AIoT：樹莓派應用

Chapter 4：OpenCV X 人臉辨識

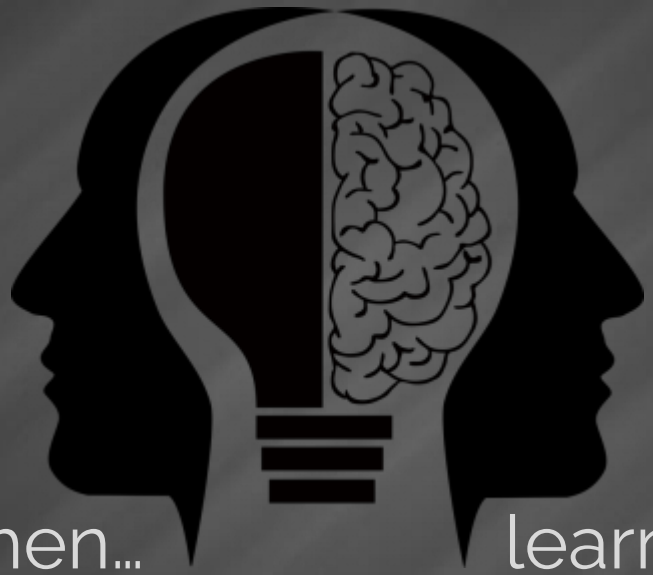
賴秉樑 debugger
學院創辦人

課程網址 https://max543.com/debugger

max 極限翻轉學院
青少年與兒童程式教育

何謂 AIoT？

Artificial Intelligence



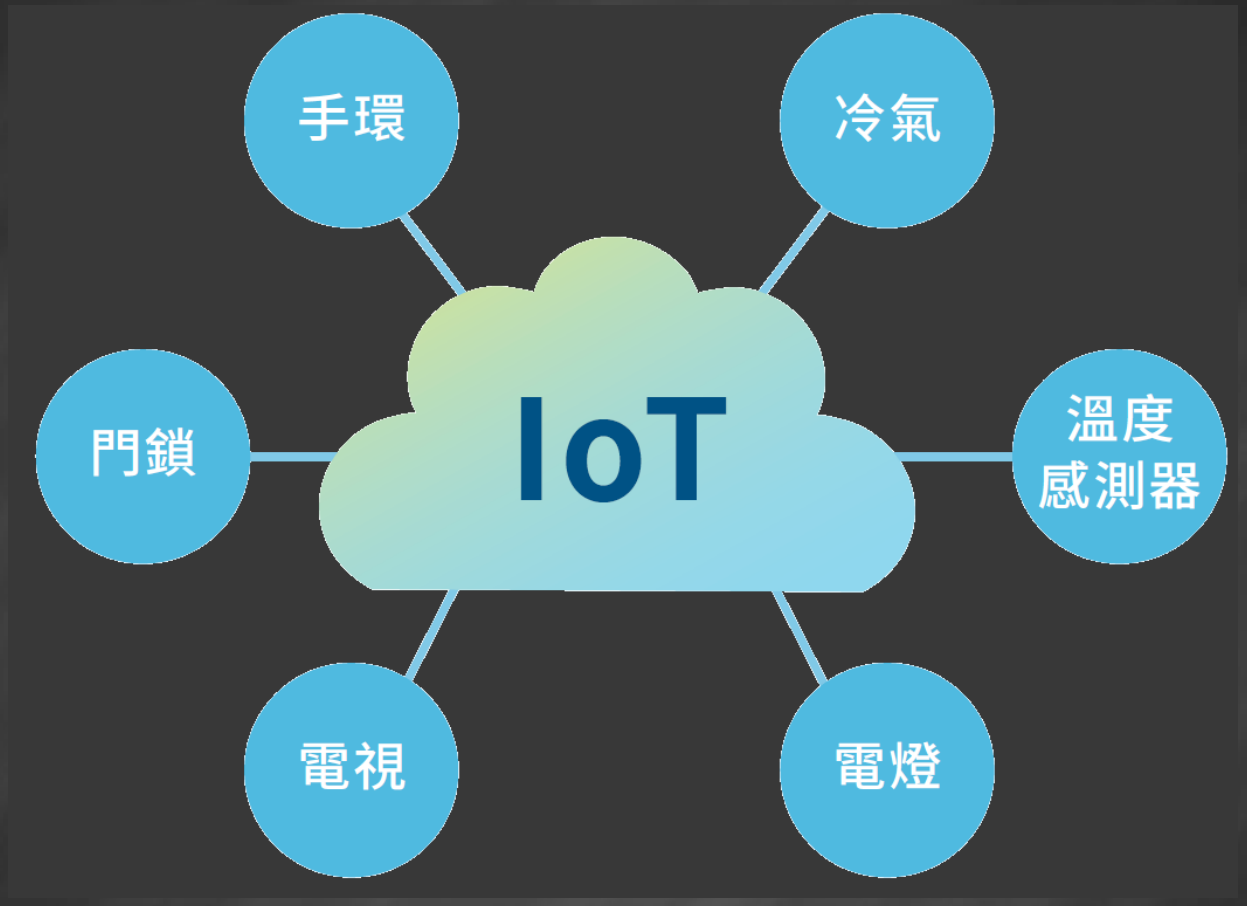
If...then...

learn...

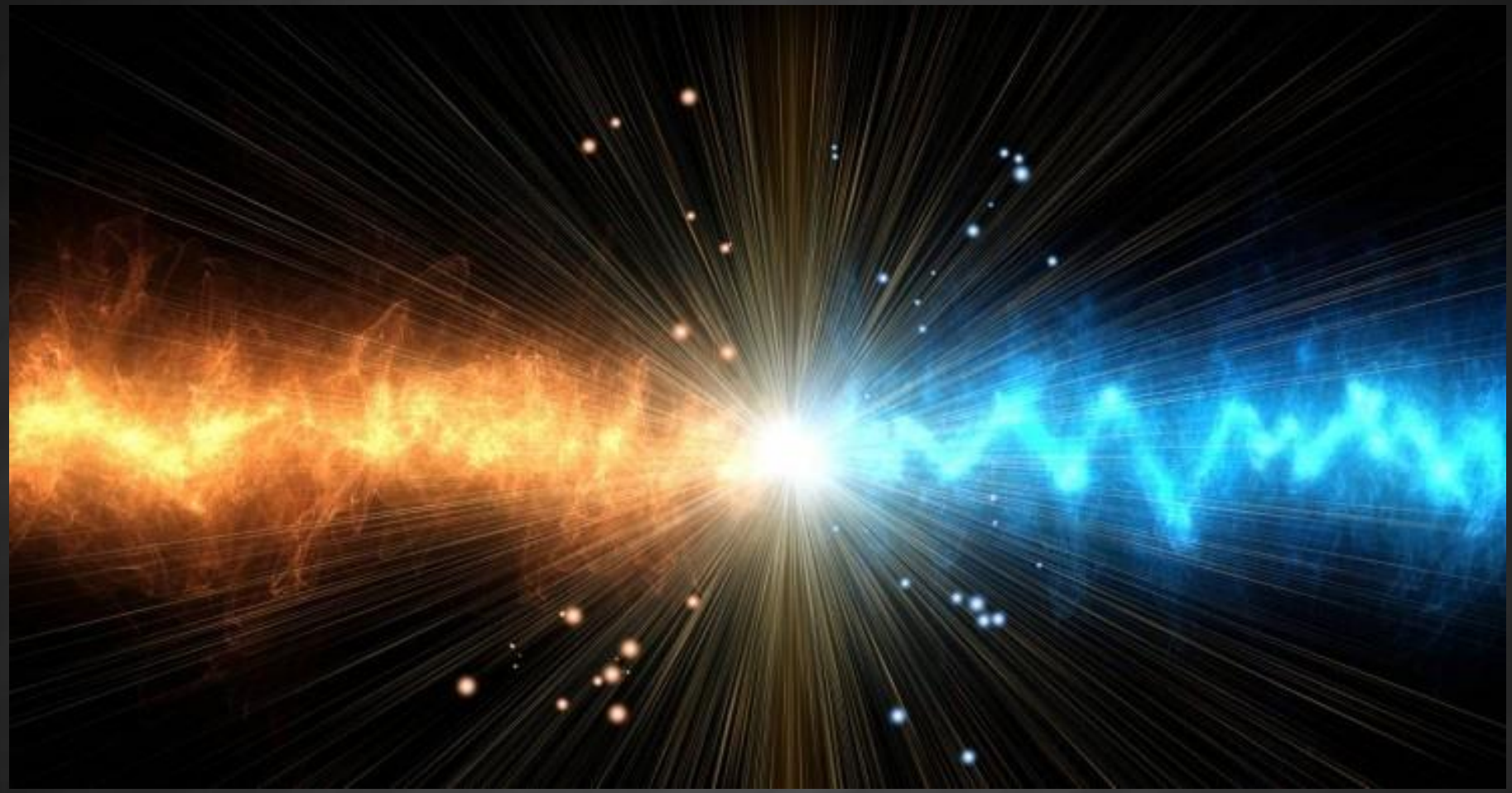
AI 的主流技術：機器學習

- 早期的技術：將問題以人工分析後，轉成程式語言的規則法 (rule-based)。
- 現在的技術：機器學習，準備一些問題與對應的答案給電腦後，讓它自行找出其中的規則，並且有能力針對類似的問題給出正確的答案。
  - ✓ 歸功於網路龐大的資訊量，或各式感測器收集的資料。

