



Development and Practice of Intelligent Unmanned Cluster System Full Stack

Development Case Based on RflySim Toolchain

Lesson 8 Visual Perception and Obstacle Avoidance Decision



Outline

1. General introduction
2. The base interface uses
3. Visual control example
4. Visual AI is advanced
5. Distributed visual simulation

The video address of this PPT public welfare course is:
Station B:

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



Mobile phone
scanning code
to watch



1. General introduction

1.1 Visual Routine Folder

- The current vision control routines for the platform are in the directory [PX4PSP\RflySimAPIs\8.RflySimVision](#) As shown in the figure on the right, it contains four folders, namely:
 - **0. Api Exps: Free Edition Routine**
 - **1. Basic Exps: Free Edition Routine**
 - **2. AdvExps: Free Edition Routine**
 - **3. CustExps: Full Version Routine**
- **API. PDF: Chapter 8 introduces various interfaces and some basic knowledge**
- **Readme. PDF: Introduction to Each Routine Table of Content**
- **Intro. PDF: Introduction to Chapter 8**

Note: It is recommended that you use the Python 38 environment that comes with the platform to run the routines. If you use another Python environment, be sure to install the following components:
pip3 install pymavlink pyserial opencv

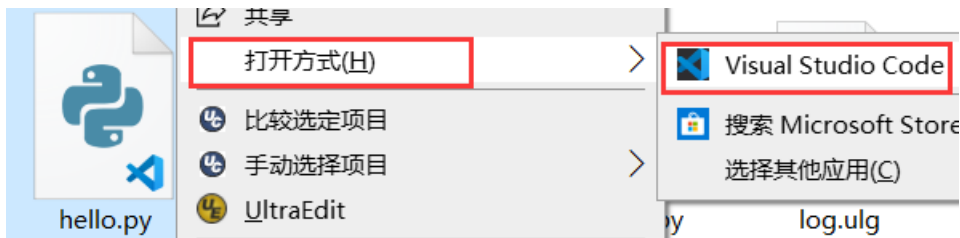
0.ApiExps	2024/1/29 9:06	文件夹	
1.BasicExps	2024/1/26 18:40	文件夹	
2.AdvExps	2024/1/26 18:40	文件夹	
3.CustExps	2024/1/26 18:40	文件夹	
API.pdf	2023/12/29 14:40	Microsoft Edge PD...	3,649 KB
Intro.pdf	2024/1/19 15:01	Microsoft Edge PD...	659 KB
Readme.pdf	2024/1/22 18:33	Microsoft Edge PD...	751 KB



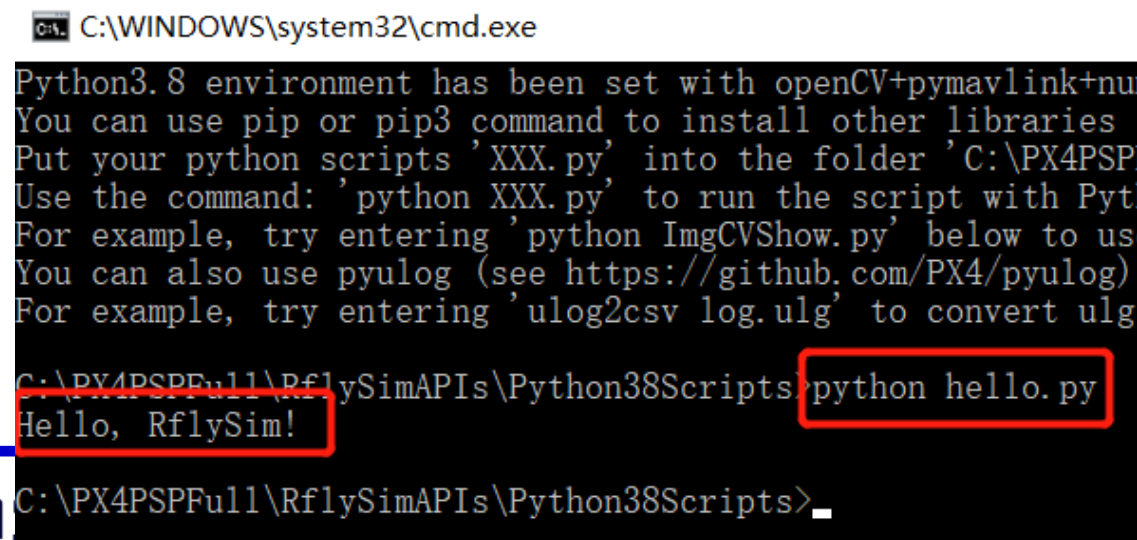
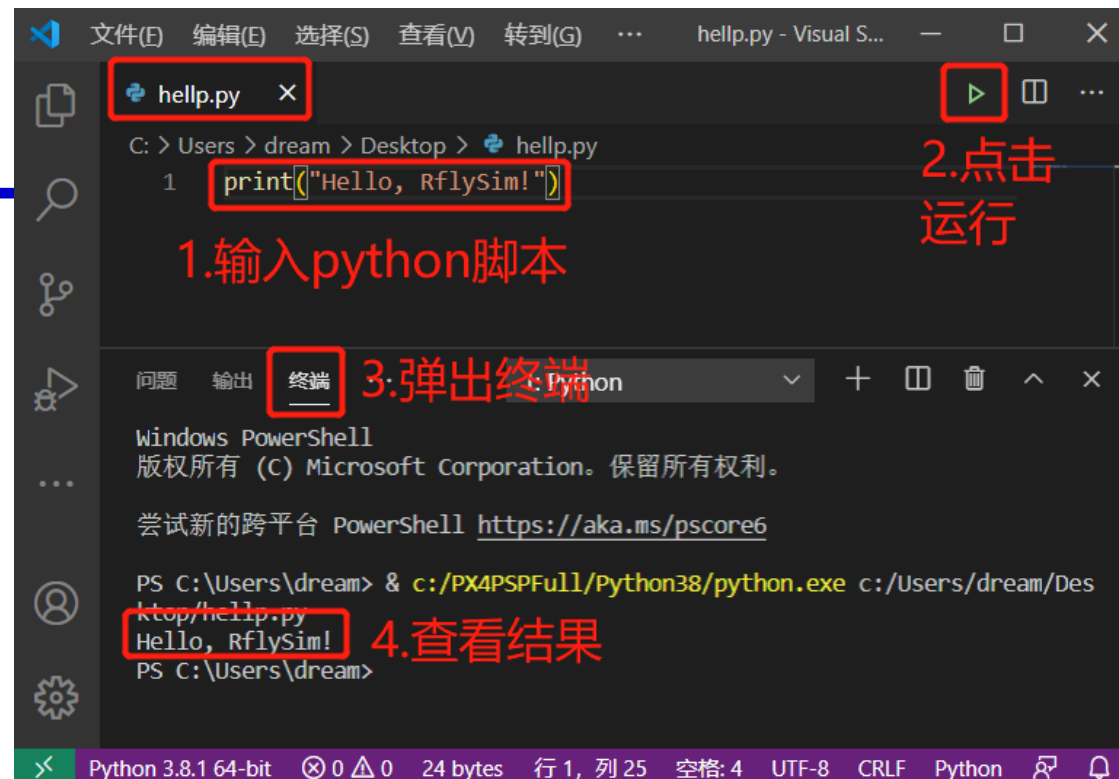
1. General introduction

1.2 Python Program Running

- In the "RflySimAPIs \ Python38 Scripts" folder, create a new txt file and rename it to hello. Py.
- Right click the file, open it with VS Code, and type the following code in it
`print("Hello, RflySim!")`
- Run in VS Code, as shown in the figure on the right.



- Similarly, double-click the Python38Env desktop shortcut or "RflySimAPIs \ Python38Env.bat" to view the running results as shown below.





1. General introduction

1.3 Python Syntax Learning

- Visit: <https://runoob.com/python3> Learn the basics and programming methods of Python 3 (similar to MATLAB language) Easy to get started.
- Note: Python2 has stopped updating at present. It is recommended that you learn Python3 directly for development.
- Python 3 uses UTF-8 encoding by default, so Chinese characters are natively supported.
- Python uses indentation to distinguish levels of code, so write code with care about indentation. (Guides, indent-rainbow and other plug-ins can be installed in VS Code for assistance)

RUNOOB.COM

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

Python 3 教程

Python 3 教程

Python 的 3.0 版本, 常被称为 Python 3000, 或简称 Py3k. 相对于 Python 的早期. 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 Python 2.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. General introduction

1.4 OpenCV Learning

- **OpenCV is an open source, cross-platform computer vision and machine learning software library with a wide range of applications.**
- **Official tutorial website:**
- <https://docs.opencv.org/4.0.0/>
- **Recommended Chinese translation document website:**
- <https://github.com/makelove/OpenCV-Python-Tutorial>

OpenCV 4.0.0
Open Source Computer Vision

Main Page | Related Pages | Modules | Namespaces | Classes | Files | Examples

Java documentation

OpenCV modules

- 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. General introduction

1.5 MAVLink communication protocol

- **Pymavlink is the Python version of MAVLink communication protocol, which has been pre-installed in the platform Python environment. Through it, it is convenient to communicate with Pixhawk real machine through serial port, UDP, TCP and other ways for top-level control.**
- **The official document URL is as follows:**
https://mavlink.io/en/mavgen_python
- **Learn how to use it on your own**
- **Note: Pymavlink is just a convenient library function. If you have higher customization requirements, you need to learn how to use the MAVLink protocol.**
- **The MAVLink message protocol can be read at the following address**
<https://mavlink.io/en/messages/common.html>

Using Pymavlink Libraries (mavgen)

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

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

This topic explains how to get and use the *Pymavlink* MAVLink Python libraries (generated by mavgen).



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



If you're writing a MAVLink application to communicate with an autopilot you may want to use [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 `pip install pymavlink`



Outline

1. General introduction
2. The base interface uses
3. Visual control example
4. Visual AI is advanced
5. Distributed visual simulation



2. The base interface uses

2.1 UAV Control Interface-PX4MavCtrlV4.py

- Open the PX4MavCtrlV4.py file in the "[RflySimSDK\ctrl](#)" directory, and you can see the interface file shown in the right figure.
- This interface defines a PX4MavCtrl class used to implement interfaces such as MAVLink message sending and receiving, RflySim3D scene control, and Offboard control of Pixhawk/PX4.
- This interface can send a message to CopterSim/PX4 to control the aircraft, or send it to RflySim3D to control the scene or request for drawing.
- This interface file communicates with RflySim3D through UDP, controls the aircraft through UDP-CopterSim-MAVLink-PX4, or controls the aircraft directly through "Serial-MAVLink-PX4".

```
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 PX4MavCtrl:

    """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
    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
    使用port+1端口收和port端口发,例如,1号飞机是20101端口收,20100端口发,与CopterSim
    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,按平台规则,port=20100+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对应服务器端口号self.redisPort = port。如果port=0,则self.redisPort = 0

    接口示例:
    UDP模式
    PX4MavCtrl(1) # 默认IP
    PX4MavCtrl(1,'192.168.31.24') # 指定IP,用于远程控制

    串口模式
    PX4MavCtrl(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 PX4MavCtrl('COM:baud','IP:CopterID') format
    Windows use format PX4MavCtrl('COM3') or PX4MavCtrl('COM3:115200') for Pixhawk USB
    Windows use format 'COM4:57600' for Pixhawk serial port connection
    Linux use format PX4MavCtrl('/dev/ttyUSB0') or PX4MavCtrl('/dev/ttyUSB0:115200') f
    PX4MavCtrl('COM3:115200:2'): the second input is the CopterID, in this case CopterID

    For real flight
    PX4MavCtrl(port,'IP:CopterID:Flag'), Flag set to 1 to enable real flight com mode
    for example PX4MavCtrl(15551,'192.168.1.123:1:1') to set IP=192.168.1.123, port=15551

    """
```



2. The base interface uses

2.1 UAV Control Interface-Internal Principle of PX4 MavCtrlV4

Class PX4_CUSTOM_MAIN_MODE: # PX4 Main Module Enumeration Variables to set the mode

Class PX4_CUSTOM_SUB_MODE_AUTO: # PX4 Submodule Enumeration Variables

Class PX4MavCtrler: # The main communication interface class of RflySim, which can be connected by UDP or serial port.

Def InitMavLoop: # Enable the MAVLink receiving thread to receive and update MAVLink messages at any time

Def sat: # A saturation function that controls the clipping of a variable

Def SendMavCmdLong: # Send COMMAND_LONG message for MAVLink message

Def sendMavOffboard Cmd: # Send the Offboard command to the flight control to make it enter the Offboard mode

Def sendMavOffboard API: # Update the data of the offboard message (the data will be sent at a certain frequency)

Def SendVelNED: # Send Earth coordinate system velocity commands

```
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. The base interface uses

2.1 UAV Control Interface-Internal Principle of PX4 MavCtrlV4

Def SendVelFRD: # Send body speed

Def SendPosNED: # Send the NED position to let the aircraft fly to the specified position (relative to the unlocking point)

Def initOffboard: # Initialize Offboard mode

Def endOffboard: # End Offboard mode

Def sendMavSetParam: # Send a MAVLink message to change the Pixhawk parameter

Def SendHILCtrlMsg: # Send the rfly _ MSG message to the flight control (see Section 4.3 of Lesson 3)

Def SendMavArm: # Send unlock command

Def SendRcOverride: # Send and simulate remote control signals

Def sendMavManualCtrl: # Send and simulate the normalized remote control signal

Def SendSetMode: # Send and set Pixhawk mode

Def stopRun: # Stop running MAVLink data receiving thread

Def getMavMsg: # Update the data received by MAVLink

For the detailed definition of each function, you can read the internal source code implementation of PX4MavCtrlV4.py.



2. The base interface uses

2.1 UAV Control Interface-Basic Use Example

“[PX4PSP\RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\](#)
[e1_PX4MavCtrlAPITest\PX4MavCtrlAPITest.py](#)”

Is a Python example used by the interface. The specific code is parsed as follows:

Create a new MAVLink communication instance. The CopterSim interface is the 20100.

mav = PX4MavCtrl.PX4MavCtrler(1)

RflySim3D style adjustment API: sendUE4Cmd function belongs to UE4CtrlAPI interface

style ue.sendUE4Cmd (cmd, windowID = -1), where cmd should enter a string

Send the window adjustment command to RflySim3D, cmd is the specific command string, windowID is the received window number (assuming that multiple RflySim3D windows are open at the same time), -1 means send to all windows

The # RflyChangeMapbyName command switches the map, followed by the map name, which switches all open windows to the Grasslands map

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 sendin
```

```
mav = PX4MavCtrl.PX4MavCtrler(1)
```

创建控制接口实例



2. The base interface uses

2.1 UAV Control Interface-Basic Use Example

```
# RflySim3D generates 3D objects and controls the pose and attitude API: sendUE4Pos function, belonging to UE4CtrlAPI interface
# Style ue.sendUE4Pos (CopterID, VehicleType, RotorSpeed, PosM, AngEulerRad, windowsID)
ue.sendUE4Pos(100,30,0,[2.5,0,-8.086],[0,0,math.pi])
# Send to RflySim3D and generate a 3D object, where: object ID is CopterID = 100;
# Aircraft Type VehicleType = 30 (person); Rotor Speed RotorSpeed = 0 RPM; Position Coordinates PosM = [2.5,0, -8.086] m
# Aircraft attitude angle AngEulerRad = [0, 0, math. Pi] rad (rotate 180 degrees to face the aircraft), default receiving window ID = -1 (send to all opened RflySim3D programs)
# VehicleType: 3 Quadrotor, 5/6 Hexacopter, 30 Figure, 40 Calibrated Checkerboard, 50/51 Vehicle, 60 Ball Light, 100 Flying Wing Fixed Wing, 150/152 Ring Square Target
# RflyChange3D Model command followed by Airplane ID + Desired Style, where the style of Desired 12 is a walking person
ue.sendUE4Cmd(b'RflyChange3DModel 100 12')
# Send a message to make CopterID = 100 (the character just created) in all scenes become the style of a walking person.
```



2. The base interface uses

2.1 UAV Control Interface-Basic Use Example

The command `RflyChangeViewKeyCmd` means to simulate the shortcut key operation of `RflySim3D`, and the `B 1` shortcut key means to switch the focus to the object with `CopterID = 1`

It is set here to send to window 0, and other windows do not send.

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

The `# V 1` shortcut key switches the field of view to the first onboard field of view.

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

The `RflyCameraPosAng X y Z roll pith yaw` sets the position and direction of the camera relative to the center of the body. The default is 0.

Set the position of the front camera to [0.100] here.

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

`R. Setres 720x405 w` is the built-in command of UE4, which means to switch the resolution to `720x405`

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



2. The base interface uses

2.1 UAV Control Interface-Basic Use Example

Send a shortcut command to window 1 to switch focus to aircraft 1

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

Send the shortcut key control command to window 1. The N 1 shortcut key means to switch the visual angle to the ground fixed visual angle 1.

```
ue.sendUE4Cmd('RflyChangeViewKeyCmd N 1',1)
```

Set the current camera field angle to 90 degrees (90 degrees by default in RflySim3D). The field angle range is 0 to 180 degrees.

```
ue.sendUE4Cmd('RflyCameraFovDegrees 90',1)
```

Set the current camera position here to [-20 -9.7]

```
ue.sendUE4Cmd('RflyCameraPosAng -2 0 -9.7',1)
```

Enable MAVLink to monitor CopterSim data and update it in real time. The data is shown in the lower right figure.

```
mav.InitMavLoop()
```

Display location information received from CopterSim

```
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. The base interface uses

2.1 UAV Control Interface-Basic Use Example

Turn on Offboard mode

mav.initOffboard()

Send the desired position signal, fly to the target point 0, 0, -1.7, and the yaw angle is 0.

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

Send unlock command

mav.SendMavArm(True)

Send the expected speed signal, 0.2m/s downward descent, and the z-axis is positive.

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

Exit Offboard control mode

mav.endOffboard()

Exit MAVLink data receiving mode

mav.stopRun()

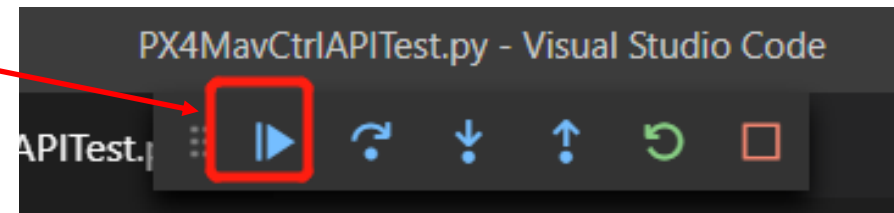
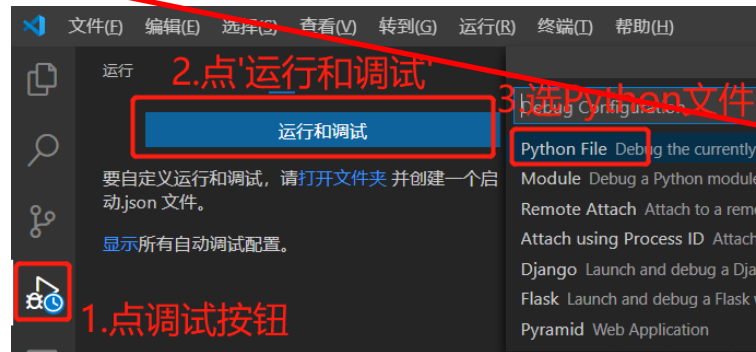


2. The base interface uses

2.1 UAV Control Interface-Lab 1: Interface Debugging Lab

- In Windows Explorer, navigate to the “PX4PSP\RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\e1_PX4MavCtrlAPITest” folder.
- Double-click the "PX4MavCtrlAPITest.bat" script to open the PX4 SITL simulation system for an aircraft.
- Use VS Code to open the "PX4MavCtrlAPITest.py" file. As shown in the right figure, click **the breakpoint (red dot)** in front of each key statement. Press the following figure to start the debugging mode. Click the **arrow button** in the lower right figure to execute the statements in turn.

```
9 #Create a new MAVLink communication instance, UDP
10 mav = PX4MavCtrl.PX4MavCtrl(1)
11
12
13 # sendUE4Cmd: RflySim3D API to modify scene display
14 # Format: ue.sendUE4Cmd(cmd>windowID=-1), where cmd
15 # Augument: RflyChangeMapbyName command means to st
16 ue.sendUE4Cmd('RflyChangeMapbyName Grasslands')
17 time.sleep(2)
18
19 # sendUE4Pos: RflySim3D API to generate 3D objects
20 # Formart: ue.sendUE4Pos(CopterID, VehicleType, Rot
21 ue.sendUE4Pos(100,30,0,[2.5,0,-8.086],[0,0,math.pi
22 # Send and generate a 3D object to RflySim3D, where
23 # Vehicle type VehicleType=30 (a man); RotorSpeed=
```

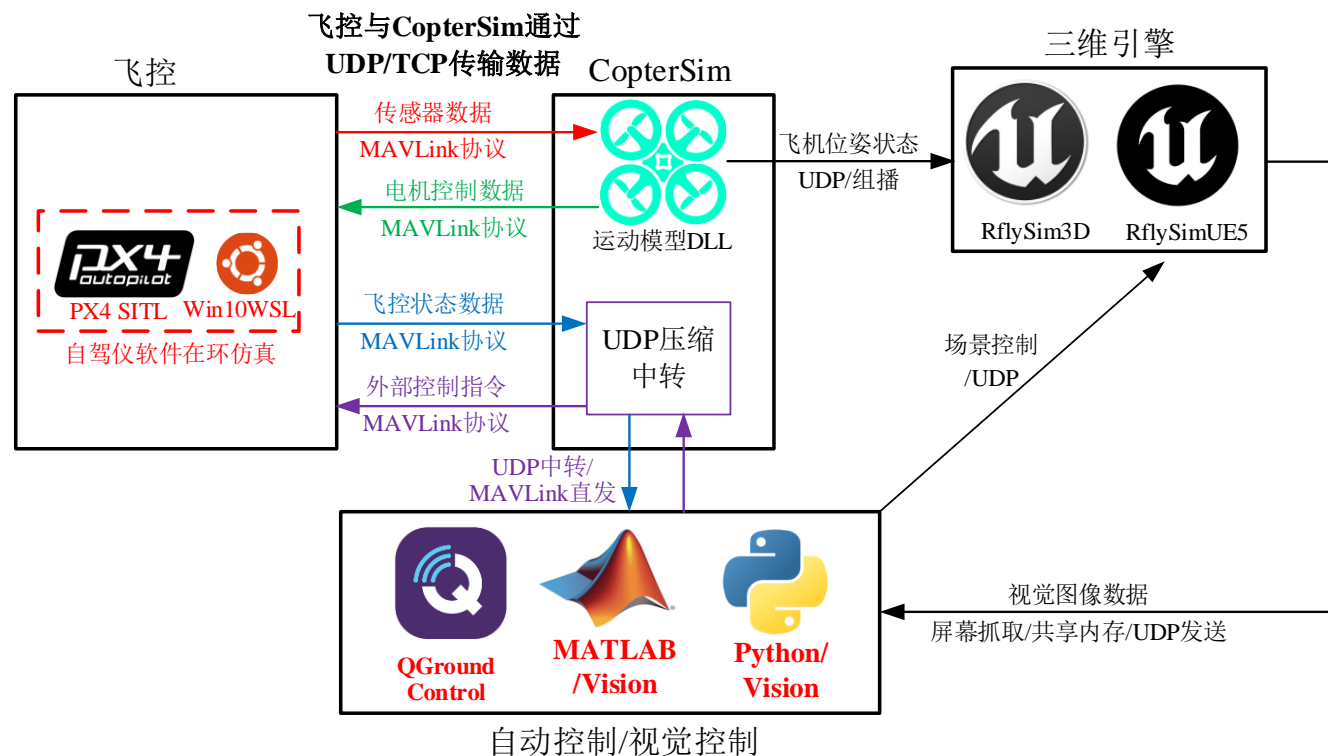




2. The base interface uses

2.1 UAV Control Interface-Lab 1: SITLRun Communication Framework

- In the SITL software in-loop simulation process, the PX4 flight control runs completely in the Win10WSL virtual machine, and the px4_sitl firmware is used.
- PX4 communicates with CopterSim directly through the network MAVLink protocol, and then CopterSim sends and receives external Python messages through the 20100/20101 port.
- The Python program communicates directly with the RflySim3D program through UDP, and obtains visual images through shared memory/UDP sending, etc.



SITLRun Software-in-the-Loop Simulation Communication Architecture

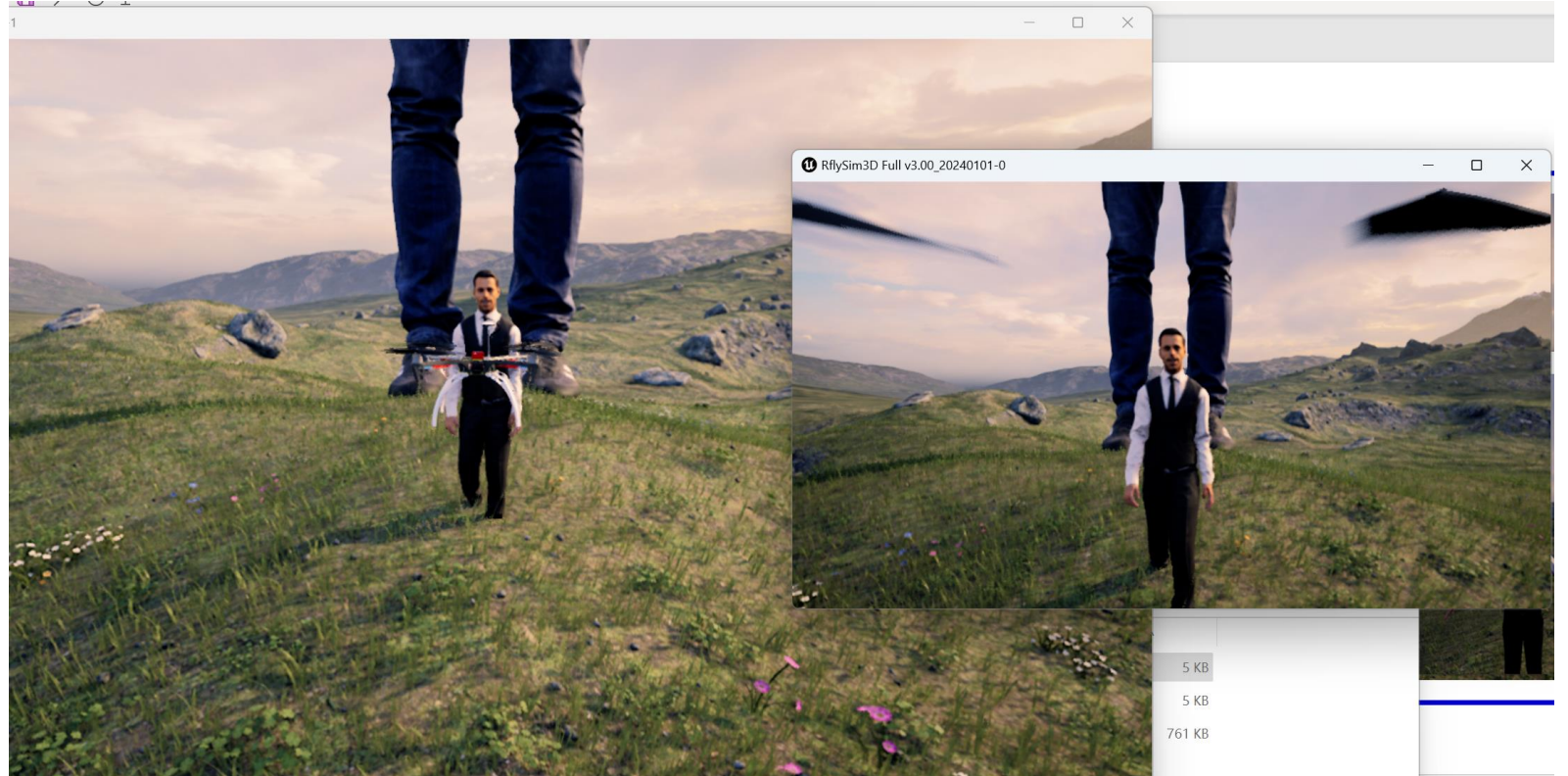


2. The base interface uses

2.1 UAV Control

Interface — Experiment 1: Experimental Results

- The phenomenon of this routine is that the python program sends a series of instructions, creates a new target of a walking person in the RflySim3D program, sets the visual angle form, size and position, and sends control instructions to the simulated UAV to make it take off and land.
- As shown in the figure on the right, this example will open two RflySim3D windows, one for the front-facing camera and the other for God's perspective observation.





2. The base interface uses

2.1 UAV Control Interface-Lab 1: End of Simulation

- In the command prompt CMD window opened by the "PX4MavCtrlAPITest.bat" script shown in the figure below, press the Enter key (any key) to quickly close all programs such as CopterSim, QGC, and RflySim3D.
- As shown in the right figure below, in VS Code, click "Terminate Terminal" to exit the script completely.

```
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. The base interface uses

2.1 UAV Control Interface-Lab 2: Pixhawk Hardware-in-the-Loop Simulation with Data Link

- Connect the computer and Pixhawk flight control with MicroUSB cable. Then double-click "PX4ComAPITest.bat" in "[PX4PSP\RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\e2_PX4ComAPITest](#)" and input the flight control string number to start the hardware-in-the-loop simulation of an aircraft.
- Connect TELEM1 of Pixhawk to a computer with a digital or TTL serial port cable, and record the serial port number at this time, such as COM14
- Open "PX4ComAPITest.py" with VS Code, and modify the COM14 of the following code to your own data string slogan:
- `mav = PX4MavCtrl.PX4MavCtrler(1,'127.0.0.1','COM9',57600)`
- Running the PX4ComAPITest.py program in VS Code, you can observe that the unlock message is received in CopterSim, and Python can output the global positioning data of uavPosGPS.
- `print(mav.uavPosGPS)`

Note: For Linux systems, the format of the string number is /dev/ttyUSB0 or /dev/ttyAMA0
The colon is the baud rate, and the default baud rate of PX4's TELEM1 is the 57600.

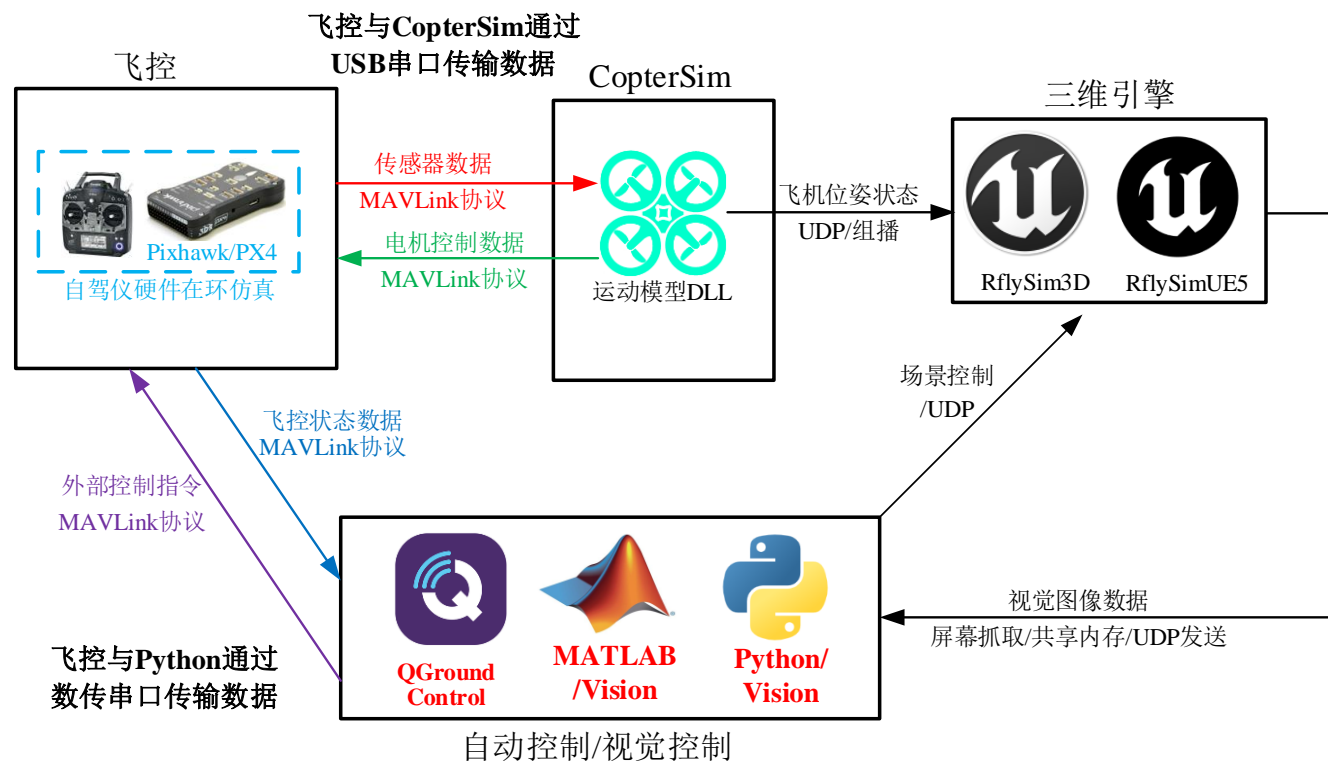




2. The base interface uses

2.1 UAV Control Interface-Lab 2: HITL + Data Communication Framework

- In the HITL hardware-in-the-loop simulation, the PX4 flight control algorithm runs in Pixhawk hardware using the px4 _ fmu-v5 firmware.
- PX4 communicates with CopterSim directly through the MAVLink protocol of USB serial port, and then CopterSim sends and receives messages with external Python programs through the data transmission serial port.
- The Python program communicates directly with the RflySim3D program through UDP, and obtains visual images through shared memory/UDP sending, etc.



HITLRun hardware-in-the-loop simulation communication architecture

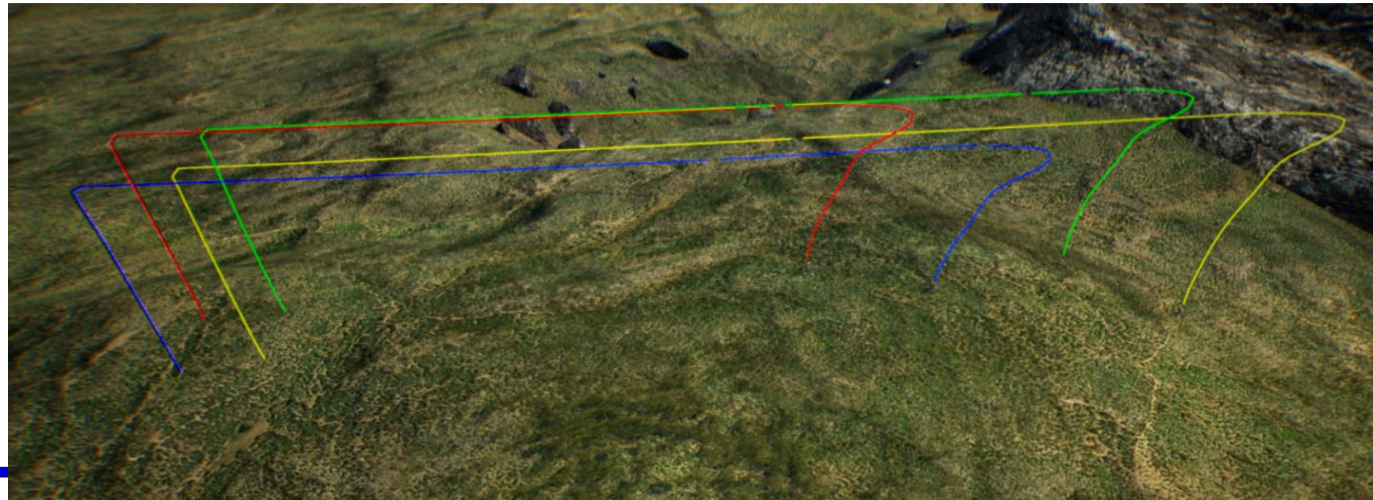


2. The base interface uses

2.1 UAV Control Interface-Lab 3: Multi-Machine SITL Control Lab

- In the "[PX4PSP\RflySimAPIs\6.RflySimExtCtrl\0.ApiExps\e5 PX4MultiUavTest](#)" folder, double-click the "PX4MultiUavTest. Bat" to open the SITL simulation closed loop of the four aircraft.
- Open "PX4MultiUavTest.py" with VS Code, and you can see that four new PX4MavCtrler instances are created in the code, and the connection ports 20100/20102/20104/20106 correspond to aircraft 1 to 4 respectively.
- Then, the key simulation command of UE4 is called to simulate the S key (display the aircraft label) and the T key (display the aircraft trajectory).
- Finally, control the aircraft to unlock, take off, fly forward, and then descend.
- The experimental effect is shown in the right figure.

RflySim3D-0





2. The base interface uses

Note: To make the generated object fit the ground, you can use the UEMapServe class to calculate the terrain height, or use the sendUE4PosScale2Ground interface to create an object that automatically fits the ground. See the source code of the routine for details.

2.2 UE4 Scene Control Interface-Lab 1: Scene Configuration Interface (Importing Obstacles, etc.)

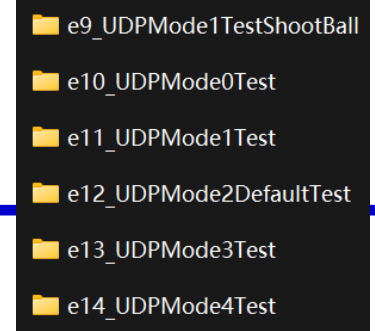
- "[PX4PSP\RflySimAPIs\3.RflySim3DUE\0.ApiExps\e6_RflySim3DCtrlAPI\12.DamageModel](#)" folder, Double-click UE4CtrlAPITest.bat to open two RflySim3D windows.
- The next step is to control the two windows through the Python interface, import obstacles (targets), and configure the window display.
- Open "UE4CtrlAPITest.py" with VS Code, set a breakpoint in the key statement, and then enter the debugging operation mode to execute the statement sentence by sentence and check the execution effect.

```
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 # Augment: RflyChangeMapbyName command mea
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
```



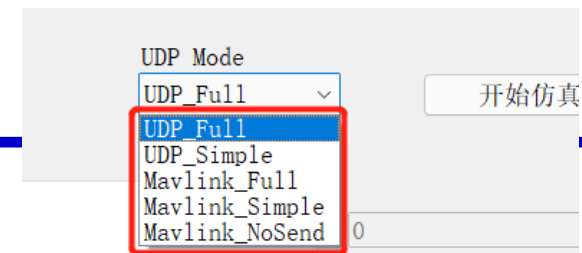


2. The base interface uses



2.3 Python/CopterSim Data Mode (UDP _ Mode)

- In the "[6.RflySimExtCtrl\0.ApiExps](#)" folder, you can see six routine folders, which can be run by yourself to see the effect. The main difference is the statement `InitMavLoop`
- **Mav. InitMavLoop (0)** # corresponds to the **UDP _ Full** mode. Python transmits complete UDP data to CopterSim. The amount of data transmitted is small. After receiving the data, CopterSim converts it into Mavlink and transmits it to PX4 flight control; It is suitable for the simulation of small and medium-sized clusters (the number is less than 10).
- **Mav. InitMavLoop (1)** # corresponds to the **UDP _ Simple** mode, and the packet size and transmission frequency are smaller than those of the **UDP _ Full** mode; it is suitable for large-scale cluster simulation, and the number of UAVs is less than 100.
- **Mav. InitMavLoop (2)** # corresponds to **Mavlink _ Full** mode (default mode). Python directly sends MAVLink message to CopterSim, and then forwards it to PX4. It has a large amount of data and is suitable for single machine control. It is suitable for single machine or a small number of aircraft simulation. The number of UAVs is less than 4;
- **Mav. InitMavLoop (3)** # corresponds to the **Mavlink _ Simple** mode. It will shield part of the MAVLink message packets and reduce the data frequency. The amount of data sent is much smaller than the **MAVLink _ Full**. It is suitable for multi-aircraft cluster control. It is suitable for small-scale cluster simulation. The number of UAVs is less than 8.
- **Mav. InitMavLoop (4)** # corresponds to the **Mavlink _ NoSend** mode. CopterSim will not send MAVLink data to the outside. This mode needs to cooperate with hardware-in-the-loop simulation + data transmission serial communication. MAVLink is transmitted through wired mode. The data volume in the LAN of this mode is the smallest. It is suitable for distributed vision hardware-in-the-loop simulation, and the number of UAVs is not limited.





