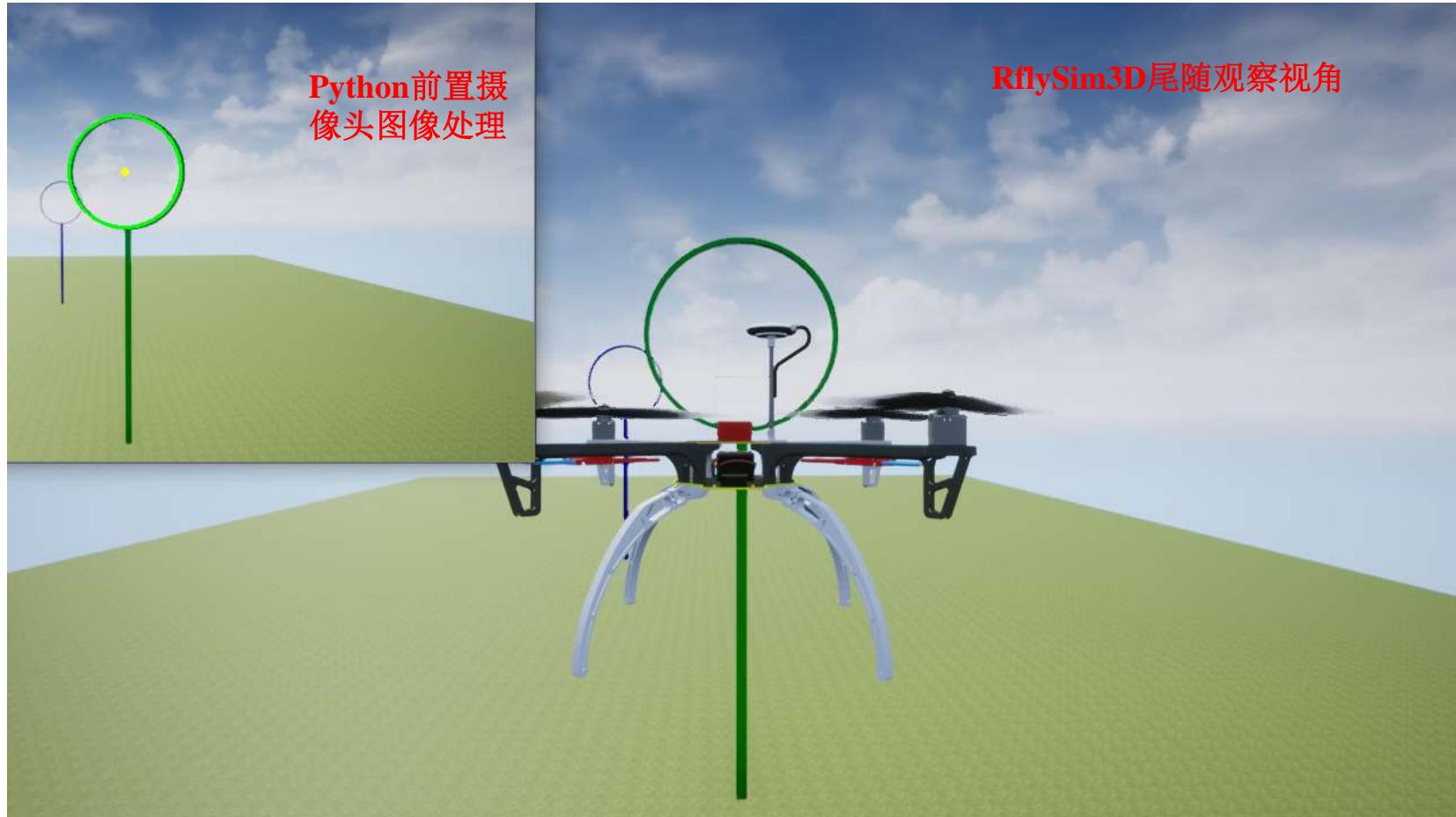
7.主函数。包含  
MAVLink接口初始化  
和RflySim3D视角控  
制



### 3.视觉控制例子

#### 3.2 无人机穿环实验—运行效果

- 双击运行“CrossRing3SITL.bat”文件开启软件在环仿真系统，再运行“CrossRing3.py”程序。
- 飞机起飞后按照顺序穿过三个环，最后自动降落。
- 若要使用硬件在环仿真，设置好飞控后，运行“CrossRing3HITL.bat”脚本，并输入飞控串口号，来开启硬件在环仿真系统。



## RflySim. 多旋翼无人机进行视觉导航控制 —— 多旋翼穿环实验

本视频观看地址：

优酷：[https://v.youku.com/v\\_show/id\\_XNDcwNjA4NTYwNA==.html](https://v.youku.com/v_show/id_XNDcwNjA4NTYwNA==.html)

YouTube：<https://youtu.be/PvxEfY7oMq4>

B站：<https://www.bilibili.com/video/BV13a411i7sH?p=13&t=53.8>



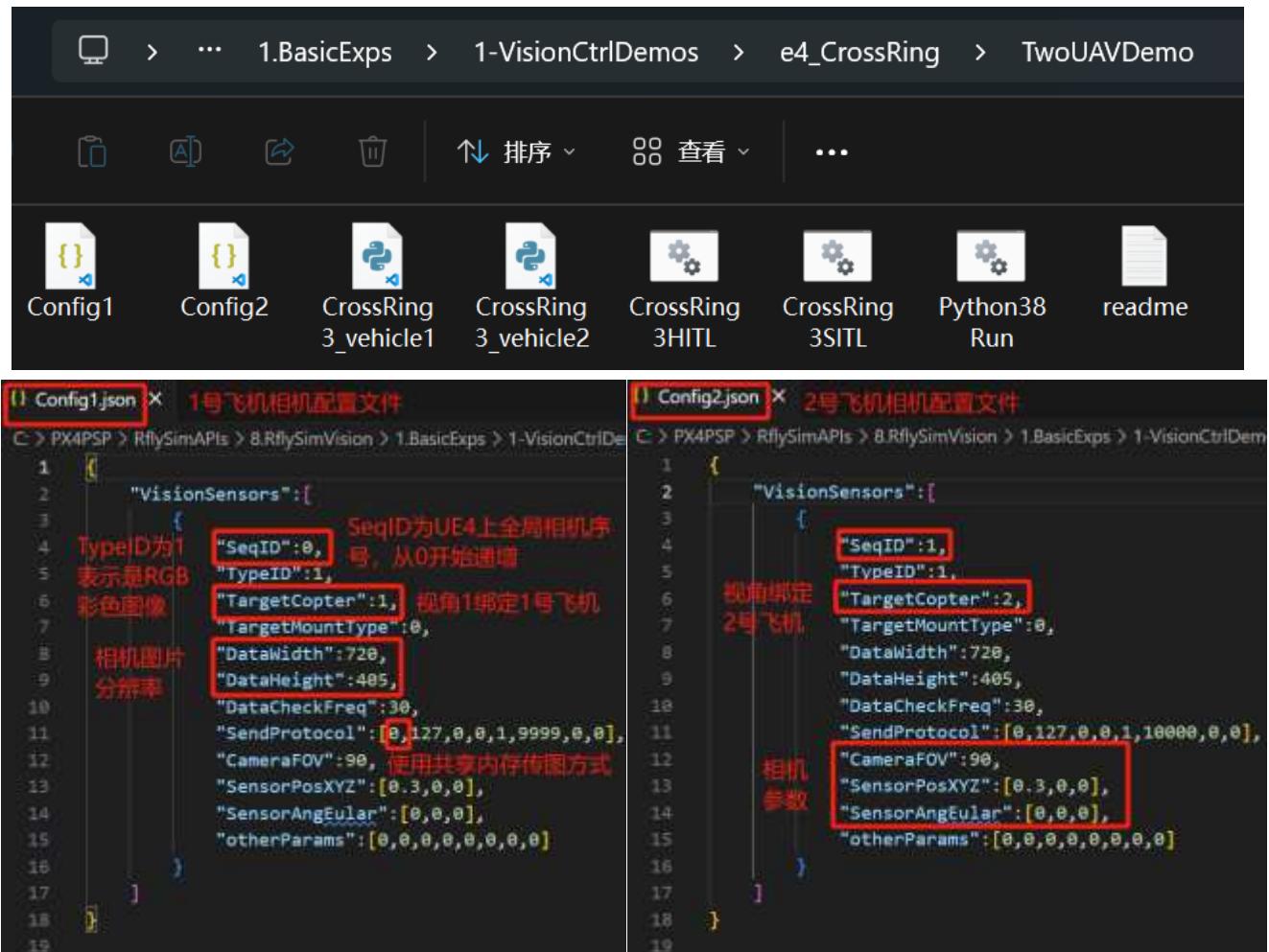
创建一个前置摄像头视角和一个尾随观察视角



### 3. 视觉控制例子

### 3.3 双无人机分布式控制—例程介绍

- “RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e4\_CrossRing\TwoUAVDemo”如右图所示。
  - 相比于单机例程，这里有两个Json文件和两个Python控制主程序，分别对应了两个飞机的图像和控制。
  - 两个Json文件对应的相机的参数区别主要在于：
    - 1) SeqID不同，用于区分图像内存区块；2) TargetCopter绑定的飞机不同，分别绑定在1号和2号飞机上。





### 3.视觉控制例子

#### 3.3 双无人机分布式控制—关键代码解析

- 1号飞机的Python控制程序为“CrossRing3\_vehicle1.py”，而2号飞机的为“CrossRing3\_vehicle2.py”，两程序与上一小节穿环例程基本相同。
- 两者主要差别在于视觉取图接口**VisionCaptureApi**的**jsonLoad**函数调用了不同的Json文件路径；其次，无人机控制接口**PX4MavCtrler**配置了不同的CopterSim通信端口（对应不同飞机）。两者都是取**json**定义相机列表中第一个相机图像，因此都用**Img[0]**读图。

```
14 vis = VisionCaptureApi.VisionCaptureApi()
15 # VisionCaptureApi 中的配置函数
16 vis.jsonLoad(0,'Config1.json') # 使用共享内
17 mav = PX4MavCtrl.PX4MavCtrler(1) #对应1号飞
77 # object detect function
78 def objectDetect(task):
79     """According task to detect objects"""
80     if vis.hasData[0]:
81         img_bgr=vis.Img[0]
82     else:
83         return -1,-1,-1
```

```
14 vis = VisionCaptureApi.VisionCaptureApi()
15 # VisionCaptureApi 中的配置函数
16 vis.jsonLoad [0,'Config2.json') # 使用共享内
17 mav = PX4MavCtrl.PX4MavCtrler(2) #对应2号
77 # object detect function
78 def objectDetect(task):
79     """According task to detect objects"""
80     if vis.hasData[0]:
81         img_bgr=vis.Img[0]
82     else:
83         return -1,-1,-1
```



### 3.视觉控制例子

#### 3.3 双无人机分布式控制—实验效果

- 进入“RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e4\_CrossRing\TwoUAVDemo”，双击一键运行脚本“CrossRing3SITL.bat”或“CrossRing3HITL.bat”开启两个飞机的软件/硬件在环仿真。
- 等待两飞机初始化完毕后（RflySim3D会提示），双击“Python38Run.bat”两次，打开两个Python环境；在第一个Python窗口中输入“python CrossRing3\_vehicle1.py”（先不按回车）；在第二个Python环境中输入“python CrossRing3\_vehicle2.py”（先不按回车）。
- 回到第一个Python窗口，按下回车键，运行1号飞机的视觉穿环程序；间隔几秒钟后，切换到2号Python窗口，按下回车键，开启2号飞机的视觉穿环程序。可看到飞机依次起飞，并穿环。

The screenshot shows two separate Windows Command Prompt windows, each titled with its respective Python environment number.

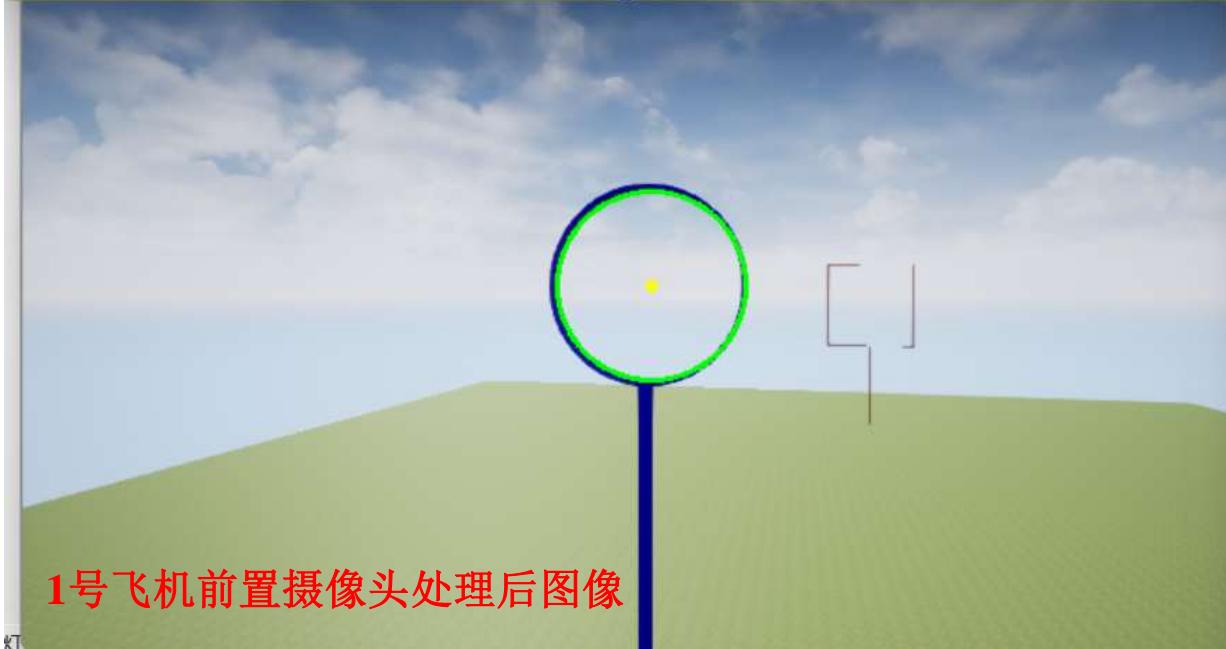
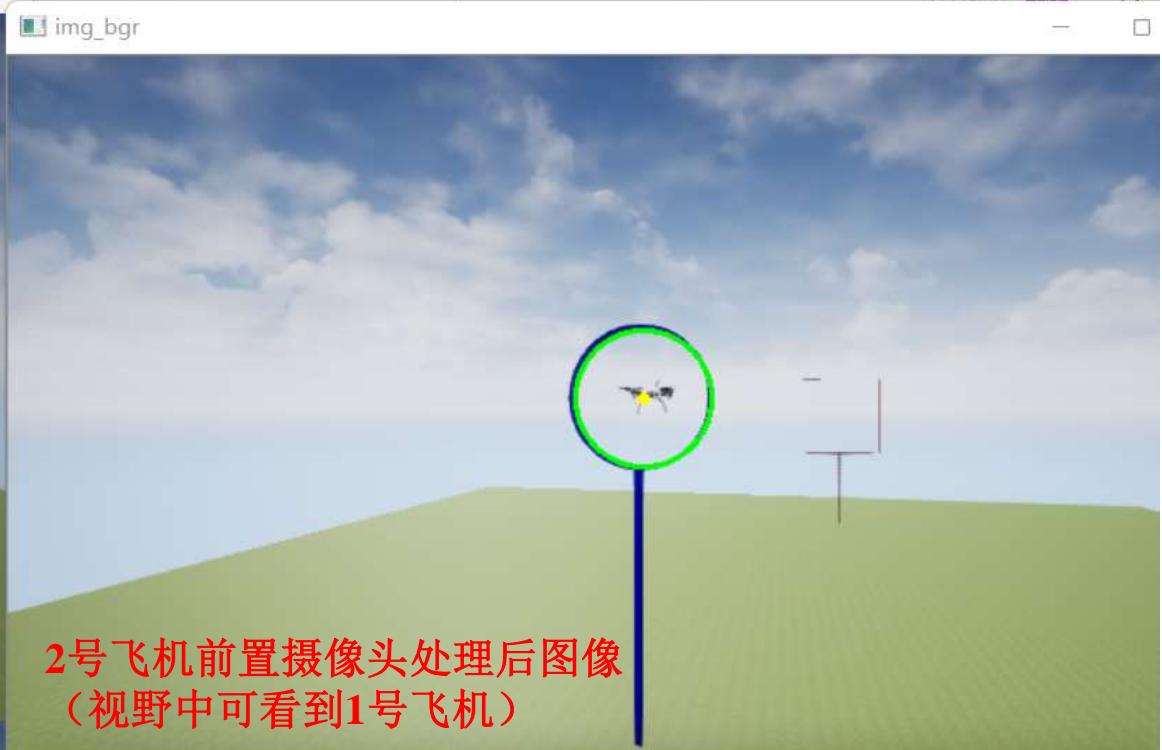
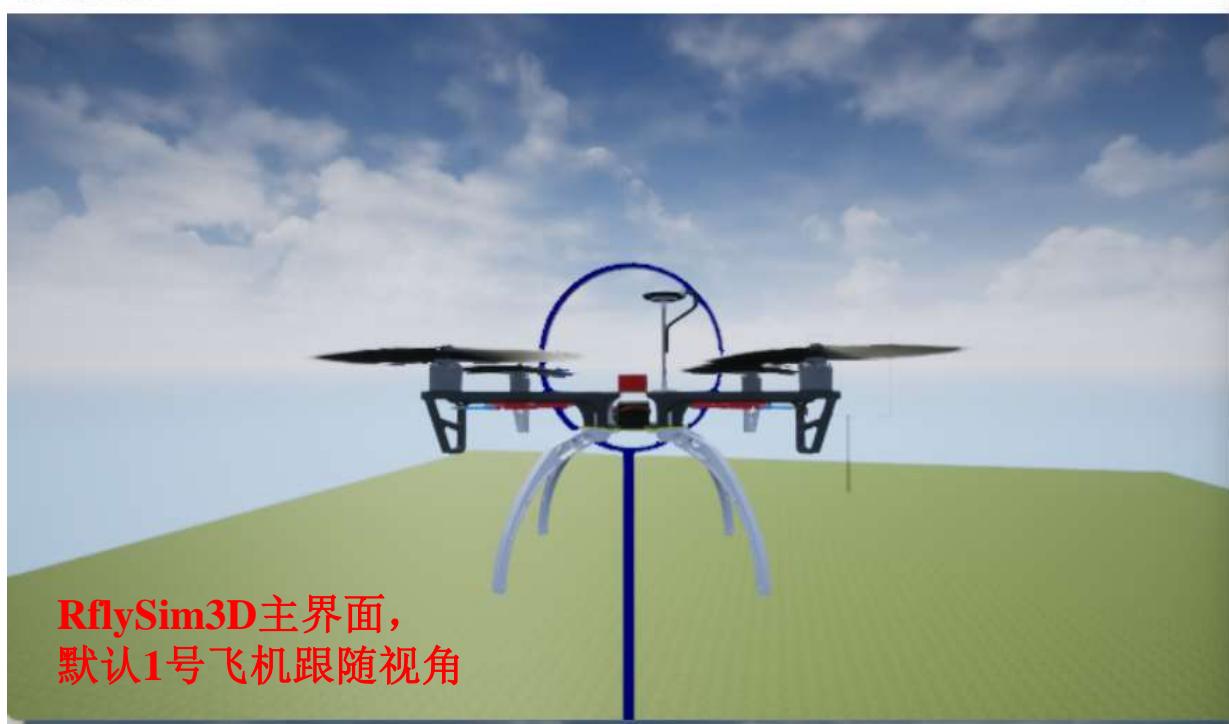
**1号Python窗口 (Top Window):**

```
C:\Windows\system32\cmd.e + × 1号Python窗口  
Python3.8 environment has been set with openCV+pymavlink+numpy+pyulog etc.  
You can use pip or pip3 command to install other libraries  
Put your python scripts 'XXX.py' into the folder 'C:\PX4PSP\RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e4_CrossRing\TwoUAVDemo'  
Use the command: 'python XXX.py' to run the script with Python  
C:\PX4PSP\RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e4_CrossRing\TwoUAVDemo>python CrossRing3_vehicle1.p  
y|
```

**2号Python窗口 (Bottom Window):**

```
C:\Windows\system32\cmd.e + × 2号Python窗口  
Python3.8 environment has been set with openCV+pymavlink+numpy+pyulog etc.  
You can use pip or pip3 command to install other libraries  
Put your python scripts 'XXX.py' into the folder 'C:\PX4PSP\RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e4_CrossRing\TwoUAVDemo'  
Use the command: 'python XXX.py' to run the script with Python  
C:\PX4PSP\RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e4_CrossRing\TwoUAVDemo>python CrossRing3_vehicle2.p  
y|
```

RflySim3D-0

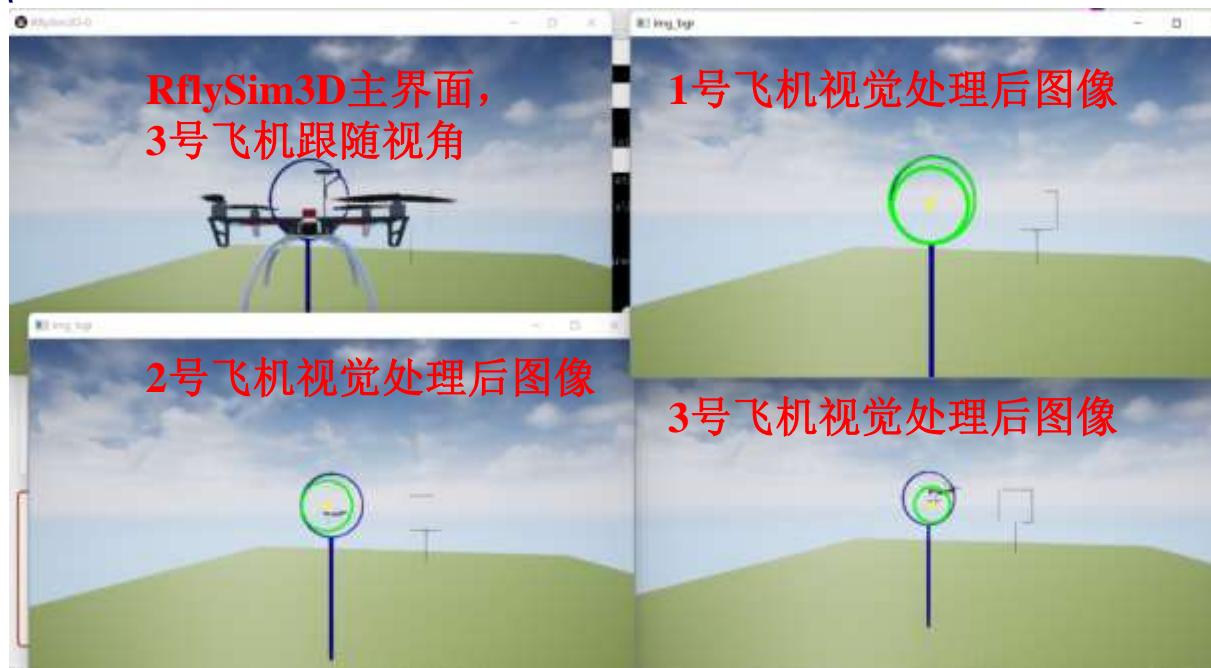




### 3.视觉控制例子

#### 3.3 双无人机分布式控制—扩展到3个飞机（限完整版，电脑配置需求高）

- 通过双无人机分布式控制的例子，我们可以很轻松地将无人机数量扩展到更多，例如三个飞机。注：免费版仅支持最多两路图像输出，因此只能做双机视觉。
- 进入“RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e4\_CrossRing\ThreeUAVDemo”
- 目录，按照同样的方法可以
- 开启三个飞机的视觉控制，效果如下图。
- 可以看到三个飞机依次起飞，后面的飞机可以看到前面的飞机在视野中。
- 由于前面飞机会遮挡圆环这种模式的圆环识别会存在波动，降低控制效果。

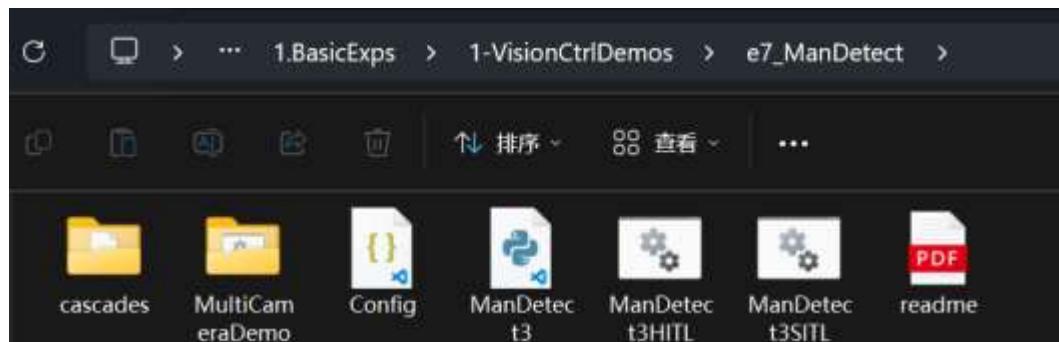




### 3.视觉控制例子

#### 3.4 双目视觉人脸识别实验—例程介绍

- 在Windows资源管理器中，打开并进入“RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e7\_ManDetect”文件夹，其中的内容如下图。
- 其中，“ManDetect3.py”是本例程的主Python程序；文件夹“cascades”内装了人脸识别的一些XML格式的特征文件；“ManDetect3HITL.bat”和“ManDetect3SITL.bat”相对于桌面快捷方式的区别在于：“UDPSIMMODE”通信UDP模式也选择了“Mavlink\_Full”模式；文件夹“MultiCameraDemo”内包含了一个进阶例子，两个飞机各自带双目相机，共四个相机图像（限完整版）。





