Multicopter Design and Control Practice Experiments

RflySim Advanced Courses
Lesson 06: Vison-Based Control

Dr. Xunhua Dai, Associate Professor,
School of Computer Science and Engineering,
Central South University, China;
Email: dai.xh@csu.edu.cn ;
https://faculty.csu.edu.cn/daixunhua

北航可靠飞行控制研究组
BUAA Reliable Flight Control Group

Content

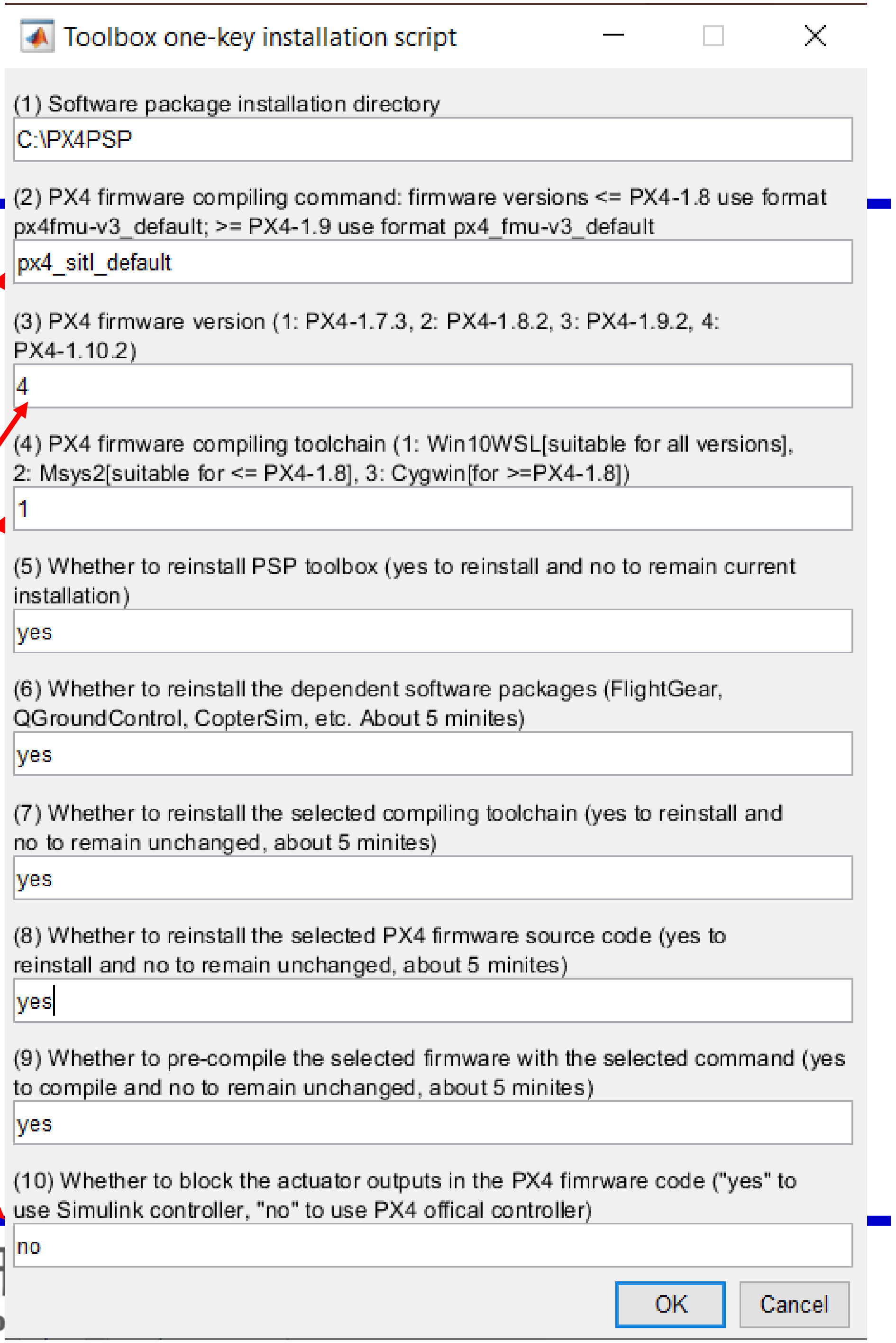
Path of demo source code of this lesson: RflySimAPIs\PythonVisionAPI

1. Setup Instructions

1.1 RflySim platform configuration

- When this platform is used for vison controller development, it is recommended to re-run the "**OnekeyScript.p**" script and use the configuration as the right figure to run the script as follows:
- Use the PX4 SITL software-in-the-loop firmware to compile and enter command "**px4_sitl_default**"
- Use the latest PX4 firmware **PX4-1.10.2**, select "**4**" for the firmware version
- Use **Win10WSL** compiler, so select "**1**" for the compiler
- Whether to shield PX4 output items, select "**no**"
- Click the "**OK**" button to start the installation.

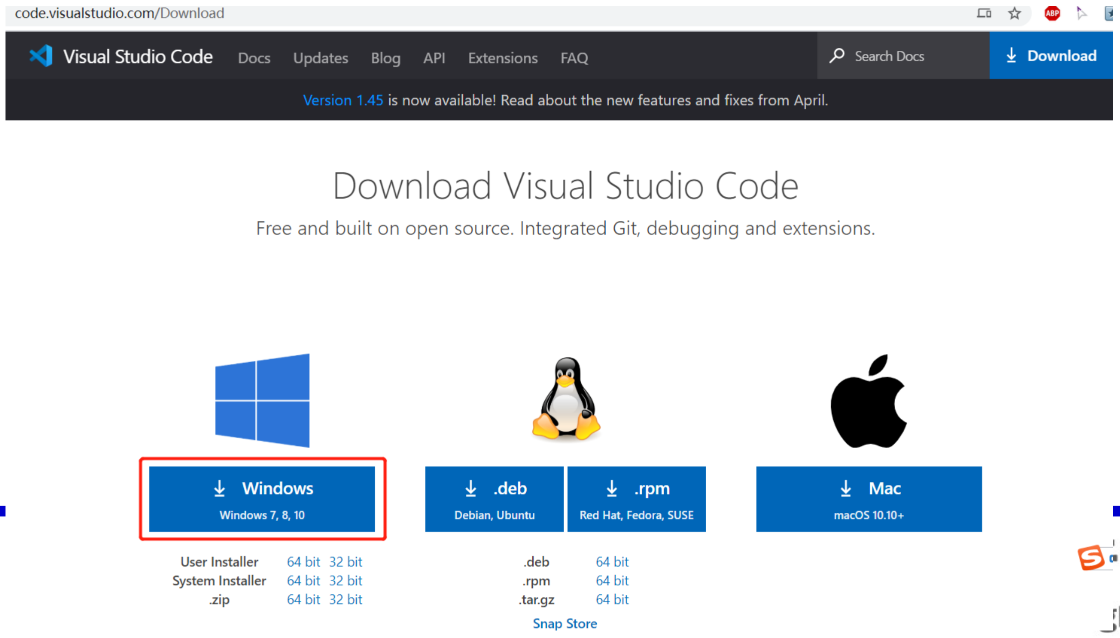
If you want to use Pixhawk for Hardware-In-the-Loop (HIL) simulation, you also need to correctly configure the Pixhawk autopilot through QGC according to the method in **Section 2.5** of "**RflySim_Lesson_01_Introduction.pdf**"

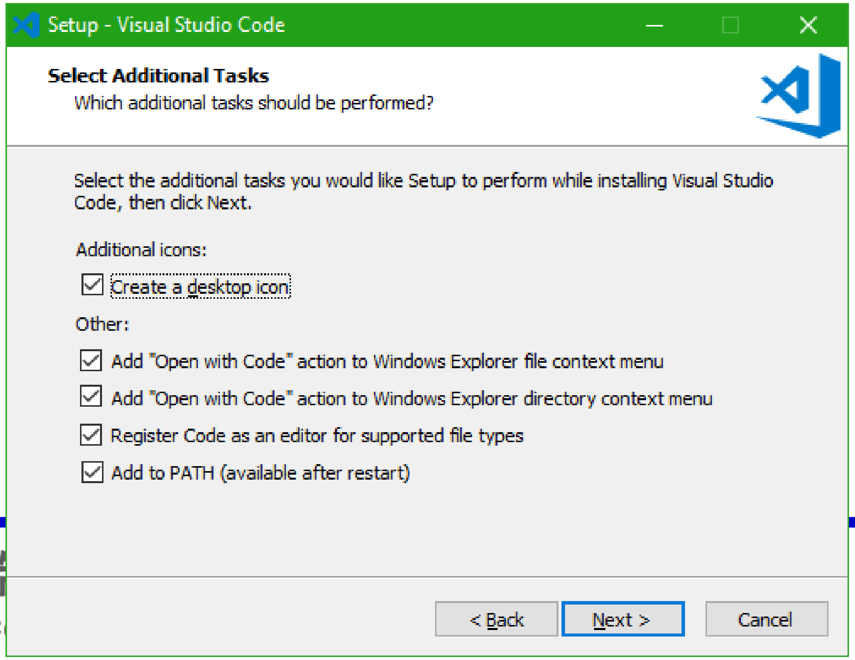


1. Setup Instructions

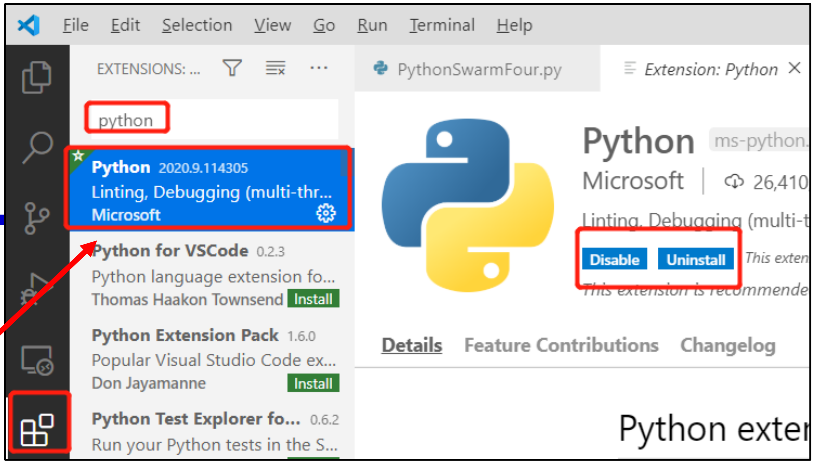
1.2 VS Code editor installation

- The Python editor of this course recommends using **VS Code** (not necessary, but easy to read the source code and run), the installation steps are as follows:
- Visit https: //code.visualstudio.com/Download to download the latest VS code installation package (you can also use **VSCodeUserSetup-x64-***.exe** in the **RflySimAPIs\PythonVisionAPI** folder)
- When installing, just select the default configuration. Pay attention to the settings in the lower right figure to facilitate opening Python files directly.

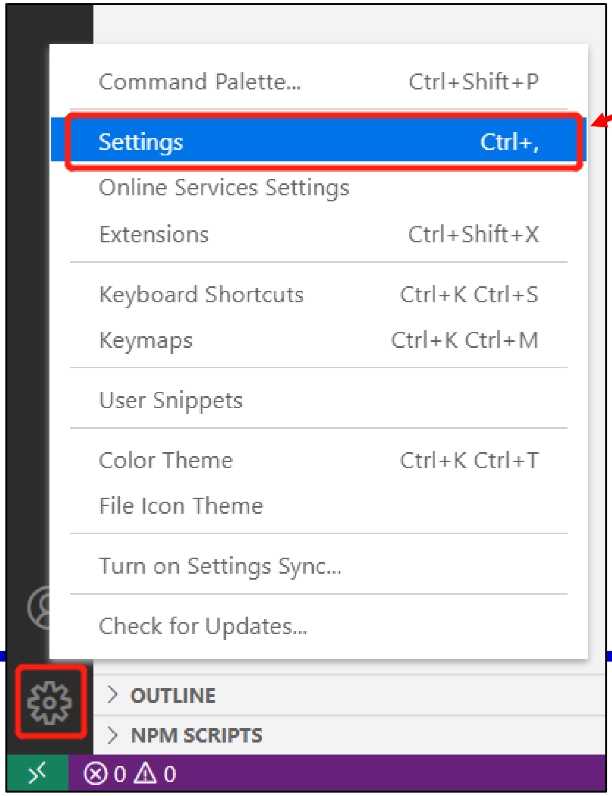


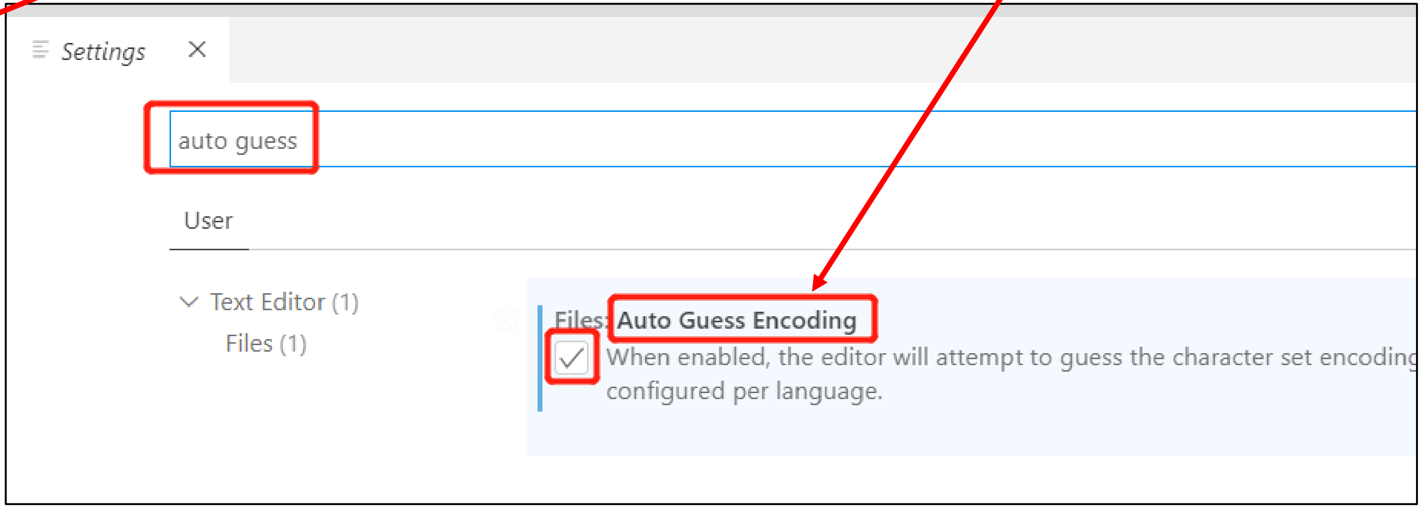


1.3 VS Code editor configuration



- Go to VS Code "**Extensions**" page, search and install "**Python**".
- Automatically identify file encoding (solve the problem of Chinese gibberish characters). Open VS Code, click **setting icon** and then "**Settings**" button; or click Menu bar: **File** → **Preferences** → **Settings** → Search for "**auto guess**", check to enable automatic guessing of file encoding function

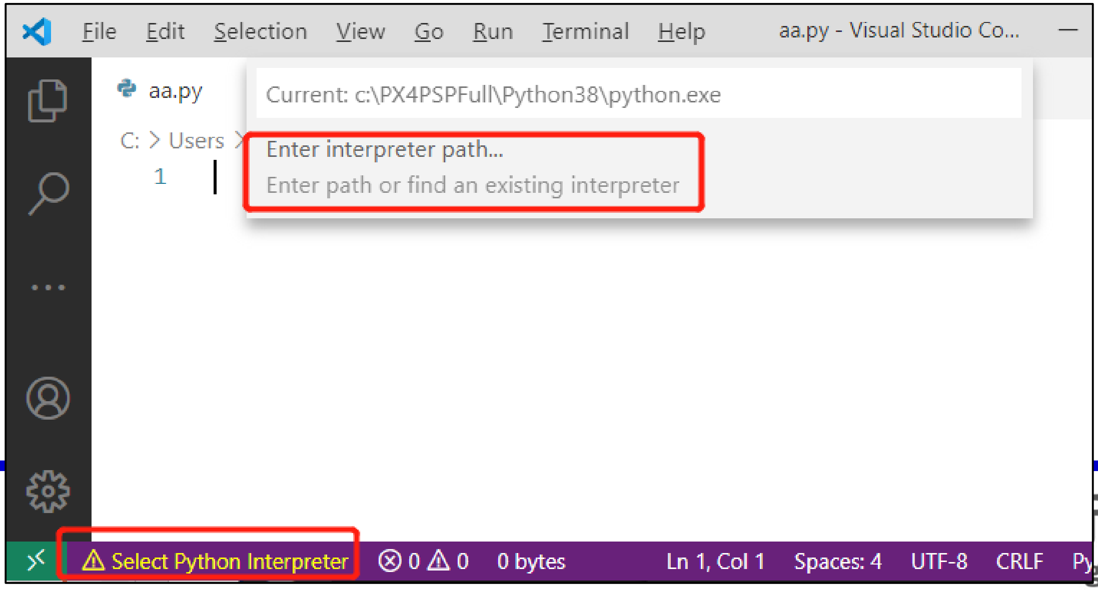


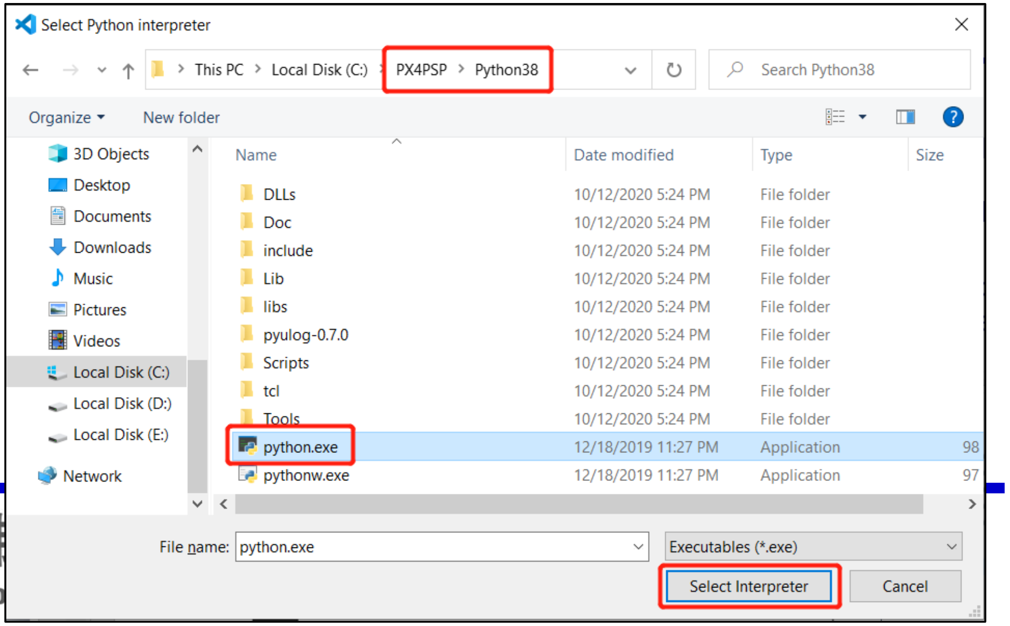


1. Setup Instructions

1.4 VS Code Python environment configuration

- Open the "**RflySimAPIs\Python38Scripts\ImgCVShow.py**" file (or any **.py** suffix file) with VS code. As shown in the figure below, click the yellow word "**Select Python interpreter**" option in the lower left corner, and click "**Enter interpreter path**" in the pop-up item.
- As shown in the figure on the right, in the pop-up explorer window, select the python.exe file in the **Python38** folder under the installation directory (default **C:\PX4PSP**), and click "**Select Interpreter**".

