

OPENCV in Python

電腦視覺與人臉辨識入門教學

吳智鴻 王啟勳

國立臺中教育大學 數位內容科技學系

2024

大綱

- Opencv概說
- 在windows下安裝opencv
- 檢查是否安裝
- 程式一:讀檔秀檔
- 程式二:啟動鏡頭
- 程式三:按按鍵擷取圖檔
- 程式四:靜態人臉辨識
- 程式五:動態人臉辨識

Opencv概說

OpenCV的全稱是Open Source Computer Vision Library，是一個跨平台的電腦視覺庫。OpenCV是由**英特爾**公司發起並參與開發，以BSD授權條款授權發行，可以在商業和研究領域中免費使用。OpenCV可用於開發即時的圖像處理、電腦視覺以及圖型識別程式。該程式庫也可以使用英特爾公司的IPP進行加速處理。

Opencv概說

1. OpenCV可用於解決如下領域的問題：
2. 擴增實境、臉部辨識、手勢辨識、人機互動、動作辨識、運動跟蹤、物體辨識、圖像分割、機器人。
3. OpenCV用C++語言編寫，它的主要介面也是C++語言，但是依然保留了大量的C語言介面。也有大量的Python,Java and MATLAB/OCTAVE (版本2.5)的介面。
4. OpenCV可以在Windows、Linux、Android、Maemo、FreeBSD、OpenBSD、iOS、和Mac OS等平台上執行。

人臉辨識原理

Haar串聯分類器

1 串聯分類器

大部分的情況下，分類器需要對多個影像特徵進行識別。舉例來說，識別一個動物到底是狗（正類別）還是其他動物（負類別），我們可能需要根據多個條件進行判斷，這樣比較下來是非常煩瑣的。但是，如果首先就比較它們有幾條腿：

- 有「四條腿」的動物被判斷為「可能為狗」，並對此範圍內的物件繼續進行分析和判斷。
- 沒有「四條腿」的動物直接被否決，即不可能為狗。

這樣，僅比較腿的數目，根據這個特徵就能排除樣本集中大量的負類別（例如雞、鴨、鵝等不是狗的其他動物實例）。

串聯分類器就是以這種想法，將多個簡單為基礎的分類器按照一定的順序串聯而成的。

串聯分類器的基本原理如圖 23-1 所示。

串聯分類器的優勢是，在開始階段僅進行非常簡單的判斷，就能夠排除明顯不符合要求的實例。在開始階段被排除的負類別，不再參與後續分

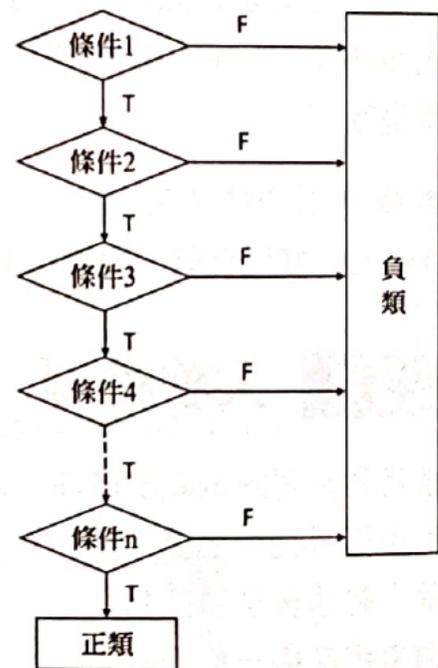


圖 23-1 串聯分類器示意圖

Haar串聯分類器

128	108	47	27
96	76	88	68

圖 23-2 影像範例

用兩幅影像左側像素值之和減去右側像素值之和：

- 針對左圖， $\text{sum}(\text{左側像素}) - \text{sum}(\text{右側像素}) = (128+96) - (108+76) = 40$
- 針對右圖， $\text{sum}(\text{左側像素}) - \text{sum}(\text{右側像素}) = (47+88) - (27+68) = 40$

這兩幅影像中，「左側像素值之和」減去「右側像素值之和」都是 40。所以，可以認為在「左側像素值之和」減去「右側像素值之和」這個角度，這兩幅影像具有一定的相關性。

進一步擴充，我們可以從更多的角度考慮影像的特徵。學者 Papageorgiou 等人提出了如圖 23-3 所示的 Haar 特徵，這些特徵包含垂直特徵、水平特徵和對角特徵。他們利用這些特徵分別實現了行人檢測 (*Pedestrian Detection Using Wavelet Templates*) 和人臉檢測 (*A General Framework For Object Detection*)。