### 3.视觉控制例子

#### 3.4 双目视觉人脸识别实验—关键源码解析

```
59 timeInterval = 1/30.0 # time interval of the timer 1.图像处理帧率30Hz
60 face_cascade=cv2.CascadeClassifier(sys.path[0]+'\cascades\haarcascade_frontalface_default.xml')
61
62 num=0 2.载入本目录的XML人脸特征库
63 lastClock=time.time()
64 while True:
65     #gImgList = sca.getCVImgList(ImgInfoList)
66     #img1=sca.getCVImg(ImgInfo1)
67     # Get the first camera view and change to gray
68
69     if vis.hasData[0]: 3.如果1号放飞机有图像
70         pic1=cv2.cvtColor(vis.Img[0], cv2.COLOR_BGR2GRAY) 4.转化为灰度图
71         faces1=face_cascade.detectMultiScale(pic1,1.3,5) # face recognition for the first camera
72         for (x,y,w,h) in faces1: 5.调用OpenCV识别函数
73             pic1=cv2.rectangle(pic1,(x,y),(x+w,y+h),(255,0,0),1) # Draw a rectangle to mark the face
74             cv2.imshow("pic1",pic1) # Show the processed image 6.识别到的人脸画框
75                         7.显示1号飞机图片
76
77     if vis.hasData[1]:
78         #img2=sca.getCVImg(ImgInfo2)
79         # Get the second camera view and change to gray
80         pic2=cv2.cvtColor(vis.Img[1], cv2.COLOR_BGR2GRAY)
81         faces2=face_cascade.detectMultiScale(pic2,1.3,5) # face recognition for the second camera
82         for (x,y,w,h) in faces2: 8.处理2号飞机图像
83             pic2=cv2.rectangle(pic2,(x,y),(x+w,y+h),(255,0,0),1)
84             cv2.imshow("pic2",pic2)
```



### 3.视觉控制例子

#### 3.4 双目视觉人脸识别实验—运行效果

- 运行“ManDetect3SITL.bat”或“ManDetect3HITL.bat”开启软/硬件在环仿真，再运行控制主程序“ManDetect3.py”即可。
- RflySim3D中生成走动的人，并设置面对飞机，飞机起飞后开启人脸识别算法，双目框选出人脸。
- 作业1：实时更新人的位置，实现人物走动的模拟，并编写飞机跟踪控制器。
- 作业2：改成前视+下视摄像头，验证跟踪+光流算法。



RflySim: 获取双目相机图像并进行人脸识别

本视频观看地址:

优酷: [https://v.youku.com/v\\_show/id\\_XNDcwNjA4NzgxMg==.html](https://v.youku.com/v_show/id_XNDcwNjA4NzgxMg==.html)

YouTube: <https://youtu.be/hm6i6UCQjCI>

B站: <https://www.bilibili.com/video/BV13a411i7sH?p=14>



实现双目面部识别和悬停飞行的效果



### 3.视觉控制例子

#### 3.4 双目视觉人脸识别实验—扩展双飞机双目（共四个相机，限完整版）

- 进入“RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e7\_ManDetect\MultiCameraDemo”目录，可查看Config.json文件（左下两图）和MultiCameraDemo.py文件关键源码解析。（右下图）

```
"VisionSensors": [
    {
        "SeqID":0,
        "TypeID":1,
        "TargetCopter":1,
        "TargetMountType":0,
        "DataWidth":640,
        "DataHeight":480,
        "DataCheckFreq":30,
        "SendProtocol": [0,127,0,0,1,9999,0,0],
        "CameraFOV":90,
        "SensorPosXYZ": [0.3,-0.15,0],
        "SensorAngEular": [0,0,0],
        "otherParams": [0,0,0,0,0,0,0]
    },
    {
        "SeqID":1,
        "TypeID":1,
        "TargetCopter":1,
        "TargetMountType":0,
        "DataWidth":640,
        "DataHeight":480,
        "DataCheckFreq":30,
        "SendProtocol": [0,127,0,0,1,10000,0,0],
        "CameraFOV":90,
        "SensorPosXYZ": [0.3,0.15,0],
        "SensorAngEular": [0,0,0],
        "otherParams": [0,0,0,0,0,0,0]
    }
],
```

```
"SeqID":2,
        "TypeID":1,
        "TargetCopter":2,
        "TargetMountType":0,
        "DataWidth":640,
        "DataHeight":480,
        "DataCheckFreq":30,
        "SendProtocol": [0,127,0,0,1,10001,0,0],
        "CameraFOV":90,
        "SensorPosXYZ": [0.3,-0.15,0],
        "SensorAngEular": [0,0,0],
        "otherParams": [0,0,0,0,0,0,0]

},
{
        "SeqID":3,
        "TypeID":1,
        "TargetCopter":2,
        "TargetMountType":0,
        "DataWidth":640,
        "DataHeight":480,
        "DataCheckFreq":30,
        "SendProtocol": [0,127,0,0,1,10002,0,0],
        "CameraFOV":90,
        "SensorPosXYZ": [0.3,0.15,0],
        "SensorAngEular": [0,0,0],
        "otherParams": [0,0,0,0,0,0,0]
```

```
40 # send vehicle position command to create a man, when
41 # the man is located before the drone, and rotated
42 ue.sendUE4Pos(100,30,0,[1,0,-8.086],[0,0,math.pi])
43 time.sleep(1) 1.向所有窗口发送创建人物命令
44
45 # send command to change object with copterID=100
46 ue.sendUE4Cmd('RflyChange3DModel 100 16')
47 time.sleep(0.5) 2.向所有窗口发送改变人物为行走
48
49 # send command to the first RflySim3D window, to set
50 ue.sendUE4Cmd('RflyChangeViewKeyCmd B 1',0)
51 time.sleep(0.5)
52 # change the first window to size 720x485
53 ue.sendUE4Cmd('r.setres 720x485w',0)
54 time.sleep(2) 3.向0号窗口发送命令，改变视角到1
55 #号飞机，设置分辨率
56
57 #send command to the second RflySim3D window to co
58 ue.sendUE4Cmd('RflyChangeViewKeyCmd B 2',1)
59 time.sleep(0.5) 4.向1号窗口发送命令，绑定2号飞
60 ue.sendUE4Cmd('r.setres 720x485w',1)
61 time.sleep(2)
```

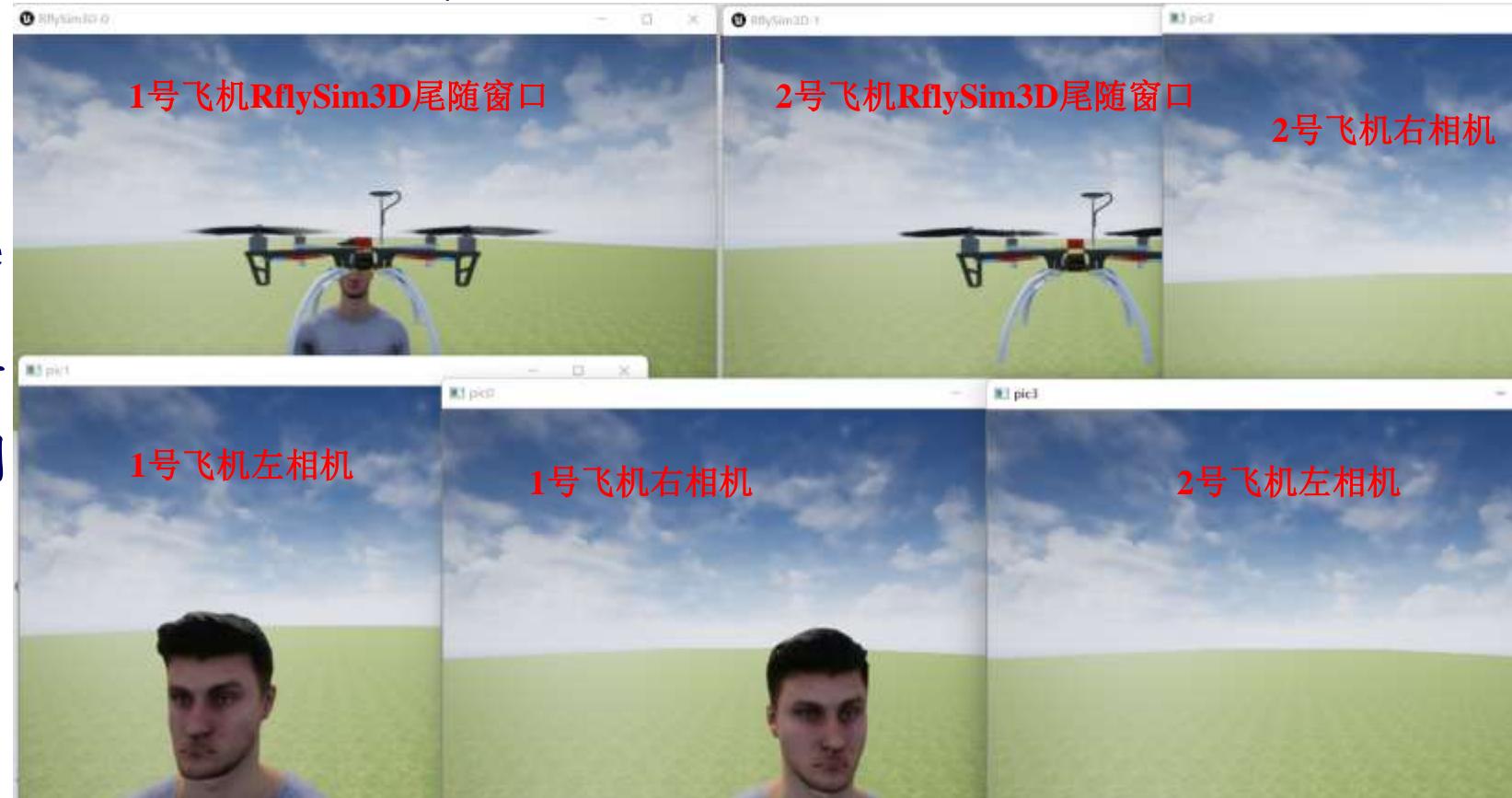




### 3.视觉控制例子

#### 3.4 双目视觉人脸识别实验—扩展双飞机双目（共四个相机，限完整版）

- 先双击“MultiCameraDemoSITL.bat”或“MultiCameraDemoHITL.bat”可以打开两个CopterSim和两个RflySim3D的双机软/硬件在环仿真。
- 然后运行“MultiCameraDemo.py”可以看到飞机起飞后，得到如右图所示的四个视角的图像。

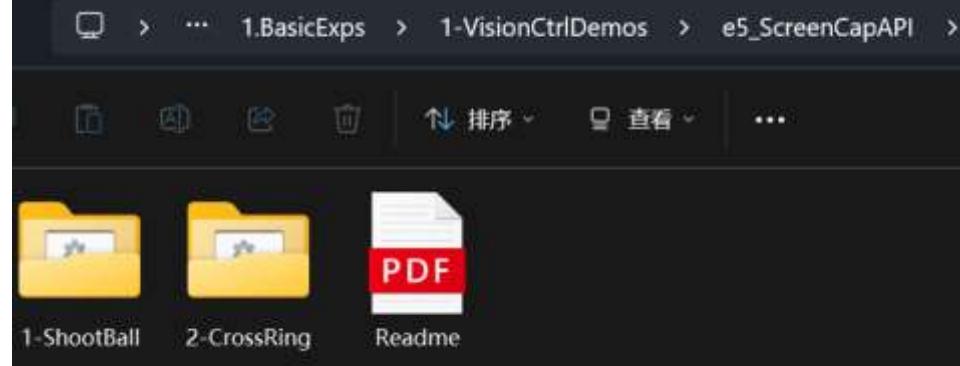




### 3.视觉控制例子

#### 3.5 屏幕截图接口—例程介绍

- 在Windows资源管理器中，打开并进入“RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e5\_ScreenCapAPI”文件夹，如下图所示，其中包含了撞击小球和穿环的例子，使用的接口为从屏幕截图的方式。
- UE4内部共享内存传图：效率高，一个窗口可实现多路传输，图像非最终效果。
- 屏幕截屏取图方式：效率低，一个窗口只能一路图像，图像为最终渲染效果。
- 总结起来，屏幕取图方式图像为最终最佳效果，而共享内存方式图像为中间渲染结果非最佳，但是屏幕取图方式效率较低，因此作为备用方案。
- 注：屏幕取图方式还可用于任意其他三维引擎（例如FlightGear，Unity）的取图。



注：推荐使用平台自带的Python38环境来运行例程，如果使用其他的Python环境，请确保安装以下组件：  
`pip3 install d3dshot pywin32`

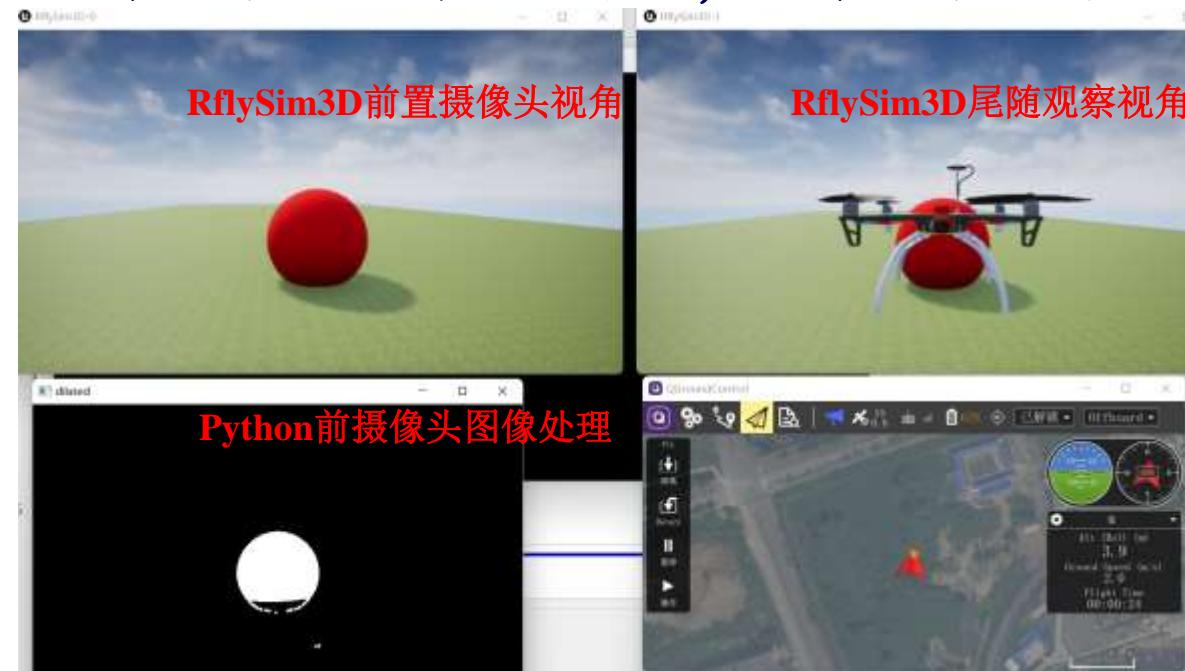


### 3.视觉控制例子

注：运行该例程时，打开SITL.bat或者HITL.bat脚本文件时请双击直接打开，不要用管理员方式运行。

#### 3.5 屏幕截图接口—撞击小球实验

- 打开“RflySimAPIs\8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e5\_ScreenCapAPI\1-ShootBall”文件夹，双击 ShootBall3SITL.bat 或 ShootBall3HITL.bat 后会打开一个CopterSim飞机的仿真闭环，同时打开两个RflySim3D窗口，一个用于显示前置摄像头，一个用于全局观察。
- 运行“ShootBall3.py”可以观察到右图的效果。
- 由于是屏幕取图的方式，左侧的前置摄像头图像必须始终保持在最前端，否则会存在遮挡现象。
- Python的图片会比窗口小，这是受到DPI设置的影响。

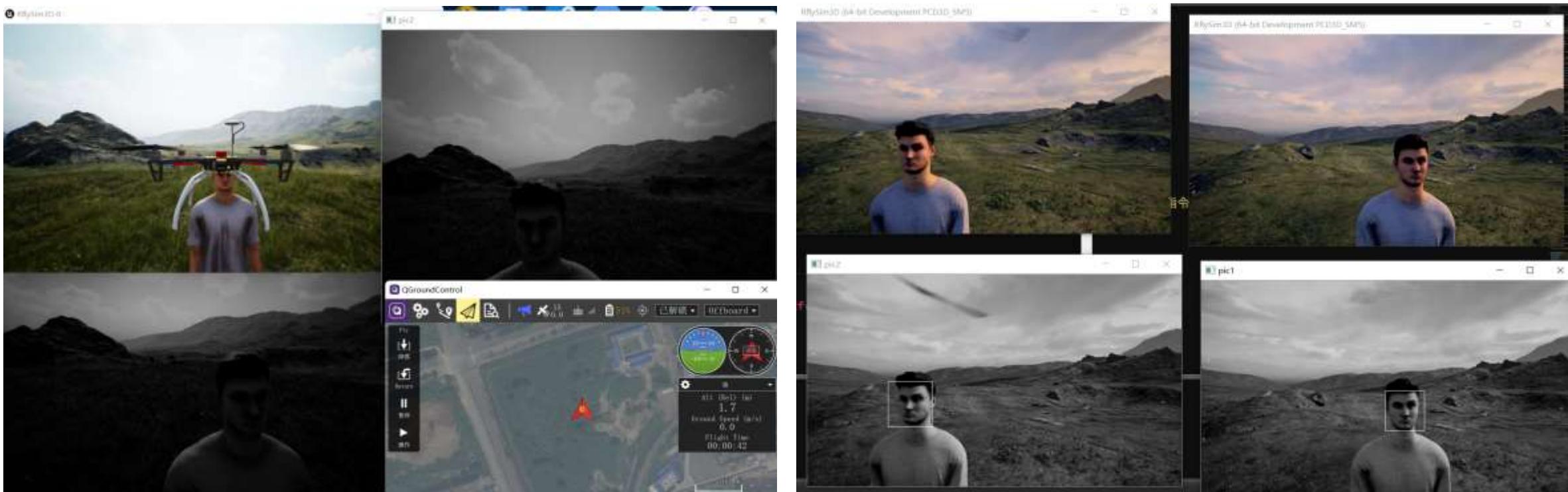




### 3.视觉控制例子

#### 3.5 屏幕截图接口—双目人脸识别实验效果对比

- 左下为UE4共享内存取图，右下为屏幕截图方式。可见屏幕取图方式的光影变化与实际窗口显示效果一致，UE4内部取图在背光时存在一定的显示局限。





# 大纲

---

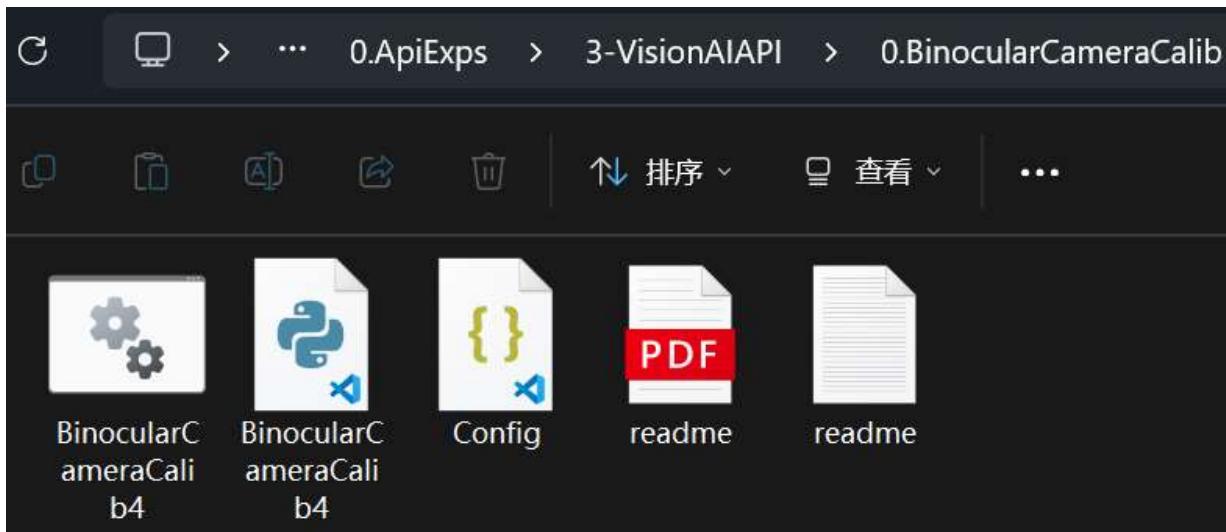
1. 总体介绍
  2. 基础接口使用
  3. 视觉控制例子
  4. 视觉AI进阶
  5. 分布式视觉仿真
-



## 4.视觉AI进阶

### 4.1 双目相机标定实验—例程介绍

- 在Windows资源管理器中，打开并进入“RflySimAPIs\8.RflySimVision\0.ApiExps\3-VisionAI API\0.BinocularCameraCalib”文件夹，其中的内容如下图。
- 其中，“BinocularCameraCalib4.py”是本例程的主Python程序；  
“BinocularCameraCalib4.bat”是一个批处理启动脚本，双击它会自动打开三个RflySim3D窗口（左右两个摄像头+尾随全局观察视角）。





## 4.视觉AI进阶

### 4.1 双目相机标定实验—核心代码解析

- 用VS Code打开“BinocularCameraCalib4.py”文件。
- 关键代码行如右图所示，本脚本可以同时获取多个窗口的图像。
- 其余代码请根据前边的讲解自行阅读学习。

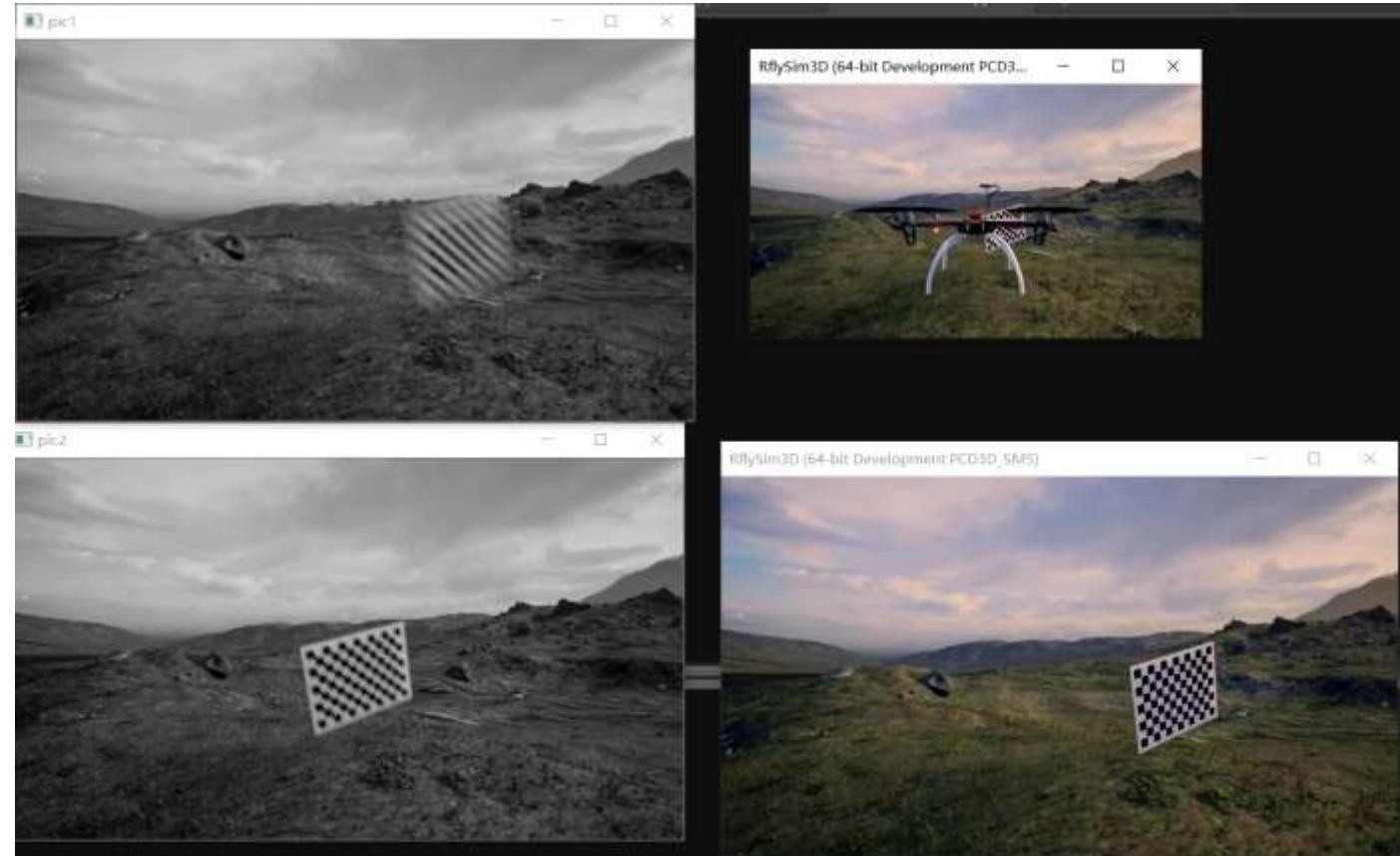
```
49 startTime = time.time()
50 lastTime = time.time()
51 timeInterval = 0.1
52 num=0
53 while True:
54     lastTime = lastTime + timeInterval
55     sleepTime = lastTime - time.time()
56     if sleepTime > 0:
57         time.sleep(sleepTime)      1.定时器时间间隔3s
58     else:                      执行头一次
59         lastTime = time.time()
60
61 num=num+1
62 # Call every 3 seconds 2.随机设置靶的位置和姿态
63 if num%30==0:
64     TargePos = [InitTargePos[0]+random.randint(0,100)/:
65     TargeAng = [InitTargeAng[0]+random.randint(-50,50),
66
67 ue.sendUE4Pos(100,40,0,TargePos,TargeAng,-1)
68 time.sleep(0.5)            3.从左右两个摄像头获取图像
69
70 if vis.hasData[0]:
71     # The following code will be executed per 3s
72     img1=vis.Img[0]
73     # Get the first camera view and change to gray
74     pic1=cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
75     cv2.imshow("pic1",pic1) # Show the processed image
76
77 if vis.hasData[1]:          4.这里可以加入图像校准算法
78     img2=vis.Img[1]
79     # Get the first camera view and change to gray
80     pic2=cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
81
82     # add image storing algorithm for MATLAB offline
83     # or add online calibration algorithm here
84     cv2.imshow("pic2",pic2) 5.显示处理后图片
85     cv2.waitKey(1)
```