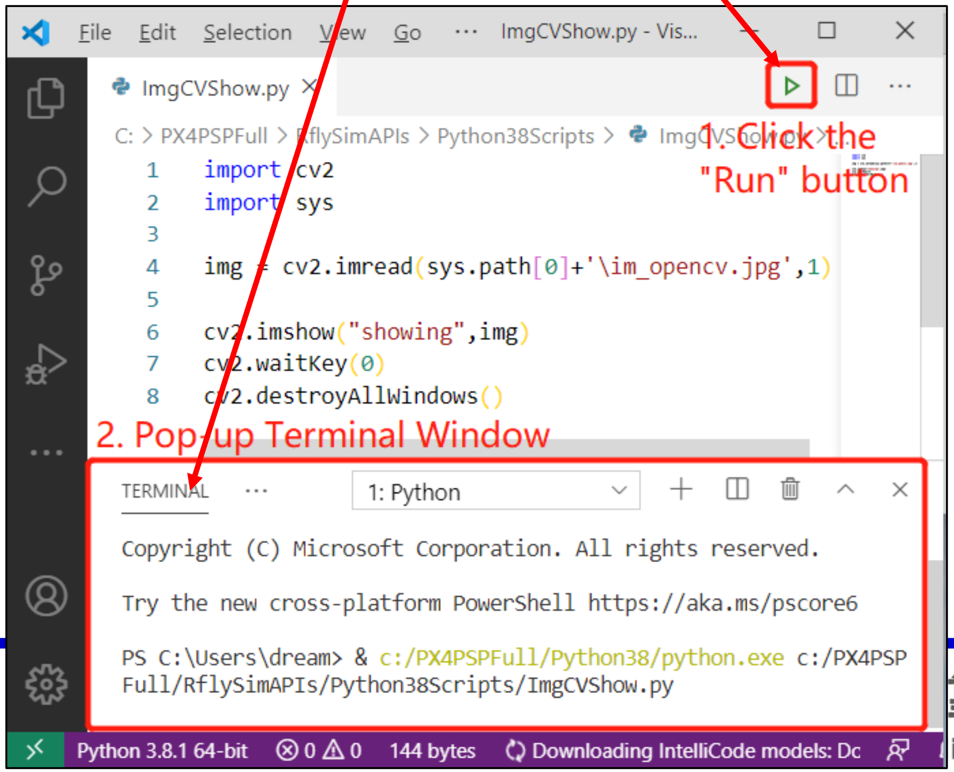




1. Setup Instructions

1.4 VS Code Python program running

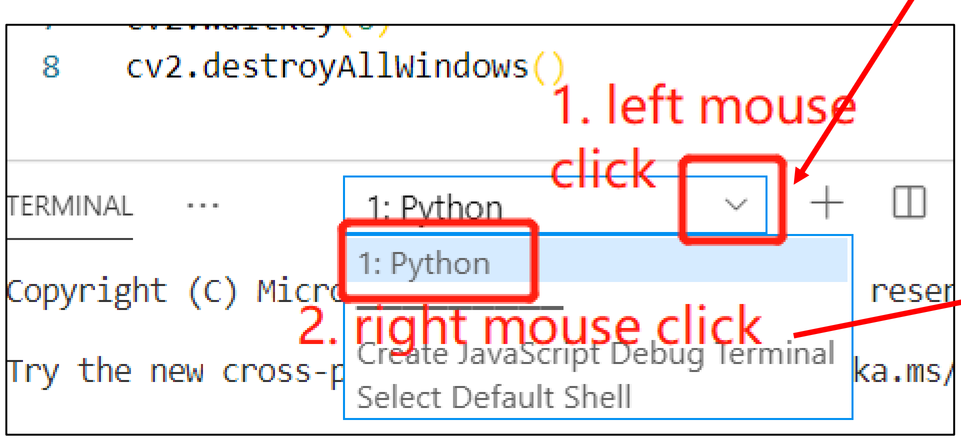
- Open the "**RflySimAPIs\Python38Scripts\ImgCVShow.py**" file with VS Code.
- Click the **triangle arrow** on the upper right side of VS Code to run, and the terminal window of "**TERMINAL**" pops up at the bottom of the VS Code interface to check the running status of the script. At the same time, a picture shown in the lower right pops up, indicating that the environment is installed correctly.

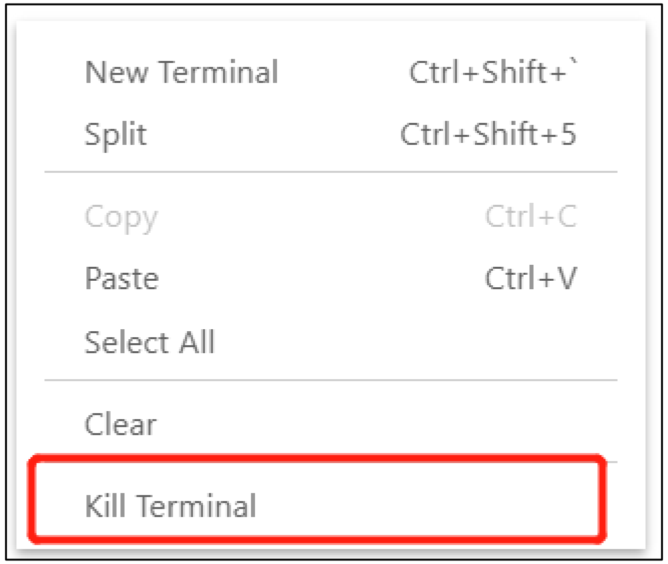


1. Setup Instructions

1.5 VS Code Python program force close

- As shown on the left, click the down **triangle arrow** next to the Python button at the bottom right to expand and display all terminal windows, and press the right mouse button on the current terminal window entry (“**1: Python**” in the figure below). As shown in the figure on the right, click “**Kill Terminal**" in the pop-up menu to terminate the Python program.





1. Setup Instructions

1.6 Python38Env environment

- The **Python38** folder under the installation directory (**C: \PX4PSP** by default) contains the latest Python operating environment, which already contains a series of tools such as **OpenCV, pymavlink, pip** and etc., which can be directly used for basic top-level control algorithm development for drone
- This Python environment is completely independent from other environments installed on Windows. Also, not affect other Python environments or be affected by their configuration.
- Double-click the desktop "**Python38Env**" shortcut or double-click the "**RflySimAPIs\Python38Env.bat**" script, the terminal window shown below (registered Python directory) will pop up, and the Python environment can be called

C:\WINDOWS\system32\cmd.exe

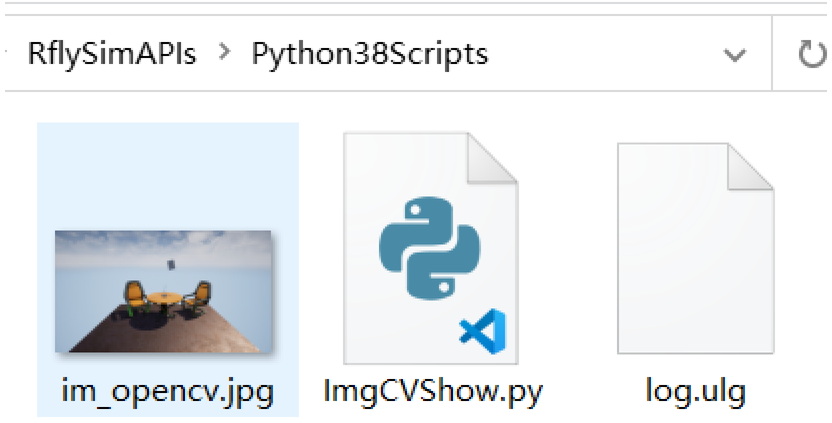
```
Python3.8 environment has been set with openCV+pymavlink+numpy+pyulog etc.
You can use pip or pip3 command to install other libraries
Put your python scripts 'XXX.py' into the folder 'C:\PX4PSPFull\RflySimAPIs\Python38Scripts'
Use the command: 'python XXX.py' to run the script with Python
For example, try entering 'python ImgCVShow.py' below to use OpenCV to read and show a image
You can also use pyulog (see https://github.com/PX4/pyulog) to convert PX4 log file
For example, try entering 'ulog2csv log.ulg' to convert ulg file to excel files for MATLAB

C:\PX4PSPFull\RflySimAPIs\Python38Scripts>
```

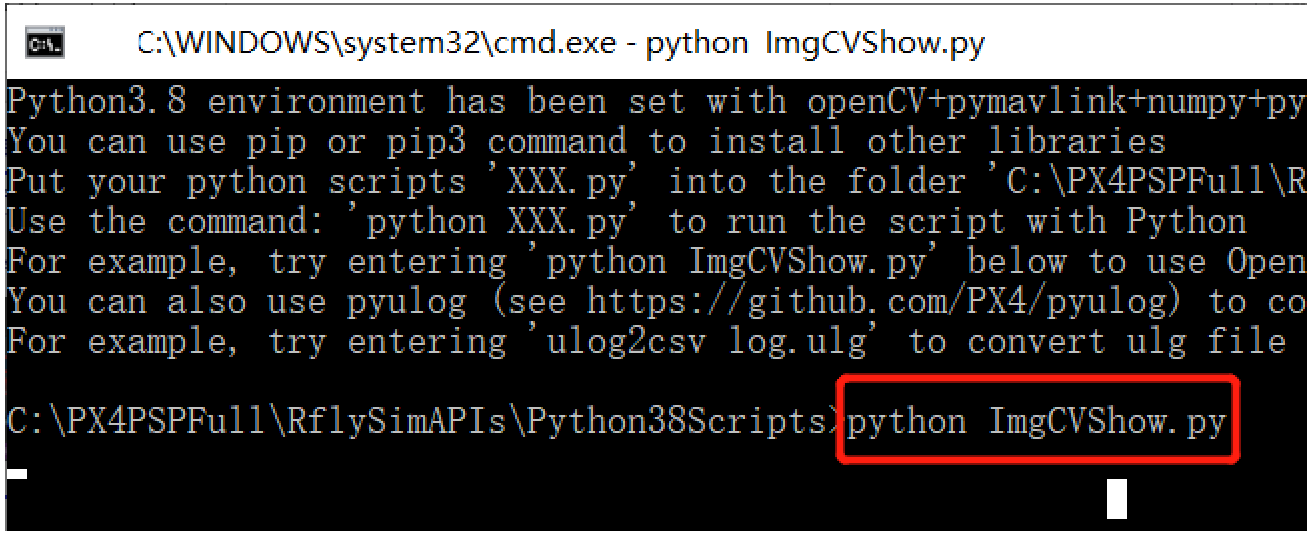
Input commands at here

1. Setup Instructions

1.7 Use of Python38Env

- The root directory of this Python environment is "**RflySimAPIs\Python38Scripts**", copy the Python script file *****.py** ending in **.py** to this folder, and then execute the command "**python ***.py**" to run the script
- For example: enter the command "**python ImgCVShow.py**" to run the python script and open an image like the example in **Section 1.4**.

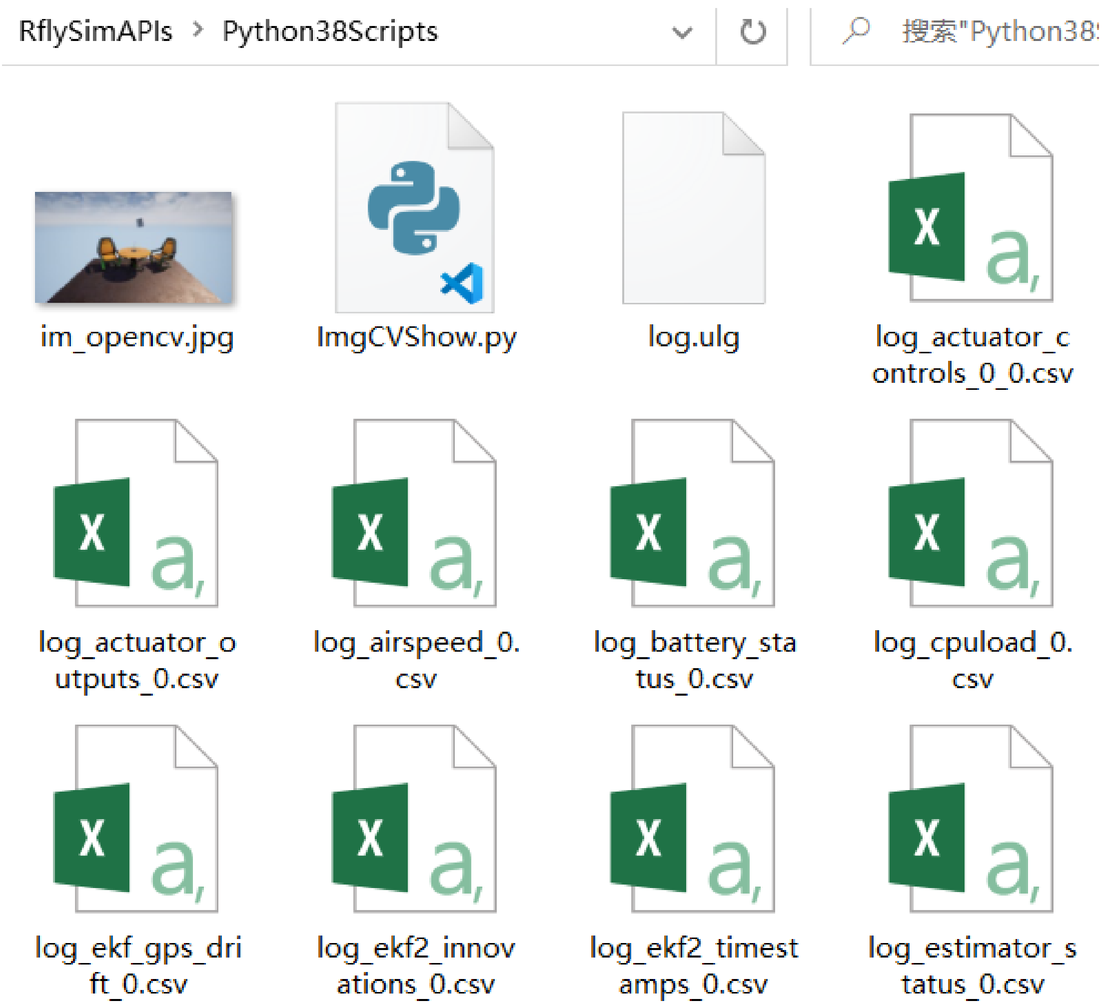




1. Setup Instructions

1.8 Python38Env environment ulog log reading

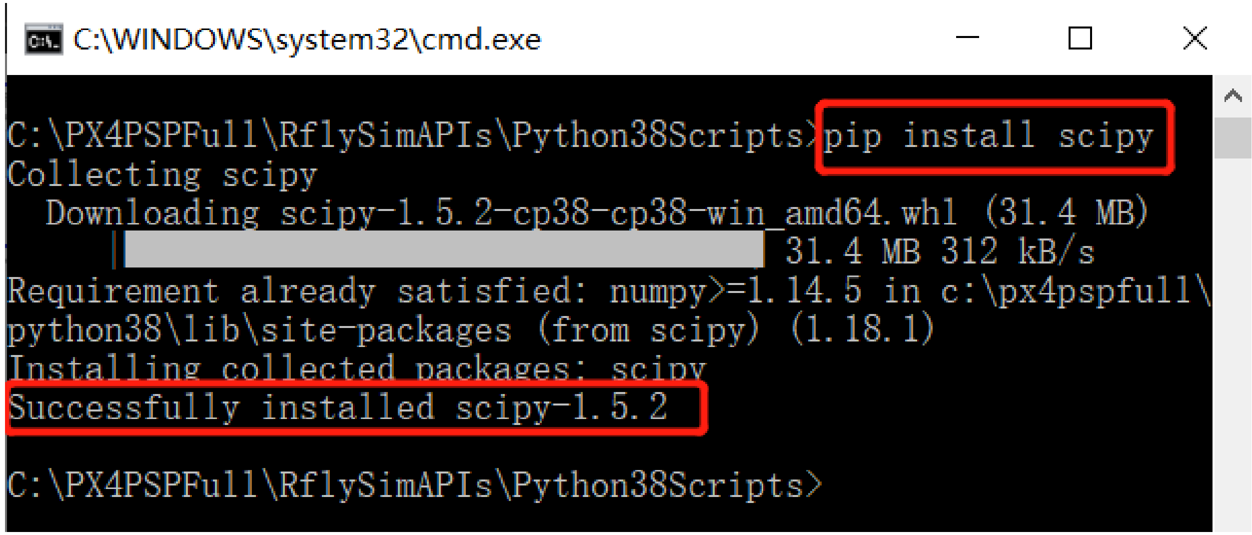
- This Python environment has installed pyulog, which can be used to parse the PX4 controller's **.ulg** format logs and generate excel files, which can be used for log data analysis.
- Since a "**log.ulg**" file has been stored in the "**RflySimAPIs\Python38Scripts**" folder, you can directly double-click "**RflySimAPIs\Python38Env.bat**" to open the Python environment, and enter "**ulog2csv log.ulg**" to get the log file
- These files can be opened with Excel, MATLAB or other statistical analysis software for flight data analysis.



1. Setup Instructions

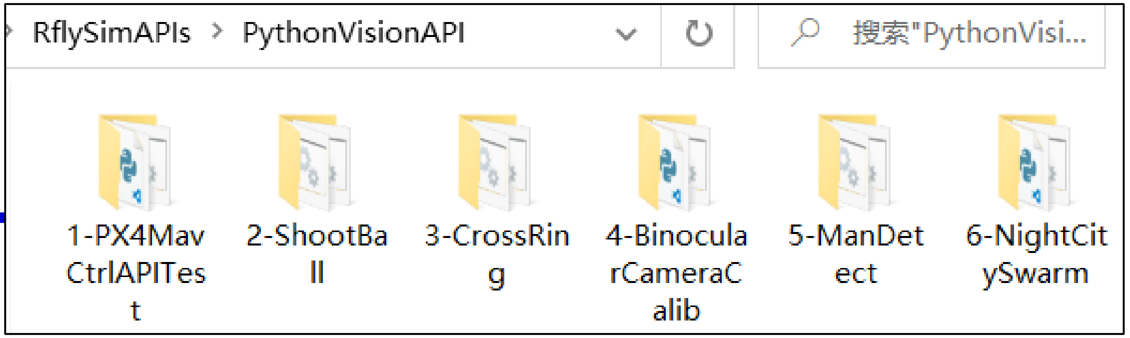
1.9 Python38Env function extension

- This Python environment has installed **OpenCV**, **Pymavlink**, **numpy**, **pyulog** and other module libraries, and supports the installation of other components.
- The installation of other components can be installed with the **pip install ***** command
- The current Python libraries commonly used include **Scipy, Pillow, Matplotlib, Panda**. Take the installation of Scipy as an example, just double-click the "**RflySimAPIs\Python38Env.bat**" file and enter "**pip install scipy**" in the command line to automatically download and complete the installation
- Users can test and install other modules/libraries by themselves



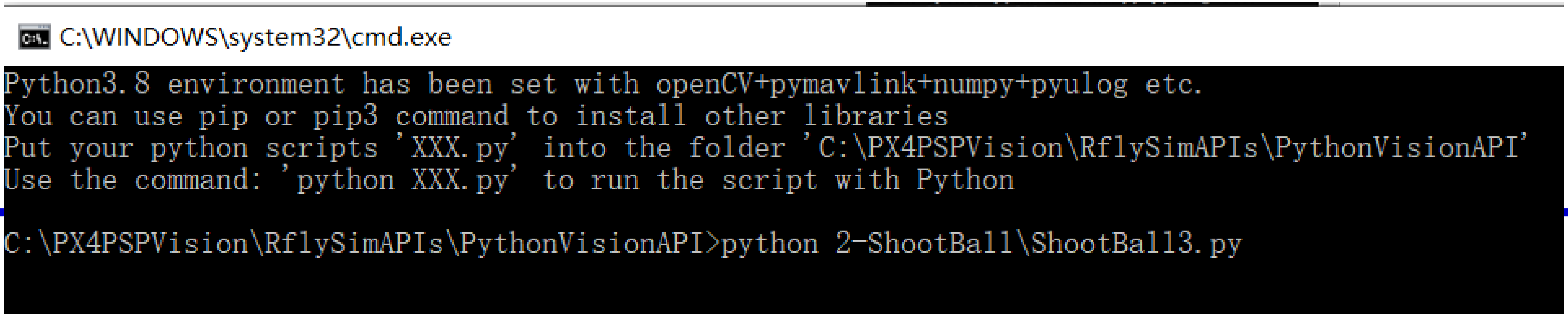
If the online installation is too slow (poor network or the installation package too large), you can go to https://pypi.org/ to search and download the installation package file in *****.whl** format offline, and copy the *****.whl** file to " **RflySimAPIs\Python38Scripts**" directory, and then use the command "pip install *****.whl"** to install offline