2. The base interface uses

Note: The free version only supports 2 RGB images and receives images in shared memory

2.4 RflySim3D Mapping Interface-Vision Sensor Profile Config. JSON.

- "[8.RflySimVision\0.ApiExps\1-UsageAPI\0.VisionSensorAPI\1.CameraImageGet](#)" can open the Config. JSON file, which contains two visual sensor structures, defined as follows
- SeqID; //Sensor serial number ID, starting from 0 (free version only supports 2 images)
- TypeID; //Sensor type ID, 1: RGB map, 2: Depth map, 3: Grayscale map,
- 4: Segmentation map, 5: Ranging, 20-22: Lidar, 40: Infrared grayscale, 41: Thermal map
- TargetCopter; //The ID of the target aircraft loaded by the camera//can be changed
- TargetMountType; //Coordinate type, 0: on fixed aircraft (relative to geometric center), 1: on fixed aircraft (relative to bottom center), 2: on fixed ground (monitoring), 3: on the fixed aircraft, but the camera attitude does not change with the aircraft (ground coordinate system),
- 4: Attach a sensor to another sensor, when MountType = 4, TargetCopter = SeqID in the Config. JSON (because MountType = 4 is to attach a sensor to a sensor, So TargetCopter is used to give the vehicle ID, but it's not used at this time. It's used to set the ID of the sensor, that is, SeqID)//variable

Note: TargetMountType determines whether the value of SensorPosXYZ is relative to the center of the aircraft, the center of the bottom of the aircraft, or the ground. In addition, in order to ensure that the object can be attached to the ground, the coordinates sent by the sendUE4 ** command are the center coordinates of the bottom of the object, not the center coordinates, which are separated by the height of the object (see XML definition).

```
{
  "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": [0.3, 0, 0],
      "SensorAngEular": [0, 0, 0],
      "otherParams": [0, 0, 0, 0, 0, 0, 0, 0]
    }
  ],
}
```



2. The base interface uses

Note: The free version only supports 2 RGB images and receives images in shared memory

2.4 RflySim3D Mapping Interface-Vision Sensor Profile Config. JSON.

- **DataWidth://data or image width (not available for distance sensor)**
- **DataHeight://data or image height (not available for distance sensor)**
- **DataCheckFreq://Check the data update frequency (the ranging sensor does not have this parameter)**
- **SendProtocol [8]://transmission mode and address, SendProtocol [0] value 0: shared memory (the free version only supports shared memory), 1: UDP direct PNG compression, 2: UDP direct image compression, 3: UDP JPG compression; SendProtocol [1-4]: IP address; SendProtocol [5] port number**
- **EulerOrQuat: 0 means using Euler angles, that is, SensorAngEular, and 1 means using quaternion SensorAng Quat**
- **CameraFOV://camera field of view (vision sensors only) in degrees//changeable**
- **SensorPosXYZ [3]://Sensor installation position in meters//changeable**
- **SensorAngEular [3]://sensor installation angle, unit ° ° /changeable**
- **SensorAngQuat [4]://sensor mounting angle, expressed in quaternion.**
- **There are also the following parameters for the ranging sensor:**

Distance: The maximum distance that can be detected

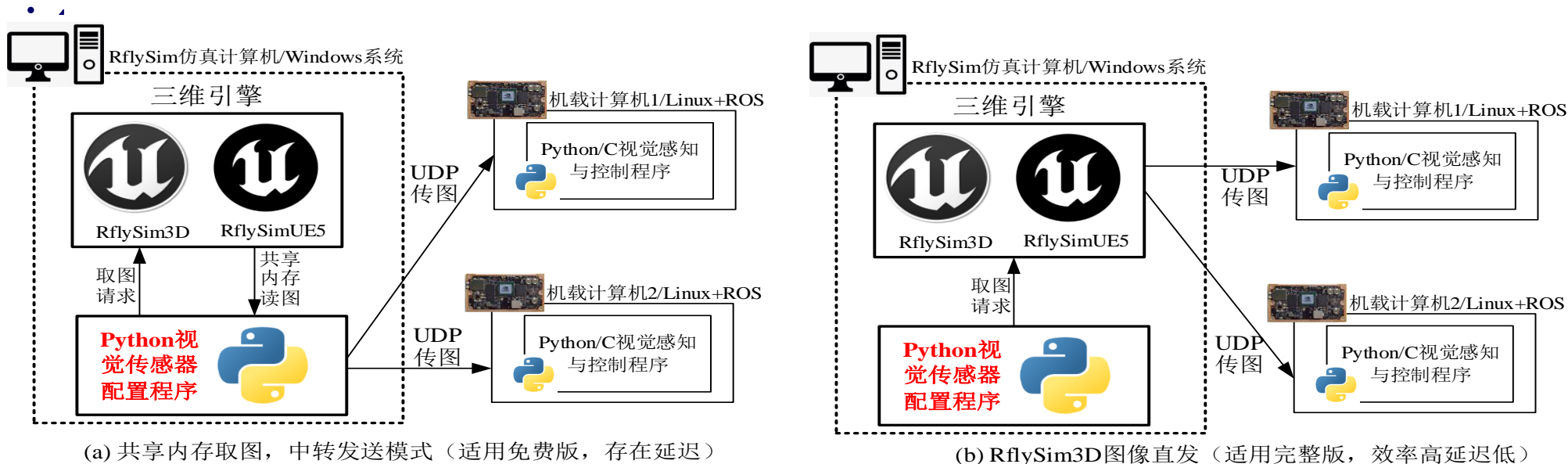
Note: TargetMountType determines whether the value of SensorPosXYZ is relative to the center of the aircraft, the center of the bottom of the aircraft, or the ground. In addition, in order to ensure that the object can be attached to the ground, the coordinates sent by the sendUE4 * * command are the center coordinates of the bottom of the object, not the center coordinates, which are separated by the height of the object (see XML definition).



2. The base interface uses

2.4 RflySim3D drawing interface — drawing and distribution principle

- The RflySim platform must run on a Windows computer, but its image stream can be transmitted to the local vision program, other computers (Windows or Linux), and other embedded computers (Linux + ROS) through shared memory and network communication for the development and simulation of vision algorithms.
- Send the configuration information in the Config. JSON file to RflySim3D to request





2. The base interface uses

2.4 RflySim3D Image Acquisition Interface — VisionCaptureApi. Py of Image Acquisition Interface

- **VisionCaptureApi. Py** is the interface file of the platform, including JSON loading, image request, image forwarding, etc.
- Class `VisionSensorReq`: # data structure, sent to RflySim3D to fetch image data package
- Class `imuDataCopter`: # data structure, IMU data packet returned by CopterSim
- Class `SensorReqCopterSim`: # data structure packet, send to CopterSim request sensor packet
- Class `VisionCaptureApi`: # The main interface class, which implements the drawing request and receiving
- `AddVisSensor (VSR = VisionSensorReq ())`: # Class function to add a vision sensor
- `SendReqToCopterSim (srcs = SensorReqCopterSim (), copterID = 1)`: # class function, which sends a data packet to CopterSim, and can specify the copterSim sequence number of the response request
- `SendImuReqCopterSim (copterID = 1, IP = '127.0.0.1', port = 31000, freq = 200)`: # function, send a data packet to CopterSim to request to send IMU data (IP and port frequency), and start to monitor data
- `SendUpdateUEImage (vs = VisionSensorReq (), windID = 0)`: # Send a request to RflySim3D to update the parameters and position of the specified vision sensor. You can specify the windID of the received RflySim3D window number
- `SendReqToUE4 (windID = 0)`: # Send the stored visual sensor list to RflySim3D, create the sensor, and check whether the creation is successful. You can specify the windID of the received RflySim3D window number
- `StartImgCap (isRemoteSend = False)`: # Start receiving pictures and store them in the `Img` list. `IsRemoteSend` can configure whether the shared memory pictures are forwarded to other systems through UDP.
- `JsonLoad (ChangeMode = -1, jsonPath =)`: # Load the local Json file and store it in the visual sensor list. `ChangeMode` can override the `SendProtocol [0]` transmission mode in Json. `JsonPath` specifies the Json file address.



2. The base interface uses

2.4 RflySim3D Mapping Interface-Mapping Interface Routine File

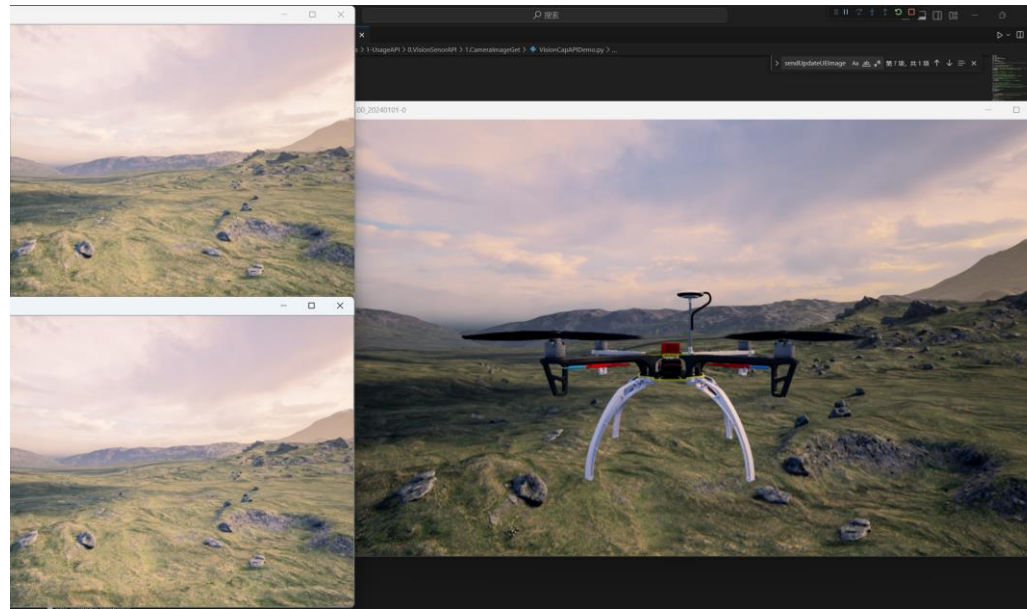
- Open the "VisionCapAPIDemo. Py" file with VS Code, and you can see the implementation principle of the routine
- `Vis = VisionCaptureApi. VisionCaptureApi ()` # Create a graph interface class instance
- `Ue.sendUE4 Cmd ('R. Setres 1280x720w', 0)` # Set the UE4 window resolution. Note that this window is only used for display. The image resolution is configured in JSON. The smaller this window is, the less resources are required.
- `Ue.sendUE4Cmd ('t. MaxFPS 30 ', 0)` # Set the maximum refresh frequency of UE4, which is also the drawing frequency
- `Vis. JsonLoad ()` # Load the sensor configuration file in the Config. JSON.
- `IsSuss = vis.sendReqToUE4 ()` # Send an image fetching request to RflySim3D and verify it
- `Vis. StartImgCap ()` # to open the map, and enable the shared memory image forwarding, forwarding to the filled directory
- `If vis. HasData [I]:` # Determine whether the picture has been received
- `CV2.imshow ('Img' + str (I), vis. Img [I])` # OpenCV displays the current image
- # Image processing algorithms can be added here
- `Vis. SendUpdateUEImage (vs)` # Send request to update vision sensor parameters



2. The base interface uses

2.4 RflySim3D Image Acquisition Interface-Experimental Verification

- Open "[8.RflySimVision\0.ApiExps\1-UsageAPI\0.VisionSenorAPI\1.CameraImageGet\VisionCapAPIDemo.bat](#)" runs as an administrator and opens."
- In this experiment, Json defines two left and right front RGB cameras and displays them in real time.

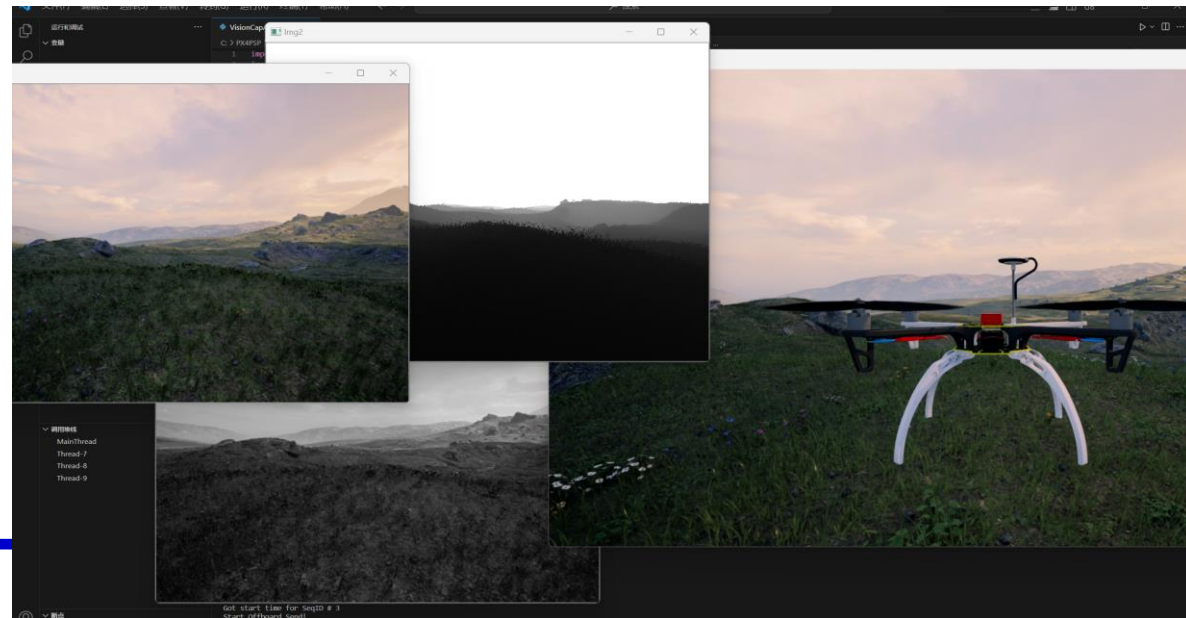




2. The base interface uses

2.4 RflySim3D Image Acquisition Interface-UE5 Multi-camera Experimental Verification

- Edit "[8.RflySimVision\0.ApiExps\1-UsageAPI\0.VisionSenorAPI\2.MutCameraImageGet\VisionCapAPIDemo.bat](#)", You can see that "% PSP _ PATH% \ RflySimUE5" indicates the use of UE5 engine; open the "Config. JSON" "to see three cameras including RGB, grayscale and depth;
- Double-click the "VisionCapAPIDemo. Bat" and Run with VS Code "VisionCapAPIDemo. Py" to see the following effect





2. The base interface uses

2.5 Three modes of PX4 UAV position + speed control-principle explanation

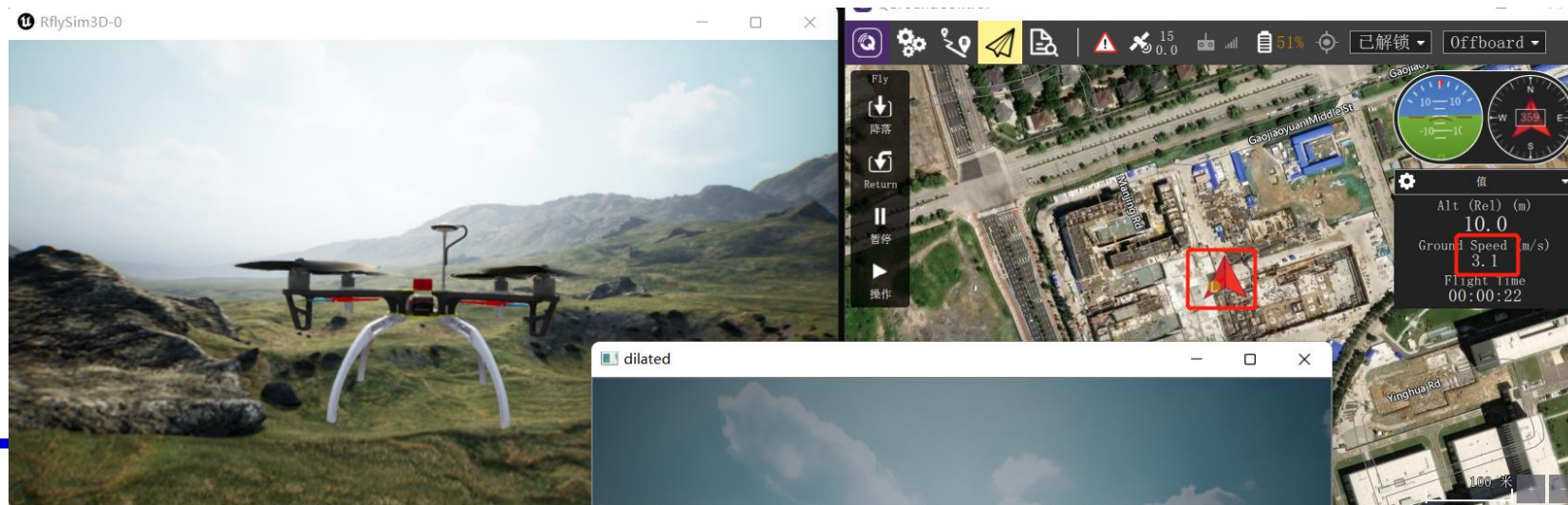
- In visual control, we often need to control the forward speed of the aircraft while controlling the aircraft to fly to the designated target position, so as to achieve good tracking effect.
- Enter the "[8.RflySimVision\0.ApiExps\1-UsageAPI\2.ThreeCtrlModes](#)" directory to see the control examples of the three methods.
- Method 1, see "ThreeCtrlModes _ PosCtrl. Py" for the routine, and use the Offboard location interface
 - `Mav. SendPosNED (12, 13, -10, 0) # send desired position`
 - The `Mav. SendCopterSpeed (3) # calls the function to set the maximum speed of the aircraft.`
- The second method is to use `SendVelNED` speed control interface to realize position control on the basis of `SendVelNED` speed control interface. See "ThreeCtrlModes _ VelCtrlEarth. Py" for the routine.
- Method 3, see "ThreeCtrlModes _ VelCtrlBody. Py" for routine, use `SendVelFRD` speed control interface to realize position control on this basis, and the nose of this mode always faces to the target direction.



2. The base interface uses

2.5 PX4 UAV Position + Speed Control Three Modes-Experimental Results

- Run the "ThreeCtrlModesSITL. Bat" or "ThreeCtrlModesHITL. Bat" to start a software-in-the-loop or hardware-in-the-loop simulation of the aircraft.
- Run the "ThreeCtrlModes _ PosCtrl. Py" ". It can be seen that the aircraft flies to the specified position [120, 130, -10]. On the QGC, it can be seen that the speed is the set value of 3m/s. At the same time, the nose always faces north and the speed direction is inconsistent.
- Run "*" _ PosCtrlFRD. Py" ", "*" _ VelCtrlBody. Py "" and "*" _ VelCtrlEarth. Py" in turn. Please read the code by yourself and confirm the experimental effect. Note: All three modes can control the speed of the aircraft and always point to the target.





2. The base interface uses

Note: For the method of obtaining the initial height of the essential point model and the detailed method of multi-machine simulation, please refer to the relevant contents in the next chapter.

2.6 Control Interface in Lightweight UAV Model-Experimental Principles

- In the above example, running the bat script will start the software-in-the-loop or hardware-in-the-loop simulation of the aircraft, requiring the participation of CopterSim + flight control + QGC, which occupies more resources and may be limited by performance in multi-aircraft visual simulation.
- In the directory "[8.RflySimVision\0.ApiExps\1-UsageAPI\1.UAVCtrlNoPX4Demo\1.UAVCtrlNoPX4Demo](#)", we developed a particle based UAV control model in Python.

It can provide UAV dynamic effects similar to hardware and software in-loop simulation, but greatly reduce the occupancy of computer performance and enhance flight.

Machine stability.

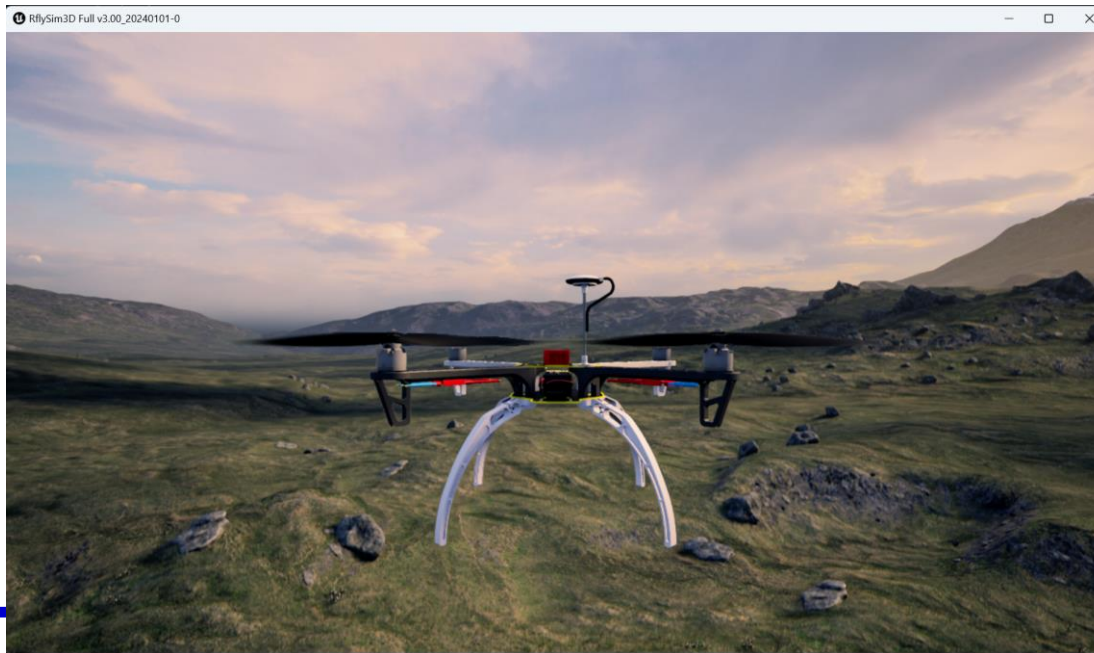
- Open "UAVCtrlNoPX4Demo.py" with VS Code. It can be seen that this mode is exactly the same as the MAVLink-based control interface of SITL or HITL. The difference lies in the following statement.
- `Mav.InitPointMassModel (-8.086, [0,0,0])` # Replace original initOffboard statement
- After the above statement is executed, a new particle UAV model will be automatically created (set the initial ground height, XY position and yaw angle), and the position and speed commands will be monitored. (Exactly the same as the original control method)
- Note: By opening "UAVCtrlNoPX4Demo.bat" with VS Code, we can see that only one RflySim3D program is opened, and there is no need to open other programs such as CopterSim.



2. The base interface uses

2.6 Control interface in a lightweight UAV model – experimental results

- Double-click to run "UAVCtrlNoPX4Demo.bat", and you can see that an RflySim3D window is opened, and no other program is opened.
- Open "UAVCtrlNoPX4Demo.py" with VS Code and run it. You can see that the aircraft takes off and flies according to the position or speed command sent. The flight control effect is similar to that of the software/hardware in-loop, but it is smoother.



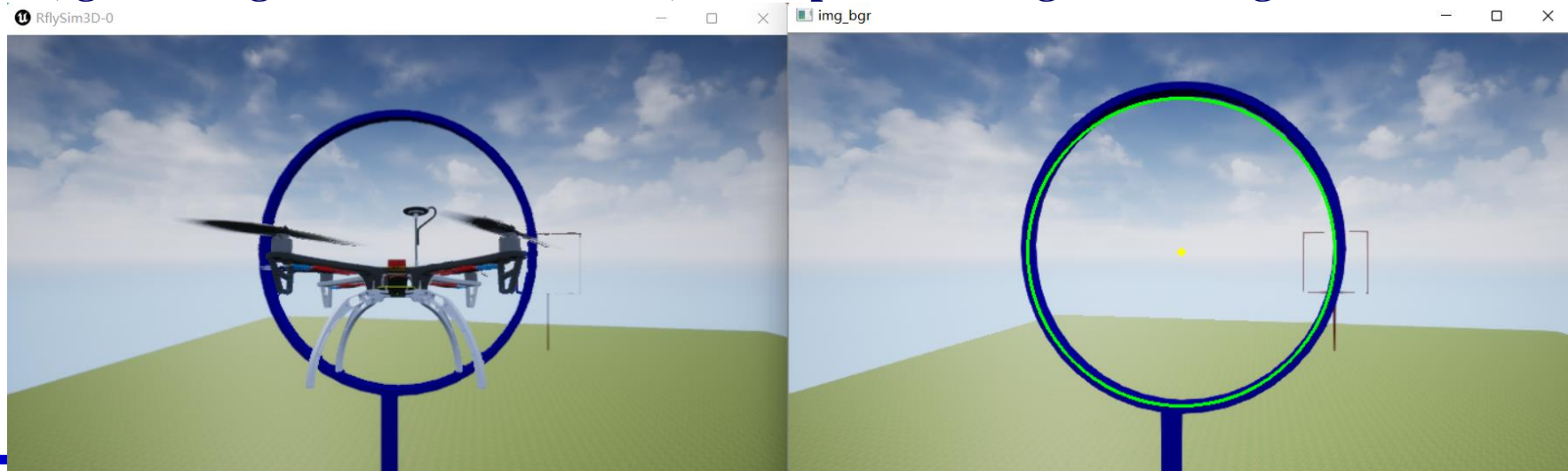
```
C: > PX4PSP > RflySimAPIs > 8.RflySimVision > 0.ApiExps > 1-UsageAPI > 1.UAVCtrl
1 import time
2 import math
3 import sys
4 import cv2
5 import UE4CtrlAPI
6 ue = UE4CtrlAPI.UE4CtrlAPI()
7
8 import PX4MavCtrlV4 as PX4MavCtrl
9
10
11 #Create a new MAVLink communication instance, UDP sendin
12 mav = PX4MavCtrl.PX4MavCtrl(1)
13
14
15 # sendUE4Cmd: RflySim3D API to modify scene display styl
16 # Format: ue.sendUE4Cmd(cmd>windowID=-1), where cmd is a
17 # Augment: RflyChangeMapbyName command means to switch
18 ue.sendUE4Cmd('RflyChangeMapbyName Grasslands')
19 time.sleep(2)
20
21 # Create a vehicle and setting the ground height and ini
22 mav.initPointMassModel([-8.086, [0,0,0]])
23
24 time.sleep(2)
25 mav.SendVelNED(0,0,-2,0) # takeoff with speed 2m/s
26
```



2. The base interface uses

2.6 Control Interface under Lightweight UAV Model-Visual Loop Penetration Experiment

- Open the "[8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\01_CrossRingNoPX4](#)" directory, and you can see the loop threading experiment routine based on the particle model. (For the specific visual control principle, please refer to the ring-threading experiment in the next section)
- Double-click CrossRingNoPX4.bat to open a RflySim3D window
- Open "CrossRingNoPX4.py" with VS Code and run it, you can see that the scene switches to the grass ring scene, generating a multi-rotor aircraft, which passes through three rings in turn after taking off.





2. The base interface uses

Note: The LIDAR sensor is currently limited to the premium full version, and the free experience version does not have this feature.

2.7 Lidar-Source Code Description

At present, there are two ways for data interaction: UDP direct transmission and memory sharing. The following is a description of the source code directory

[8. RflySimVision \ 0.Api Exps \ 4-AdvApiExps \ 7.LidarAPIDemo](#)

- **1. SharedMemory 10Hz:** shared memory mode, data transmission frequency 10hz;
- **2. SharedMemory ClientServer:** In the shared memory mode, after the Client receives the shared memory, UDP is sent and the Server receives the point cloud.
- **3. UDPDirect30Hz:** UDP directly transmits 30hz frequency, Python sends the drawing request to RflySim3D, and the latter directly transmits the point cloud through UDP;
- **4. UDPDirect ClientServer:** UDP direct transmission between the client and the server. The Client sends the request, and the Server receives the point cloud. **Note: The Server supports virtual machines or other computers.**
- **5. UDPDirect ClientServerType5:** UDP direct client and server transmission mode, using the map coordinate system (default to use the laser radar coordinate system);

1.SharedMemory10Hz

2.SharedMemoryClientServer

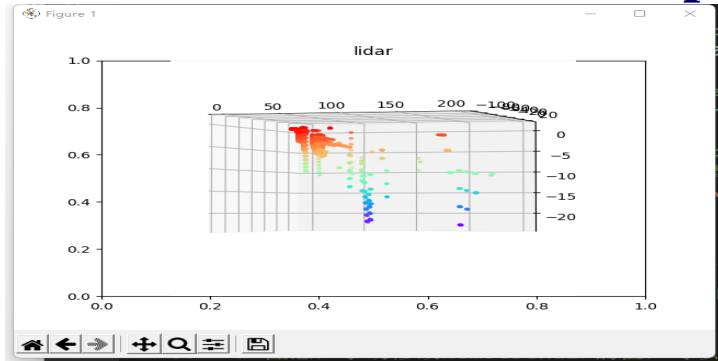
3.UDPDirect30Hz

4.UDPDirectClientServer



2. The base interface uses

2.7 Lidar-Source Code Description 2



Note: The value of SendProtocol [0] determines whether the shared memory is used for UDP direct transmission. When the jsonLoad function is called in Python, the attached parameter can be used to forcibly set the value of SendProtocol [0] and change the transmission mode.

```
...
"DataCheckFreq":10,
"SendProtocol":[1,127,0,0]
"CameraFOV":0.0

# VisionCaptureApi
vis.jsonLoad(1) #
```

- Although there are five projects in the project directory, there are only two transmission modes. Now, the code operation of these two modes is described as follows. Take 1.SharedMemory10Hz and 4.UDPDirect ClientServer as examples:
- 1. SharedMemory 10Hz: After running the LidarAPIDemo. Bat script, run the LidarAPIDemo. Py script directly to see the visual point cloud.
- 4. UDP direct client serv: IP configuration is required for remote communication,

First check the IP of your virtual machine Ubuntu:

Press Ctrl + Alt + T to open the terminal and enter the command "ifconfig" (without quotation marks) as shown in the figure:



2. The base interface uses

2.7 Lidar-Source Code Description 3

- If you want to transmit the point cloud to the remote Linux \ ROS system, you need to modify the JSON configuration file or client _ ue4.py to set the IP address.
- If the IP of my virtual machine or Linux computer is 192.168.31.88, the JSON configuration file is shown in the following figure:

```

{
  "VisionSensors": [
    {
      "SeqID": 0,
      "TypeID": 20,
      "TargetCopter": 1,
      "TargetMountType": 0,
      "DataWidth": 900,
      "DataHeight": 32,
      "DataCheckFreq": 10,
      "SendProtocol": [1, 192, 168, 31, 88, 9999, 0, 0],
      "CameraFOV": 90,
      "SensorPosXYZ": [0, 0, -0.3],
      "SensorAngEular": [0, 0, 0],
      "otherParams": [200, 0.05, -45, 45, -20, 20, 0, 0]
    }
  ]
}

```

替换成虚拟机IP地址

1表示UDP直传

```

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::1179:12a:1a5a:f74f prefixlen 64 scopeid 0x20<link>
    inet6 2001:250:4400:400f:c4b2:eae0:5b66:696f prefixlen 64 scopeid 0x0<
global>
global>
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 3993400 bytes 142917850805 (142.9 GB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 3993400 bytes 142917850805 (142.9 GB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Note: It is also possible to leave the IP address in SendProtocol in JSON unchanged and use 127.0.0.1, but uncomment the vis. RemotSendIP in client _ ue4.py and set it to the IP address

```

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

```

Note: This method does not modify the destination IP addresses of all sensors defined in JSON.



2. The base interface uses

2.7 Lidar-Source Code Description 3

- In addition, the parameters in the `_SITL.Bat` of the script client `_ue4` need to be modified so that other computers can receive the aircraft data from CopterSim and control the aircraft in real time.
- `IS_BROADCAST = 1` corresponds to broadcast mode, all computers can receive data, low efficiency, but convenient
- `IS_BROADCAST = 192.168.31.88` (configure according to your own virtual machine), high efficiency.

```
40 SET /G VEHICLE_INTERVAL 2
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. The base interface uses

2.7 Lidar-Source Code Test Summary

- **1. SharedMemory10Hz:** Run the LidarAPIDemo. Bat script. After RflySim3D prompts Fixed, open and run the LidarAPIDemo. Py script with VS Code to view the point cloud. Control the aircraft movement through QGC and observe the change of the point cloud.
- **2. SharedMemory ClientServer:** Run the client _ ue4 _ SITL. Bat script, double click Python 38Run.bat after it is Fixed, enter the python client _ ue4.py, read the point cloud from the shared memory and send it out with UDP. Open server _ ue4.py with VS Code and run it.
- **Note 1:** The reason why the client transmits the point cloud through UDP is that the function of the "vis. StartImgCap (True)" statement (True parameter data is added compared with Routine 1) will trigger the point cloud forwarding function.
- **Note 2:** Change the IP address of SendProtocol in the JSON file to the remote address to transmit the point cloud to the Linux computer. A simpler way is to directly uncomment the following statements in the client without changing SendProtocol and fill in the target IP address to automatically forward the point cloud to this IP address.

```
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jso
```



2. The base interface uses

2.7 Lidar-Source Code Test Summary

- **3. UDPDirect30Hz:** The test method is the same as that of 1. SharedMemory10Hz, run bat first (administrator mode can be used), then run python script with VS Code, control the aircraft through QGC, and observe the change of point cloud. Features of this routine: RflySim3D runs at 90 frames, the point cloud acquisition speed is 30 frames, and UDP is directly transmitted to the local IP.
- **4. UDPDirect ClientServer:** Run the client _ ue4 _ SITL. Bat (administrator), then run Python 38Run.bat, and run the Python client _ ue4.py from the command line. VS Code runs server _ ue4.py.
- **Note:** If you want to upload the point cloud to other computers, you need to modify the IS _ BROADCAST value of .bat and the IP address corresponding to the vis. RemotSendIP of client _ ue4.py. If it is a ROS system, you can run server _ ue4ROS.py to publish the point cloud ROS package and preview it.
- **5. UDPDirect ClientServerType5:** The test method is the same as that of the previous routine, except that the axis of the coordinate system corresponding to Type5 is relative to the earth, and the map point cloud can be obtained by superimposing the radar position in ImgData. In Type 4 mode, the point cloud is relative to the radar attitude and needs to be mapped to the ground coordinate system to obtain the map point cloud.



2. The base interface uses

2.7 Lidar-Data Interface:

- **Vis. VisSensor [I]:** The JSON data structure of the *i*th sensor (corresponding to the SeqID of the JSON file and the serial number of the sensor), which can be used to obtain the parameter configuration information of the current sensor.
- **Vis. HasData [I]:** Whether the *i*th sensor has updated data. If yes, it will become True.
- **Vis. Img [I]:** Image/point cloud data of the *i*th sensor, $N * 3$ dimensional vector array (the value of N is equal to the vis. ImgData [I] [6]). Refer to the routine for the specific data acquisition method. Point cloud data in meters (front-left-top coordinate system).
- **Vis. TimeStmp [I]:** timestamp of the *i*th sensor, in millisecond
- **Vis. ImgData [I]:** 7-dimensional vector, vis. ImgData [I] [0 ~ 2] position XYZ of the *i*-th sensor (unit: m, Front-left-top coordinate system), vis. ImgData [I] [3 ~ 5] pose Euler angle roll, pitch, yaw (unit radian), vis. ImgData [I] [6] total number of point clouds.
- **Note:** In this example, there is only one sensor, so the value of *I* is 0.
- **Note:** The interface for controlling the aircraft and obtaining flight control data is Mav. See Interface Routine for Aircraft Control.



2. The base interface uses

2.7 Laser radar-parameter configuration description

- The configuration parameters are in the Config. JSON file in the 8-Lidar APIDemo directory
- The parameters specific to the LIDAR configuration are described here, and other parameters are described in 2.6 Config. JSON configuration file description:
 - TypeID: value 20-22; 20: the output point cloud is the laser radar coordinate system; 21: the output point cloud is the world coordinate system; 22: stands for livox lidar;
 - DataWidth: the number of point clouds within a ring of the laser radar;
 - DataHeight: is the number of laser radar harnesses. Data CheckFreq: Point cloud publishing frequency (Hz)
 - DataHeight: number of laser radar harnesses;
 - SendProtocol: transmission mode and IP, where SendProtocol [0] represents the shared memory output mode, and SendProtocol [1] represents the UDP direct sending mode.
 - OtherParams: [maximum laser distance (m), precision (m), lower limit of horizontal scanning angle (degree), upper limit of horizontal scanning angle (degree), lower limit of vertical scanning angle (degrees), upper limit of vertical scanning angle [degrees], reservation, reservation];

Note: The horizontal resolution of the laser radar is represented by DataWidth and the horizontal scanning angle range, and the vertical resolution is represented by the processing scanning angle (as shown in the figure, horizontal resolution = 90/900, vertical resolution = 40/32). The above angle values are expressed by degree.

Note: There is a delay of UE frame rate in the interface of this point cloud, so "vis.sendUE4Cmd (B't. Max FPS 30 ', 0)" in Python

Statement can speed up the sending frequency of UE4 and reduce the data sending delay. For example, if the UE frame rate is set to 100 frames, the output delay is only 0.01s.

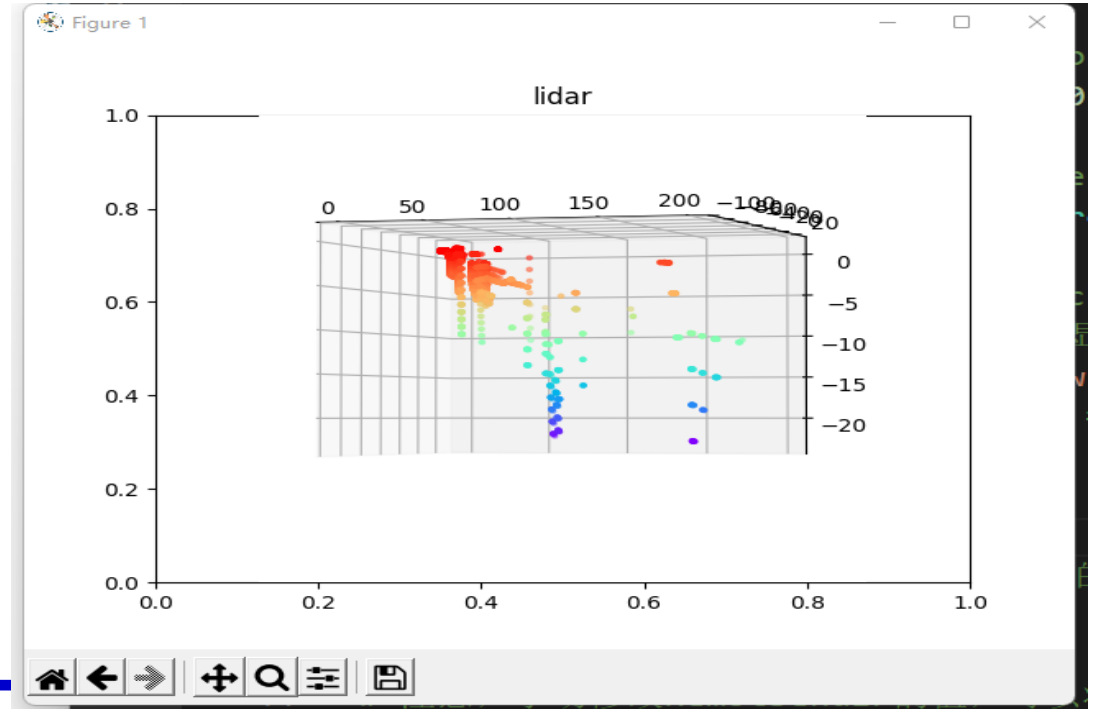
```
{
  "VisionSensors": [
    {
      "SeqID": 0,
      "TypeID": 20,
      "TargetCopter": 1,
      "TargetMountType": 0,
      "DataWidth": 900,
      "DataHeight": 32,
      "DataCheckFreq": 10,
      "SendProtocol": [0, 127, 0, 0, 1, 9999, 0, 0],
      "CameraFOV": 90,
      "SensorPosXYZ": [0,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



2. The base interface uses

See the routine of 1. SharedMemory10Hz, and the SendProtocol [0] in the key point JSON is set to 0.

- 2.7 Lidar-shared memory operation
- Start RFlySim3D first, that is, run the LdiarAPIDemo. Bat file, and then run the LidarAPIDemo. Py script, you can see the following operation
- Note that the current point cloud output is the North-East coordinate system, that is, the Z axis is downward, so the point cloud is inverted.





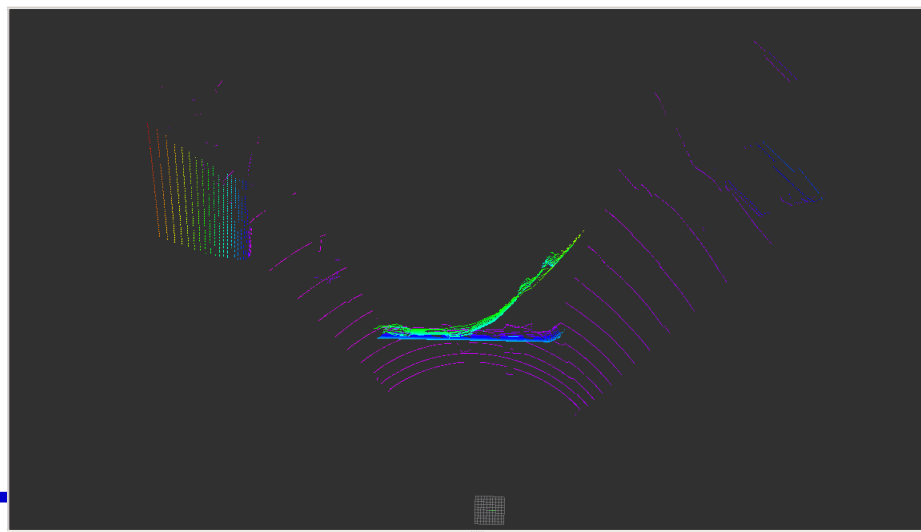
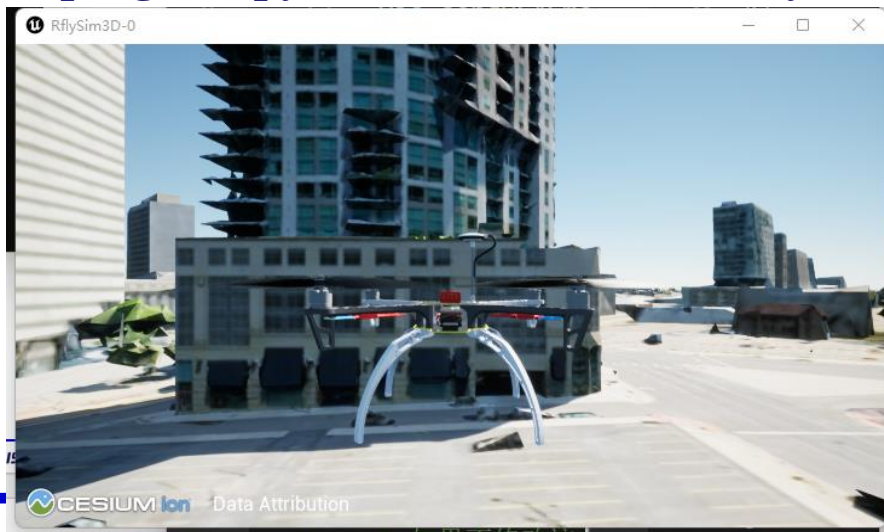
2. The base interface uses

See 4. Routine of UDP
DirectClientServer. Set SendProtocol [0]
in the key point JSON to 1, and fill in the
IP address of the remote Linux computer.

2.7 Laser radar-UDP direct transmission to remote computer mode

- Configure the parameters of remote communication before operation. See 2.6 Parameter Configuration Description for details.
- Run RflySim3D, which is the client _ ue4 _ SITL. Bat script (administrator mode), and then run the data conversion script client _ ue4.py locally with VS Code or Python 38Run.bat command box. Run server _ ue4.py on the remote side (virtual machine or Linux computer) to see the same running results as the shared memory mode, but it is recommended to use ROS to display the point cloud, so do not run server _ ue4.py on the remote side, run roscore first, and then use

Run the program python server _ ueROS. Py and open rviz to see the results below.





2. The base interface uses

2.7 Lidar-UDP direct transmission mode operation (ROS communication description)

- After running server _ ue4ROS.py, two topics will be published.
 - /rflysim/vehicle 0 _ pose: The number 0 inside is the vehicle number;
 - /rflysim/vhicle _ 0/lidar _ 0: The following 0 is the lidar number. Note this radar.

The number is the unified radar number in the whole RFLySim, not the radar number relative to a certain carrier;

View data through rostopic echo topic _ name, such as:

You can see the pose and point cloud data, and the other data.

The value of frame _ ID can be modified according to the needs of your TF tree.

Modify the code yourself. The point cloud coordinates may be relative to the lidar.

Coordinate system can also be relative to the ground, depending on your own configuration.