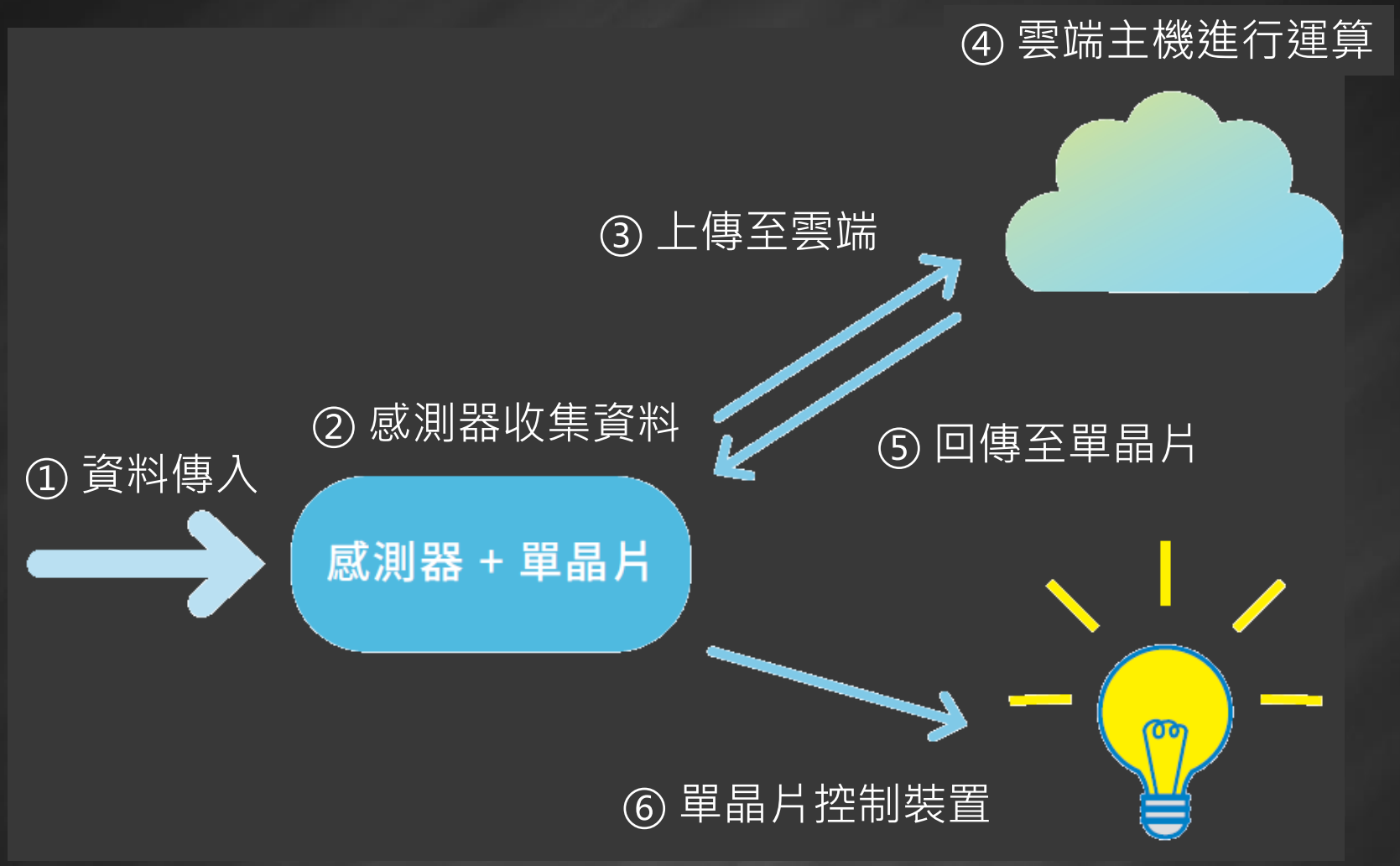
何謂 AIoT？



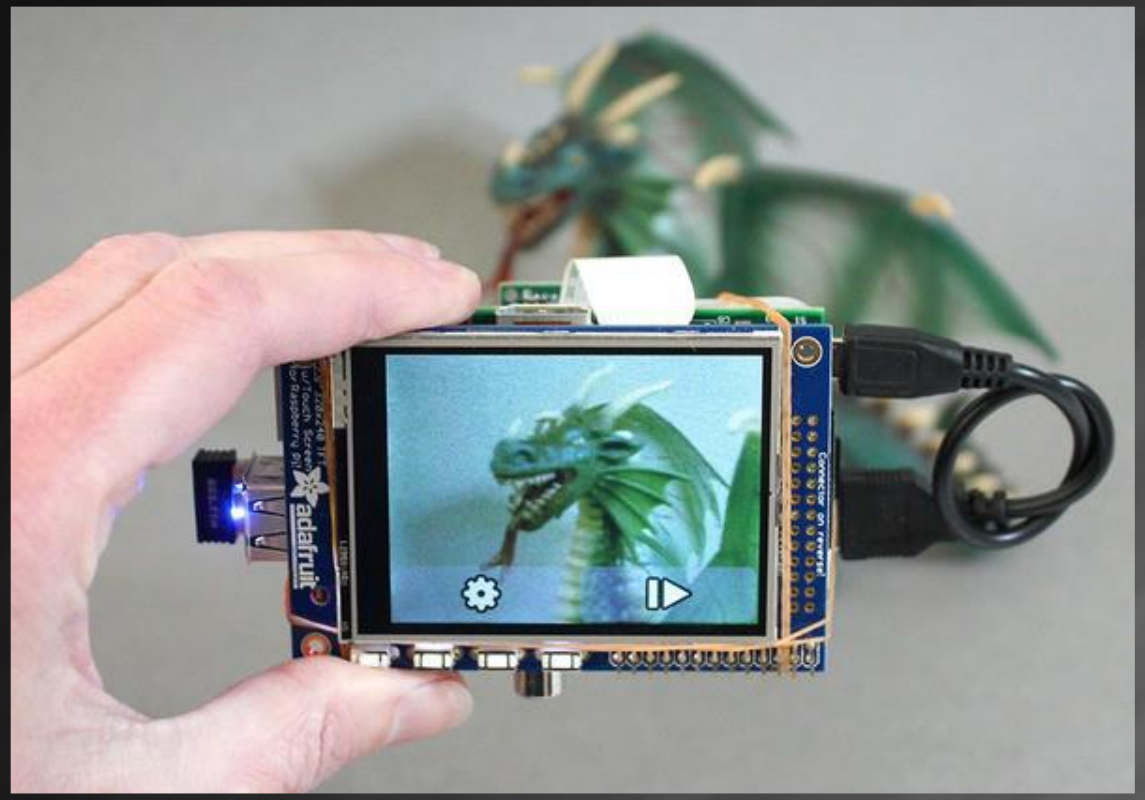
何謂 AIoT？

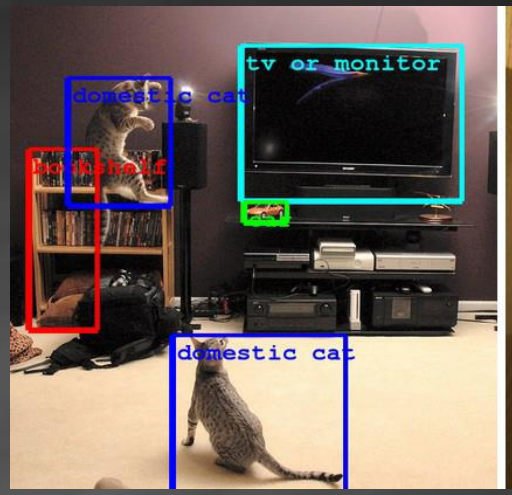


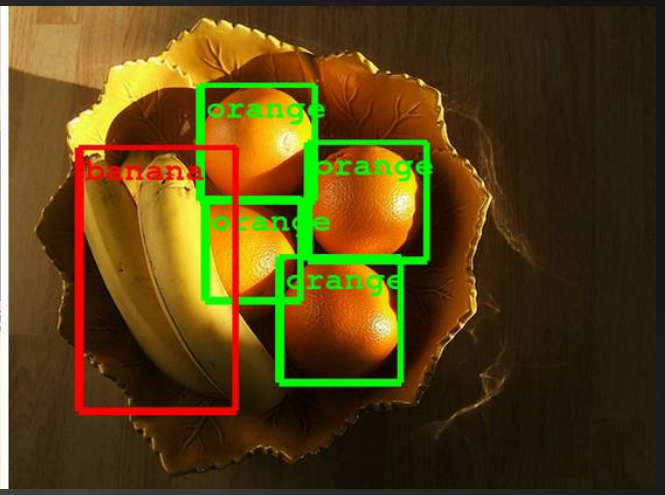
雲端運算



具有影像辨識的雲端相機





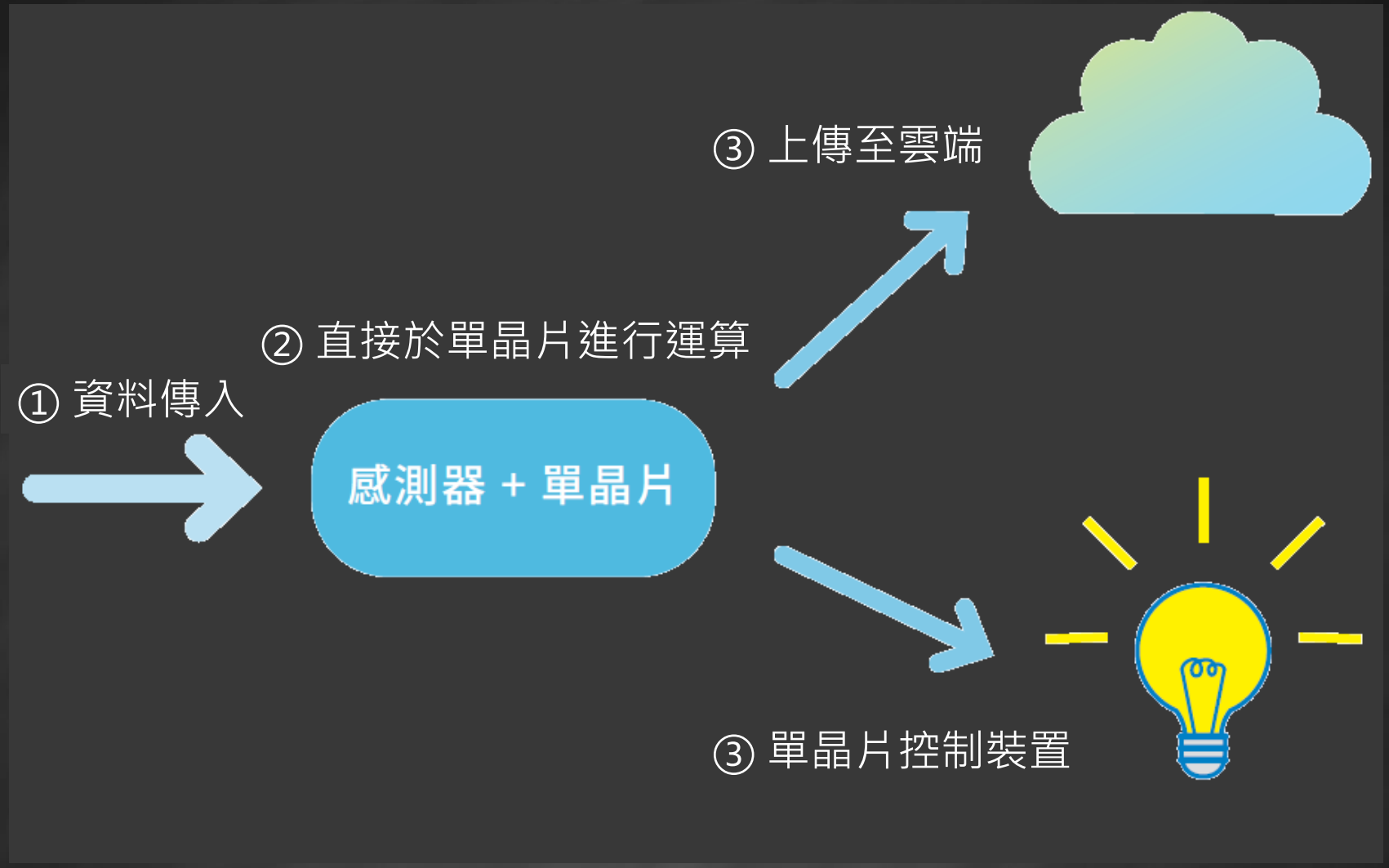


imagga

可做影像辨識的相機

https://learn.adafruit.com/diy-wifi-raspberry-pi-touch-cam

邊緣運算



人臉偵測與追蹤

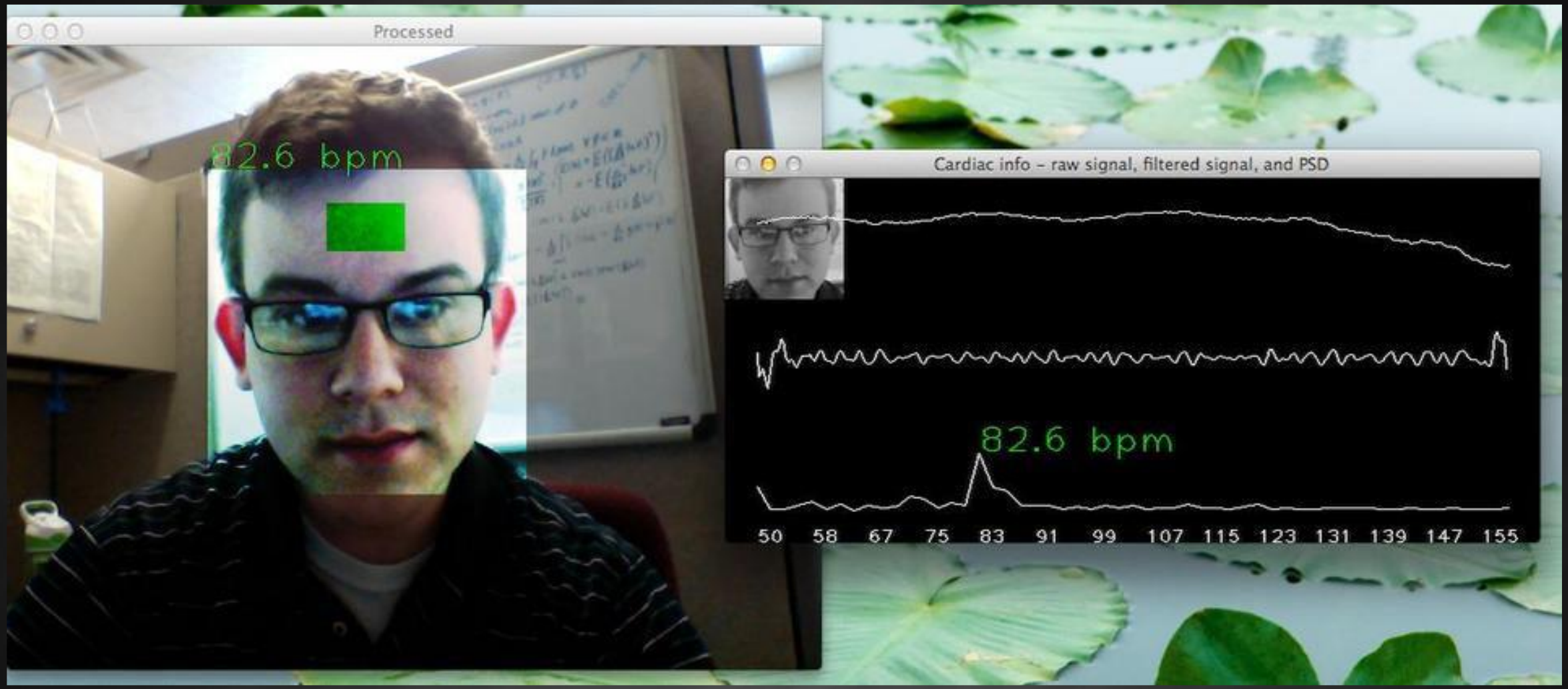
OpenCV

https://learn.adafruit.com/diy-wifi-raspberry-pi-touch-cam

IP Camera（網路攝影機）

https://www.codeproject.com/Articles/665518/Raspberry-Pi-as-low-cost-HD-surveillance-camera