-1	1
----	---

-1	-1
1	1

1	-1
-1	1

圖 23-3 Haar 特徵

Haar擴充特徵

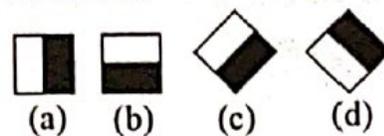
4個邊特徵

8個線特徵

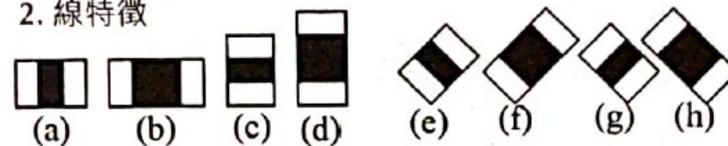
2個中心點特徵

1個對角特徵

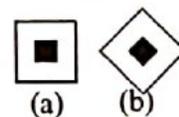
1. 邊特徵



2. 線特徵



3. 中心點特徵



4. 對角特徵



OpenCV串聯分類支援的類型

其中，Harr 串聯分類器多達 20 多種（隨著版本更新還會繼續增加），提供了對多種物件的檢測功能。部分串聯分類器如表 23-1 所示。

表 23-1 串聯分類器

XML 檔案名稱	串聯分類器類型
harcascade_eye.xml	眼睛檢測
haarcascade_eye_tree_eyeglasses.xml	眼鏡檢測
haarcascade_mcs_nose.xml	鼻子檢測
haarcascade_mcs_mouth.xml	嘴巴檢測
harcascade_smile.xml	表情檢測
hogcascade_pedestrians.xml	行人檢測
lbpcascade_frontalface.xml	正面人臉檢測
lbpcascade_profileface.xml	人臉檢測
lbpcascade_silverware.xml	金屬檢測

載入串聯分類器的語法格式為：

```
<CascadeClassifier object> = cv2.CascadeClassifier( filename )
```

建立環境(非常重要)

Python 版本

- 建立一個虛擬環境

- `conda create -n opencv python==3.8 jupyter`
- `conda activate opencv`
- `pip install opencv-python`

- 使用本機的jupyter notebook來開發

- Colab (雲端開圖片需另外處理)
- VS code安裝比較麻煩

如果遇到套件不相容，需指定python版本為某一個版本例如3.8，才能與opencv相容

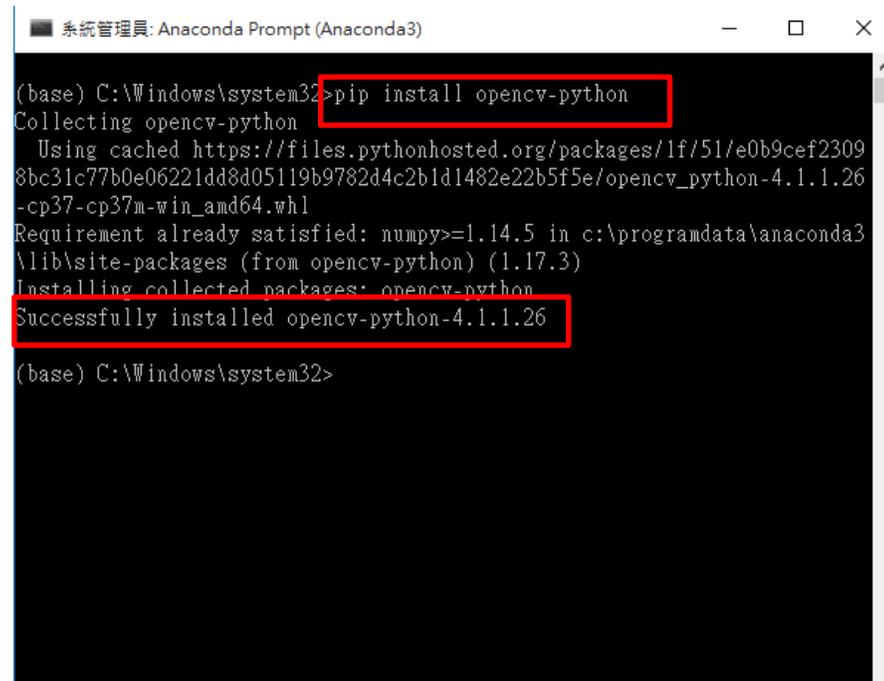
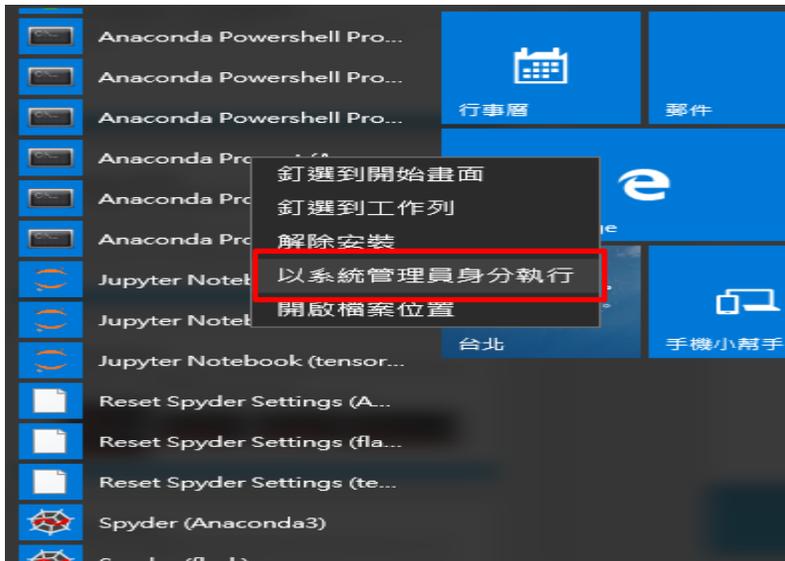
在windows下安裝opencv (方法二)