```
→ ~ rostopic list
/rflysim/vehicle0_pose 发布载体位姿
/rflysim/vehicle_0/lidar_0 发布点云
/rosout
/rosout_agg
→ ~ █
```

```
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,
---
```

```
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
```



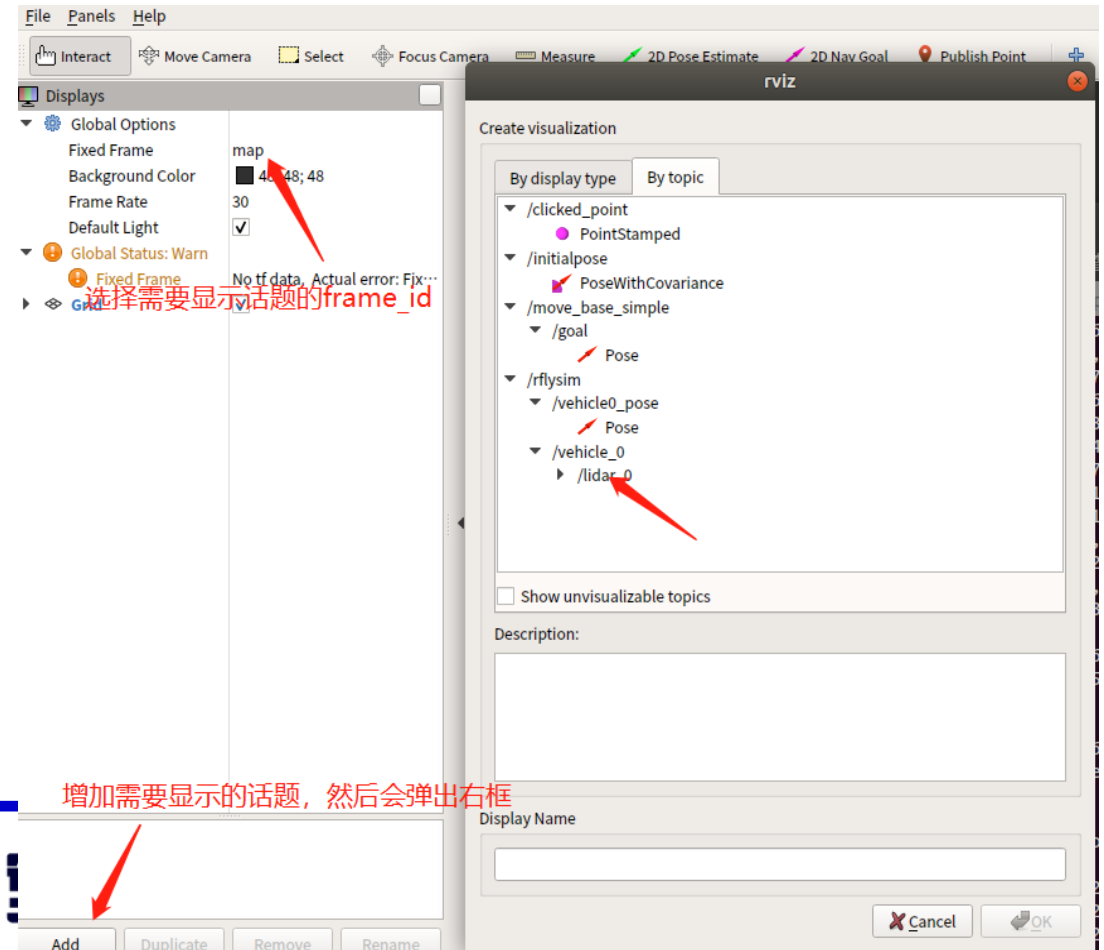
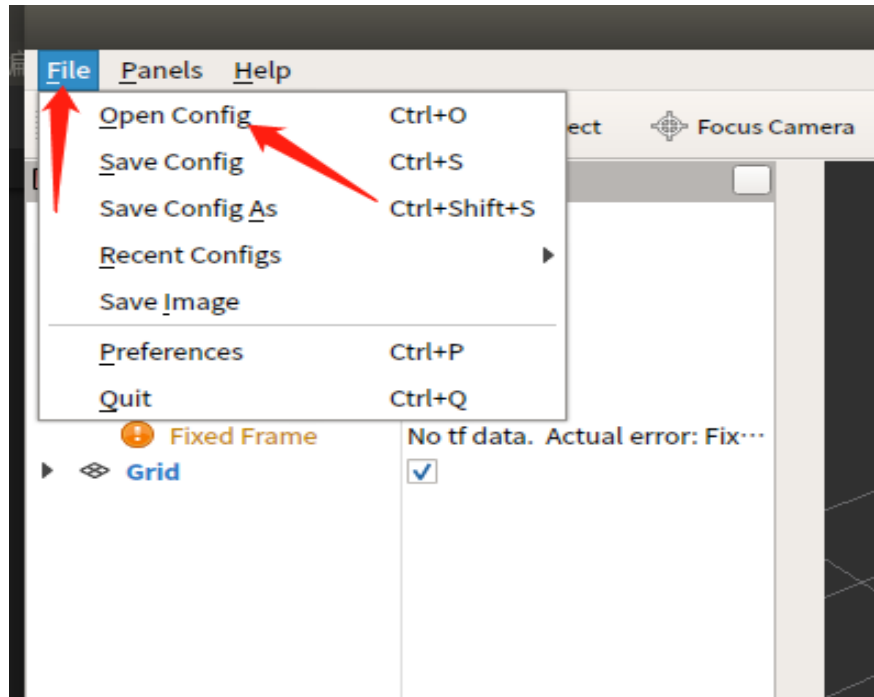

2. The base interface uses

2.7 Lidar-UDP transmission mode operation (ROS, rviz visualization)

The Rviz tool can easily view the environment point cloud data, which needs to be configured.

If it is the default code, you can directly load the configured files under the source file directory.

Okay, lidar. Rviz the files.





2. The base interface uses

2.8 Depth Camera-Json Profile

- The JSON file configuration method for depth cameras is basically the same as that for RGB and grayscale cameras.
- The TypeID of the depth camera needs to be set to 2. See Section 2.3 for other parameters. Includes resolution DataWidth and DataHeight, binding aircraft TargetCopter, binding type TargetMountType, refresh rate DataCheckFrequently, angle of view CameraFOV, Mounting position SensorPosXYZ in meters and mounting attitude SensorPosEular. (Unit degree)
- Other parameters are stored in the OtherParams vector, including minimum range, maximum range, and pixel precision.

```
{  
  "SeqID":1,  
  "TypeID":2, 类型ID为2  
  "TargetCopter":1,  
  "TargetMountType":0,  
  "DataWidth":640, 分辨率和刷新率  
  "DataHeight":480,  
  "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. The base interface uses

2.8 Depth Camera-Unique Parameters

- **Note:** The output data of depth camera is stored and transmitted in uint16, and its data range is 0 ~ 65535. By default, one unit represents 1mm (controlled by otherParams [2]), which means that the maximum range is 0 to 65.535 meters. However, the data range does not represent the actual detection distance of the camera, and otherParams [0] is required to set the minimum detection distance and otherParams [1] is required to set the maximum detection distance.
- **OtherParams [0]:** The minimum recognition distance of the depth camera (unit: m). If the depth distance is less than this value, the 65535 corresponding to NaN will be output.
- **OtherParams [1]:** The maximum recognition distance of the depth camera (unit: m). If the depth distance is greater than this value, the 65535 corresponding to NaN will be output.
- **OtherParams [2]:** The scale unit (in meters) of the output value of the depth camera uint16. By default, the depth value is in milliseconds, so 0.001 is required. **Note:** If the default value is 0, it will be replaced by otherParams [2] = 0.001.
- **Actual depth value (in meters) = depth picture value (uint16 range) * otherParams [2]**



2. The base interface uses

2.8 Depth Camera-Experimental Procedure

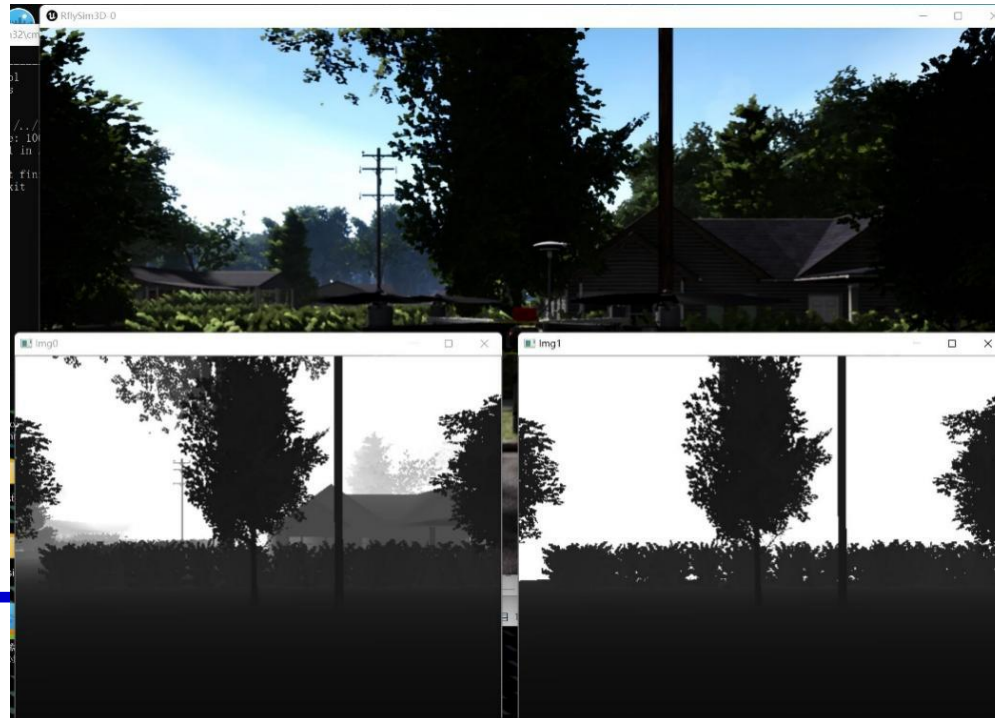
- For the routine of the depth camera, see "[8.RflySimVision\0.ApiExps\1-UsageAPI\0.VisionSensorAPI\5.DepthCameraDemo](#)". First run the DepthCameraDemo. Bat script to start the simulation. After CopterSim displays 3DFixed, turn on control and visual recognition in the run DepthCameraDemo. Py.
- This example contains two depth cameras, both of which are included in the Config. JSON.
- In the first visual structure of the Config. JSON, otherParams is all 0, indicating that the default configuration is used. The unit is mm, the minimum distance is 0 m, and the maximum distance is 65.535 m.
- In the second visual structure of the Config. JSON, otherParams [0] = 0.3, otherParams [1] = 12, and otherParam [0] = 0.001. Indicates that the nearest recognition distance is 0.3 meters, the farthest distance is 12 meters, and the scale unit is millimeter.
- In the Python routine, the depth image matrix can be read directly from the vis. Img [I]. The vis. HasData [I] is used to judge whether the data is updated.
- During the movement, the camera parameters can be dynamically adjusted by the vis. SendUpdateUEImage (vs).



2. The base interface uses

2.8 Depth Camera-Test Results

- The test result is shown in the figure below. The aircraft takes off from the ground, and then opens the visual detection window, which can output two depth pictures. The image on the left uses the default configuration, and remote buildings can be seen without range constraints; the depth image on the right has a maximum detection distance of 12 meters, so distant buildings cannot be seen.





2. The base interface uses

2.9 Timestamp-Structure Definition

- The structure definition code of the timestamp is shown in the right figure.
- When receiving, you need to receive data from the UDP port of the 20005, check whether the packet length is equal to 32 bytes, check whether checksum is the preset 123456789, and judge whether copterID is the ID of your aircraft. The Python routine code is shown in the right figure.
- The specific code can be seen in the [ReadTimeS TMP. Py](#) of "[8.RflySim Vision\0.ApiExps\1-UsageAPI\6.ReadTimeStmp](#)".

```
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.TimeStmploop, args=()) //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 TimeStmploop(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]

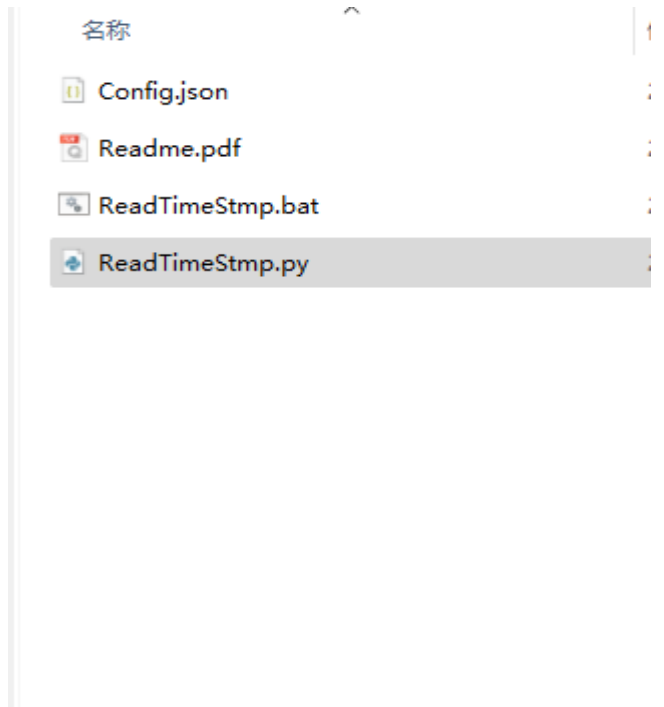
        except:
            print("Error to listen to Time Msg!")
            sys.exit(0)
```



2. The base interface uses

2.9 Timestamp-Lab Flow

- See [8.RflySimVision\0.ApiExps\1-UsageAPI\6.ReadTimeStmp](#) **ReadTimeStmp. Py** for the timestamp routine. **Run the ReadTimeS TMP. Bat script to start the simulation. After CopterSim shows 3DFixed, the running ReadTimeStmp. Py can subscribe to get the timestamp.**
- **Key code:** The **PX4MavCtrlV4** and **VisionCaptureApi** interfaces both implement the timestamp listener function. First, call **StartTimeStmplisten ()** in **jsonLoad** to enable the timestamp listener. The **vis. RflyStartStmp** is the time (system time or ROS time) of the computer where **py** is located when **CopterSim** is started (the aircraft corresponding to **TargetCopter**), and the **vis. TimeStmp** is the time from the start of **CopterSim** to the current data generation. The **vis. IMG Stmp** is that true timestamp of the image





2. The base interface uses

2.9 Timestamp — Experimental Effect

The result of the test is shown in the following figure. Open the ReadTimeStmp. Bat and start the software in-loop simulation of an aircraft.

Python runs ReadTimeS TMP. Py subscriptions to get timestamps.

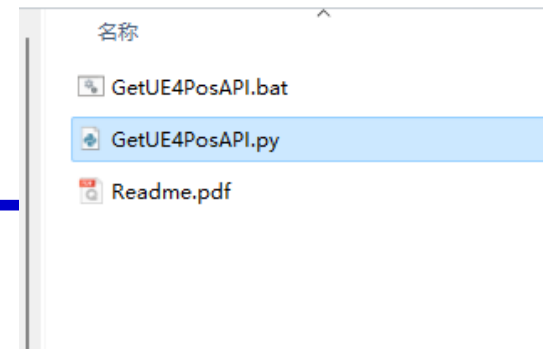
```
62  
  
问题 输出 调试控制台 终端 端口  
  
Got 1 vision sensors from json  
Start lisening to timeStmp Msg  
Got time msg from CopterSim # 1  
CopterSim running on this PC  
Got CopterSim time Data for img  
Sensor req success from UE4.  
Got start time for SeqID #Start lisening to IMU Msg 0  
  
Start Image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




2. The base interface uses

2.10 Obtaining the UE Interface — Interface Introduction

- This interface file is an interface for Python to obtain the positions and collision data of all dynamically created objects in UE.
- `Ue.sendUE4Cmd ('RflyReqVehicleData 1')` This interface is used to request UE4 to return the received data of all aircraft (obstacles). Note: In order to reduce the bandwidth consumption, only when the aircraft data changes, the data will be returned. This means that objects created by obstacles sent before this command will not be sent out. Therefore, this command needs to be sent before the aircraft is created.
- `Ue.initUE4MsgRec ()` is a function for python to monitor all aircraft status data sent by UE4. After calling this function, aircraft data can be received.





2. The base interface uses

2.10 Obtaining the UE Interface — Interface Introduction

- The aircraft data is stored in the list `inReqUpdateVect` (Boolean, update flag or not) and the `inReqVect` list (`reqVeCrashData` structure, store collision data). The length of this list is variable, and each bit of data is a struct `reqVeCrashData`, as shown in the right figure:
- The `ue.getUE4Pos (TargetCopterID)` interface will search whether there is a `copterID` aircraft in the list and output the location. The output format is a 4-dimensional vector, the first three dimensions are the position XYZ vector of the aircraft, and the last dimension is the bool variable of whether there is an aircraft in the list.
- `Ue.getUE4Data (TargetCopterID)` can obtain the data structure `reqVeCrashData` of the current aircraft.

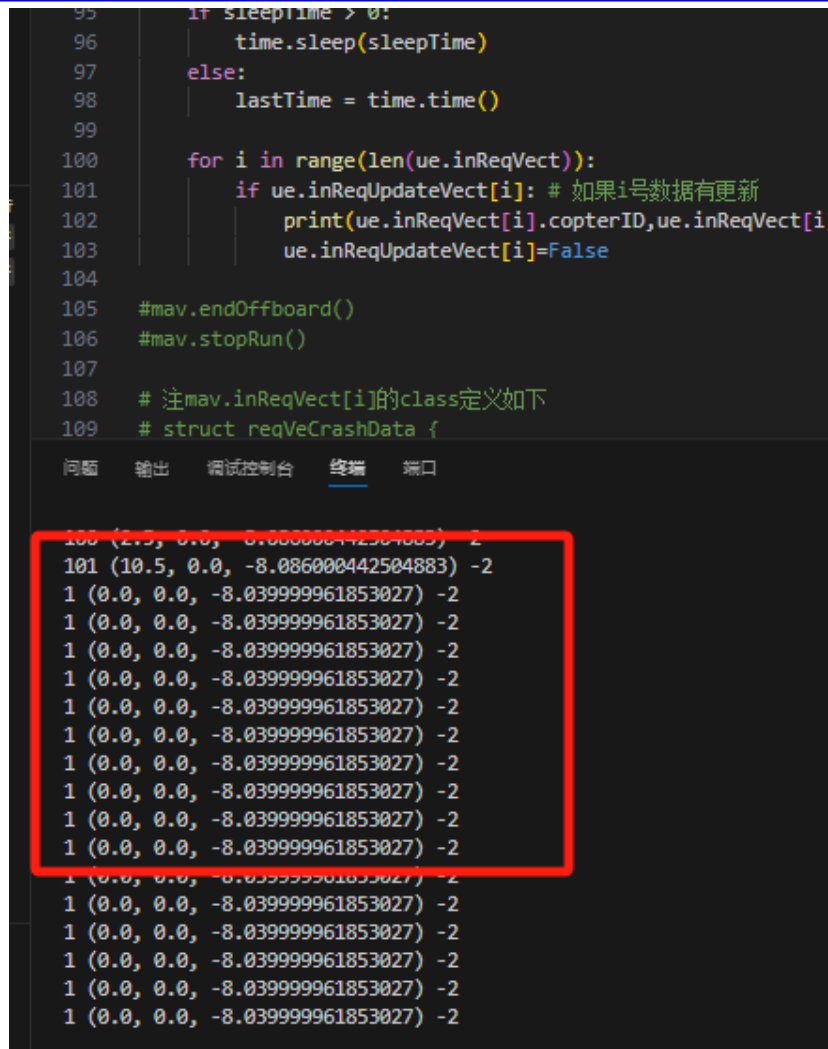
```
# struct reqVeCrashData {  
#   int checksum; //数据包校验码1234567897  
#   int copterID; //当前飞机的ID号  
#   int vehicleType; //当前飞机的样式  
#   int CrashType; //碰撞物体类型, -2表示地面, -1表示场景静态物体, 0表示无碰撞, 1以上表示被碰飞机的ID号  
#   double runnedTime; //当前飞机的时间戳  
#   float Vele[3]; // 当前飞机的速度  
#   float PosE[3]; //当前飞机的位置  
#   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  
# } 4i1d29f20s
```



2. The base interface uses

2.10 Acquiring the UE Interface — Experimental Process and Effect

- Refer to "[8.RflySimVision\0.ApiExps\1-UsageAPI\4.RflySim3DAPI\1.RflySim3DPosGet](#)" for the routine to get the UE interface. Run the `GetUE4PosAPI.bat` script to start the simulation. After `CopterSim` displays `3DFixed`, use `VSCold` to run `GetUE4PosAPI.py` by debugging single-step execution
- The running effect is shown in the right figure. Different interfaces can be obtained in each step, including all interfaces for dynamically creating object positions and collision data.



```
95     if sleeptime > 0:
96         time.sleep(sleepTime)
97     else:
98         lastTime = time.time()
99
100     for i in range(len(ue.inReqVect)):
101         if ue.inReqUpdateVect[i]: # 如果i号数据有更新
102             print(ue.inReqVect[i].copterID,ue.inReqVect[i]
103                 ue.inReqUpdateVect[i]=False
104
105     #mav.endOffboard()
106     #mav.stopRun()
107
108     # 注mav.inReqVect[i]的class定义如下
109     # struct reqVeCrashData {
110 (2.5, 0.0, 0.00000042384883) -2
111 101 (10.5, 0.0, -8.086000442504883) -2
112 1 (0.0, 0.0, -8.039999961853027) -2
113 1 (0.0, 0.0, -8.039999961853027) -2
114 1 (0.0, 0.0, -8.039999961853027) -2
115 1 (0.0, 0.0, -8.039999961853027) -2
116 1 (0.0, 0.0, -8.039999961853027) -2
117 1 (0.0, 0.0, -8.039999961853027) -2
118 1 (0.0, 0.0, -8.039999961853027) -2
119 1 (0.0, 0.0, -8.039999961853027) -2
120 1 (0.0, 0.0, -8.039999961853027) -2
121 1 (0.0, 0.0, -8.039999961853027) -2
122 1 (0.0, 0.0, -8.039999961853027) -2
123 1 (0.0, 0.0, -8.039999961853027) -2
124 1 (0.0, 0.0, -8.039999961853027) -2
125 1 (0.0, 0.0, -8.039999961853027) -2
```



Outline

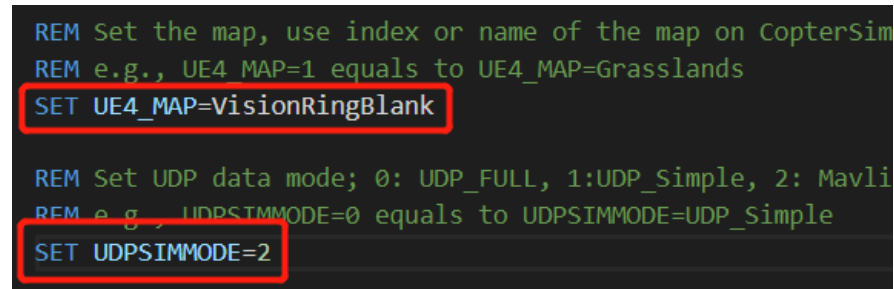
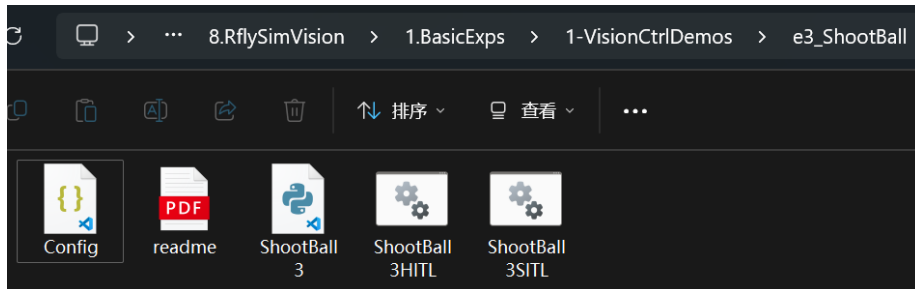
1. General introduction
2. The base interface uses
3. Visual control example
4. Visual AI is advanced
5. Distributed visual simulation



3. Visual control example

3.1 UAV Impact on Small Ball Experiment-Routine Introduction

- In Windows Explorer, open and enter the "[8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e3_ShootBall](#)" folder, as shown below.
- "ShootBall 3.py" is the main Python program of this routine, "ShootBall 3HITL.bat" is the script to quickly start hardware-in-the-loop simulation, and "ShootBall3 SITL. Bat" is the script to quickly start software-in-the-loop simulation. As shown in the lower right figure, the difference between the latter two relative to the S/HITLRun shortcut of the desktop is that the "UE4 _ MAP" map scene variable selects the flat grass scene "VisionRingBlank" for vision; Next, "UDP SIMMDe" communication UDP mode selects "Mavlink _ Full" mode for easy communication with Python; finally, 1 RflySim3D window is opened.





3. Visual control example

3.1 Experiment of UAV impacting small ball — code analysis

- Open the "ShootBall 3.py" file with VS Code and view the source code and comments below.

```
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
9 import VisionCaptureApi
10 import UE4CtrlAPI
11 ue = UE4CtrlAPI.UE4CtrlAPI()
12
13 vis = VisionCaptureApi.VisionCaptureApi()
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
17
18 isSuss = vis.sendReqToUE4() # 向RflySim3D发送取图请求, 并验证
19 if not isSuss: # 如果请求取图失败, 则退出
20     sys.exit(0)
21 vis.startImgCap() # 开启共享内存取图
```

1.导入依赖库

2.导入PX4控制接口
以及视觉取图接口

3.新建取图接口实例

4.加载json传感器参数列表

5.向UE4发送取图请求

6.开始读取图片, 并存储在Img列表

```
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)
27
28 # Create MAVLink control API instance
29 mav = PX4MavCtrl.PX4MavCtrl(1)
30 # Init MAVLink data receiving loop
31 mav.InitMavLoop()
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])
35 time.sleep(0.5)
36
37 # Function to calculate the location and radius of red ball
38 def calc_centroid(img):
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)
```

7.修改UE4的窗口显示分辨率
修改UE4的刷新率, 等于取图帧率

8.创建1号飞机控制实例

9.开启PX4数据监听

10.创建一个红色球

11.计算球的中心



3. Visual control example

3.1 Experiment of UAV impacting small ball — code analysis

```
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 procssImage():
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)):
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

14.饱和函数



3. Visual control example

3.1 Experiment of UAV impacting small ball — code analysis

```
107 lastTime = time.time()      15.记录开始仿真时间
108 startTime = time.time()
109 # time interval of the timer
110 timeInterval = 1/30.0 #here is 0.0333s (30Hz)
111 flag = 0                    16.设置更新频率间隔
112
113 # parameters
114 width = 640
115 height = 480
116 channel = 4                17.设置控制器参数
117 min_prop = 0.000001
118 max_prop = 0.3
119 K_z = 0.003 * 640 / height
120 K_yawrate = 0.005 * 480 / width
121
122 num=0
123 lastClock=time.time()      18.设置帧率统计参数
124
125 # Start a endless loop with 30Hz, timeInterval=1/30.0
126 ctrlLast = [0,0,0,0]
127 while True:
128     lastTime = lastTime + timeInterval
129     sleepTime = lastTime - time.time()    19.死循环, 使程序一直能运行下去
130     if sleepTime > 0:
131         time.sleep(sleepTime) # sleep until the desired clock
132     else:                                20.通过休眠函数, 确保每隔1/30s执行后续
133         lastTime = time.time() 代码
134     # The following code will be executed 30Hz (0.0333s)
135
136     num=num+1
137     if num%100==0:                    21.统计并显示程序执行帧率
138         tiem=time.time()
139         print('MainThreadFPS: '+str(100/(tiem-lastClock)))
140         lastClock=tiem
141
142     # 22.在第5秒进入模式1
143     if time.time() - startTime > 5 and flag == 0:
144         # The following code will be executed at 5s
145         print("5s, Arm the drone")
146         mav.initOffboard()            23.初始化外部控制Offbord模式
147         flag = 1                    解锁飞机, 发送期望位置5米高
148         mav.SendMavArm(True) # Arm the drone
149         print("Arm the drone!, and fly to NED 0,0,-5")
150         mav.SendPosNED(0, 0, -5, 0) # Fly to target position [0, 0, -5], i.e
```




3. Visual control example

• 3.1 Experiment of UAV impacting small ball — code analysis

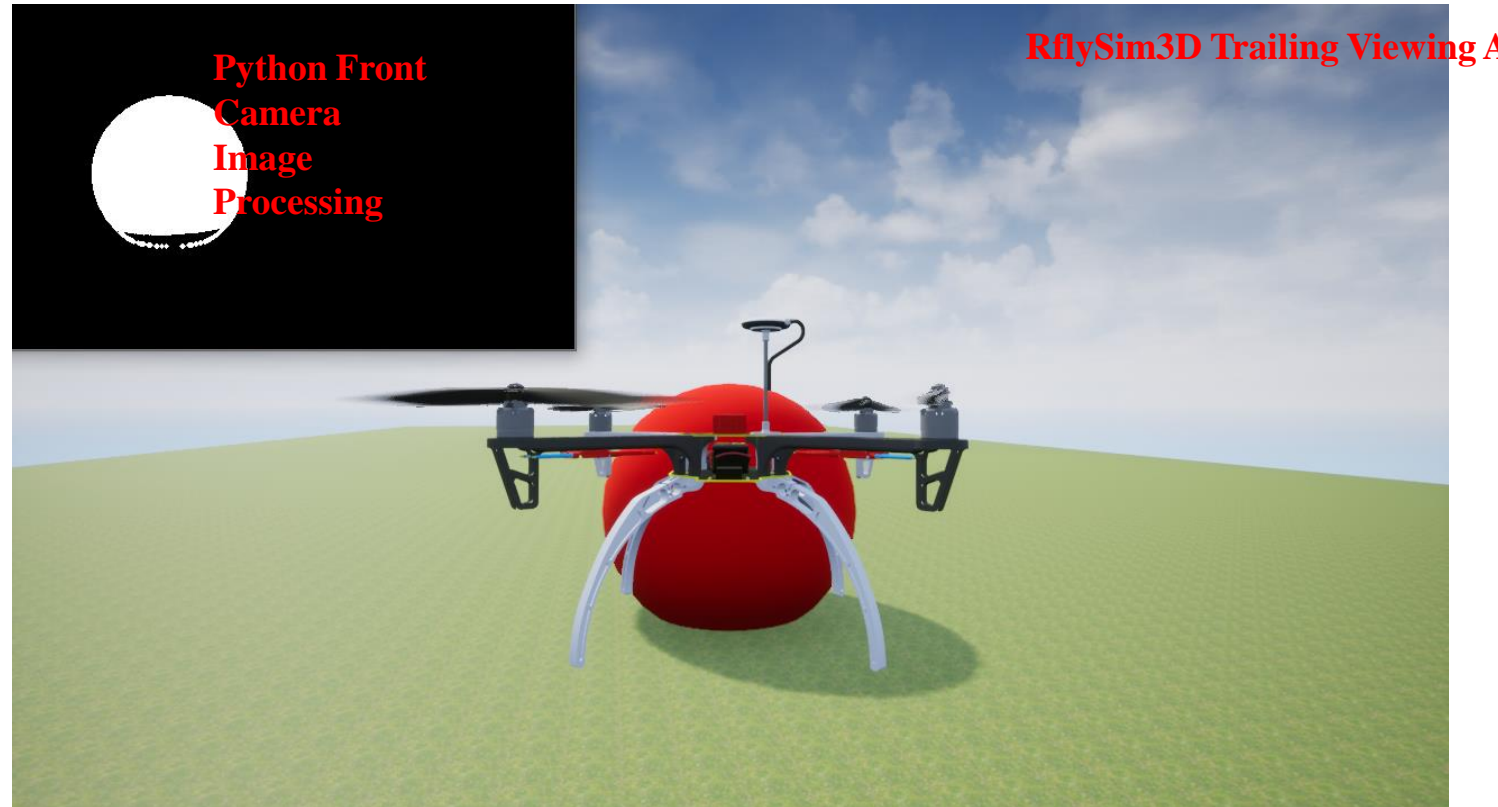
```
141 if time.time() - startTime > 15 and flag == 1: 158
142     flag = 2 24.第15s进入模式2 159
143     # The following code will be executed at 15s 160
144     mav.SendPosNED(-30,-5, -5, 0) # Fly to target position [-30, 161
145     print("15s, fly to pos: -30,-5, -5") 25.飞机向后飞到30m后 162
146
147 if time.time() - startTime > 25 and flag == 2: 164
148     flag = 3 26.第25s进入模式3 165
149     # Show CV image and set the position 166
150     if vis.hasData[0]: 167
151         img_bgr=vis.Img[0] 168
152         cv2.imshow("dilated", img_bgr) 27.显示接收图片 169
153         cv2.waitKey(1) 170
154         #time.sleep(0.5) 171
155
156     print("25s, start to shoot the ball.") 172
173
174
175 if time.time() - startTime > 25 and flag == 3: 176
177     ctrlNow = procImage() 28.从25开始进入模式4 177
178     ctrl = sat(ctrlNow,5) 178
179     # add a inertial component here to restrain the speed variation rate 179
180     if ctrl[0]-ctrlLast[0] > 0.5: 180
181         ctrl[0]=ctrlLast[0]+0.05 29.处理图像, 得到飞机控制量, 181
182         # 进行必要的滤波和限幅 182
183     elif ctrl[0]-ctrlLast[0] < -0.5: 183
184         ctrl[0]=ctrlLast[0]-0.05 184
185     if ctrl[1]-ctrlLast[1] > 0.5: 185
186         ctrl[1]=ctrlLast[1]+0.05 186
187     elif ctrl[1]-ctrlLast[1] < -0.5: 187
188         ctrl[1]=ctrlLast[1]-0.05 188
189     ctrlLast = ctrl 189
190     # if control signals is obtained, send to Pixhawk 190
191     # 30.将速度控制指令发给PX4 191
192     mav.SendVelFRD(ctrl[0], ctrl[1], ctrl[2], ctrl[3]) 192
```



3. Visual control example

3.1 UAV Impact on Small Ball Experiment — Experimental Effect

- Double-click to run the "ShootBall3SITL.bat" file to start the software in-the-loop simulation system. You can also plug in the flight control, run the HILS script "ShootBall3HITL.bat", and enter the serial number to start the HITL simulation.
- Run the "ShootBall 3.py" program again. Generate a red sphere in front, let the aircraft fly to the left rear for some distance, and turn on visual tracking, fly to the front of the ball and stop.



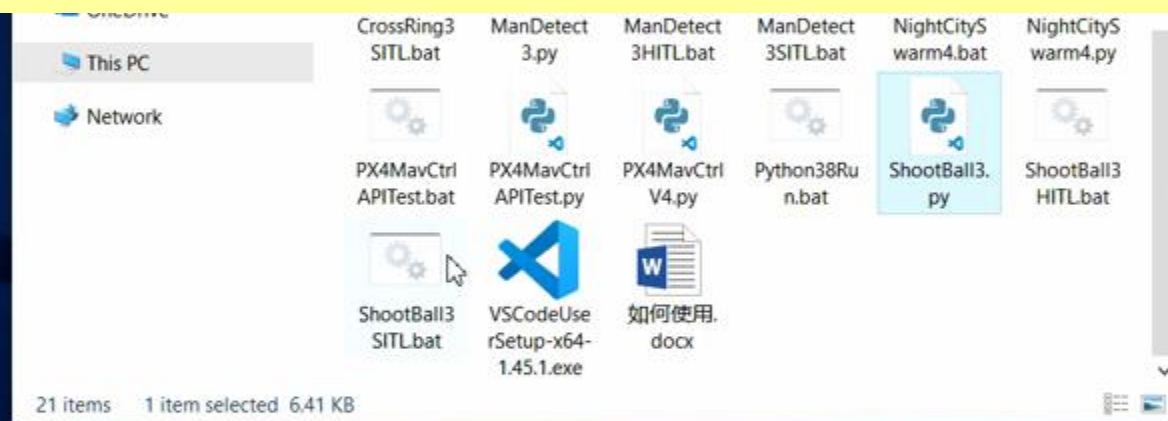
RflySim. Visual Navigation Control of Multi-rotor Unmanned Aerial Vehicle — — Experiment of Impacting Small Ball

This video can be viewed at:

Youku: https://v.youku.com/v_show/id_XNDcwNjA4NTYwNA==.html

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

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



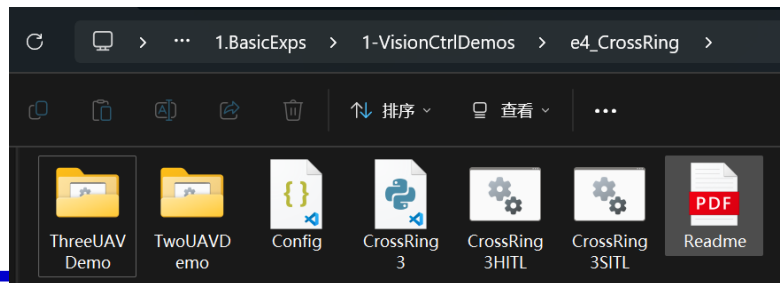
双击例程提供的**bat**脚本可以快速启动软/硬在环软件和期望数量的三维程序



3. Visual control example

3.2 UAV Loop Penetration Experiment-Routine Introduction

- In Windows Explorer, open and enter the "[8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e4_CrossRing](#)" folder, as shown below.
- Where "CrossRing3.py" is the main Python program for this routine; The difference between "ShootBall3HITL.bat" and "ShootBall3SITL.bat" relative to the previous example of hitting a small ball is that the "UE4 _ MAP" map scene variable has been selected for the visual ring-piercing scene "VisionRing". Note: Mavlink _ Full "Mode is also selected for the UDPSIMMode communication UDP mode.
- The subdirectories "TwoUAVDemo" and "ThreeUAVDemo" are two examples of distributed vision control, generating two or three UAVs to independently process their own vision and complete independent ring crossing tasks in the same scene.