脈博辨識



https://github.com/thearn/webcam-pulse-detector

Linux 套件有版本相依性，先安裝虛擬環境

目的：解決 Python 模組版本不相容的問題

4-1 建立與管理 Python 虛擬環境

ex4-1

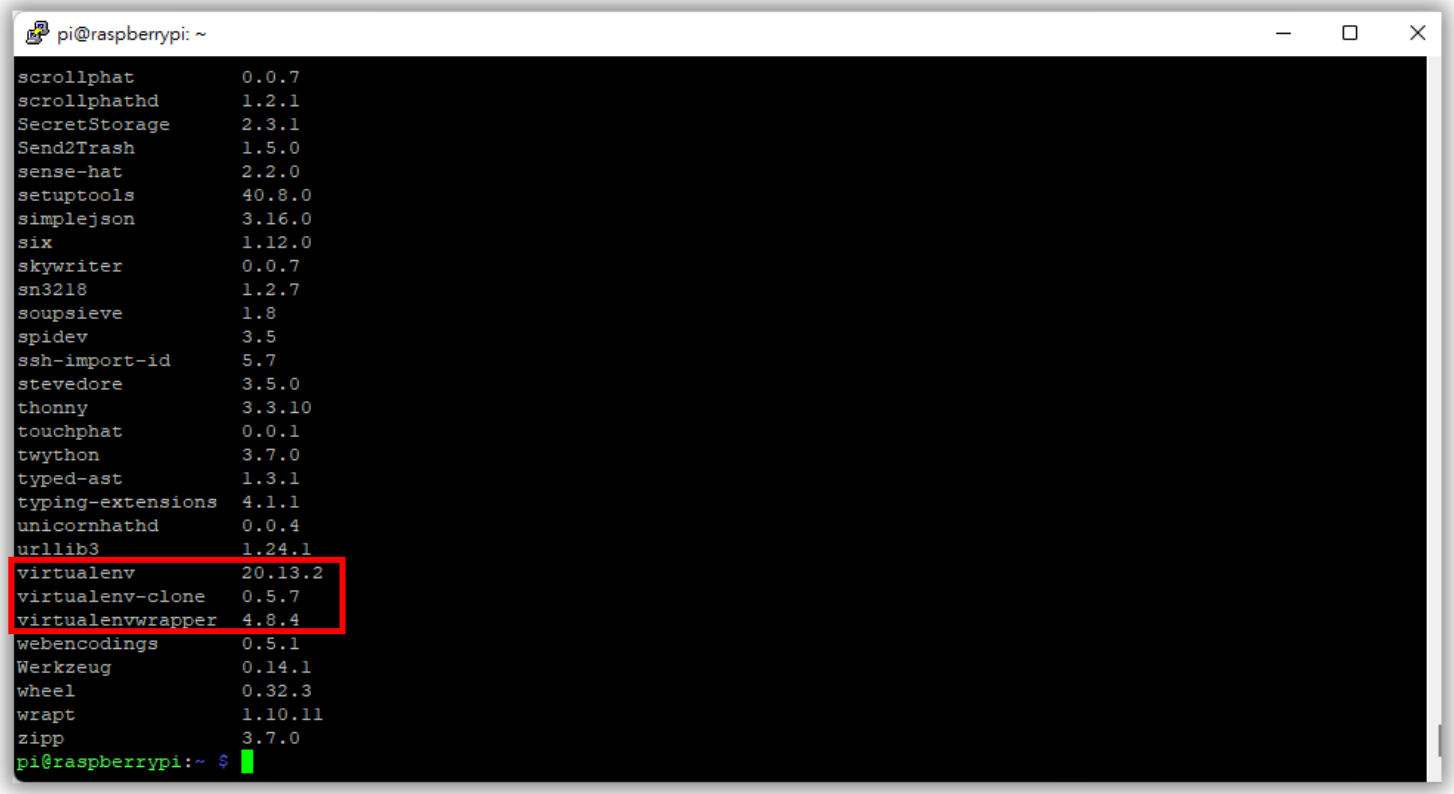
- Python 虛擬環境可以針對不同 Python 專案建立專屬的開發環境。
  - ✓ 例如：特定 Python 版本和不同套件安裝的需求，特別是哪些需要特定版本套件的 Python 專案，我們可以針對此專案建立專屬的虛擬環境，而不會因為套件版本的相容性問題，而影響到其他 Python 專案的開發環境。
- Raspberry Pi OS 可以使用 venv 或 virtualenv 來建立、啟動、刪除與管理 Python 虛擬環境。
- 在本課程是使用 virtualenv + virtualenvwrapper 套件。

```
$ sudo pip3 install virtualenv
$ sudo pip3 install virtualenvwrapper
```

確認使否安裝成功

- Raspberry Pi OS 支援 Python 2 和 Python 3，所以使用 pip3 執行檢查目前的 Python 模組，確認有安裝 virtualenv 和 virtualenvwrapper

```
$ pip3 list
```



編輯 .bashrc 檔

ex4-2

- 編輯 .bashrc 檔，新增路徑設定，以便可以執行 virtualenvwrapper

```
$ sudo nano ~/.bashrc
```

加入下列三行到 .bashrc

```
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```

重新載入更新的路徑設定

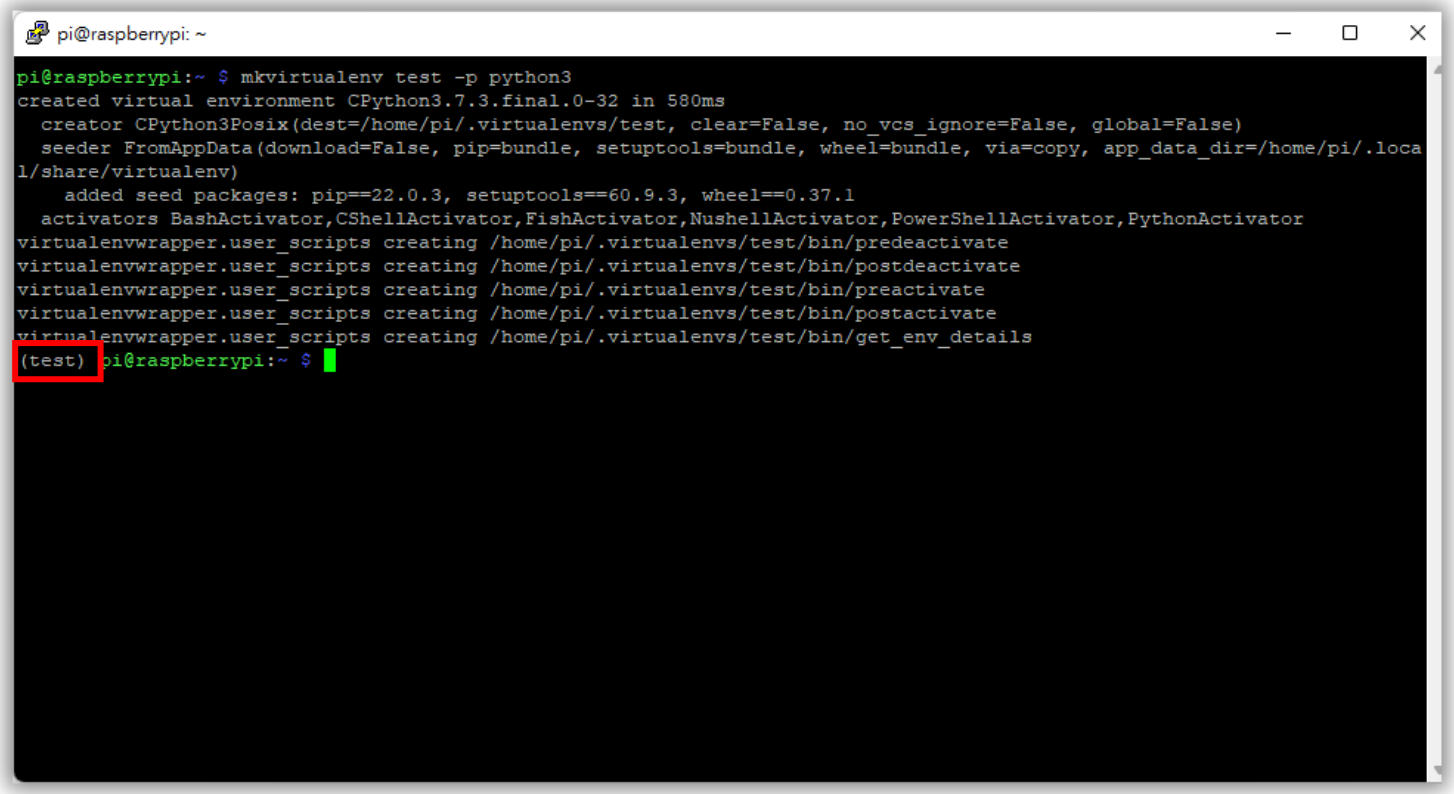
```
$ source ~/.bashrc
```

建立 Python 3 的虛擬環境

ex4-3

- 完成相關套件的安裝和設定後，準備新增名為 test 的 Python 虛擬環境。在 Raspberry Pi OS 建立 Python 虛擬環境的指令是 mkvirtualenv，參數 -p 指定 Python 版本是 Python 3 版，如下所示：

```
$ mkvirtualenv test -p python3
```



虛擬環境管理指令

- 關閉目前虛擬環境

  ```
  $ deactivate
  ```

- 顯示已建立的虛擬環境

  ```
  $ lsvirtualenv
  ```

- 啟動虛擬環境：當成功新增 test 虛擬環境後，我們可以使用 workon 指令來啟動 Python 虛擬環境，如下所示：

  ```
  $ workon test
  ```

- 移除虛擬環境

  ```
  $ rmvirtualenv test
  ```

- 複製虛擬環境

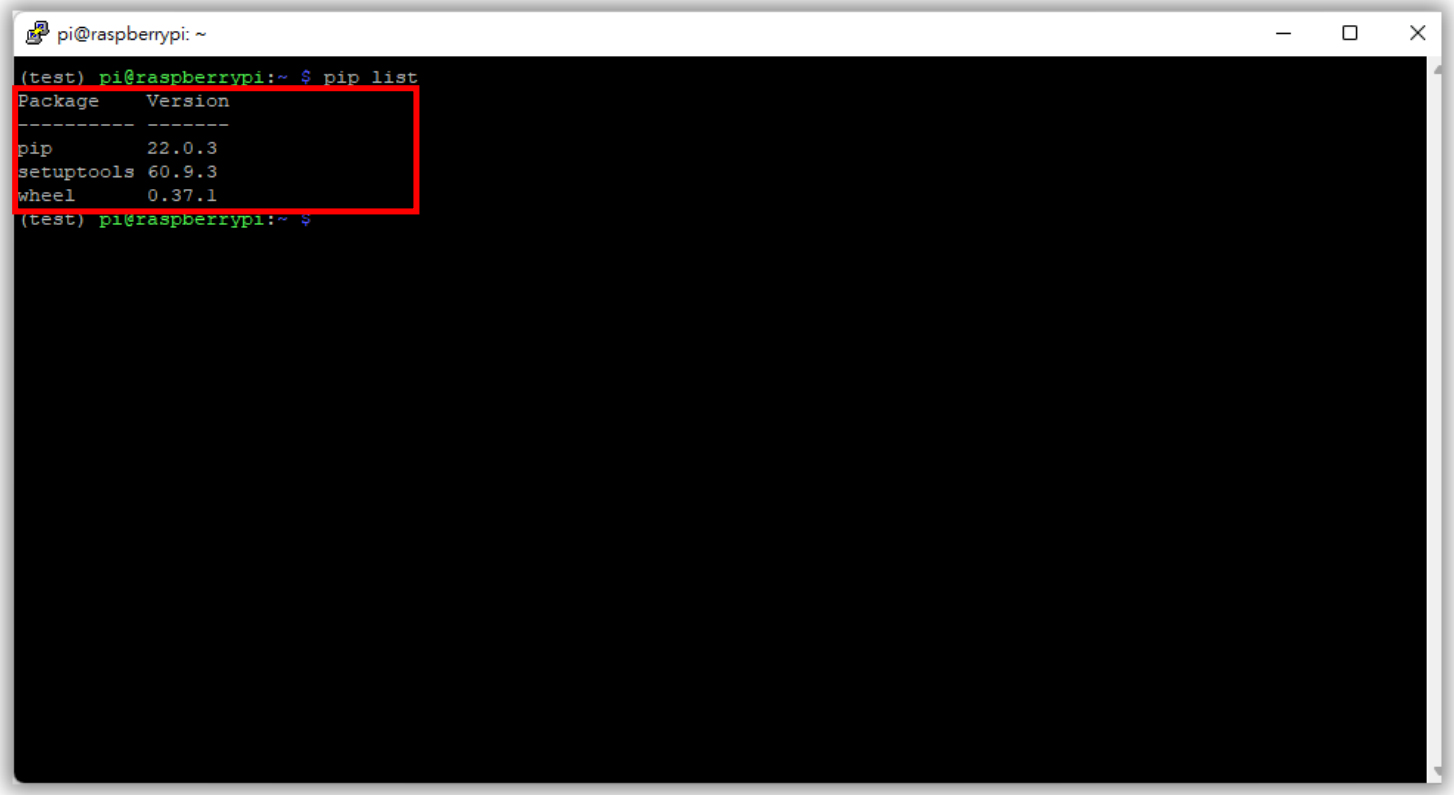
  ```
  $ cpvirtualenv test (來源) test1 (目的)
  ```

檢視虛擬環境安裝的 Python 套件清單

ex4-4

- 在成功啟動 test 虛擬環境後，可以看到前方改成虛擬環境名稱 (test)，然後，請輸入下列指令來檢視虛擬環境安裝的 Python 套件清單，如下所示：