Step1:左下角開始圖示---打開ananconda資料夾

Step2:在anaconda prompt上按滑鼠右鍵---點選” 以系統管理員身分執行” ---點選確定

Step3:輸入pip install opencv-python

Step4:安裝成功



檢查是否安裝

打開jupyter,輸入程式碼:

```
import cv2          #引入opencv函式庫  
cv2.__version__    #輸出目前安裝的版本
```

```
In [2]: 1 import cv2  
        2 cv2.__version__
```

```
Out[2]: '4.1.1'
```

```
In [ ]: 1
```

程式一:讀檔秀檔 (prg1)

Step1:網路上找一張圖,將檔名改成英文

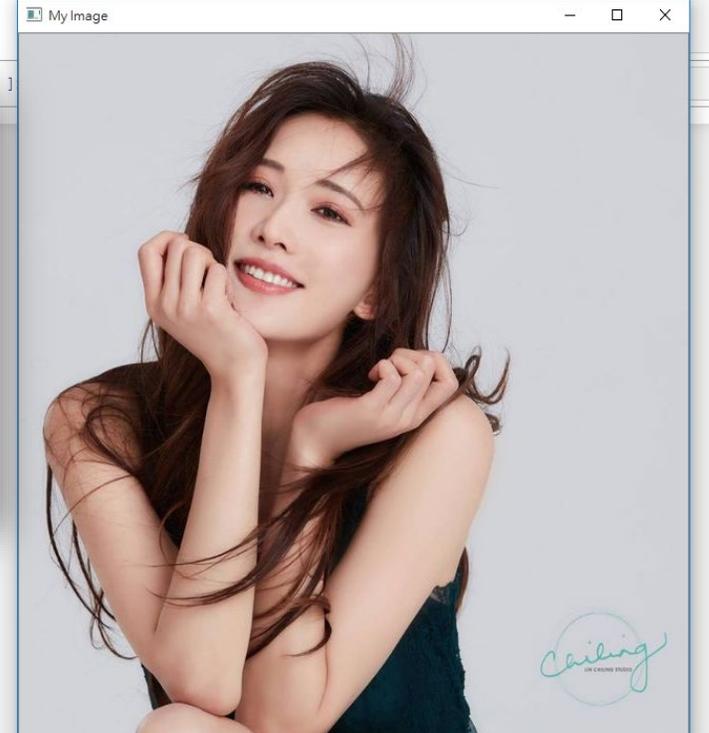
Stes2:打開Jupyter,輸入程式

```
import cv2  
cv2.__version__
```

```
'3.4.2'
```

```
img = cv2.imread('star1.png') → # 讀取圖檔  
cv2.imshow('My Image',img) # 顯示圖片,第一個參數是視窗的名字,第二個參數是圖檔的變數名稱  
cv2.waitKey(0) # 參數是等待時間(單位為毫秒),若設定為 0 就表示持續等待至使用者按下任按鍵  
cv2.destroyAllWindows() # 按下任意鍵後關閉所有 OpenCV 的視窗。
```

```
In [*]: 1 import cv2 #引入opencv函式庫  
2  
3 img = cv2.imread('lin.jpg') # 讀取圖檔  
4 cv2.imshow('My Image',img) # 顯示圖片,第一個參數是視窗的名字,第二個參數是圖檔的變數名稱  
5 cv2.waitKey(0) # 參數是等待時間(單位為毫秒),若設定為 0 就表示持續等待至使用者按下任按鍵  
6 cv2.destroyAllWindows() # 按下任意鍵後關閉所有 OpenCV 的視窗。
```