3. Visual control example

3.2 UAV Ring Penetration Experiment — Key Code Analysis

```
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
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:
56         task = "range3"
57     elif pos_x < 140:
58         task = "land"
59     else:
60         task = "finish"
61     return task
```

1.偏航角速率控制
饱和函数

2.基于飞行距
离的任务切换
逻辑

```
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]
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
100 cv2.drawContours( img_bgr, squares, -1, (0, 255, 0), 3)
101 cv2.imshow("img_bgr", img_bgr)
102
103 cv2.waitKey(1)
104 height, width, channel = img_bgr.shape
```

3.目标检测函数

4.方形靶标检测函数





3. Visual control example

3.2 UAV Ring Penetration Experiment — Key Code Analysis

```
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)
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
```

5.圆形靶标检测函数

```
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()
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)
```

6.目标接近控制函数



3. Visual control example

3.2 UAV Ring Penetration Experiment — Key Code Analysis

```
185 # main function
186 if __name__ == '__main__':
187     mav = PX4MavCtrl.PX4MavCtrler(1)
188     mav.InitMavLoop()
189     print("Simulation Start.")
190     print("Enter Offboard mode.")
191     time.sleep(5)
192     mav.initOffboard()
193     time.sleep(0.5)
194     mav.SendMavArm(True) # Arm the drone
195     mav.SendPosNED(0, 0, -5, 0) # Fly to target position 0,0, -5
196
197     # Show CV image and set the position
198     if vis.hasData[0]:
199         img_bgr=vis.Img[0]
200         cv2.imshow("img_bgr", img_bgr)
201         cv2.waitKey(1)
202         #time.sleep(0.5)
203
204     time.sleep(5)
205
206     # start crossing ring task
207     approachObjective()
```

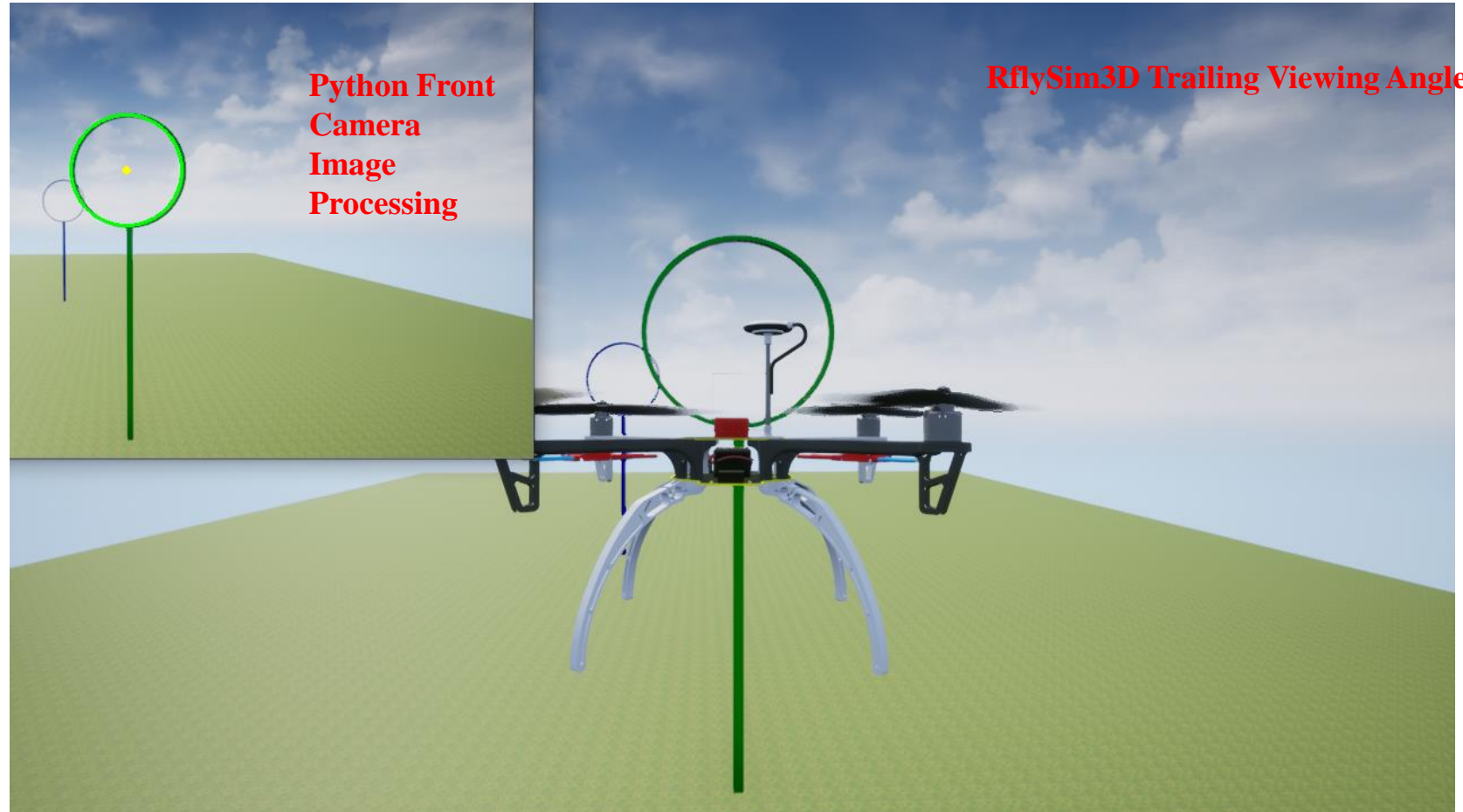
7.主函数。包含
MAVLink接口初始化
和RflySim3D视角控
制



3. Visual control example

3.2 UAV Ring Penetration Test-Operation Effect

- Double-click to run the "CrossRing3SITL.bat" file to start the software in-the-loop simulation system, and then run the "CrossRing3.py" program.
- After taking off, the plane passes through three rings in order, and finally lands automatically.
- To use the hardware-in-the-loop simulation, after setting the flight control, run the "CrossRing3HITL.bat" script and input the flight control serial number to start the hardware-in-the-loop simulation system.



RflySim. Visual Navigation Control of Multi-rotor Unmanned Aerial Vehicle — — Multi-rotor Loop Penetration Experiment

This video can be viewed at:

Youku: https://v.youku.com/v_show/id_XNDcwNjA4NTYwNA==.html

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

Station B: <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

link2

e6

SP/RflySimAPIs/P

Python



3. Visual control example

3.3 Dual UAV Distributed Control-Routine Introduction

- "[8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e4_CrossRing\TwoUAVDemo](#)" is shown in the right figure.
- In contrast to the stand-alone routines, there are two Json files and two Python control main programs, which correspond to the images and controls of the two aircraft.
- The main differences between the camera parameters corresponding to the two Json files are:
 - 1) SeqID is different, which is used to distinguish image memory blocks;
 - 2) TargetCopter is bound to different aircraft, which are bound to aircraft 1 and 2 respectively.

```
1 {
2   "VisionSensors": [
3     {
4       "SeqID": 0,
5       "TypeID": 1,
6       "TargetCopter": 1,
7       "TargetMountType": 0,
8       "DataWidth": 720,
9       "DataHeight": 405,
10      "DataCheckFreq": 30,
11      "SendProtocol": [0, 127, 0, 0, 1, 9999, 0, 0],
12      "CameraFOV": 90,
13      "SensorPosXYZ": [0.3, 0, 0],
14      "SensorAngEular": [0, 0, 0],
15      "otherParams": [0, 0, 0, 0, 0, 0, 0, 0]
16    }
17  ]
18 }
19
```

```
1 {
2   "VisionSensors": [
3     {
4       "SeqID": 1,
5       "TypeID": 1,
6       "TargetCopter": 2,
7       "TargetMountType": 0,
8       "DataWidth": 720,
9       "DataHeight": 405,
10      "DataCheckFreq": 30,
11      "SendProtocol": [0, 127, 0, 0, 1, 10000, 0, 0],
12      "CameraFOV": 90,
13      "SensorPosXYZ": [0.3, 0, 0],
14      "SensorAngEular": [0, 0, 0],
15      "otherParams": [0, 0, 0, 0, 0, 0, 0, 0]
16    }
17  ]
18 }
19
```



3. Visual control example

3.3 Dual UAV Distributed Control-Key Code Analysis

- The Python control program for aircraft 1 is "CrossRing3_vehicle 1.py", while that for aircraft 2 is "CrossRing3_vehicle 2.py". The two programs are basically the same as the threading routine in the previous section.
- The main difference between the two is that the jsonLoad function of VisionCaptureApi calls different Json file paths; secondly, the UAV control interface PX4MavCtrler configures different CopterSim communication ports (corresponding to different aircraft). Both take the first camera image in the JSON definition camera list, so both use Img [0] to read the image.

```
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
```



3. Visual control example

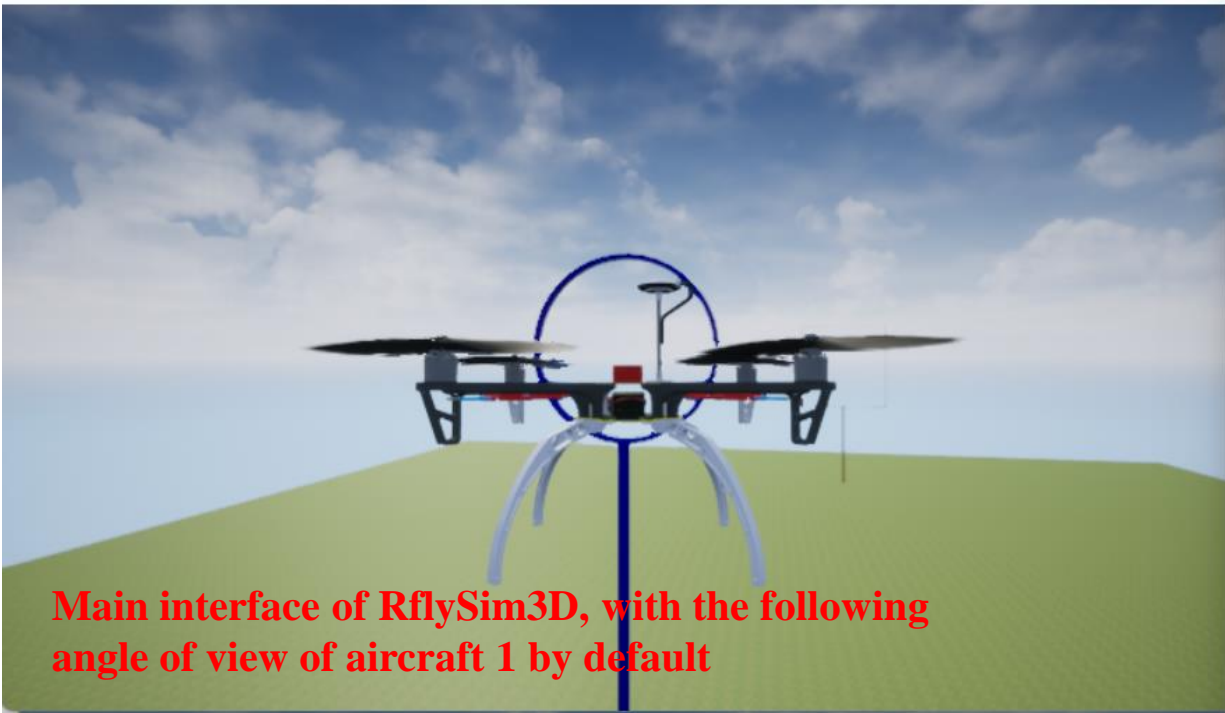
3.3 Dual UAV Distributed Control-Experimental Results

- Enter "8.RflySimVision \ 1.Basic Exps \ 1-VisionCtrlDemos \ E4 _ CrossRing \ TwoUAVDemo", Double-click to run the script "CrossRing3SITL.bat" or "CrossRing3HITL.bat" to start the software/hardware in-the-loop simulation of two aircrafts.
- After the two aircraft are initialized (RflySim3D will prompt), double-click "Python38Run.bat" twice to open the two Python environments; enter "python CrossRing3 _ vehicle 1.py" in the first Python window (do not press Enter first); Type "python CrossRing3 _ vehicle2.py" in the second Python environment (without pressing Enter first).
- Go back to the first Python window and press the Enter key to run the visual loop threading program for aircraft 1. After a few seconds, switch to the Python window 2 and press the Enter key to start the visual loop threading program for aircraft 2. It can be seen that the aircraft take off in turn and penetrate the ring.

```
C:\Windows\system32\cmd.e. x + v 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.py
^

C:\Windows\system32\cmd.e. x + v 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
```



Main interface of RflySim3D, with the following angle of view of aircraft 1 by default

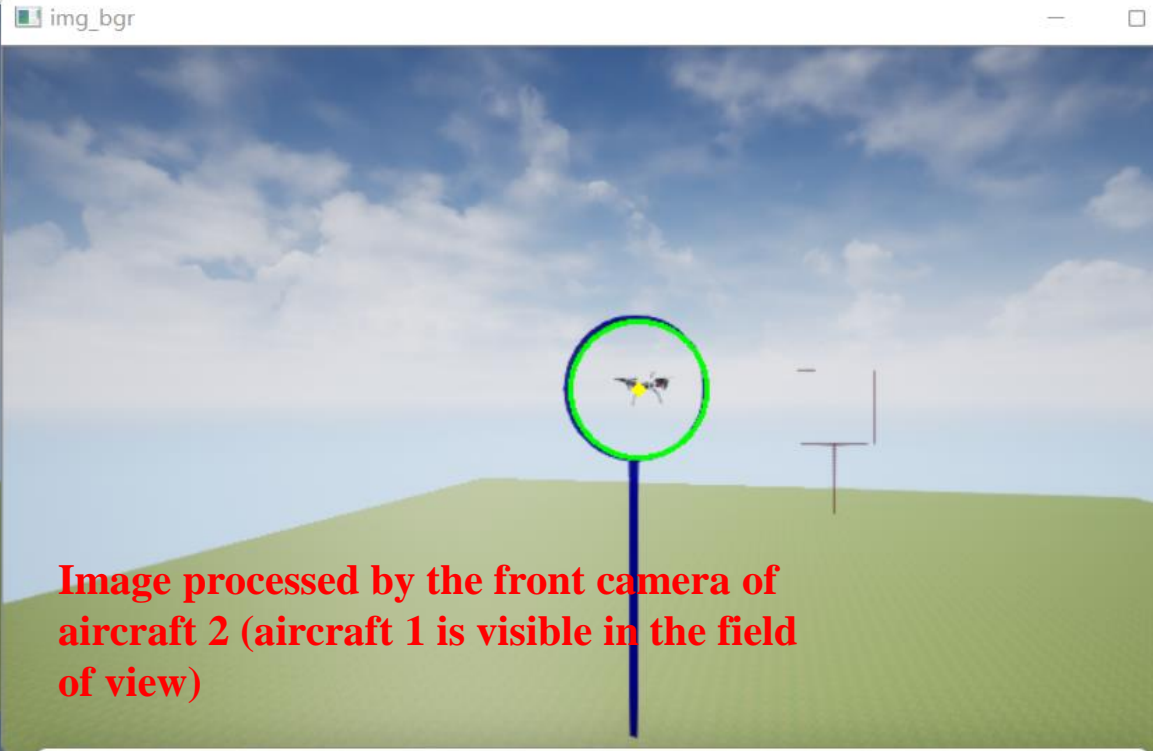


Image processed by the front camera of aircraft 2 (aircraft 1 is visible in the field of view)

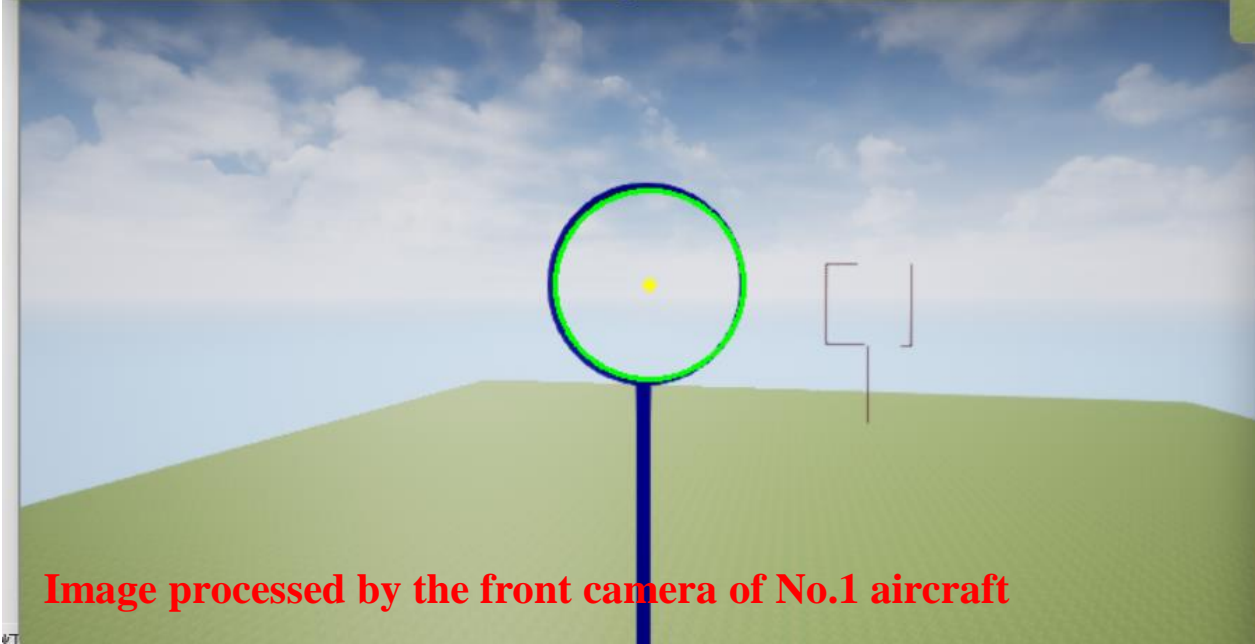


Image processed by the front camera of No.1 aircraft



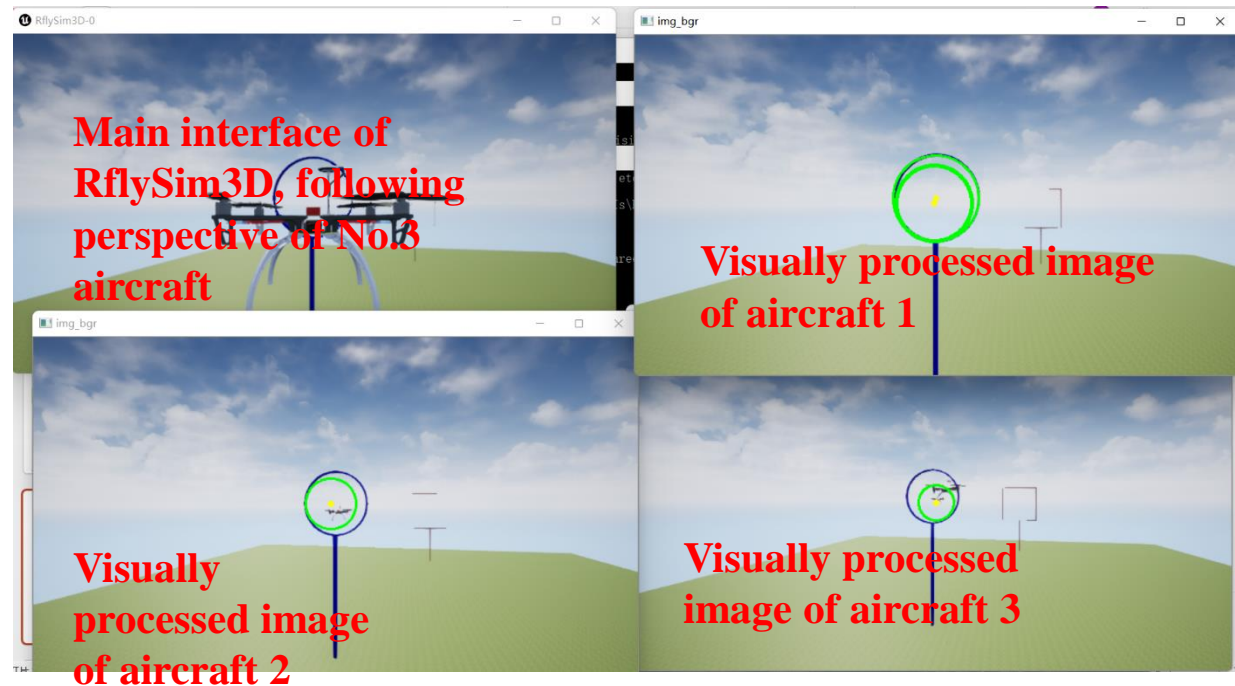
QGroundControl Trajectory Observations



3. Visual control example

3.3 Dual UAV Distributed Control – Scaled to 3 Aircraft

- With the example of distributed control of two UAVs, we can easily expand the number of UAVs to more, such as three aircraft. Note: The free version only supports up to two image outputs, so it can only be used for dual vision.
- Enter the directory "[8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\04_CrossRing\ThreeUAVDemo](#)" and follow the same method.
- You can turn on the visual control of three aircraft, as shown in the lower right picture.
- Three planes can be seen taking off in turn, and the plane behind can see the plane in front in view.
- Because the front aircraft will block the ring, there will be fluctuations in the ring recognition of this mode, which will reduce the control effect.

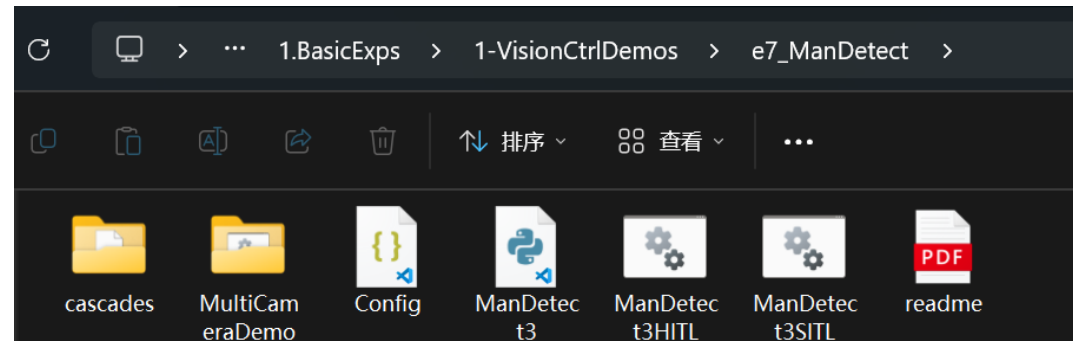




3. Visual control example

3.4 Binocular Vision Face Recognition Experiment-Routine Introduction

- In Windows Explorer, open and enter the "[8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e7_ManDetect](#)" folder, as shown below.
- Where, "ManDetect3.py" is the main Python program of this routine; the folder "cascades" contains some feature files in XML format for face recognition; The difference between "ManDetect3HITL.bat" and "ManDetect3SITL.bat" relative to the desktop shortcut is that the communication UDP mode of "UDPSIMMode" also selects "Mavlink _ Full" mode; The folder "MultiCameraDemo" contains an advanced example of two aircraft each with a binocular camera, a total of four camera images (full version only).





3. Visual control example

3.4 Binocular Vision Face Recognition Experiment — Key Source Code Analysis

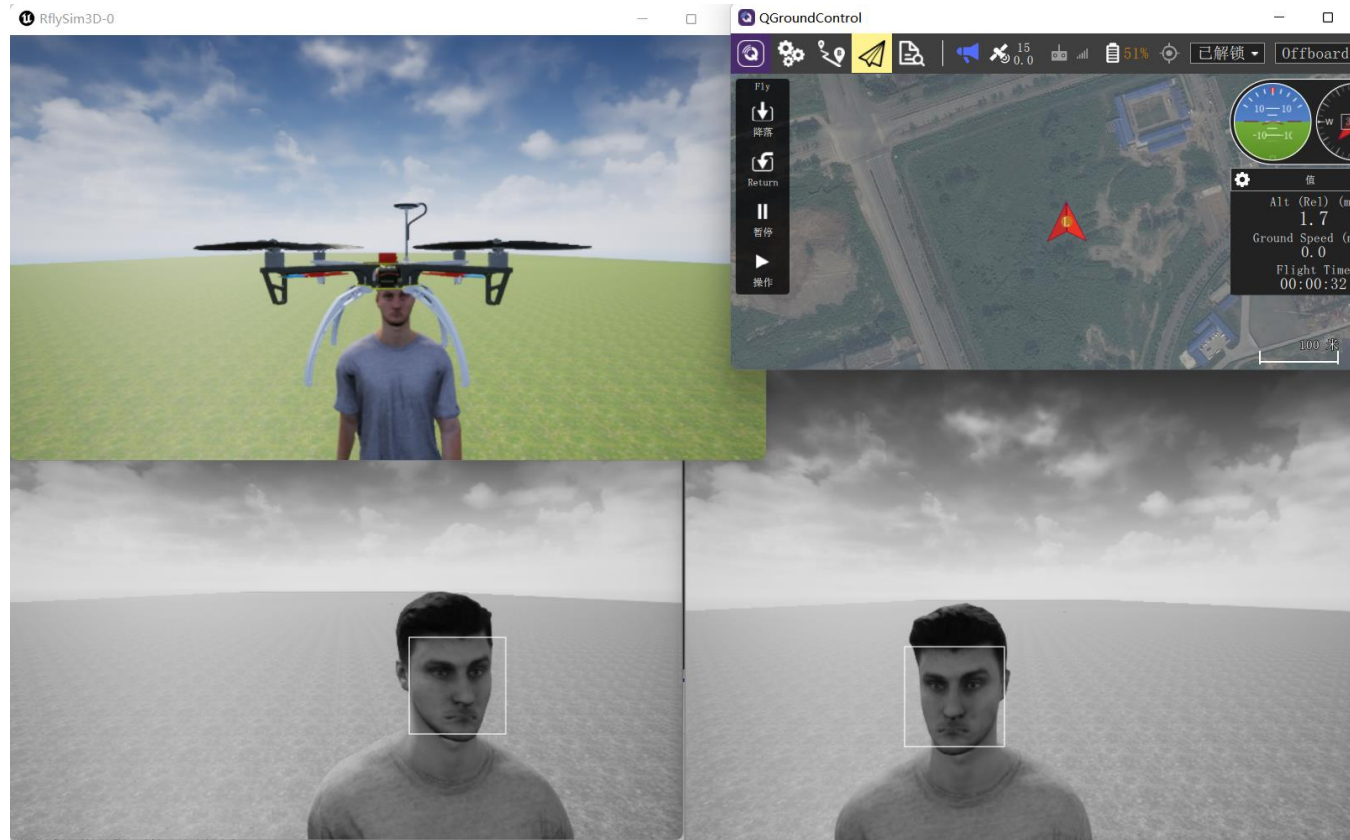
```
59  timeInterval = 1/30.0 # time interval of the timer      1.图像处理帧率30Hz
60  face_cascade=cv2.CascadeClassifier(sys.path[0]+'\\cascades\\haarcascade_frontalface_default.xml')
61
62  num=0          2.载入本目录的XML人脸特征库
63  lastClock=time.time()
64  while True:
65      #gImgList = sca.getCVImgList(ImgInfoList)
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:
83              pic2=cv2.rectangle(pic2,(x,y),(x+w,y+h),(255,0,0),1)      8.处理2号飞机图像
84              cv2.imshow("pic2",pic2)
```




3. Visual control example

3.4 Binocular vision face recognition experiment — running effect

- Run "ManDetect3SITL.bat" or "ManDetect3HITL.bat" to start the HW/SW in-the-loop simulation, and then run the main control program "ManDetect 3.py".
- In RflySim3D, a walking person is generated and set to face the plane. After the plane takes off, the face recognition algorithm is turned on, and the binocular frame selects the face.
- Assignment 1: Update the location of people in real time, realize the simulation of people walking, and write the aircraft tracking controller.
- Job 2: Change to the forward-looking + downward-looking camera, and verify the tracking + optical flow algorithm.



RflySim: Obtain binocular camera images and perform face recognition

This video can be viewed at:

Youku: https://v.youku.com/v_show/id_XNDcwNjA4NzgxMg==.html

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

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





3. Visual control example

3.4 Binocular Vision Face Recognition Experiment — Extended Binocular

- Enter the directory "[8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e7_ManDetect\MultiCameraDemo](#)". You can view the key source code analysis of the Config. JSON file (lower left two figures) and the MultiCameraDemo. Py file. (Bottom right)

```
"VisionSensors":[
{
  "SeqID":0, 1号飞机左相机
  "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":[0,0,0,0,0,0,0,0]
},
{
  "SeqID":1 1号飞机右相机
  "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":[0,0,0,0,0,0,0,0]
}
]
```

```
{
  "SeqID":2, 2号飞机左相机
  "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":[0,0,0,0,0,0,0,0]
},
{
  "SeqID":3, 2号飞机右相机
  "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":[0,0,0,0,0,0,0,0]
}
}
```

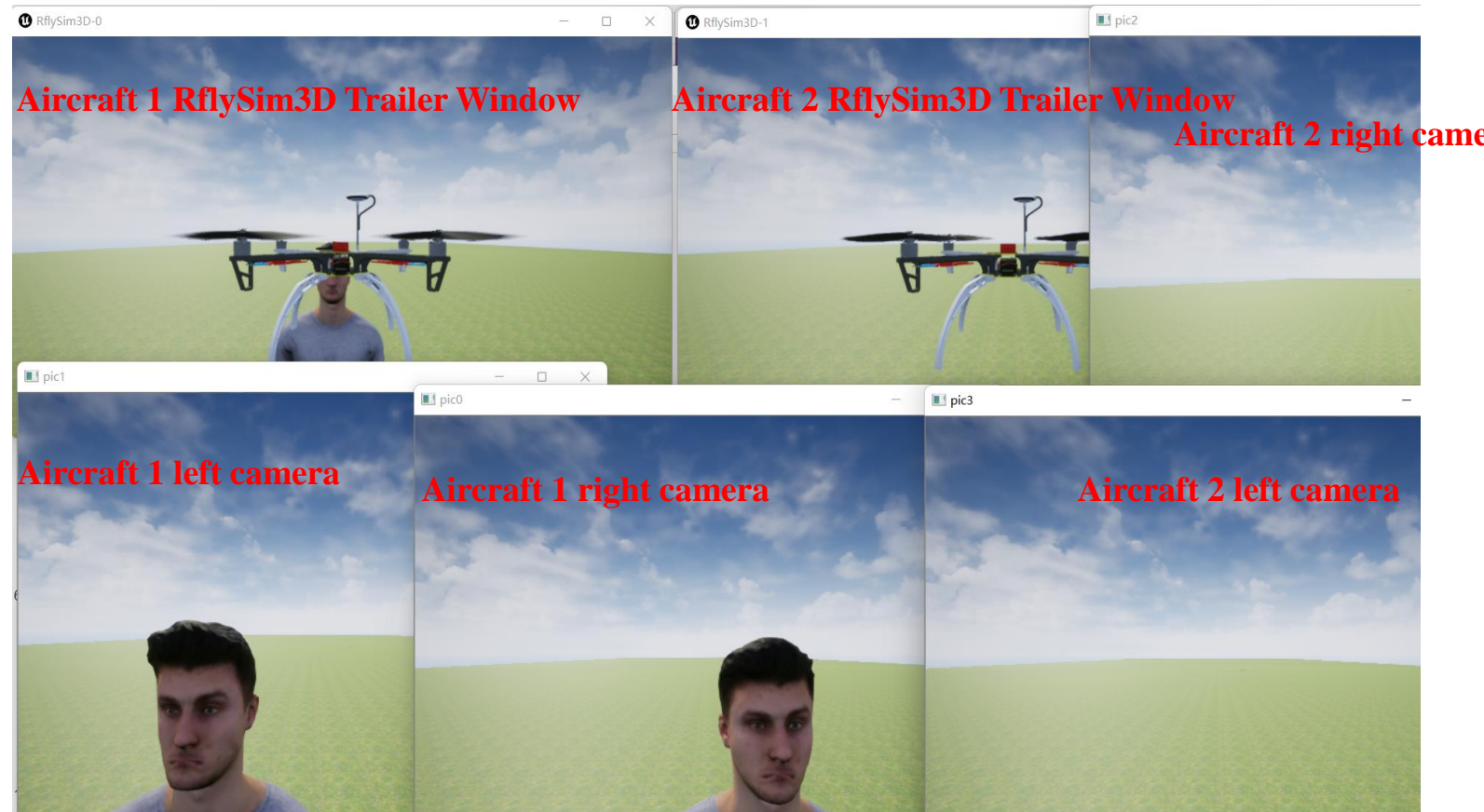
```
40 # send vehicle position command to create a man, w
41 # the man is located before the drone, and rotated
42 ue.sendUE4Cmd(100,30,0,[1,0,-8.086],[0,0,math.pi])
43 time.sleep(1) 1.向所有窗口发送创建人物命令
44
45 # send command to change object with copterID=100
46 ue.sendUE4Cmd('RflyChange3DModel 100 16')
47 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 s
50 ue.sendUE4Cmd('RflyChangeViewKeyCmd B 1',0)
51 time.sleep(0.5)
52 # change the first window to size 720x405
53 ue.sendUE4Cmd('r.setres 720x405w',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 720x405w',1)
61 time.sleep(2)
```



3. Visual control example

3.4 Binocular Vision Face Recognition Experiment — Extended Binocular

- First double click the "MultiCameraDemoSITL. Bat" or "MultiCameraDemoHITL. Bat" to open the dual computer software/hardware in the loop simulation of two CopterSim and two RflySim3D.
- Then run the "MultiCameraDemo. Py" "to see the aircraft take off and get the right.





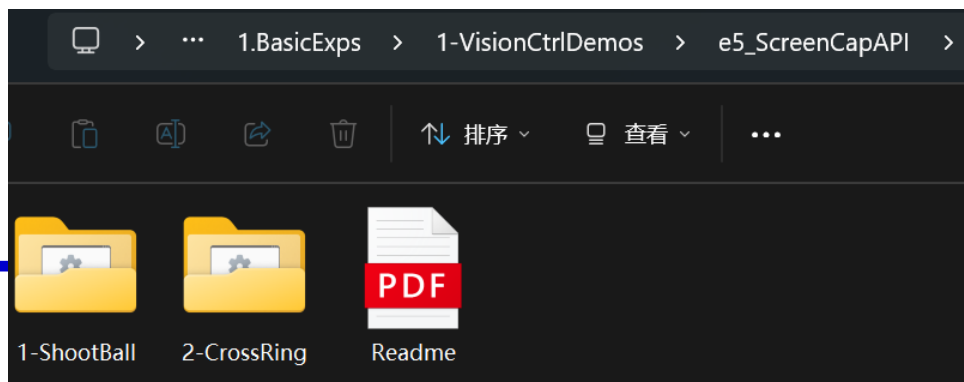
3. Visual control example

Note: It is recommended that you use the Python 38 environment that comes with the platform to run the routines. If you use another Python environment, be sure to install the following components:

```
pip3 install d3dshot pywin32
```

3.5 Screen Capture Interface-Routine Introduction

- In Windows Explorer, open and enter the "[8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e5_ScreenCapAPI](#)" folder, as shown in the following figure, which contains examples of hitting balls and piercing rings. The interface used is the way to take screenshots.
- UE4 internal shared memory image transmission: high efficiency, one window can achieve multi-channel transmission, the image is not the final effect.
- Screen screenshot: low efficiency, a window can only be an image, the image is the final rendering effect.
- To sum up, the screen mapping mode image is the final best effect, while the shared memory mode image is not the best intermediate rendering result, but the screen mapping mode is inefficient, so it is used as an alternative scheme.
- Note: Screen mapping can also be used for any other 3D engine (such as FlightGear, Unity).



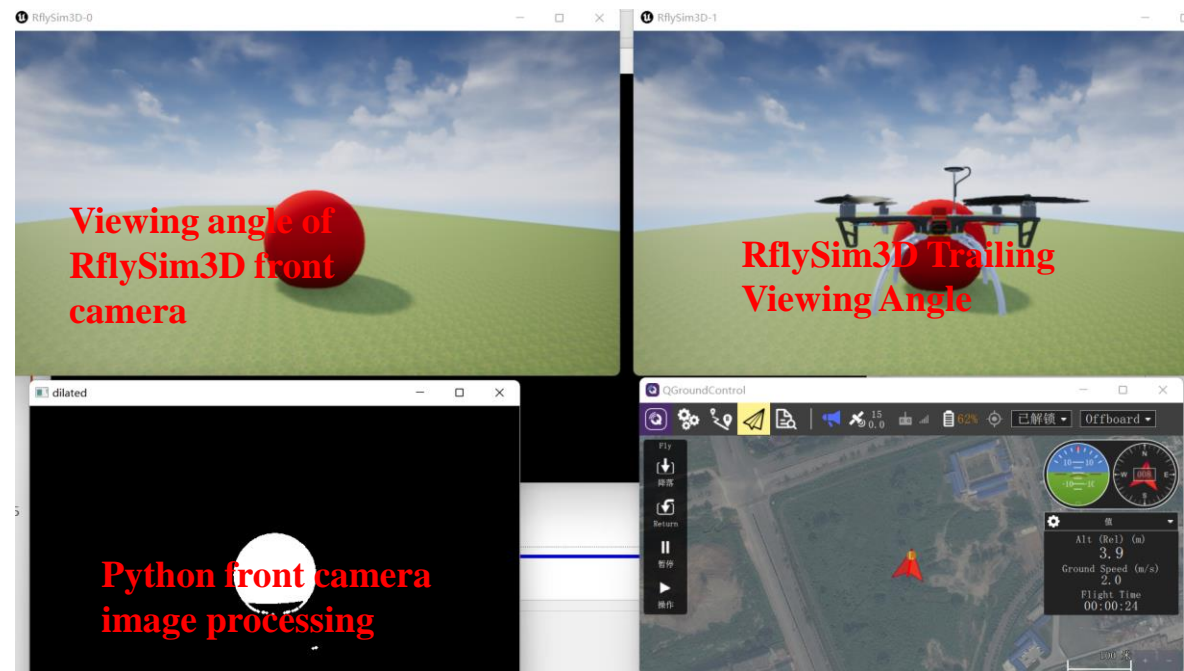


3. Visual control example

Note: When running this routine, please double click to open the SITL. Bat or HITL. Bat script file directly, and do not run it in the administrator mode.

3.5 Screen capture interface-impact ball experiment

- Open the [8.RflySimVision\1.BasicExps\1-VisionCtrlDemos\e5_ScreenCapAPI\1-ShootBall](#) folder. Double-clicking ShootBall3SITL.bat or ShootBall3HITL.bat will open a simulation closed loop of CopterSim aircraft, and open two RflySim3D windows at the same time, one for displaying the front camera and the other for global observation.
- Run "ShootBall 3.py" to see the effect on the right.
- Because of the way the screen takes pictures, the front-facing camera image on the left side must always be kept at the forefront, otherwise there will be occlusion.
- Python images will be smaller than the window, which is affected by the DPI setting.