```
(test) $ pip list
```



強大的 OpenCV

4-2 在樹莓派安裝 OpenCV (1/2)

- OpenCV (Open Source Computer Vision Library) 是跨平台 BSD 授權的一套著名的電腦視覺函式庫，這是 Intel 發起並參與開發，幫助開發圖片處理、影片處理、電腦視覺的人臉辨識和物體辨識等人工智慧的相關應用。
  - ✓ OpenCV 可以在 Android、iOS 上開發，支援的程式語言也有 C/C++、Java、Python，重點是免費的。
- 基本上，安裝 OpenCV 有兩種方式，如下所示：
  - ✓ 使用 pip 指令：使用 Python 套件管理 pip 安裝 OpenCV，安裝過程比較簡單，但是不會優化 OpenCV 的執行效能。
  - ✓ 自行編譯 OpenCV 程式碼進行安裝：如果有特殊 OpenCV 版本和優化需求，我們需要自行編譯 OpenCV 程式碼。

方法一：以 pip 安裝 OpenCV 套件

4-2　在樹莓派安裝 OpenCV (2/2)

- 步驟一：更新與升級作業系統
- 步驟二：安裝 OpenCV 的相關支援套件
- 步驟三：建立 Python 虛擬環境 OpenCV
- 步驟四：在虛擬環境安裝支援的 Python 套件
- 步驟五：在 Python 虛擬環境安裝 OpenCV
- 步驟六：檢查 OpenCV 是否成功安裝

步驟一：更新與升級已安裝的套件

ex4-5

- 在了解如何建立和管理 Python 虛擬環境後，我們準備新增名為 opencv 的 Python 虛擬環境後，在此 Python 虛擬環境安裝 OpenCV。
- 先更新當前安裝的套件：

```
$ sudo apt-get update          # 軟體資料庫同步，升級前都先做
$ sudo apt-get upgrade         # 升級已安裝的軟體
```

步驟二：安裝 OpenCV 的相關支援套件

ex4-6

- 安裝 HDF (Hierarchical Data Format version 5) 資料集的套件，這是支援大型、複雜和異質資料的開源檔案格式，可以用來儲存大量的圖片資料：

```
$ sudo apt-get install -y libhdf5-dev
```

- 安裝矩陣運算和 JPEG-2000 圖片函式庫：

```
$ sudo apt-get install -y libatlas-base-dev
$ sudo apt-get install -y libjasper-dev
```

- 安裝 QT GUI 圖形介面的套件：

```
$ sudo apt-get install -y libqtgui4
$ sudo apt-get install -y libqt4-test
$ sudo apt-get install -y python3-pyqt5
```

步驟三：建立 Python 虛擬環境 OpenCV

ex4-7

- 新增名為 opencv 的 Python 3 虛擬環境：

```
$ mkvirtualenv opencv -p python3
```

- 檢查目前安裝的 Python 模組：

```
(opencv) $ pip list
```

步驟四：在虛擬環境安裝支援的 Python 套件

ex4-8

- 在虛擬環境安裝支援的 Python 套件，numpy 為陣列容器、matplotlib 為 Python 及其數值計算庫 numpy 的繪圖庫、imutils 可以讓我們更容易使用 OpenCV 圖片處理：

```
(opencv) $ pip install numpy==1.21.4
(opencv) $ pip install matplotlib
(opencv) $ pip install imutils
```

步驟五：在 Python 虛擬環境安裝 OpenCV

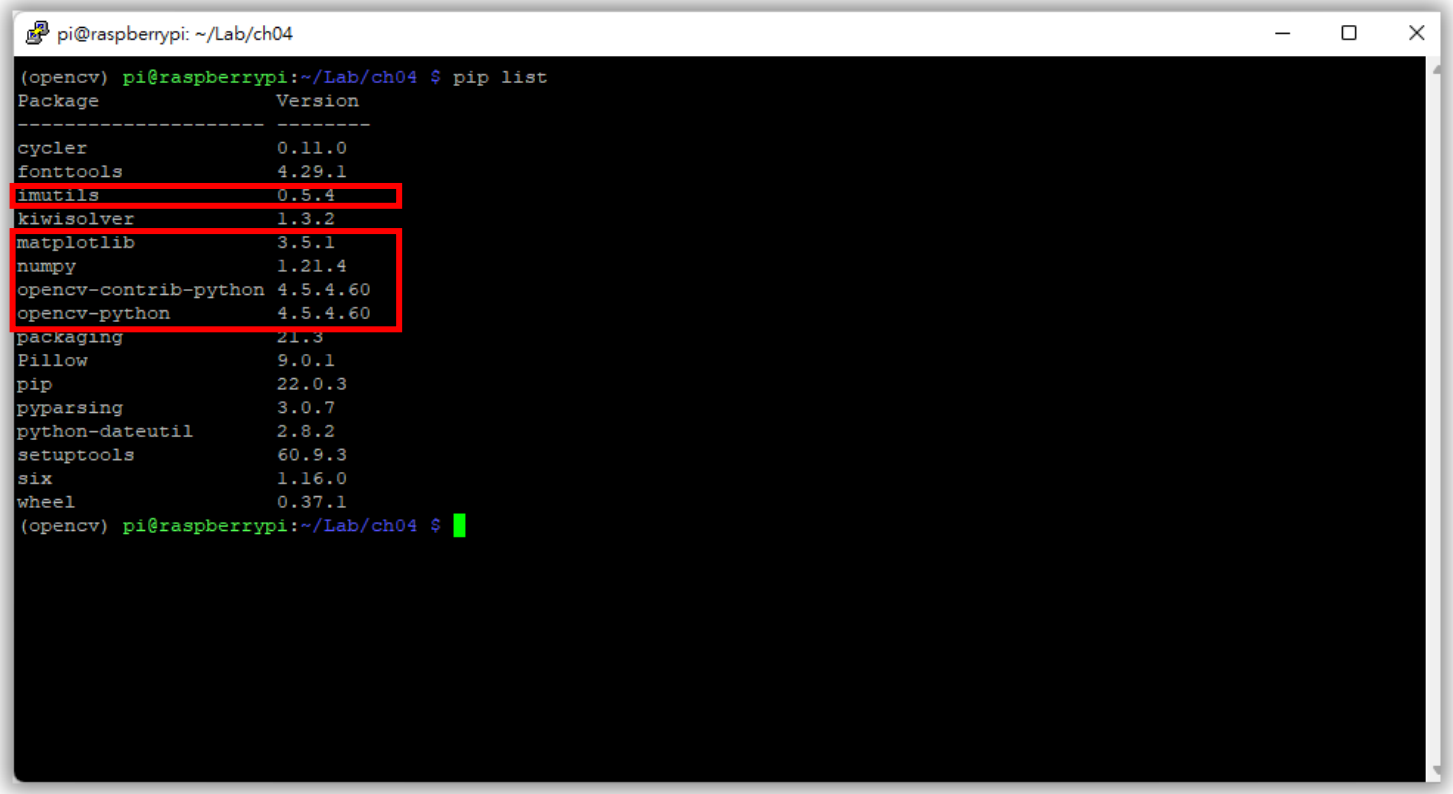
ex4-9

- 使用 pip 安裝 OpenCV：

```
(opencv) $ pip install opencv-python==4.5.4.60
(opencv) $ pip install opencv-contrib-python==4.5.4.60
```

- 輸入指令來檢視虛擬環境安裝的套件清單：

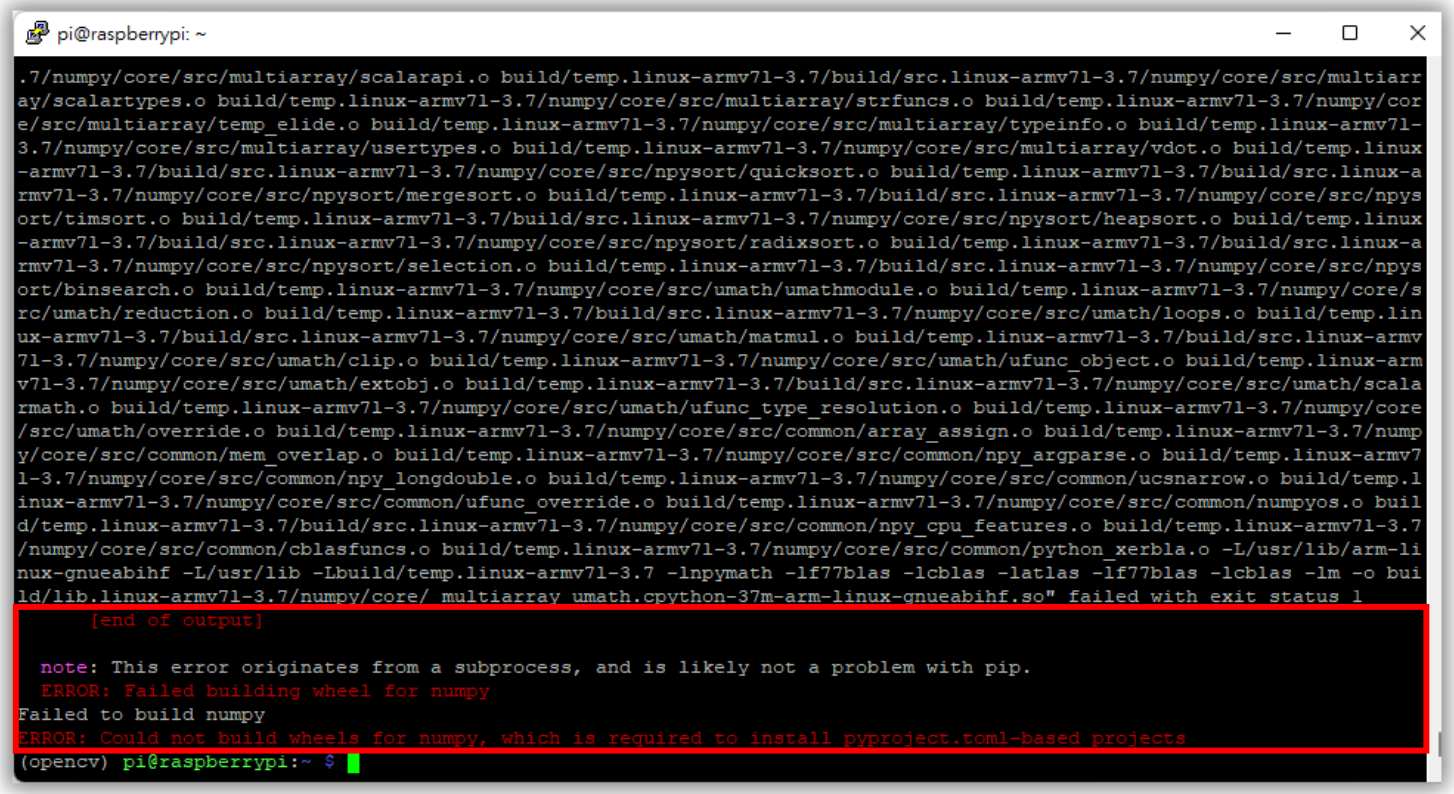
```
(opencv) $ pip list
```



補充：套件有相依性

- 例如：不指定 numpy 的版本，發生錯誤。

  (opencv) $ pip install numpy (會安裝最新的 1.22.2)



補充：原因是 numpy 有 Python 版本問題

幾件你應該要會的系統管理

- 檢查 Python 3 的版本：

```
$ python3 –-version
Python 3.7.3     # 顯示目前 Python 3 的版本
```

- 檢查 Python 3 安裝的模組：

```
$ pip3 list
```

- 檢查 Python 2 的版本：

```
$ python --version
Python 2.7.16    # 顯示目前 Python 2 的版本
```

- 檢查 Python 2 安裝的模組：

```
$ pip list
```

步驟六：檢查 OpenCV 是否成功安裝

- 檢查 Python 3 是否安裝好 OpenCV。

```
(opencv) $ python -version
>>> import cv2
>>> cv2.__version__
'4.5.4'    # 顯示目前 OpenCV 的版本
```

方法二：自行編譯 OpenCV 程式碼進行安裝

安裝 OpenCV 步驟

- 安裝 OpenCV 步驟
  - ✓ 安裝 OpenCV 軟件包
  - ✓ 準備編譯 OpenCV
  - ✓ 在 Raspberry Pi 編譯 OpenCV
  - ✓ 編譯後的清理
  - ✓ 在 Raspberry Pi 測試 OpenCV
  - ✓ 常見問題

安裝 OpenCV 相關套件 (1/2)

ex4-10

1. 更新當前安裝的軟件包

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

2. 安裝編譯套件