程式二:啟動鏡頭 (prg2)

Step2:輸入以下程式

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while(True):
    ret, frame = cap.read()#傳回值有兩個,第一個是否有讀到圖片,值為True或False,第二個是擷取當前的一幀圖片
    cv2.imshow('frame',frame)
    if cv2.waitKey(5) == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```



程式三:按按鍵擷取圖檔

Step:輸入程式

```
import cv2
cap = cv2.VideoCapture(0)
while(True):
    ret, frame = cap.read()
    cv2.imshow("frame", frame)
    if cv2.waitKey(1) == ord('q'):
        cv2.imwrite("test.png", frame) #儲存成test.png
        break

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



Test.png
儲存後結果

Exercise#1

Requirement:

將檔案另存成另外名稱。Ex. Adt106001.png

按 s 鍵儲存檔案。

按 q 鍵離開

程式四:靜態人臉辨識

需確認位置

```
import cv2
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml') # 載入分類器

img = cv2.imread('star1.png')# 轉成灰階圖片
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 需轉換成灰階

# 偵測臉部
faces = face_cascade.detectMultiScale(gray,scaleFactor=1.08,minNeighbors=5,minSize=(32, 32)) # 繪製人臉部份的方框
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2) #(0, 255, 0)欄位可以變更方框顏色(Blue,Green,Red)# 顯示成果

cv2.namedWindow('img', cv2.WINDOW_NORMAL) #正常視窗大小
cv2.imshow('img', img) #秀出圖片
cv2.imwrite( "result.jpg", img ) #保存圖片
cv2.waitKey(0) #等待按下任一按鍵
cv2.destroyAllWindows() #關閉視窗
```



Exercise#2

Requirement

把人臉方框印成紅色

找一個兩個人的照片，把faces印出來看看

把faces印出來看看

```
import cv2
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml') # 載入分類器

img = cv2.imread('star1.png')# 轉成灰階圖片
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 偵測臉部
faces = face_cascade.detectMultiScale(gray,scaleFactor=1.08,minNeighbors=5,minSize=(32, 32)) # 繪製人臉部份的方框
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2) #(0, 255, 0)欄位可以變更改方框顏色(Blue,Green,Red)# 顯示成果

cv2.namedWindow('img', cv2.WINDOW_NORMAL) #正常視窗大小
cv2.imshow('img', img) #秀出圖片
cv2.imwrite( "result.jpg", img ) #保存圖片
cv2.waitKey(0) #等待按下任一按鍵
cv2.destroyAllWindows() #關閉視窗
```

faces

```
array([[185, 69, 140, 140]], dtype=int32)
```

(x,y), (w, h)

誤判的情況(star2.png)

```
import cv2
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml') # 載入分類器

img = cv2.imread('star2.png')# 轉成灰階圖片
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 偵測臉部
faces = face_cascade.detectMultiScale(gray,scaleFactor=1.08,minNeighbors=5,minSize=(32, 32)) # 繪製人臉部份的方
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2) #(0, 255, 0)欄位可以變更方框顏色(Blue,Green,Red):

cv2.namedWindow('img', cv2.WINDOW_NORMAL) #正常視窗大小
cv2.imshow('img', img) #秀出圖片
cv2.imwrite( "result.jpg", img ) #保存圖片
cv2.waitKey(0) #等待按下任一按鍵
cv2.destroyAllWindows() #關閉視窗
```

faces