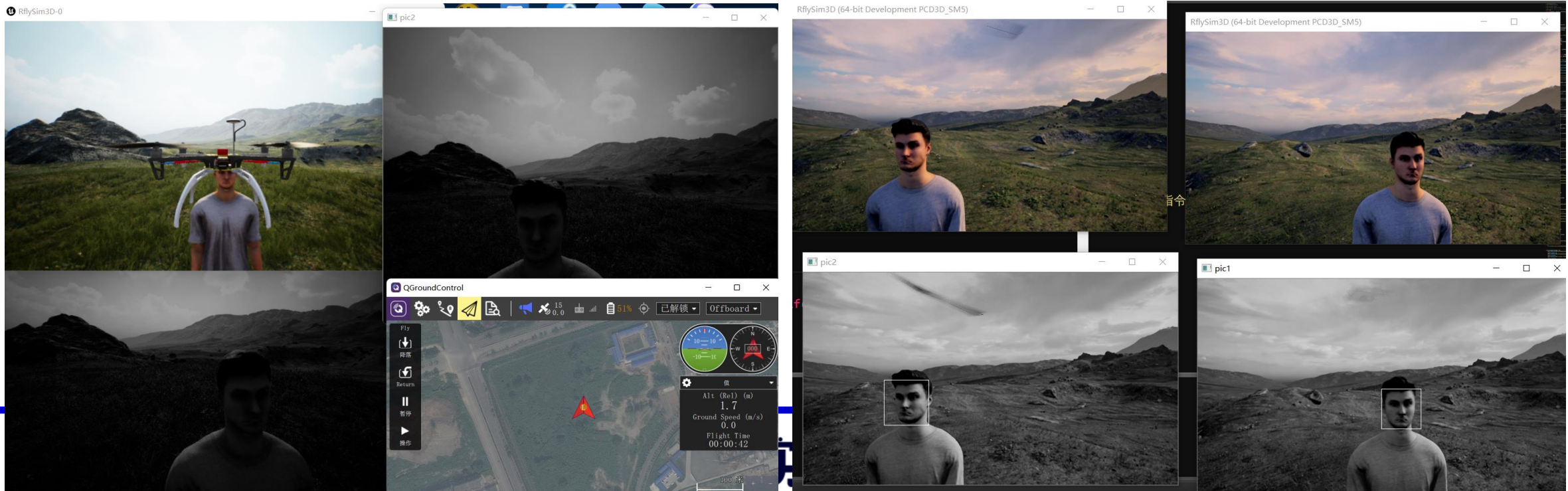




3. Visual control example

3.5 Screenshot Interface — Comparison of Experimental Effects of Binocular Face Recognition

- The lower left is the drawing of UE4 shared memory, and the lower right is the screenshot mode. It can be seen that the change of light and shadow in the way of screen drawing is consistent with the actual window display effect, and there is a certain display limitation in the backlighting of UE4 internal drawing.





Outline

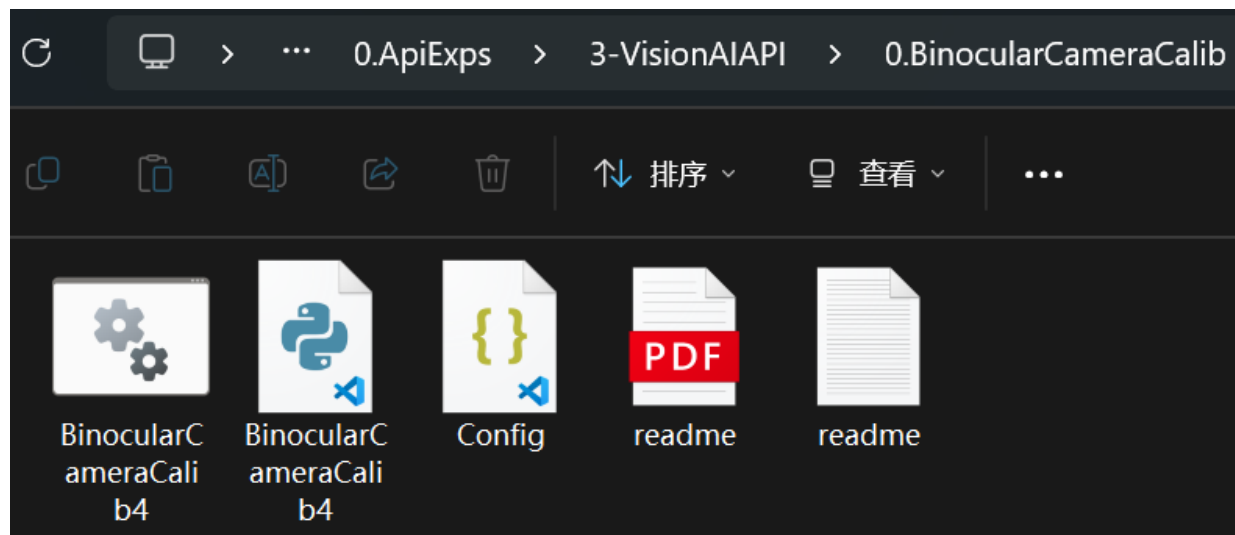
1. General introduction
2. The base interface uses
3. Visual control example
4. Visual AI is advanced
5. Distributed visual simulation



4. Visual AI is advanced

4.1 Binocular Camera Calibration Experiment-Routine Introduction

- In Windows Explorer, open and enter the "[8.RflySimVision\0.ApiExps\3-VisionAIAPI\0.BinocularCameraCalib](#)" folder, as shown below.
- Where "Binocular CameraCalib4.py" is the main Python program for this routine; "Binocular CameraCalib4.bat" is a batch startup script, double-clicking it will automatically open three RflySim3D windows (left and right cameras + trailing global viewing angle).





4. Visual AI is advanced

4.1 Binocular Camera Calibration Experiment — Core Code Analysis

- Open the "Binocular CameraCalib4.py" file with VS Code.
- The key code lines are shown in the right figure. This script can obtain images of multiple windows at the same time.
- Please read and learn the rest of the code according to the previous explanation.

```
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.随机设置标靶的位置和姿态
63      if num%30==0:
64          TargePos = [InitTargePos[0]+random.randint(0,100)/1
65          TargeAng = [InitTargeAng[0]+random.randint(-50,50)/1
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. Visual AI is advanced

4.1 Binocular Camera Calibration Experiment — Experimental Effect

- After running "Binocular CameraCalib4.bat", run "Binocular CameraCalib4.py".
- Open multiple RflySim3D scenes, create a new aircraft, configure binocular position information, create a new target, and place the targets according to random rules.
- Job 1: After obtaining the images of the left and right cameras, implement the online calibration algorithm.
- Homework 2: Store the images of the left and right cameras locally in the form of pictures, and then use the calibration toolbox of MATLAB to calculate the parameters.



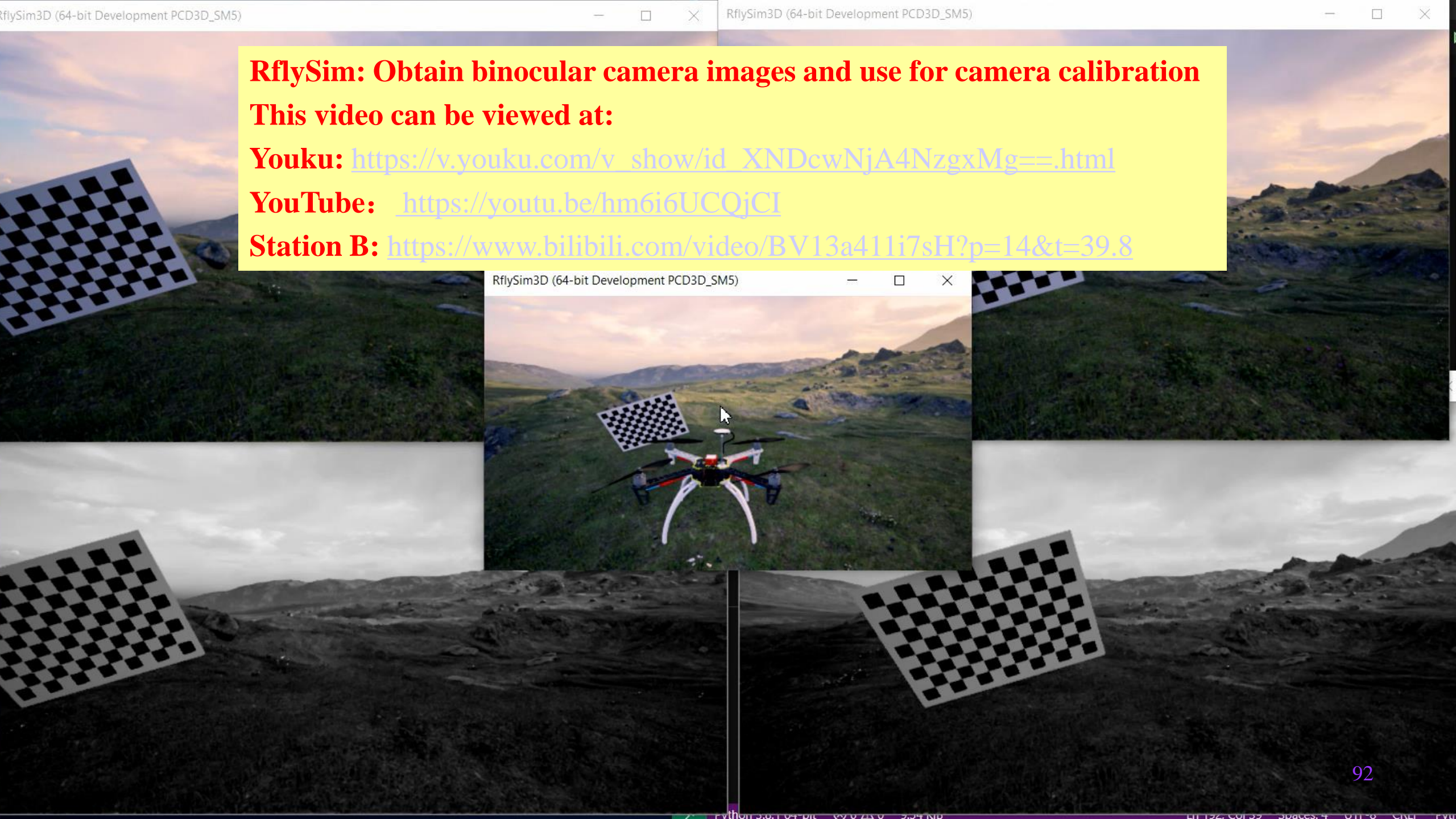
RflySim: Obtain binocular camera images and use for camera calibration

This video can be viewed at:

Youku: https://v.youku.com/v_show/id_XNDcwNjA4NzgxMg==.html

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

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





4. Visual AI is advanced

4.2 Virtual Camera Calibration Principle-Introduction

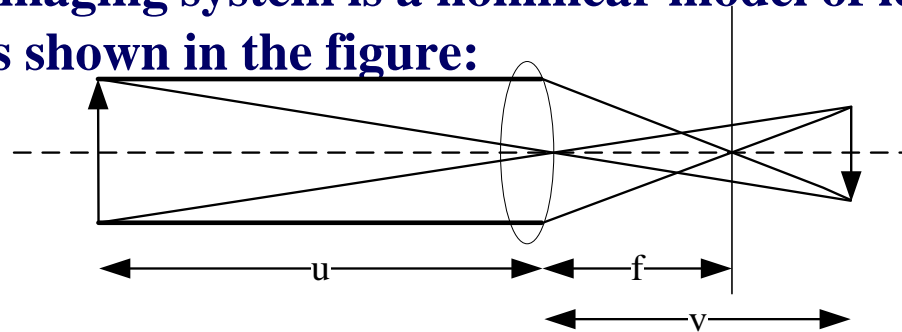
- **Camera calibration concept:** In the process of image measurement and computer vision, in order to determine the relationship between the three-dimensional geometric position of a point in a space object and its corresponding point in the image, it is necessary to establish a geometric model of camera imaging, and the parameters of the model are the parameters of the camera. The process of solving the parameters is called camera calibration.



4. Visual AI is advanced

4.2 Principle of virtual camera calibration — camera model

- The process of digital camera image shooting is actually a process of optical imaging. The imaging process of the camera involves four coordinate systems: the world coordinate system, the camera coordinate system, the image coordinate system and the pixel coordinate system.
- Ideal perspective model-pinhole imaging model: The camera model is a simplification of the optical imaging model. At present, there are two kinds of models: linear model and nonlinear model. The actual imaging system is a nonlinear model of lens imaging. The most basic lens imaging principle is shown in the figure:



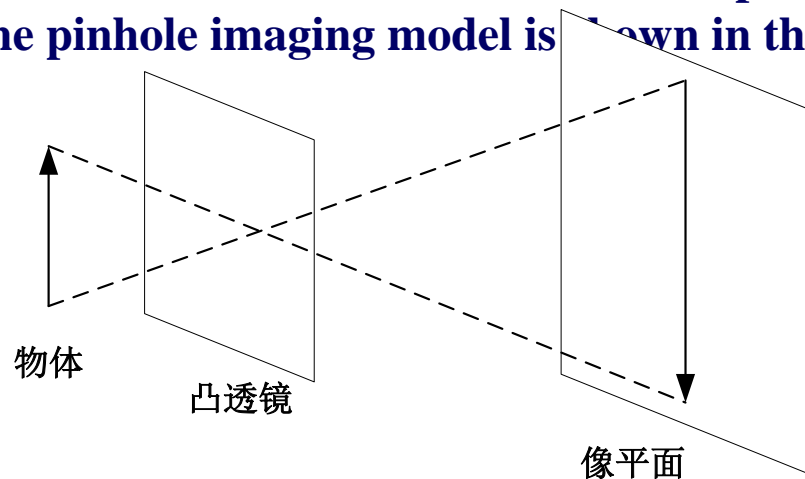
- Where u is the object distance, f is the focal length, and v is the distance. The three satisfy the relation $\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$



4. Visual AI is advanced

4.2 Principle of virtual camera calibration — camera model

- The lens of a camera is a group of lenses. When light rays parallel to the main optical axis pass through the lens, they converge on a point. This point is called the focus. The distance from the focus to the center of the lens is called the focal length. The lens of a digital camera is equivalent to a convex lens, and the photosensitive element is located near the focus of the convex lens. When the focal length is approximated by the distance from the center of the convex lens to the photosensitive element, it becomes a pinhole imaging model. The pinhole imaging model is shown in the figure.



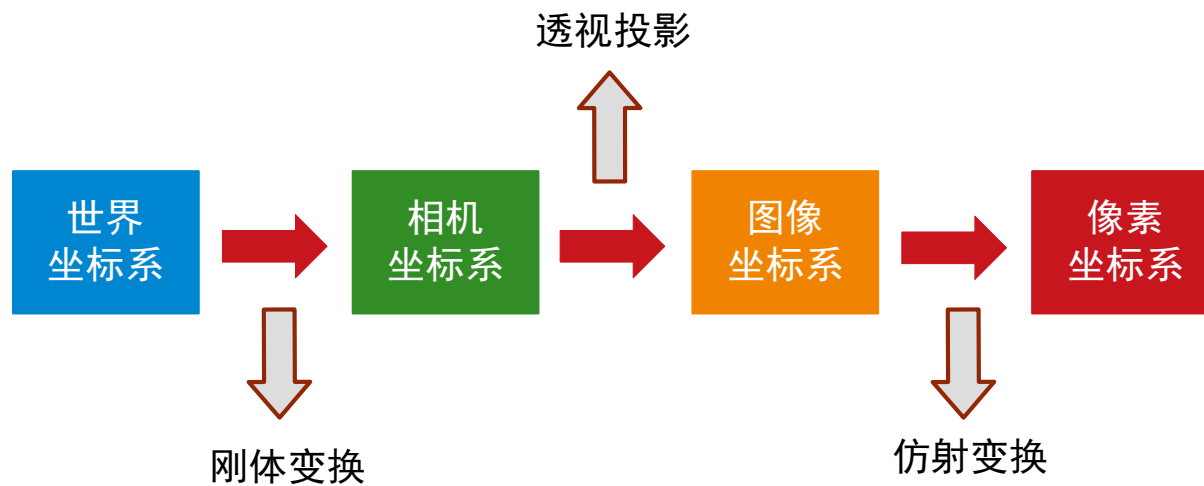
- The pinhole imaging model is the most widely used model for camera imaging. In this model, the relationship between the spatial coordinates of the object and the image coordinates is linear, so the solution of the camera parameters comes down to the solution of linear equations.



4. Visual AI is advanced

4.2 Principle of virtual camera calibration — definition of coordinate system

- The camera imaging system consists of four coordinate systems: world coordinate system, camera coordinate system, image coordinate system and pixel coordinate system.





4. Visual AI is advanced

4.2 Principle of virtual camera calibration — definition of coordinate system

- The transformation relationship between these four coordinate systems is:

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

仿射变换
透视投影
刚体变换

内参矩阵
外参矩阵

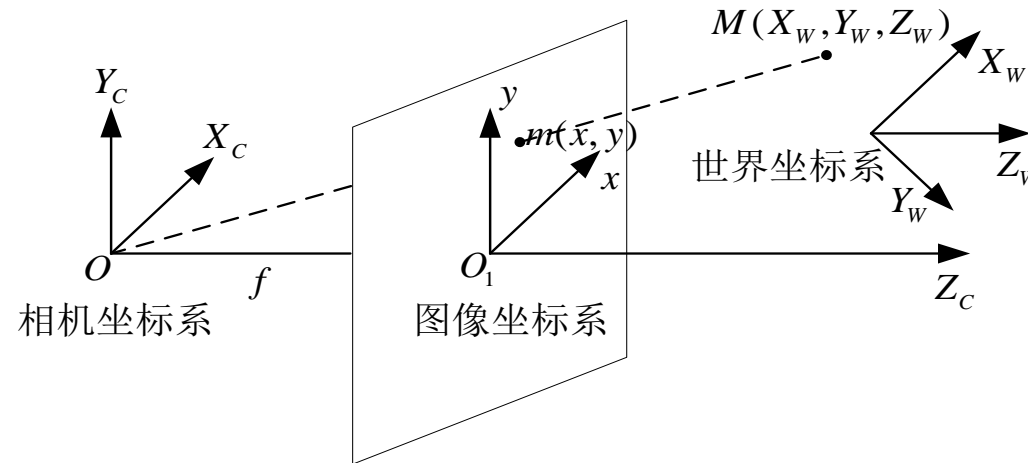
- Where, $(U, +V, +W)$ is the physical coordinate of a point in the world coordinate system, (u, v) is the pixel coordinate of the point in the pixel coordinate system, and Z is the scale factor.



4. Visual AI is advanced

4.2 Principle of virtual camera calibration — definition of coordinate system

- The relationship diagram of the four coordinate systems is shown in the following figure, where M is the three-dimensional space point, and m is the image point projected on the image plane.



- ① World coordinate system: It is the absolute coordinate system of the objective three-dimensional world, also called objective coordinate system. Because the digital camera is placed in the three-dimensional space, we need the world coordinate system to describe the position of the digital camera, and use it to describe the position of any other object placed in the three-dimensional environment to express its coordinate value.



4. Visual AI is advanced

4.2 Principle of virtual camera calibration — definition of coordinate system

- ② Camera coordinate system (optical center coordinate system): The optical center of the camera is taken as the coordinate origin, the X axis and Y axis are parallel to the X axis and Y axis of the image coordinate system respectively, and the optical axis of the camera is taken as the axis to represent its coordinate value.
- ③ Image coordinate system: the center of the CCD image plane is taken as the coordinate origin, the X axis and the Y axis are respectively parallel to the two vertical sides of the image plane, and the coordinate values are represented by. The image coordinate system represents the position of a pixel in an image in physical units, such as millimeters.
- ④ Pixel coordinate system: take the vertex of the upper left corner of the CCD image plane as the origin, X axis and Y axis are parallel to the X axis and Y axis of the image coordinate system respectively, and their coordinate values are represented by. The images captured by digital cameras are first formed into standard electrical signals, and then converted into digital images through analog-to-digital conversion. Each image is stored as an array, and the value of each element in the row and column of the image represents the gray level of the image point. Each such element is called a pixel, and the pixel coordinate system is the image coordinate system in pixels.



4. Visual AI is advanced

4.2 Calibration principle of virtual camera — calibration principle

- Why do you want to calibrate the camera? For example, when we get a picture and recognize it, the distance between the two parts is 1 pixel, but how many meters does this pixel correspond to in the real world? This requires the use of camera calibration results to convert pixel coordinates to physical coordinates to calculate the distance.
- If we want to model an imaging system and then calculate the corresponding parameters, the necessary parameters are the camera.

Internal parameter matrix: $\begin{pmatrix} \frac{f}{dX} & -\frac{f \cot \theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY \sin \theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ and the external parameter matrix of the camera. $\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}$, Therefore, the camera is calibrated.

The first purpose is to obtain the intrinsic and extrinsic parameter matrices of the camera.



4. Visual AI is advanced

4.2 Virtual camera calibration principle — internal parameter matrix and external parameter matrix

• We put the matrix:

$$\begin{pmatrix} \frac{1}{dX} & -\frac{\cot \theta}{dX} & u_0 \\ 0 & \frac{1}{dY \sin \theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \frac{f}{dX} & -\frac{f \cot \theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY \sin \theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

- It is called the intrinsic parameter matrix of the camera, which depends on the intrinsic parameters of the camera. Where, f is the image distance, dX , dY respectively represents the physical length of a pixel on the camera plate in the direction of X, Y , (that is, how many millimeters is a pixel on the plate), and u_0, v_0 respectively represents the coordinates of the center of the camera plate in the pixel coordinate system.



4. Visual AI is advanced

4.2 Virtual camera calibration principle — internal parameter matrix and external parameter matrix

- When the camera is an ideal camera, the relationship between the focal length, the resolution, and the field angle is: $f = \frac{\omega}{2 \tan \frac{\theta}{2}}$
- A simplified internal reference matrix can be derived:
$$\begin{pmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$
- Where f is the focal length, ω is the width of the resolution, and θ is the angle of view.
- For example, when the field angle is 90 and the resolution is (640 480), the internal reference matrix is:
$$\begin{pmatrix} 320 & 0 & 320 \\ 0 & 320 & 240 \\ 0 & 0 & 1 \end{pmatrix}$$



4. Visual AI is advanced

4.2 Virtual camera calibration principle — internal parameter matrix and external parameter matrix

- For the same camera, the internal parameter matrix of the camera depends on the internal parameters of the camera. No matter what the position relationship between the calibration board and the camera is, the internal parameter matrix of the camera does not change. This is also the reason why we can use the matrix H obtained from different pictures (the calibration board and the camera have different positions) to solve the camera internal parameter matrix A . However, the extrinsic parameter matrix reflects the positional relationship between the calibration board and the camera. For different pictures, the position relationship between the calibration board and the camera has changed, and the extrinsic parameter matrix corresponding to each picture is different.

- We call the matrix: $\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}$ the extrinsic parameter matrix of the camera, which depends on the relative position of the camera coordinate system and the world coordinate system, \mathbf{R} represents the rotation matrix, and \mathbf{T} represents the translation vector.

- That is, the camera imaging model without distortion at a single point is as follows:
$$Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{dX} & -\frac{f \cot \theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY \sin \theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$



4. Visual AI is advanced

4.2 Virtual camera calibration principle — internal parameter matrix and external parameter matrix

- Each column vector of represents the orientation of each coordinate axis of the world coordinate system in the camera coordinate system; rather, the origin of the world coordinate system is represented in the camera coordinate system. So we need to do a coordinate transformation to convert the internal coordinates of the camera to the world coordinate system.
- The rotation matrix for this model is as follows:

$$\mathbf{R}_X(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & \sin \gamma \\ 0 & -\sin \gamma & \cos \gamma \end{bmatrix} \mathbf{R}_Y(\beta) = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix} \mathbf{R}_Z(\alpha) = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Since the known condition is the Euler angles of the camera, it is easier to derive the world coordinate system from the camera coordinate system. So we use another model: assume that the point P is a point in three-dimensional space with a position of P_C in the camera coordinate system and a position of in the world coordinate system. And may be interconvert by a transformation matrix P_W and P_C , which may be subdivided into a rotation matrix \mathbf{R}' and a translation matrix \mathbf{t} . Its mathematical expression is: $P_W = \mathbf{R}' P_C + \mathbf{t}$



4. Visual AI is advanced

4.2 Virtual camera calibration principle — internal parameter matrix and external parameter matrix

- The relationship between the rotation direction and the corresponding matrix transformation: left multiplication and right multiplication
- Left Multiplication – Transforms with respect to a fixed coordinate system. (For example, the world coordinate system, for example, First Z, then Y, then X, rotate around the world coordinate system, and then multiply the rotation matrix $\mathbf{R} = \mathbf{R}_X \mathbf{R}_Y \mathbf{R}_Z$ to the left in order.)
- Right multiplication-transform with respect to its own coordinate system, and each time it changes, the next time it needs to be transformed with the new coordinate system as the standard. For example, after the first transformation, the position of the original axis changes to the axis, then the next transformation around the axis will transform around the previous axis. For example, rotate according to the body coordinate system, for example, around the body coordinate system, First Z, then Y, then X, multiply to the right in order $\mathbf{R} = \mathbf{R}_Z \mathbf{R}_Y \mathbf{R}_X$, multiply to the right in order.
- Representation of the rotation matrix rotated according to the coordinate axis of the body:

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$

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

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





4. Visual AI is advanced

4.2 Virtual camera calibration principle — internal parameter matrix and external parameter matrix

- \mathbf{R}' — a 3×3 rotation matrix =
$$\begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \gamma & 0 & \sin \gamma \\ 0 & 1 & 0 \\ -\sin \gamma & 0 & \cos \gamma \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$
 , where (θ, γ, α) Is the rotation angle of the calibration plate around the three axes of the camera coordinate system, which is called the rotation vector;
- $\mathbf{t} = (t_x, t_y, t_z)$ is the translation vector. The combination with \mathbf{R}' and \mathbf{t} is called camera external reference. The complete external parameter matrix is $\begin{pmatrix} \mathbf{R}' & \mathbf{t} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}$.

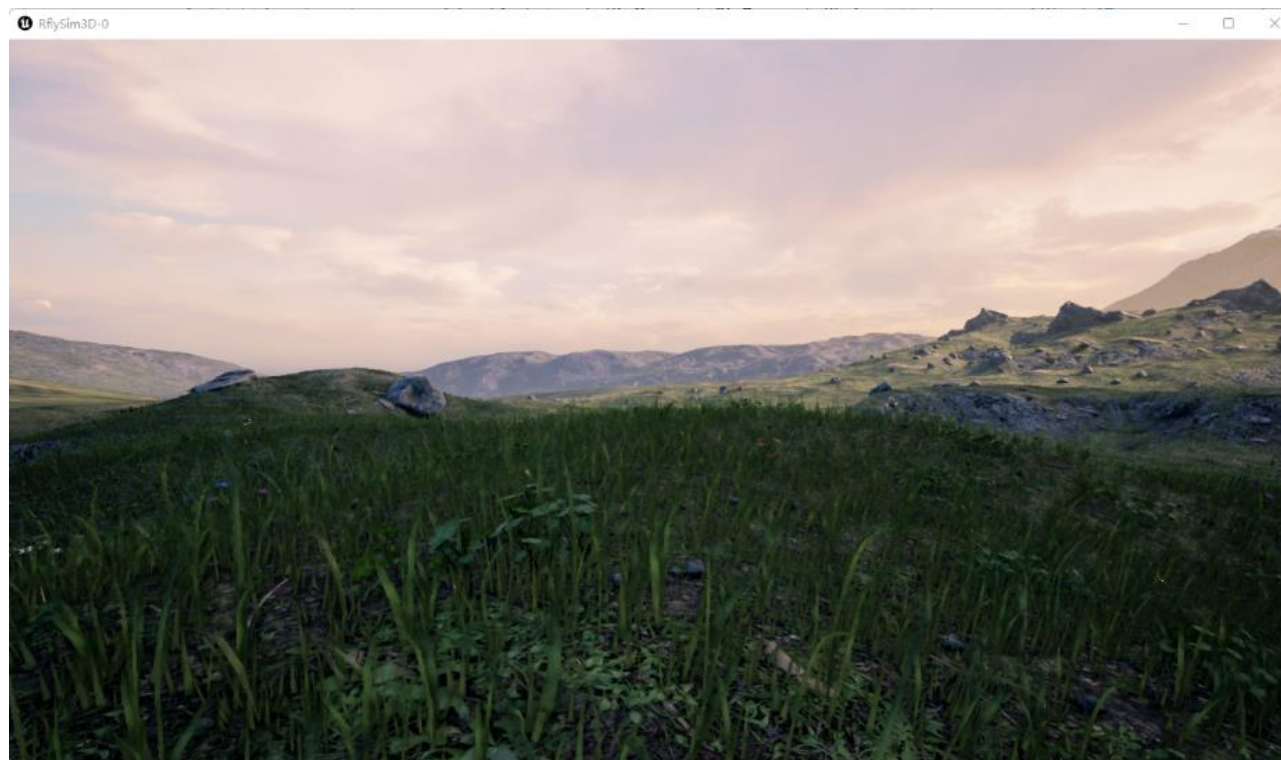


4. Visual AI is advanced

Routines see: RflySimAPIs \ 8. RflySimVision \ 0.ApiExps\3-VisionAI\2.CameraCalcDemo

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- Task 1: Calibration of Matlab calibration board
- 1. Storage of monocular camera image: start the OneCameraCal. Bat to obtain the following image:

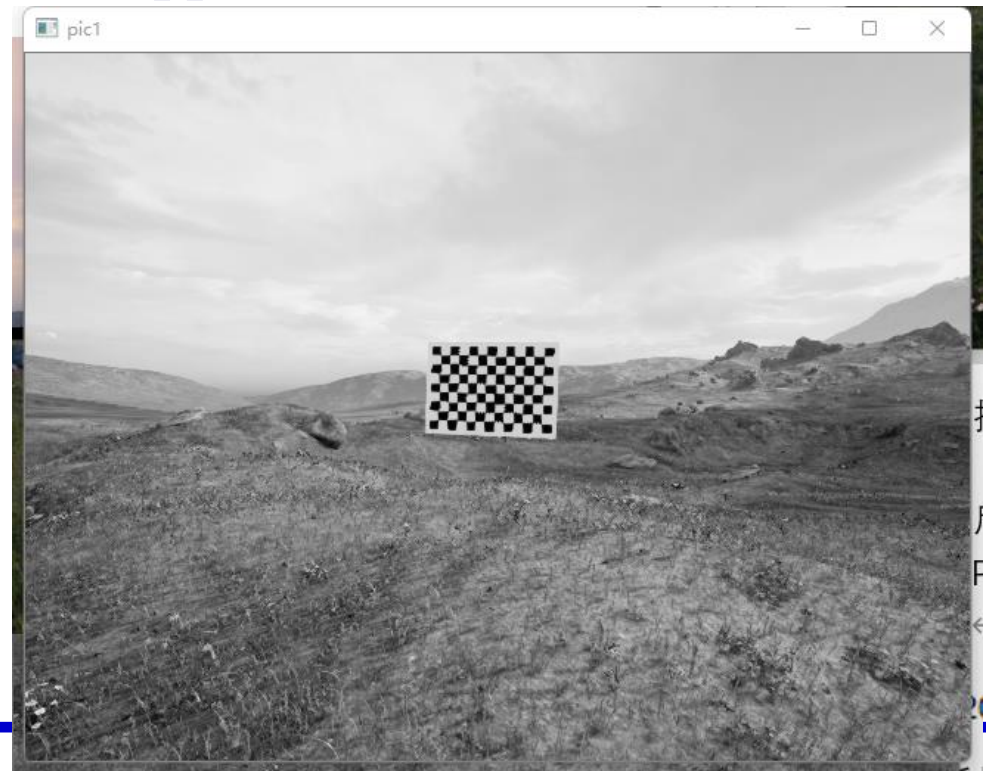
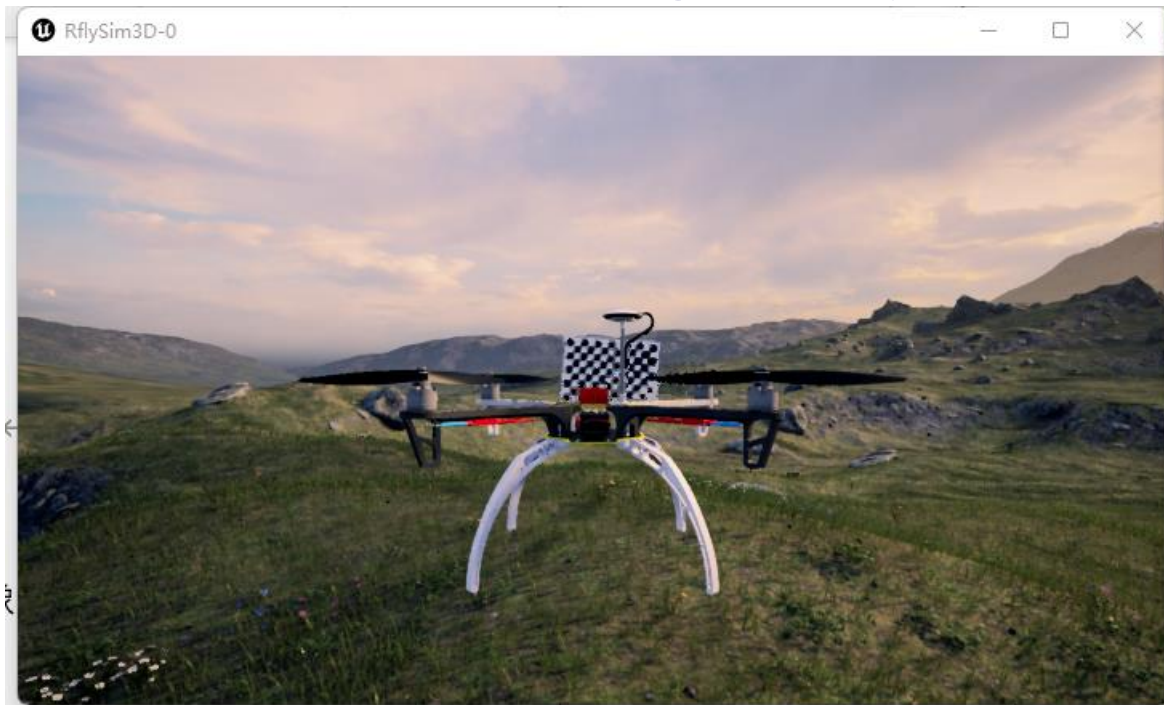




4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- **Task 1: Calibration of Matlab calibration board**
- **2. Run the One CameraCal. Py in VS Code, and the calibration board and the calibration board image shot by the camera will appear in the interface:**





4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

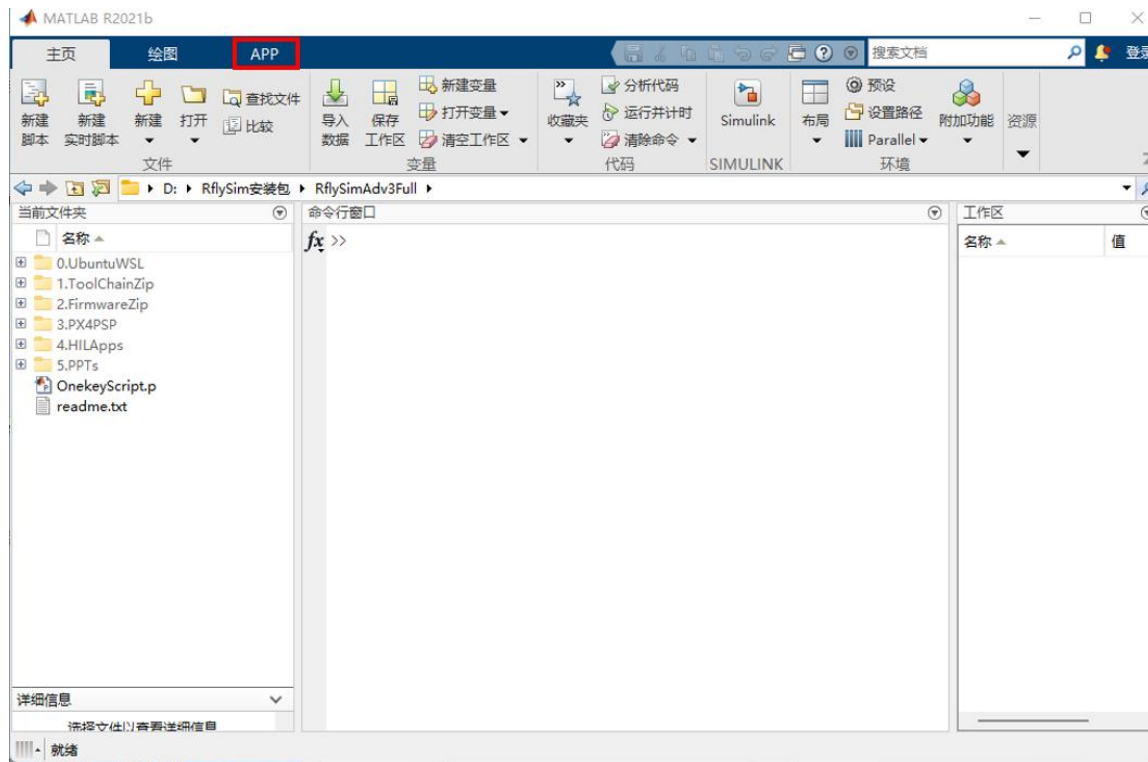
- **Task 1: Calibration of Matlab calibration board**
- **3. The captured picture is saved in a folder under the working path, for example, "[8.RflySimVision\0.ApiExps\3-VisionAI\2.CameraCalcDemo\20220207_220418](#)".**
- **If the distance between the calibration board and the camera is too far or too close, it will affect the test results. You can modify the first value in InitTargePos to change the distance. After getting about thirty pictures, the program can be stopped.**



4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- Task 1: Calibration of Matlab calibration board
- 4. Open MATLAB and click on the app bar above.

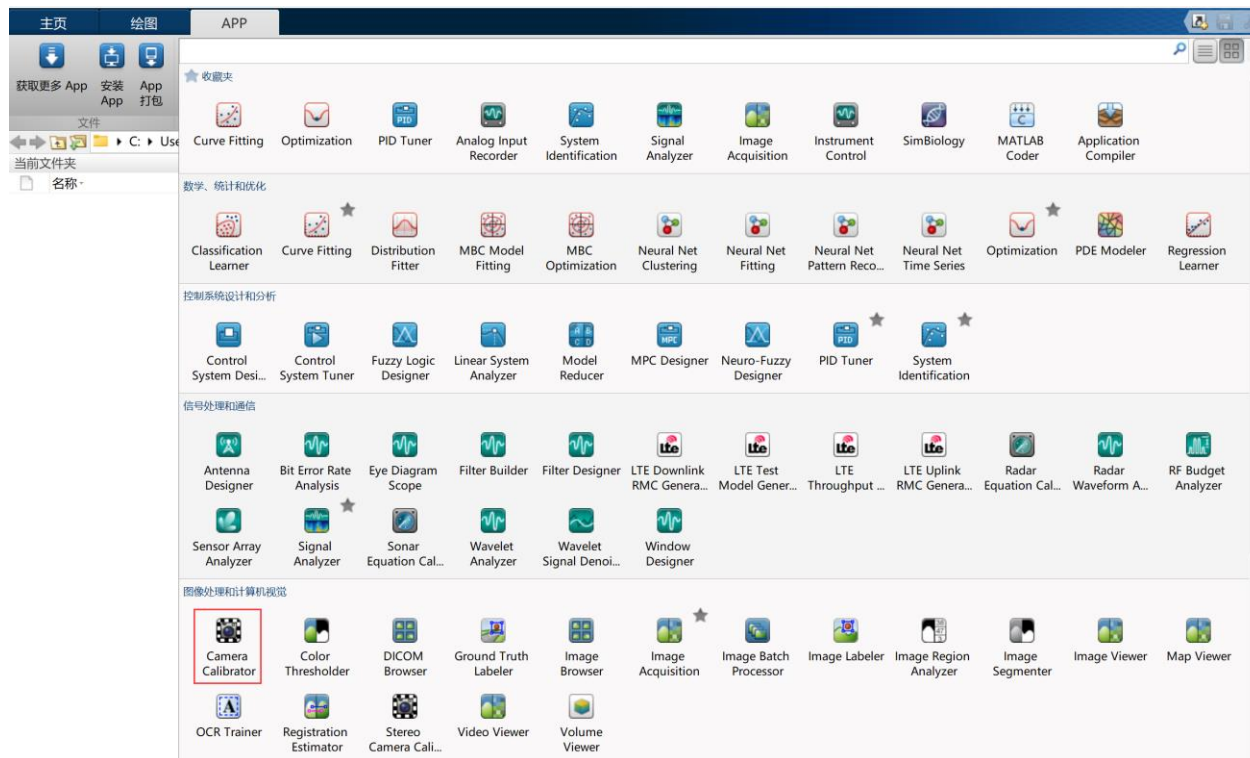




4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- Task 1: Calibration of Matlab calibration board
- 5. Pull down the toolbar and select the Camera Calibrator toolbox.

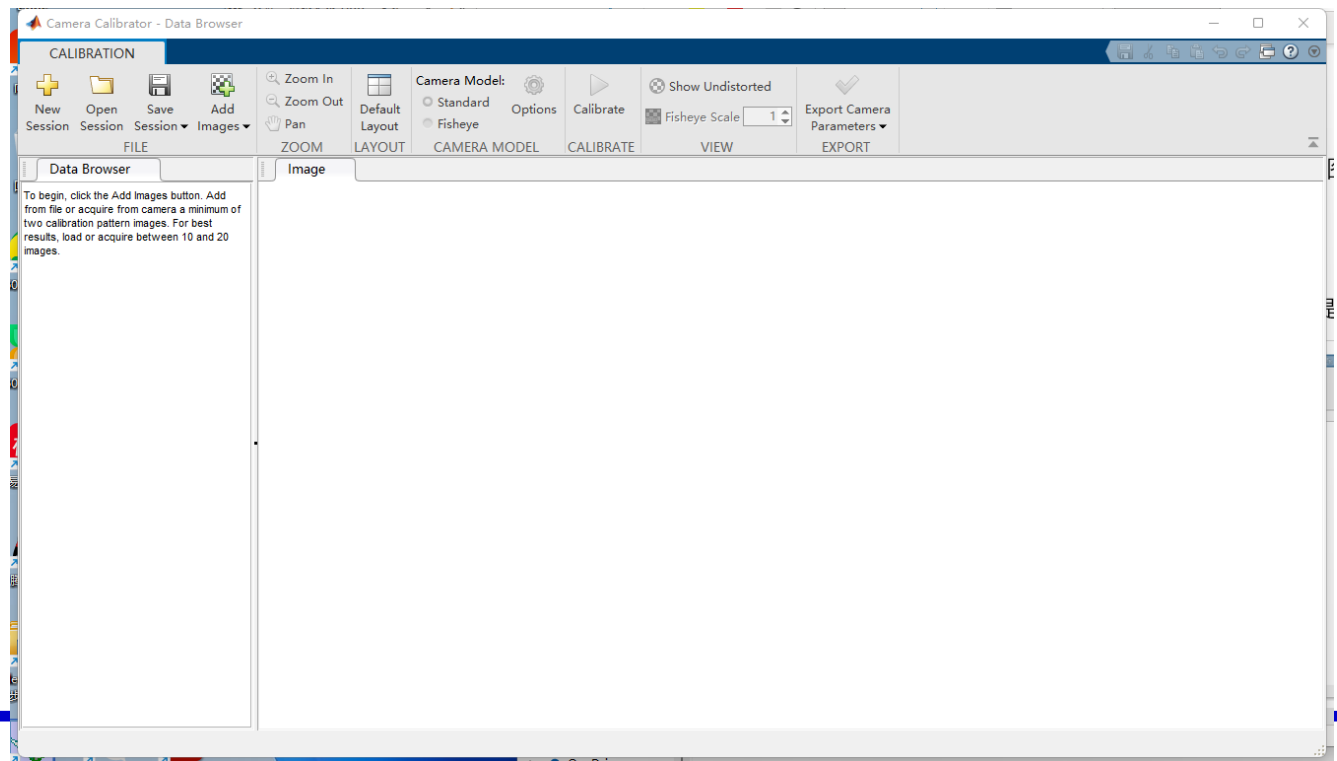




4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- **Task 1: Calibration of Matlab calibration board**
- **6. After opening the toolbox, click Add Images, open to the location where we grabbed the picture, and select the picture. Click the Open button when the selection is complete.**

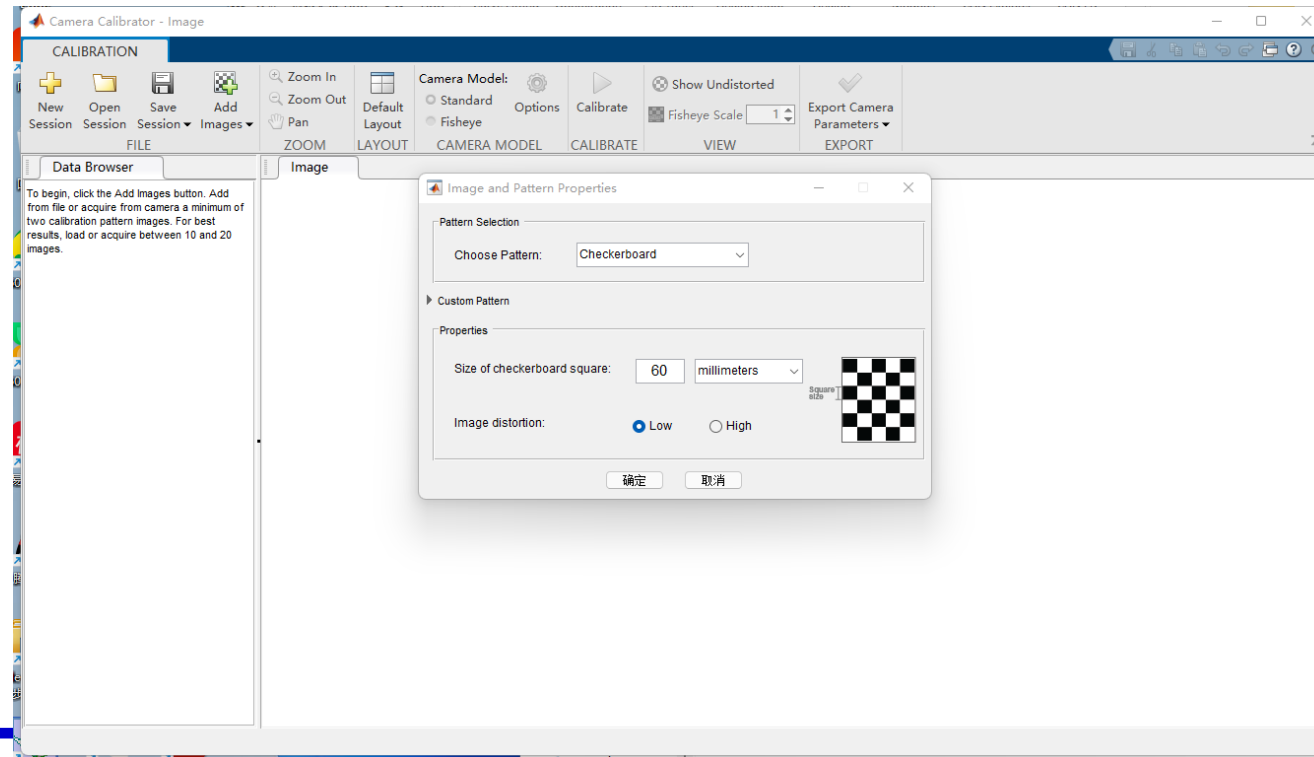




4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- **Task 1: Calibration of Matlab calibration board**
- **7. In the pop-up window below, enter the size of the checkerboard. 60mm is provided in the scene. Click OK.**





4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- **Task 1: Calibration of Matlab calibration board**
- **8. After a few seconds, the toolbox will tell you the accepted picture. Click OK directly.**

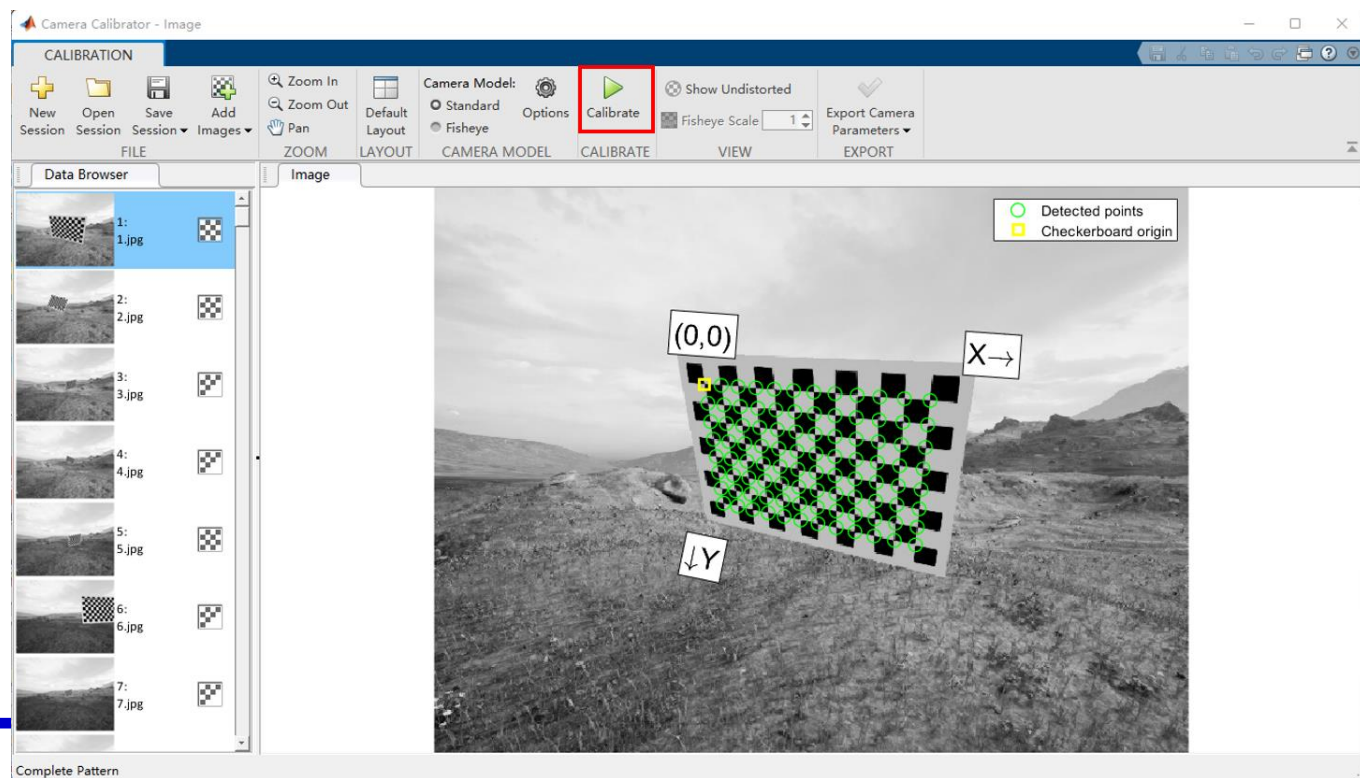




4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- Task 1: Calibration of Matlab calibration board
- 9. You can roughly browse the checkerboard extraction quality, which should be all on the image. Click the Calibrate button to start the calibration.

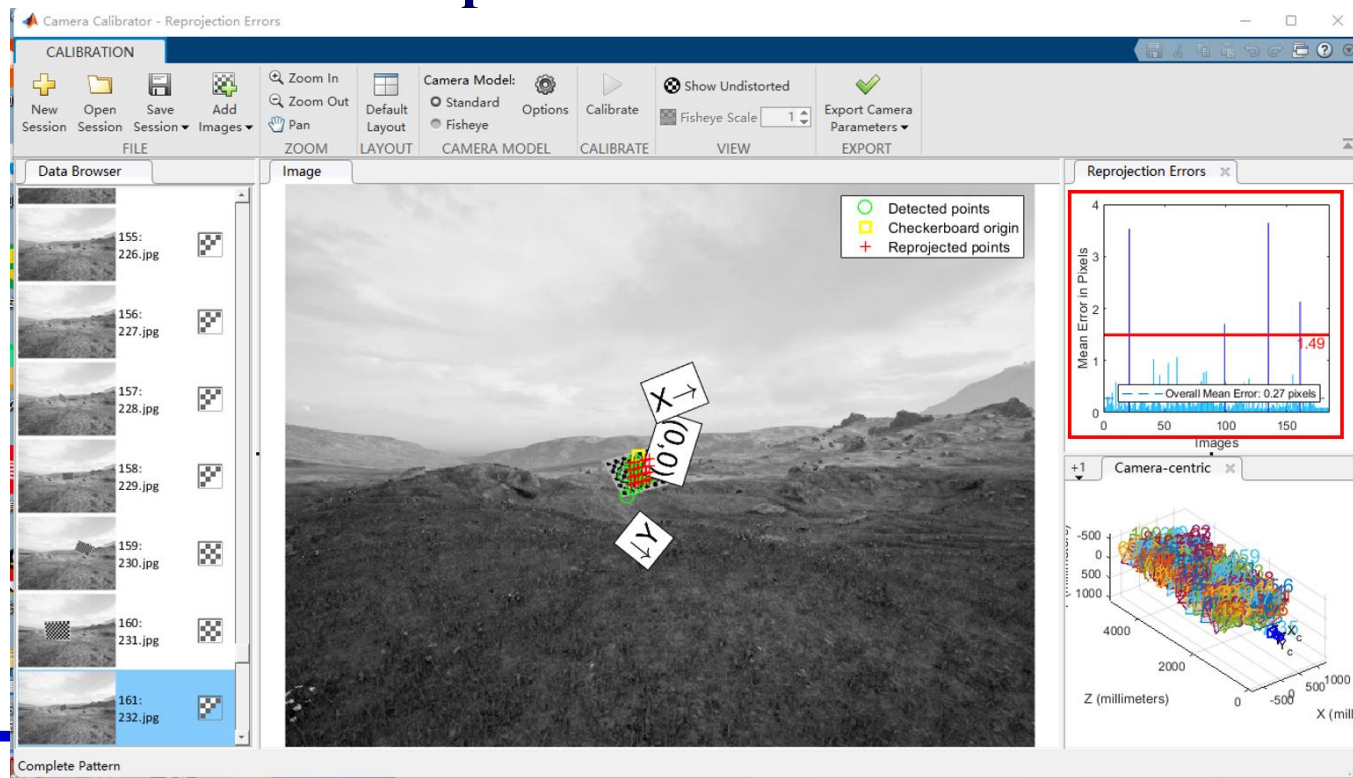




4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- Task 1: Calibration of Matlab calibration board
- 10. You can drag the red line to select the pictures with large backprojection error, and press Delete to delete these pictures and recalibrate them.

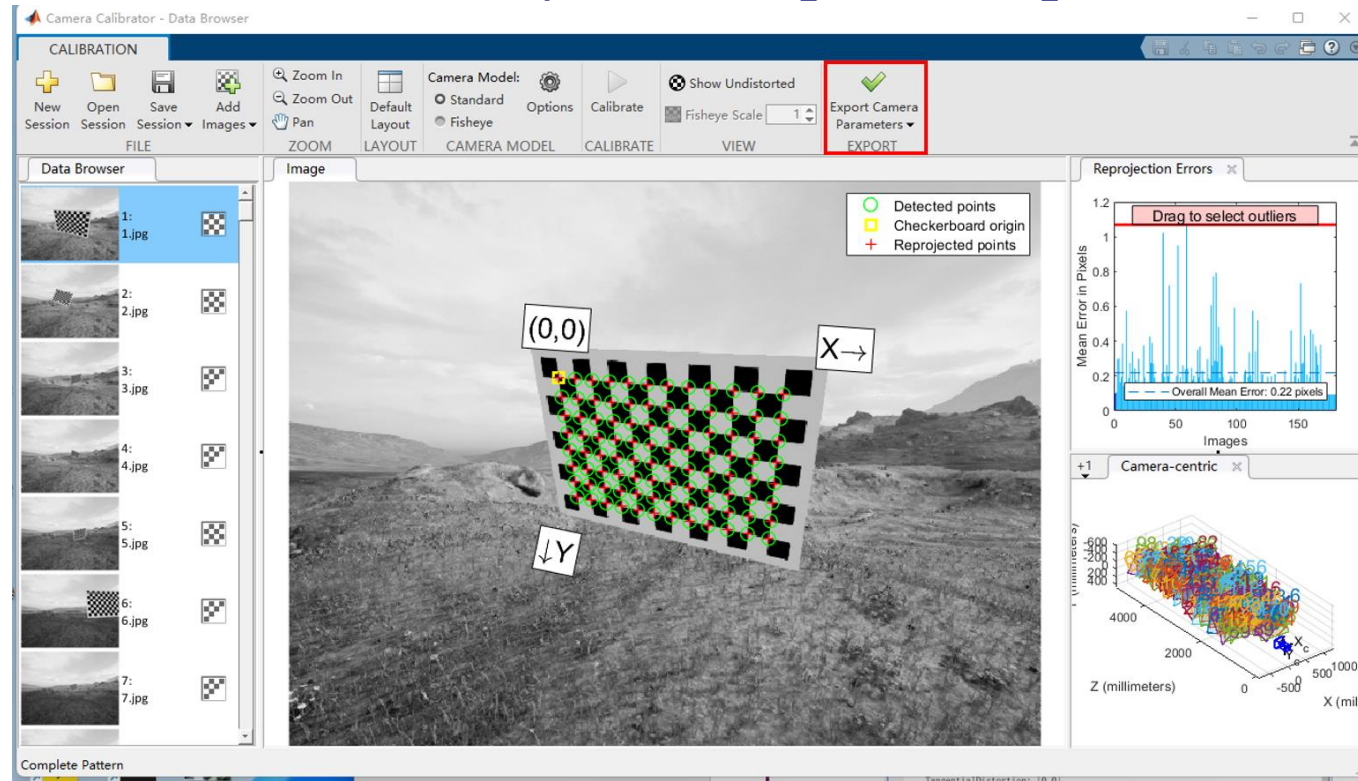




4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- Task 1: Calibration of Matlab calibration board
- 11. After the results are satisfactory, click Export to export the calibration results.





4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- **Task 1: Calibration of Matlab calibration board**
- **12. Back in the Matlab workspace, you'll see a structure called cameraParams, and you can double-click to see the member variables. You can also enter the cameraParams.Intrinsics below to view the camera information and enter the cameraParams.IntrinsicM atrix to view the internal reference matrix.**

The screenshot shows the MATLAB R2021b workspace with the following content:

```
cameraParameters - 属性:
    Camera Intrinsics
        Intrinsics: [1x1 cameraIntrinsics]
    Camera Extrinsic
        RotationMatrices: [3x3x180 double]
        TranslationVectors: [180x3 double]
    Accuracy of Estimation
        MeanReprojectionError: 0.2186
        ReprojectionErrors: [96x2x180 double]
        ProjectedPoints: [96x2x180 double]
    Calibration Settings
        NumPatterns: 180
        DetectedKeypoints: [96x180 logical]
        WorldPoints: [96x2 double]
        WorldUnits: 'millimeters'
        EstimateSkew: 0
        NumRadialDistortionCoefficients: 2
        EstimateTangentialDistortion: 0

estimationErrors =

cameraCalibrationErrors - 属性:
    IntrinsicErrors: [1x1 intrinsicEstimationErrors]
    ExtrinsicErrors: [1x1 extrinsicEstimationErrors]

>> cameraParams.Intrinsics

ans =
```



4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- Task 2: Python Calibration
- 1. Start the Camera CalcDemo. Bat, run the Camera CalcDemo. Py for some time, and a folder with captured pictures will be obtained under the working path. For example, "[8.RflySimVision\0.ApiExps\3-VisionAI\2.CameraCalcDemo\20220207_220418](#)".
- 2. In the bord-Calibration. Py file, change the path in line 19 to the photo path to be calibrated and run it
`images = glob.glob(“.\20220207_220418\img1*.jpg”)`.
- 3. The resulting internal reference matrix is as follows.

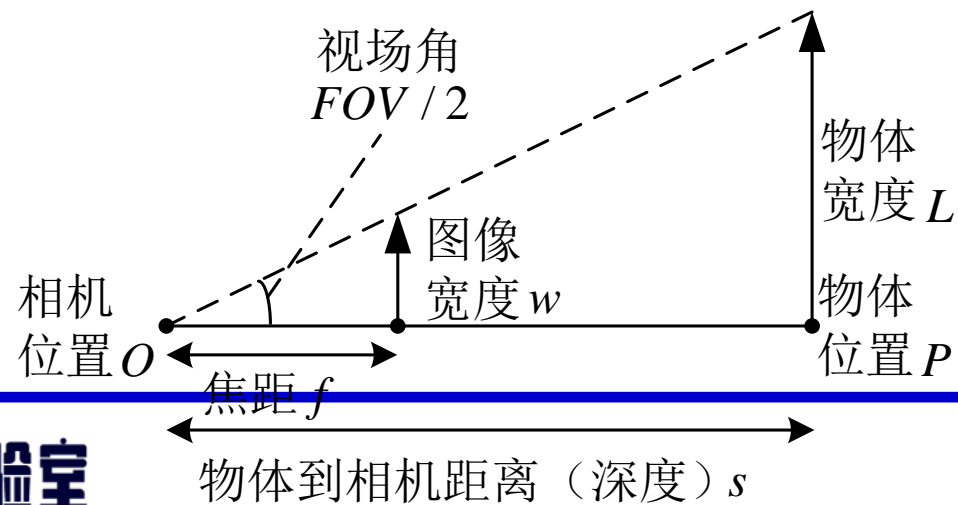
```
[[322.36795044    0.    318.71025401]
 [  0.    319.37265015  235.49389293]
 [  0.    0.    1.    ]]
```



4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- Task 3: Calibration Process of Python Double Small Ball
- 1. Principle: Two small balls are generated, and their distance is fixed at one meter under the world coordinates. The proportional relationship between the world coordinate system and the image coordinate system is judged according to the distance between the two small balls on the image. The translation of the coordinate origin is then measured by the distance from the midpoint of the ball line to the camera. The focal length can be obtain from that distance L between the two small balls, the width w of the small ball on the image and the distance s from the middle point of the line connecting the small balls to the camera. Formula: $f = \frac{w \cdot s}{L}$

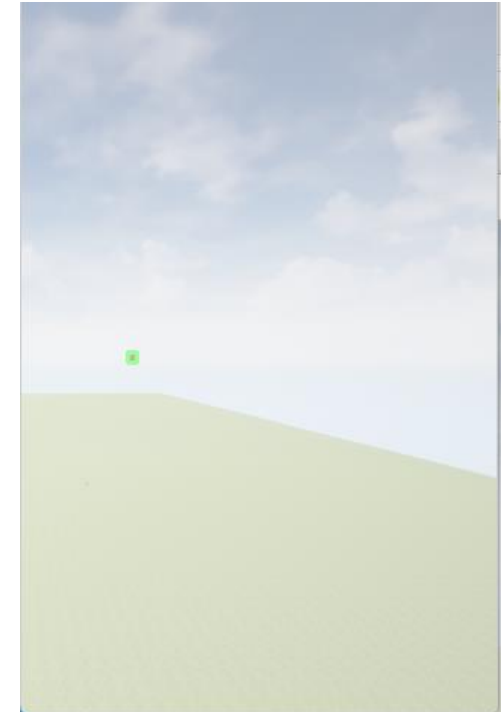
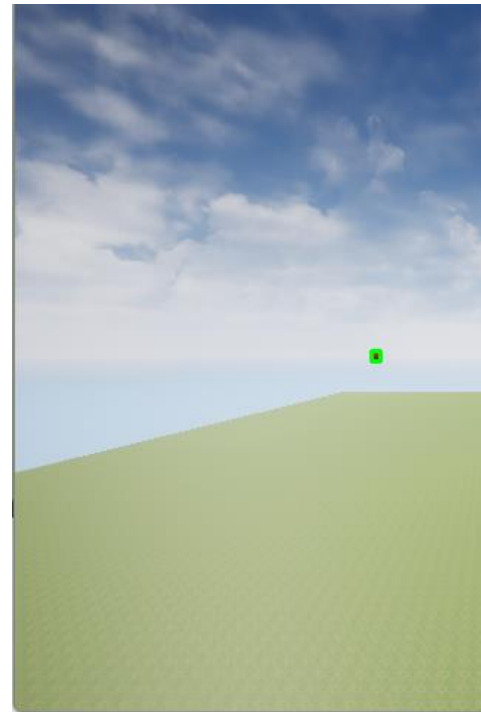
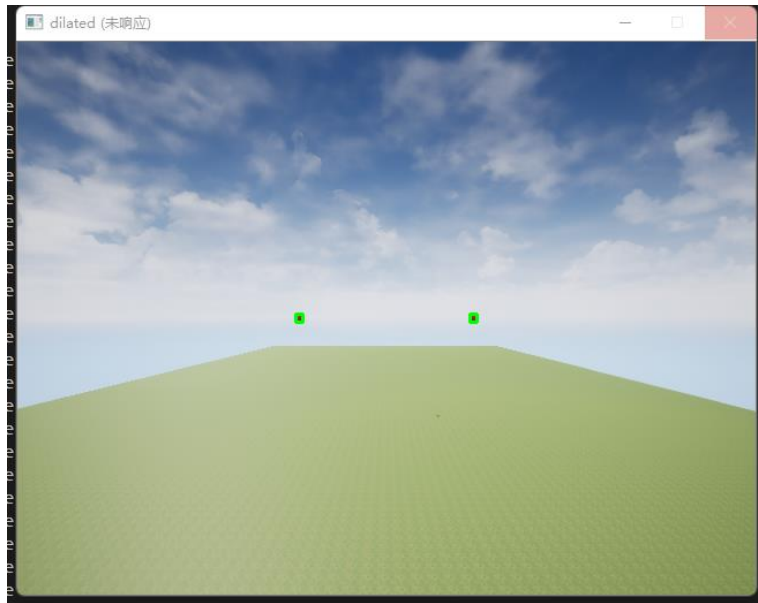




4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- **Task 3: Calibration Process of Python Double Small Ball**
- **2. Run ball2-Calibration. Py in VS Code, and the result is as shown in the following figure.**





4. Visual AI is advanced

4.2 Virtual Camera Calibration Experiment-Routine Introduction

- Task 3: Calibration Process of Python Double Small Ball

```
ball2-Calibration.py X
C:\Users\RFLYSIM\Desktop> biaoding\OneCameraCal-gather > ball2-Calibration.py {} random
1 # import required libraries
2 import time
3 import sys
4 import cv2 as cv
5 import random
6 import numpy as np
7

问题 输出 调试控制台 终端
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 3.5750
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 324.7000 ; 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 323.0000 ; The distance is 2.4250
The mean focal length of the camera is: 323.3808 100
```



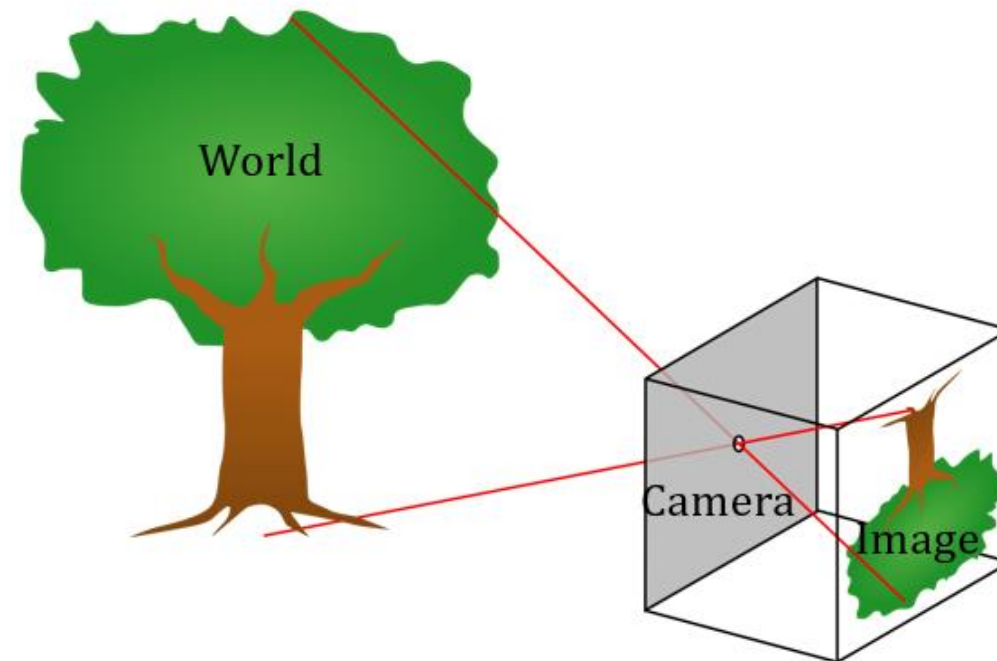


4. Visual AI is advanced

Routines see: RflySimAPIs \ Python
VisionAPI \ 3-VisionAIDemos \ 3-
GetRelativePosDemo

4.3 Camera coordinate conversion — introduction to the principle

- The camera is based on the pinhole principle for imaging, as shown in the right figure, which is a schematic diagram of the pinhole principle. The pinhole model corresponds to the imaging process, the object in the real world is the imaging target in the three-dimensional space, the pinhole is the center of the camera, and the reflection imaging plane is a two-dimensional image plane.
- One of the characteristics of pinhole imaging is that any point in the real world, its projection point on the imaging plane and the center of the camera are in a straight line, which is called central projection or perspective projection, and is also the basis of imaging analysis. Perspective projection is a reduced-rank spatial transmission transformation, which projects a three-dimensional space onto a two-dimensional plane.





4. Visual AI is advanced

4.3 Camera coordinate conversion — introduction to the principle

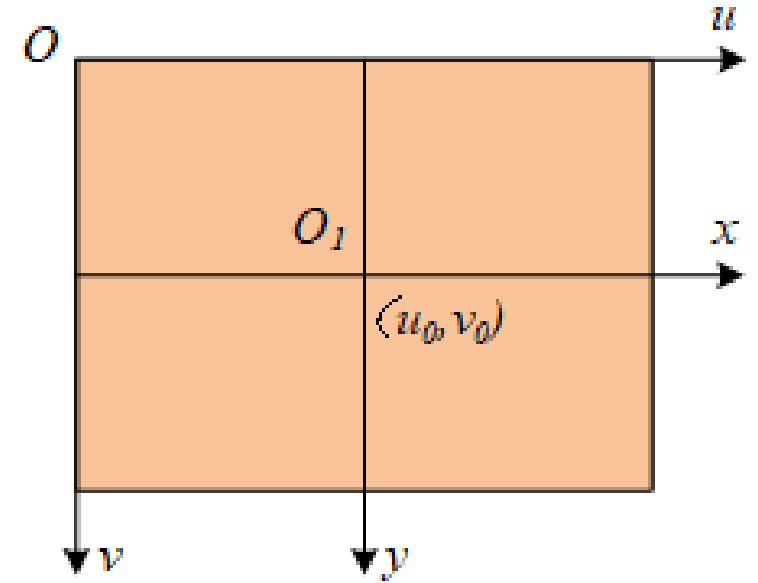
- In order to make complex 3D measurements with pictures, we must rely on a series of complex calculations, which are based on the Cartesian coordinate system. Generally, the coordinate systems related to the visual image of the UAV are as follows:
 - Image pixel coordinate system;
 - Image physical coordinate system;
 - Camera coordinate system;
 - Body coordinate system;
 - World coordinate system;



4. Visual AI is advanced

4.3 Camera coordinate conversion — introduction to the principle

- **Image pixel coordinate system:** It is a two-dimensional rectangular coordinate system, which reflects the arrangement of pixels in the camera. Its origin O is located in the upper left corner of the image, and the u and V coordinate axes coincide with the two sides of the image respectively. Pixel coordinates are discrete values $(0, 1, 2, \dots)$, in pixels.
- **Image physical coordinate system:** In order to relate the image to the physical space, the image needs to be transformed into the physical coordinate system. The origin O is located at the center of the image (ideally) and is the intersection of the camera's optical axis and the image plane (called the principal point). The X and y axes are parallel to the u and V axes, respectively. The two coordinate systems are actually translation relations, and the translation quantity is (u_0, v_0) .
- The above two can be collectively referred to as the image coordinate system, and both coordinate systems are two-dimensional coordinate systems.





4. Visual AI is advanced

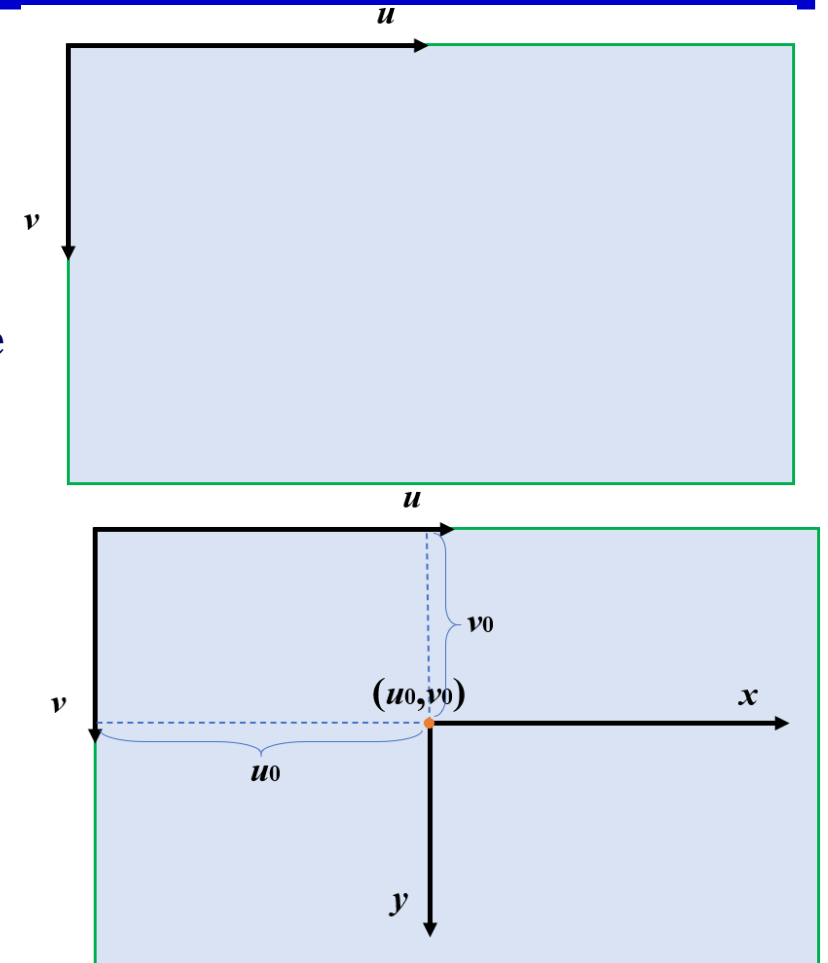
4.3 Camera coordinate conversion — introduction to the principle

- The right figure shows two different expressions of the image coordinate system. If we know the conversion relationship from pixel to physical size, that is, the physical size of a pixel, that is, the pixel size is $DX \times DY$ (the size in the X direction is DX , and the size in the y direction is dy), we can convert between the two coordinate systems:

$$u - u_0 = x/d_x$$

$$v - v_0 = y/d_y$$

- In order to facilitate matrix operations, it can be written in matrix form. The three-dimensional vector on both sides of the formula is a homogeneous expression, that is, the third dimension is set to 1 to represent the two-dimensional vector with the three-dimensional vector. The advantage of this is that the transformation from three-dimensional to two-dimensional can be completed by matrix operation.



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



4. Visual AI is advanced

4.3 Camera coordinate conversion — introduction to the principle

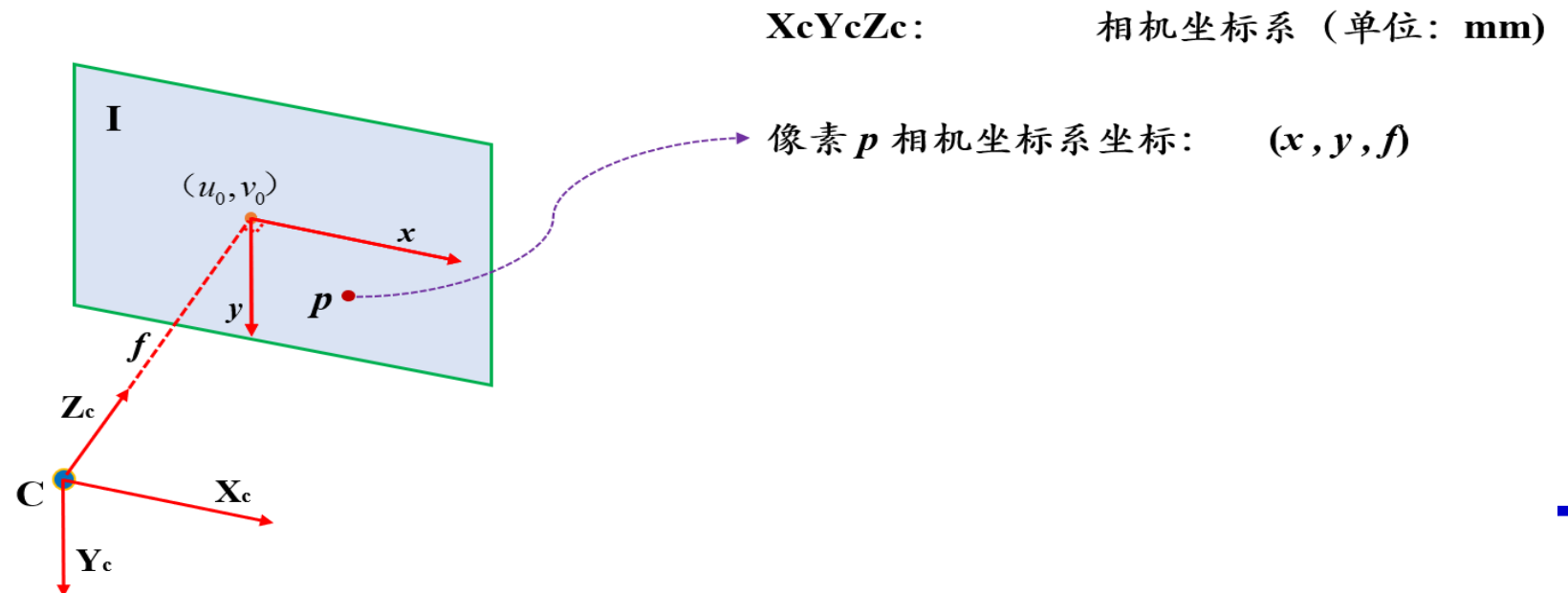
- The camera coordinate system, the body coordinate system and the world coordinate system are all three-dimensional coordinate systems, and in the quad-rotor unmanned aerial vehicle, the relative position between the camera coordinate system and the body coordinate system is fixed, so that after the relative position is measured at the beginning, the camera coordinate systems and the body coordinate systems can be considered as a whole to be discussed together, Both of them can be regarded as a coordinate system in which the direction of the origin and coordinate axis is bound to the initial state of the UAV, and it is a coordinate system that will move relative to the ground;
- The world coordinate system is an absolute coordinate system with a fixed position on the ground as the origin. Its function is to unify all points in space into the same coordinate system.
- The conversion between 3D coordinate systems is more convenient to be expressed by rotation and translation, and the key and difficult point is the conversion between 3D coordinate systems such as camera coordinate system and 2D coordinate systems such as image coordinate system.



4. Visual AI is advanced

4.3 Camera coordinate conversion — introduction to the principle

- The origin of the camera coordinate system is at the center of the camera. The XY axis of the camera coordinate system is parallel to the XY axis of the image coordinate system. The Z axis is perpendicular to the image plane and faces the image plane. The intersection point of the Z axis and the image plane is the origin of the image XY coordinate system (image principal point), as shown in the following figure:



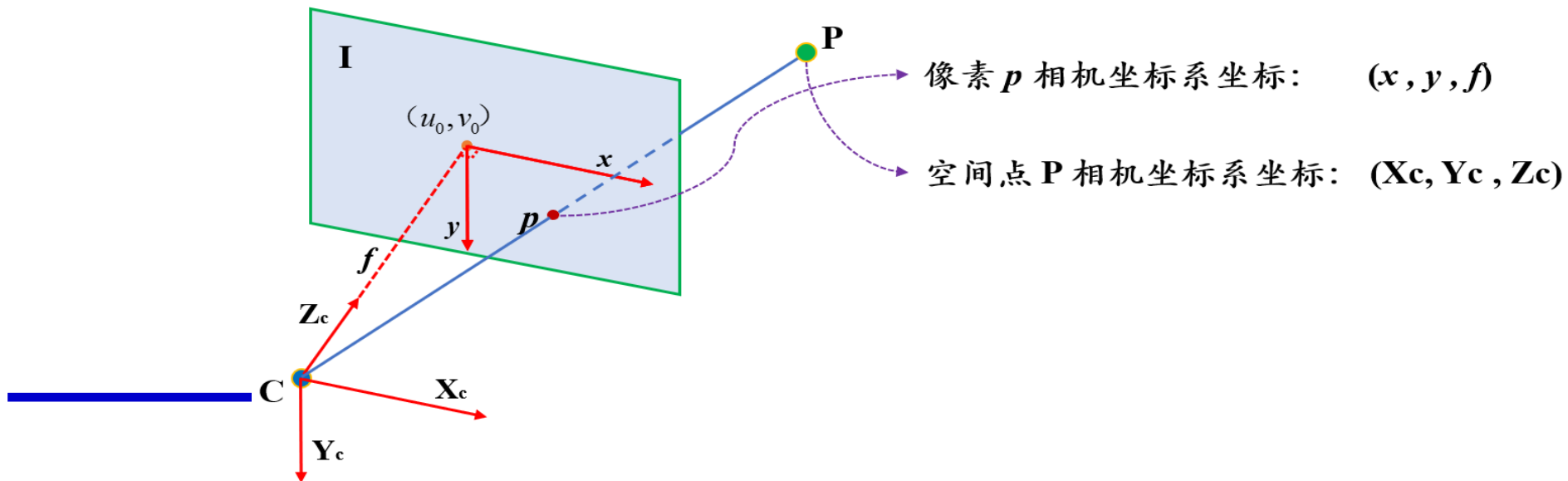


4. Visual AI is advanced

4.3 Camera coordinate conversion — introduction to the principle

- In this scheme, the Z coordinates of all pixel points on the image plane in the camera coordinate system are equal to the focal length f , and the values in the camera XY coordinate system and the image XY coordinate system are equal, that is, if the coordinates of the pixel point p in the image XY coordinate system is (X, y) , then its coordinates in the camera XY coordinate system is (X, y, f) .

$X_c Y_c Z_c$: 相机坐标系 (单位: mm)

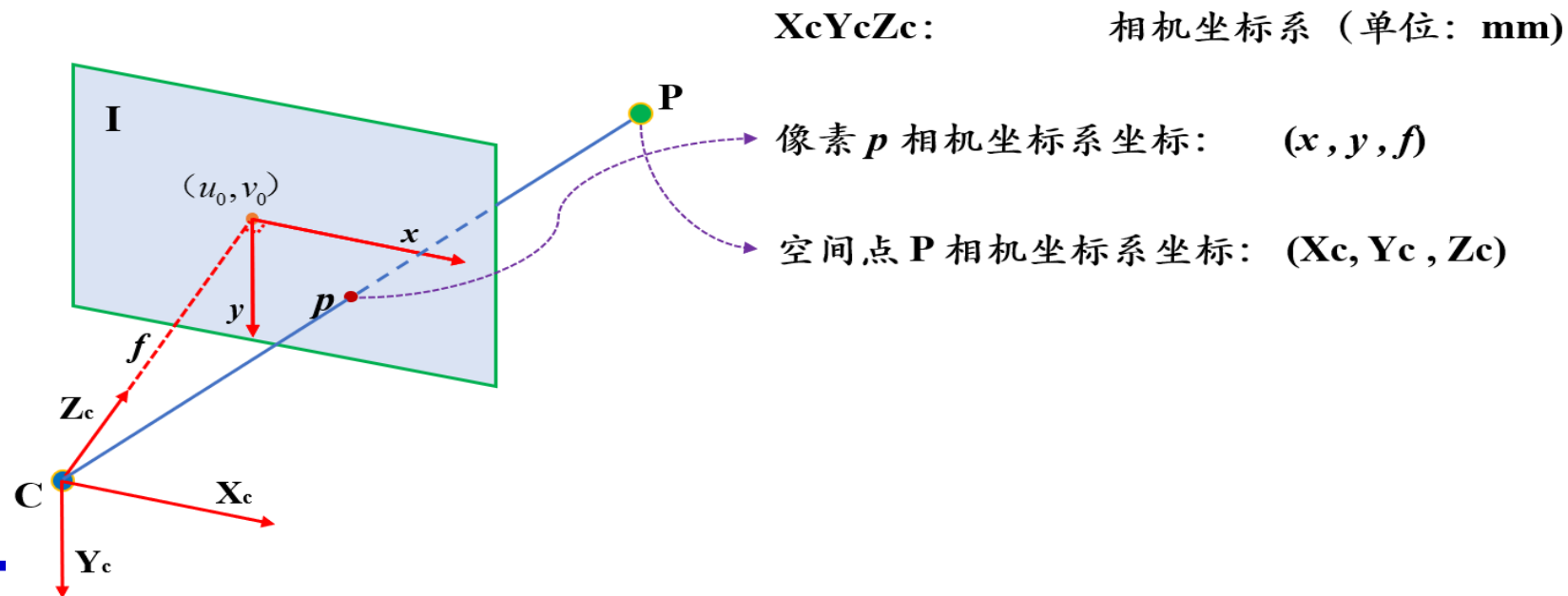




4. Visual AI is advanced

4.3 Camera coordinate conversion — introduction to the principle

- Assume that the coordinates of the space point corresponding to the pixel p in the camera coordinate system are (X_c, Y_c, Z_c) . If two points lie on the $\frac{x}{X_c} = \frac{y}{Y_c} = \frac{f}{Z_c}$ line from the origin of the coordinate system, their coordinates are in a proportional relationship. That is





4. Visual AI is advanced

4.3 Camera coordinate conversion — introduction to the principle

- Assume that the coordinates of the space point corresponding to the pixel p in the camera coordinate system are (X_c, Y_c, Z_c) . If two points lie on the same straight line from the origin of the coordinate system, their coordinates are in a proportional relationship. That is $\frac{x}{X_c} = \frac{y}{Y_c} = \frac{f}{Z_c}$
- In order to facilitate matrix operations, it is written in matrix form:

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

- You can also convert XY coordinates to UV coordinates:

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

- Z_c is usually called the scale factor λ , and the 3x3 matrix in the middle is called the internal reference matrix K . Obviously, the internal reference matrix K describes the conversion relationship from the camera coordinate system to the UV coordinate system.
-



4. Visual AI is advanced

4.3 Camera coordinate conversion — introduction to the principle

- The internal reference matrix K is one of the key parameters of the camera, $\frac{f}{dx}$ and $\frac{f}{dy}$ is actually used to convert the focal length f in the unit of physical size into the focal length value in the unit of pixel, where $f_x = \frac{f}{dx}$, $f_y = \frac{f}{dy}$, f_x and f_y are respectively the pixel unit values of the focal lengths in the two pixel directions. Finally, the matrix expression of the internal reference is obtained:

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



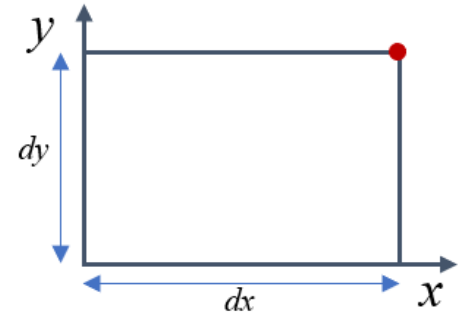
4. Visual AI is advanced

4.3 Camera coordinate conversion — introduction to the principle

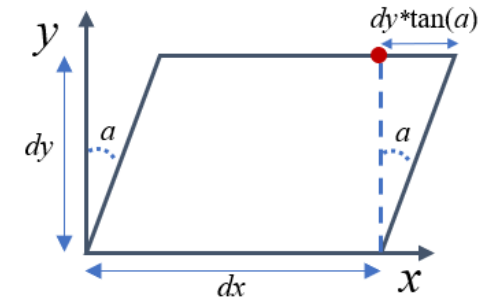
- Due to the deviation of the manufacturing process, the pixel in reality is not an absolute rectangle, but a parallelogram, as shown in the right figure. At this time, the vertical boundary of the pixel is not parallel to the y-axis but is tilted at a certain angle, so a tilt factor s is introduced into the K matrix, and the K matrix is expressed as

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

rectangular pixel



non-rectangular pixel





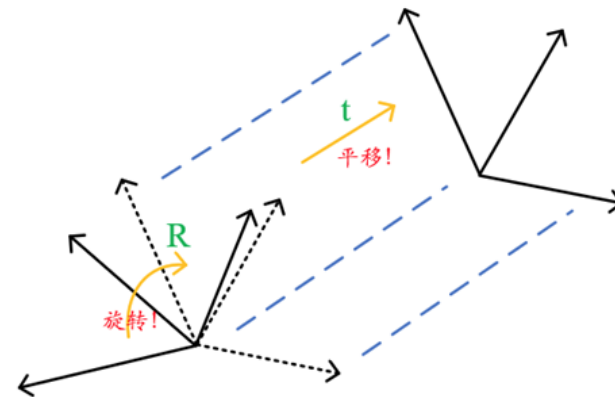
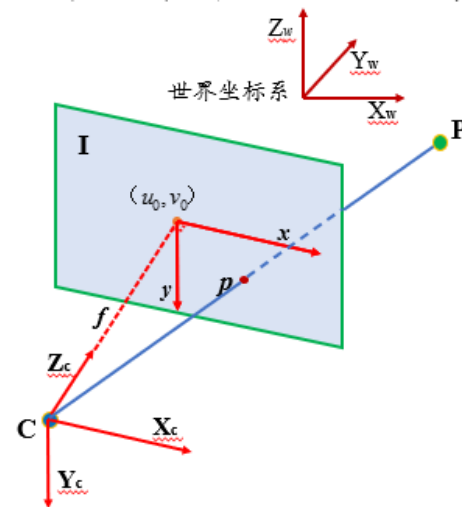
4. Visual AI is advanced

4.3 Camera coordinate conversion — introduction to the principle

- For a world coordinate system, a fixed absolute coordinate system.
- The world coordinate system and the camera coordinate system are both three-dimensional coordinate systems, and they can be transformed by rotation and translation.
- Assuming that the coordinates of the space point P in the world coordinate system are (X_w, Y_w, Z_w) , it can be converted into the camera coordinate system coordinates (X_c, Y_c, Z_c) through a 3x3 unit orthogonal rotation matrix R and a 3x1 translation vector t.

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

- The rotation matrix R and the translation vector t are called the extrinsic parameter matrix of the camera.





4. Visual AI is advanced

4.3 Camera Coordinate Conversion-Routine Introduction

- Open the “ [8.RflySimVision\0.ApiExps\3-VisionAI\4.GetRelativePosDemo](#) ” folder.
- Double-click to start the Get RelativePosDemo. Bat file.
- Open the GetRelative PosDemo. Py script, set breakpoints for key statements, and execute the statements one by one in debug mode.
- This script shows several ways to set and obtain coordinate position information in the software, both in the ground coordinate system and in the UAV coordinate system.

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

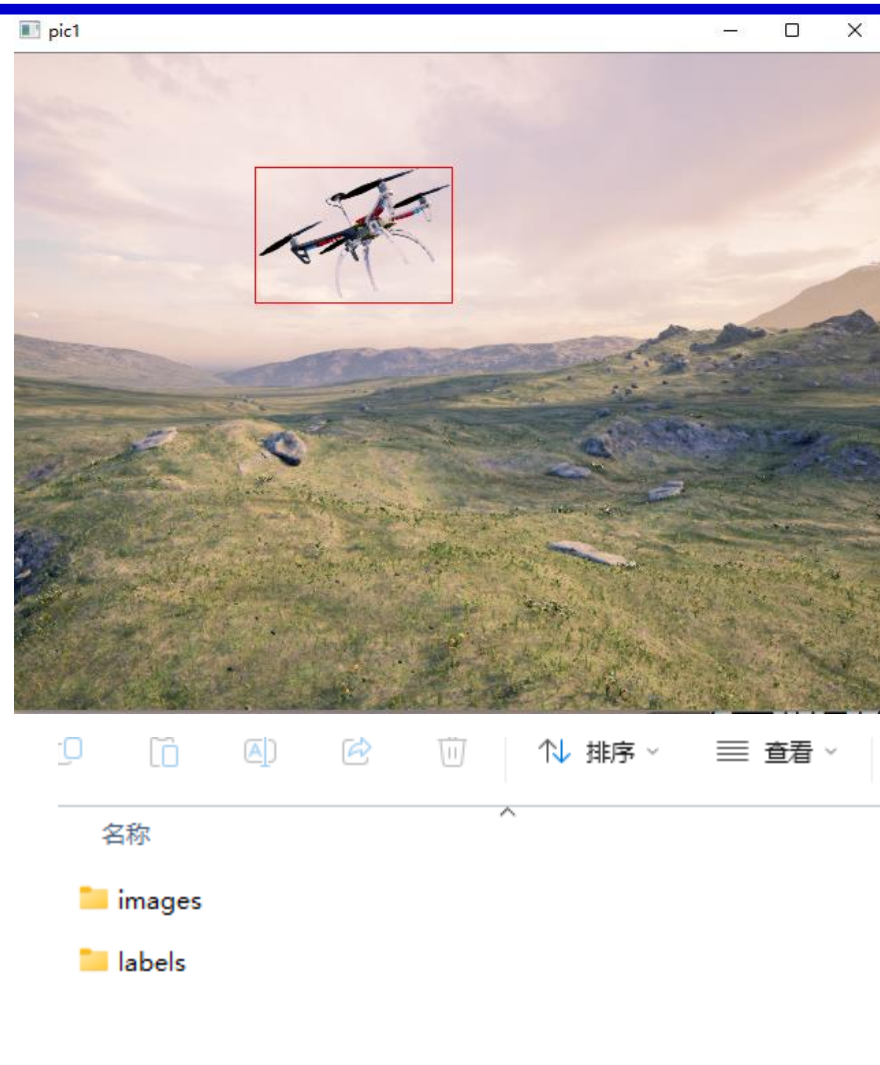


4. Visual AI is advanced

4.4 Methods for Generating Data Sets — Introduction to Routines

- Open the “ [8.RflySimVision\0.ApiExps\3-VisionAI\5.GenVisionDataSet](#) ” folder.
- Double-click the launch One CameraCal. Bat file to launch RflySim3D.
- Run the get _ dateset. Py, and the aircraft will appear in a random position on the screen and be marked with a red box. A folder named by timestamp is added to the directory, in which the images directory stores the collected aircraft images, and the labels directory stores the labels (YOLO format) corresponding to each image.

20231226_144402





4. Visual AI is advanced

4.4 Methods for Generating Data Sets — Introduction to Routines

- **Modify the ROOT path in the maketxt. Py to the generated folder path. Be careful to change "\" to "/" to prevent escaping.**

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

- **Running the program will generate a data directory and an ImageSets directory in the folder. The data directory is the data set directory read during training. By default, the training set and the test set are randomly divided according to the ratio of 9:1.**

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. Visual AI is advanced

4.4 Methods for Generating Data Sets — Introduction to Routines

- At present, the model of No.3 quadrotor UAV is used. If you want to change it to another model, you need to know the coordinates of all convex points of the model, and change the coordinates corresponding to each point of the function shown in the following figure. Note that the following parameters include the wing.