1. Setup Instructions



1.10 Operation of vision control demo

- The vision control demos of this course are stored in the "**RflySimAPIs\PythonVisionAPI**" path. These demos can be run using VS Code or **Python38Env**.
- Suppose you want to run the "**2-ShootBall\ShootBall3.py**" example script in the "**RflySimAPIs\PythonVisionAPI**" directory. The first method is to open the **ShootBall3.py** file with VS Code, and then run it directly according to the method in **Section 1.4**.
- The second method is to double-click to open the "**RflySimAPIs\PythonVisionAPI\Python38Run.bat**" file, and enter: "**python 2-ShootBall\ShootBall3.py**" in the pop-up window to run this script



1. Setup Instructions

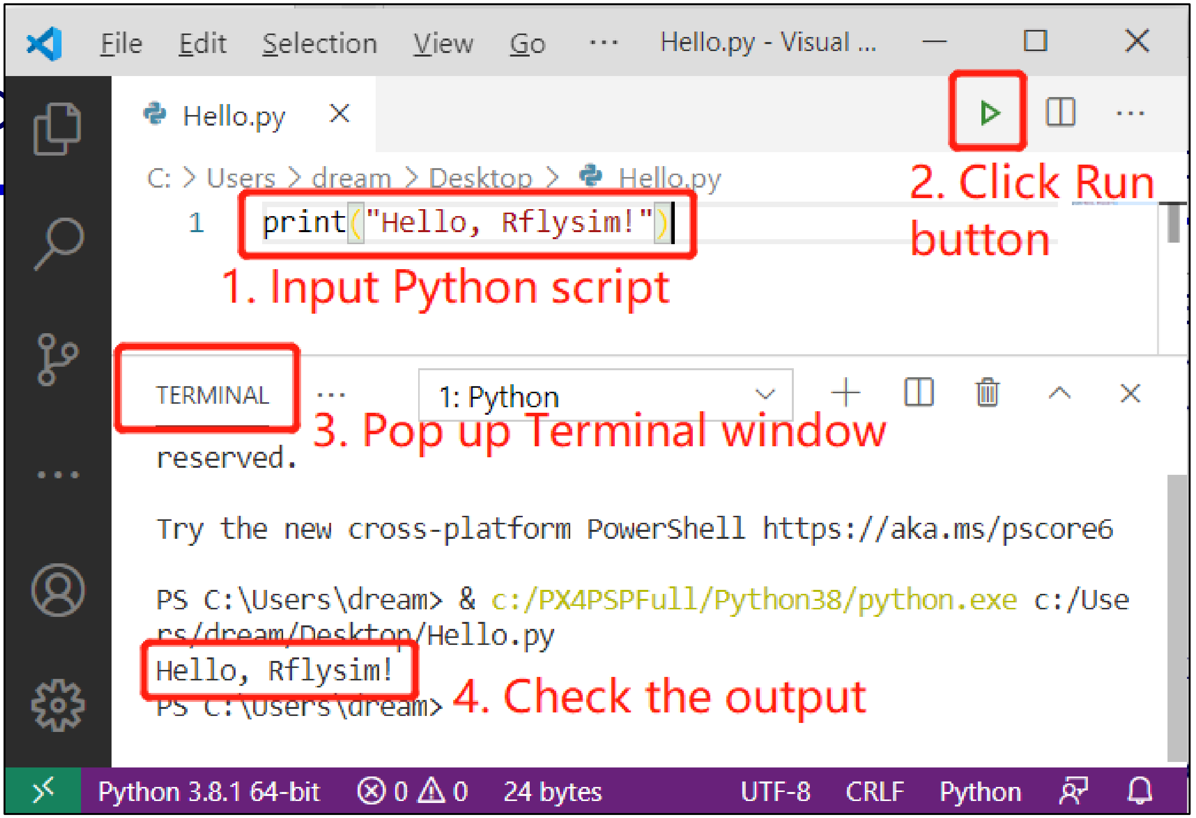
1.11 vision simulation difficulties and RflySim platform solutions

- Question 1: How to achieve high-speed image capture?
  - Solution: Don't take pictures inside the UE4 program (Airsim uses this method, and the image acquisition will cause serious frame drop). Use Python/C/Simulink to directly read RflySim3D images, reducing intermediate links, and 720P multi-window image reading consumes time Within 5ms (above 200Hz), and will not interfere with UE4 rendering efficiency
- Question 2: How to achieve multi-lens camera acquisition?
  - Solution: Support to open any RflySim3D window, and each window can be independently configured to display the viewing angle (airborne camera or ground observation viewing angle and etc.)
- Question 3: How to configure the resolution, camera position, and which vehicle to take pictures?
  - Solution: It supports adjustment via keyboard shortcuts, and also supports sending commands via UDP to control the viewing angle display parameters of RflySim3D.
- Question 4: How to ensure that the simulation results can be directly used in the real vehicle?
  - Solution: The bottom layer of the control interface we provide can directly send and receive MAVLink data. Since the cross-platform Python language is used, it can be used directly by copying the airborne computer. (Follow-up will provide Simulink vision interface to support code generation)

Content

1. Setup Instructions
2. Use of basic interface
3. Examples of monocular vision control
4. Examples of binocular vision control
5. Summary

2. Use of basic interfac

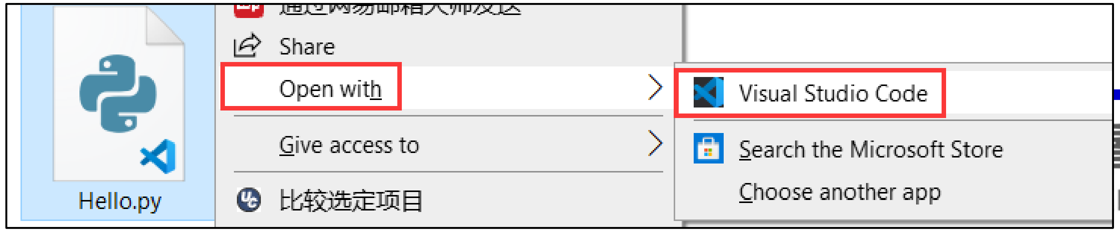


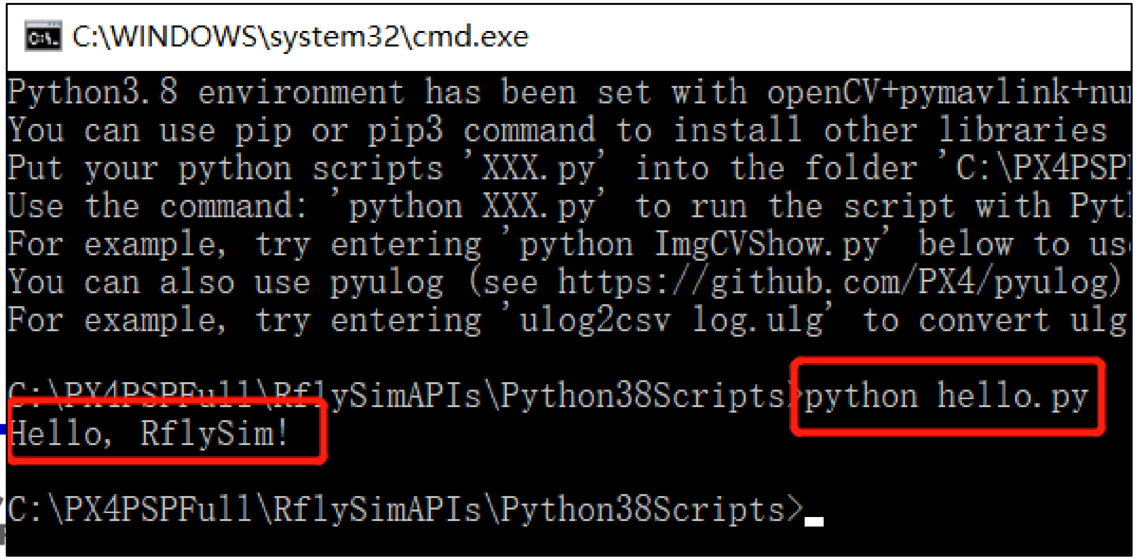
2.1 The simplest Python3 file running test

- In the "**RflySimAPIs\Python38Scripts**" 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, check the result as shown on the right
- In the same way, double-click the **Python38Env** desktop shortcut or "**RflySimAPIs\Python38Env.bat**" to view the running results as shown below.

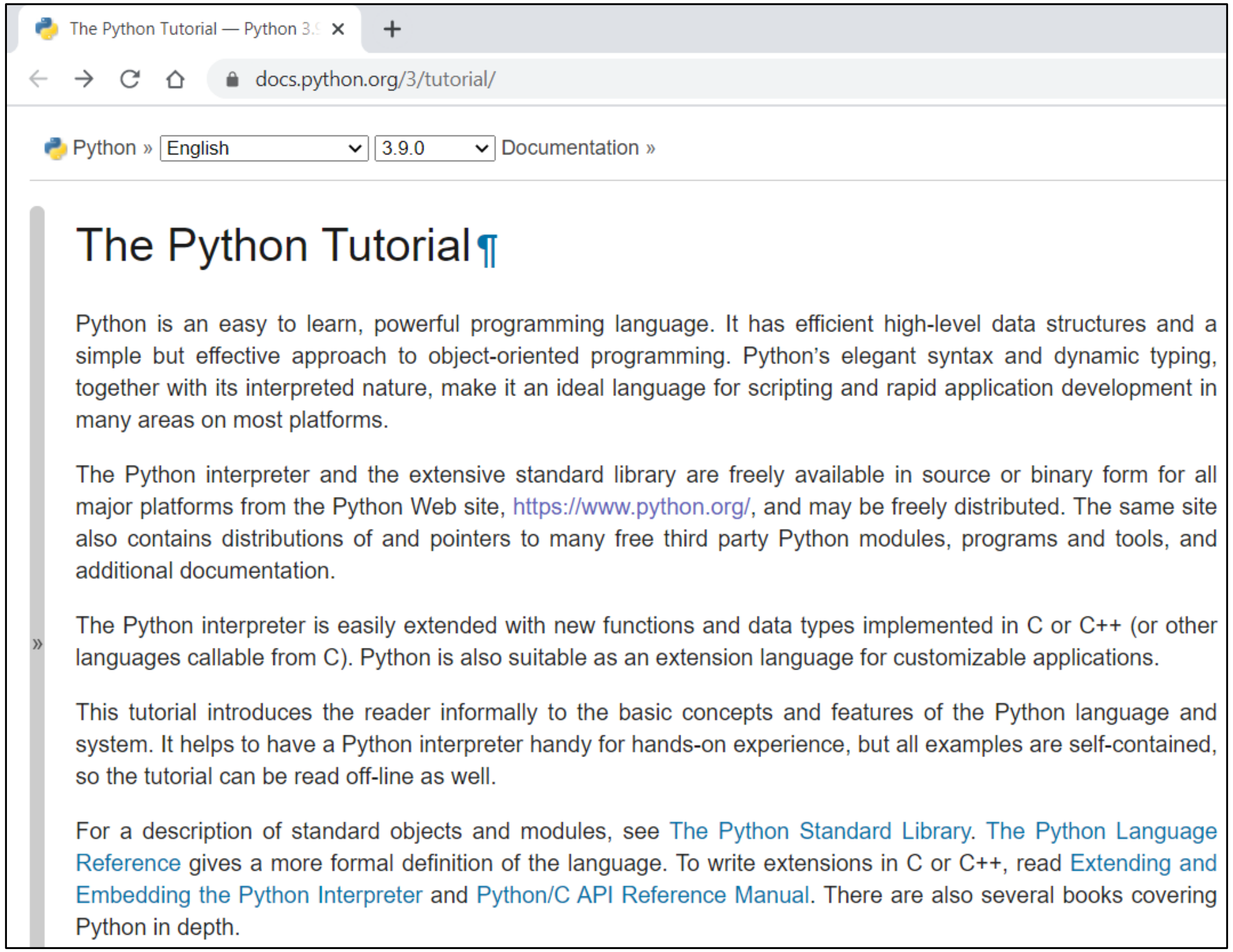




2. Use of basic interface

2.2 Basic Python3 syntax learning

- Visit: [https: //docs.python.org/3/tutorial/](https://docs.python.org/3/tutorial/) to understand the basic knowledge and programming method of learning Python3 (similar to MATLAB language) easy to get started
- **Note**: Python2 has stopped updating, it is recommended that you learn Python3 directly for development
- Python3 uses UTF-8 encoding by default, so non-english characters are natively supported
- Python uses indentation to distinguish code levels, so you must pay attention to indentation when writing code (Guides, indent-rainbow and other plugins can be installed in VS Code to assist)


The Python Tutorial — Python 3.9
docs.python.org/3/tutorial/

Python » English 3.9.0 Documentation »

The Python Tutorial ¶

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

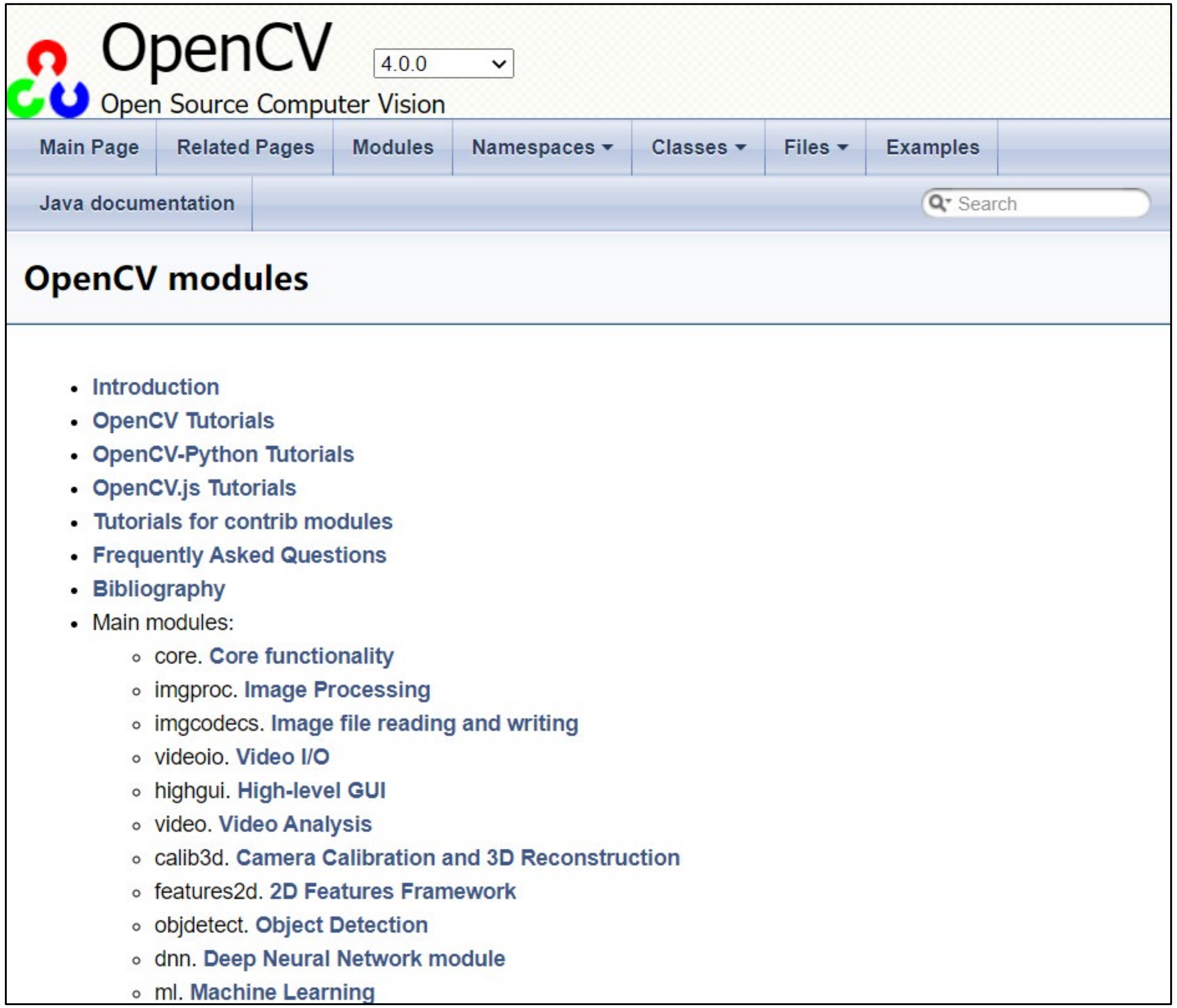
This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

For a description of standard objects and modules, see The Python Standard Library. The Python Language Reference gives a more formal definition of the language. To write extensions in C or C++, read Extending and Embedding the Python Interpreter and Python/C API Reference Manual. There are also several books covering Python in depth.

2. Use of basic interface

2.3 Basic Python-OpenCV learning

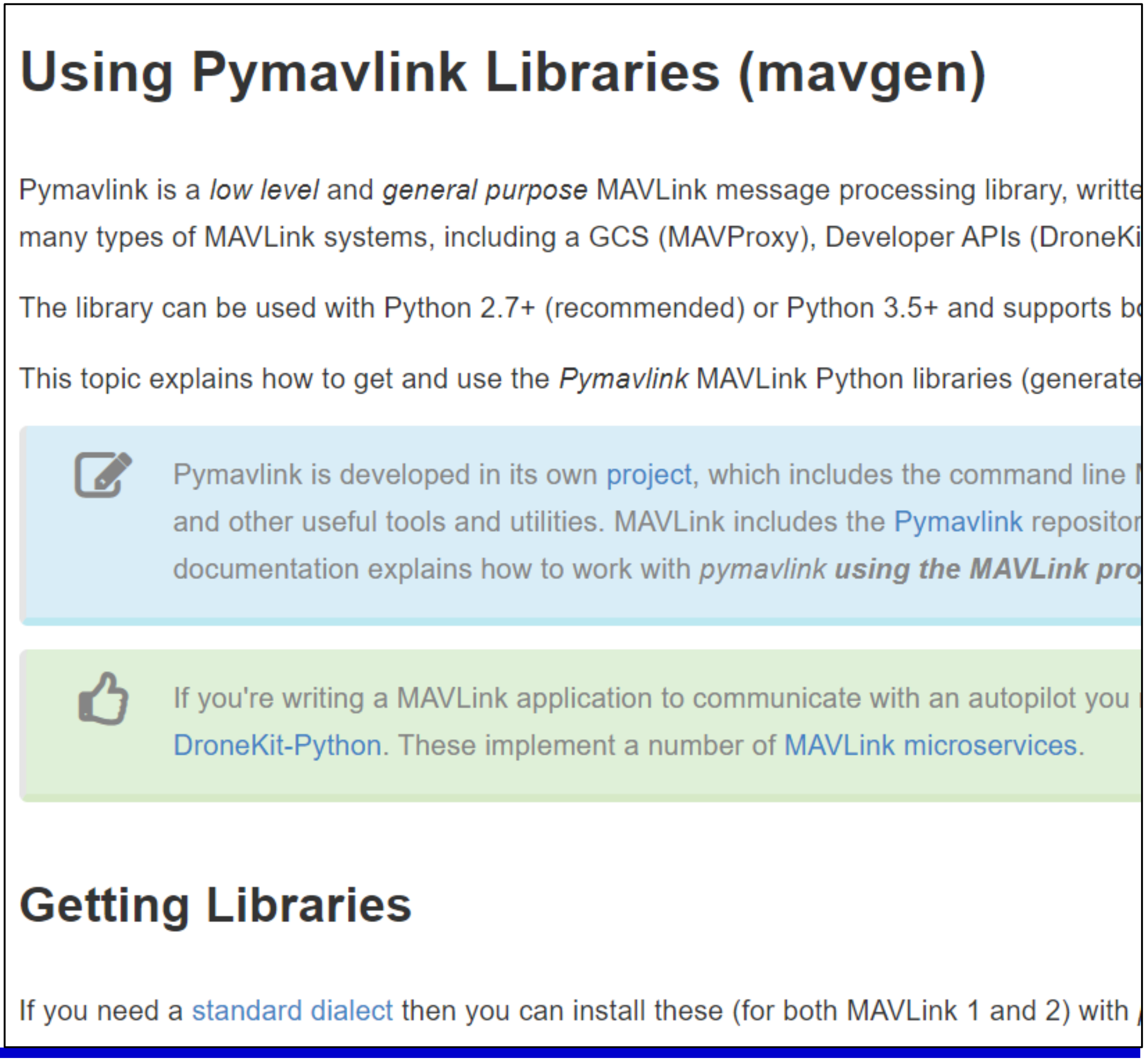
- **OpenCV** is an open source cross-platform computer vision and machine learning software library, widely used
- Official tutorial URL: https://docs.opencv.org/4.0.0/



2. Use of basic interface

2.4 Pymavlink learning

- **Pymavlink** is the Python version of the MAVLink communication protocol. It is currently pre-installed in the platform Python environment. It can easily communicate with the real Pixhawk hardware through serial port, UDP, TCP and etc., for top-level control
- The official document URL is as follows: https: //mavlink.io/en/mavgen_python. Please learn its detailed use instructions by yourself
- **Note**: Pymavlink is just a convenient library function. If you have higher customize requirements, you need to learn how to use the MAVLink protocol



Using Pymavlink Libraries (mavgen)

Pymavlink is a *low level* and *general purpose* MAVLink message processing library, writte
many types of MAVLink systems, including a GCS (MAVProxy), Developer APIs (DroneKi

The library can be used with Python 2.7+ (recommended) or Python 3.5+ and supports bo

This topic explains how to get and use the *Pymavlink* MAVLink Python libraries (generate

Pymavlink is developed in its own project, which includes the command line
and other useful tools and utilities. MAVLink includes the Pymavlink repositor
documentation explains how to work with *pymavlink* ***using the MAVLink pro***

If you're writing a MAVLink application to communicate with an autopilot you
DroneKit-Python. These implement a number of MAVLink microservices.