```
array([[425, 90, 100, 100],
       [730, 85, 106, 106],
       [735, 203, 121, 121],
       [ 89, 95, 117, 117]], dtype=int32)
```

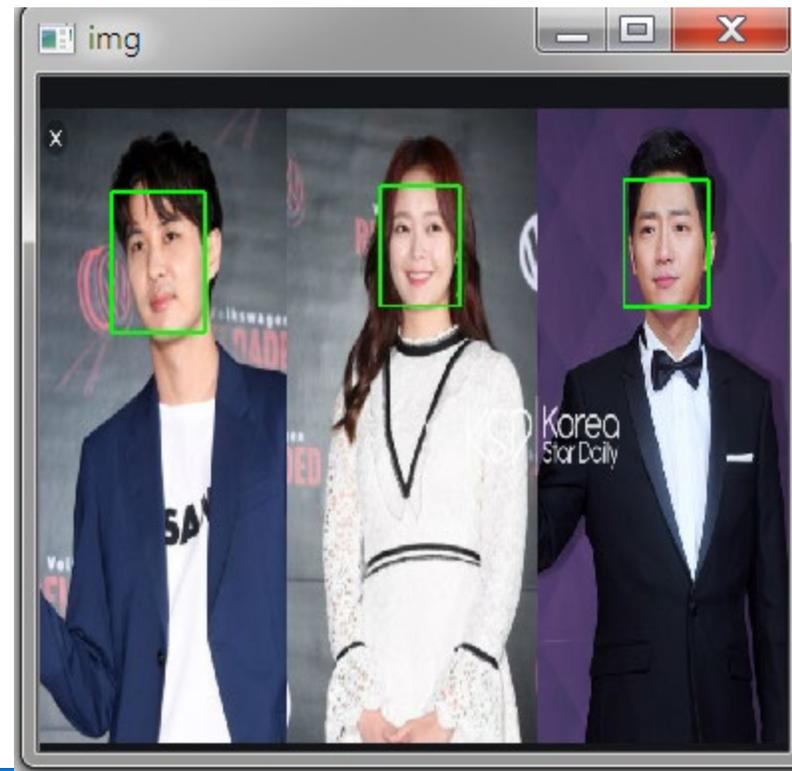


Exercise#3

修正誤判的狀況

上網搜尋一下參數的意義

試著調整參數，以獲得正確的結果



Face_cascade.detectMultiScale ()

參數介紹

上網搜尋一下參數的意義

透過修正參數，得到較正確的結果

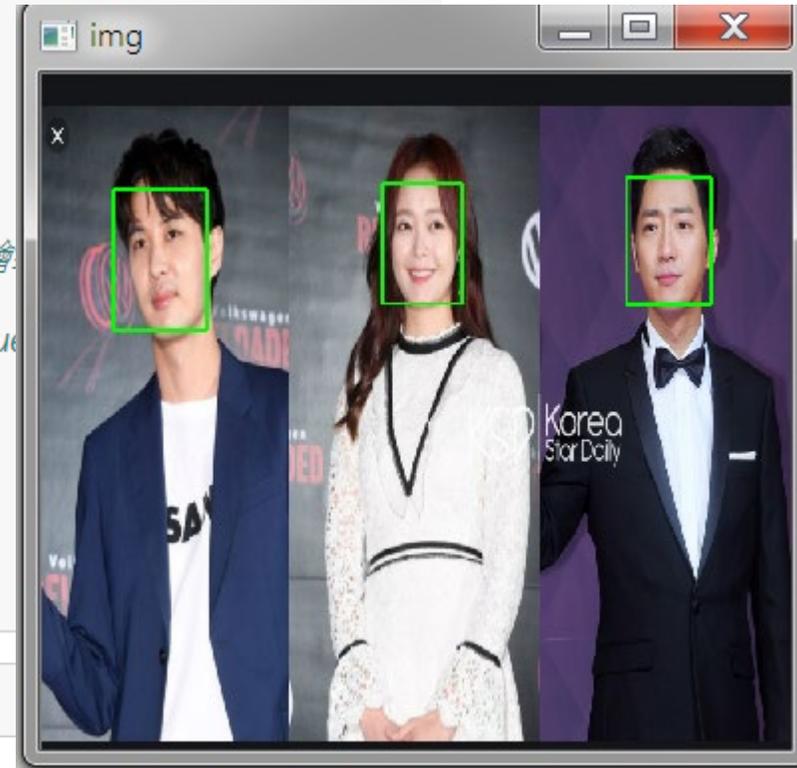
```
import cv2
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml') # 載入分類器