## 4.视觉AI进阶

### 4.1 双目相机标定实验—实验效果

- 运行“BinocularCameraCalib4.bat”后，再运行“BinocularCameraCalib4.py”即可。
- 开启多个RflySim3D场景，新建一个飞机，配置双目位置信息，新建一个靶标，让靶标按随机规则摆放。
- 作业1：在获取到左右两个相机的图像后，实现在线的标定算法。
- 作业2：将左右两个相机的图像以图片形式存储到本地，然后用MATLAB的标定工具箱求参数。



RflySim: 获取双目相机图像并用于相机校准

本视频观看地址:

优酷: [https://v.youku.com/v\\_show/id\\_XNDcwNjA4NzgxMg==.html](https://v.youku.com/v_show/id_XNDcwNjA4NzgxMg==.html)

YouTube: <https://youtu.be/hm6i6UCQjCI>

B站: <https://www.bilibili.com/video/BV13a411i7sH?p=14&t=39.8>





## 4.视觉AI进阶

---

### 4.2 虚拟相机标定原理—介绍

- 相机标定概念：图像测量过程以及计算器视觉中，为确定空间物体某点的三维几何关系位置与其在图像中对应点之间的相互关系，必须建立相机成像的几何模型，模型的参数就是相机的参数。求解参数的过程称为相机标定。

本讲的例程源码路径：

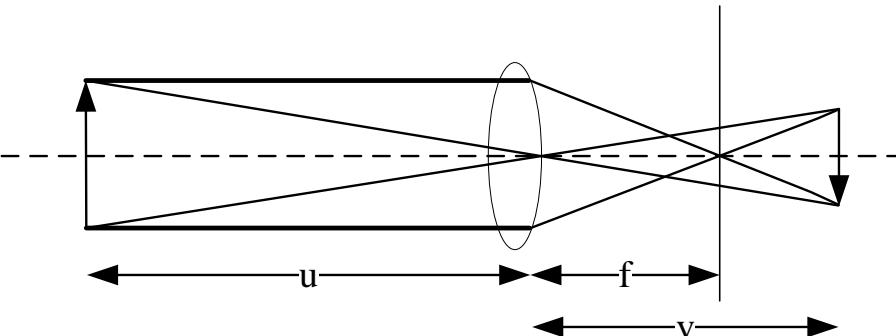
RflySimAPIs\8.RflySimVision\0.ApiEx  
ps\3-VisionAI API\2.CameraCalcDemo



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—相机模型

- 数码相机图像拍摄的过程实际上是一个光学成像的过程。相机的成像过程涉及到四个坐标系：世界坐标系、相机坐标系、图像坐标系、像素坐标系以及这四个坐标系的转换。
- 理想透视模型——针孔成像模型：相机模型是光学成像模型的简化，目前有线性模型和非线性模型两种。实际的成像系统是透镜成像的非线性模型。最基本的透镜成像原理如图所示：



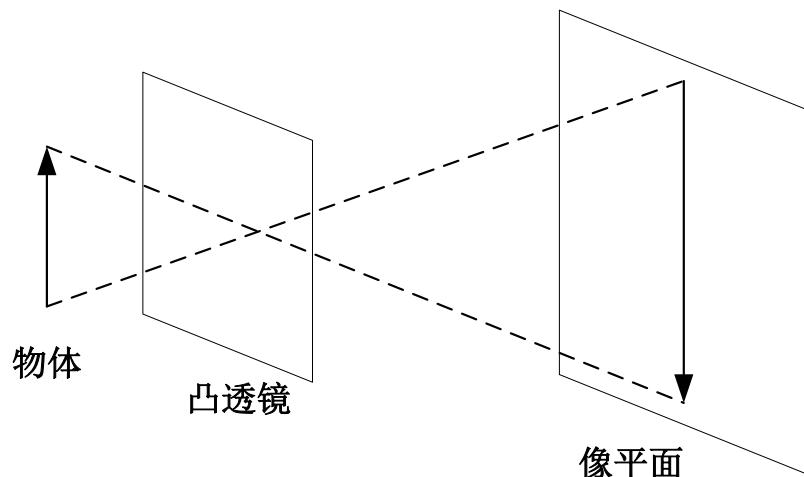
- 其中  $u$  为物距， $f$  为焦距， $v$  为相距。三者满足关系式  $\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—相机模型

- 相机的镜头是一组透镜，当平行于主光轴的光线穿过透镜时，会聚到一点上，这个点叫做焦点，焦点到透镜中心的距离叫做焦距。数码相机的镜头相当于一个凸透镜，感光元件就处在这个凸透镜的焦点附近，将焦距近似为凸透镜中心到感光元件的距离时就成为小孔成像模型。小孔成像模型如图所示。



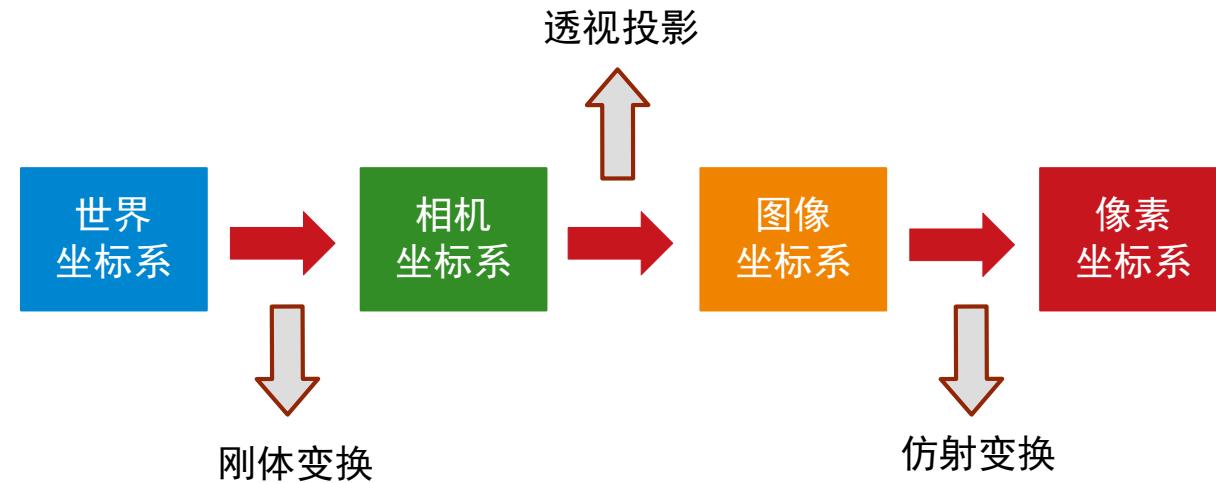
- 小孔成像模型是相机成像采用最多的模型。在此模型下，物体的空间坐标和图像坐标之间是线性的关系，因而对相机参数的求解就归结到求解线性方程组上。



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—坐标系定义

- 相机成像系统中，共包含四个坐标系：世界坐标系、相机坐标系、图像坐标系、像素坐标系。





## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—坐标系定义

- 这四个坐标系之间的转化关系为：

$$Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{dX} & -\frac{\cot\theta}{dX} & u_0 \\ 0 & \frac{1}{dY \sin\theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$

仿射变换                    透视投影                    刚体变换  
内参矩阵                    外参矩阵

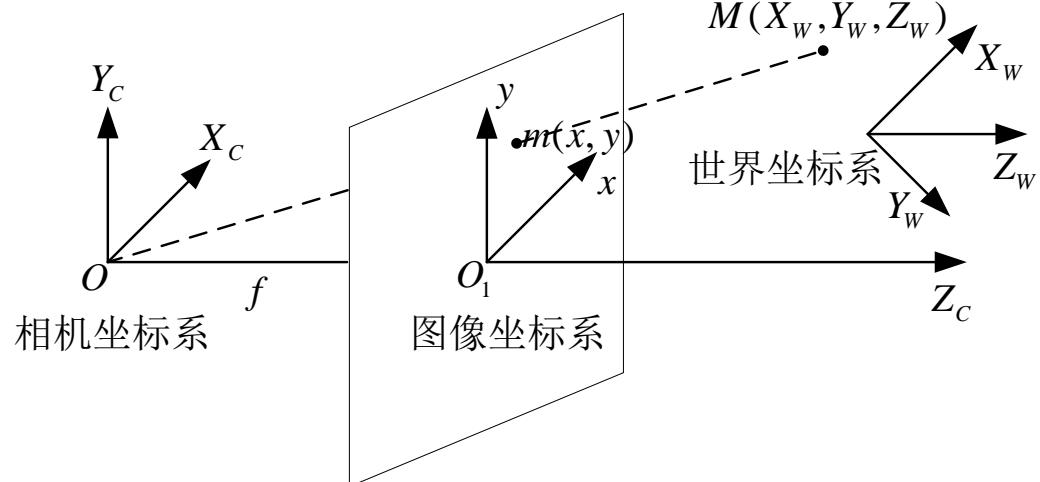
- 其中： $(U,+V,+W)$ 为在世界坐标系下一点的物理坐标， $(u,v)$ 为该点对应的在像素坐标系下的像素坐标， $Z$ 为尺度因子。



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—坐标系定义

- 四个坐标系的关系图如下图所示，其中 $M$ 为三维空间点， $m$ 为 $M$ 在图像平面投影成的像点。



- ①**世界坐标系：**是客观三维世界的绝对坐标系，也称客观坐标系。因为数码相机安放在三维空间中，我们需要世界坐标系这个基准坐标系来描述数码相机的位置，并且用它来描述安放在此三维环境中的其它任何物体的位置，用 $(X_W, Y_W, Z_W)$ 表示其坐标值。



## 4.视觉AI进阶

---

### 4.2 虚拟相机标定原理—坐标系定义

- ②相机坐标系（光心坐标系）：以相机的光心为坐标原点， $X$ 轴和 $Y$ 轴分别平行于图像坐标系的 $X$ 轴和 $Y$ 轴，相机的光轴为 $Z$ 轴，用 $(X_c, Y_c, Z_c)$ 表示其坐标值。
- ③图像坐标系：以CCD图像平面的中心为坐标原点， $X$ 轴和 $Y$ 轴分别平行于图像平面的两条垂直边，用 $(x, y)$ 表示其坐标值。图像坐标系是用物理单位（例如毫米）表示像素在图像中的位置。
- ④像素坐标系：以CCD图像平面的左上角顶点为原点， $X$ 轴和 $Y$ 轴分别平行于图像坐标系的  $x$  轴和  $y$  轴，用  $(u, v)$  表示其坐标值。数码相机采集的图像首先是形成标准电信号的形式，然后再通过模数转换变换为数字图像。每幅图像的存储形式是一个  $M \times N$  行  $N$  列的矩阵，图像中的每一个元素的数值代表的是图像点的灰度。这样的每个元素叫像素，像素坐标系就是以像素为单位的图像坐标系。



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—标定原理

- 为什么要进行相机标定呢？举个例子，当我们拿到一张图片，进行识别之后，得到的两部分之间的距离为1像素，但是这1像素究竟对应实际世界中的多少米呢？这就需要利用相机标定的结果来将像素坐标转换到物理坐标来计算距离。
- 我们要想对一个成像系统建模，进而进行相应的计算，所必须的参数就是相机的

内参矩阵：
$$\begin{pmatrix} \frac{f}{dX} & -\frac{f \cot \theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY \sin \theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$
 和相机的外参矩阵，
$$\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix}$$
，因此，相机标定的第一个目的就是获得相机的内参矩阵和外参矩阵。



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—内参矩阵和外参矩阵

- 我们将矩阵:
$$\begin{pmatrix} \frac{1}{dX} & -\frac{\cot\theta}{dX} & u_0 \\ 0 & \frac{1}{dY \sin\theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \frac{f}{dX} & -\frac{f \cot\theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY \sin\theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$
- 称为相机的内参矩阵，内参矩阵取决于相机的内部参数。其中， $f$  为像距， $dX, dY$  分别表示  $X, Y$  方向上的一个像素在相机感光板上的物理长度（即一个像素在感光板上是多少毫米） $u_0, v_0$ ，分别表示相机感光板中心在像素坐标系下的坐标。



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—内参矩阵和外参矩阵

- 在相机为理想相机的情况下，焦距、分辨率、视场角关系为： $f = \frac{\omega}{2 \tan \frac{\theta}{2}}$
- 可推出简化的内参矩阵：
$$\begin{pmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$
- 其中  $f$  为焦距， $\omega$  为分辨率的宽， $\theta$  为视场角。
- 例如，当视场角为 90 时，分辨率为 (640 480) 时，可得内参矩阵为：
$$\begin{pmatrix} 320 & 0 & 320 \\ 0 & 320 & 240 \\ 0 & 0 & 1 \end{pmatrix}$$



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—内参矩阵和外参矩阵

- 对于同一个相机，相机的内参矩阵取决于相机的内部参数，无论标定板和相机的位置关系是怎么样的，相机的内参矩阵不变。这也正是在求解内参矩阵中，我们可以利用不同的图片（标定板和相机位置关系不同）获取的矩阵H，共同求解相机内参矩阵A的原因。但是，外参矩阵反映的是标定板和相机的位置关系。对于不同的图片，标定板和相机的位置关系已经改变，此时每一张图片对应的外参矩阵都是不同的。
- 我们将矩阵： $\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix}$ 称为相机的外参矩阵，外参矩阵取决于相机坐标系和世界坐标系的相对位置， $\mathbf{R}$ 表示旋转矩阵， $\mathbf{T}$ 表示平移矢量。

- 即单点无畸变的相机成像模型如下： $Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{dX} & -\frac{f \cot \theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY \sin \theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—内参矩阵和外参矩阵

- 的每一个列向量表示世界坐标系的每一个坐标轴在相机坐标系下的指向；而是世界坐标系原点在相机坐标系下的表示。所以我们需要进行坐标转换，将相机内部坐标转换为世界坐标系。
- 此模型的旋转矩阵如下：

$$\mathbf{R}_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & \sin \gamma \\ 0 & -\sin \gamma & \cos \gamma \end{bmatrix} \quad \mathbf{R}_y(\beta) = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix} \quad \mathbf{R}_z(\alpha) = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- 由于我们已知的条件是相机的欧拉角，因此用相机坐标系推出世界坐标系的做法更为简便。因此我们使用另外一个模型：假设点  $P$  是一个三维空间中的点，其在相机坐标系下的位置为  $P_C$ ，在世界坐标系下的位置为  $P_W$ 。 $P_W$  和  $P_C$  可以通过一个变换矩阵相互转换，该变换矩阵可以细分为旋转矩阵  $\mathbf{R}'$  和平移矩阵  $\mathbf{t}$ 。其数学表达形式为： $P_W = \mathbf{R}'P_C + \mathbf{t}$



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—内参矩阵和外参矩阵

- 旋转方向与对应矩阵变换的关系：左乘右乘
- 左乘——相对于固定坐标系进行变换。（比如世界坐标系，例如绕世界坐标系旋转，先Z后Y再X，就可以得到旋转矩阵  $R = R_X R_Y R_Z$  按照顺序往左边乘。）
- 右乘——相对于自身坐标系进行变换，每变一次下一次需要以新坐标系为标准进行变换。比如第一次变换后，原 $x$ 轴的位置变为 $y$ 轴，那么下一次绕 $y$ 轴的变换，就会绕之前的 $x$ 轴变换。比如按照机体坐标系进行旋转，例如绕机体坐标系先Z后Y再X，就有，按照顺序往右乘  $R = R_Z R_Y R_X$ ，按照顺序往右乘。
- 按机体坐标轴旋转的旋转矩阵表示：

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \quad R_y(\gamma) = \begin{bmatrix} \cos\gamma & 0 & \sin\gamma \\ 0 & 1 & 0 \\ -\sin\gamma & 0 & \cos\gamma \end{bmatrix} \quad R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



## 4.视觉AI进阶

### 4.2 虚拟相机标定原理—内参矩阵和外参矩阵

- $R'$ —— $3 \times 3$ 的旋转矩阵 = 
$$\begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \gamma & 0 & \sin \gamma \\ 0 & 1 & 0 \\ -\sin \gamma & 0 & \cos \gamma \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$
, 其中  
 $(\theta, \gamma, \alpha)$  为标定板围绕相机坐标系3个轴的转角, 称为旋转向量;
- $t$ —— $(t_x, t_y, t_z)$  为平移向量。  $R'$ 与  $t$  合并称为相机外参。 完整的外参矩阵为  $\begin{pmatrix} R' & t \\ 0 & 1 \end{pmatrix}$ 。

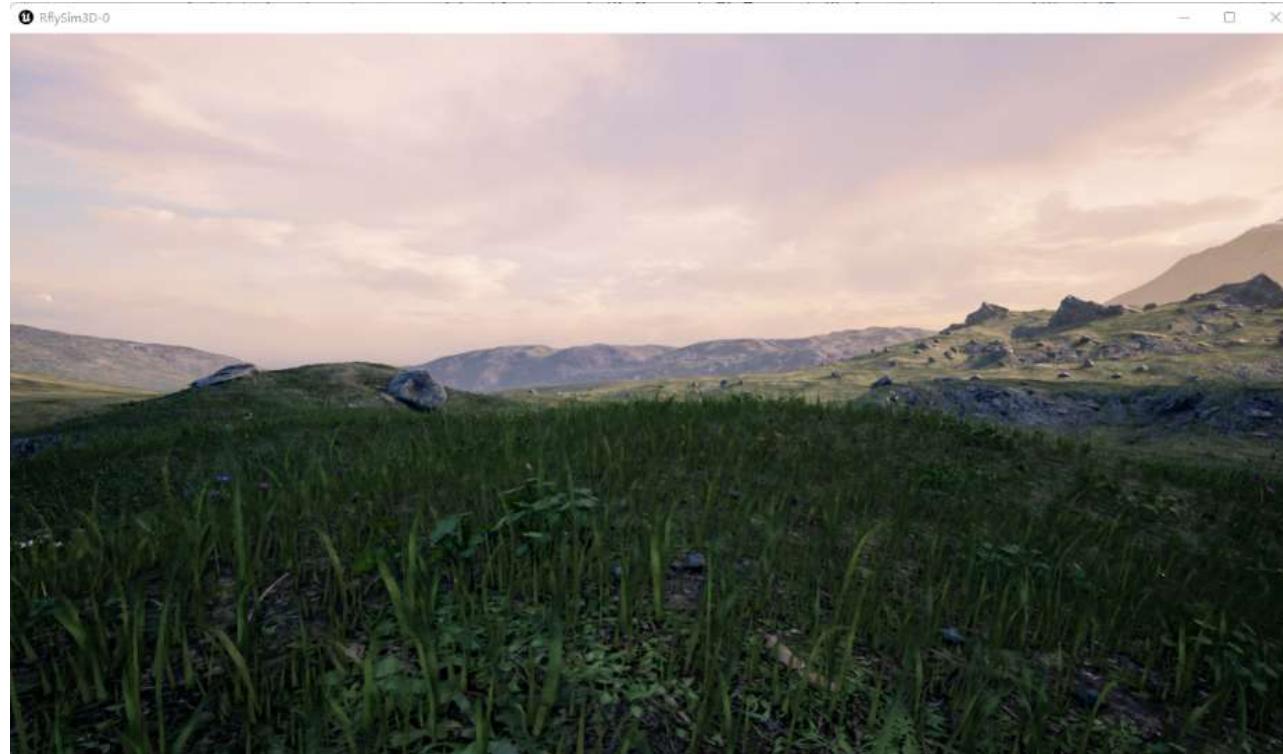


## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 1.单目相机图像存储：启动OneCameraCal.bat，得到如下图像：

例程见：[RflySimAPIs\8.RflySimVision\0.ApiExps\3-VisionAI API\2.CameraCalcDemo](#)

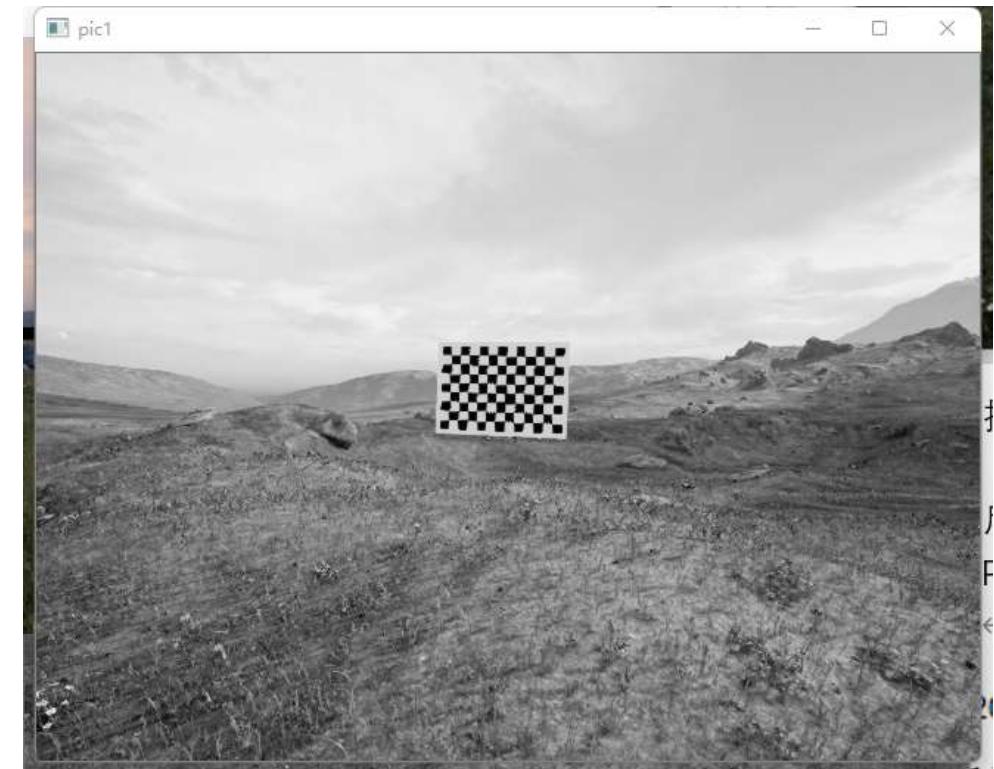




## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 2.在VS Code中运行OneCameraCal.py，界面中会出现标定板和相机拍摄的标定板图像：





## 4.视觉AI进阶

---

### 4.2 虚拟相机标定实验—例程介绍

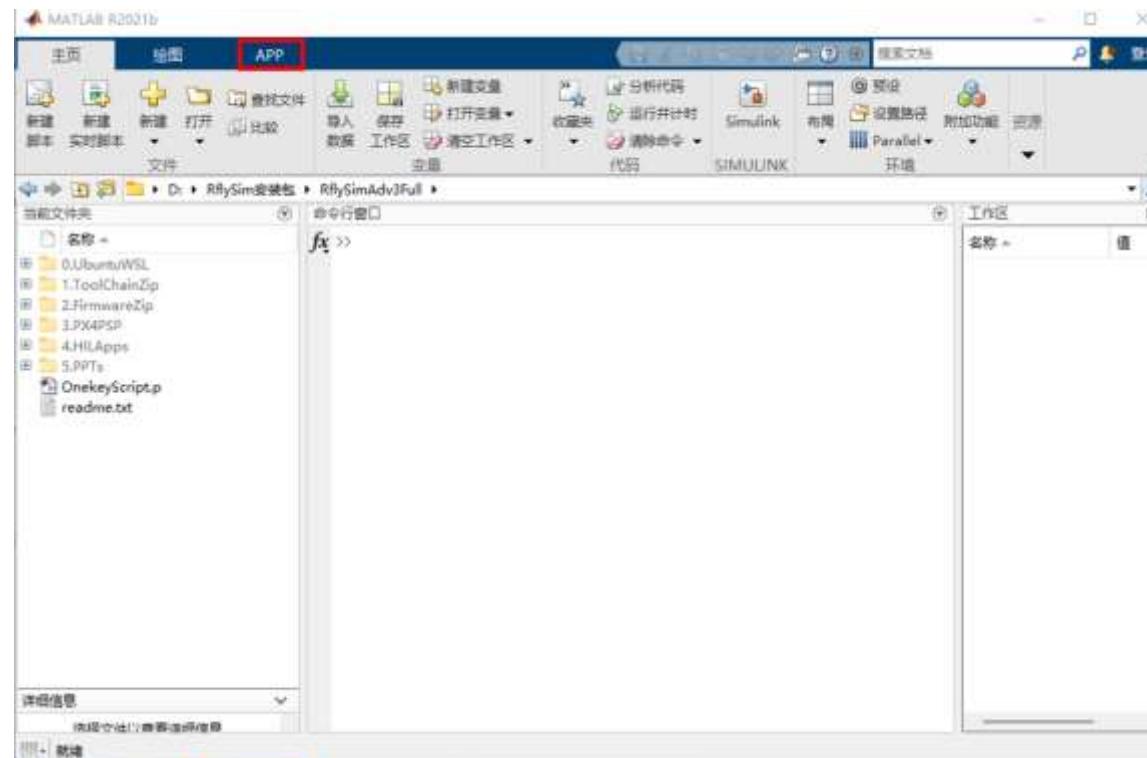
- 任务1：Matlab标定板标定
- 3.捕获的图片保存在工作路径下的一个文件夹，例如  
**PX4PSP\RflySimAPIs\8.RflySimVision\0.ApiExps\3-VisionAI API\2.CameraCalcDemo\20231226\_154138**：
- 若标定板与相机的距离太远或太近，会影响检测结果，可以修改InitTargePos 中的第一个数值来改变距离。获得三十张图片左右后就可将程序停止。



## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 4.打开matlab，点击上方的APP栏。

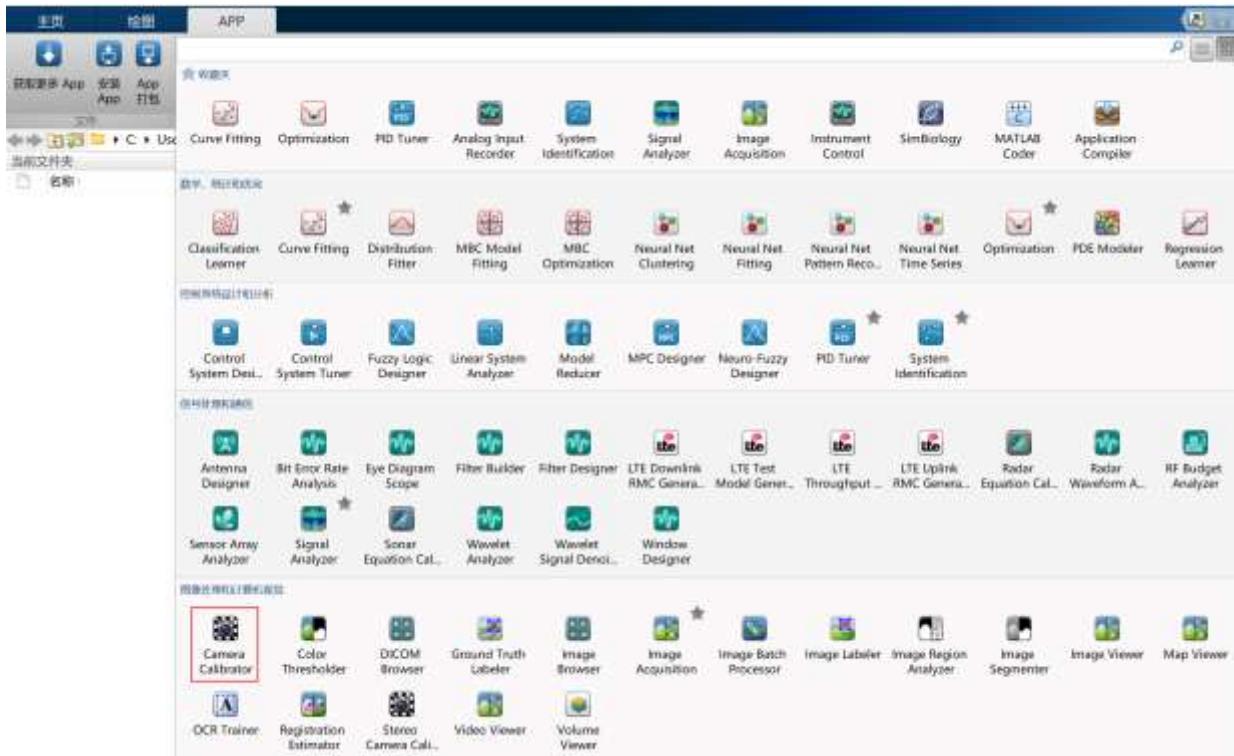




## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 5.下拉工具栏，选择Camera Calibrator工具箱。

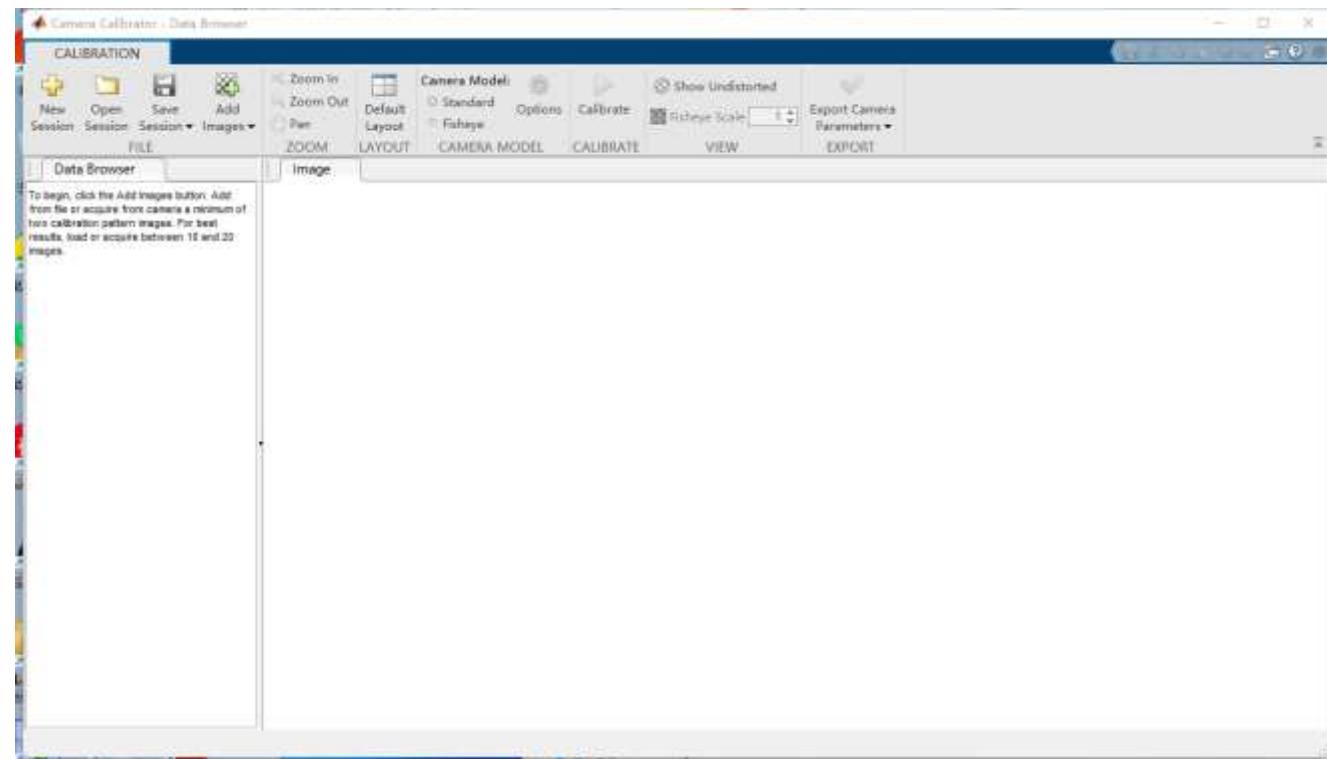




## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 6.打开工具箱后，点击Add Images，打开到我们抓取图片的位置，选取图片。选取完成后点击打开按钮。

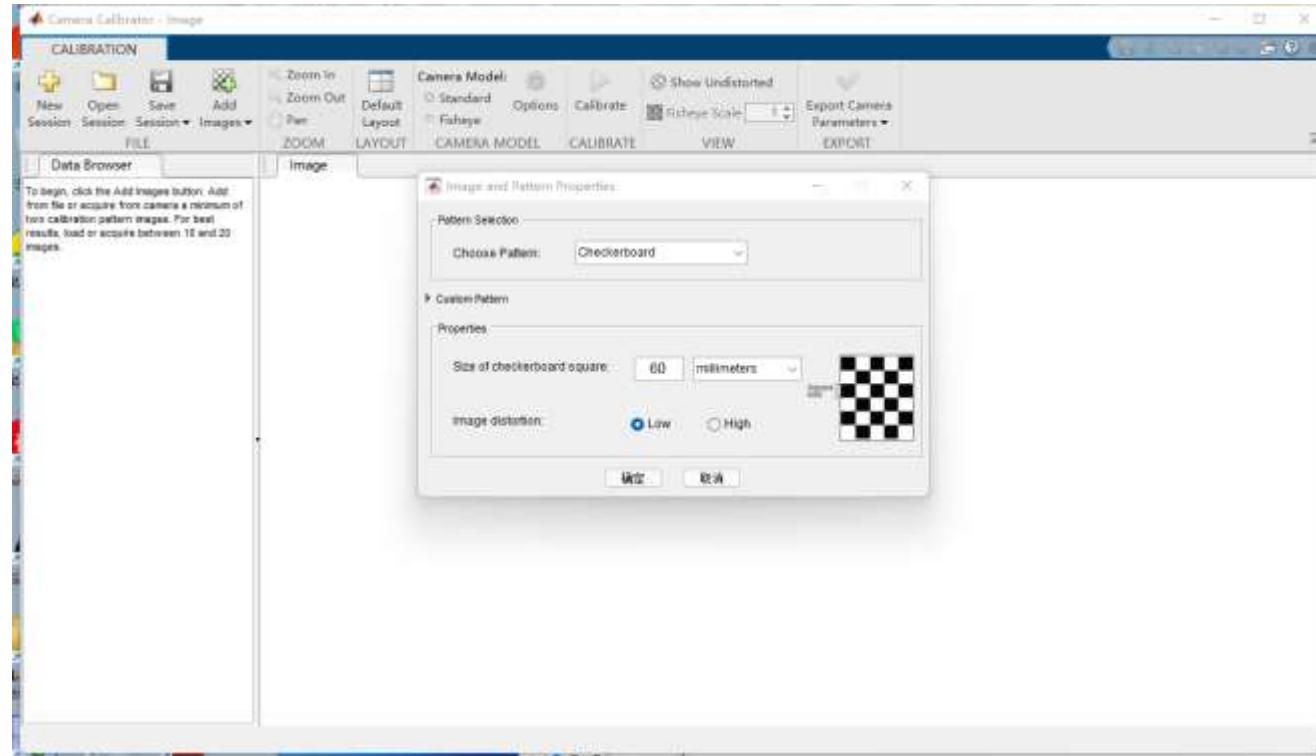




## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 7.弹出下面窗口，输入棋盘格的大小。场景中为大家提供的是60mm，点击确定。





## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 8.几秒钟后，工具箱会告诉你接受的图片，直接点确定。

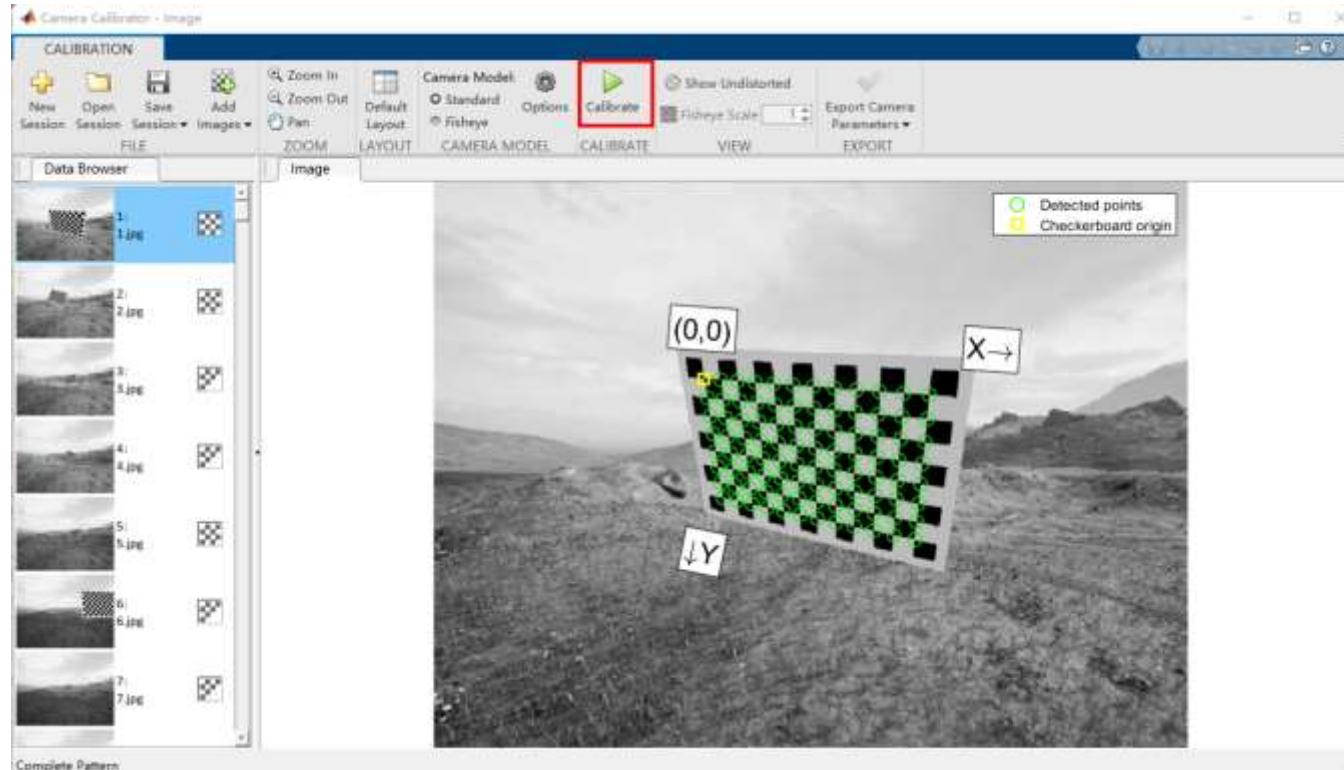




## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 9.可以大概浏览一下棋盘格提取质量，应该是都在图像上。点击Calibrate按钮开始标定。

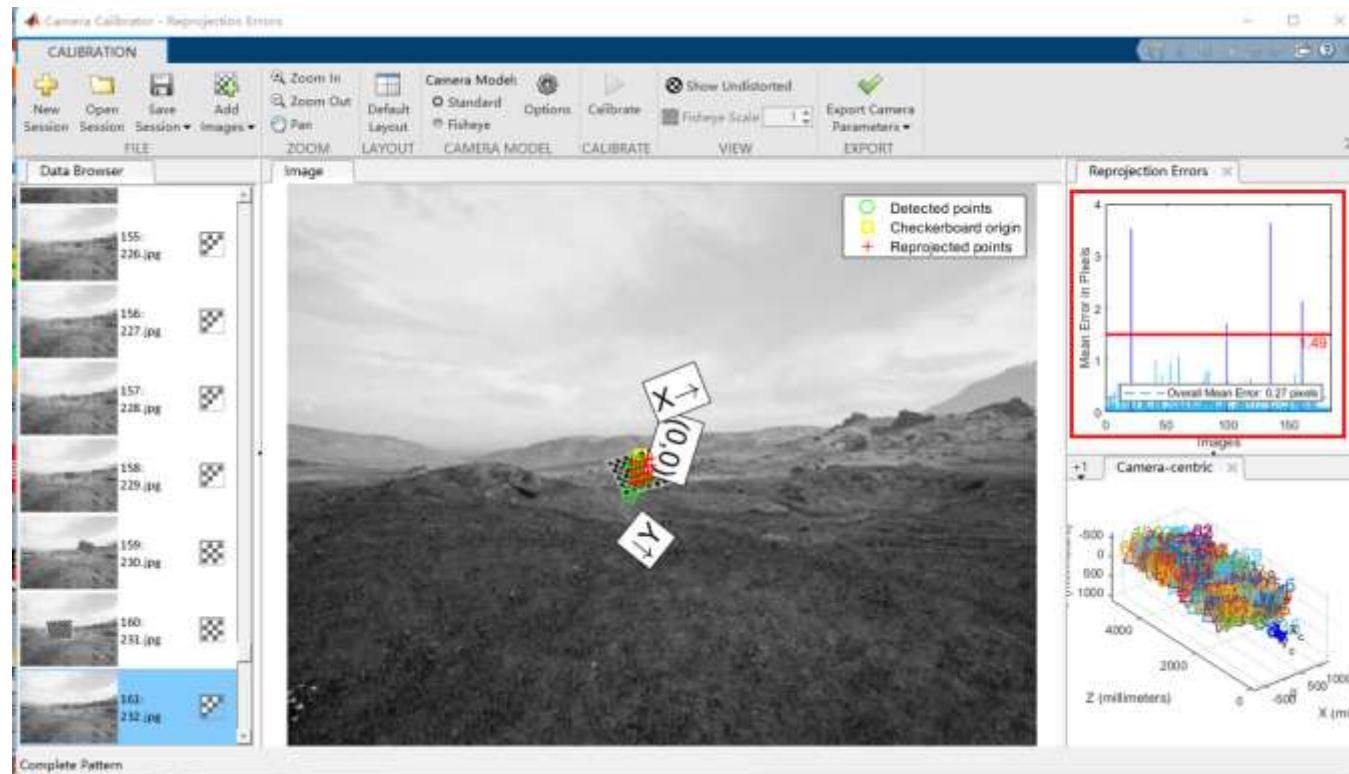




## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 10. 可以拖动红线选择反投影误差较大的图片，按Delete删除这些图片并重新标定。

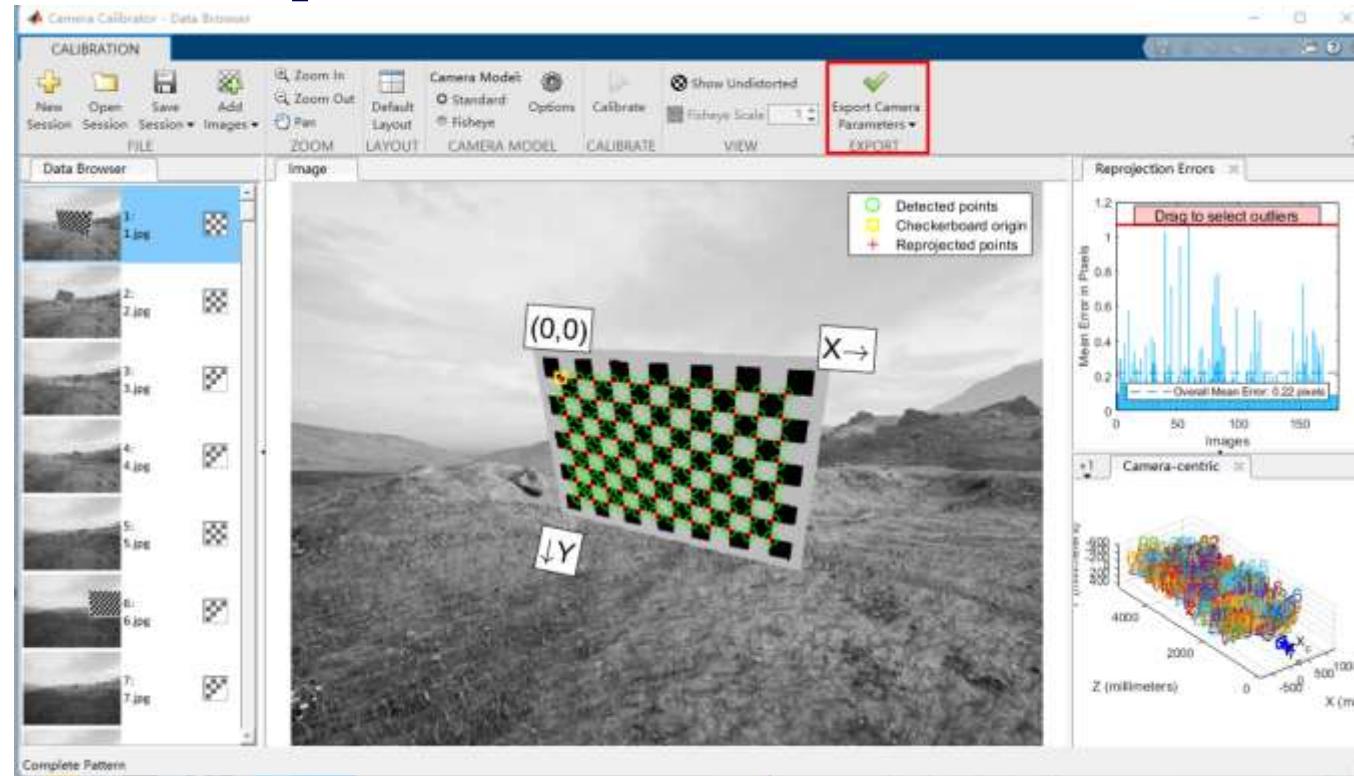




## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 11.结果满意之后点击Export导出标定的结果。

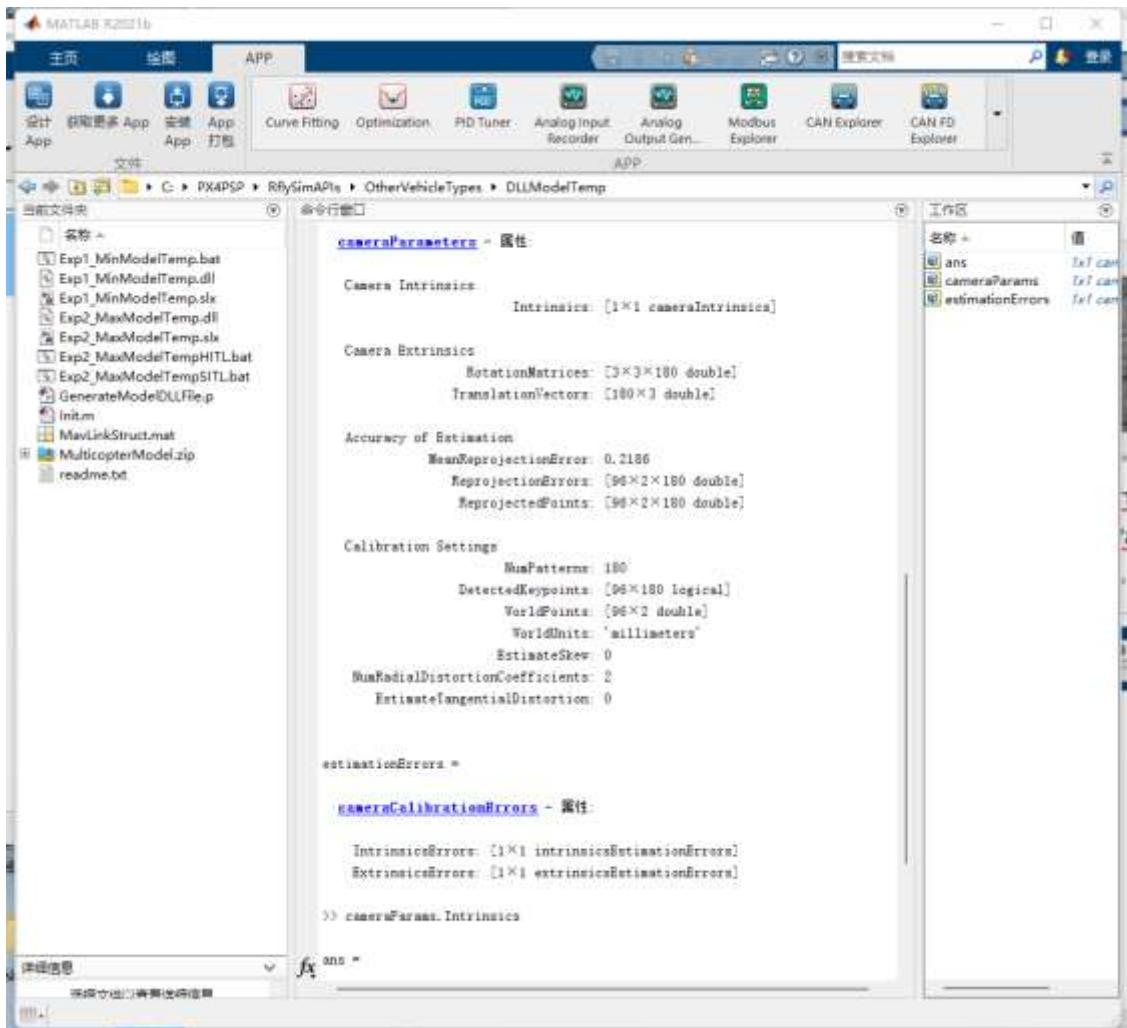




## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务1：Matlab标定板标定
- 12.回到Matlab工作区，会看到一个名为cameraParams的结构体，双击可以看到各成员变量。也可以在下方输入cameraParams.Intrinsics查看相机信息，输入cameraParams.IntrinsicMatrix查看内参矩阵。





## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务2：Python标定
- 1.启动CameraCalcDemo.bat，运行CameraCalcDemo.py一段时间后，工作路径下会得到一个存有捕获图片的文件夹，例如8.RflySimVision\0.ApiExps\3-VisionAI API\2.CameraCalcDemo\20231226\_154138
- 2.在bord-Calibration.py文件中将第19行的路径改为要标定的照片路径并运行  
`images = glob.glob(".\20220207_220418\img1\*.jpg")`
- 3.得到的内参矩阵如下

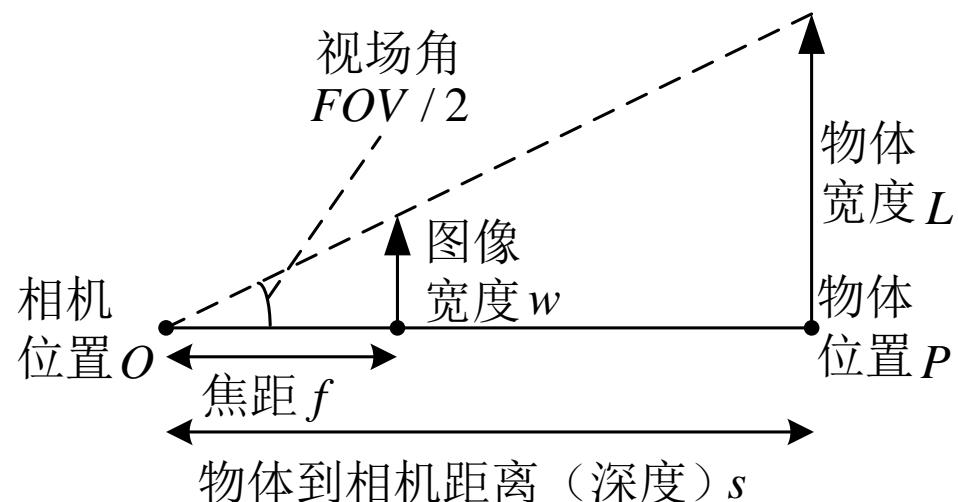
```
[[322.36795044  0.          318.71025401]
 [ 0.            319.37265015  235.49389293]
 [ 0.            0.            1.          ]]
```



## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务3：Python双小球的标定流程
- 1. 原理：生成两个小球，其距离固定为世界坐标下一米。根据两个小球在图像上的间距，来判断世界坐标系和图像坐标系的比例关系。再通过小球连线中点到相机的距离来量度坐标原点的平移。由两小球距离 $L$ 和小球在图像上的宽度 $w$ 和小球连线中点到相机的距离 $s$ 可得出焦距 $f$ 。公式：
$$f = \frac{w \cdot s}{L}$$