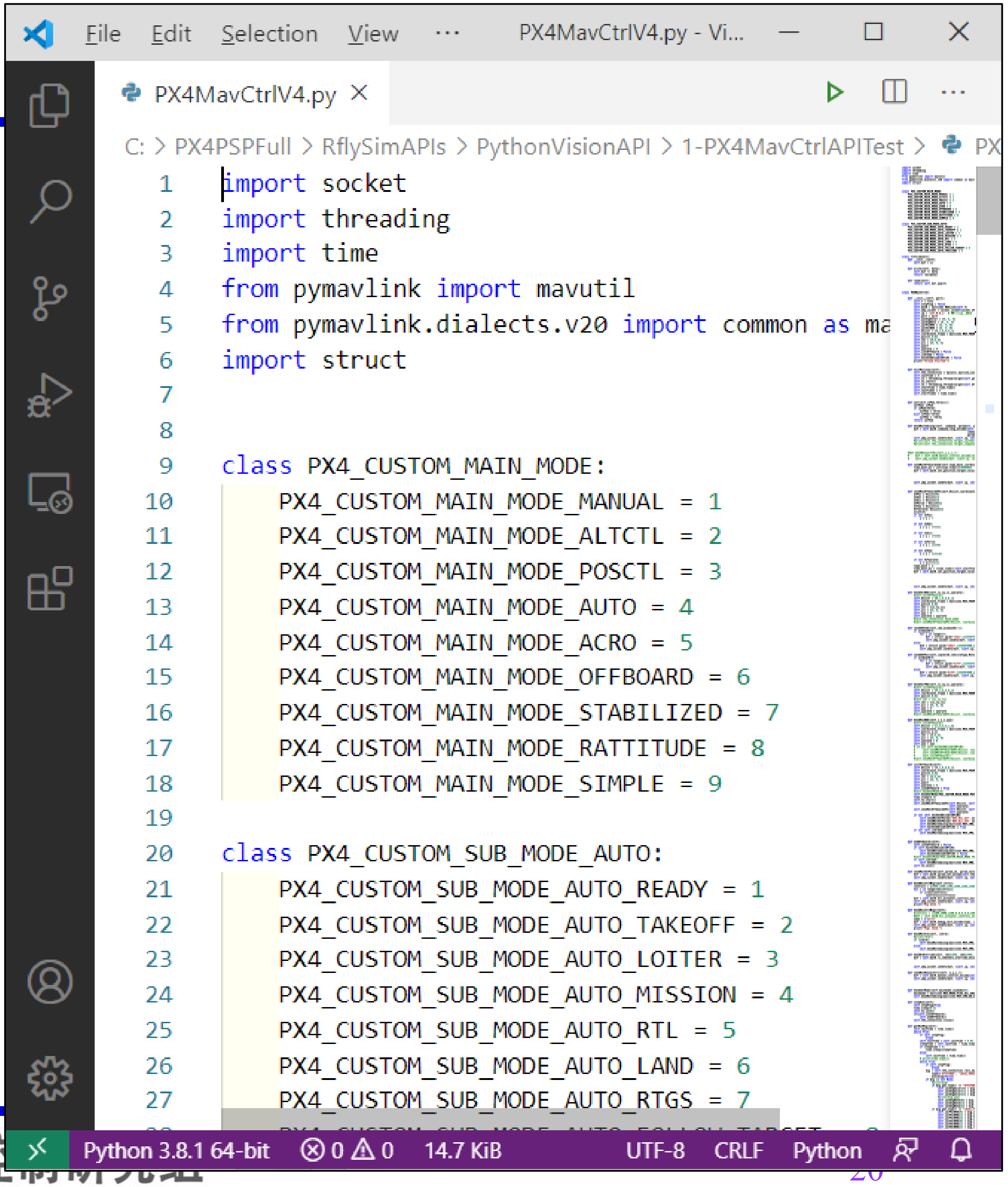
Getting Libraries

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

2. Use of basic interface

2.5 Introduction to the Python control interface in RflySim

- The "**RflySimAPIs\PythonVisionAPI\1-PX4MavCtrlAPITest\PX4MavCtrlV4.py**" file is the Python control interface file of the RflySim platform
- This interface includes interfaces for sending and receiving MAVLink messages, UE4 scene control, and Pixhawk Offboard control.
- The underlying data of this interface is in MAVLink format, which can be connected to the RflySim system for software/hardware-in-the-loop simulation, or can be connected to real Pixhawk hardware (via serial port or UDP network) for real vehicle control.



2. Use of basic interface

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

Pixhawk real-time state data from MAVLink

2.6 Internal principle of PX4MavCtrlV4

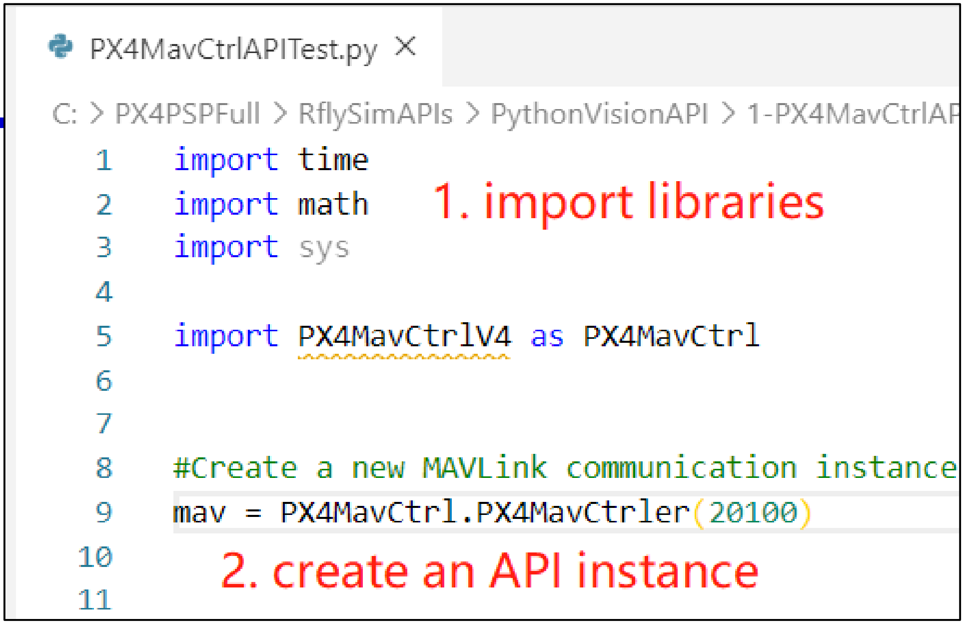
- **class PX4_CUSTOM_MAIN_MODE:** #PX4 main module enumerated variable, used to set the mode
- **class PX4_CUSTOM_SUB_MODE_AUTO:** #PX4 submodule enumeration variable
- **class PX4MavCtrler:** # RflySim's main communication interface class
- **def InitMavLoop:** #Enable MAVLink receiving thread, receive and update MAVLink messages at any time
- **def sat:** #A saturation function, used to control the limit of the variable
- **def SendMavCmdLong:** #Send the COMMAND_LONG message of the MAVLink message
- **def sendMavOffboardCmd:** #Send the Offboard command to the flight controller to enter the Offboard mode
- **def sendMavOffboardAPI:** # Update the data of Offboard message (the data will be sent at a certain frequency)
- **def SendVelNED:** # Send the earth coordinate system speed command
- **def sendUE4Cmd:** # Send a command to UE4 to control the display of UE4
- **def sendUE4Pos:** # Send the three-dimensional coordinates of an object to UE4 to display an object

2. Use of basic interface

2.6 Internal principle of PX4MavCtrlV4

- **def SendVelFRD:** #Send the body speed
- **def SendPosNED:** #Send the NED position, let the vehicle fly to the specified position (relative to the unlock point)
- **def initOffboard:** # Initialize Offboard mode
- **def endOffboard:** # End Offboard mode
- **def sendMavSetParam:** # Send MAVLink message to change Pixhawk parameters
- **def SendHILCtrlMsg:** #Send rfly_msg message to the flight controller (see **Section 4.3** of **Lesson 3**)
- **def SendMavArm:** #Send unlock command
- **def SendRcOverride:** #Send and simulate remote control signal
- **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

2. Use of basic interface



2.7 Interface usage example

- # "**1-PX4MavCtrlAPITest\PX4MavCtrlAPITest.py**" is an example of Python used in the interface.
- #The specific code analysis is as follows:
- #Create a new MAVLink communication instance, CopterSim's UDP receving port is 20100
- mav = PX4MavCtrl.PX4MavCtrler(20100)
- # **sendUE4Cmd:** RflySim3D API to modify scene display style
- # Format: **mav.sendUE4Cmd(cmd,windowID=-1)**, where **cmd** is a command string, **windowID** is the received window number (assuming multiple RflySim3D windows are opened at the same time), **windowID =-1** means sent to all windows
- # **RflyChangeMapbyName** command means to switch the map (scene), the following string is the map name, here will switch all open windows to the grass map
- mav.sendUE4Cmd(b'RflyChangeMapbyName Grasslands')

2. Use of basic interface

2.7 Interface usage example

- # **sendUE4Pos**: RflySim3D API to generate 3D objects and control position
- # Formart: **mav.sendUE4Pos(CopterID, VehicleType, RotorSpeed, PosM, AngEulerRad, windowsID=0)**
- mav.sendUE4Pos(100,30,0,[2.5,0,-8.086],[0,0,math.pi])
- # Send and generate a 3D object to RflySim3D, where: the vehicle ID is **CopterID=100**;
- # Vehicle type **VehicleType=30** (a man); **RotorSpeed=0**RPM; Position coordinate **PosM=[2.5,0,-8.086]**m
- # Vehicle attitude angle **AngEulerRad=[0,0,math.pi]**rad (rotate 180 degrees to face the vehicle), the receiving window number default **windowsID=-1** (sent to all open RflySim3D programs)
- # **VehicleType options**: **3** for quadcopters, **5/6** for hexacopters, **30** for persons, **40** for checkerboard grids, **50/51 for** cars, **60** for luminous lights, **100** for flying-wing or fixed-wing aircraft, **150/152** for circular square targets

- # command **RflyChange3DModel** followed by vehicle ID + 3D style to switch
- mav.sendUE4Cmd(b'RflyChange3DModel 100 12')
- #Send a message to make **CopterID=100** (the character just created) in all scenes, here **style=12** represents a walking person

2. Use of basic interface

2.7 Interface usage example

- # Command **RflyChangeViewKeyCmd** means to simulate the shortcut key pressed in RflySim3D, shortcut key **B 1** means to switch the focus to the object with **CopterID=1**
- # Here is set to send to window 0, other windows do not send
- mav.sendUE4Cmd(b'RflyChangeViewKeyCmd B 1',0)

- # Shortcut key **V 1** means to switch to the 1st onboard camera (front camera)
- mav.sendUE4Cmd(b'RflyChangeViewKeyCmd V 1',0)

- # **RflyCameraPosAng x y z roll pith yaw**
- # Set the position of the camera relative to the center of the body, the default is 0
- # Here set the position of the front camera to **[0.1 -0.25 0]**
- mav.sendUE4Cmd(b'RflyCameraPosAng 0.1 0 0',0)

- # **r.setres 720x405w** is a built-in command of UE4, which means to switch the resolution to 720x405
- mav.sendUE4Cmd(b'r.setres 720x405w',0)

2. Use of basic interface

2.7 Interface usage example

- # Send a shortcut command to window 1 to switch the focus to vehilce 1
- mav.sendUE4Cmd(b'RflyChangeViewKeyCmd B 1',1)

- # Send a shortcut key control command to window 0, N 1 shortcut key means to switch the perspective to the ground fixed perspective 1
- mav.sendUE4Cmd(b'RflyChangeViewKeyCmd N 1',1)
-
- # Set the current camera Field of View (FOV) to 90 degrees (the default value is 90 degrees in RflySim3D), the range of FOV is 0 to 180 degrees
- mav.sendUE4Cmd(b'RflyCameraFovDegrees 90',1)

- # Set the current camera position here as [-2 0 -9.7]
- mav.sendUE4Cmd(b'RflyCameraPosAng -2 0 -9.7',1)

- #Turn on MAVLink to monitor CopterSim data and update it in real time.
- 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 real-time state data from MAVLink

2. Use of basic interface

2.7 Interface usage example

- #Turn on Offboard mode
- mav.initOffboard()

- # Send the desired position signal, fly to the target point position 0,0, -1.7, the yaw angle is 0 rad
- mav.SendPosNED(0, 0, -1.7, 0)

- #Send arm command to arm the drone
- mav.SendMavArm(True)

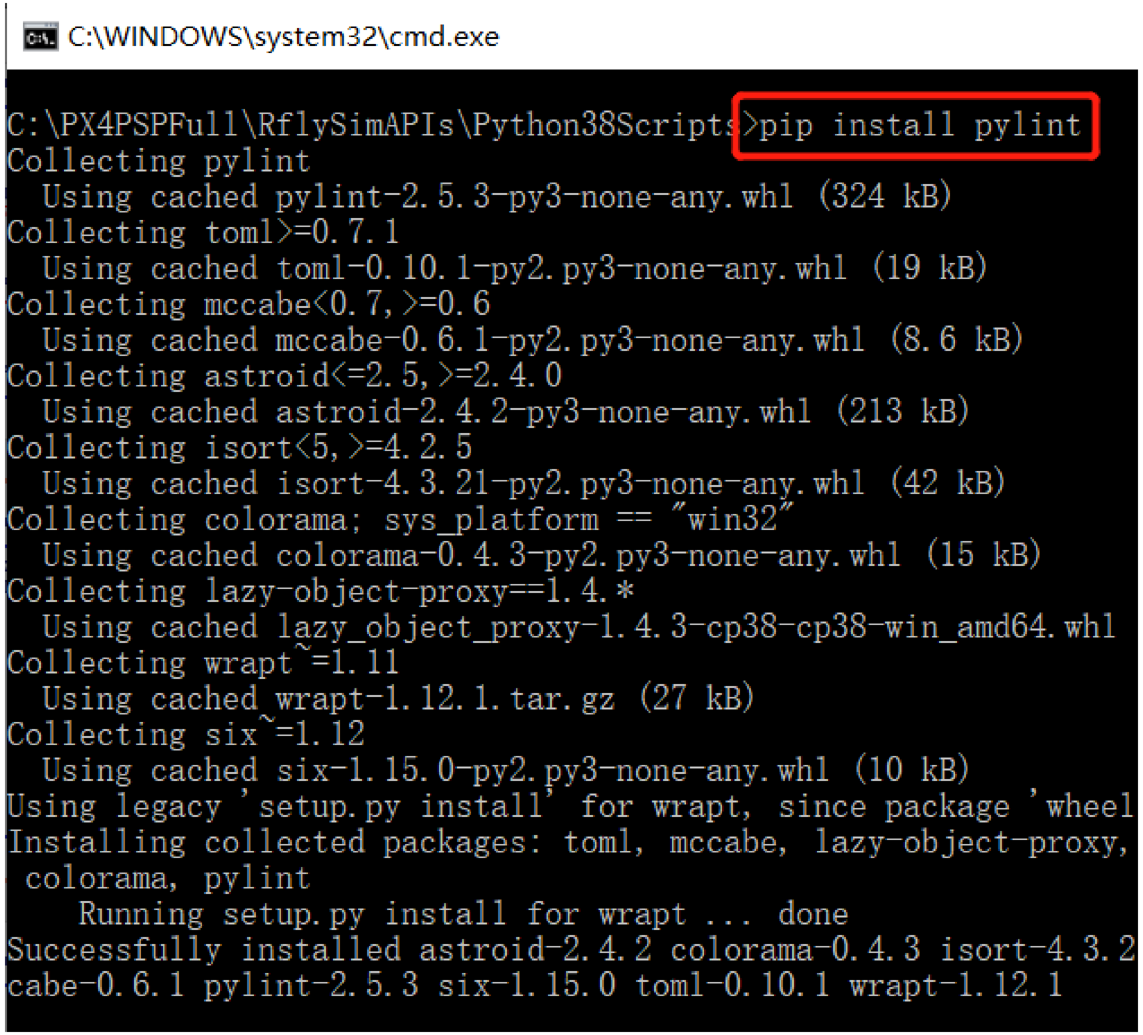
- # Send the desired speed signal, 0.2m/s downwards, the z-axis downward is positive
- mav.SendVelNED(0, 0, 0.2, 0)
- #Exit Offboard control mode
- mav.endOffboard()

- #Exit MAVLink data receiving mode
- mav.stopRun()

2. Use of basic interface

2.8 Enable VS Code Python debugging function

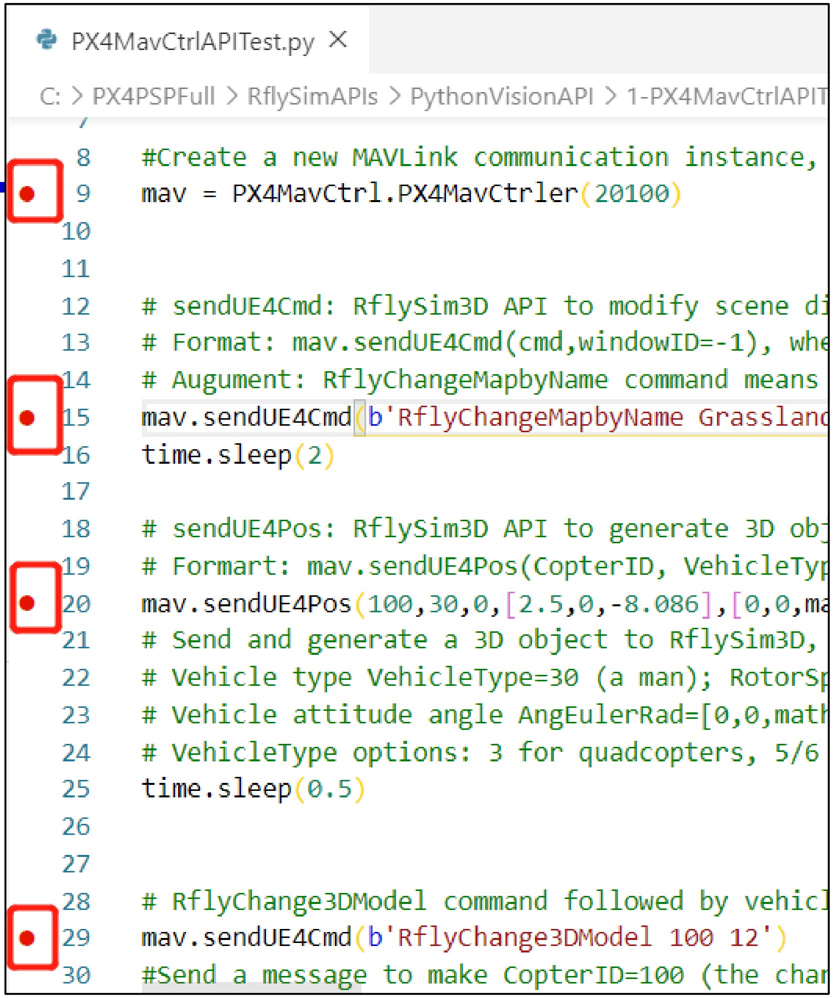
- In order to facilitate the observation of the running results of each line of Python code, the debugging function of Python needs to be turned on here. The configuration process is as follows:
- Double-click "**RflySimAPIs\Python38Env.bat**" to open the Python environment, and enter the command "**pip install pylint**" to install the Python checker **pylint**. The result of successful installation is shown on the right
- Close the window and start code debugging with VS Code