img = cv2.imread('star2.png')# 轉成灰階圖片
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 偵測臉部
faces = face_cascade.detectMultiScale(img, 1.1, 3, 5, cv2.CASCADE_SCALE_IMAGE) # 繪
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2) #(0, 255, 0)欄位可以變更方框顏色(Blue

cv2.namedWindow('img', cv2.WINDOW_NORMAL) #正常視窗大小
cv2.imshow('img', img) #秀出圖片
cv2.imwrite( "result.jpg", img ) #保存圖片
cv2.waitKey(0) #等待按下任一按鍵
cv2.destroyAllWindows() #關閉視窗
```

faces



Hint

調整 minNeighbors 數值 介於10~20之間

OpenCV的人臉檢測函數介紹

```
Faces = cv2.CascadeClassifier.detectMultiScale ( image, scaleFactor, minNeighbors, flags, minSize)
```

參數

1. faces回傳幾張臉 &座標 (x,y)(w,h)
2. image: 待檢測照片，通常為灰階影像
3. scaleFactor: 前後兩次相繼的掃描中，搜索視窗的縮放比例
4. minNeighbors: 組成檢測目標的相鄰矩形的最小個數。預測情況數值為3，表示有3個以上的檢測標記存在時，才認為人臉存在。如果希望加強檢測的準確率，可以將值設的更大，但可能會讓一些人臉無法被檢測到。
5. Flags: 通常可被省略。
6. minSize: 目標的最小尺寸，小於這個尺寸的目標將被忽略。
7. maxSize: 目標的最大尺寸，大於這個尺寸的目標將被忽略。

程式五:動態人臉辨識

```
import cv2# 載入分類器

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# 從視訊鏡頭擷取影片
cap = cv2.VideoCapture(0)
while True:
    # Read the frame
    _, img = cap.read()
    # 轉成灰階
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # 偵測臉部
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)
    # 繪製人臉部份的方框
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
    # 顯示成果
    cv2.namedWindow('img', cv2.WINDOW_NORMAL) #正常視窗大小
    cv2.imshow('img', img) #秀出圖片
    if cv2.waitKey(30)== ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```



Exercise#4

1. 按下s可以儲存照片 save.png
2. 偵測到人臉的話就自動儲存照片 people.png (安全監控)
3. 列印出幾個人臉 利用 len()函數
4. 以及臉部區域

Exercise#5

1. 偵測到人臉的話就自動儲存照片 people.png (安全監控)
2. 依照有幾個人就存幾張照片
people1.png, people2.png, …..

提示：利用str(i) 與字串相加函數
利用cv2.imwrite

3. 裁切臉部區域存檔即可

裁切影像

首先按照普通的方式，將圖片用 OpenCV 的 `imread` 函數讀取進來：

```
import cv2

# 讀取圖檔
img = cv2.imread("lena.jpg")
```

這裡用 OpenCV 讀取進來的圖片 `img` 其實就是 NumPy 的陣列，如果想要對圖片進行裁切，就用索引的方式，將指定的區域取出即可：

```
# 裁切區域的 x 與 y 座標 (左上角)
x = 100
y = 100

# 裁切區域的長度與寬度
w = 250
h = 150

# 裁切圖片
crop_img = img[y:y+h, x:x+w]
```

接著使用 OpenCV 的 `imshow` 顯示裁切的結果：

```
# 顯示圖片
cv2.imshow("cropped", crop_img)
cv2.waitKey(0)
```

Exercise#4, 5解答

```
import cv2# 載入分類器

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# 從視訊鏡頭擷取影片
cap = cv2.VideoCapture(0)
while True:
    # Read the frame
    _, img = cap.read()
    # 轉成灰階
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # 偵測臉部
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    i=1
    # 繪製人臉部份的方框
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # 顯示成果

    cv2.namedWindow('Human Face Found!', cv2.WINDOW_NORMAL) #正常視窗大小
    cv2.imshow('Human Face Found!', img)#秀出圖片

    if cv2.waitKey(30)== ord('s'):
        cv2.imwrite("save.png",img)
    if cv2.waitKey(30)== ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

```
print(faces)
print('偵測到 {0}人臉 !'.format(len(faces)))
```