```
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. Visual AI is advanced

4.4 Methods for Generating Data Sets — Introduction to Routines

- When increasing or decreasing the number of raise points of the function in the figure above, you also need to change the number of columns in the position array shown in the figure below.

```
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
```

- When changing the automatically generated model, you also need to change the number of the model in the data set, that is, the "0" below here. Here is the name corresponding to the number 0 in the class during training. If you want to change it to another model, please change the number, and change the corresponding number position in the corresponding class to the required name during training.

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

- Finally, if the data sets of different models need to be generated and put together to form a data set, it is also necessary to change the CNT starting parameter as shown in the figure, which corresponds to the starting number of the image and txt document name when generating the data set.

```
200 cnt = 0
```



4. Visual AI is advanced

4.5 Visual Object Recognition Experiment-Environment Configuration

- For this lab, you need to first install PyTorch in the python environment, which is divided into a CPU version and a GPU version. If the computer's graphics card is an AMD graphics card, only the CPU version can be installed; if it is an NVIDIA graphics card, it is recommended to install the GPU version, which is faster.
- To install the GPU version, you need to install the CUDA Toolkit and cuDNN. You can find the installation method of the corresponding version of the graphics card in the link <https://developer.nvidia.com/cuda-toolkit-archive> and <https://developer.nvidia.com/rdp/cudnn-archive> respectively.
- After the installation is completed, start to install PyTorch, enter the link <https://pytorch.org/> and select the corresponding installation command. If the CPU version is installed, select CPU in Compute Platform.

The screenshot shows the PyTorch installation configuration interface. The 'Compute Platform' dropdown is highlighted with a red box. Below it, a message states: 'CUDA-10.2 PyTorch builds are no longer available for Windows, please use CUDA-11.6'.

PyTorch Build	Stable (1.12.0)	Preview (Nightly)	LTS (1.8.2)		
Your OS	Linux	Mac	Windows		
Package	Conda	Pip	LibTorch	Source	
Language	Python		C++ / Java		
Compute Platform	CUDA 10.2	CUDA 11.3	CUDA 11.6	ROCm 5.1+	CPU

Run this Command:
CUDA-10.2 PyTorch builds are no longer available for Windows, please use CUDA-11.6



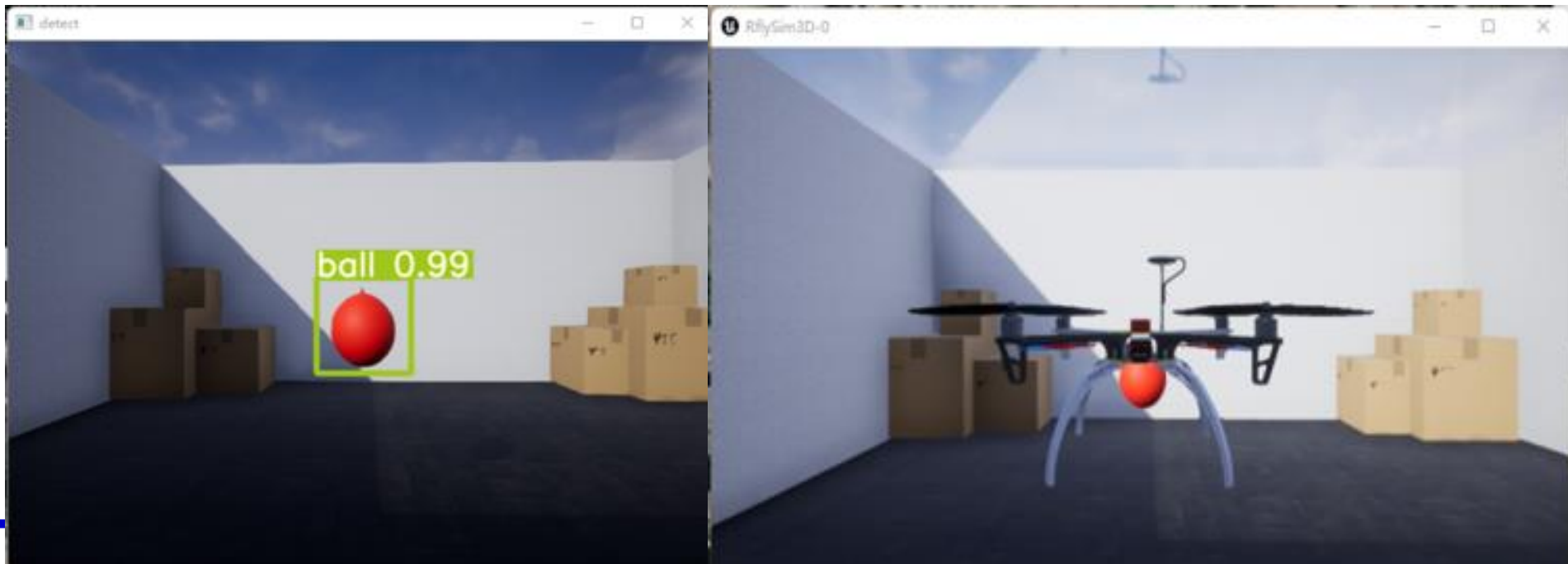
4. Visual AI is advanced

4.5 Visual Object Recognition Experiment-Routine Introduction

This routine is located in the "

[8.RflySimVision\2.AdvExps\c7_ObjDetectYolo\ShootBallBaseOnYolo](#)".

- Double-click the Launch ShootBall3SITL.bat file to launch RflySim3D.
- Run the ShootBall 3.py program, and you can see the effect of the plane hitting the ball.



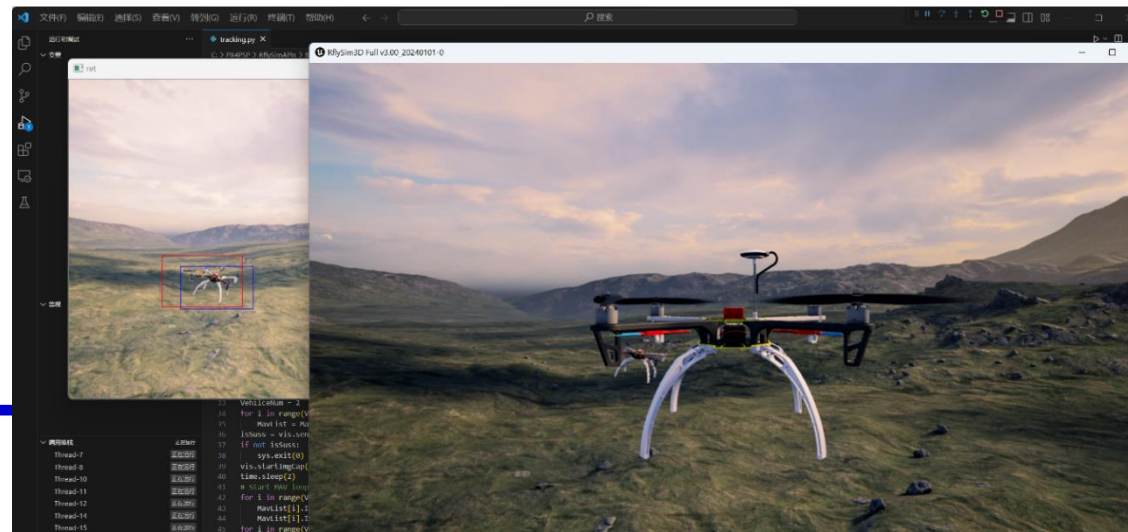


4. Visual AI is advanced

4.6 Visual AI Training Experiment-Routine Introduction

Target tracking experiment

- Open the " [8.RflySimVision\2.AdvExps\e8_SingleObjTracking](#) "folder.
- Right click the Tracking. Bat and select Run as Administrator to start RflySim3D.
- Run the tracking. Py to see that another aircraft is generated on the right side of the aircraft. After the two aircraft take off, the aircraft on the right side will fly to the front and move. At the same time, a window is generated to display the coincidence effect of the target aircraft identification detection frame and the tracking detection frame.



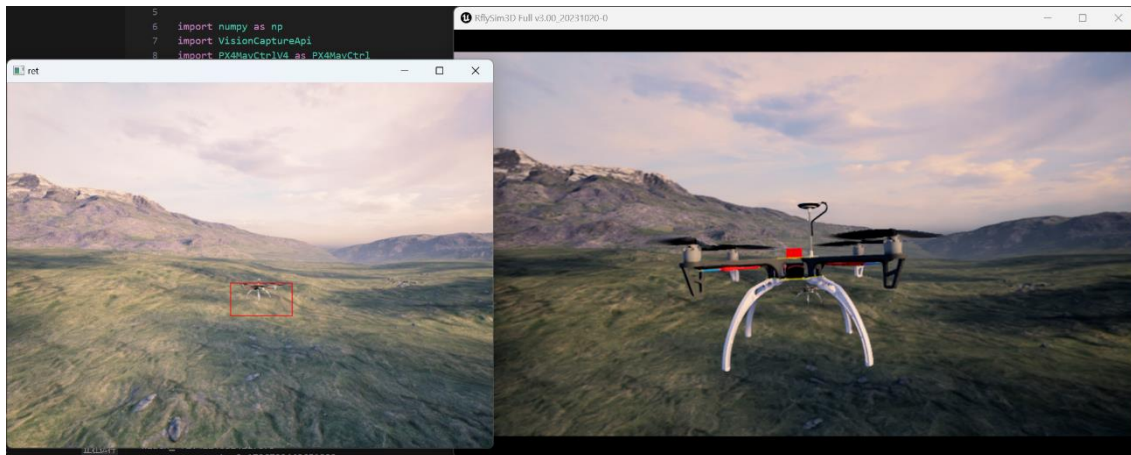


4. Visual AI is advanced

4.6 Visual AI Training Experiment-Routine Introduction

Target following experiment

- This routine is located in the "[8RflySimVision\2.AdvExps\e9_Object-Follow](#)" folder.
- Run the target _ follow. Bat as an administrator.
- Run the target _ follow. Py, it can be seen that two planes take off, the right plane flies to the front, at the same time, a window is generated to display the front plane, it can be seen that the target plane identification detection frame, the target plane starts to move, and the main plane follows the target plane to move.



```
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
```



Outline

1. General introduction
2. The base interface uses
3. Visual control example
4. Visual AI is advanced
5. Distributed visual simulation



5. Distributed visual simulation

5.1 General introduction to routines

- In the actual UAV top-level development and testing process, there are often the following two requirements.
- 1) On-board computers (Raspberry Pi, NVIDIA Jetson Xavier NX, NANO, etc.) Running Linux/ROS environment for visual perception algorithm (recognition, SLAM, obstacle avoidance, etc.) Development.
- 2) Multiple UAV clusters perform distributed perception and control, that is, to simulate multiple UAV clusters, each UAV needs to have its own independent vision to complete the exploration and cooperation tasks of a specific scene.
- This section builds a series of solutions for these two requirements and provides routines and documentation. As shown in the figure below, the "[8.RflySimVision\0.ApiExps\2-DistributedSimAPI\0.Preparation](#)" folder contains the remote connection debugging methods for Raspberry Pi, NX and other hardware, which is convenient for the experiment.

📁 RaspberryPI_connection method

📁 NX-TX2Method of enabling remote access...



5. Distributed visual simulation

5.1 General introduction to routines

- "[8.RflySimVision\0.ApiExps\4-AdvApiExps\9.VisionAPIsTest](#)" and "[8.RflySimVision\0.ApiExps\2-DistributedSimAPI\1.VisionAPIsTest](#)" introduces the routines of different image transmission interfaces; "[8.RflySimVision\3.CustExps\2-DistributedSimDemos](#)" directory, "E1_OneVehilceCtrls" introduces the routines of single unmanned vision; "E2 introduces the multi-UAV routine _ MultipleVehicles"; "E3 introduces the vision-controlled routine for an arbitrary number of UAVs _ AnyVehilces".

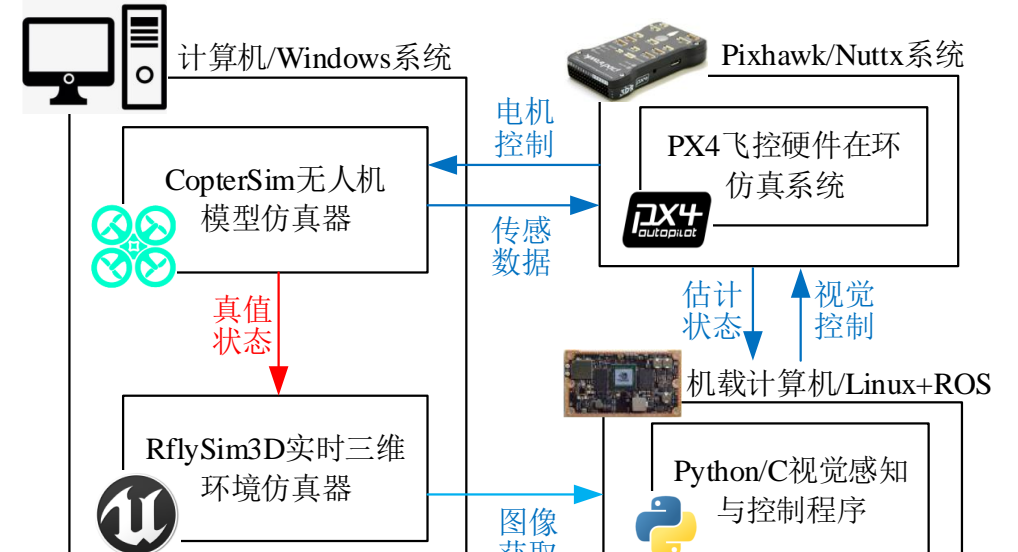
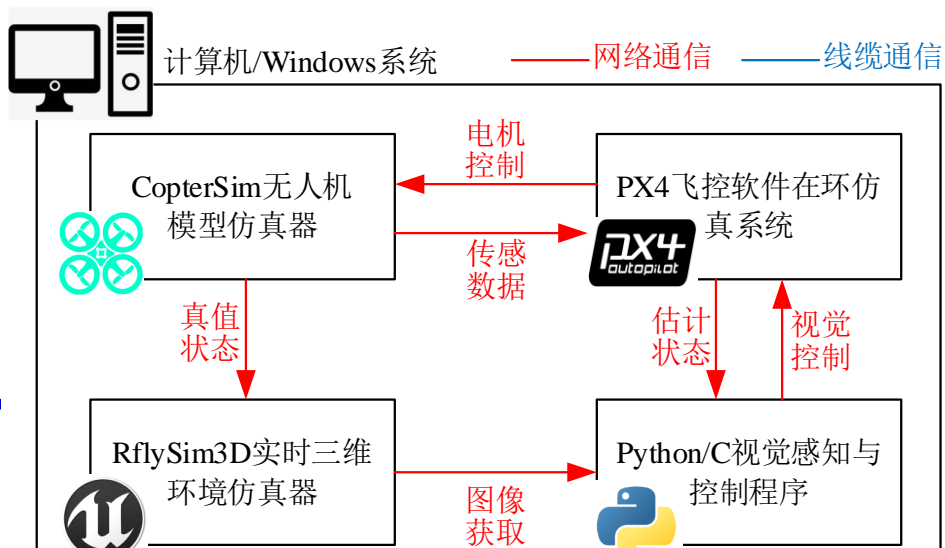
e1_OneVehilceCtrls	2024/1/26 18:40	文件夹	
e2_MultipleVehicles	2024/1/26 18:40	文件夹	
e3_AnyVehilces	2024/1/26 18:40	文件夹	
Readme.pdf	2024/1/22 18:33	Microsoft Edge PD...	234 KB



5. Distributed visual simulation

5.2 Access to the ROS system for simulation — from algorithm development to deployment

- For ease of development and ease of use, the RflySim platform runs only on the Windows platform, but supports the transfer of images to other computers (embedded hosts, virtual machines, desktops, etc.) via media such as network cables.
- Figure (a) below shows the implementation architecture of the previous routine. This architecture is suitable for the rapid development and verification of a single perception algorithm. After completion, it is transplanted to the on-board Linux/ROS system shown in Figure (B) for deployment and hardware-in-the-loop testing.
- As an intermediate transition, the Python-aware algorithm module in Figure (a) can be replaced by a Linux/ROS virtual machine environment, so that the deployment and preliminary test of the algorithm can be realized on a single computer to achieve the purpose of improving efficiency and saving costs.





5. Distributed visual simulation

5.2 Access to the ROS system for simulation — hardware-in-the-loop simulation to the real machine experiment

- After the airborne computer receives the image from the network (currently using UDP protocol), it can directly process the image, or forward it to other sensing modules through ROS nodes.
- Pixhawk is directly connected to the onboard computer through a serial port cable, which is the same as the hardware connection on the real machine.
- Directly insert the output of Pixhawk/PX4 flight control into the electric controller, and connect the image acquisition interface to the camera to complete the migration from the hardware-in-the-loop simulation shown in Figure (B) to the real machine system shown in Figure (C).

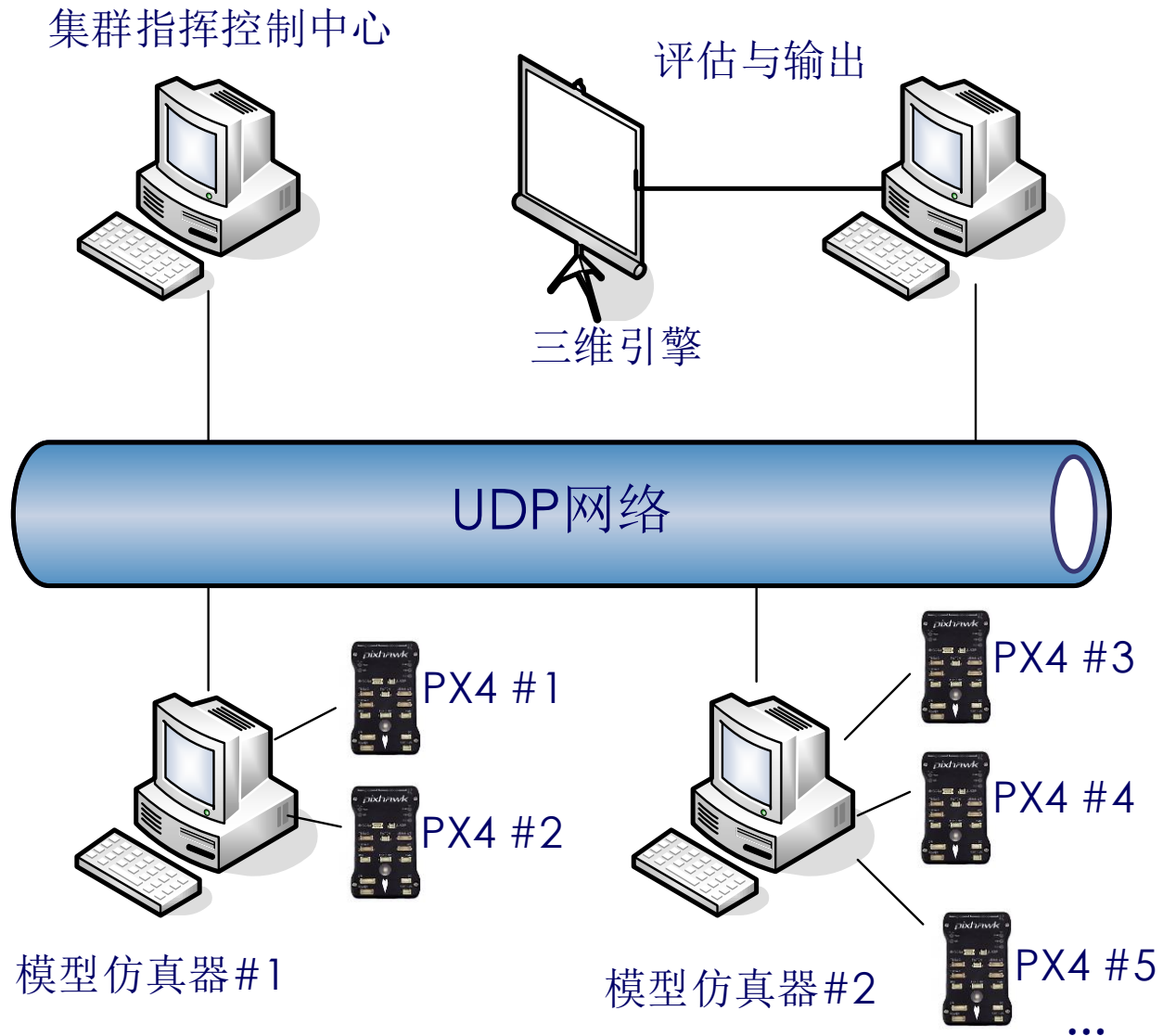




5. Distributed visual simulation

5.3 Distributed Cluster Networking Simulation — UAV Quantity Expansion Scheme

- Whether from the perspective of multi-machine simulation or from the perspective of UAV real cluster control, communication bandwidth and computing performance are always important bottlenecks restricting the increase of the number of clusters.
- Due to the performance bottleneck of the simulation computer, the number of hardware-in-the-loop simulations that a single computer can connect to Pixhawk is limited, so it is necessary to realize the arbitrary expansion of the number of UAVs by networking multiple computers.
- As the number of UAVs increases, the amount of data that aircraft communicate with each other increases dramatically until the communication bandwidth reaches saturation. Therefore, it is necessary to divide the whole UAV cluster into several subgroups, and use the network hierarchical approach to achieve a larger scale of cluster simulation.

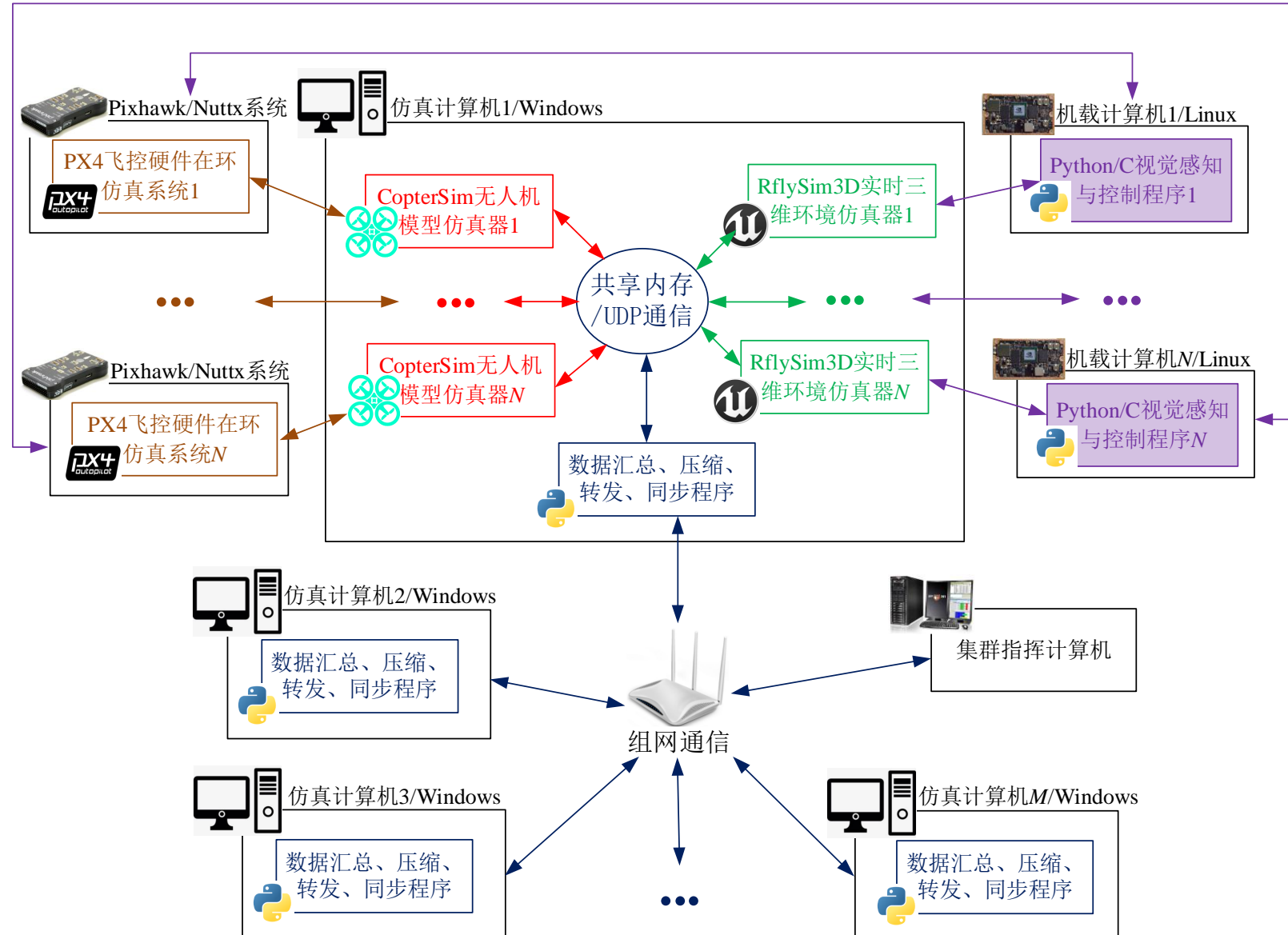




5. Distributed visual simulation

5.3 Distributed Cluster Network Simulation-Communication Optimization Scheme

- The communication between each program inside the computer adopts the shared memory or UDP communication mode, and the transmission of large data such as images is directly operated on the memory, with the lowest delay and the fastest speed.
- Each computer can open multiple hardware/software in-loop simulation systems to simulate multiple UAVs, and the flight control and airborne computers are connected through wired data transmission to reduce latency.
- The data sent and received by each computer is collected, compressed and synchronized by a specific module, and then sent out to ensure smooth communication within the network.
- Support the use of request communication (DDS protocol), support large-scale cluster simulation.

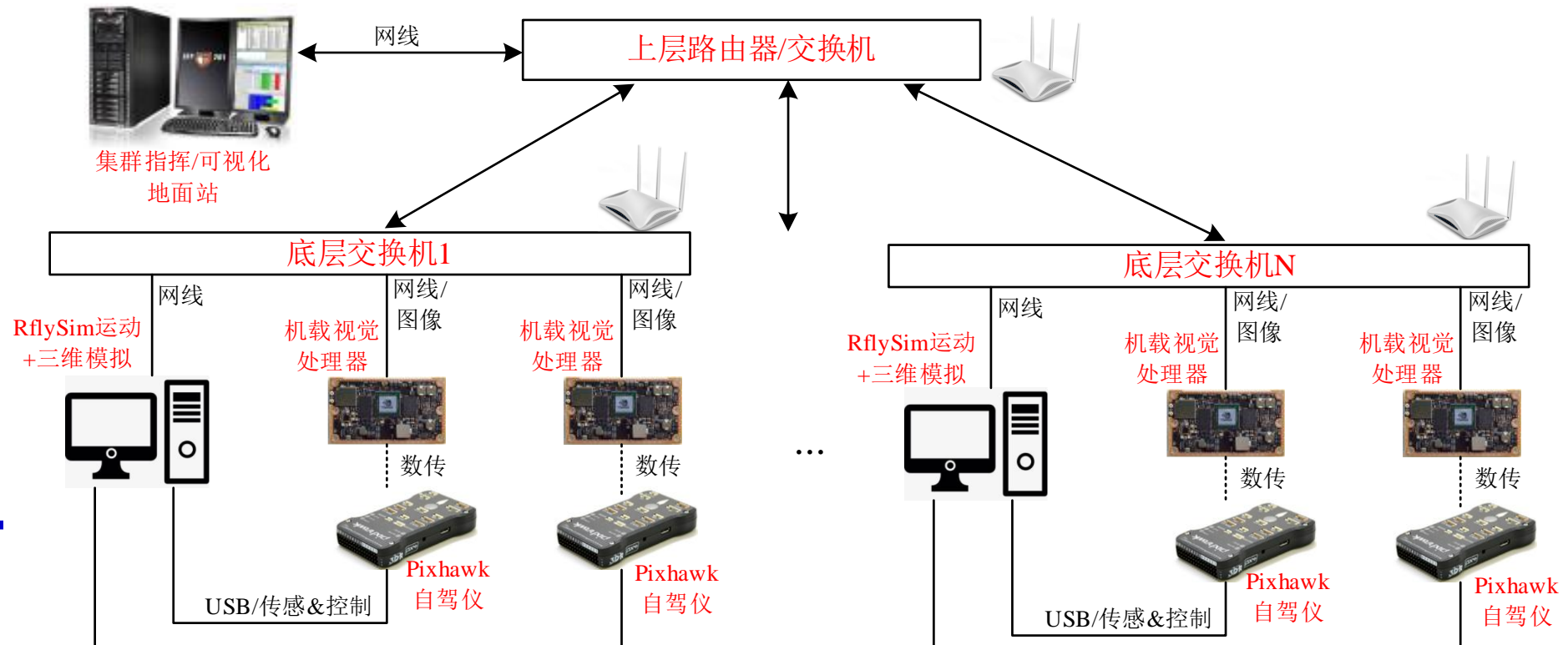




5. Distributed visual simulation

5.3 Distributed Cluster Network Simulation — Image Transmission Network Optimization

- In order to avoid transmitting a large number of images in the LAN, in this simulation framework, the visual images of the RflySim simulation computer are directly sent to the embedded host under the underlying switch in the form of a specified IP, and are not transmitted to the upper layer.
- In the upper router (switch), the state data of UAV is mainly transmitted, and the subscription and publication mechanism is adopted to avoid useless communication.