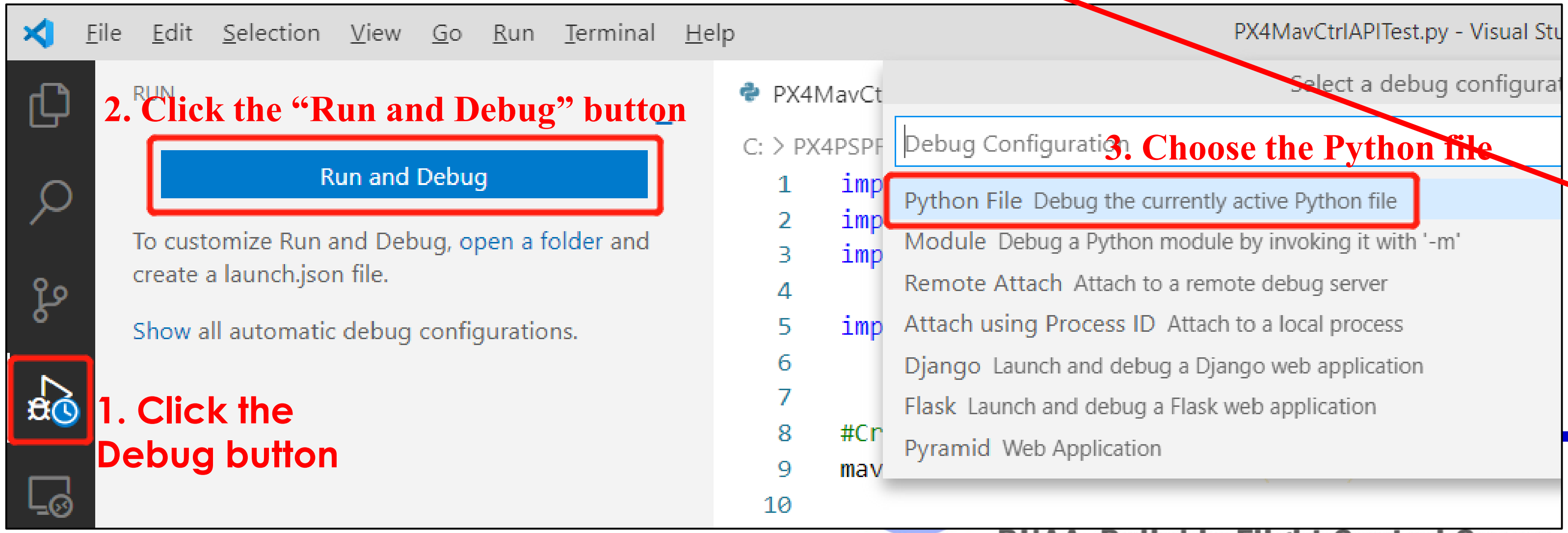


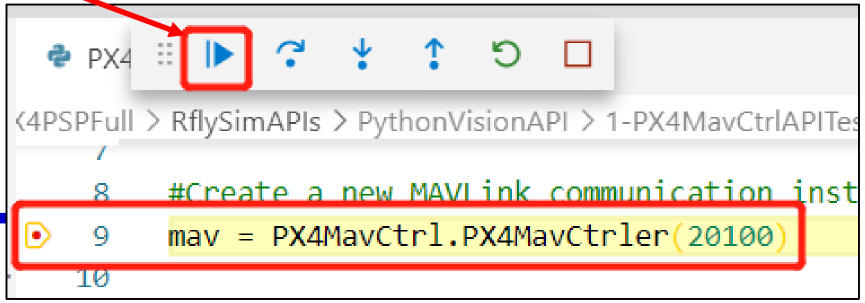
2. Use of basic interface

2.9 Interface usage example

- Locate the "**RflySimAPIs\Python VisionAPI\1-PX4MavCtrlAPITest**" folder in Windows Explorer
- Double-click the "**PX4MavCtrlAPITest.bat**" script to open the PX4 SITL simulation system of one vehicle
- Open the "**PX4MavCtrlAPITest.py**" file with VS Code, click the **breakpoints (red dot)** in front of each key statement as shown in the right picture, and turn on the debugging mode as shown in the picture below. Click the **arrow button** in the lower right picture to execute the statements in sequence



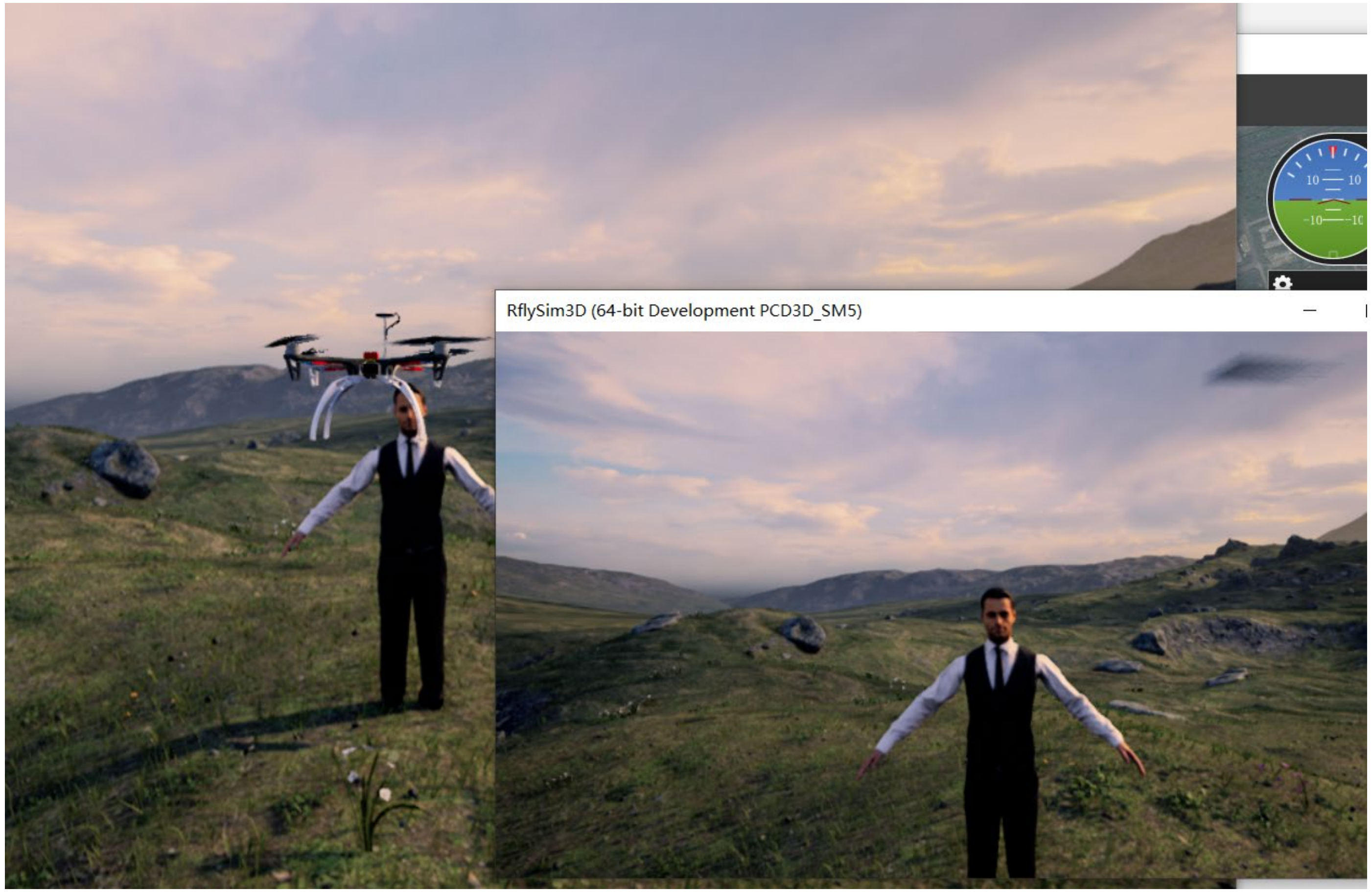




2. Use of basic interface

2.10 Experimental results of interface examples

- The phenomenon of this demo is that the python program sends a series of instructions, a new target of a walking person is created in the RflySim3D program, the angle of view form, size, and position are set, and control instructions are sent to the simulated drone to make it take off and land. .
- As shown on the right, this example will open two RflySim3D windows, one is the front camera, the other is the observation from God

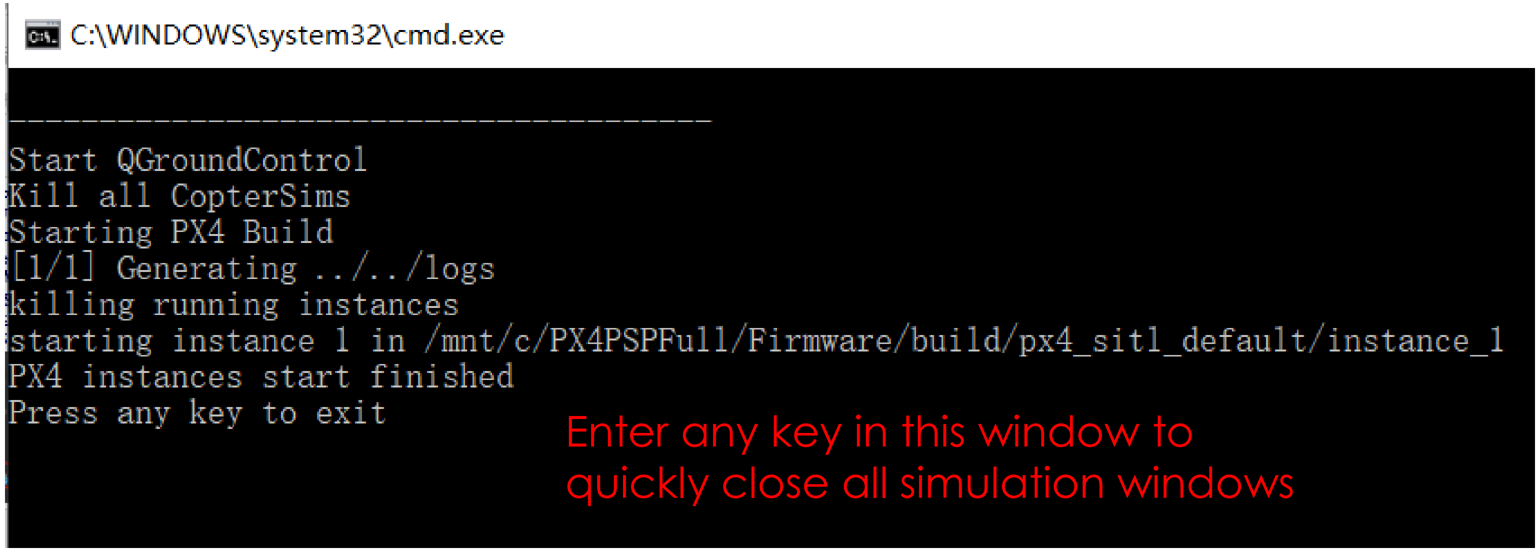


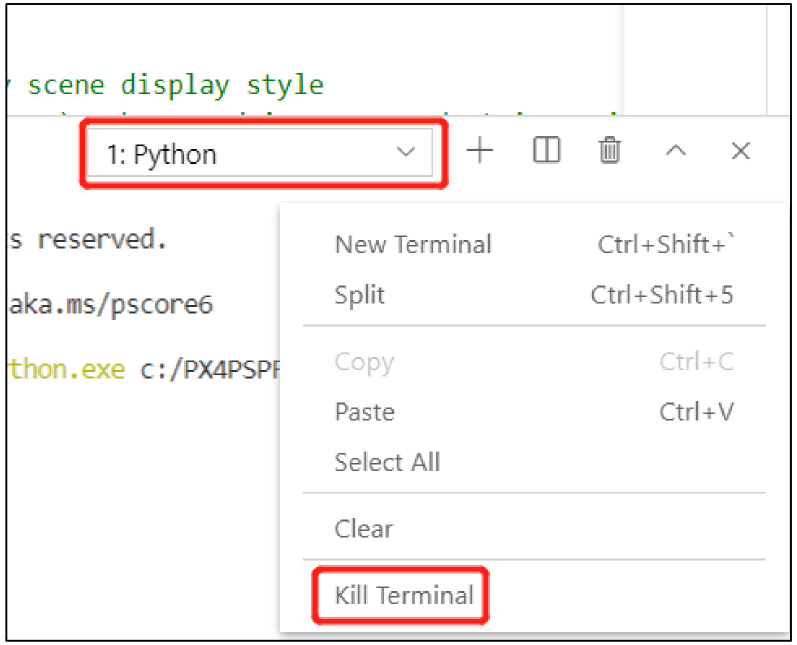
Note: If the computer performance is poor and the flight shakes, you can manually close the opened RflySim3D window (observation angle)

2. Use of basic interface

2.10 End simulation of interface example

- In the command prompt 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**, **RflySim3D**, etc.
- As shown in the figure on the right, follow the steps in **Section 1.5** of this lesson in VS Code and click "**Kill Terminal**" to exit the script
- Close the VS Code window opened by "**PX4MavCtrlAPITest.py**"





2. Use of basic interface

2.11 Python interface to capture image from RflySim3D

- This script use the **win32api** of Windows to obtain the handles of all RflySim3D windows, and capture image from specific window for vison-based control.

ScreenCapApiV4.py ×

C: > PX4PSP > RflySimAPIs > PythonVisionAPI > 2-ShootBall > ScreenCapApiV4.py

```python
# import all libraries
import win32gui, win32ui, win32con, win32api
from ctypes import windll
import numpy

# get all RflySim3D window handles with class name UnrealWindow
def window_enumeration_handler(hwnd, window_hwnds):
    if win32gui.GetClassName(hwnd) == "UnrealWindow":
        window_hwnds.append(hwnd)

def getWndHandls():
    window_hwnds = []
    win32gui.EnumWindows(window_enumeration_handler, window_hwnds)
    return window_hwnds
```

```python
# define a class to store information of window handles
class WinInfo:
    def __init__(self, hWnd, width, height, saveDC, saveBitMap, mf
        self.hWnd = hWnd
        self.width = width
        self.height = height
        self.saveDC = saveDC
        self.saveBitMap = saveBitMap
        self.mfcDC = mfcDC
        self.hWndDC = hWndDC

# get the window infomation of a handle
def getHwndInfo(hWnd):
    left, top, right, bot = win32gui.GetClientRect(hWnd)
    width = right - left
    height = bot - top
    print((width,height))
    if hWnd and width == 0 and height == 0:
        print("The RflySim3D window cannot be in minimized mode")
        sys.exit(1)

    # retrieve the device context (DC) for the entire window,
    # including title bar, menus, and scroll bars.
    hWndDC = win32gui.GetWindowDC(hWnd)
    mfcDC = win32ui.CreateDCFromHandle(hWndDC)

    # creates a memory device context (DC) compatible with the spe
    saveDC = mfcDC.CreateCompatibleDC()

    # create a bitmap object
    saveBitMap = win32ui.CreateBitmap()
    saveBitMap.CreateCompatibleBitmap(mfcDC, width, height)

    # return image info.
    info = WinInfo(hWnd, width, height, saveDC, saveBitMap, mfcDC,
    return info
```

2. Use of basic interface

2.11 Python interface to capture screen from RflySim3D

- The following image presents the most important interface to acquire real-time frame for OpenCV image processing and then MAVLink control

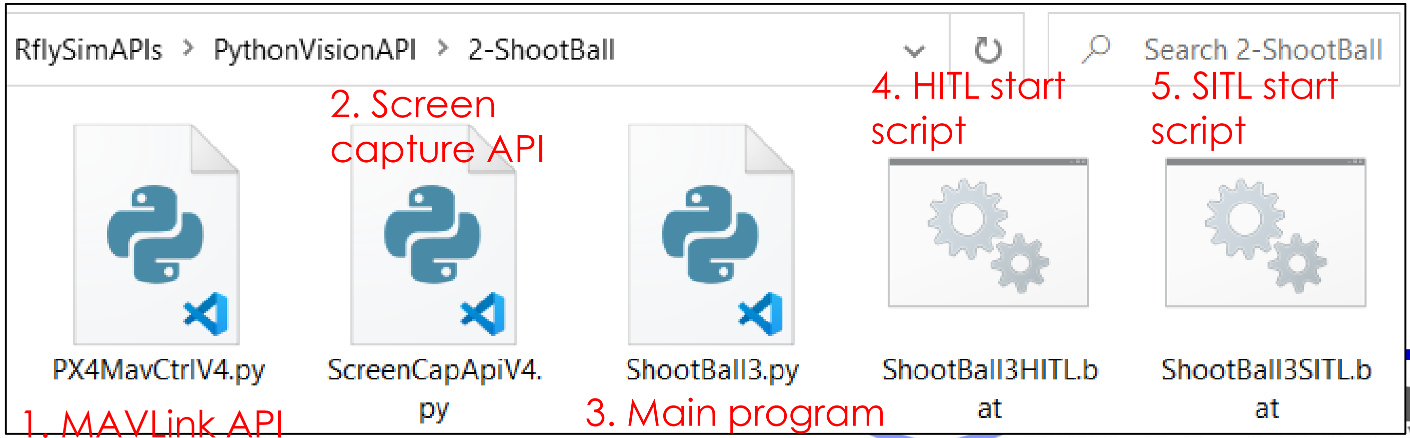
```python

# get the image from RflySim3D window's client area
def getCVImg(wInfo):
    wInfo.saveDC.SelectObject(wInfo.saveBitMap)

    #  copies a visual window into the specified device context (DC)
    # The last int value: 0-save the whole window, 1-only client area
    result = windll.user32.PrintWindow(wInfo.hWnd, wInfo.saveDC.GetSafeHdc(), 1)
    signedIntsArray = wInfo.saveBitMap.GetBitmapBits(True)

    # get the image from bitmap array
    im_opencv = numpy.frombuffer(signedIntsArray, dtype='uint8')
    im_opencv.shape = (wInfo.height, wInfo.width, 4)

    # return the image
    return im_opencv