```
$ sudo apt-get install build-essential cmake
```

3. 安裝圖片套件

```
$ sudo apt-get install libjpeg-dev libpng-dev libtiff5-dev libjasper-dev
```

安裝 OpenCV 相關套件 (1/2)

4. 安裝影片套件

```
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

5. 安裝 GTK 套件

```
$ sudo apt-get install libgtk-3-dev libcanberra-gtk*
```

6. 最佳化套件

```
$ sudo apt-get install libatlas-base-dev gfortran
```

7. 樹莓派加裝

```
$ apt-get install at-spi2-core
```

安裝 OpenCV 相關套件 (1/2)

ex4-10

8. 安裝 Python 開發套件

```
$ sudo apt-get install python-dev python3-dev
```

9. 安裝 numpy 函式庫 (預設已有安裝)

```
$ pip install numpy          # 安裝 python2 的 numpy
$ pip3 install numpy         # 安裝 python3  的 numpy
```

安裝 OpenCV 步驟

- 安裝 OpenCV 軟件包
- 準備編譯 OpenCV
- 在 Raspberry Pi 上 編譯 OpenCV
- 編譯後的清理
- 在 Raspberry Pi 上測試 OpenCV
- 常見問題

準備編譯 OpenCV (1/3)

- 現在已經安裝好所有 OpenCV 所需要的軟件包，但我們需要做一些前置作業，讓我們可以加快我們的編譯過程。
- 我們需要暫時增加交換空間 (swap) 的大小，以加快我們在 Raspberry Pi 上編譯 OpenCV 的過程。
- 當設備的 RAM 用完時，作業系統將使用 swap。儘管 swap 比 RAM 慢很多，但在某些情況下它仍然很有用。

準備編譯 OpenCV (2/3)

1. 通過運行以下指令開始修改交換文件配置。

```
$ sudo nano /etc/dphys-swapfile
```

2. 在此文件中，找到下面這行，加大 swap，並存檔：

```
CONF_SWAPSIZE=100
```

```
CONF_SWAPSIZE=2048
```

3. 更改 swap 文件配置後，我們需要使用以下指令重新啟動其服務。

```
$ sudo systemctl restart dphys-swapfile
```

準備編譯 OpenCV (3/3)

ex4-12

4. 讓我們將所需的兩個 OpenCV 儲存到 Raspberry Pi 中

```
$ mkdir opencv                                  # 建立一個 opencv 資料夾擺放檔案
$ cd opencv                                     # 進入 opencv 資料夾
$ wget https://github.com/opencv/opencv/archive/master.zip
$ unzip master.zip && rm master.zip             # 解壓縮並刪除檔案
$ wget https://github.com/opencv/opencv_contrib/archive/master.zip
$ unzip master.zip && rm master.zip             # 解壓縮並刪除檔案
```

安裝 OpenCV 步驟

- 安裝 OpenCV 軟件包
- 準備編譯 OpenCV
- 在 Raspberry Pi 上 編譯 OpenCV
- 編譯後的清理
- 在 Raspberry Pi 上測試 OpenCV
- 常見問題

在 Raspberry Pi上 編譯 OpenCV (1/4)

ex4-13

1. 首先，在剛剛解壓縮的檔案，在 opencv-master 資料夾中，創建一個名為 "build" 的目錄，然後將工作目錄更改為該目錄。

```
$ cd opencv-master
$ mkdir build
$ cd build
```

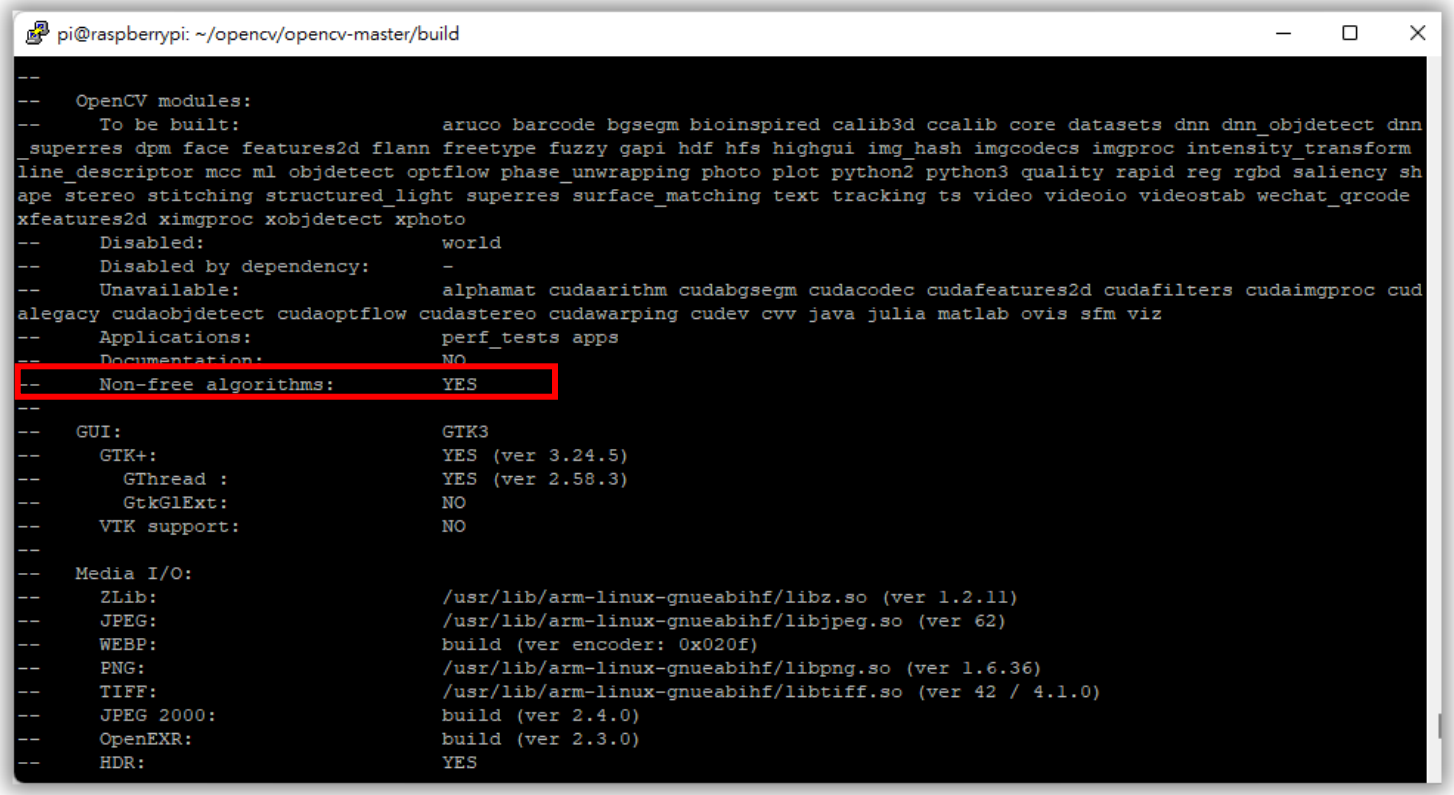
在 Raspberry Pi上 編譯 OpenCV (2/4)

2. 在 build 中，以 cmake 用來準備 OpenCV，以便在 Pi 上進行編譯，運行以下指令以生成所需的 makefile。

```
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-master/modules \
    -D ENABLE_NEON=ON \
    -D ENABLE_VFPV3=ON \
    -D WITH_TBB=ON \
    -D WITH_OPENMP=ON \
    -D BUILD_TESTS=OFF \
    -D OPENCV_ENABLE_NONFREE=ON \
    -D INSTALL_PYTHON_EXAMPLES=OFF \
    -D BUILD_EXAMPLES=OFF \
    -D OPENCV_EXTRA_EXE_LINKER_FLAGS=-latomic \
    -D PYTHON3_EXECUTABLE=/usr/bin/python3 \
    -D PYTHON_EXECUTABLE=/usr/bin/python \
     ..
```

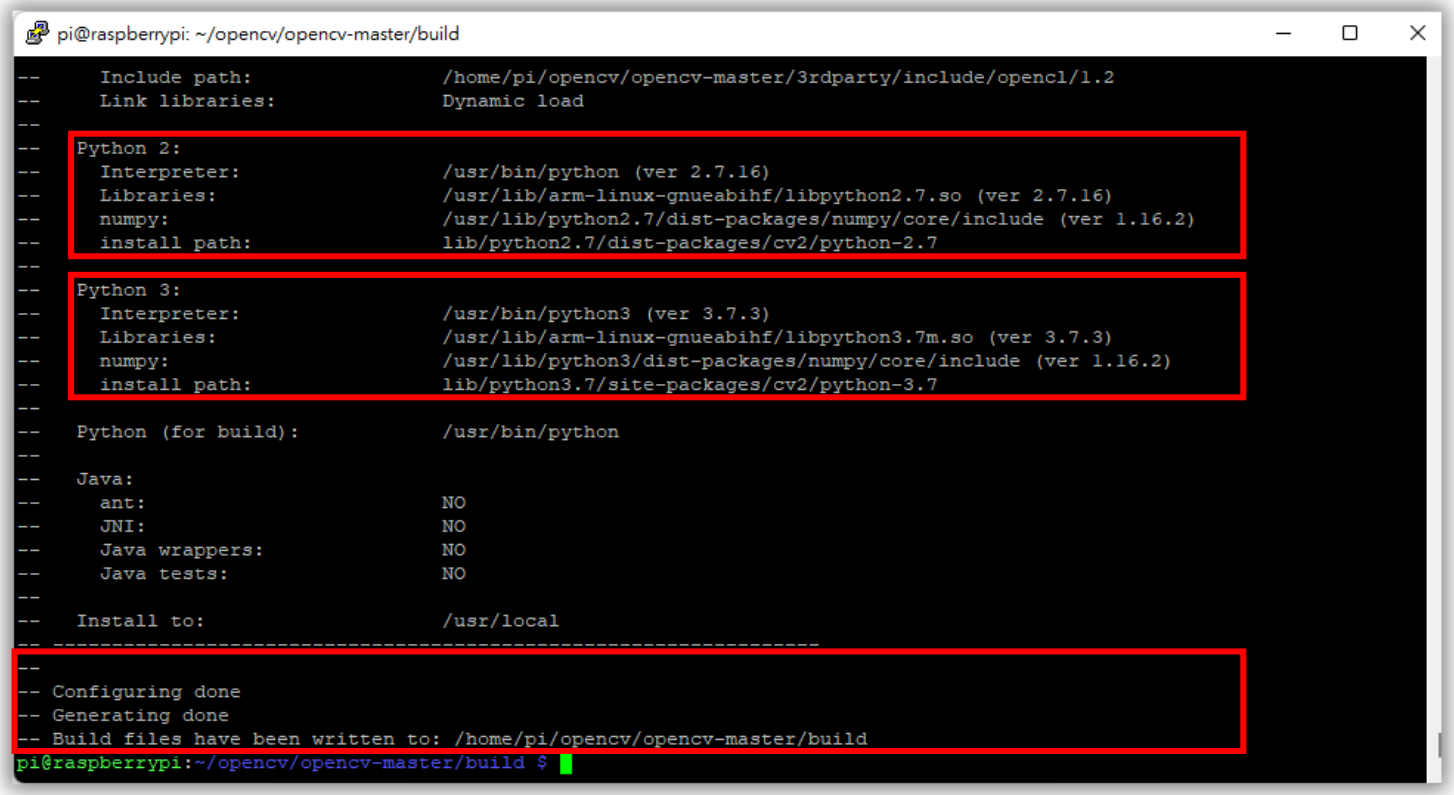
生成後的檢查 (1/2)

- Cmake 執行後，需檢查兩個地方，首先檢查 Non-free algorithms 是否為 YES。



生成後的檢查 (2/2)