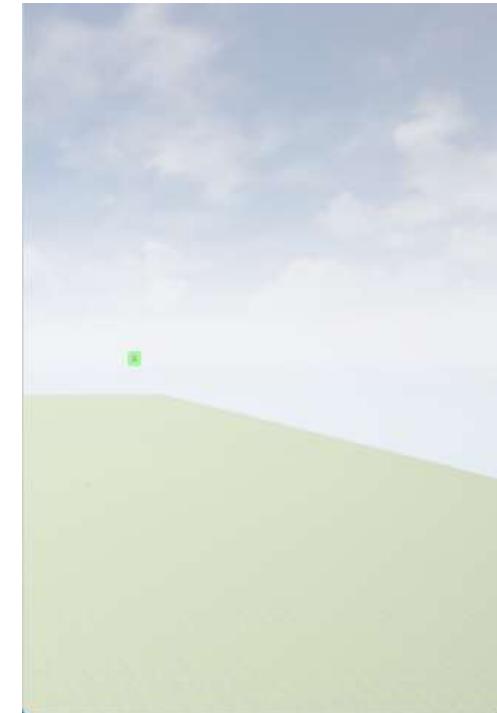
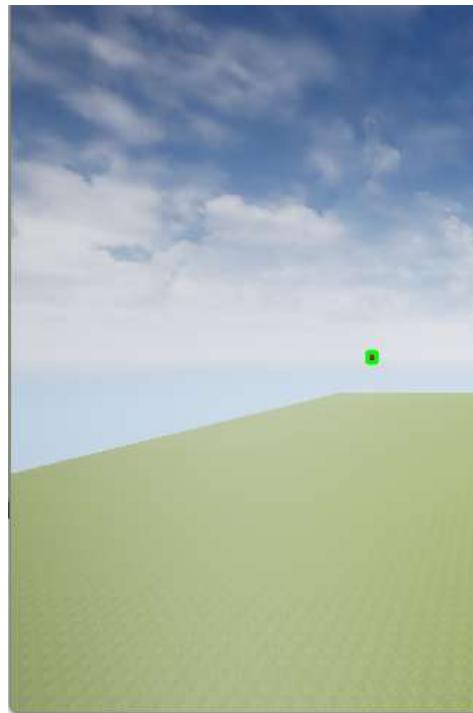
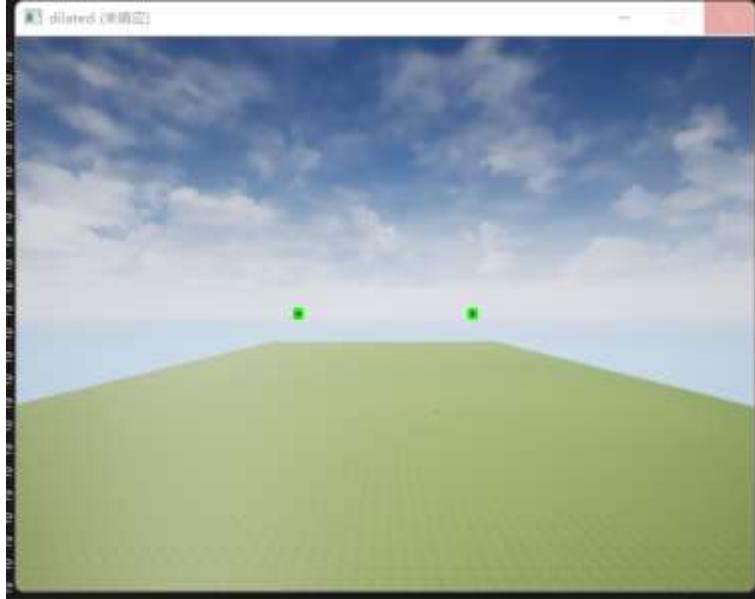




## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务3：Python双小球的标定流程
- 2.在VS Code中运行ball2-Calibration.py，可以得到结果如下图。





## 4.视觉AI进阶

### 4.2 虚拟相机标定实验—例程介绍

- 任务3：Python双小球的标定流程

The screenshot shows a Visual Studio Code interface with a Python script named 'ball2-Calibration.py' open. The code imports required libraries (random, time, sys, cv2, numpy) and performs a loop to calculate focal length and distance. The output window displays numerous iterations of calculations, showing values for 'focallength\_x' and 'distance'.

```
# import required libraries
import time
import sys
import cv2 as cv
import random
import numpy as np

The current focallength_x is 325.0000 ; The distance is 3.5500
The current focallength_x is 322.8750 ; The distance is 1.8750
The current focallength_x is 320.4500 ; The distance is 1.7500
The current focallength_x is 324.0750 ; The distance is 4.0250
The current focallength_x is 322.5000 ; The distance is 1.8000
The current focallength_x is 323.8500 ; The distance is 3.4750
The current focallength_x is 321.6000 ; The distance is 2.7000
The current focallength_x is 324.3000 ; The distance is 3.8250
The current focallength_x is 322.4000 ; The distance is 5.5000
The current focallength_x is 321.9500 ; The distance is 3.7250
The current focallength_x is 322.5000 ; The distance is 3.5250
The current focallength_x is 324.9000 ; The distance is 4.5750
The current focallength_x is 320.8500 ; The distance is 2.6250
The current focallength_x is 324.9000 ; The distance is 3.1500
The current focallength_x is 326.0250 ; The distance is 5.0250
The current focallength_x is 320.9500 ; The distance is 3.9750
The current focallength_x is 324.9000 ; The distance is 3.1500
The current focallength_x is 322.0000 ; The distance is 4.3250
The current focallength_x is 322.0500 ; The distance is 3.1250
The current focallength_x is 326.4000 ; The distance is 5.4000
The current focallength_x is 322.5000 ; The distance is 3.5250
The current focallength_x is 320.8500 ; The distance is 2.6250
The current focallength_x is 325.6000 ; The distance is 4.7000
The current focallength_x is 322.5000 ; The distance is 1.8000
The current focallength_x is 326.2500 ; The distance is 3.9250
The current focallength_x is 324.0000 ; The distance is 3.9000
The current focallength_x is 322.5250 ; The distance is 2.7250
The current focallength_x is 324.7800 ; The distance is 5.0750
The current focallength_x is 321.3000 ; The distance is 1.6500
The current focallength_x is 323.7000 ; The distance is 4.4500
The current focallength_x is 325.0500 ; The distance is 5.2250
The current focallength_x is 326.0250 ; The distance is 5.4750
The current focallength_x is 323.0800 ; The distance is 2.4250
The mean focal length of the camera is: 323.3800 100
```

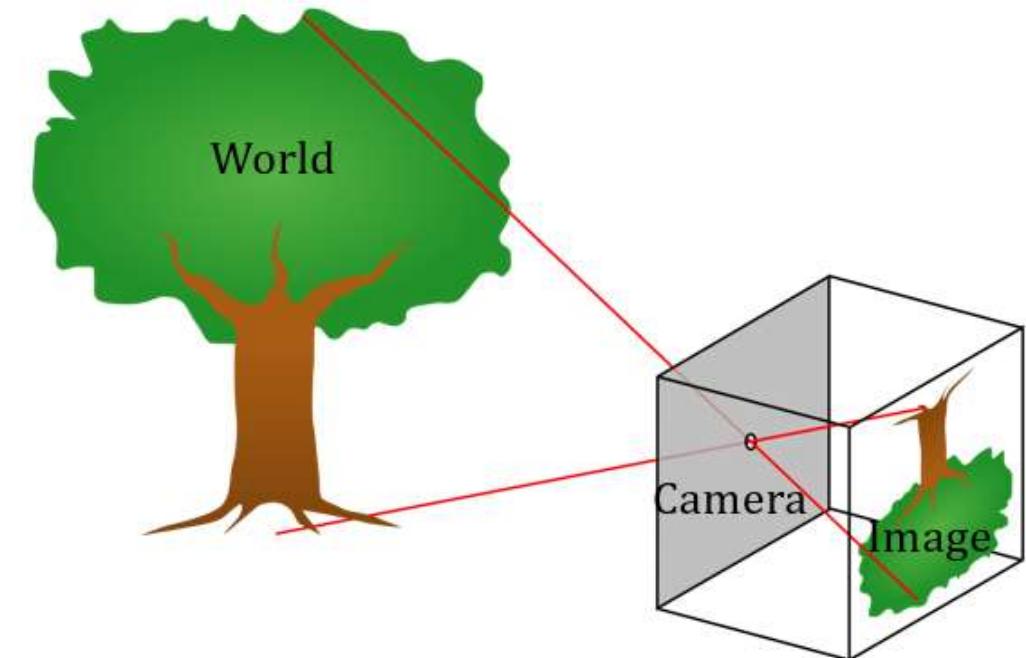


## 4.视觉AI进阶

例程见：RflySimAPIs\PythonVisionAPI\3-VisionAIDemos\3-GetRelativePosDemo

### 4.3 相机坐标转换—原理介绍

- 照相机是基于针孔原理来进行成像的，如右图为针孔原理示意图。将针孔模型对应到成像过程中，现实世界的物体即为三维空间的成像目标，针孔为摄像机中心，倒影成像平面则为二维影像平面。
- 针孔成像的一个特征是现实世界的任一点、其在成像平面上的投影点、相机中心在一条直线上，这种特征称为中心投影或者透视投影，也是做成像分析的基础。透视投影将三维空间投影到二维平面上，是一种降秩空间透射变换。





## 4.视觉AI进阶

---

### 4.3 相机坐标转换—原理介绍

- 要想用图片进行复杂的3D测量，必须借助一连串的复杂计算，而这些计算公式建立的基础就是笛卡尔坐标系。一般与无人机视觉图像相关的坐标系有以下几个：

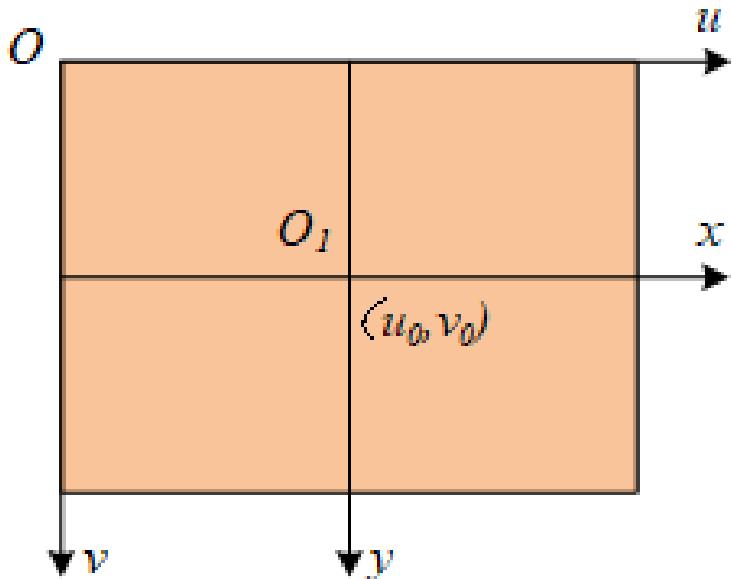
- 图像像素坐标系；
- 图像物理坐标系；
- 相机坐标系；
- 机体坐标系；
- 世界坐标系；



## 4.视觉AI进阶

### 4.3 相机坐标转换—原理介绍

- 图像像素坐标系：是一个二维直角坐标系，反映了相机中像素的排列情况。其原点 $O$ 位于图像的左上角， $u$ 、 $v$ 坐标轴分别与图像的两条边重合。像素坐标为离散值 $(0, 1, 2, \dots)$ ，以像素（pixel）为单位。
- 图像物理坐标系：为了将图像与物理空间相关联，需要将图像转换到物理坐标系下。原点 $O$ 位于图像中心（理想状态下），是相机光轴与像平面的交点（称为主点）。 $x$ 、 $y$ 坐标轴分别与 $u$ 、 $v$ 轴平行。两坐标系实为平移关系，平移量为 $(u_0, v_0)$ 。
- 以上两者可以统称为影像坐标系，两种坐标系都是二维的坐标系。





## 4.视觉AI进阶

### 4.3 相机坐标转换—原理介绍

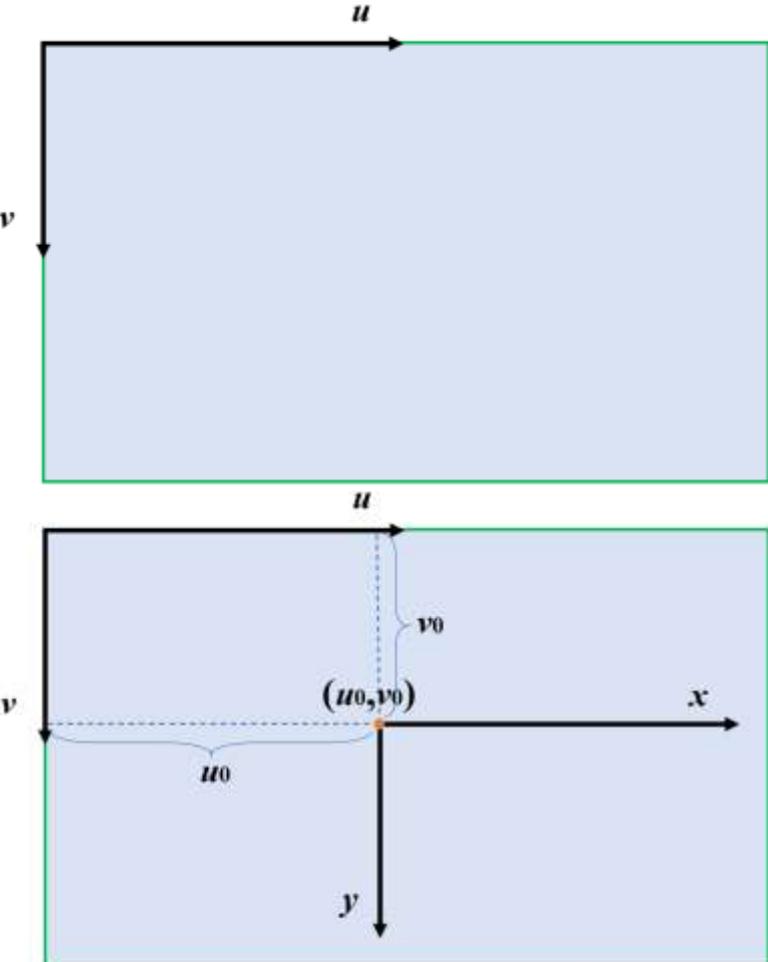
- 右图为影像坐标系的两种不同表达方式，如果我们知道像素到物理尺寸的转换关系，即一个像素的物理尺寸，也就是像元尺寸为 $dx \times dy$ （x方向尺寸为 $dx$ , y方向尺寸为 $dy$ ），就可以在两坐标系之间相互转换：

$$u - u_0 = x/d_x$$

$$v - v_0 = y/d_y$$

- 为了便于矩阵运算，可以写成矩阵形式。公式两边的三维矢量是一种齐次表达方式，即把第三维设置为1来用三维矢量表示二维矢量，这样做的好处是可以用矩阵运算的方式完成三维到二维的变换。

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{d_x} & 0 & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$





## 4.视觉AI进阶

---

### 4.3 相机坐标转换—原理介绍

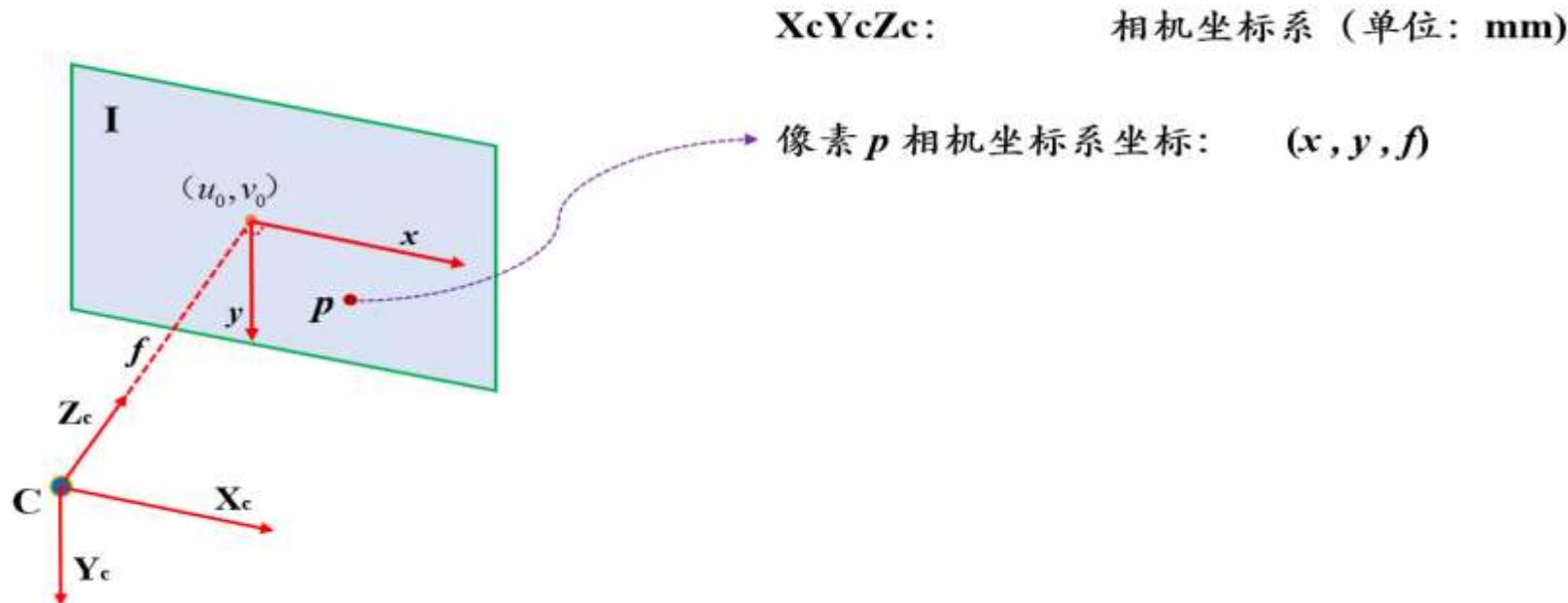
- 而之后的相机坐标系、机体坐标系和世界坐标系都是三维的坐标系，在四旋翼无人机中，相机坐标系和机体坐标系之间的相对位置是固定不变的，所以在一开始测得相对位置之后即可将相机坐标系和机体坐标系看作一个整体一起讨论，这两者可以看成是原点和坐标轴的方向与无人机初始状态绑定的坐标系，是一个相对于地面来说会运动的坐标系；
- 世界坐标系则是以地面上某一个固定位置为原点的坐标系，是一个绝对坐标系，它的作用是将空间中所有的点都统一到同一个坐标系下表达。
- 三维坐标系之间的转换用旋转和平移运动会更方便表达，重难点是相机坐标系等三维坐标系与图像坐标系等二维坐标系之间的转换。



## 4.视觉AI进阶

### 4.3 相机坐标转换—原理介绍

- 相机坐标系的原点在相机中心，相机坐标系的XY轴和影像坐标系的xy轴平行，Z轴垂直于像平面且朝向像平面，Z轴和像平面的交点正是影像xy坐标系的原点（像主点）如下图所示：

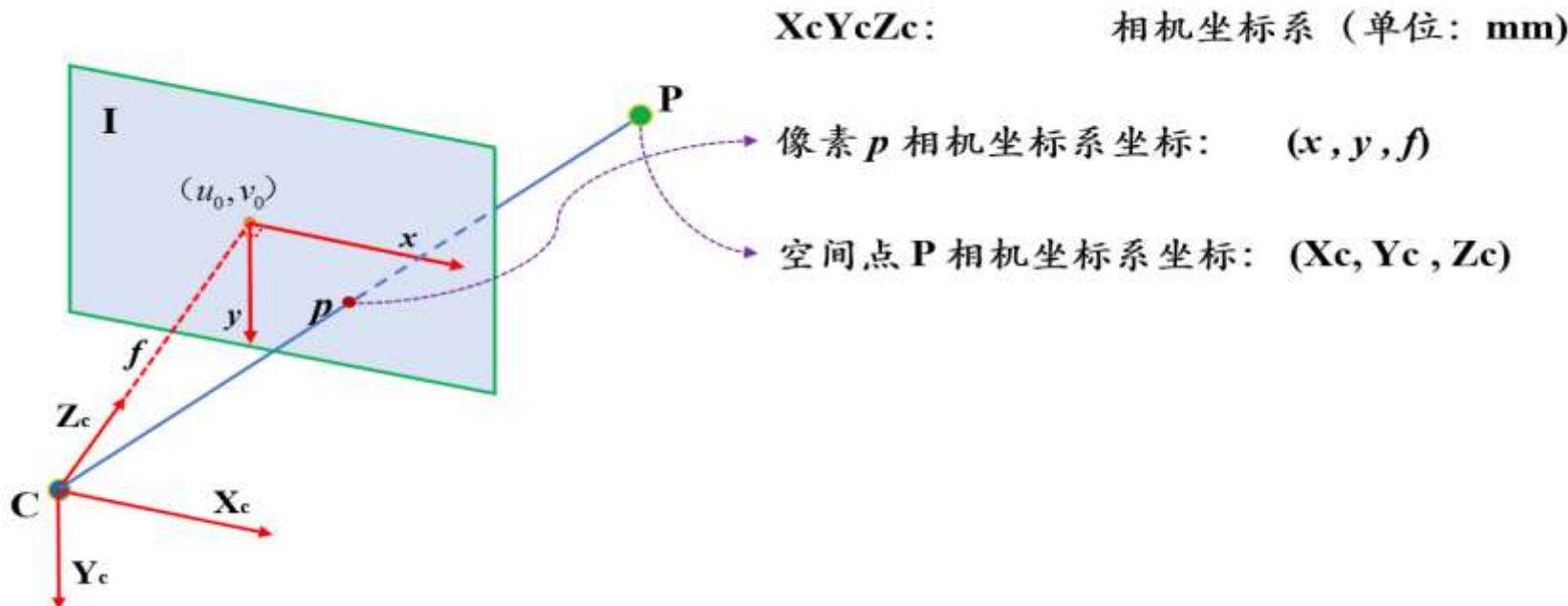




## 4.视觉AI进阶

### 4.3 相机坐标转换—原理介绍

- 在此方案下，像平面上的所有像素点在相机坐标系下的Z坐标等于焦距f，相机XY坐标系和影像xy坐标系下的值相等，即若像素点p在影像xy坐标系下的坐标为(x,y)，则其在相机XY坐标系下的坐标为(x,y,f)。

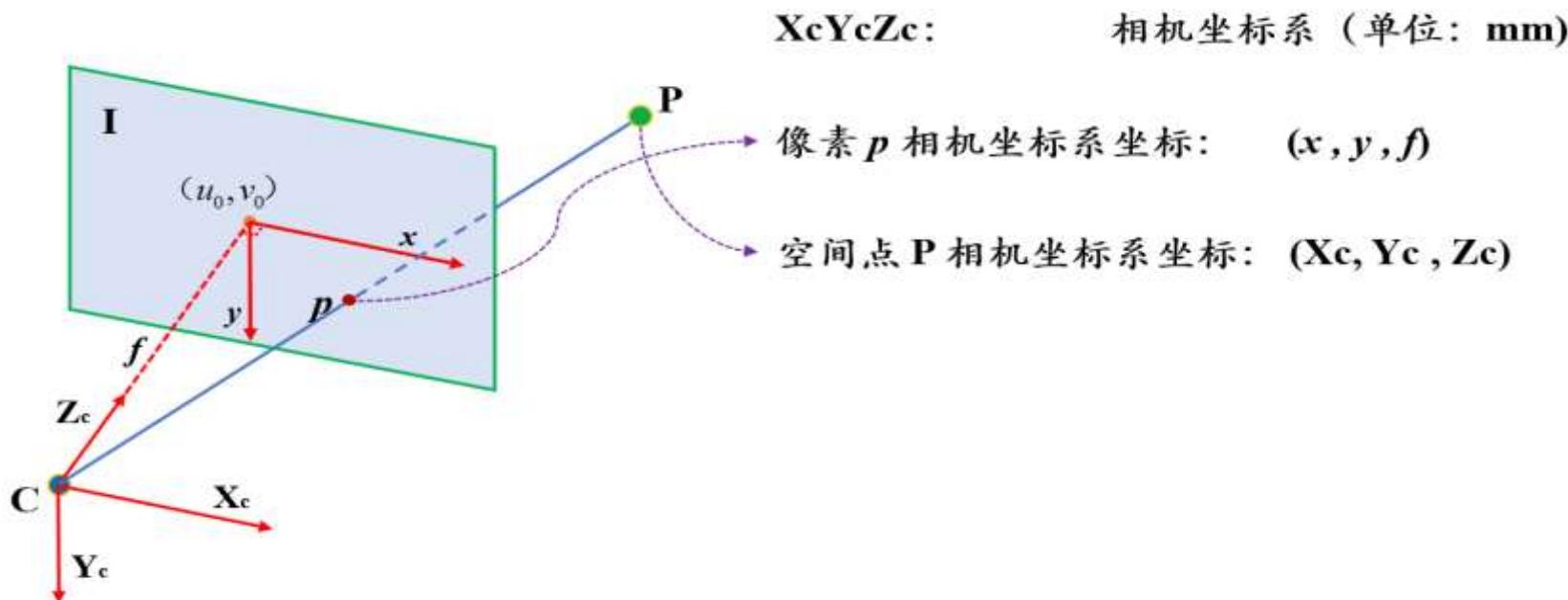




## 4.视觉AI进阶

### 4.3 相机坐标转换—原理介绍

- 假设像素 $p$ 对应的空间点在相机坐标系下的坐标为 $(X_c, Y_c, Z_c)$ 。若两点位于从坐标系原点发出的同一条直线上，那么他们的坐标是等比例的关系。即  $\frac{x}{X_c} = \frac{y}{Y_c} = \frac{f}{Z_c}$





## 4.视觉AI进阶

### 4.3 相机坐标转换—原理介绍

- 假设像素p对应的空间点在相机坐标系下的坐标为(Xc,Yc,Zc)。若两点位于从坐标系原点发出的同一条直线上，那么他们的坐标是等比例的关系。即

$$\frac{x}{X_c} = \frac{y}{Y_c} = \frac{f}{Z_c}$$

- 为了便于矩阵运算，写成矩阵形式：

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{Z_c} & 0 & 0 \\ 0 & \frac{f}{Z_c} & 0 \\ 0 & 0 & \frac{1}{Z_c} \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

- 也可以将xy坐标转换为uv坐标：

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{d_x} & 0 & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{f}{Z_c} & 0 & 0 \\ 0 & \frac{f}{Z_c} & 0 \\ 0 & 0 & \frac{1}{Z_c} \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \frac{1}{Z_c} \begin{bmatrix} \frac{f}{d_x} & 0 & u_0 \\ 0 & \frac{f}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