```

Content

3. Examples of monocular vision control

3.1 Experiment of drone impact on small ball

- In Windows Explorer, open and enter the "**RflySimAPIs\PythonVisionAPI\2-ShootBall**" folder, the contents of which are as shown in the figure below
- Among them, "**PX4MavCtrlV4.py**" is the interface file introduced in the previous section, "**ShootBall3.py**" is the main Python program of this demo, "**ScreenCapApiV4.py**" is the API to capture screen from RlfySim3D, "**ShootBall3HITL.bat**" is the script to quickly start the hardware-in-the-loop simulation, "**ShootBall3SITL.bat**" It is a script to quickly start software-in-the-loop simulation. As shown in the figure on the right, the difference between the desktop **S/HITLRun** shortcut and the scripts is: "**UE4_MAP**" map scene variable selects the vision flat grass scene "**VisionRingBlank**"; secondly, "**UDPSIMMODE**" communication UDP mode Select the "**Mavlink_Full**" mode to facilitate communication with Python; finally open two RflySim3D windows




```
REM Set broadcast to other computer; 0: only this computer,
REM e.g., IS_BROADCAST=0 equals to IS_BROADCAST=127.0.0.1,
SET IS_BROADCAST=0

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

3. Examples of monocular vision control

3.2 Analysis of Impact Ball Experiment Code

- Open the "**ShootBall3.py**" file with VS Code, the algorithm to acquire front camera image from RflySim3D and compute velocity commands to the Pixhawk are presented as follows.

ShootBall3.py ×

C: > PX4PSP > RflySimAPIs > PythonVisionAPI > 2-ShootBall > ShootBall3.py

```python
# import required libraries
import time
import mmap
import numpy as np
import cv2.cv2 as cv2
from pymavlink.dialects.v20 import common as mavlink2
import win32gui, win32ui, win32con
from ctypes import windll
import sys

# import RflySim APIs
import PX4MavCtrlV4 as PX4MavCtrl
import ScreenCapApiV4 as sca

```

1. Import all libraries

2. Import MAVLink and Vision APIs

```python
def calc_centroid(img):
    """Get the centroid and area of green in the image"""
    low_range = np.array([0,0,80])
    high_range = np.array([100,100,255])
    th = cv2.inRange(img, low_range, high_range)
    dilated = cv2.dilate(th, cv2.getStructuringElement(
        cv2.MORPH_ELLIPSE, (3, 3)), iterations=2)
    cv2.imshow("dilated", dilated)
    cv2.waitKey(1)

    M = cv2.moments(dilated, binaryImage=True)
    if M["m00"] >= min_prop*width*height:
        cx = int(M["m10"] / M["m00"])
        cy = int(M["m01"] / M["m00"])
        return [cx, cy, M["m00"]]
    else:
        return [-1, -1, -1]
```

3. Vision function to calculate the location and radius of red ball

3. Examples of monocular vision control

3.2 Analysis of Impact Ball Experiment Code

4. Controller function to obtain vehicle velocity error based on pixel error

```
# Function to obtain velocity commands for Pixhawk
# according to the image processing results
def controller(p_i):
    # if the object is not in the image, search in clockwise
    if p_i[0] < 0 or p_i[1] < 0:
        return [0, 0, 0, 1]

    # found
    ex = p_i[0] - width / 2
    ey = p_i[1] - height / 2

    vx = 2 if p_i[2] < max_prop*width*height else 0
    vy = 0
    vz = K_z * ey
    yawrate = K_yawrate * ex

    # return forward, rightward, downward, and rightward-yaw
    # velocity control sigals
    return [vx, vy, vz, yawrate]
```

5. Image processing function to process image and obtain velocity control signals for Pixhawk

```
# Process image to obtain vehicle velocity control command
def procssImage():
    img_rgba=sca.getCVImg(ImgInfo1)
    img_bgr = cv2.cvtColor(img_rgba, cv2.COLOR_RGBA2RGB)
    img_bgr = cv2.resize(img_bgr, (width, height))
    p_i = calc_centroid(img_bgr)
    ctrl = controller(p_i)
    return ctrl

# saturation function to limit the maximum velocity
def sat(inPwm,thres=1):
    outPwm= inPwm
    for i in range(len(inPwm)):
        if inPwm[i]>thres:
            outPwm[i] = thres
        elif inPwm[i]<-thres:
            outPwm[i] = -thres
    return outPwm
```

6. Saturation function to limit maximum velocity output for safety

3. Examples of monocular vision control

3.2 Analysis of Impact Ball Experiment Code

```python
# Create MAVLink control API instance
mav = PX4MavCtrl.PX4MavCtrler(20100)
# Init MAVLink data receiving loop
mav.InitMavLoop()

isEmptyData = False
lastTime = time.time()
startTime = time.time()
# time interval of the timer
timeInterval = 0.01 #here is 0.01s (100Hz)
flag = 0

# parameters
width = 720
height = 405
channel = 4
min_prop = 0.000001
max_prop = 0.3
K_z = 0.003 * 640 / height
K_yawrate = 0.005 * 480 / width
```

7. Initialize MAVLink connection and all variables

```python
# send command to all RflySim3D windows to switch to the blank grass scene
# by default, there are two windows, the first is front camera
# for vison control, and the second window for observation
mav.sendUE4Cmd(b'RflyChangeMapbyName VisionRingBlank')
time.sleep(1)

# create a ball, set its position and altitude, use the default red color
mav.sendUE4Pos(100,152,0,[3,0,-2],[0,0,0])
time.sleep(0.5)

# Change the target vehicle to copterID=1's vehicle
mav.sendUE4Cmd(b'RflyChangeViewKeyCmd B 1',0)
time.sleep(0.5)
# Switch its viewpoint to oboard #1 (front camera view)
mav.sendUE4Cmd(b'RflyChangeViewKeyCmd V 1',0)
time.sleep(0.5)
# move the camera to the position [0.3,0,0.05] related to the body
mav.sendUE4Cmd(b'RflyCameraPosAng 0.3 0 0.05 0 0 0',0)
time.sleep(0.5)
# set the RflySim3D window size to 720x405
mav.sendUE4Cmd(b'r.setres 720x405w',0)
time.sleep(2)
```

8. Send commands to change the display image in RflySim3D

3. Examples of monocular vision control

3.2 Analysis of Impact Ball Experiment Cod

```python
window_hwnds = sca.getWndHandls()
Wd01 = window_hwnds[0]
hasFoundWd = False
for hwd in window_hwnds:
    info = sca.getHwndInfo(hwd)
    if info.width == 720:
        Wd01 = hwd
        hasFoundWd = True
        window_hwnds.remove(hwd)
        break

# if no window is found, throw an error
if not hasFoundWd:
    print("The first RflySim3D window does not response the command,
    mav.stopRun()
    sys.exit(1)
else:
    print("The first RflySim3D window is found with desired size.")

ImgInfo1 = sca.getHwndInfo(Wd01)

ctrlLast = [0,0,0,0]
while True:
    lastTime = lastTime + timeInterval
    sleepTime = lastTime - time.time()
    if sleepTime > 0:
        time.sleep(sleepTime)
    else:
        lastTime = time.time()
    # The following code will be executed 100Hz (0.01s)
```

9. Acquire the RflySim3D front camera view for image processing

10. Generate a timer with 0.01s interval (100Hz)

```python
    if time.time() - startTime > 5 and flag == 0:
        # The following code will be executed at 5s
        print("5s, Arm the drone")
        mav.initOffboard()
        flag = 1
        mav.SendMavArm(True) # Arm the drone
        print("Arm the drone!, and fly to NED 0,0,-5")
        mav.SendPosNED(0, 0, -5, 0) # Fly to target position [0, 0, -5], i.e.

    if time.time() - startTime > 15 and flag == 1:
        flag = 2
        # The following code will be executed at 15s
        mav.SendPosNED(-30,-5, -5, 0)  # Fly to target position [-30,-5, -5]
        print("15s, fly to pos: -30,-5, -5")

    if time.time() - startTime > 25 and flag == 2:
        flag = 3
        print("25s, start to shoot the ball.")

    if time.time() - startTime > 25 and flag == 3:
        ctrlNow = procssImage()
        ctrl = sat(ctrlNow,5)
        # add a inertial component here to restrain the speed variation rate
        if ctrl[0]-ctrlLast[0] > 0.5:
            ctrl[0]=ctrlLast[0]+0.05
        elif ctrl[0]-ctrlLast[0] < -0.5:
            ctrl[0]=ctrlLast[0]-0.05
        if ctrl[1]-ctrlLast[1] > 0.5:
            ctrl[1]=ctrlLast[1]+0.05
        elif ctrl[1]-ctrlLast[1] < -0.5:
            ctrl[1]=ctrlLast[1]-0.05
        ctrlLast = ctrl
        # if control signals is obtained, send to Pixhawk
        if not isEmptyData:
            mav.SendVelFRD(ctrl[0], ctrl[1], ctrl[2], ctrl[3])
```

11. Arm the drone and fly to 5m above ground

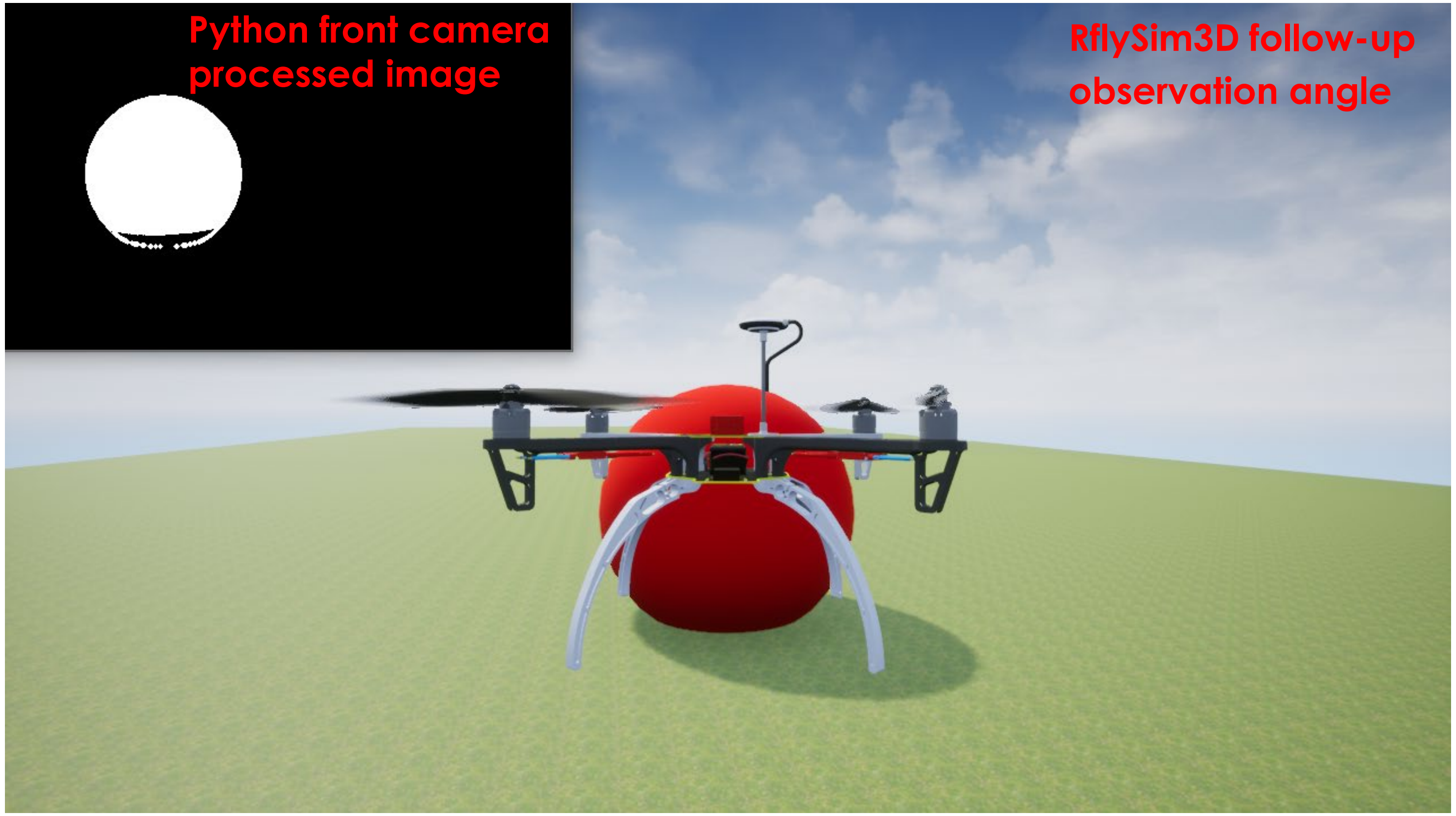
12. Fly to -30, -5m behind the red ball

13. Process image and obtain velocity commands to approach the ball

3. Examples of monocular vision control

3.3 The running effect of the impact ball experiment

- Double-click to run the "**ShootBall3SITL.bat**" file to start the software-in-the-loop simulation system, and then run the "**ShootBall3.py**" program. Generate a red sphere in the front, let the drone fly a distance to the left and rear, and turn on vision tracking, and fly close to the small ball then stop.
- To use hardware-in-the-loop simulation, after setting up the flight controller, run the "**ShootBall3HITL.bat**" script and enter the flight controller serial port number to start the hardware-in-the-loop simulation system.



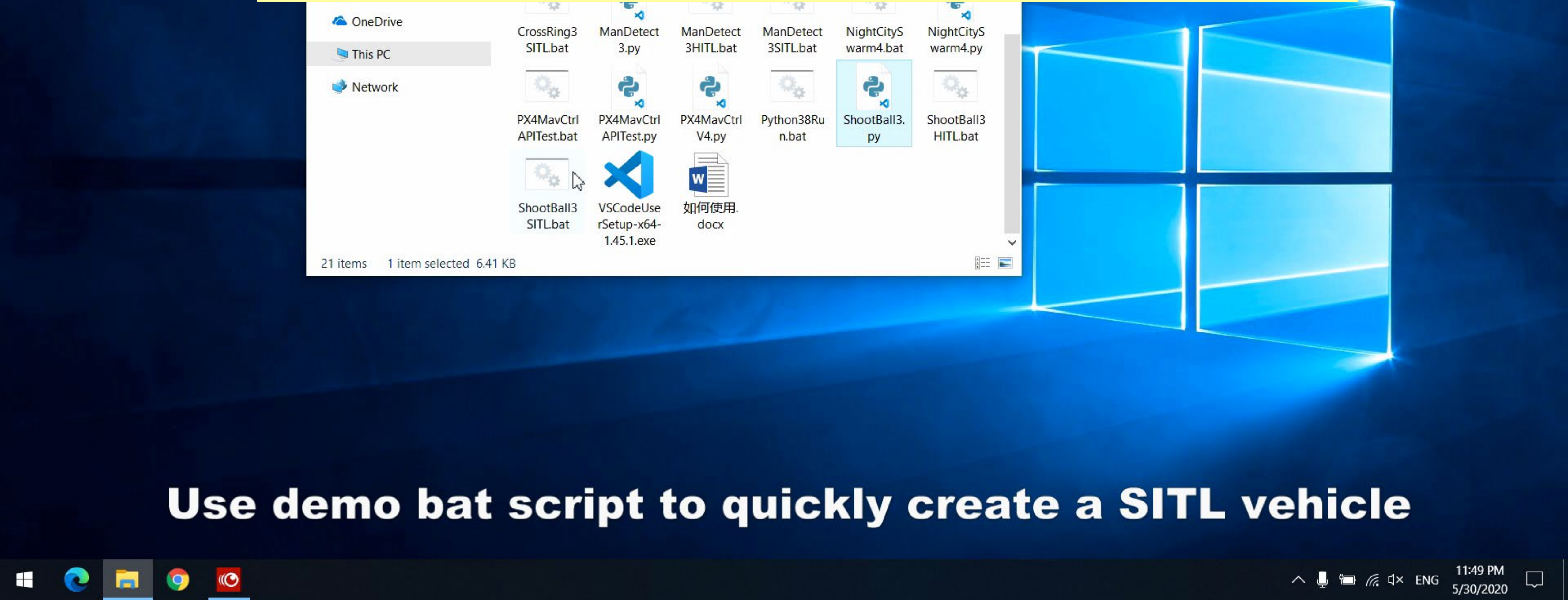
Note: If the computer performance is poor and the flight shakes, you can manually close the opened RflySim3D window (trailing observation angle)

RflySim: How to use Python/OpenCV to perform vision-based control of a multicopter UAV to track a ball

Watch this video by clicking the following links:

YouTube: https: //youtu.be/PvxEfY7oMq4

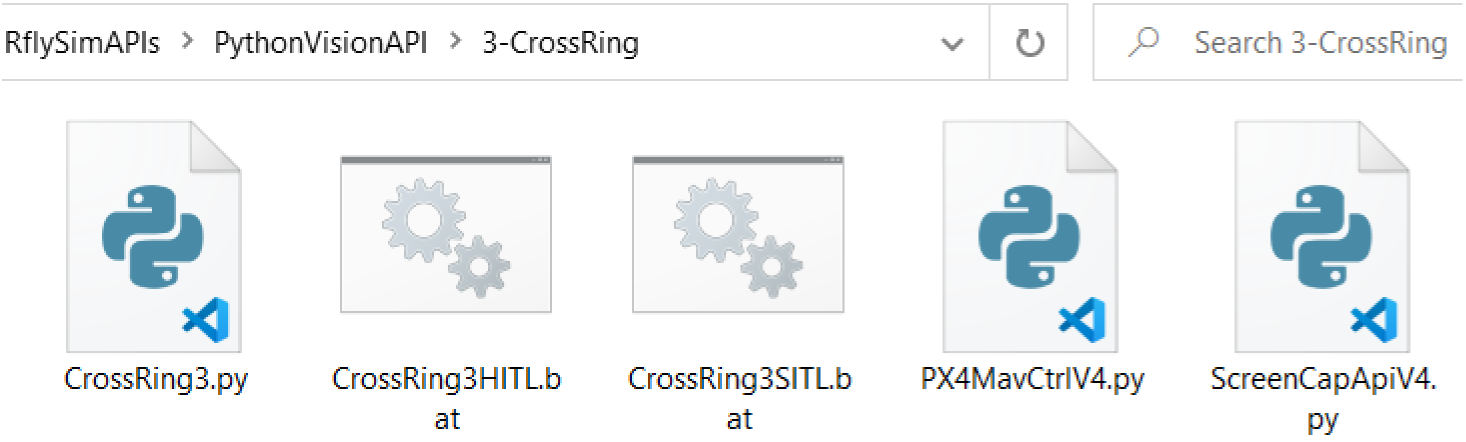
Youku: https: //v.youku.com/v_show/id_XNDcwNjA4NTYwNA==.html



3. Examples of monocular vision control

3.4 UAV cross rings experiment

- In Windows Explorer, open and enter the "**RflySimAPIs\PythonVisionAPI\3-CrossRing**" folder, the contents of which are as shown in the figure below
- Among them, "**CrossRing3.py**" is the main Python program of this demo; "**ShootBall3HITL.bat**" and "**ShootBall3SITL.bat**" are different from the previous example of hitting a small ball: "**UE4_MAP**" map scene variable is selected for "**VisionRing**" in the vision looping scene. Note that the "**UDPSIMMODE**" communication UDP mode also selects the "**Mavlink_Full**" mode.



3. Examples of monoc

3.5 Code analysis of cross rings experi

```python
def saturationYawRate(yaw_rate):
    yr_bound = 20.0
    if yaw_rate > yr_bound:
        yaw_rate = yr_bound
    if yaw_rate < -yr_bound:
        yaw_rate = -yr_bound
    return yaw_rate

def taskChange(pos_x):
    if pos_x < 40:
        task = "range1"
    elif pos_x <70:
        task = "range2"
    elif pos_x < 130:
        task = "range3"
    elif pos_x < 140:
        task = "land"
    else:
        task = "finish"
    return task
```

1. Saturation function for yaw rate

2. Task switching function based on flying distance

```python
# object detect function
def objectDetect(task):
    """According task to detect objects"""
    img_rgba=sca.getCVImg(ImgInfo1)
    img_bgr = cv2.cvtColor(img_rgba, cv2.COLOR_RGBA2RGB)
    img_bgr = cv2.resize(img_bgr, (width, height))

    b,g,r = cv2.split(img_bgr)
    img_edge = cv2.Canny(b, 50, 100)
    if task == "range1" or task == "range2":
        return circleDetect(img_bgr, img_edge, b)
    else:
        return squareDetect(img_bgr, img_edge)
```

3. Object detection function