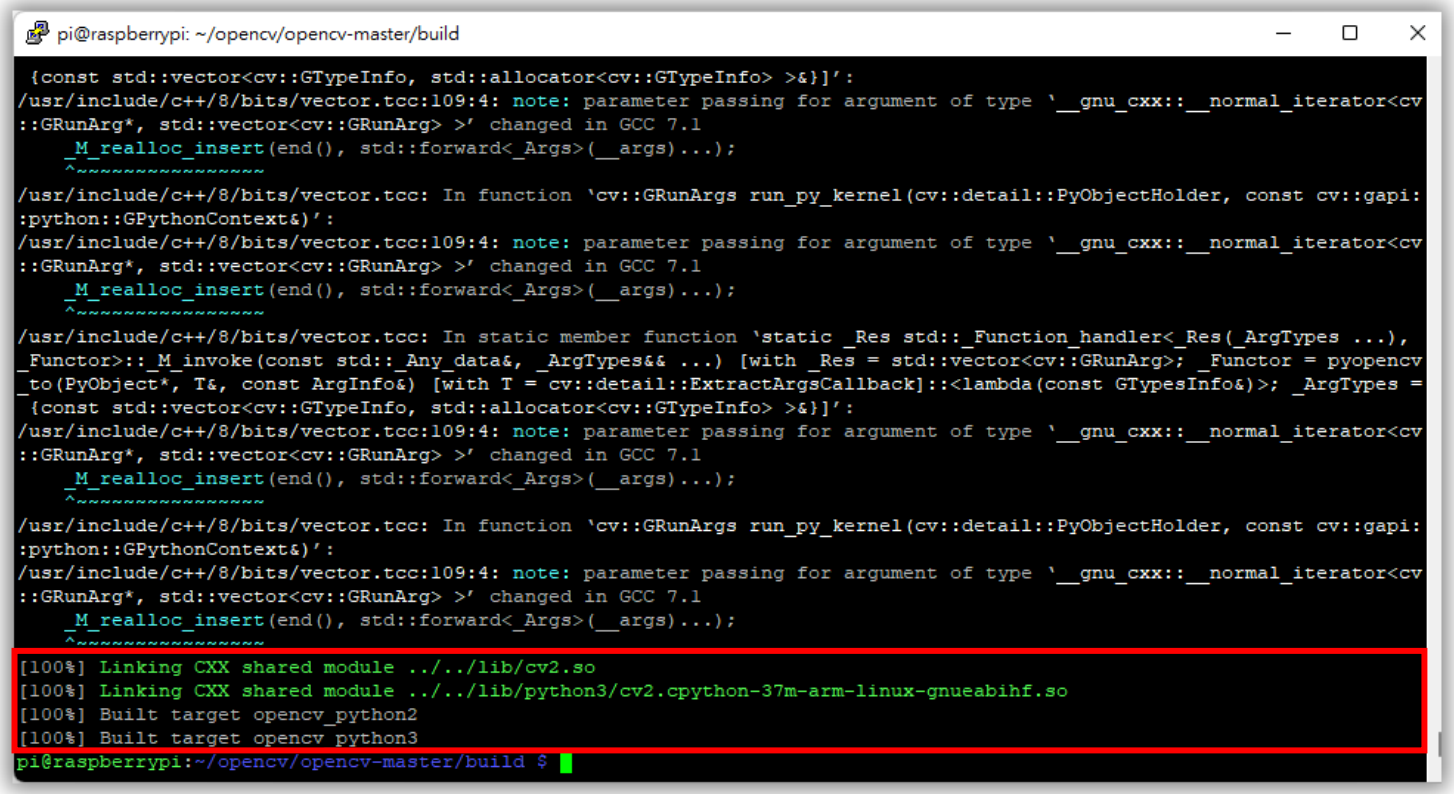
- 再來確認 Python 2 與 Python 3 這兩個區段是否存在，以及最後的生成結果訊息，看看是否有錯誤訊息。



在 Raspberry Pi上 編譯 OpenCV (3/4)

3. 所需的編譯文件生成後，繼續編譯 OpenCV。使用參數 -j4 使 Pi 上的每個內核都可以編譯 OpenCV，加快編譯過程。

```
$ make -j4
```



編譯會花費大量時間。在我的 Pi 4 上，此過程耗時約 1 個小時

在 Raspberry Pi上 編譯 OpenCV (4/4)

4. 若編譯結果沒有錯誤發生，就可以安裝了。

```
$ sudo make install
```

5. 我們還需要重新設定作業系統。

```
$ sudo ldconfig
```

6. 進入 python_loader

```
$ cd python_loader
```

7. 設定 python (python2) 的執行環境 (setup.py 是幫你寫好的設定檔)

```
$ sudo python setup.py install
```

8. 設定 python3 的執行環境 (setup.py 是幫你寫好的設定檔)

```
$ sudo python3 setup.py install
```

安裝 OpenCV 步驟

- 安裝 OpenCV 軟件包
- 準備編譯 OpenCV
- 在 Raspberry Pi 上 編譯 OpenCV
- 編譯後的清理
- 在 Raspberry Pi 上測試 OpenCV
- 常見問題

編譯後的清理（若 SD 卡夠大，不用改回）

1. 現在我們已經完成了 OpenCV 的編譯，不再需要這麼大的 swap 。

   ```
   $ sudo nano /etc/dphys-swapfile
   ```

2. 再次更改 swap 大小，並存檔：

   ```
   CONF_SWAPSIZE=2048
   ```

   ```
   CONF_SWAPSIZE=100
   ```

3. 重新啟動其服務。

   ```
   $ sudo systemctl restart dphys-swapfile
   ```

安裝 OpenCV 步驟

- 安裝 OpenCV 軟件包
- 準備編譯 OpenCV
- 在 Raspberry Pi 上 編譯 OpenCV
- 編譯後的清理
- 在 Raspberry Pi 上測試 OpenCV
- 常見問題

在 Raspberry Pi 上測試 OpenCV

ex4-14

1. 啟動 Python 終端：

   ```
   $ python3
   ```

2. 通過匯入模組，我們可以首先檢查 OpenCV 是否匯入到 Pi 上：

   ```
   >>> import cv2      # 執行後沒有任何反應，代表匯入成功
   ```

3. 查詢 OpenCV 版本：

   ```
   >>> cv2.__version__
   ```

這樣就代表你已經成功將 OpenCV 安裝在 Raspberry Pi 裡面了。接下來你可以盡情使用 OpenCV 進行各種各樣的開發工作了。

安裝 OpenCV 步驟

- 安裝 OpenCV 軟件包
- 準備編譯 OpenCV
- 在 Raspberry Pi 上 編譯 OpenCV
- 編譯後的清理
- 在 Raspberry Pi 上測試 OpenCV
- 常見問題

OpenCV 基本操作

目的：基本 OpenCV 處理圖片與影像

程式碼下載

```
$ cd ~/Lab
$ wget https://max543.com/debugger/class/python02/PiCamera/Lab/code/ch04.zip
$ unzip ch04.zip
```

準備工作

ex4-15

1. 進入 opencv 的虛擬環境：(請確認你有建立 opencv 的虛擬環境)

```
$ workon opencv
$ mkdir ~/Lab       # 建立一個資料夾，存放範例程式
$ cd Lab
$ mkdir ./ch04      # 建立一個資料夾，存放章節程式
$ cd ch04
```

2. 開始撰寫 Python 程式碼，進行 OpenCV 的應用。

4-3 OpenCV 的基本使用

4-3-1 OpenCV 圖片處理

4-3-2 OpenCV 影片處理

4-3-3 OpenCV 網路攝影機操作

4-3 OpenCV 的基本使用

- 當成功在樹莓派安裝 OpenCV 後，我們就可以使用 OpenCV 函式庫來進行圖片處理、影片處理和 Webcam 網路攝影機操作 (一樣適用在樹莓派的相機模組)。
- 在 Python 程式使用 OpenCV 圖片處理需要匯入 cv2，如下所示：

```
>>> import cv2
```

讀取與顯示圖片

ex4-16

- Python 程式是呼叫 OpenCV 的 imread() 方法讀取圖片，和 imshow() 方法顯示圖片，如下所示：

4-1 讀取與顯示圖片。(4-1.py)

```
import cv2

img = cv2.imread("koala.jpg")
cv2.imshow("Koala", img)

gray_img = cv2.imread("koala.jpg", cv2.IMREAD_GRAYSCALE)
cv2.imshow("Koala:gray", gray_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
cv2.IMREAD_COLOR：彩色，預設值。
cv2.IMREAD_UNCHANGED：沒有改變，包含透明度的圖片內容。
cv2.IMREAD_GRAYSCALE：灰階。
```

取得圖片資訊

- 我們可以使用 shape 屬性取得圖片資訊的尺寸和色彩數，例如：分別讀取成彩色和灰階圖片後，顯示圖片資訊，如下所示：

4-1a 取得圖片資訊。(4-1a.py)

```
import cv2

img = cv2.imread("koala.jpg")
img2 = cv2.imread("koala.jpg", cv2.IMREAD_GRAYSCALE)
print(img.shape)
print(img2.shape)
h, w, c = img.shape
print("圖片高:", h)
print("圖片寬:", w)
```

調整圖片尺寸

ex4-18

- OpenCV 的 cv2.resize() 在調整圖片尺寸時，會改變圖片的長寬比例，我們準備改用 imutils 模組的方法來調整圖片尺寸。
  - ✓ 然後呼叫 imutils.resize() 方法調整圖片尺寸，如下所示：

4-1b 調整圖片尺寸。(4-1b.py)

```
import cv2, imutils

img = cv2.imread("koala.jpg")
print(img.shape)
resized_img = imutils.resize(img, width = 300)
print(resized_img.shape)
cv2.imshow("Koala", img)
cv2.imshow("Koala:resized", resized_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

剪裁圖片

ex4-19

- 在 OpenCV 使用 imread() 方法讀取的圖片內容是一個 Numpy 陣列，剪裁圖片就是在切割 Numpy 陣列，如下所示：

4-1c 剪裁圖片。(4-1c.py)

```
import cv2

img = cv2.imread("koala.jpg")
cv2.imshow("Koala", img)
print(img.shape)
x = 10; y = 10
w = 150; h= 200
crop_img = img[y:y + h, x:x + w]
cv2.imshow("crop_Koala", crop_img)
print(crop_img.shape)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

旋轉、翻轉和位移圖片

ex4-20

- OpenCV 沒有旋轉和位移圖片的方法 (只有 cv2.flip() 方法來翻轉圖片)，需自行運算來旋轉和位移圖片，在 imutils 提供有相關方法來旋轉和位移圖片。

4-1d 剪裁圖片。(4-1d.py)

```
import cv2, imutils

img = cv2.imread("koala.jpg")
rotated_img = imutils.rotate(img, angle = 90)
fliped_img = cv2.flip(img, -1)
translated_img = imutils.translate(img, 25, -75)
cv2.imshow("Koala:rotated", rotated_img)
cv2.imshow("Koala:fliped", fliped_img)
cv2.imshow("Koala:translated", translated_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

BGR 圖片和轉換成灰階

ex4-21

- OpenCV 讀取圖片後，可以呼叫 cvtColor() 方法轉換彩色圖片成為灰階圖片，方法的第 2 個參數是 cv2.COLOR_BGR2GRAY。
  - ✓ 圖片是 RGB 格式，可以使用參數 cv2.COLOR_RGB2BGR，將 RGB 轉換成 BGR (同理，如果需要 RGB 格式，可以使用參數 cv2.COLOR_BGR2RGB 將 BGR 轉換成 RGB)。

4-1e 轉換成灰階和 BGR 圖片。(4-1e.py)

```
import cv2

img = cv2.imread("koala.jpg")
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
bgr_img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
cv2.imshow("Koala:gray", gray_img)
cv2.imshow("Koala:bgr", bgr_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

imread() 與 imshow() 方法都是使用 BGR 格式，所以不需要特殊處裡。

從 URL 取得圖片

ex4-22

- 在 imutils 提供 url_to_image() 方法，可以讓我們直接從網路讀取圖檔內容，如下所示：

4-1f 從 URL 取得圖片。(4-1f.py)

```
import cv2, imutils

url = "https://fchart.github.io/img/koala.png"
img = imutils.url_to_image(url)
cv2.imshow("Koala", img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

註記圖片

ex4-23

- 註記圖片就是在圖片上繪圖，我們可以在圖片上畫線、畫長方形、畫圓形、畫橢圓形和加上文字內容，如下所示：

4-1g 註記圖片。(4-1g.py)

```
import cv2

img = cv2.imread("koala.jpg")

cv2.line(img, (0, 0), (200, 200), (0, 0, 255), 5)
cv2.rectangle(img, (20, 70), (120, 160), (0, 255, 0), 2)
cv2.rectangle(img, (40, 80), (100, 140), (255, 0, 0), -1)
cv2.circle(img, (90, 210), 30, (0, 255, 255), 3)
cv2.circle(img, (140, 170), 15, (255, 0, 0), -1)
cv2.putText(img, "OpenCV", (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 5,
cv2.LINE_AA)

cv2.imshow("Koala", img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

寫入圖片

ex4-24

- OpenCV 可以呼叫 imwrite() 方法將圖片內容寫入圖檔，如下所示：

4-1h 寫入圖片。(4-1h.py)

```
import cv2

img = cv2.imread("koala.jpg")
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imwrite("result_gray.jpg", gray_img)
cv2.imwrite("result.png", img)
```

4-3 OpenCV 的基本使用

4-3-1 OpenCV 圖片處理

4-3-2 OpenCV 影片處理

4-3-3 OpenCV 網路攝影機操作

OpenCV 影片處理

- 影片事實上就是一種動態影像，這是一連串連續的靜態影像圖片所組成，每一個靜態影像稱為『影格』(Frame，或稱幀)，每秒播放的靜態影像圖片數稱為『影格率』(Frame per Second，或稱幀率)。
- OpenCV 支援讀取和播放影片檔案，我們可以取得影片資訊和播放出彩色、灰階黑白內容的影片。