- 通常把Zc称为尺度因子 $\lambda$ ，把中间的3x3矩阵叫做内参矩阵K，显然内参矩阵K描述的是相机坐标系到uv坐标系的转换关系。



## 4.视觉AI进阶

### 4.3 相机坐标转换—原理介绍

- 内参矩阵K是相机的关键参数之一， $\frac{f}{dx}$ 和 $\frac{f}{dy}$ 实际是将以物理尺寸为单位的焦距f转换成像素为单位的焦距值，记 $f_x = \frac{f}{dx}$ ， $f_y = \frac{f}{dy}$ ， $f_x$ 和 $f_y$ 分别是焦距在两个像元方向上的像素单位值。最终得到内参的矩阵表达：

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$



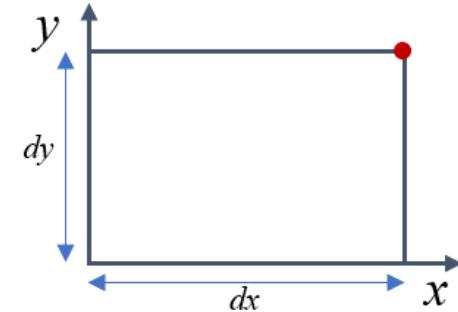
## 4.视觉AI进阶

### 4.3 相机坐标转换—原理介绍

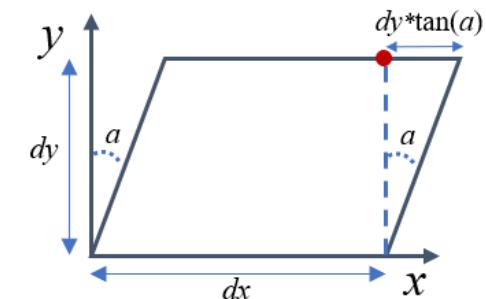
- 由于制造工艺的偏差，现实中的像素不是绝对的矩形，而是平行四边形，日如右图所示。这时候像素的纵向边界和y轴并不平行而是倾斜一定的角度，于是在K矩阵中引入一个倾斜因子s，此时K矩阵表示为

$$K = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

rectangular pixel



non-rectangular pixel





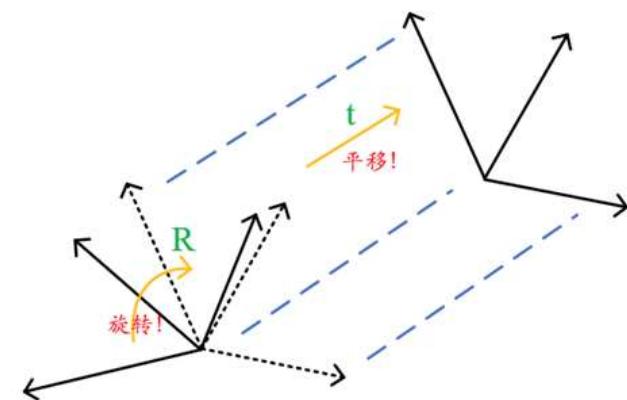
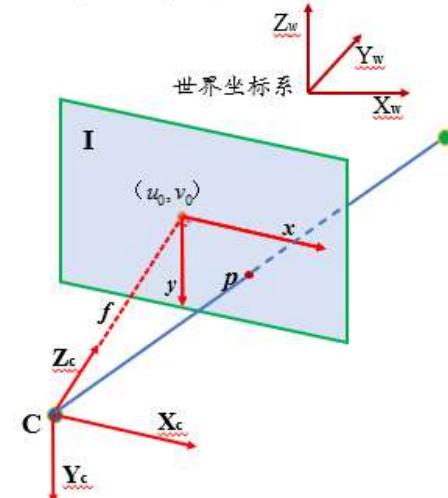
## 4.视觉AI进阶

### 4.3 相机坐标转换—原理介绍

- 对于世界坐标系，则是一个固定的绝对坐标系。
- 世界坐标系和相机坐标系都是三维坐标系，它们之间可以用旋转平移来做转换。
- 假设空间点P在世界坐标系中的坐标为(X<sub>w</sub>,Y<sub>w</sub>,Z<sub>w</sub>)，则可以通过一个3x3的单位正交旋转矩阵R和3x1的平移矢量t来转换成相机坐标系坐标(X<sub>c</sub>,Y<sub>c</sub>,Z<sub>c</sub>)：

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R_{3 \times 3} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + t_{3 \times 1} \text{ 或 } \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [R_{3 \times 3} \quad t_{3 \times 1}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

- 旋转矩阵R和平移矢量t称为相机的外参矩阵。





## 4.视觉AI进阶

### 4.3 相机坐标转换—例程介绍

- 打开“RflySimAPIs\8.RflySimVision\0.ApiExps\3VisionAI API\4.GetRelativePos Demo”文件夹。
- 双击启动**GetRelativePosDemo.bat**文件。
- 打开**GetRelativePosDemo.py**脚本，对关键语句设置断点，在调试模式下逐句执行语句。
- 本脚本展示了几种设置和获取软件中坐标位置信息的方法，包括地面坐标系中和无人机坐标系中。

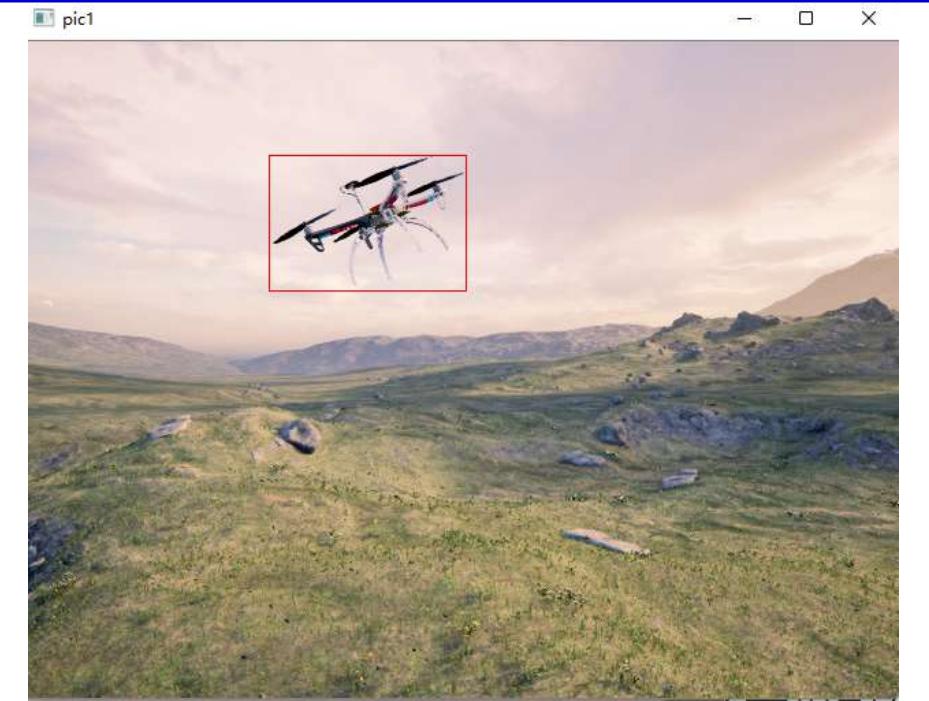
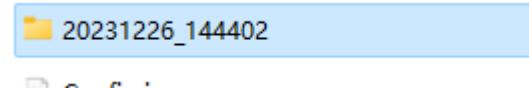
名称	修改日期
GetRelativePosDemo.bat	2023/12/25 11:49
GetRelativePosDemo.py	2023/10/27 10:50
method of application.txt	2023/10/24 10:19
readme.pdf	2023/12/7 16:07



## 4.视觉AI进阶

### 4.4 生成数据集的方法—例程介绍

- 打开“RflySimAPIs\8.RflySimVision\0.ApiExps\3-VisionAI API\5.GenVisionDataSet文件夹。
- 双击启动OneCameraCal.bat文件启动RflySim3D。
- 运行get\_dataset.py，会在画面的随机位置出现飞机，并用红色方框标注出来。目录中新增以时间戳命名的文件夹，其中images目录存储采集到的飞机图像，labels目录存储每张图像对应的标签（YOLO格式）。





## 4.视觉AI进阶

### 4.4 生成数据集的方法—例程介绍

- 修改maketxt.py中的ROOT路径为生成的文件夹路径，注意将“\”改为“/”防止转义。

```
9 ROOT = 'C:/PX4PSP/RflySimAPIs/PythonVisionAPI/3-VisionAIDemos/4-GenVisionDataSet/20220729_104555/' # 根据自己的目标进行替换
```

- 运行程序，会在该文件夹中生成**data**目录和**ImageSets**目录，**data**目录即训练时读取的数据集目录，默认按照9:1的比例随机划分训练集和测试集。

3-VisionAIDemos > 4-GenVisionDataSet > 20220729_104555 >			
名称	修改日期	类型	大小
data	2022/7/29 10:54	文件夹	
images	2022/7/29 10:46	文件夹	
ImageSets	2022/7/29 10:54	文件夹	
labels	2022/7/29 10:46	文件夹	



## 4.视觉AI进阶

### 4.4 生成数据集的方法—例程介绍

- 目前使用的是3号的四旋翼无人机模型，如果要换成其他的模型，需要知道该模型所有凸起点的坐标，并且更换下图所示函数的每个点对应的坐标，注意以下的参数是包含了机翼在内的。

```
center + np.transpose(np.dot(np.linalg.inv(eul2rot(uavAng)),
    np.transpose(np.array([-0.035, -0.035, -0.135-0.015])))),
# 需要把机翼也要包含在框内
center + np.transpose(np.dot(np.linalg.inv(eul2rot(uavAng)),
    np.transpose(np.array([0.245, -0.245, -0.035])))),
center + np.transpose(np.dot(np.linalg.inv(eul2rot(uavAng)),
    np.transpose(np.array([0.245, 0.245, -0.035])))),
center + np.transpose(np.dot(np.linalg.inv(eul2rot(uavAng)),
    np.transpose(np.array([-0.245, -0.245, -0.035])))),
center + np.transpose(np.dot(np.linalg.inv(eul2rot(uavAng)),
    np.transpose(np.array([-0.245, 0.245, -0.035])))),
center + np.transpose(np.dot(np.linalg.inv(eul2rot(uavAng)),
    np.transpose(np.array([0.13, -0.13, 0.17])))),
center + np.transpose(np.dot(np.linalg.inv(eul2rot(uavAng)),
    np.transpose(np.array([0.13, 0.13, 0.17])))),
center + np.transpose(np.dot(np.linalg.inv(eul2rot(uavAng)),
    np.transpose(np.array([-0.13, -0.13, 0.17])))),
center + np.transpose(np.dot(np.linalg.inv(eul2rot(uavAng)),
    np.transpose(np.array([-0.13, 0.13, 0.17]))))
```





## 4.视觉AI进阶

### 4.4 生成数据集的方法—例程介绍

- 增加或者减少上图函数提起点的数量时，还需要更改下图所示位置数组的列数。

```
202 shape_num = 9 # 目标自凸点个数以及自身原点,当前目标是UAV,使用box包裹,那就是有8个顶点和一个原点
```

- 更改自动生成的模型时也需要对数据集中模型的编号进行改变，也就是此处下方的'0'，这里是将四旋翼作为训练时class中的0号所对应的名字，如果要换成其他模型请更改编号，并且在训练时将相应class中的对应编号位置改成所需要的名字。

```
288 file.writelines(['0', ' ', str(y1), ' ', str(289 y2), ' ', str(y3), ' ', str(y4)]) # YOLO格式生成
```

- 最后，如果需要把不同模型的数据集，分次生成放在一起形成一个数据集，还需要更改如图所示的cnt起始参数，也就对应着生成数据集时图片和txt文档名字的起始数字。

```
200 cnt = 0
```



## 4.视觉AI进阶

### 4.5 视觉目标识别实验—环境配置

- 本实验需要首先在python环境中安装PyTorch，其分为CPU版本和GPU版本。如果电脑的显卡为AMD显卡，只能安装CPU版本；如果是NVIDIA显卡，推荐安装GPU版本，速度更快。
- 安装GPU版本需要先安装CUDA Toolkit和cuDNN，可以分别在链接<https://developer.nvidia.com/cuda-toolkit-archive>和<https://developer.nvidia.com/rdp/cudnn-archive>中找到显卡对应版本的安装方式。
- 安装完成之后开始安装PyTorch，进入链接<https://pytorch.org/>选择相应的安装命令，如果安装CPU版本，在Compute Platform中选择CPU即可。





## 4.视觉AI进阶

### 4.5 视觉目标识别实验—例程介绍

本例程位于“8.RflySimVision\2.AdvExps\e7\_ObjDetectYolo\ShootBallBaseOnYolo”

- 双击启动ShootBall3SITL.bat文件启动RflySim3D。
- 运行ShootBall3.py程序，就能看到飞机撞击小球的效果。



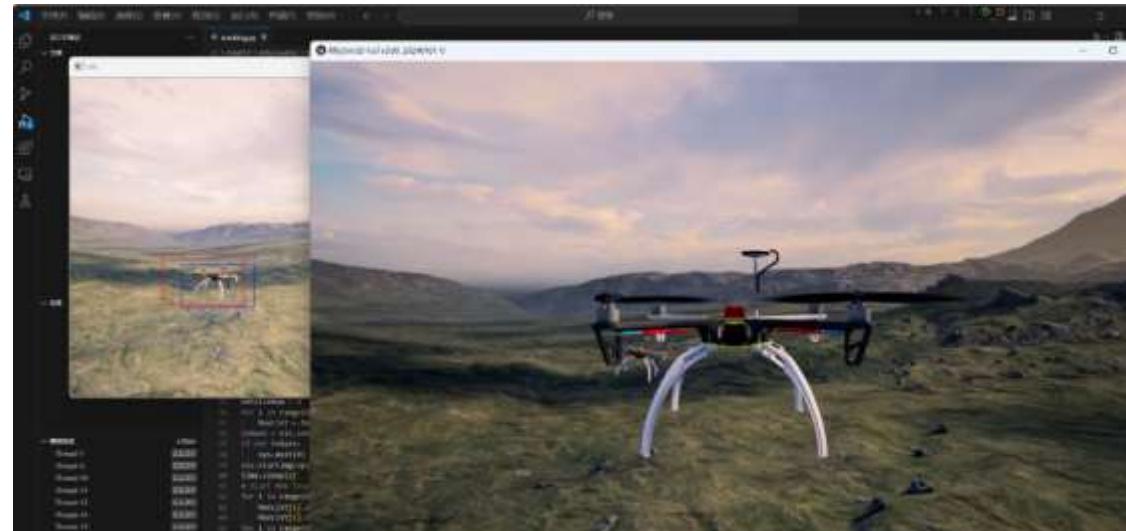


## 4.视觉AI进阶

### 4.6 视觉AI训练实验—例程介绍

#### 目标跟踪实验

- 打开“RflySimAPIs\8.RflySimVision\2.AdvExps\e8\_SingleObjTracking”文件夹。
- 右键选中Tracking.bat选择以管理员身份运行，启动RflySim3D。
- 运行tracking.py，会看到在飞机右侧又生成了一架飞机，两架飞机起飞后右侧飞机会飞行到前方并进行运动，同时生成一个窗口，显示目标飞机识别检测框与跟踪检测框重合效果。



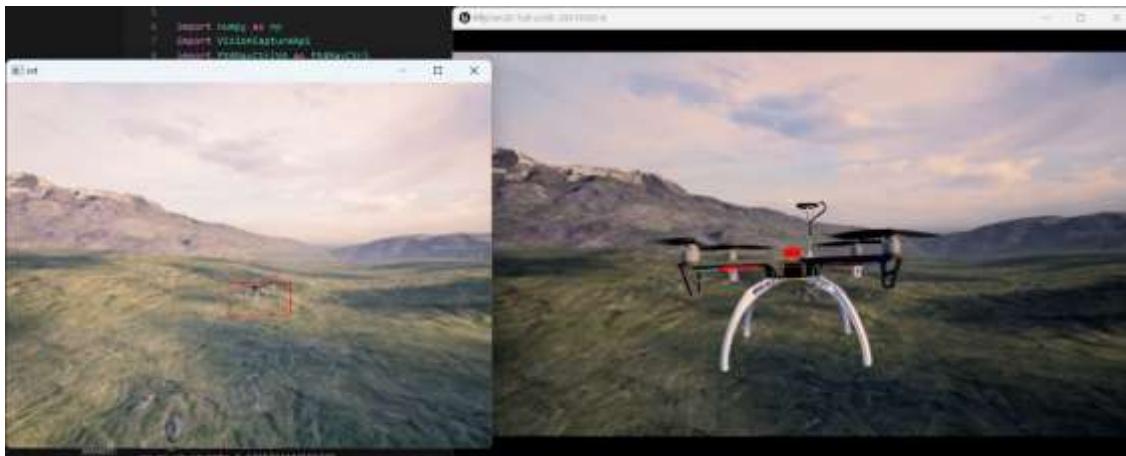


## 4.视觉AI进阶

### 4.6 视觉AI训练实验—例程介绍

#### 目标跟随实验

- 本例程位于“RflySimAPIs\8.RflySimVision\2.AdvExps\e9\_Object-Follow”文件夹。
- 以管理员身份运行target\_follow.bat。
- 运行target\_follow.py,可以看到两架飞机起飞，右侧飞机飞行到前方，同时生成一个窗口显示前方飞机，可以看到目标飞机识别检测框，目标飞机开始运动，主飞机跟随目标飞机进行运动。



```
copter_id: 1
center: [319, 336]
dis_err [ 1. -96.]
width_ 31.709096677631123
vx,vy,vz,yawrate 0.9512729003289336
copter_id: 1
center: [316, 375]
dis_err [ 4. -135.]
vx,vy,vz,yawrate 0
copter_id: 1
center: [316, 363]
dis_err [ 4. -123.]
vx,vy,vz,yawrate 0
copter_id: 1
center: [317, 344]
dis_err [ 3. -104.]
vx,vy,vz,yawrate 0
copter_id: 1
center: [317, 338]
dis_err [ 3. -98.]
vx,vy,vz,yawrate 0
copter_id: 1
center: [315, 373]
dis_err [ 5. -133.]
vx,vy,vz,yawrate 0
copter_id: 1
```



# 大纲

---

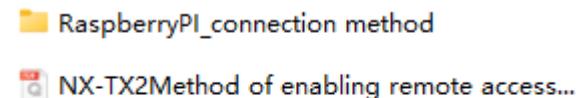
1. 总体介绍
  2. 基础接口使用
  3. 视觉控制例子
  4. 视觉AI进阶
  5. 分布式视觉仿真
-



## 5.分布式视觉仿真

### 5.1 例程总体介绍

- 在实际无人机顶层开发和测试过程中，常常有如下两方面的需求
- 1) 利用机载计算机（树莓派、NVIDIA Jetson Xavier NX、NANO等）运行Linux/ROS环境来进行视觉感知算法（识别、SLAM、避障等）开发。
- 2) 多个无人机集群进行分布式的感知与控制，即仿真多无人机集群，每个无人机需要具备自己独立的视觉，完成特定场景的探索与协同任务。
- 本小节针对上述两项需求，构建了一系列的解决方案，并提供例程和文档。如下图所示，“8.RflySimVision\0.ApiExps\2-DistributedSimAPI\0.Preparation”文件夹包含了树莓派、NX等硬件远程连接调试的方法，便于实验的进行；

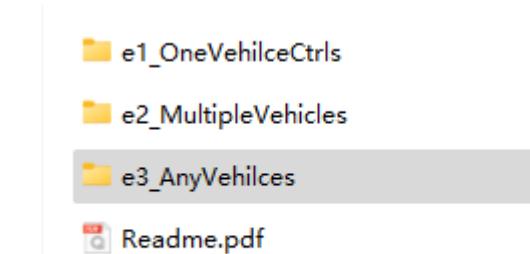




## 5.分布式视觉仿真

### 5.1 例程总体介绍

- “8.RflySimVision\2.AdvExps\e0\_AdvApiExps\9.VisionAPIsTest” 和 “8.RflySimVision\0.ApiExps\2-DistributedSimAPI\1.VisionAPIsTest” 介绍了不同图像外传接口的例程； “8.RflySimVision\3.CustExps\2-DistributedSimDemos” 目录下包含三个文件， “e1\_OneVehilceCtrls” 介绍了单无人机视觉例程； “e2\_MultipleVehicles” 介绍多无人机例程； “e3\_AnyVehilces” 介绍任意无人机数量的视觉控制例程。

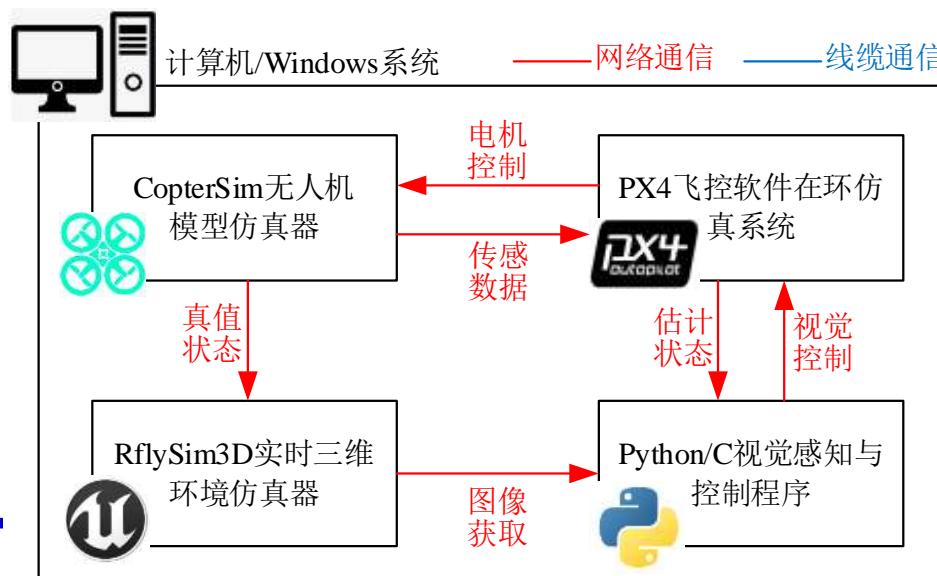




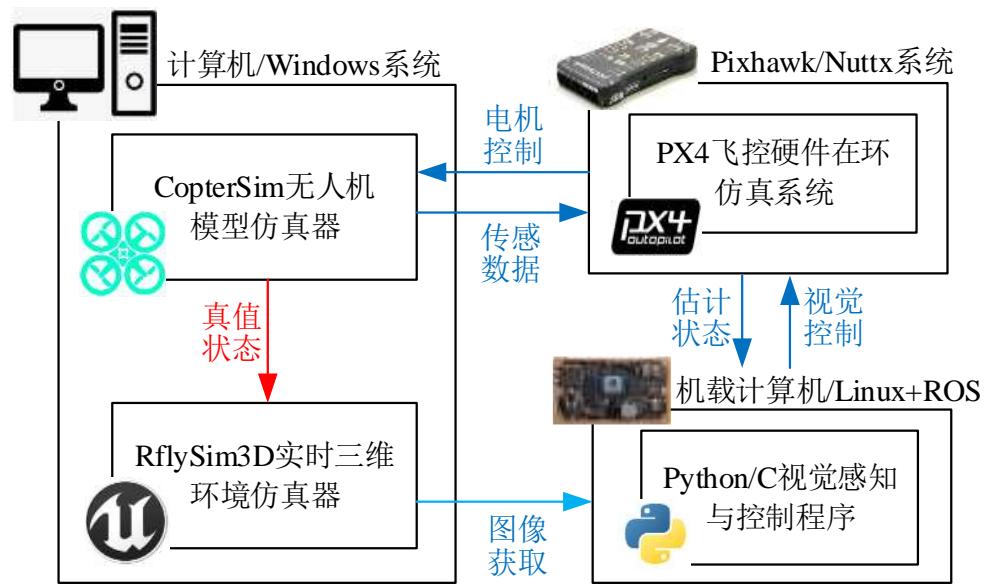
## 5.分布式视觉仿真

### 5.2 接入ROS系统进行仿真—算法开发到部署

- 从便于开发与易于使用的角度，RflySim平台仅限运行于Windows平台，但是支持将图像通过网线等介质传输到其他计算机（嵌入式主机、虚拟机、台式机等）不同系统。
- 如下图(a)所示为前文例程的实现构架，本构架适合单个感知算法的快速开发与验证，完成后再移植到图(b)所示机载Linux/ROS系统中，进行部署与硬件在环测试。
- 作为中间过渡，图(a)中的Python感知算法模块可替换成Linux/ROS虚拟机环境，这样在单台电脑上就能实现算法的部署与初步测试，达到提高效率、节约成本的目的。