5. Distributed visual simulation

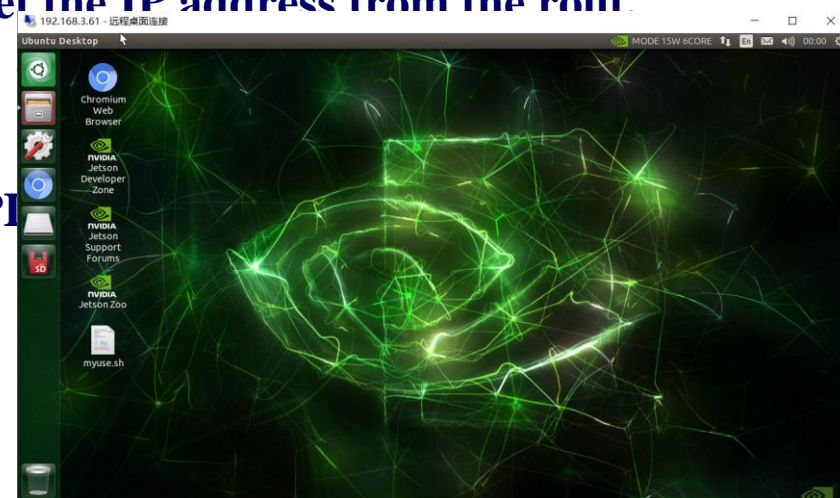
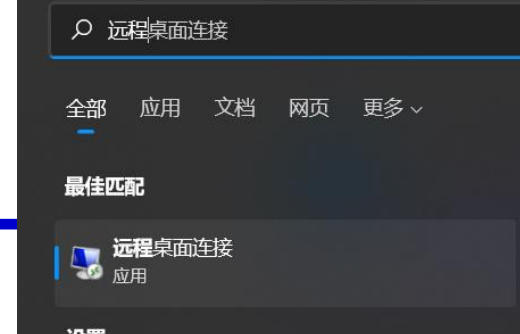
5.4 Remote access method of embedded host

- Because the distributed experiment involves many computers or embedded small hosts, in order to facilitate the observation of the results of subsequent experiments, it is recommended to use the "remote desktop connection" tool to connect other hosts under the Windows computer running on the RflySim platform. Note: Of course, you can also use a remote access tool dedicated to the embedded host (with a graphical interface, not a single command line).
- If the remote host is a Windows system, then after upgrading the professional version platform and enabling remote access, you can directly access the remote through the host name and enter the account password.
- If the remote host is a Raspberry Pi (or an Ubuntu system), You can go to the directory "8. RflySimVision \ 0. Api Exps \ 2-Distributed SimAPI \ 0. Preparation \ " RaspberryPI _ connection method ",
- And follow that steps in the document to configure the Raspberry Pi and get the IP address from the rout.

Graphical access is available through the Remote Desktop Connection tool.

Note: Our hardware is ready for direct connection.

- If it is TX2 or NX, it can be configured through the document "RflySimAPI Distributed SimAPI \ 0.Preparation", and then through the IP address Address and login password to connect. The effect is shown in the right figure.





5. Distributed visual simulation

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

5.5 Interface Test Lab-Introduction to Routine

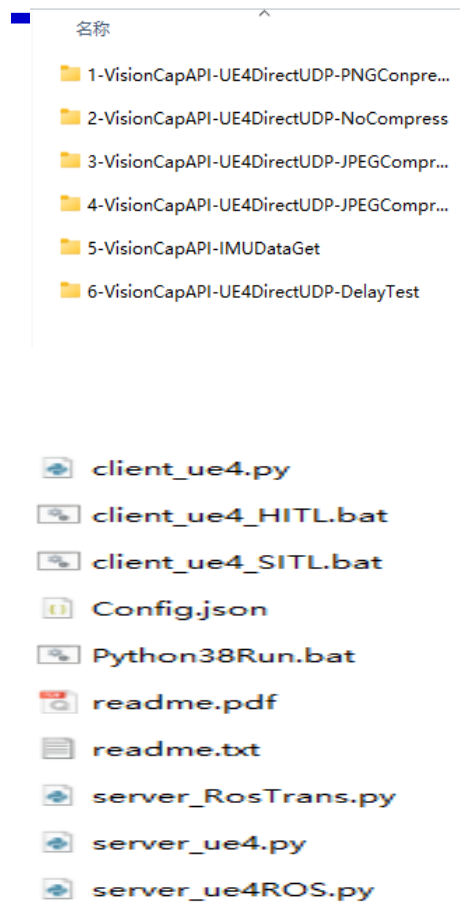
- See the directory "[8.RflySimVision\0.ApiExps\2-DistributedSimAPI\1.VisionAPIsTest](#)" for the experiment routines
- The key configuration item is the SendProtocol [8] vector in the Config. JSON configuration file in section 2.4 of this lecture.
- **Where, SendProtocol [0] represents four graph transfer protocols.**
- **0: UE4 writes the image into the shared memory.** Other programs of this computer can read the image through the memory. If the remote computer needs the image, it can forward the image again. Features: The shared memory mode allows the local machine to read pictures at a high speed, but when the remote computer reads pictures, it needs additional transfer, and the amount of calculation and delay are large.
- **1: UDP is directly transmitted to the remote computer (through the IP specified by SendProtocol [1-4]), and PNG compression is used for sending compression and receiving decompression.** Features: Low latency, PNG lossless compression, low compression ratio but good quality.
- **2: UDP is directly transmitted to the remote computer, and the picture is not compressed.** Features: minimum delay, small amount of calculation, high network pressure.
- **3: UDP is directly transmitted to the remote computer, and JPG compression is used.** Features: Low network pressure, moderate delay and computation, but the image is lossy compression, with a slight loss of accuracy, but it meets the real-time visual control requirements of UAV.
- **Note: The free version can only use the 0 mode, and the full version is recommended to use the 3 mode (default for subsequent routines).**



5. Distributed visual simulation

5.5 Interface Test Experiment-Routine File Structure

- See "[8.RflySimVision\0.ApiExps\4-AdvApiExps\9.VisionAPIsTest](#)" for the experiment routine.
- It can be seen from the right figure that the six routines correspond to four image transmission modes, namely, shared memory, PNG direct transmission, uncompressed direct transmission, JPEG direct transmission, IMU data acquisition, image acquisition and transmission limit delay experiment.
- The following types of files are included in the routine:
- Client _ * *.bat file: enable the software/hardware in-loop simulation, and set the number of aircraft to 1 in this routine.
- Config. JSON file: configure camera parameters, including several cameras, camera position, camera type (RGB, depth, etc.)
- File client _ ue4.py: start the script for image transmission
- File server _ ue4.py: Enable image reception and control
- Python 38Run.bat: Platform Python shortcut
- * * ROS. Py files: ROS-enabled routines and interfaces



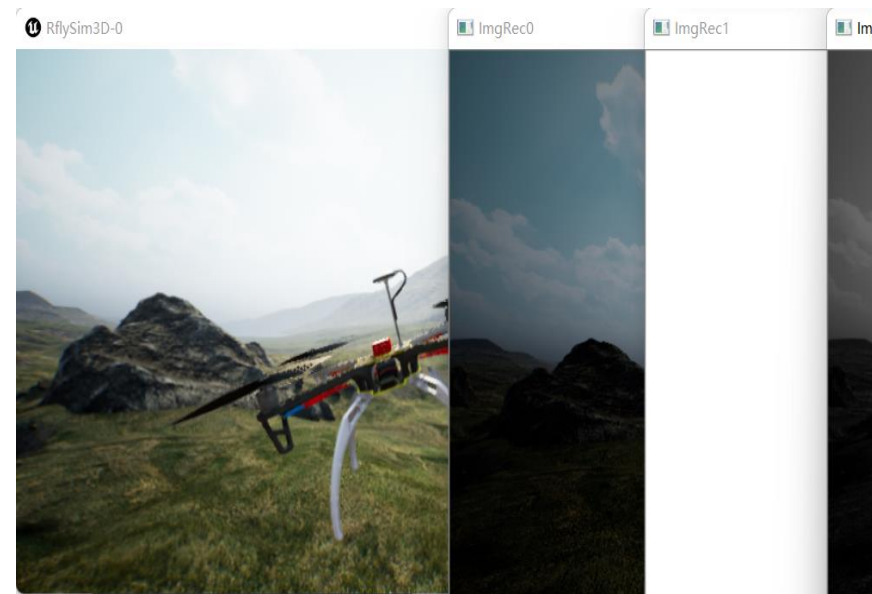


5. Distributed visual simulation

5.6 Interface test experiment — single computer experiment

- Four modes of test routines, on a computer, can be tested in the following way
- Double-click to run the "client _ ue4" _ SITL. Bat "to open the software in-the-loop simulation of an aircraft
- Double-click to run "Python 38Run.bat" to start the platform Python environment, and enter the command "python client _ ue4.py" to start the image request and image forwarding. (VS Code can only run one program, which is reserved for server)
- Open "server _ ue4.py" with VS Code, read the code and run the routine.
- The motion of the four experimental aircraft is the same. They all take off to 10 meters and begin to fly in circles.
- After all server programs are run, three image data are obtained, which are RGB, depth, and grayscale.
- In mode 0, the client can preview three images. In other modes, there is no image without python transfer.
- The depth map has a distance limit, and the results can only be observed at close objects. In the experiment, it is white after takeoff.

```
C:\WINDOWS\system32\cmd.exe - python client_ue4.py
Put your python scripts 'XXX.py' into the folder 'C:\PX4PSP\RflyS
s\PythonVisionAPI\4-DistributedSimAPI\1-VisionAPITest\0-VisionCap
SharedMemory'
Use the command: 'python XXX.py' to run the script with Python
C:\PX4PSP\RflySimAPIs\PythonVisionAPI\4-DistributedSimAPI\1-Vision
Test\0-VisionCapAPI-SharedMemory>python client_ue4.py
Got 3 vision sensors from json
Sensor req success from UE4.
Start Transfer Img
```





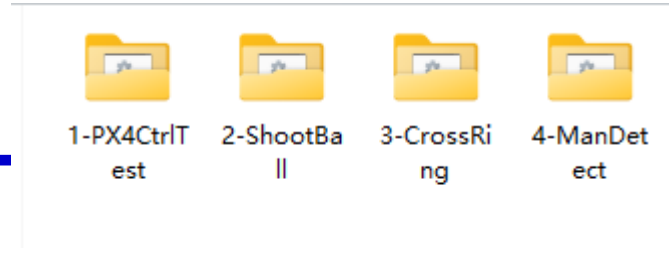
5. Distributed visual simulation

5.6 Interface test experiment — dual computer experiment

- The process is basically the same as that of a single computer, except that the client needs to add the IP of a remote computer (Windows computer, Linux computer or Raspberry Pi), and the server program needs to run on the remote computer. The steps are as follows:
- Observe the local IP address (for example, 192.168.3.80) and the IP address of the remote computer (for example, 192.168.3.81) through the router, and record the IP address.
- Double-click to run the "client _ ue4" _ SITL. Bat "to open the software in-the-loop simulation of an aircraft. Note: The routine uses the broadcast mode for MAVLink data transmission by default. In order to improve the efficiency, the "SET IS _ BROADCAST = 1" statement can be changed into "SET IS _ BROADCAST = 192.168.3.81", where the IP address of the remote computer needs to be filled in.
- Open "client _ ue4.py" with VS Code, uncomment the following statement "# vis. RemotSendIP =", and set the IP address to the IP address of the remote computer, for example, "vis. RemotSendIP = '192.168.3.81'"
- Copy all the files in the folder to another computer, open "server _ ue4.py" through Python environment or VS Code and run it to receive and display the pictures. Note: In order to improve the communication efficiency, the "255.255.255.255" broadcast address can be changed to the "192.168.3.80" host address.
- Note: Uncomment "# print ('Img', idx)" in the "VisionCaptureApi. Py" "to observe the category and receiving timestamp of each picture. This data can be used to analyze and test the synchronism and packet loss rate.



5. Distributed visual simulation



5.7 Single Aircraft Visual Control Experiment

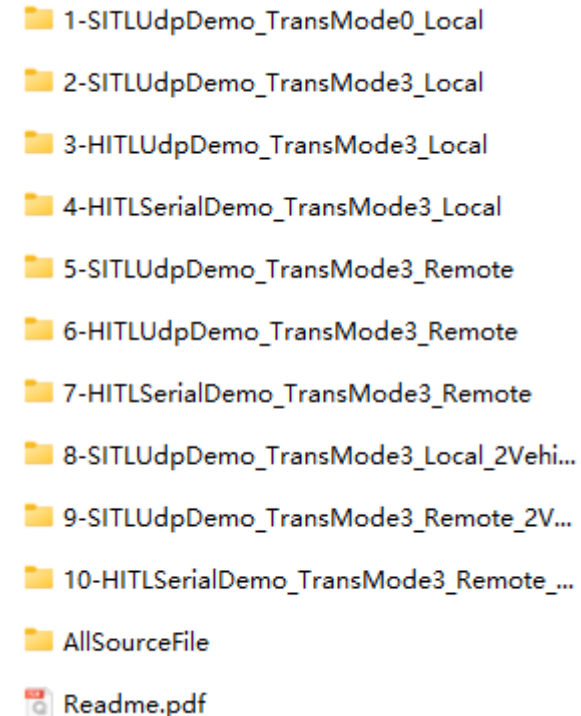
- Enter the "[8.RflySimVision\3.CustExps\2-DistributedSimDemos\e1_OneVehicleCtrls](#)" directory, and you can see the previous visual API, ball impact, ring piercing, and binocular face recognition routines. It is reproduced here in a distributed way.
- The experimental process is basically the same as the previous one, and the experimental effect is the same as Section 3.
- For stand-alone experiments, run bat first, then Python client, and finally server.
- For online experiment, first record the IP addresses of the two hosts, then run bat, open the client with VS Code and set the IP address of the host, finally copy all files to the remote host, and then run server.
- Note: If there are not many computers in the LAN, you can directly modify the client and set the IP address of the remote host to the broadcast address of "255.255.255.255" without going to the router to check the IP address of the computer.
- Note: It is recommended to set the IP address of the computer or embedded host as static IP in the router, so that the IP address will not change after each restart, and there is no need to query the IP address frequently.
- Note: It is recommended to use remote access to operate the lab on one computer.



5. Distributed visual simulation

5.8 Multi-Aircraft Visual Control Experiment-General Introduction

- Enter the directory of "[8.RflySimVision\3.CustExps\2-DistributedSimDemos\2_MultipleVehicles](#)", and you can see several experiment routines. The file names have the following meanings:
- TransMode indicates the transfer mode, 0 is the shared memory, and 3 is the JPEG direct transfer.
- SITL indicates the use of PX4 software in-loop simulation, which requires running SITL *.bat to start the simulation closed loop; HITL indicates hardware in-loop simulation.
- UDP indicates that the communication between the flight control and the Server is in UDP mode, while Serial indicates that the data transmission module needs to be used to communicate between the flight control and the Server through the serial port.
- Local indicates that the lab can be run on a single computer; Remote indicates that the lab requires at least two computers and requires an IP address.
- The suffix of "_" indicates the number of aircraft. If there is no suffix, the default is 1 aircraft with 3 cameras. 2V4C indicates 2 aircraft with 4 cameras, that is, 2 cameras for each aircraft.
- The AllSourceFile folder is the template source file for all routines.

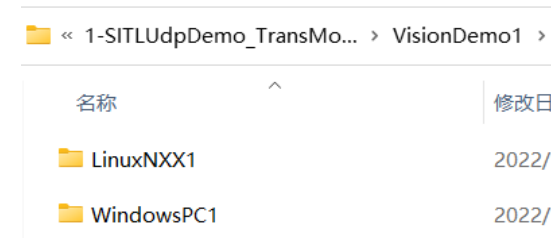
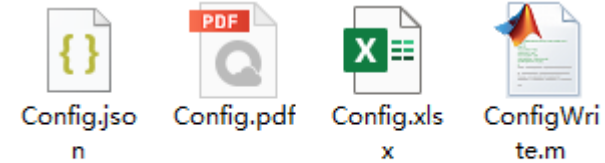




5. Distributed visual simulation

5.8 Multi-Aircraft Visual Control Experiment-Routine Introduction

- Entering the "1-SITLUdpDemo _ TransMode0 _ Local" directory, you can see several files used to generate the required aircraft and camera routines with one click.
- The Config. JSON is the same as the previous profile, defining parameters for multiple cameras. Note: The maximum number of cameras supported by each aircraft should be defined here, and the first few of them can be selected and enabled in the config. Xlsx.
- The config. Xlsx defines Windows host data, the number of Linux hosts (the same as the number of aircraft) and IP addresses, the location of each aircraft, the number of cameras on each aircraft, and so on.
- According to the distributed simulation framework defined in the Config. Xlsx, the ConfigWrite. M can automatically generate distributed simulation routines of any number of aircraft, and can select a variety of simulation modes. The template file for the routine comes from the "All Source File" folder in the upper directory.
- The experimental process is also very simple: 1) Use MATLAB to locate the directory where the "ConfigWrite. M" file "is located, and then right click to run it to generate the folder of Vision Demo *, where * represents the total number of aircraft; 2) Run the bat Qidong script of Windows PC * on the simulation computer, and then run the client image transfer program; 3) Copy LinuxNXX * to the remote host, and run the server. Note: Support multi-computer and host networking.





5. Distributed visual simulation

5.8 Multi-Aircraft Visual Control Experiment-Introduction to Config. Xlsx Usage

HIL or SIL, UDP or Serial, COM Name ,Baud Num	Hardware or software in the loop (HIL/SIL), UDP or serial communication (UDP/Serial), serial port number (HIL only), baud rate (HIL only)	SIL	UDP
WindowsPCIPList	IP address of the Windows computer	127.0.0.1	
VehicleNumOnPC	Number of aircraft on each Windows PC		1
NXXIPList	IP list for each embedded computer NXX, the number of which shall be the same as the number of aircraft	127.0.0.1	
CameraNumList	Number of cameras per aircraft		3
VehicleXPosList(m)	List of X coordinates of the aircraft in m		0
VehicleYPosList(m)	List of Y coordinates of the aircraft in m		0
VehicleYawList(degree)	List of yaw angles of the aircraft in radians		0
UE4_MAP	Map name or serial number	GrassLands	
PX4SitlFrame	When the software is in the loop, set the PX4 internal rack Airframe type	iris	
DLLModel	The name or serial number of the DLL model. By default, select 0 for multi-rotor, and select DLL for fixed-wing, etc.		0
	Whether to enable synchronous startup 0: Do not enable synchronous startup 1: Enable synchronous startup, and trigger the preceding aircraft after the last aircraft command is executed 2: Trigger the preceding aircraft in turn after the last aircraft script runs; the second column of this option can set the delay time (s) for		



5. Distributed visual simulation

5.8 Multi-Aircraft Visual Control Experiment-Introduction to Config. Xlsx Usage

- Take HITLUdpDemo _ Local _ 1V3C as an example.
- You need to use HITL hardware in the loop, so you need to prepare a Pixhawk and connect it to your computer.
- In Local mode, both the computer and the remote host IP address are set to 127.0.0.1
- One computer, one plane, so the Windows PC IP List has only one column, and each plane has three cameras.
- **Experimental steps: Run the bat and client of Windows PC1, and then run the server under Linux NXX1.**

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



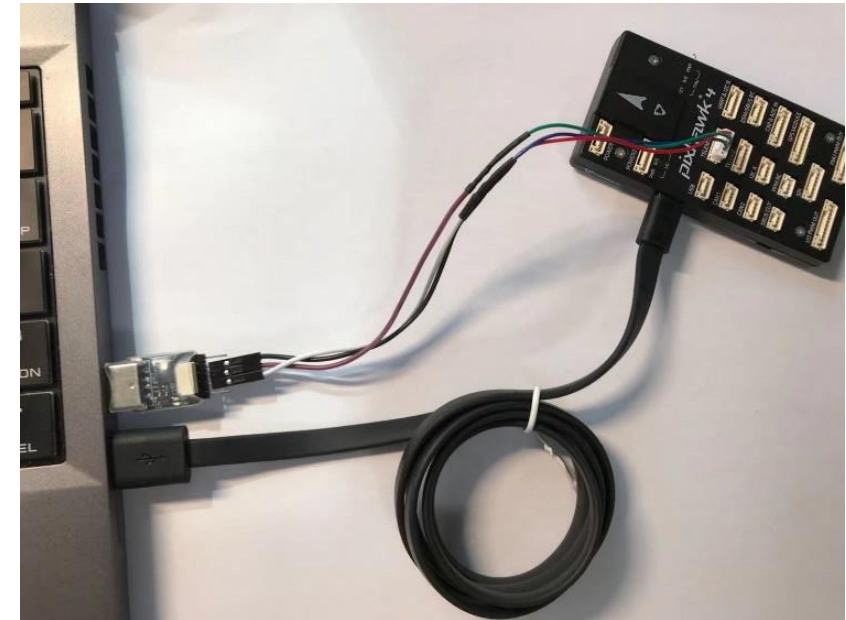


5. Distributed visual simulation

5.8 Multi-Aircraft Visual Control Experiment-Introduction to Config. Xlsx Usage

- Taking the HITLSerialDemo _ Local _ 1V3C as an example, it is necessary to prepare a data transmission connection between the flight control TELEM1 and the computer USB port, that is to say, the computer needs to occupy two USB ports, one for HITL simulation and the other for server and PX4 communication.
- In the following table, COM14 is the serial port number of data transmission, and the 57600 is the baud rate of data transmission, which can be set in QGC.

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



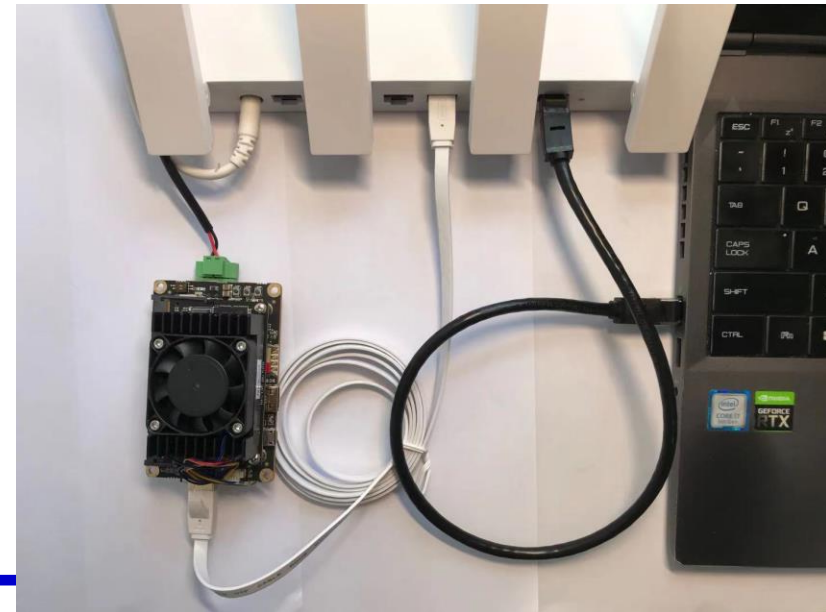


5. Distributed visual simulation

5.8 Multi-Aircraft Visual Control Experiment-Introduction to Config. Xlsx Usage

- Taking the SITLUdpDemo _ Remote _ 1V3C as an example, it is necessary to prepare: 1 computer, 1 host, 1 router (or switch), power supply, and several network cables.
- In the table below, 192.168.3.80 is the computer IP, and 192.168.3.55 is the remote host IP, which needs to be modified according to the personal network configuration.
- During the experiment, the files in the Windows PC1 directory were run on the computer, and the files in the Linux NXX1 directory were copied to the host computer to run.

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



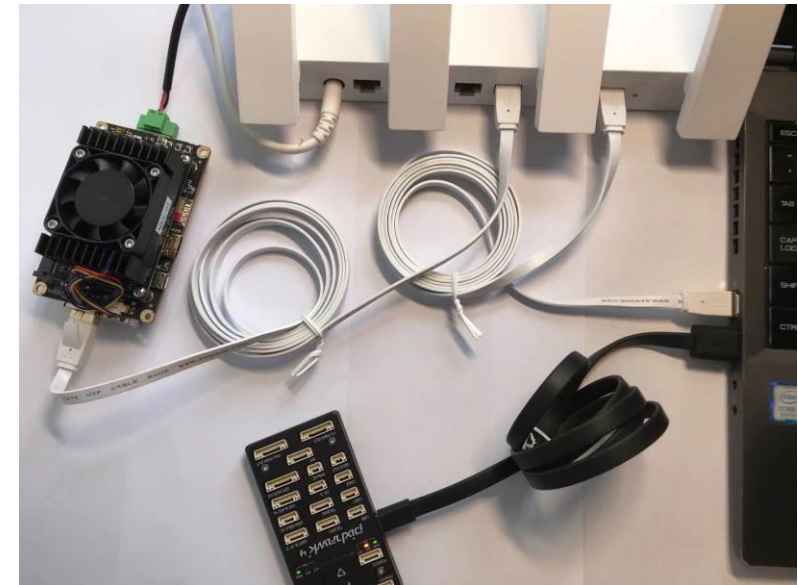


5. Distributed visual simulation

5.8 Multi-Aircraft Visual Control Experiment-Introduction to Config. Xlsx Usage

- Taking the HITLUdpDemo _ Remote _ 1V3C as an example, it is necessary to prepare: 1 computer, 1 host, 1 flight control, 1 router (or switch), power supply and several network cables.

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



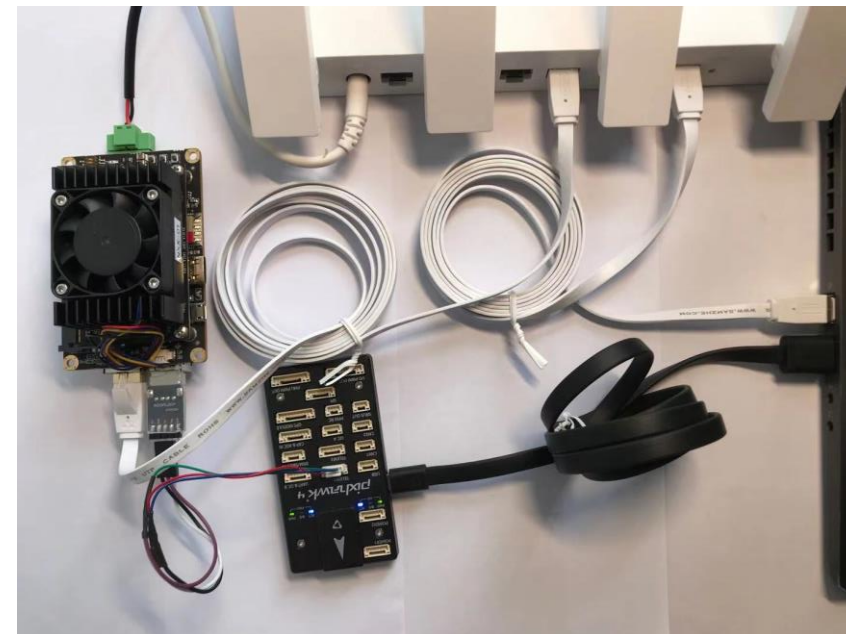


5. Distributed visual simulation

5.8 Multi-Aircraft Visual Control Experiment-Introduction to Config. Xlsx Usage

- Taking the HITLSerial Demo _ Remote _ 1V3C as an example, it is necessary to prepare: 1 computer, 1 host, 1 flight control, 1 data transmission, 1 router (or switch), power supply and several network cables.

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



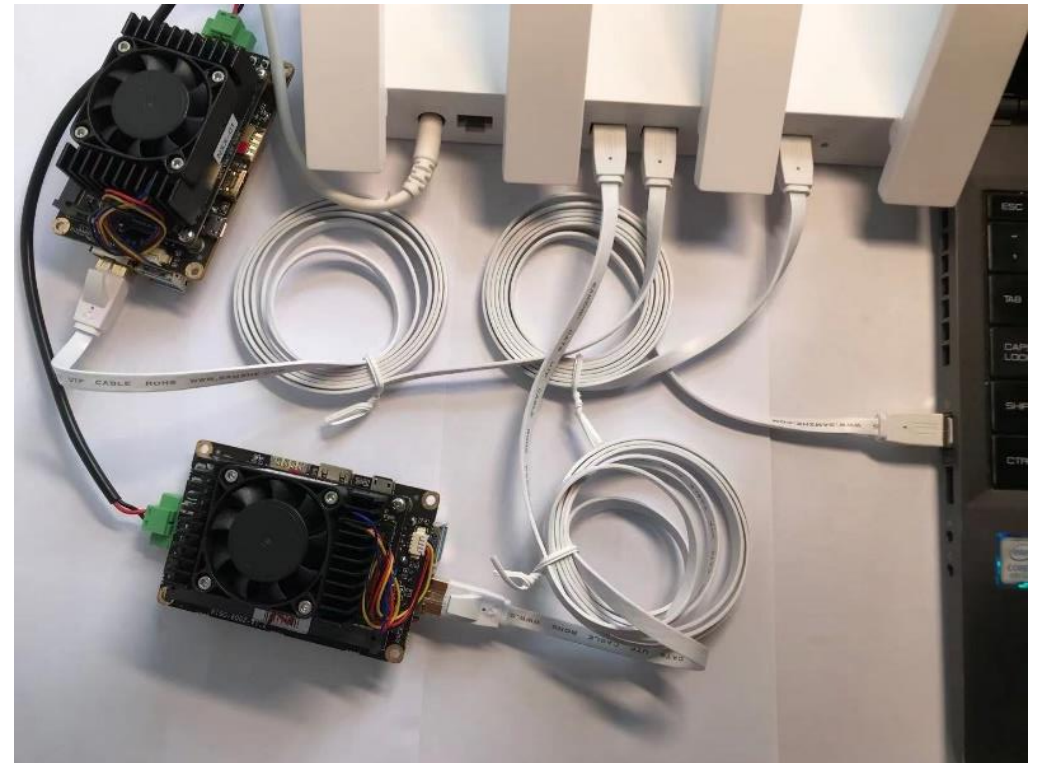


5. Distributed visual simulation

5.8 Multi-Aircraft Visual Control Experiment-Introduction to Config. Xlsx Usage

- Take SITLUdpDemo _ Remote _ 2V4C as an example, it is required to prepare: 1 computer, 2 hosts, 1 router (or switch), power supply and several network cables.

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



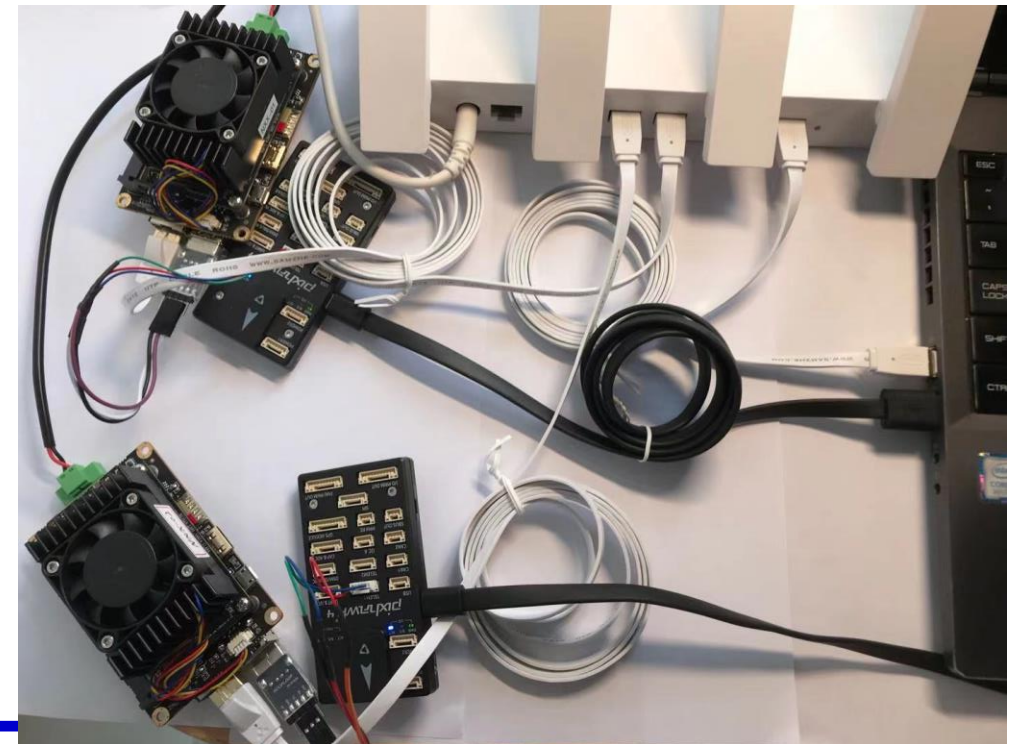


5. Distributed visual simulation

5.8 Multi-Aircraft Visual Control Experiment-Introduction to Config. Xlsx Usage

- Take the HITLSerial Demo _ Remote _ 2V4C as an example, it is necessary to prepare: 1 computer, 2 hosts, 2 flight controllers, 2 data transmitters, 1 router (or switch), power supply and several network cables.

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

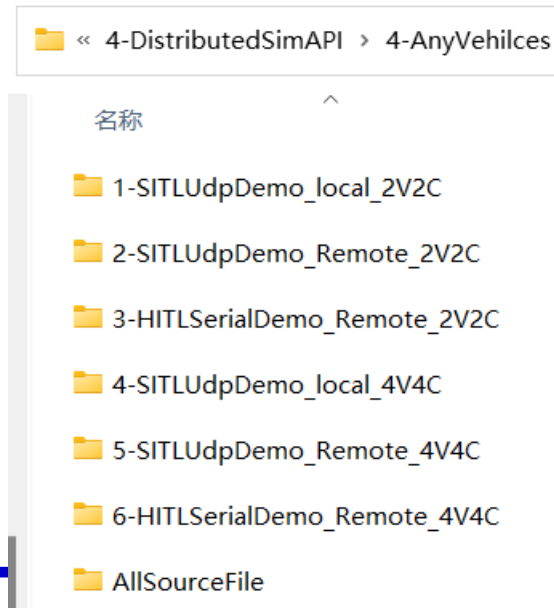
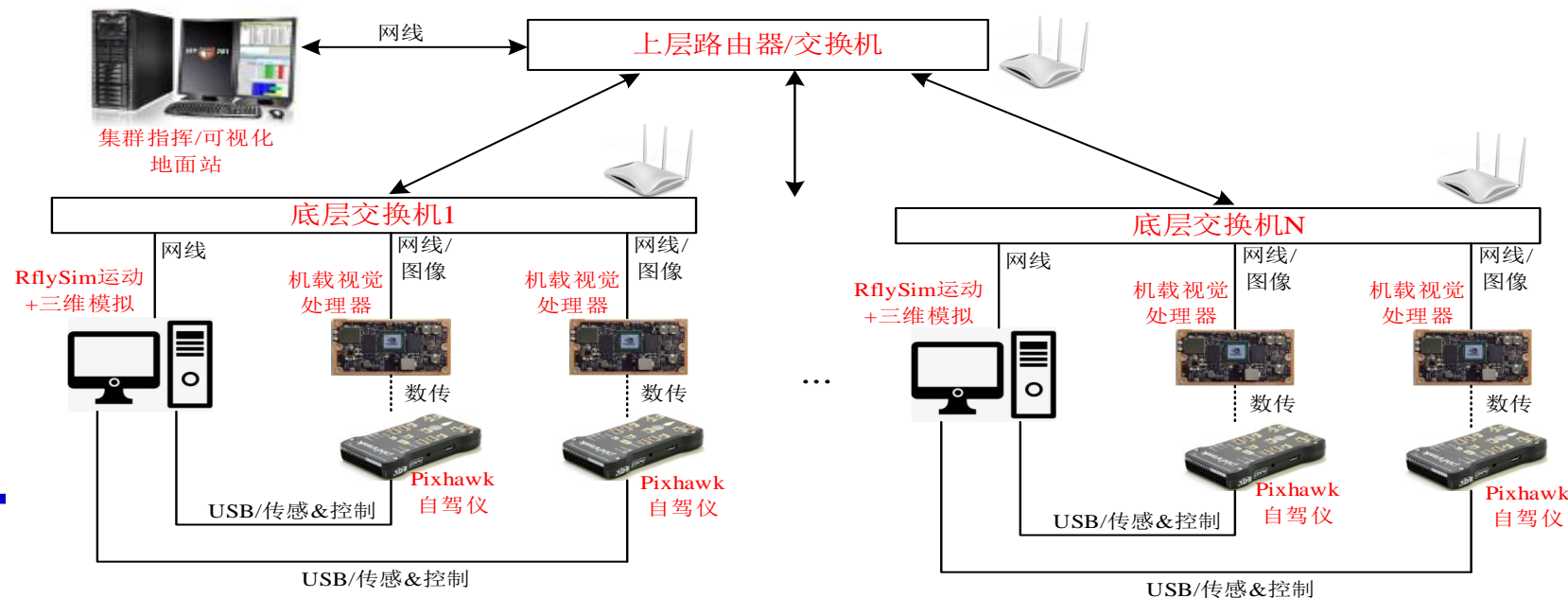




5. Distributed visual simulation

5.9 Multi-Computer Multi-Aircraft Vision Control Experiment-General Introduction

- Enter the "[8RflySimVision\3.CustExps\2-DistributedSimDemos\03 AnyVehilces](#)" directory, and you can see the routines of multi-computer and multi-aircraft. The theory class supports the networking of any number of computers and hosts in the LAN, but the architecture shown in the following figure must be adopted. Ensure that the image is only propagated within the underlying switch using the specified IP.





5. Distributed visual simulation

5.9 Multi-Computer Multi-Aircraft Visual Control Experiment-Introduction to Config. Xlsx Usage

- Take the SITLUdpDemo _ Remote _ 4V4C (each computer is equipped with two airplanes, and each airplane is equipped with one camera) as an example. It is necessary to prepare two computers, four hosts, two routers (or switches), a power supply, and several network cables.
- Six IP addresses need to be filled in, and the experimental phenomenon is that the aircraft penetrate the ring in turn and are visible to each other (occlusion may lead to the failure of penetrating the ring).

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		
NXXIPList	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			
PX4SitlFrame	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. Distributed visual simulation

5.9 Multi-Computer Multi-Aircraft Visual Control Experiment-Introduction to Config. Xlsx Usage

- Taking the HITLSerial Demo _ Remote _ 4V4C (each computer is equipped with two airplanes, and each airplane is equipped with one camera) as an example, it is necessary to prepare two computers, four hosts, four flight controllers, four data transmitters, two routers (or switches), a power supply, and several network cables.
- The synchronous start simulation function can be enabled. When the last host runs the server, the server program blocked in front will be triggered in turn to realize the phenomenon that the aircraft passes through the ring at the same time or in turn.

HIL or SIL, UDP or Serial, COM Name	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
.Baud Num					
WindowsPCIList	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		
NXXIPList	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			
PX4SitiFrame	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			



Brief summary

- This lecture mainly explains the development course of flight control algorithm, which is divided into two parts: basic experiment and advanced experiment, so that students can familiarize themselves with the development process of multi-rotor theoretical design, RflySim platform simulation and physical real machine control as soon as possible.
- The basic experiment is based on the RflySim platform software-in-the-loop and hardware-in-the-loop simulation process learning, and the advanced experiment is based on the learning route of multi-rotor theoretical design and modeling experiment → estimation experiment → control experiment → decision-making experiment.

If you have any questions, please go to the <https://doc.rflysim.com/> for more information.



RflySim More
Tutorials



Scanning code
consultation and
communication



Freescale RflySim
Technology Exchange
Group





Thank you!