播放影片檔

ex4-25

- Python 是建立 OpenCV 的 VideoCapture 物件來播放影片檔。

4-2 播放影片檔。(4-2.py)

```
import cv2

cap = cv2.VideoCapture("YouTube.mp4")

while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        cv2.imshow("frame", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

0xFF 是十六進制常數，二進制值為 11111111。這個寫法只留下原始的最後 8 位，和後面的 ASCII 碼對照，此處是為了防止 BUG。

取得影片資訊

ex4-26

- 在 Python 程式建立 VideoCapture 物件後，我們可以取得影片的尺寸和編碼的影片資訊，如下所示：

4-2a 取得影片資訊。(4-2a.py)

```
import cv2

cap = cv2.VideoCapture("YouTube.mp4")

def decode_fourcc(v):
    v = int(v)
    return "".join([chr((v >> 8 * i) & 0xFF) for i in range(4)])

width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
print("圖片尺寸:", width, 'x', height)
fourcc = cap.get(cv2.CAP_PROP_FOURCC)
codec = decode_fourcc(fourcc)
print("Codec編碼:", codec)
```

參數用法：
https://docs.opencv.org/4.5.3/d4/d15/group__videoio__flags__base.html#baeb8dd9c89c10a5c63c139bf7c4f5704d。

影片處理顯示灰階影片

ex4-27

- 我們只需使用 4-1e.py 的方法來處理圖片，就可以播放出灰階的黑白影片，如下所示：

4-2b 影片處理顯示灰階影片。(4-2b.py)

```
import cv2

cap = cv2.VideoCapture("YouTube.mp4")

while cap.isOpened():
    ret, frame = cap.read()

    if ret:
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        cv2.imshow("frame", gray_frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

取得網路攝影機的影像

- 在 OpenCV 的 VideoCapture 物件除了開啟影片檔案，也可以開啟攝影機：

4-3 取得網路攝影機的影像。(4-3.py)

```
import cv2

cap = cv2.VideoCapture(0)

while cap.isOpened():
    ret, frame = cap.read()
    frame = cv2.rotate(frame, rotateCode = 1)
    cv2.imshow("frame", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

更改影像的解析度

- 在 Python 程式建立 VideoCapture 物件後，可以呼叫 set 方法更改影片的寬、高和影格率：

4-3a 更改影像的解析度。(4-3a.py)

```
import cv2

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 180)
cap.set(cv2.CAP_PROP_FPS, 25)

while cap.isOpened():
    ret, frame = cap.read()
    cv2.imshow("frame",frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

將影像寫入影片檔案

ex4-30

- OpenCV 可以建立 VideoWrite 物件來寫入影片檔案：

4-3b 將影像寫入影片檔案。(4-3b.py)

```
import cv2

cap = cv2.VideoCapture(0)
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter("output.avi", fourcc, 20, (640, 480))
while cap.isOpened():
    ret, frame = cap.read()
    if ret == True:
        out.write(frame)
        cv2.imshow("frame", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
out.release()
cv2.destroyAllWindows()
```

編碼名稱	編碼字串	影片副檔名
YUV	*'I420'	.avi
MPEG-I	*'PIMT'	.avi
MPEG-4	*'XVID'	.avi
MP4	*'MP4V'	.mp4
Ogg Vorbis	*'THEO'	.ogv

OpenCV 人臉辨識

目的：OpenCV 內建演算法進行人臉辨識

人臉偵測

- OpenCV 內建的是哈爾特徵 (Haar-like feature) 演算法。
  - ✓ 速度快，非類神經網路，適合低階的樹莓派。
  - ✓ 使用哈爾特徵前，先下載人臉偵測的聯集分類器。
- 分類器的位置，在家目錄底下的 OpenCV-master 原始碼資料夾
  - ✓ ~/opencv/opencv-master/data/haarcascades (自行編譯才會有)
  - ✓ 或手動下載：https://github.com/opencv/opencv/tree/master/data/haarcascades
- 以下是跟人臉有關的：

分類器	說明
haarcascade_frontalface_default.xml	人臉正面與側面
haarcascade_frontalface_alt2.xml	人臉正面效果比較好
haarcascade_profileface.xml	人臉側面效果比較好
haarcascade_eye.xml	偵測眼睛

圖片的人臉辨識

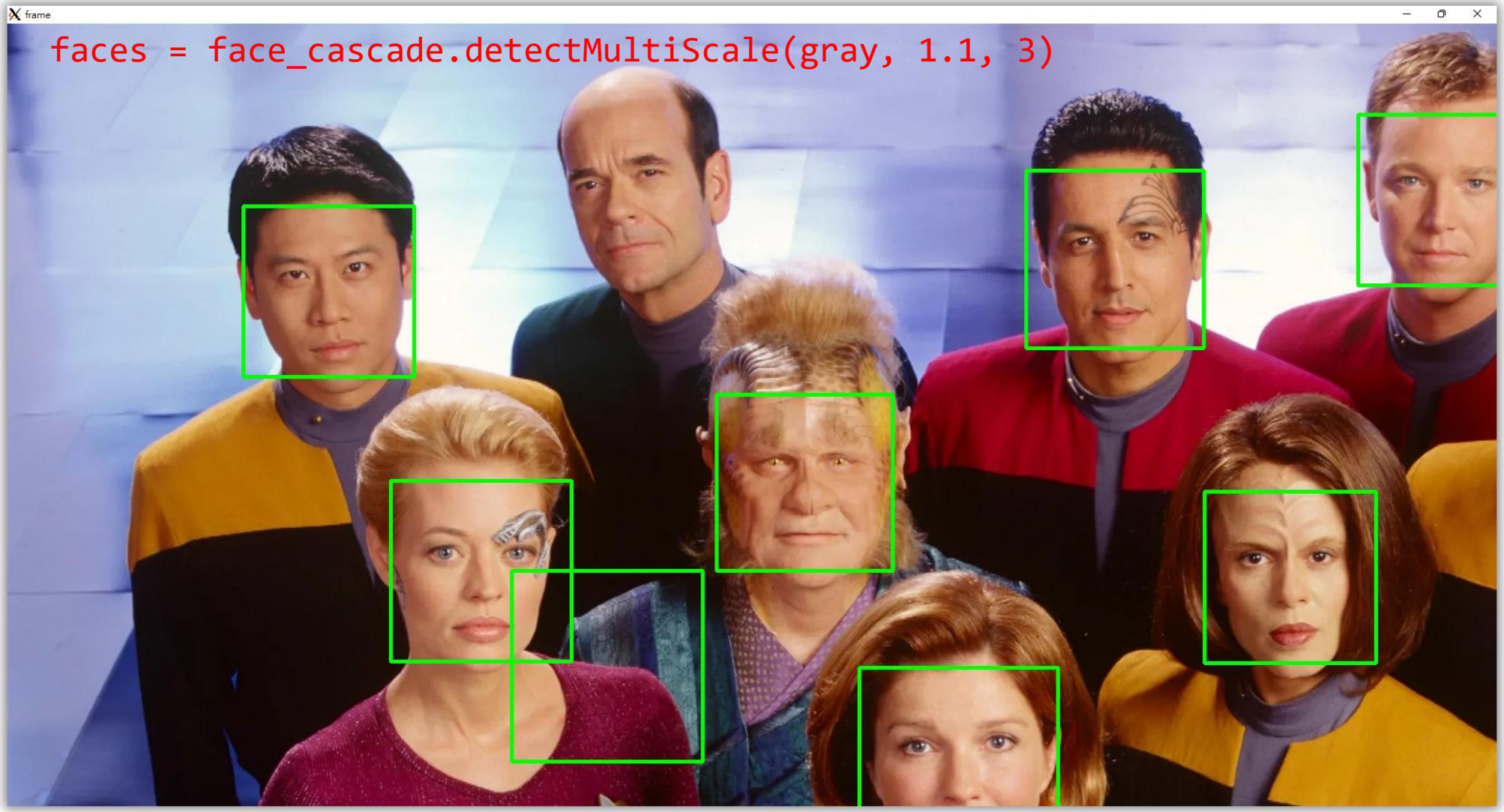
ex4-31

4-4 圖片的人臉辨識。(4-4.py)

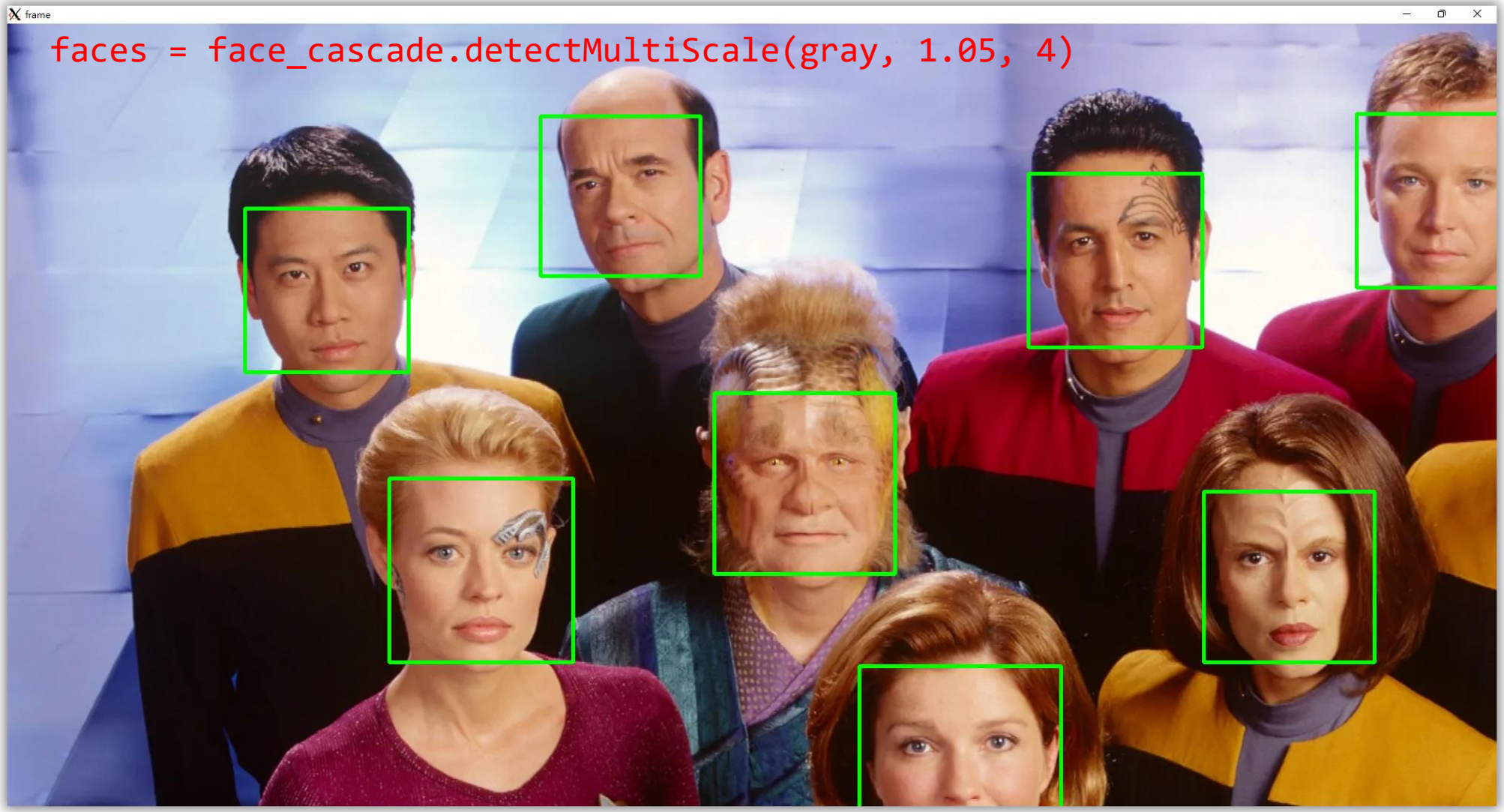
```
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt2.xml')
image = cv2.imread('demo.jpeg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.1, 3)
# faces = face_cascade.detectMultiScale(gray, 1.05, 4)
for (x, y, w, h) in faces:
    frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)

cv2.namedWindow('frame', cv2.WINDOW_NORMAL)
cv2.imshow('frame', frame)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

cv2.WINDOW_NORMAL：視窗大小可改變
cv2.WINDOW_AUTOSIZE：視窗大小不可改變
cv2.WINDOW_FREERATIO：自適應比例
cv2.WINDOW_KEEPRATIO：保持比例

當有人臉沒被辨識，需要調整參數（4-4.py）



調整參數後（4-4a.py）

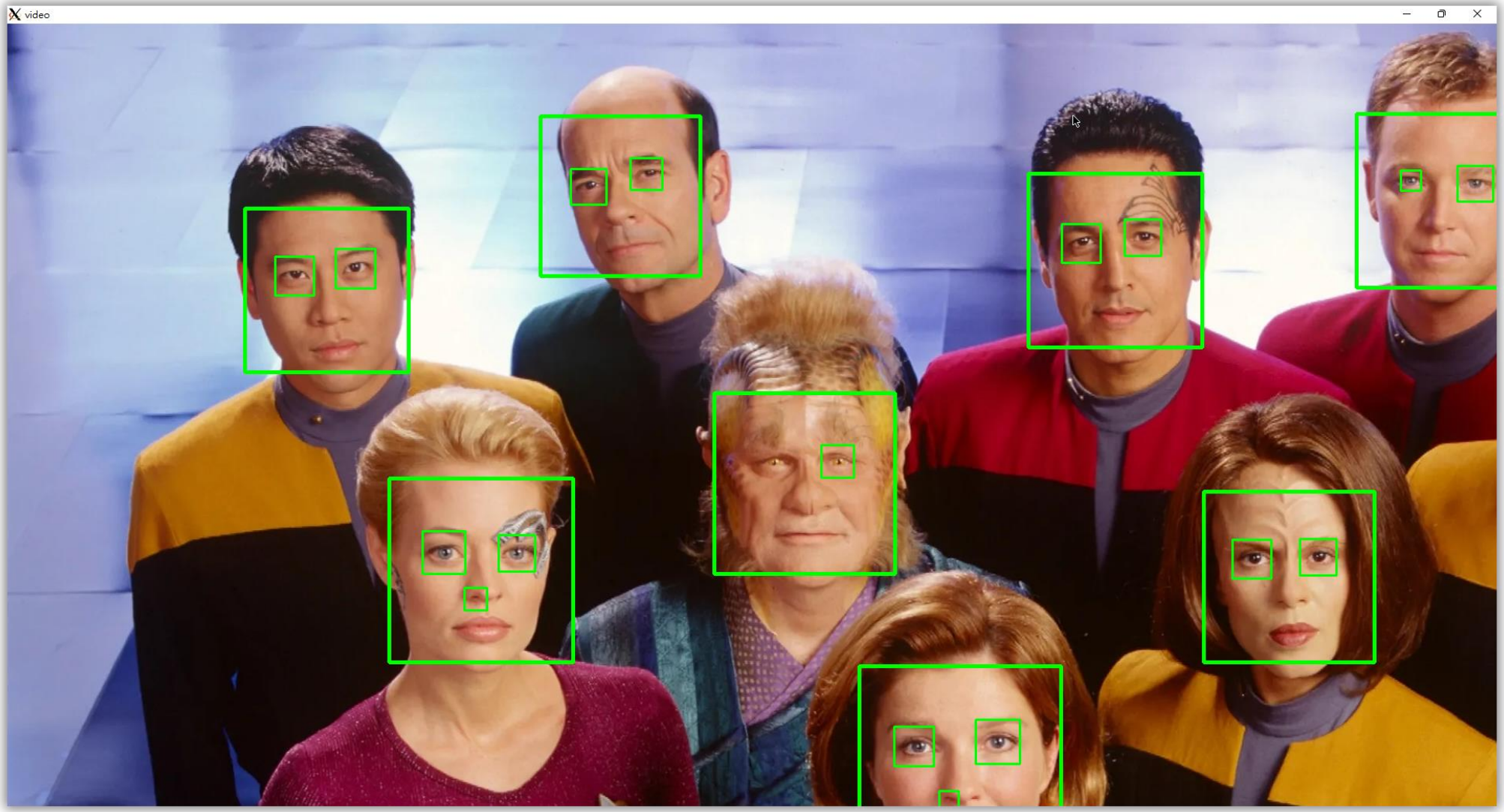


圖片的眼睛辨識

4-5 圖片人臉與眼睛辨識。(4-5.py)