```python
# square detect for object detect function
def squareDetect(img_bgr, img_edge):
    """Detect Square with PolyDP and diagonal length"""
    # find contours
    squares = []
    cnts, hierarchy = cv2.findContours(img_edge, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    for cnt in cnts:
        cnt_len = cv2.arcLength(cnt, True)
        cnt = cv2.approxPolyDP(cnt, 0.05 * cnt_len, True)
        if len(cnt) == 4 and cv2.contourArea(cnt) > 2000 and cv2.isContourConvex(cnt):
            cnt = cnt.reshape(-1, 2)
            diag_delta = diagonal_check(cnt)
            if diag_delta < 0.2:
                squares.append(cnt)
    cv2.drawContours( img_bgr, squares, -1, (0, 255, 0), 3)
    cv2.imshow("img_bgr", img_bgr)
    cv2.waitKey(1)
    height, width, channel = img_bgr.shape
    if squares:
        return (squares[0][0][0]+squares[0][2][0])/2 - width/2, (squares[0][0][1]+squares[0][
    else:
        return -1,-1,-1
```

4. Square detection function

3. Examples of monocular vision control

3.5 Code analysis of cross rings experiment

5. Circle detection function

```python
# circle detect for object detect function
def circleDetect(img_bgr, img_edge, img_b):
    """Hough Circle detect"""
    circles = cv2.HoughCircles(img_b, cv2.HOUGH_GRADIENT, 1, 20, pa
    if circles is not None:
        circles = np.uint16(np.around(circles))
        obj = circles[0, 0]
        cv2.circle(img_bgr, (obj[0], obj[1]), obj[2], (0,255,0), 2)
        cv2.circle(img_bgr, (obj[0], obj[1]), 2, (0,255,255), 3)
        cv2.imshow("img_bgr", img_bgr)
        cv2.waitKey(1)
        height, width, channel = img_bgr.shape
        return obj[0]-width/2, obj[1]-height/2, obj[2]
    else:
        return -1,-1,-1
```

6. Target approaching function

```python
# approaching Objective/ crossing rings algorithm
def approachObjective():
    # 0. parameters
    # some parameters that work:(0.03, -0.03, 1, 5.0); (0.06, -
    K_z = 0.004 * 640 / height
    K_yawrate = 0.005 * 480 / width
    task = "range1"
    # 1. start
    startAppTime= time.time()
    while (task != "finish") & (task != "land"):
        # 1.1. detect object and get error with center of image
        ex, ey, r = objectDetect(task)
        # 1.2. where is the drone
        pos_x = mav.uavPosNED[0]
        #print(time.time())
        # 1.3. update task
        task = taskChange(pos_x)
        # 1.4. attack
        if ex != -1:
            vx = sat(time.time()-startAppTime,3)
            vy = 0
            vz = K_z * ey
            yawrate = K_yawrate * ex
            mav.SendVelFRD(vx, vy, vz, yawrate)
    while task == "land":
        pos_x = mav.uavPosNED[0]
        task = taskChange(pos_x)
        mav.SendVelFRD(0, 0, 1, 0)
```

3. Examples of monocular vision control

3.5 Code analysis of cross rings experiment

7. Main function to initialize MAVLink and configure RflySim3D

```python
# main function
if __name__ == '__main__':
    mav = PX4MavCtrl.PX4MavCtrler(20100)
    mav.InitMavLoop()
    mav.sendUE4Cmd(b'RflyChangeMapbyName VisionRing',0)
    mav.sendUE4Cmd(b'RflyChangeMapbyName VisionRing',1)
    time.sleep(0.5)
    mav.sendUE4Cmd(b'RflyChangeViewKeyCmd V 1',0)
    time.sleep(0.5)
    mav.sendUE4Cmd(b'RflyCameraPosAng 0.3 0 0 0 0 0',0)

    # Get handles of all UE4/RflySim3D windows
    window_hwnds = sca.getWndHandls()

    mav.sendUE4Cmd(b'r.setres 720x405w',0)
    time.sleep(2)
```

8. Check RflySim3D display OK

```python
    # if no window is found, throw an error
    if not hasFoundWd:
        print("The first RflySim3D window does not response the command,
        mav.stopRun()
        sys.exit(1)
    else:
        print("The first RflySim3D window is found with desired size.")


    ImgInfo1 = sca.getHwndInfo(Wd01)

    print("Simulation Start.")

    print("Enter Offboard mode.")
    time.sleep(5)
    mav.initOffboard()
    time.sleep(0.5)
    mav.SendMavArm(True) # Arm the drone
    mav.SendPosNED(0, 0, -5, 0) # Fly to target position 0,0, -5

    time.sleep(10)
    # start crossing ring task
    approachObjective()
```

9. Takeoff at 5s and start crossing rings control at 15s

10. Enter vision control loop

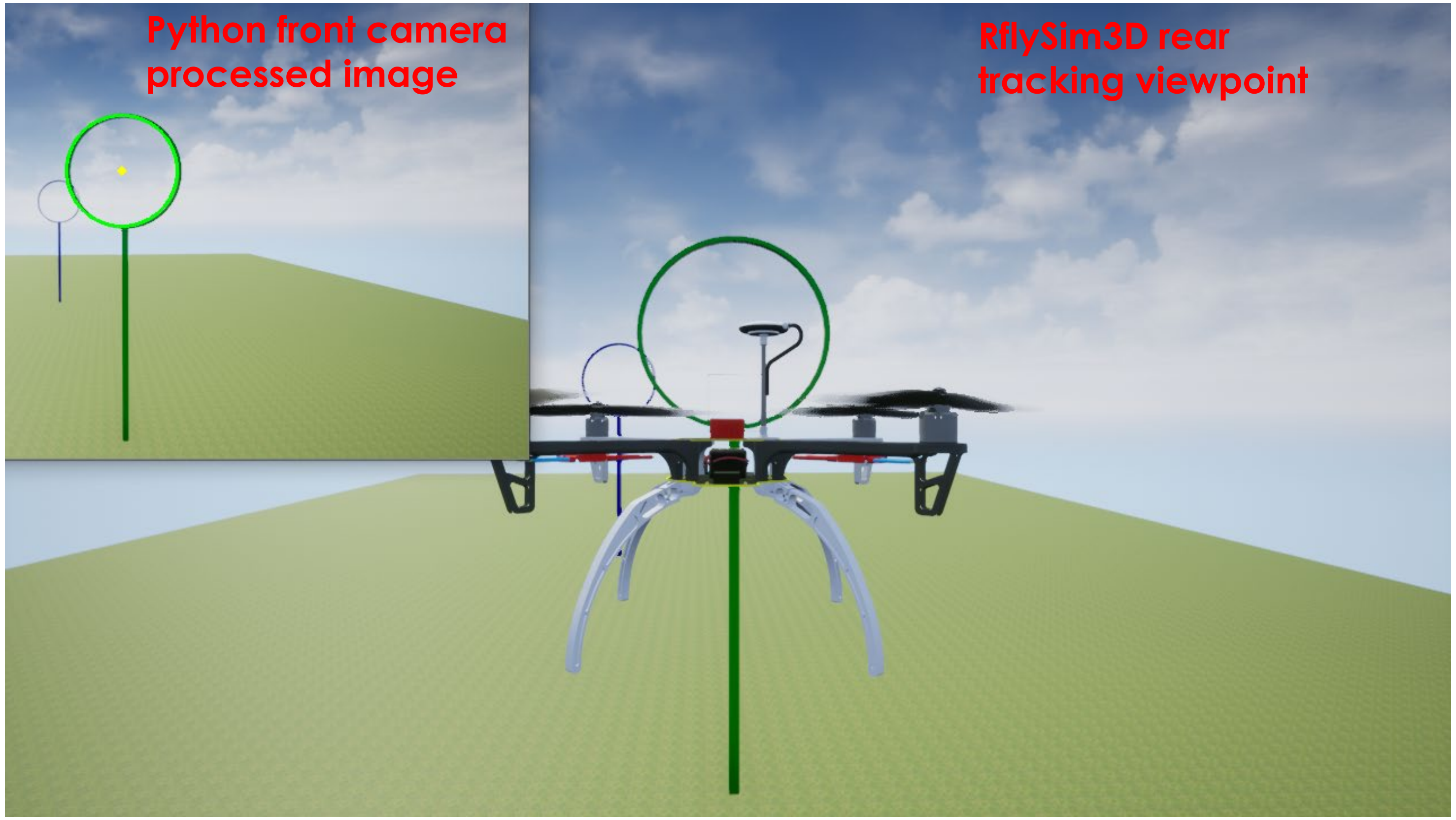
北航可靠飞行控制研究组

BUAA Reliable Flight Control Group

3. Examples of monocular vision control

3.6 Running effect of cross rings experiment

- 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 drone crosses through the three rings in sequence, and finally landed automatically.
- To use hardware-in-the-loop simulation, after setting up the flight controller, run the "**CrossRing3HITL.bat**" script and enter the flight controller serial port number to start the hardware-in-the-loop simulation system.



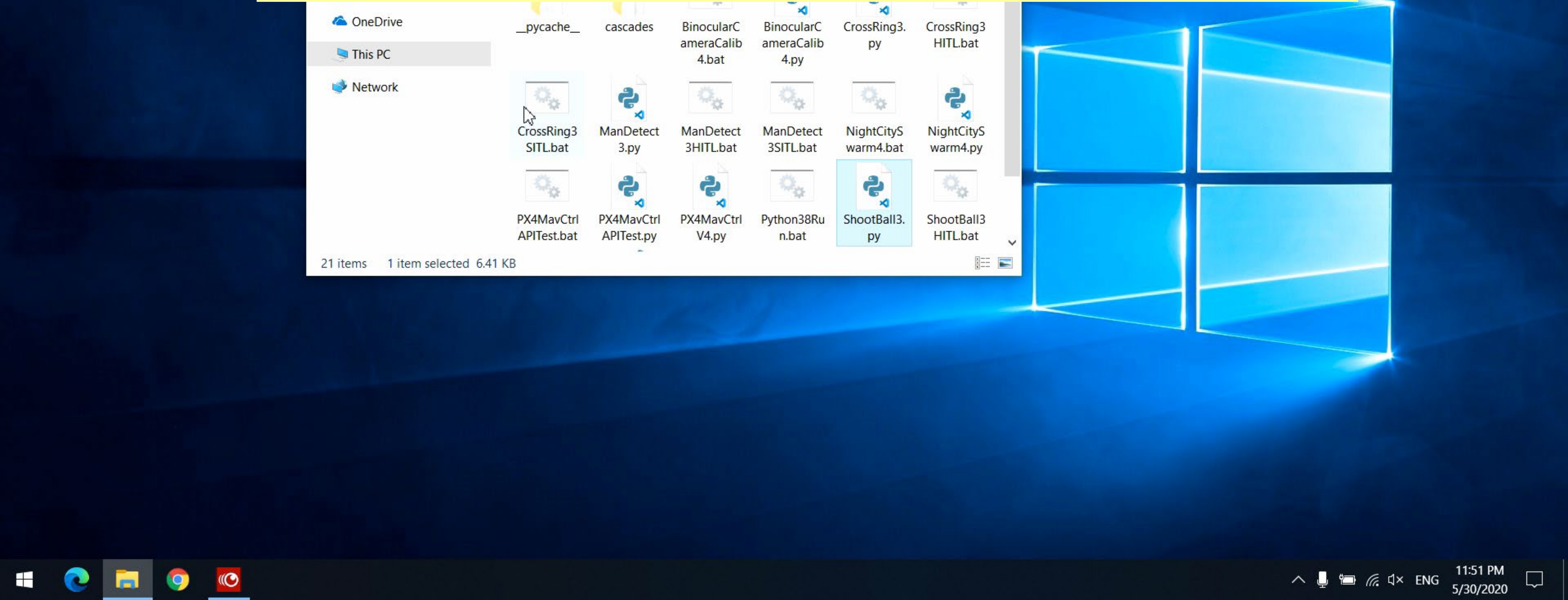
Note: If the computer performance is poor and the flight shakes, you can manually close the opened RflySim3D window (trailing observation angle)

RflySim: Vision-based navigation and control for multicopter crossing rings experiment

Watch this video by clicking the following links:

YouTube: https: //youtu.be/PvxEfY7oMq4

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

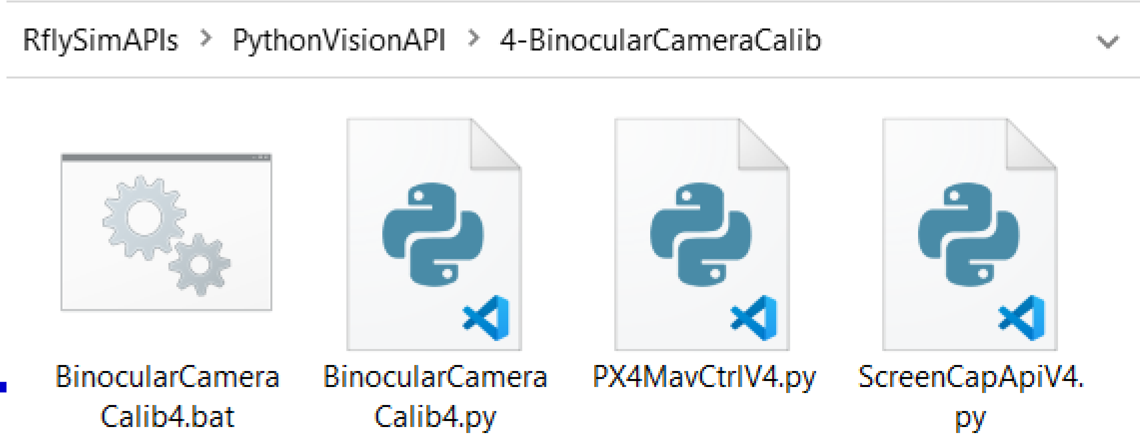


Content

4. Examples of binocular vision control

4.1 Binocular camera calibration experiment

- In Windows Explorer, open and enter the "**RflySimAPIs\PythonVisionAPI\4-BinocularCameraCalib**" folder, the contents of which are as shown in the figure below
- Among them, "**BinocularCameraCalib4.py**" is the main Python program of this demo; "**BinocularCameraCalib4.bat**" is a batch startup script, double-clicking it will automatically open three RflySim3D windows (left and right cameras + trailing global observation angle); "**PX4MavCtrlV4.py**" is the Python communication interface module of this platform.



4. Examples of binoc

4.2 Analysis of binocular camera calibration code

- Open the "**BinocularCameraCalib4.py**" file with VS Code
- The key code lines are shown on the right, this script can obtain images of multiple windows at the same time
- Please read and study the rest of the code according to the previous explanation

```
ImgInfo1 = sca.getHwndInfo(Wd01)
ImgInfo2 = sca.getHwndInfo(Wd02)


startTime = time.time()
lastTime = time.time()
timeInterval = 3
while True:
    lastTime = lastTime + timeInterval
    sleepTime = lastTime - time.time()
    if sleepTime > 0:
        time.sleep(sleepTime)
    else:
        lastTime = time.time()
    TargePos = [InitTargePos[0]+random.randint(0,100)/100.0*2,InitT
    TargeAng = [InitTargeAng[0]+random.randint(-50,50)/50.0*30/180.

    mav.sendUE4Pos(100,40,0,TargePos,TargeAng,-1)
    time.sleep(0.1)

    # The following code will be executed per 3s
    img1=sca.getCVImg(ImgInfo1)
    # Get the first camera view and change to gray
    pic1=cv2.cvtColor(img1, cv2.COLOR_BGRA2GRAY)

    img2=sca.getCVImg(ImgInfo2)
    # Get the first camera view and change to gray
    pic2=cv2.cvtColor(img2, cv2.COLOR_BGRA2GRAY)

    # add image storing algorithm for MATLAB offline calibration
    # or add online calibration algorithm here

    cv2.imshow("pic1",pic1) # Show the processed image
    cv2.imshow("pic2",pic2)