(a) 单电脑Windows下纯软件在环开发模式



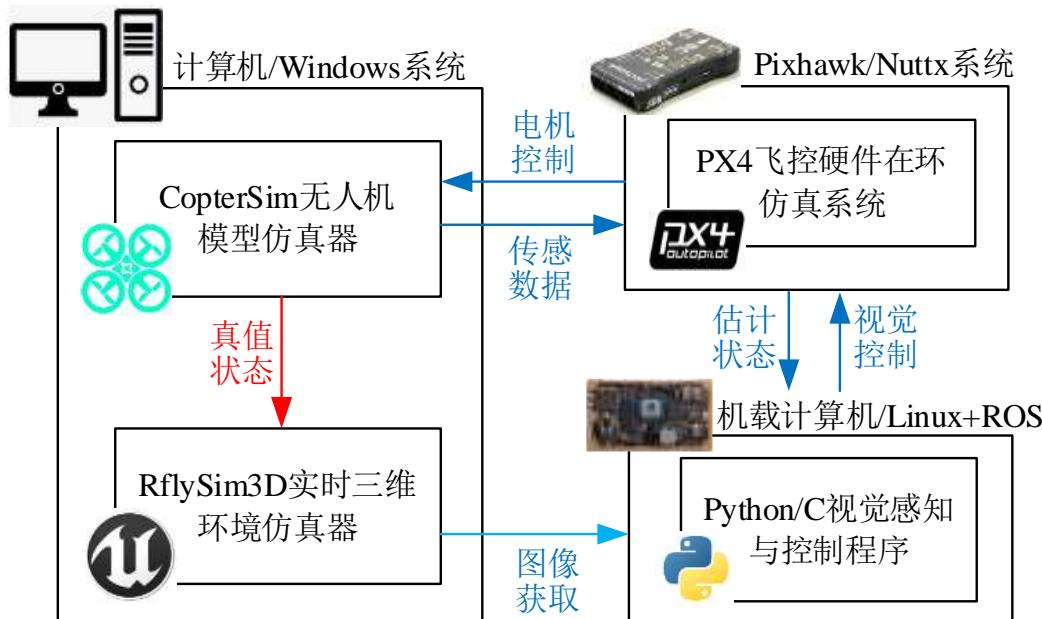
(b) “自驾仪+通信链路+机载计算机”硬件在环仿真



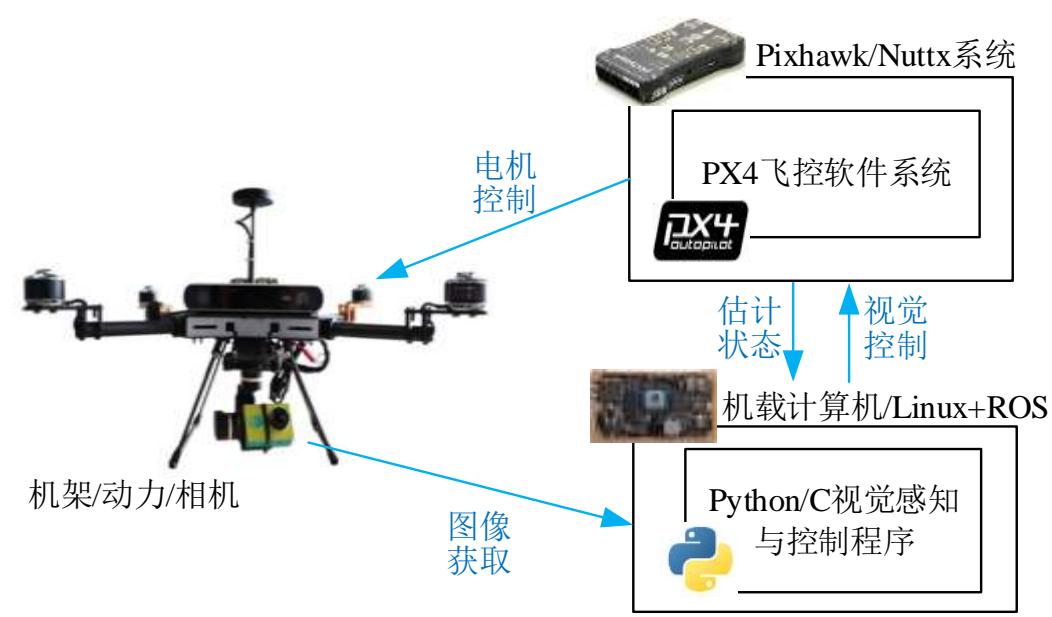
## 5.分布式视觉仿真

### 5.2 接入ROS系统进行仿真—硬件在环仿真到真机实验

- 机载计算机从网络（目前使用UDP协议）接收到图像后，可以直接进行图像处理，或者通过ROS节点转发，给其他感知模块使用。
- Pixhawk与机载计算机通过数传串口线直接连接，与真机上的硬件连接方式一致。
- 直接将Pixhawk/PX4飞控的输出插在电调，图像获取接口连接相机，即可完成图(b)所示的硬件在环仿真到图(c)所示的真机系统的迁移。



(b) “自驾仪+通信链路+机载计算机” 硬件在环仿真



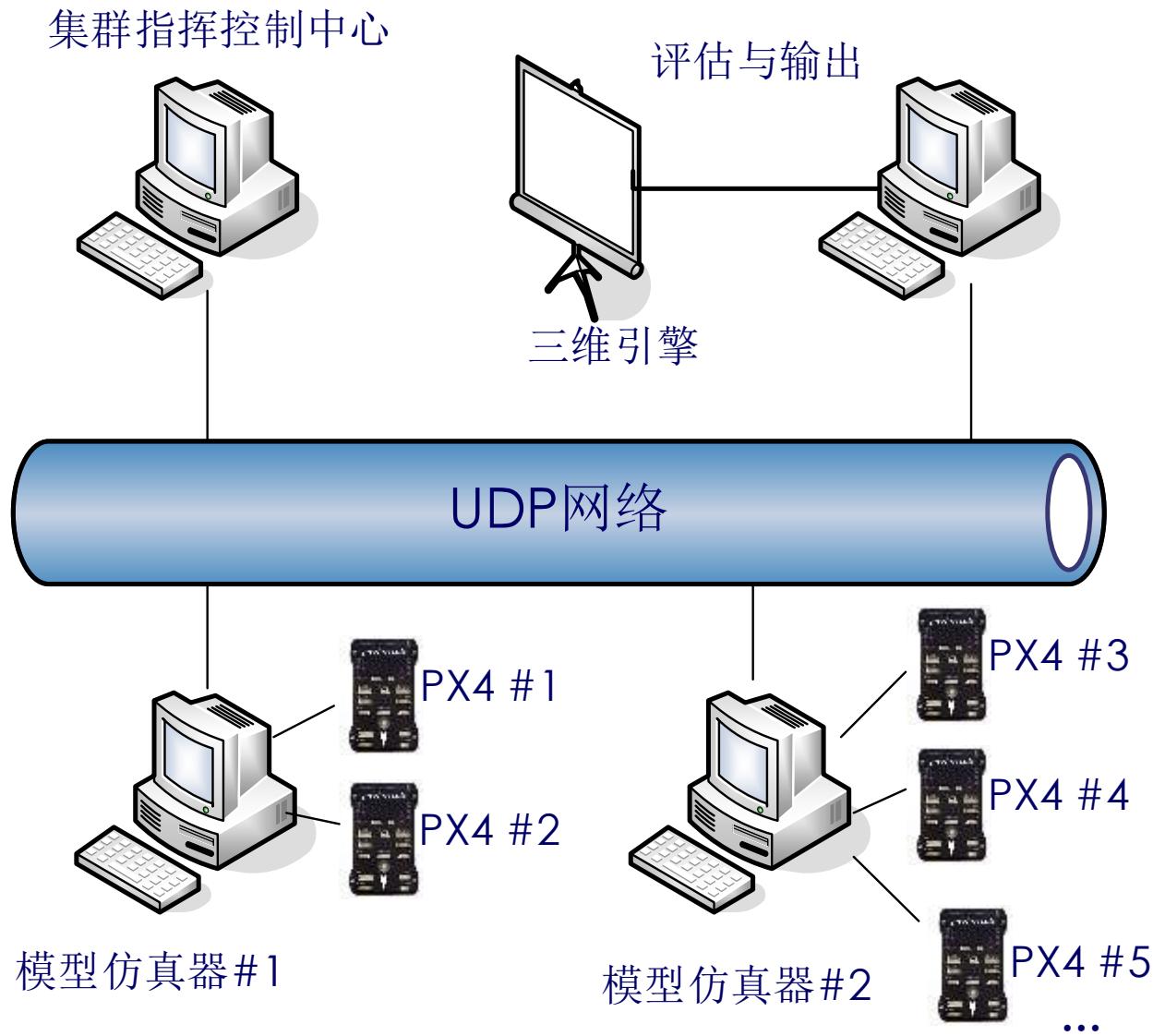
(c) “机架+自驾仪+通信链路+机载计算机” 真机实验



## 5.分布式视觉仿真

### 5.3 分布式集群组网仿真—无人机数量扩展方案

- 无论从多机仿真的角度，还是从无人机真实集群控制的角度，通信带宽和计算性能永远是制约集群数量增加的重要瓶颈
- 由于仿真计算机的性能存在瓶颈，单台计算机能够连接Pixhawk进行硬件在环仿真的数量是有限制的，因此需要通过多台电脑组网方式，实现无人机数量的任意扩展。
- 随着无人机数量的增加，飞机间相互通信的数据量暴增，直到通信带宽达到饱和。因此，需要将无人机集群整体分为若干子群，采用网络分层的方式，实现更大规模的集群仿真。



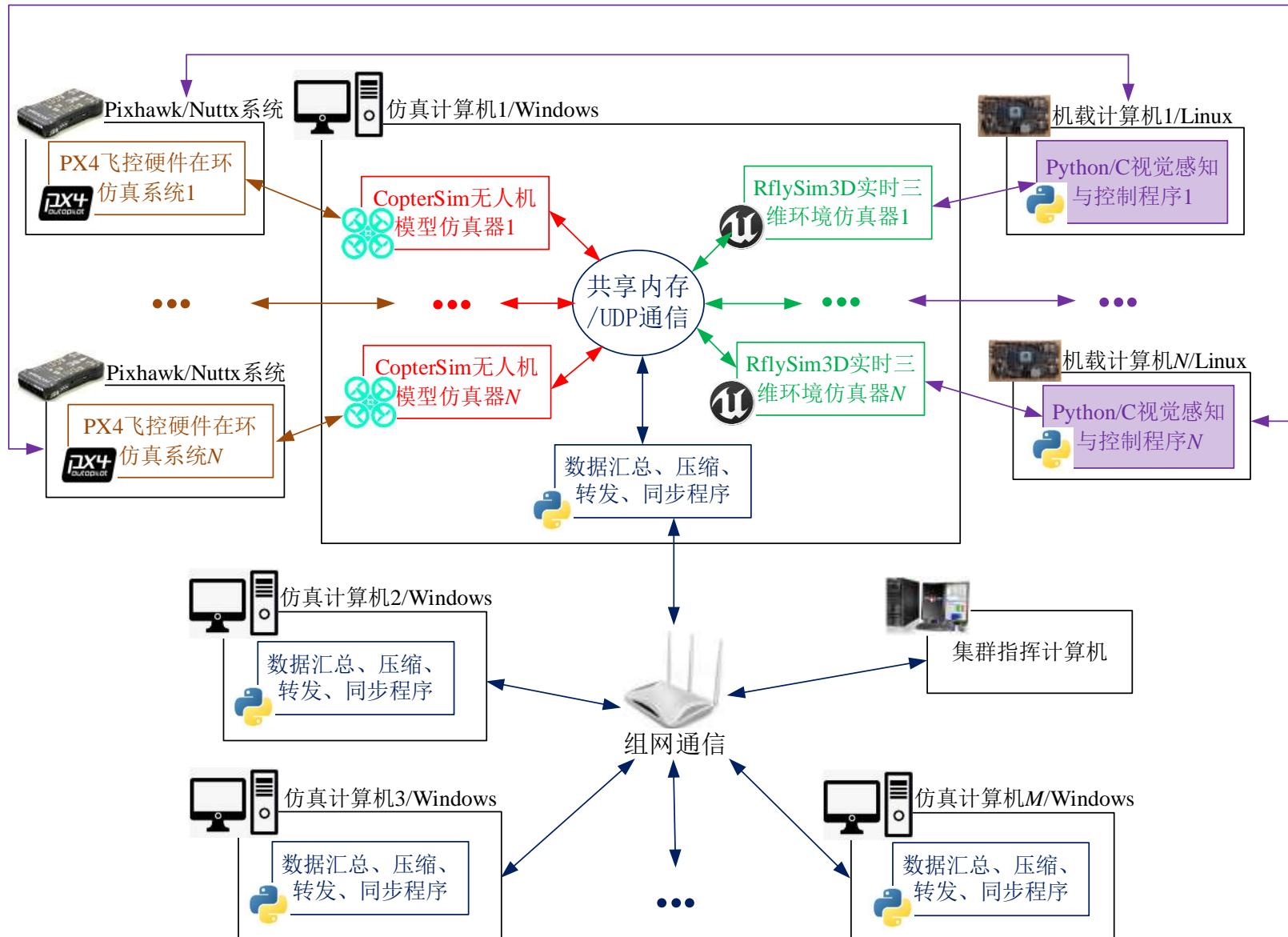


## 5.分布式视觉仿真

### 5.3 分布式集群组网仿真

#### —通信优化方案

- 本电脑内部各个程序间的通信采用共享内存或UDP通信方式，图像等大数据传输直接在内存上操作，延迟最低、速度最快。
- 每台电脑可以开启多个硬/软件在环仿真系统，模拟多个无人机，飞控与机载计算机之间通过有线数传连接，降低延迟。
- 每台电脑向外收发数据经过特定模块进行汇总、压缩、同步，再传出去，确保网内通信顺畅。
- 支持采用请求式通信（DDS协议），支持大规模集群仿真。

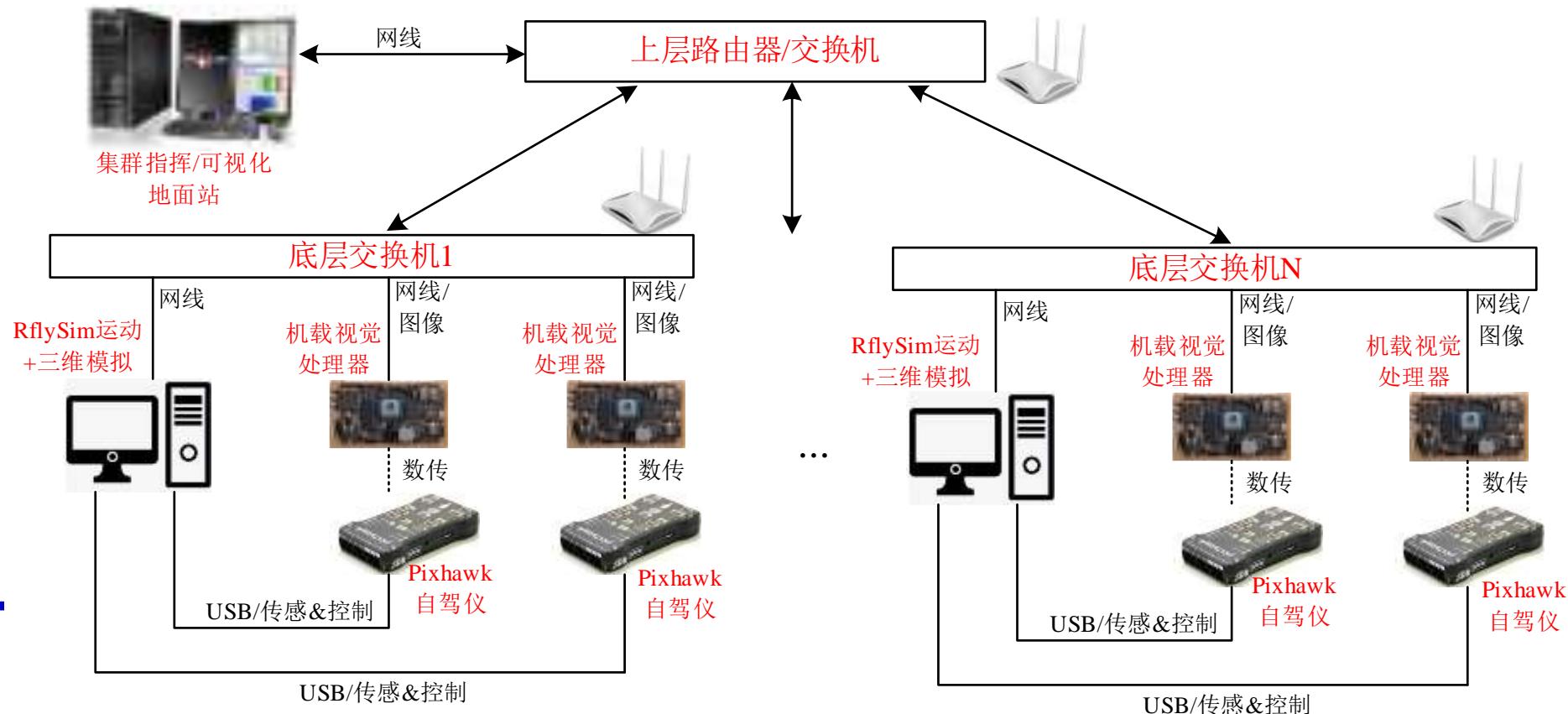




## 5.分布式视觉仿真

### 5.3 分布式集群组网仿真—图像传输组网优化

- 为了避免在局域网内大量传输图像，在本仿真框架中，RflySim仿真计算机的视觉图像直接通过指定IP的形式发送到底层交换机下的嵌入式主机，不向上层传输。
- 在上层路由器（交换机）中，主要传输无人机的状态数据，且采用订阅发布机制，避免无用通信。





## 5.分布式视觉仿真

### 5.4 嵌入式主机的远程访问方法

- 由于分布式实验涉及较多的电脑或嵌入式小主机，为了便于后续实验的结果观察，推荐在RflySim平台运行的Windows电脑下，采用“远程桌面连接”的工具来连接其他主机。  
注：当然也可以嵌入式主机专用的远程访问工具（需要带图形界面，而不是单命令行）
- 如果远程主机是Windows系统，那么升级专业版平台并开启远程访问后，可以直接通过主机名称，并输入账号密码来远程访问。
- 如果远程主机是树莓派（或Ubuntu系统），可以进入目录“8.RflySimVision\0.ApiExps\2-DistributedSimAPI\0.Preparation\RaspberryPI\_connection method”，  
并按其中文档步骤配置树莓派，然后从路由器中获取IP地址，  
就能通过“远程桌面连接”工具进行图形化访问。  
注：我们的硬件已配好，可直接连。
- 如果是TX2或NX，可以“RflySimAPIs\8.RflySimVision\0.ApiExps\2-DistributedSimAPI\0.Preparation”的文档进行配置，然后通过IP地址和登录密码进行连接。效果如右图。





## 5.分布式视觉仿真

### 5.5 接口测试实验—例程简介

- 实验例程见“8.RflySimVision\0.ApiExps\2-DistributedSimAPI\1.VisionAPISTest”目录
- 关键配置项为本讲2.4小节的Config.json配置文件中，SendProtocol[8]向量。  
其中，SendProtocol[0]表示4种传图协议，  
**0**: UE4将图像写入共享内存，本机其他程序可通过内存读取图片，如果远端电脑需要图片，可以再将图片转发出去。**特点**: 共享内存方式使得本机可以高速地去读取图片，但是远端计算机读图时，需要经过额外中转，计算量和延迟较大。
- **1**: UDP直传远端电脑（通过SendProtocol [1-4]指定的IP），并采用png压缩方式进行发送压缩和接收解压。**特点**: 延迟小，PNG无损压缩，压缩比低但质量好。
- **2**: UDP直传远端电脑，图片不压缩。**特点**: 延迟最小、计算量小，网络压力大。
- **3**: UDP直传远端电脑，并采用jpg压缩。**特点**: 网络压力小，延迟和计算量适中，但图片是有损压缩，稍微损失一些精度，但是满足无人机实时视觉控制需求。
- **注**: 免费版只能使用**0**模式，完整版推荐使用**3**模式（后续例程默认选用）。

```
"DataCheckFreq":200,  
"SendProtocol": [0,127,0,0,1,9999,0,0],  
"CameraFOV":90,
```



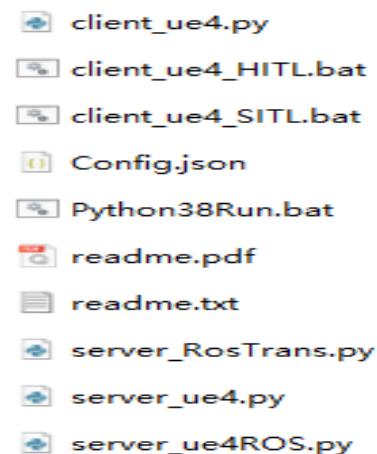
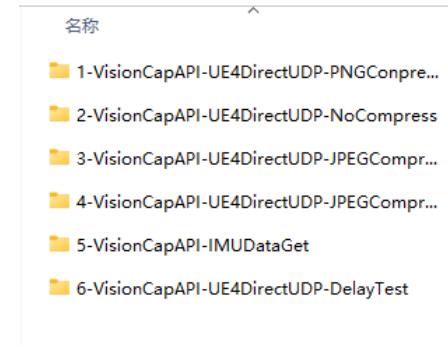
## 5.分布式视觉仿真

### 5.5 接口测试实验—例程文件结构

- 实验例程见“8.RflySimVision\2.AdvExps\e0\_AdvApiExps\9.VisionAPISTest”
- 从右图可以看出，6个例程分别对应了4种图像传输模式  
即共享内存、PNG直传、无压缩直传、JPEG直传、

IMU数据获取、取图与传输极限延迟实验。

- 例程中包含如下几类文件：
- **client\_\*\*.bat**文件：开启软/硬件在环仿真，本例程中飞机数量设为1即可。
- **Config.json**文件：配置相机参数，包括几个相机、相机位置、相机类型（RGB、深度等）
- **client\_ue4.py**文件：开启传图的脚本
- **server\_ue4.py**文件：开启图片接收并控制
- **Python38Run.bat**：平台Python快捷方式
- **\*\*ROS.py**文件：支持ROS的例程和接口





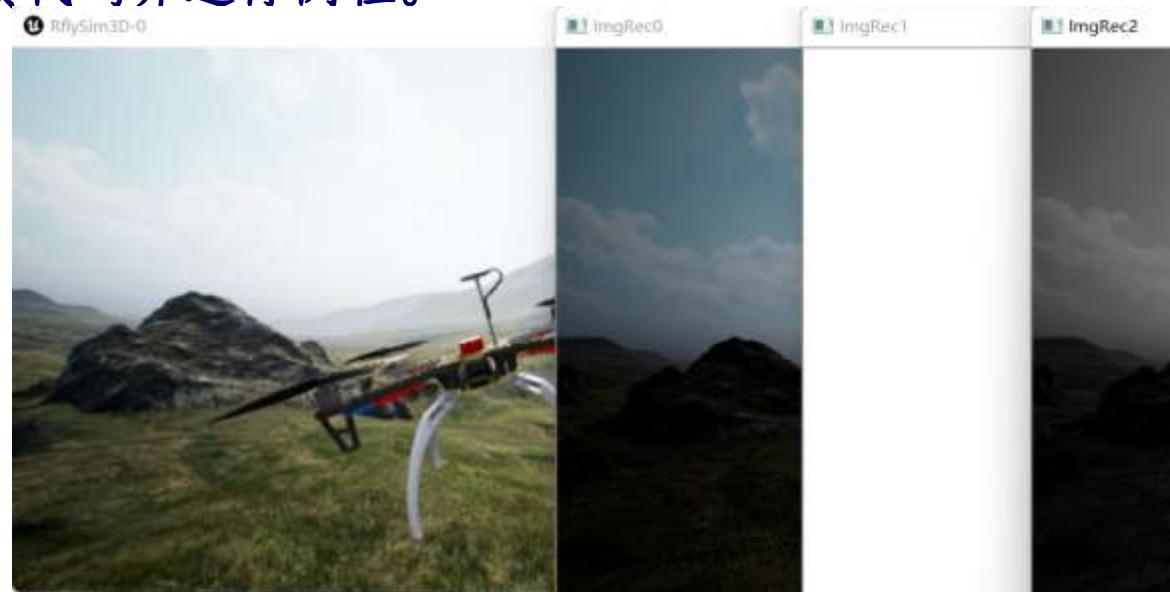
## 5.分布式视觉仿真

### 5.6 接口测试实验—单电脑实验

- 四种模式的测试例程，在一台电脑上，都可以采用如下方式进行测试
- 双击运行“client\_ue4\_SITL.bat”开启一个飞机的软件在环仿真
- 双击运行“Python38Run.bat”开启平台Python环境，并输入命令“python client\_ue4.py”开启图片请求与图片转发。（VS Code只能运行一个程序，留给server）
- 用VS Code打开“server\_ue4.py”，阅读代码并运行例程。
- 四个实验飞机的运动是相同的，都是起飞到10m后，开始画圈飞行。
- 所有server程序运行后，会获取三个图像数据，分别是RGB、深度和灰度。
- 模式0下，client运行能预览到三幅图，其他模式不经过python中转，无图像。
- 深度图有距离限制，仅能在近距离物体观察到结果，实验中起飞后为白色。

```
C:\WINDOWS\system32\cmd.exe - python client_ue4.py
Put your python scripts 'XXX.py' into the folder 'C:\PX4PSP\RflySimAPIs\PythonVisionAPI\4-DistributedSimAPI\1-VisionAPIsTest\0-VisionCapAPI-SharedMemory'
Use the command: 'python XXX.py' to run the script with Python

C:\PX4PSP\RflySimAPIs\PythonVisionAPI\4-DistributedSimAPI\1-VisionAPIsTest\0-VisionCapAPI-SharedMemory>python client_ue4.py
Got 3 vision sensors from json
Sensor req success from UE4.
Start Transfer Img
```





## 5.分布式视觉仿真

### 5.6 接口测试实验—双电脑实验

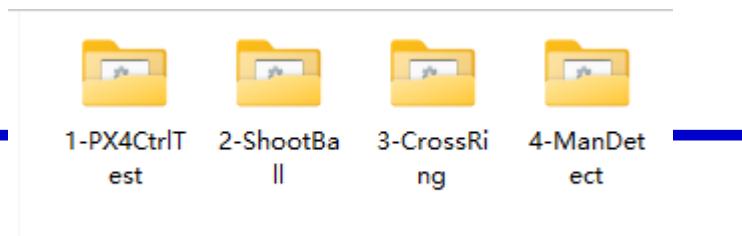
- 流程与单台电脑基本相同区别在于，client要增加远端电脑（Windows电脑、Linux电脑或树莓派均可）的IP，server程序要运行在远端电脑上。步骤如下：
- 通过路由器观察到本机IP地址（例如192.168.3.80）和远端电脑的IP地址（例如192.168.3.81），记录下IP地址。
- 双击运行“client\_ue4\_SITL.bat”开启一个飞机的软件在环仿真。**注：**例程默认使用了广播方式进行MAVLink数据传输，为提升效率，可将“SET IS\_BROADCAST=1”语句改成“SET IS\_BROADCAST=192.168.3.81”，这里需填入远端电脑的IP地址。
- 用VS Code打开“client\_ue4.py”，取消下面语句的注释“# vis.RemotSendIP=”，并将IP地址设置为远端电脑IP地址，例如“vis.RemotSendIP = '192.168.3.81'”
- 将文件夹所有文件拷贝到另一台电脑上，通过Python环境或VS Code打开“server\_ue4.py”并运行，可以接收到图片并显示。**注：**为了提高通信效率，可将其中的“255.255.255.255”广播地址，改成“192.168.3.80”的主机地址。
- **注：**“VisionCaptureApi.py”中取消注释“#print('Img', idx)”可观察到每幅图片的类别和接收时间戳，通过本数据可以分析和测试同步性和丢包率。