```
import cv2
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt2.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
frame = cv2.imread('demo.jpeg')
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
# faces = face_cascade.detectMultiScale(gray, 1.1, 3)
faces = face_cascade.detectMultiScale(gray, 1.05, 4)
for (x, y, w, h) in faces:
    frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)
    face_rect = gray[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(face_rect, 1.3, 8)
    for (ex, ey, ew, eh) in eyes:
        frame = cv2.rectangle(frame, (x + ex, y + ey), (x + ex + ew, y + ey + eh), (0, 255, 0),
2)


cv2.namedWindow('frame', cv2.WINDOW_NORMAL)
cv2.imshow('frame', frame)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Demo 4-5.py



OpenCV 即時影像人臉辨識

目的：進行即時人臉辨識

即時影像的人臉辨識

ex4-33

4-6 即時影像的人臉辨識。(4-6.py)

```
import cv2

faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

while True:
    ret, frame = cap.read()
    frame = cv2.rotate(frame, rotateCode = 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, scaleFactor = 1.1, minNeighbors = 5, minSize =
(30, 30))
    print("人臉數:", len(faces))

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    cv2.imshow("preview", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

即時影像的人臉與眼睛辨識

4-7 即時影像的人臉與眼睛辨識。(4-7.py)

```
import cv2

faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

while True:
    ret, frame = cap.read()
    frame = cv2.rotate(frame, rotateCode = 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, scaleFactor = 1.1, minNeighbors = 5, minSize = (30, 30))
    print("人臉數:", len(faces))

    for (x, y, w, h) in faces:
        frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)
        face_rect = gray[y:y + h, x:x + w]
        eyes = eye_cascade.detectMultiScale(face_rect, 1.3, 8)

        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(frame, (x + ex, y + ey), (x + ex + ew, y + ey + eh), (0, 255, 0), 2)

    cv2.imshow("preview", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

實驗 4-3：特定人臉辨識

目的：OpenCV 進行特定人臉辨識

複習 Python 的資料結構（容器）

目的：了解程式計算數值會選擇適合的容器

Python 的基本資料結構

ex4-35

- 字串容器：由字元組成。例如：`string = "52python"`。
- tuple 容器：由資料物件組成。例如：`tuple = (1, '2', 3)`。
- 串列容器：由資料物件組成。例如：`list = [1, '2', 3]`。
- 集合容器：由資料物件組成。例如：`set = {1, '2', 3}`。
- 字典容器：由資料物件組成，以鍵：值表示。例如：`dick = {'A':1, 'B':'2', 'C':3}`。

Python 的第三方資料結構：numpy.array

ex4-36

- Numpy：Python 的擴充模組，常用於資料處理。

```
>>> import numpy as np    # 匯入 numpy
>>> a = np.array([10, 2, 45, 32, 24])     # 建立五個元素
>>> len(a)      # 計算元素個數
>>> a[2:4]      # 取出第 2、3 個元素
>>> a[:4]       # 取出第 1 ~ 3 個元素
```

特定人臉偵測

- 藉由專門分析人臉特徵的演算法找出的特徵值，然後跟已經儲存的特徵值比對，判定是誰的臉。
- 分為三階段：取樣、訓練、辨識。
  - ✓ 取樣：收集訓練用的人臉圖片，建議每人有 100 張才有比較好的準確率。
  - ✓ 訓練：OpenCV 提供三種演算法，Eigen、Fisher、LBPH。
  - ✓ 辨識：依照訓練完的資料，判斷目前攝影機看到的人臉屬於哪一位。

Step 1：取樣

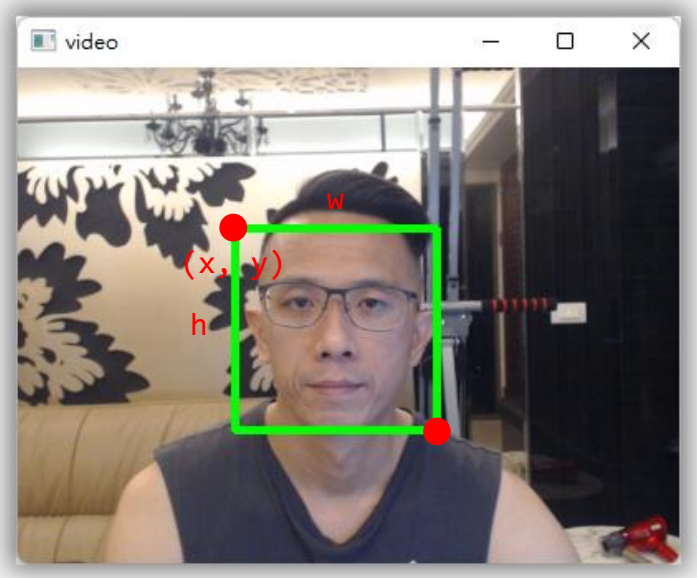
- 取樣階段：我們透過程式碼來自動抓取 100 張人臉圖片，這樣比用相機拍攝 100 張圖片要來的方便快速。

取樣程式

4-8-1 人臉取樣。(4-8-1-capture.py)

```
import cv2

ESC = 27
n = 1
index = 0
total = 100

def saveImage(face_image, index):
    filename = 'images/h0/{:03d}.pgm'.format(index)
    cv2.imwrite(filename, face_image)
    print(filename)

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) / cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)
cv2.namedWindow('video', cv2.WINDOW_NORMAL)

while n > 0:
    ret, frame = cap.read()
    frame = cv2.resize(frame, (WIDTH, HEIGHT))
```

```
frame = cv2.rotate(frame, rotateCode = 1)
frame = cv2.flip(frame, 1)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.1, 3)

for (x, y, w, h) in faces:
    frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)
    if n % 5 == 0:
        face_img = gray[y: y + h, x: x + w]
        face_img = cv2.resize(face_img, (400, 400))
        saveImage(face_img, index)
        index += 1
        if index >= total:
            print('get training data done')
            n = -1
            break
    n += 1

cv2.imshow('video', frame)

if cv2.waitKey(1) == ESC:
    cv2.destroyAllWindows()
    break
```



補充：矩陣 (Matrix)

```python
import numpy as np

A = np.array([[1, 4, 2], [3, 2, 0]])
B = A[1:, 1:3]     # 2-D array 切片，第一個索引以列切片，第二個索引以行為切片
```

列 (row) →

$$A = \begin{bmatrix} 1 & 4 & 2 \\ 3 & 2 & 0 \end{bmatrix}_{2\times 3}$$

↓ 列 (row)

$$B = \begin{bmatrix} 2 & 0 \end{bmatrix}_{1\times 2}$$

Step 2：訓練

- 訓練階段：將 h0、h1… 資料夾中的圖片取出、標籤化後送進人臉特徵演算法中計算特徵值，並將結果存檔後以供後續使用。

訓練程式

4-8-2 訓練取樣集。(4-8-2-train.py)

```
import cv2
import numpy as np

images = []
labels = []
for index in range(100):
    filename = 'images/h0/{:03d}.pgm'.format(index)
    print('read ' + filename)
    img = cv2.imread(filename, cv2.COLOR_BGR2GRAY)
    images.append(img)
    labels.append(0)    # 第一張人臉的標籤為 0

print('training...')
model = cv2.face.LBPHFaceRecognizer_create()          # 建立模型，使用 LBPH 演算法
model.train(np.asarray(images), np.asarray(labels))  # train() 只能接收 numpy 格式的陣列
model.save('faces.data')                              # 儲存訓練好的辨識檔
print('training done')
```

Step 3：辨識

- 辨識階段：這一階段是拿訓練好的結果，來驗證辨識效果是否準確，所以我們開啟攝影機，將攝影機拍到的人臉來跟訓練結果比對，如果發現有『認識』的人臉，就在畫面上顯示這個人臉的名字。

辨識

4-8-3 人臉辨識。(4-8-3-recognition.py)

```
import cv2

model = cv2.face.LBPHFaceRecognizer_create()    # 建立模型，使用 LBPH 演算法。
model.read('faces.data')                        # 載入訓練好的辨識檔。
print('load training data done')
# 載入聯集分類器 (需與取樣時的分類器一致)，並開啟攝影機。
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
cv2.namedWindow('video', cv2.WINDOW_NORMAL)
# 可識別化名稱
names = ['pllai']
# 讀取攝影機資料，並轉換成灰階影像。
while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame, (600, 400))
    frame = cv2.flip(frame, 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
faces = face_cascade.detectMultiScale(gray, 1.1, 3)
for (x, y, w, h) in faces:
    frame = cv2.rectangle(
        frame,
        (x, y), (x + w, y + h),
        (0, 255, 0), 3
    )
    face_img = gray[y: y + h, x: x + w]
    face_img = cv2.resize(face_img, (400, 400))

    val = model.predict(face_img)
    print('label:{}, conf:{:.1f}'.format(val[0], val[1]))
    if val[1] < 50:
        cv2.putText(
            frame, names[val[0]], (x, y - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 3
        )

cv2.imshow('video', frame)
if cv2.waitKey(1) == 27:
    cv2.destroyAllWindows()
    break
```

特定人臉辨識