    cv2.waitKey(1)
```

1. Get the left and right RflySim3D window information

2. Time interval 3s (0.33Hz)

3. Set position and attitude of checkerboard randomly

4. Get the images from left and right cameras

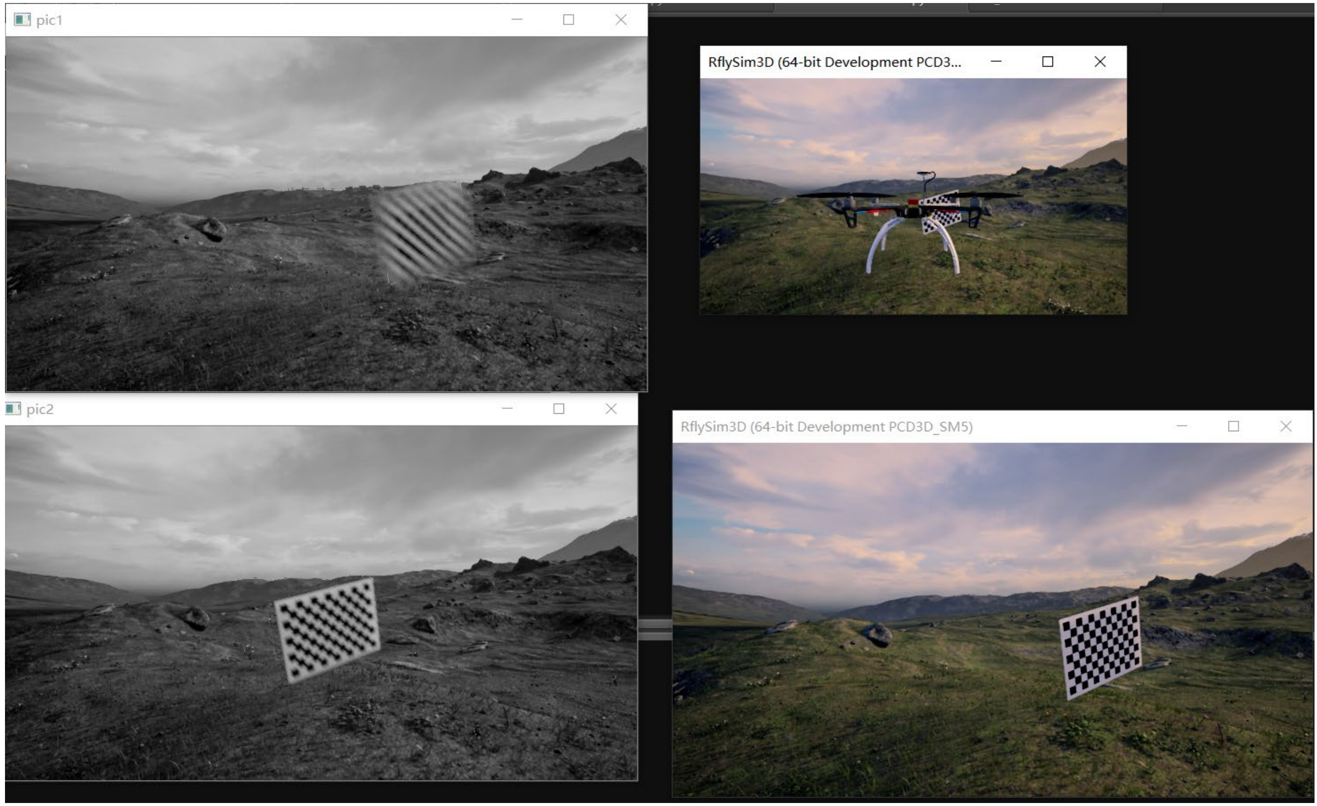
5. Calibration algorithm can be added here

6. Display the processed images

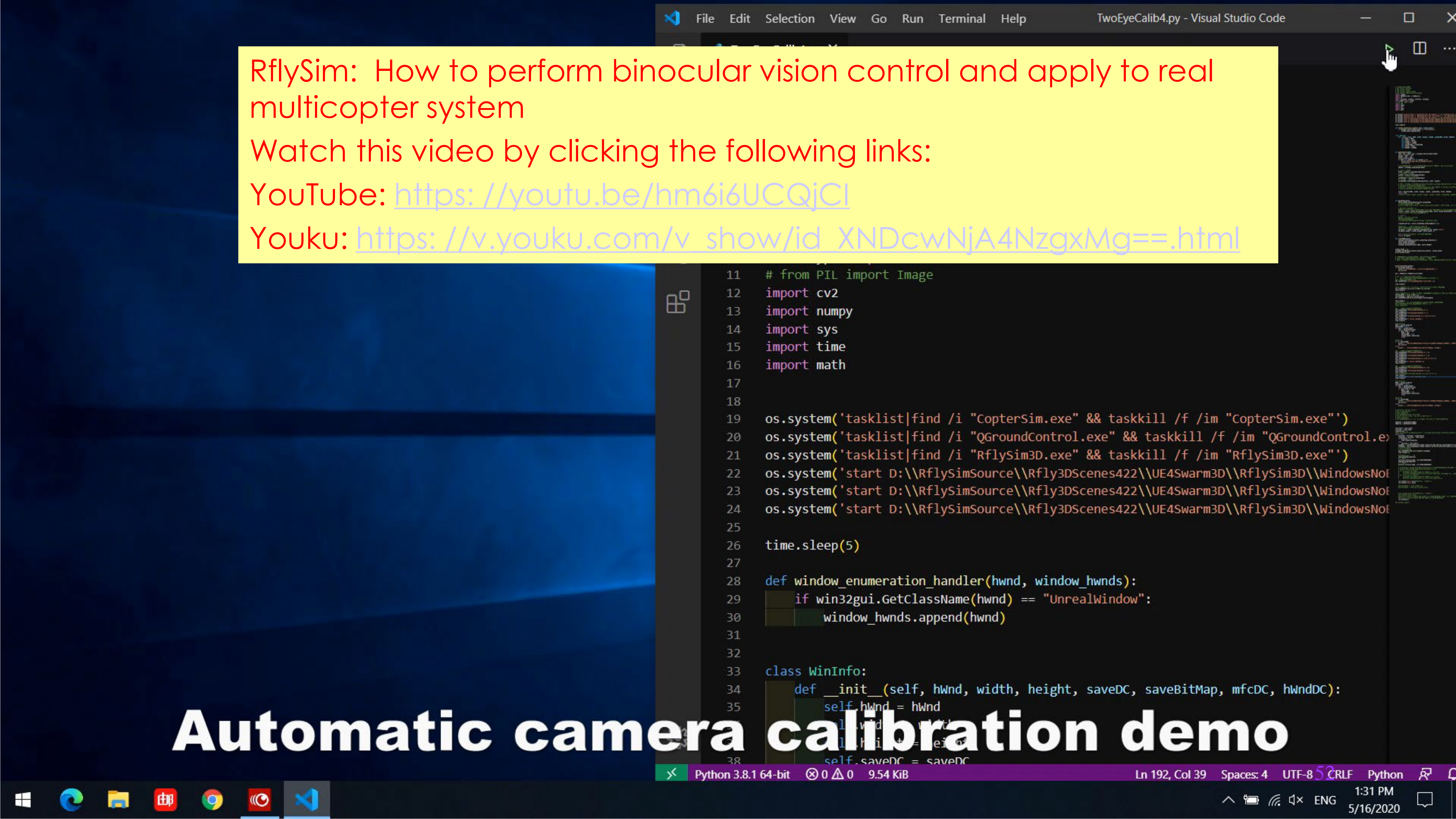
4. Examples of binocular vision control

4.3 Experimental results of binocular camera calibration

- After running "**BinocularCameraCalib4.bat**", then run "**BinocularCameraCalib4.py**" is ok.
- Open multiple RflySim3D scenes, create a new vehicle, configure binocular position information, create a new target, and place the target according to random rules.
- **Assignment 1**: After acquiring the images of the left and right cameras, implement the online calibration algorithm.
- **Assignment 2**: Store the images of the left and right cameras as pictures locally, and then use the calibration toolbox of MATLAB to find the parameters



Note: If the computer performance is poor and the flight shakes, you can manually close the last opened RflySim3D window (trailing observation angle), just use the front left and right camera angles

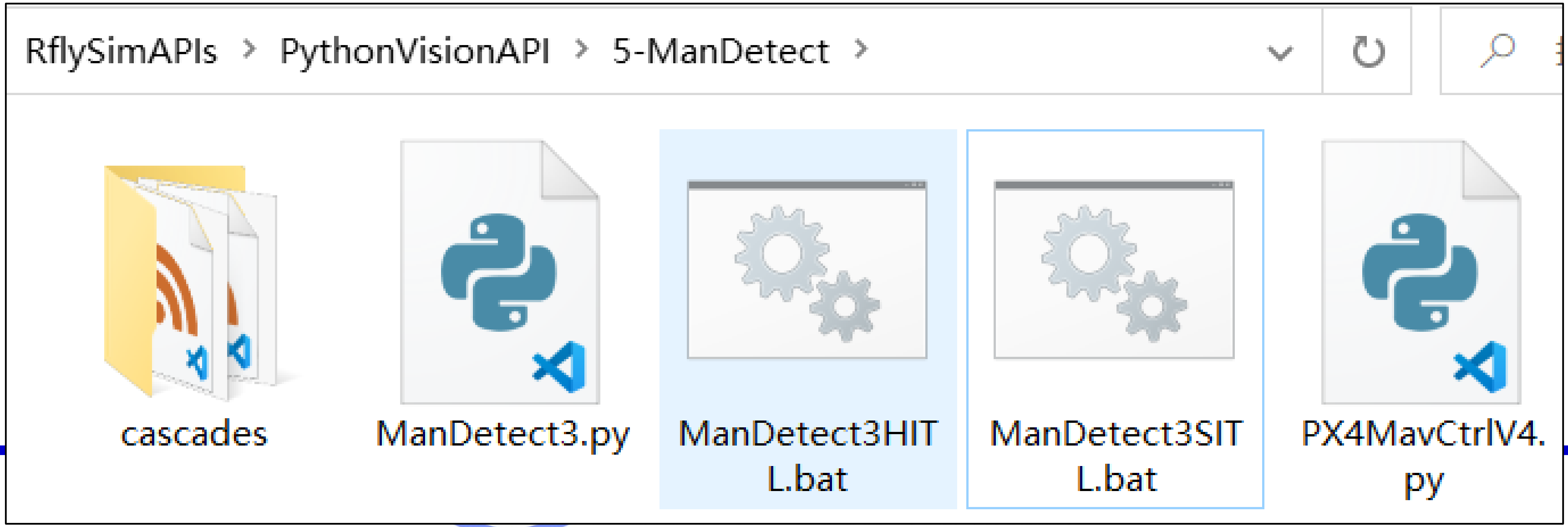


RflySim: How to perform binocular vision control and apply to real multicopter system
Watch this video by clicking the following links:
YouTube: https://youtu.be/hm6i6UCQjCI
Youku: https://v.youku.com/v_show/id_XNDcwNjA4NzgxMg==.html
File Edit Selection View Go Run Terminal Help
TwoEyeCalib4.py - Visual Studio Code
# from PIL import Image
import cv2
import numpy
import sys
import time
import math
os.system('tasklist|find /i "CopterSim.exe" && taskkill /f /im "CopterSim.exe"')
os.system('tasklist|find /i "QGroundControl.exe" && taskkill /f /im "QGroundControl.ex
os.system('tasklist|find /i "RflySim3D.exe" && taskkill /f /im "RflySim3D.exe"')
os.system('start D:\\RflySimSource\\Rfly3DScenes422\\UE4Swarm3D\\RflySim3D\\WindowsNoE
os.system('start D:\\RflySimSource\\Rfly3DScenes422\\UE4Swarm3D\\RflySim3D\\WindowsNoE
os.system('start D:\\RflySimSource\\Rfly3DScenes422\\UE4Swarm3D\\RflySim3D\\WindowsNoE
time.sleep(5)
def window_enumeration_handler(hwnd, window_hwnds):
if win32gui.GetClassName(hwnd) == "UnrealWindow":
window_hwnds.append(hwnd)
class WinInfo:
def __init__(self, hWnd, width, height, saveDC, saveBitMap, mfcDC, hWndDC):
self.hWnd = hWnd
self.saveDC = saveDC
Automatic camera calibration demo
Python 3.8.1 64-bit
Ln 192, Col 39 Spaces: 4 UTF-8 CRLF Python
1:31 PM
5/16/2020

4. Examples of binocular vision control

4.4 UAV binocular vision demo

- In Windows Explorer, open and enter the "**RflySimAPIs\PythonVisionAPI\5-ManDetect**" folder, the contents of which are as shown in the figure below
- Among them, "**ManDetect3.py**" is the main Python program of this demo; the cascades folder contains some feature files in **XML** format for face recognition; the difference between "**ManDetect3HITL.bat**" and "**ManDetect3SITL.bat**" to the desktop shortcut is: The "**UDPSIMMODE**" communication UDP mode also selects the "**Mavlink_Full**" mode; three RflySim3D windows are opened, left and right cameras + trailing observation angles.



4. Examples of binocular

4.5 Source code analysis of drone binocula

- Open the "**ManDetect3.py**" file with VS Code

```python
# Create MAVLink control API instance
mav = PX4MavCtrl.PX4MavCtrler(20100)
# Init MAVLink data receiving loop
mav.InitMavLoop()

mav.sendUE4Cmd(b'RflyChangeMapbyName Grasslands')
time.sleep(5)

# send vehicle position command to create a man, with copterID=100
# the man is located before the drone, and rotated to 180 degree (face th
mav.sendUE4Pos(100,30,0,[1,0,-8.086],[0,0,math.pi])
time.sleep(1)

# send command to change object with copterID=100 (the man just created)
mav.sendUE4Cmd(b'RflyChange3DModel 100 16')
time.sleep(0.5)

# send command to the first RflySim3D window, to switch to vehicle 1
mav.sendUE4Cmd(b'RflyChangeViewKeyCmd B 1',0)
time.sleep(0.5)
# switch to oboard camera 1 (front camera)
mav.sendUE4Cmd(b'RflyChangeViewKeyCmd V 1',0)
time.sleep(0.5)
# change the camera position to [0.1 -0.25 0] related to the body
mav.sendUE4Cmd(b'RflyCameraPosAng 0.1 -0.25 0',0)
time.sleep(1)
# change the first window to size 720x405
mav.sendUE4Cmd(b'r.setres 720x405w',0)
```

1. initialize MAVLink and configure RflySim3D windows

```python
# Process the image in the following timer
startTime = time.time()
lastTime = time.time()
timeInterval = 0.01 # time interval of the timer
face_cascade=cv2.CascadeClassifier(sys.path[0]+'\cascades\haarcascade
while True:
    lastTime = lastTime + timeInterval
    sleepTime = lastTime - time.time()
    if sleepTime > 0:
        time.sleep(sleepTime)
    else:
        lastTime = time.time()
    # The following code will be executed 100Hz (0.01s)

    img1=sca.getCVImg(ImgInfo1)
    # Get the first camera view and change to gray
    pic1=cv2.cvtColor(img1, cv2.COLOR_BGRA2GRAY)

    img2=sca.getCVImg(ImgInfo2)
    # Get the second camera view and change to gray
    pic2=cv2.cvtColor(img2, cv2.COLOR_BGRA2GRAY)

    faces1=face_cascade.detectMultiScale(pic1,1.3,5) # face recogniti
    faces2=face_cascade.detectMultiScale(pic2,1.3,5) # face recogniti
    for (x,y,w,h) in faces1:
        pic1=cv2.rectangle(pic1,(x,y),(x+w,y+h),(255,0,0),1) # Draw a
    for (x,y,w,h) in faces2:
        pic2=cv2.rectangle(pic2,(x,y),(x+w,y+h),(255,0,0),1)

    cv2.imshow("pic1",pic1) # Show the processed image
    cv2.imshow("pic2",pic2)

    # add target tracking algorithm here with mav.SendVelNED or SendV

    cv2.waitKey(1)
```

2. Timer frequency 100Hz

3. Add face recognition xml files

4. Get images from two cameras

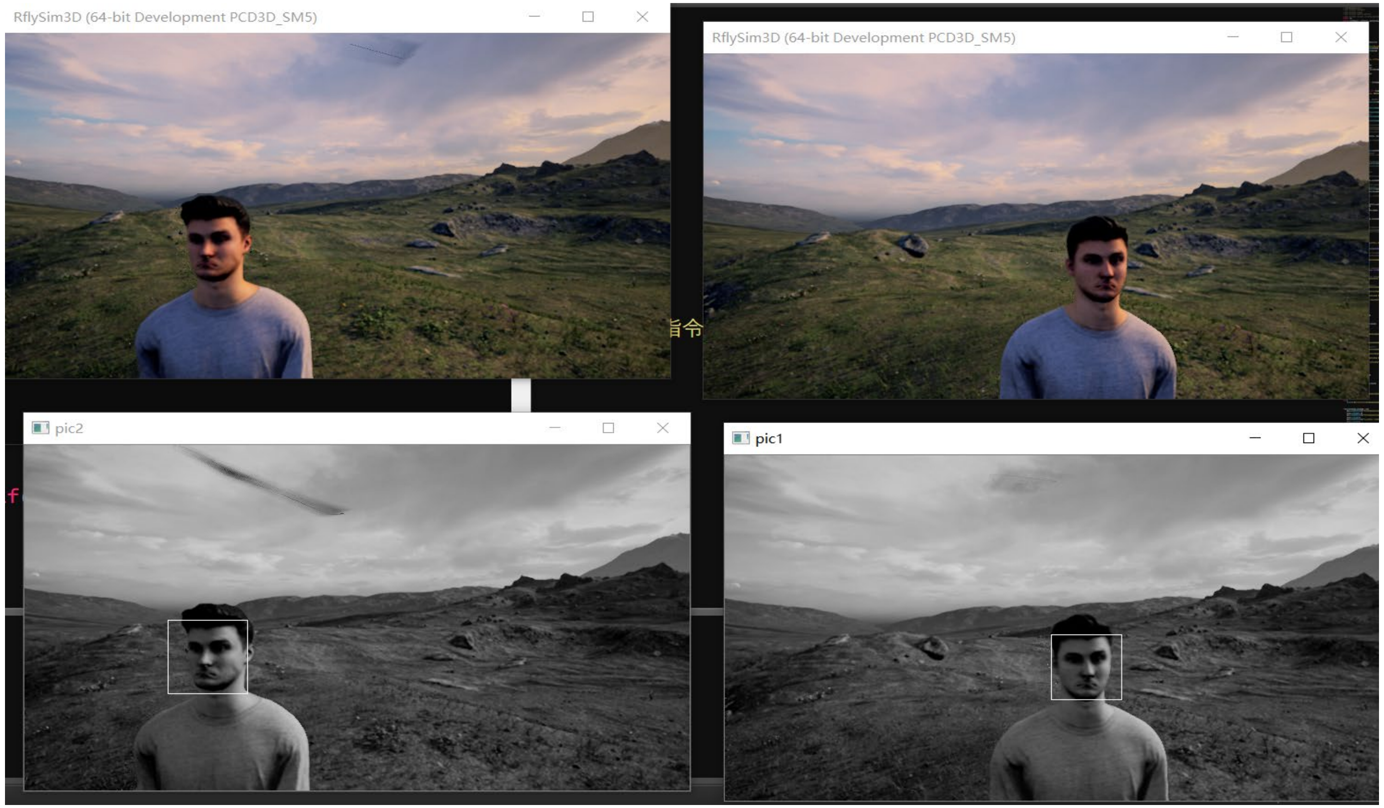
5. Image process with OpenCV

6. Show processed images

4. Examples of binocular vision control

4.6 Binocular vision operation effect of UAV

- Run "**ManDetect3SITL.bat**" or "**ManDetect3HITL.bat**", then run "**ManDetect3.py**"
- Generate a walking person in RflySim3D and set it to face the plane. After the plane takes off, the face recognition algorithm is turned on, and the face is selected by the binocular box
- **Assignment 1**: Update the position of the person in real time, achieve the simulation of the person walking, and write the vehicle tracking controller
- **Assignment 2**: Change to front-view + down-view camera, verify tracking + optical flow algorithm.



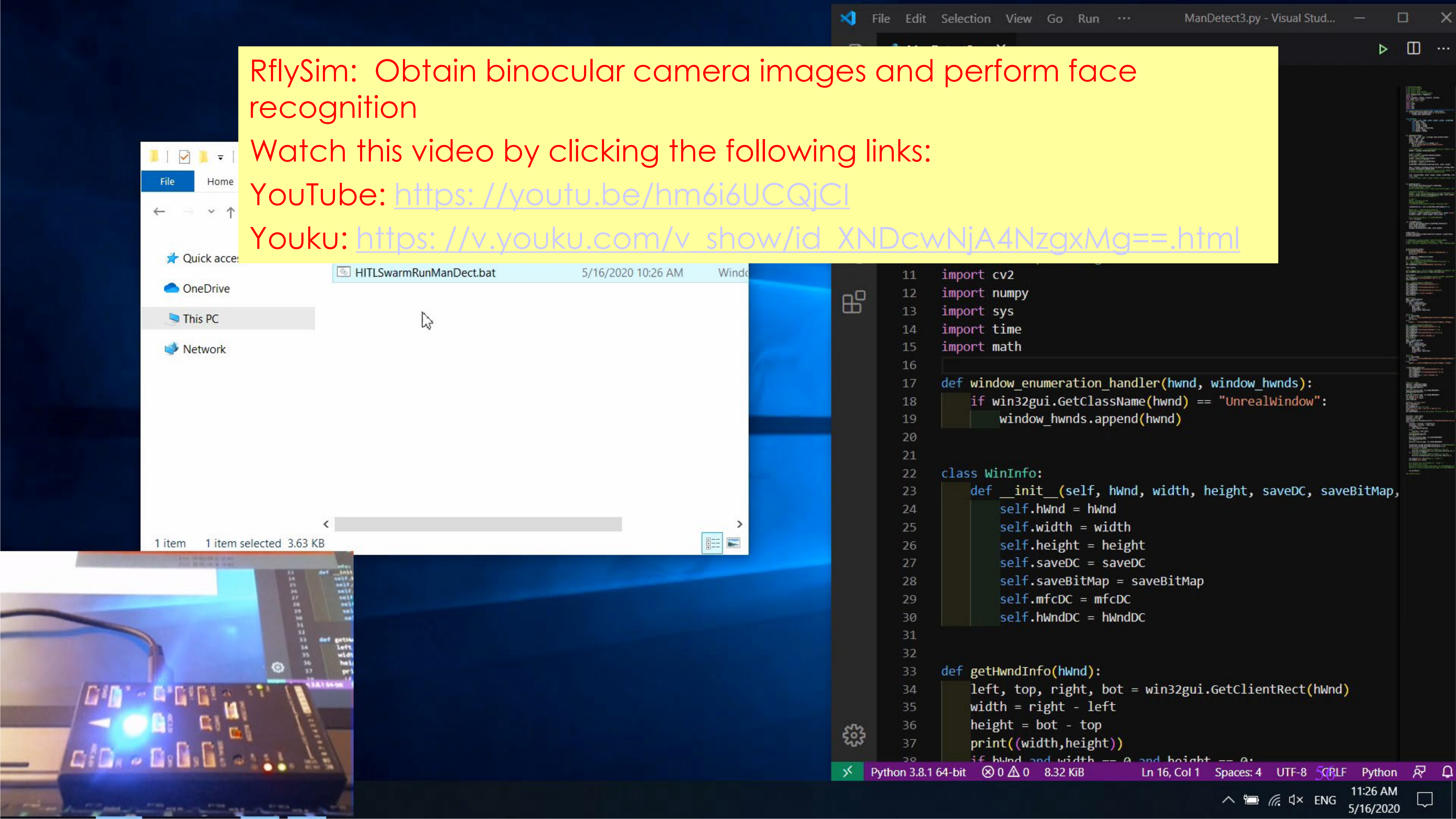
Note: If the computer performance is poor and the flight shakes, you can manually close the last opened RflySim3D window (trailing observation angle), just use the front left and right camera angles

RflySim: Obtain binocular camera images and perform face recognition

Watch this video by clicking the following links:

YouTube: https: //youtu.be/hm6i6UCQjCI

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



4. Examples of binocular vision control

4.7 Deployment of UAV Vision Algorithm

- Copy the Python code of the vision control to the onboard computer and replace the RflySim3D vision image with the camera image to complete the deployment of the vision algorithm

RflySim: Binocular flight platform introduction

Watch this video by clicking the following links:

YouTube: https: //youtu.be/hm6i6UCQjCI

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

Vision&Sw

- e 10min
- Onboard communication interface for vision&Swarm
- Optical flow & laser ranger finder for indoor experiments
- M8N GPS for outdoor experiments
- Visible light camera & Binocular camera
- Onboard TX2 or Raspberry Pi system
- Labels for Optitrack and UWB system

Thanks