## 5.分布式视觉仿真

### 5.7 单飞机视觉控制实验

- 进入8.RflySimVision\3.CustExps\2-DistributedSimDemos\e1\_OneVehilceCtrls目录，可以看到之前的视觉API、撞击小球、穿环、双目人脸识别的例程，在这里通过分布式的方法进行了复现。
- 实验过程与前文基本相同，实验效果与第3节相同。
- 若进行单机实验，直接先运行bat，再Python运行client，最后运行server即可。
- 若进行联机实验，直接先记录两主机IP，再运行bat，再VS Code打开client并设置主机IP，最后将所有文件拷贝到远程主机，接着运行server即可。
- 注意：如果局域网内电脑不多，可以不去路由器查看电脑IP地址，直接修改client，并将远程主机IP地址设置为“255.255.255.255”的广播地址。
- 注意：推荐在路由器中设置电脑或嵌入式主机的IP地址为静态IP，这样每次重启后IP地址不会改变，不需要频繁查询IP地址。
- 注意：推荐使用远程访问的方式在一台电脑上操作实验。





## 5.分布式视觉仿真

### 5.8 多飞机视觉控制实验—总体介绍

- 进入“8.RflySimVision\3.CustExps\2-DistributedSimDemos\e2\_MultipleVehicles”目录，可以看到多个实验例程，文件名的含义如下：
- TransMode表示传输模式，0为共享内存，3为JPEG直传。
- SITL表示使用PX4软件在环仿真，需要运行SITL\*\*.bat来启动仿真闭环；HITL则表示硬件在环仿真。
- UDP表示飞控和Server之间采用UDP方式通信；而Serial表示飞控和Server之间需要用数传模块通过串口通信。
- Local表示实验在单台电脑上可以运行；Remote表示实验需要至少两台电脑，且需要指定IP地址。
- “\_”的后缀表示飞机数量，无后缀默认是1个飞机3个相机，2V4C表示2个飞机共4个相机，也就是每飞机2相机。
- AllSourceFile文件夹是所有例程的模版源文件。

更详细的操作流程可以参考：  
8.RflySimVision\3.CustExps\2-DistributedSimDemos\e2\_MultipleVehicles中的Readme.pdf文件

- 1-SITLUdpDemo\_TransMode0\_Local
- 2-SITLUdpDemo\_TransMode3\_Local
- 3-HITLUdpDemo\_TransMode3\_Local
- 4-HITLSerialDemo\_TransMode3\_Local
- 5-SITLUdpDemo\_TransMode3\_Remote
- 6-HITLUdpDemo\_TransMode3\_Remote
- 7-HITLSerialDemo\_TransMode3\_Remote
- 8-SITLUdpDemo\_TransMode3\_Local\_2Vehi...
- 9-SITLUdpDemo\_TransMode3\_Remote\_2V...
- 10-HITLSerialDemo\_TransMode3\_Remote\_...
- AllSourceFile
- Readme.pdf



## 5.分布式视觉仿真

### 5.8 多飞机视觉控制实验—例程介绍

- 进入“1-SITLUpdDemo\_TransMode0\_Local”目录，可以看到多个文件，用于一键生成所需飞机和相机的例程。
- Config.json和之前的配置文件相同，定义了多个相机的参数。注：这里应该定义每个飞机最多支持装载的相机，在config.xlsx中可选启用其中的前几个。
- config.xlsx中定义了Windows主机数据，Linux主机数量（和飞机数量相同）和IP地址，每个飞机位置、每个飞机上相机数量等。
- ConfigWrite.m可以根据Config.xlsx中定义的分布式仿真构架，自动生成任意多飞机的分布式仿真例程，可选多种仿真模式。例程的模版文件来自上层目录的“AllSourceFile”文件夹。
- 实验流程也非常简单：1) 用MATLAB定位到“ConfigWrite.m”文件所在目录，然后右键运行，即可生成VisionDemo\*的文件夹，其中\*表示飞机总数；2) 在仿真电脑运行WindowsPC\*的bat启动脚本，再运行client传图程序；3) 将LinuxNXX\*拷贝到远端主机，运行server即可。注：支持多电脑和主机组网。



1-SITLUpdDemo_TransMo... > VisionDemo1 >	
名称	修改日期
LinuxNXX1	2022/1/17 3:46
WindowsPC1	2022/1/17 3:46



## 5.分布式视觉仿真

### 5.8 多飞机视觉控制实验—Config.xlsx用法介绍

HIL or SIL, UDP or Serial, COM Name ,Baud Num	硬件或软件在环(HIL/SIL), UDP或串口通信(UDP/Serial), 串口号(仅HIL填写), 波特率(仅HIL填写)	SIL	UDP
WindowsPCIList	Windows电脑的IP地址	127.0.0.1	
VehicleNumOnPC	各台Windows电脑上的飞机数量	1	
NXXIPLList	各嵌入式电脑NXX的IP列表, 数量应该与飞机数量相同	127.0.0.1	
CameraNumList	每台飞机上相机数量	3	
VehicleXPosList(m)	飞机的X坐标列表, 单位m	0	
VehicleYPosList(m)	飞机的Y坐标列表, 单位m	0	
VehicleYawList(degree)	飞机的偏航角度列表, 单位弧度	0	
UE4_MAP	地图名字或序号	GrassLands	
PX4SitrFrame	软件在环时, 设置PX4内部机架Airframe类型	iris	
DLLModel	DLL模型的名字或序号, 默认多旋翼选0, 固定翼等需要选DLL 是否开启同步启动 0: 不开启同步启动 1: 开启同步启动, 最后一个飞机命令执行后触发前面飞机 2: 最后一个飞机脚本运行后, 依次触发前面飞机; 本选项第2列可设置延迟触发的时间 (s)	0	
isEnableSyncStart		0	
isEnableIMUSend	是否开启IMU数据发送	0	



## 5.分布式视觉仿真

### 5.8 多飞机视觉控制实验—Config.xlsx用法介绍

- 以HITLUdpDemo\_Local\_1V3C为例
- 需要使用HITL硬件在环，因此需要准备一个Pixhawk，连接到电脑上。
- 使用Local模式，电脑和远端主机IP地址都设为127.0.0.1
- 一台电脑一个飞机，因此WindowsPCIPList只有1列，每个飞机有3个相机。
- 实验步骤：运行WindowsPC1的bat和client，再去运行LinuxNXX1下的server即可。

HIL or SIL, UDP or Serial, COM Name ,Baud Num	HIL	UDP
WindowsPCIPList	127.0.0.1	
VehicleNumOnPC		1
NXXIPList	127.0.0.1	
CameraNumList		3
VehicleXPosList(m)		0
VehicleYPosList(m)		0
VehicleYawList(degree)		0



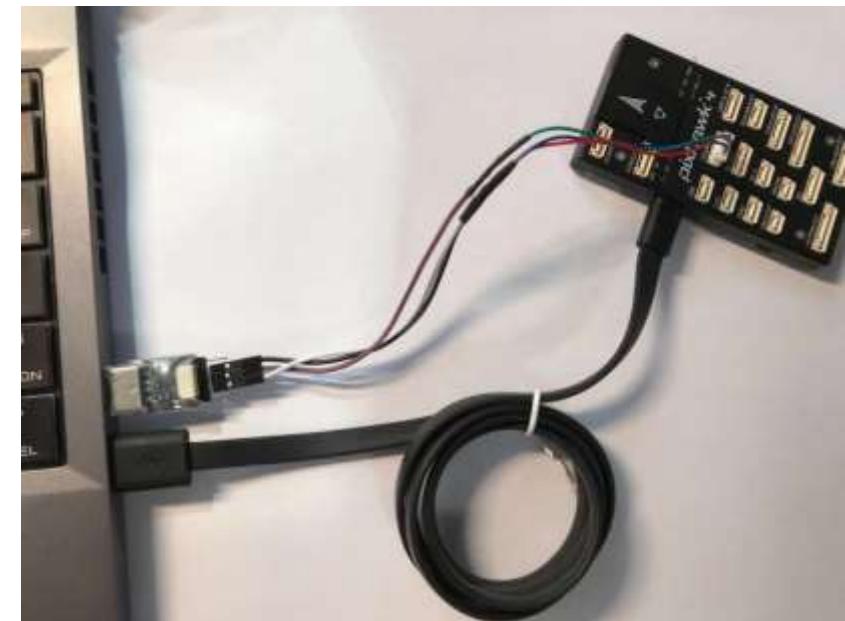


## 5.分布式视觉仿真

### 5.8 多飞机视觉控制实验—Config.xlsx用法介绍

- 以HITLSerialDemo\_Local\_1V3C为例，需要准备一个数传连接飞控TELEM1和电脑USB口，也就是说电脑需要占用两个USB口，一个用于HITL仿真，一个用于server与PX4通信。
- 其中，下表中**COM14**是数传串口号，**57600**是数传波特率，可在QGC中设置。

HIL or SIL, UDP or Serial, COM Name ,Baud Num	HIL	Serial	COM14	57600
WindowsPCIPList	127.0.0.1			
VehicleNumOnPC		1		
NXXIPLList	127.0.0.1			
CameraNumList		3		
VehicleXPosList(m)		0		
VehicleYPosList(m)		0		
VehicleYawList(degree)		0		





## 5.分布式视觉仿真

### 5.8 多飞机视觉控制实验—Config.xlsx用法介绍

- 以SITLUpDemo\_Remote\_1V3C为例，需要准备：1台电脑、1台主机、1个路由（或交换机）、电源和若干网线。
- 下表中192.168.3.80为电脑IP，192.168.3.55为远端主机IP，需要根据个人网络配置修改。
- 实验时WindowsPC1目录里面的文件在电脑上运行，LinuxNXX1目录里面的文件拷贝到主机运行。

HIL or SIL, UDP or Serial, COM Name ,Baud Num	SIL	UDP
WindowsPCIPList	192.168.3.80	
VehicleNumOnPC	1	
NXXIPList	192.168.3.55	
CameraNumList		3
VehicleXPosList(m)		0
VehicleYPosList(m)		0
VehicleYawList(degree)		0





## 5.分布式视觉仿真

### 5.8 多飞机视觉控制实验—Config.xlsx用法介绍

- 以HITLUdpDemo\_Remote\_1V3C为例，需要准备：1台电脑、1台主机、1个飞控、1个路由（或交换机）、电源和若干网线。

HIL or SIL, UDP or Serial, COM Name ,Baud Num	HIL	UDP
WindowsPCIPList	192.168.3.80	
VehicleNumOnPC	1	
NXXIPLList	192.168.3.82	
CameraNumList	3	
VehicleXPosList(m)	0	
VehicleYPosList(m)	0	
VehicleYawList(degree)	0	



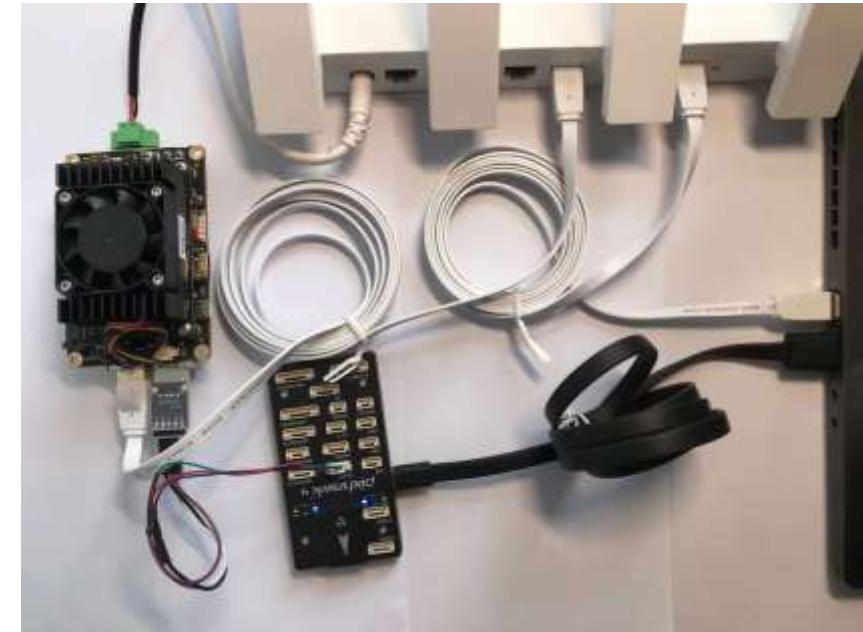


## 5.分布式视觉仿真

### 5.8 多飞机视觉控制实验—Config.xlsx用法介绍

- 以HITLSerialDemo\_Remote\_1V3C为例，需要准备：1台电脑、1台主机、1个飞控、1个数传、1个路由（或交换机）、电源和若干网线。

HIL or SIL, UDP or Serial, COM Name ,Baud Num	HIL	Serial	/dev/ttyUSB0	57600
WindowsPCIList	192.168.3.80			
VehicleNumOnPC		1		
NXXIPLList	192.168.3.82			
CameraNumList		3		
VehicleXPosList(m)		0		
VehicleYPosList(m)		0		
VehicleYawList(degree)		0		



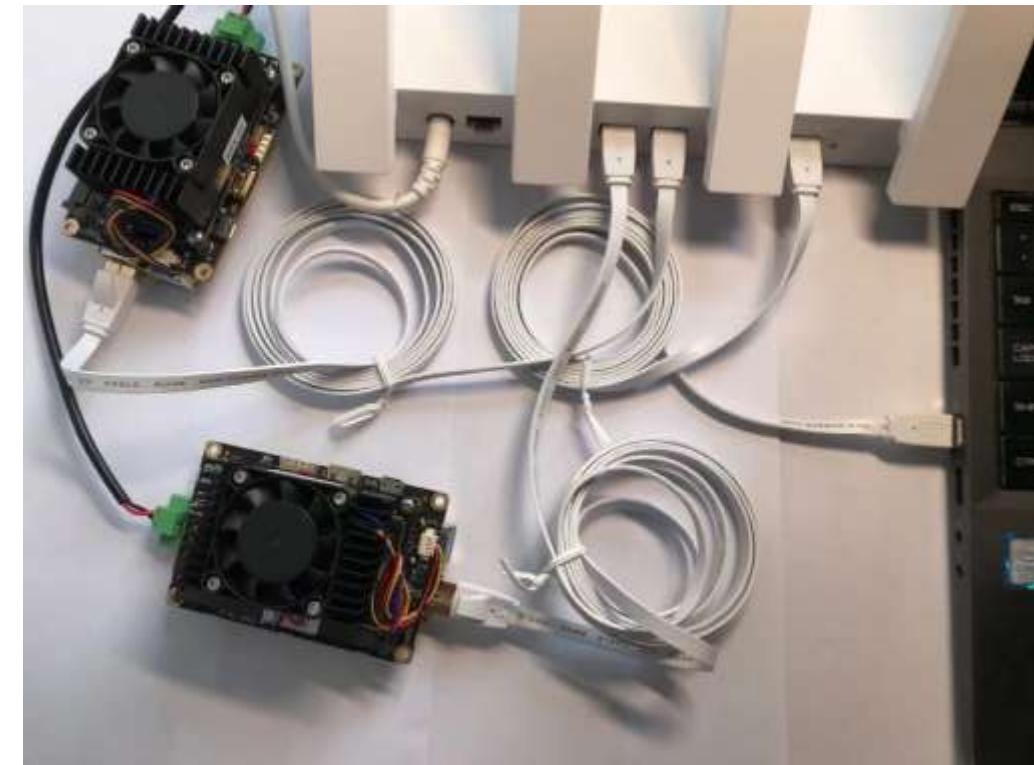


## 5.分布式视觉仿真

### 5.8 多飞机视觉控制实验—Config.xlsx用法介绍

- 以SITLUpdDemo\_Remote\_2V4C为例，需要准备：1台电脑、2台主机、1个路由（或交换机）、电源和若干网线。

HIL or SIL, UDP or Serial, COM Name		
,Baud Num	SIL	UDP
WindowsPCIPList	192.168.3.80	
VehicleNumOnPC		2
NXXIPLList	192.168.3.81	192.168.3.82
CameraNumList	2	2
VehicleXPosList(m)	0	1
VehicleYPosList(m)	0	0
VehicleYawList(degree)	0	0





## 5.分布式视觉仿真

### 5.8 多飞机视觉控制实验—Config.xlsx用法介绍

- 以HITLSerialDemo\_Remote\_2V4C为例，需要准备：1台电脑、2台主机、2个飞控、2个数传、1个路由（或交换机）、电源和若干网线。

HIL or SIL, UDP or Serial, COM Name ,Baud Num	HIL	Serial	/dev/ttyUSB0	57600
WindowsPCIPList	192.168.3.80			
VehicleNumOnPC	2			
NXXIPLList	192.168.3.81	192.168.3.82		
CameraNumList	2	2		
VehicleXPosList(m)	0	1		
VehicleYPosList(m)	0	0		
VehicleYawList(degree)	0	0		

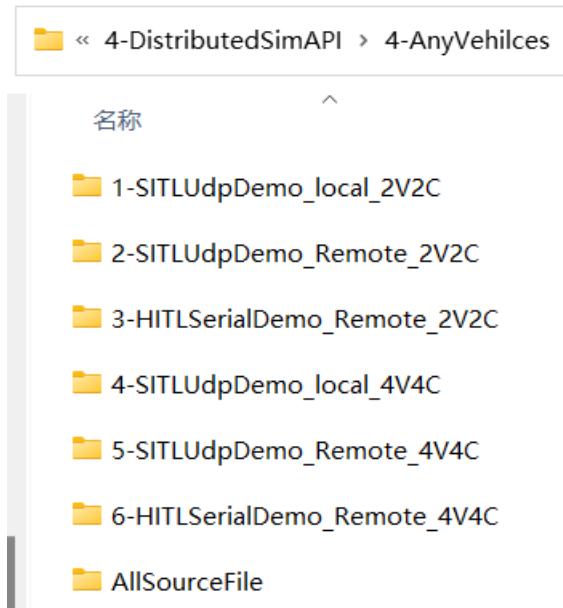
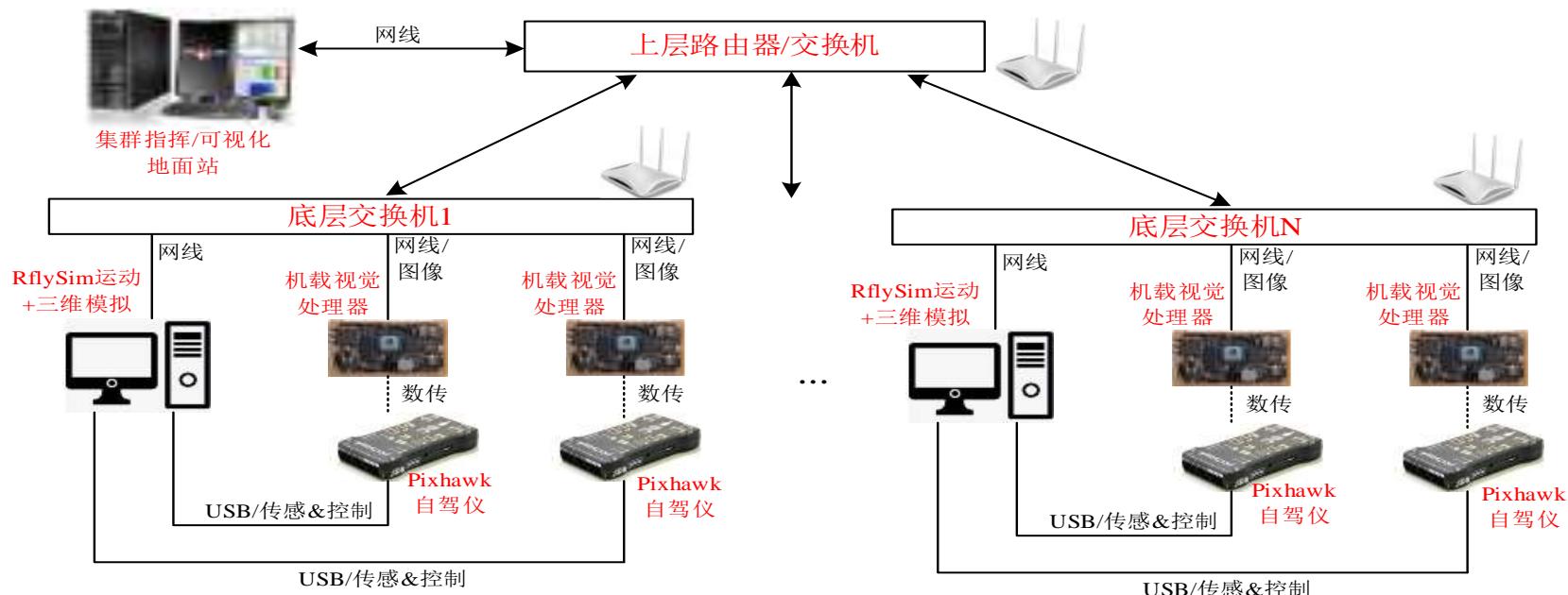




## 5.分布式视觉仿真

### 5.9 多电脑多飞机视觉控制实验—总体介绍

- 进入“RflySimAPIs\8.RflySimVision\3.CustExps\2-DistributedSimDemos\e3\_AnyVehilces”目录，可以看到多电脑多飞机的例程，理论上支持局域网内任意数量的电脑和主机组网，但是一定要采用下图的构架，确保使用指定IP方式让图像只在底层交换机内传播。





## 5.分布式视觉仿真

### 5.9 多电脑多飞机视觉控制实验—Config.xlsx用法介绍

- 以SITLUpdDemo\_Remote\_4V4C（每台电脑2个飞机，每飞机1个相机）为例，需要准备：2台电脑、4台主机、2个路由（或交换机）、电源和若干网线。
- 需要填入6个IP地址，实验现象为飞机依次穿环，且相互可见（遮挡可能导致穿环失败）。

HIL or SIL, UDP or Serial, COM Name ,Baud Num	硬件或软件在环(HIL/SIL), UDP或串口通信(UDP/Serial), 串口号(仅HIL填写), 波特率(仅HIL填写)	SIL	UDP			
WindowsPCIPList	Windows电脑的IP地址	192.168.3.80	192.168.3.79			
VehicleNumOnPC	各台Windows电脑上的飞机数量	2	2			
NXXIPLList	各嵌入式电脑NXX的IP列表, 数量应该与飞机数量相同	192.168.3.81	192.168.3.82	192.168.3.83	192.168.3.84	
CameraNumList	每台飞机上相机数量	1	1	1	1	
VehicleXPosList(m)	飞机的X坐标列表, 单位m	-0.5	-0.5	0.5	0.5	
VehicleYPosList(m)	飞机的Y坐标列表, 单位m	-0.5	0.5	-0.5	0.5	
VehicleYawList(degree)	飞机的偏航角度列表, 单位弧度	0	0	0	0	
UE4_MAP	地图名字或序号	VisionRing				
PX4SitzlFrame	软件在环时, 设置PX4内部机架Airframe类型	iris				
DLLModel	DLL模型的名字或序号, 默认多旋翼选0, 固定翼等需要选DLL	0				
isEnableSyncStart	是否开启同步启动 0: 不开启同步启动 1: 开启同步启动, 最后一个飞机命令执行后触发前面飞机 2: 最后一个飞机脚本运行后, 依次触发前面飞机; 本选项第2列可设置延迟触发的时间 (s)	0				
isEnableIMUSend	是否开启IMU数据发送	0				



## 5.分布式视觉仿真

### 5.9 多电脑多飞机视觉控制实验—Config.xlsx用法介绍

- 以HITLSerialDemo\_Remote\_4V4C（每台电脑2个飞机，每飞机1个相机）为例，需要准备：2台电脑、4台主机、4个飞控、4个数传、2个路由（或交换机）、电源和若干网线。
- 可启用同步开始仿真功能，最后一个主机运行server时，依次触发前面阻塞的server程序，实现飞机同时或者依次穿环的现象。

HIL or SIL, UDP or Serial, COM Name ,Baud Num	硬件或软件在环(HIL/SIL), UDP或串口通信(UDP/Serial), 串口号(仅HIL填写), 波特率(仅HIL填写)	HIL	Serial	/dev/ttyUSB0	57600
WindowsPCIPList	Windows电脑的IP地址	192.168.3.80	192.168.3.79		
VehicleNumOnPC	各台Windows电脑上的飞机数量	2	2		
NXXIPLList	各嵌入式电脑NXX的IP列表，数量应该与飞机数量相同	192.168.3.81	192.168.3.82	192.168.3.83	192.168.3.84
CameraNumList	每台飞机上相机数量	1	1	1	1
VehicleXPosList(m)	飞机的X坐标列表，单位m	-0.5	-0.5	0.5	0.5
VehicleYPosList(m)	飞机的Y坐标列表，单位m	-0.5	0.5	-0.5	0.5
VehicleYawList(degree)	飞机的偏航角度列表，单位弧度	0	0	0	0
UE4_MAP	地图名字或序号	VisionRing			
PX4SitrFrame	软件在环时，设置PX4内部机架Airframe类型	iris			
DLLModel	DLL模型的名字或序号，默认多旋翼选0，固定翼等需要选DLL	0			
isEnableSyncStart	是否开启同步启动 0: 不开启同步启动 1: 开启同步启动，最后一个飞机命令执行后触发前面飞机 2: 最后一个飞机脚本运行后，依次触发前面飞机；本选项第2列可设置延迟触发的时间 (s)	2	5		
isEnableIMUSend	是否开启IMU数据发送	1			



## 小结

---

- 本讲主要对飞行控制算法的开发课程进行讲解，分为基础实验和进阶实验两部分，使各位学员能够尽快熟悉多旋翼的理论设计、RflySim平台仿真、物理真机控制等开发流程。
- 基础实验是基于RflySim平台软件在环和硬件在环仿真流程学习为主，进阶实验是从多旋翼理论设计和建模实验→估计实验→控制实验→决策实验的学习路线进行教学。

如有疑问，请到<https://doc.rflysim.com/>查询更多信息。



RflySim更多教程



扫码咨询与交流



飞思RflySim技术交流群



---

谢谢！