嘉義市政府
Chiayi City Government

107年度物聯網應用試辦暨資料收集分析案 教育訓練

人臉辨識 - OpenCV 應用

人臉識別 – 特徵點抓取與比對

授課講師: Ian
授課日期 : 108/05/09

since 1982
www.pershing.com.tw

北祥股份有限公司 PERSHING SYSTEMS CORPORATION

環境建置

環境建置

Conda建立並配置python虛擬環境

1. 開啟Terminal
2. 建立虛擬環境

```
conda create --name deepface python=3.6
```

3. 啟動虛擬環境

```
activate deepface
```

4. 下載工作區及配置文件至本地
   https://reurl.cc/dOxmM
5. 移動至工作區
6. 安裝套件

```
pip install -r requirements.txt
```

```
conda install -c conda-forge dlib=19.9.0
conda install -c anaconda pyyaml=3.12
conda install -c conda-forge pillow
```

7. 開啟jupyter notebook

```
jupyter notebook
```

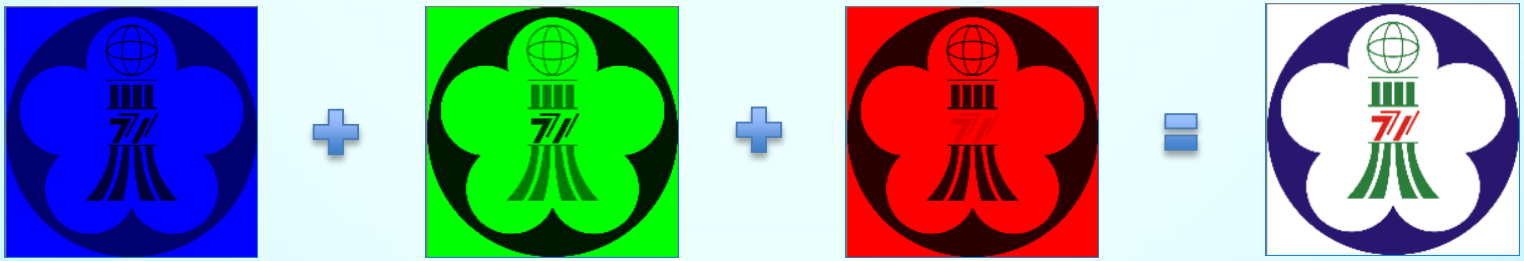
圖像基礎知識

圖像基礎知識

- 圖像都是由像素（pixel）構成的，即圖像中的小方格，這些小方格都有一個明確的位置和被分配的色彩數值，而這些方格的顏色和位置就決定該圖像所呈現出來的樣子。
- 像素是圖像中的最小單位，每一個點陣圖像包含了一定量的像素，這些像素決定圖像在屏幕上所呈現的大小。

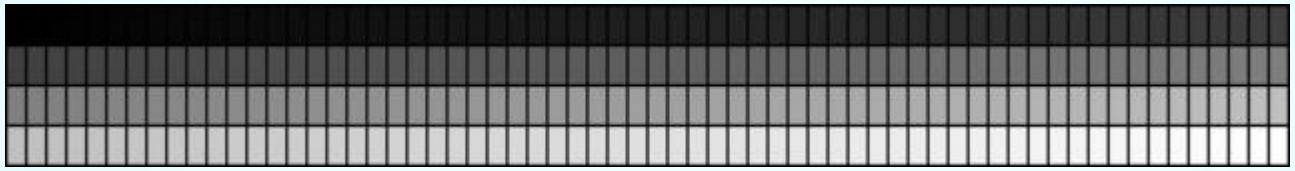
圖像基礎知識

- **RGB**
  - 紅（Red）、綠（Green）、藍（Blue）三原色的色光以不同的比例相加，電腦裡的顏色都是三原色組成的
  - 三色光疊合，得到各種顏色。
    - RGB 都是 255 ，疊合之後就是白光、呈現白色。
    - RGB 都是 0 ，就是無光、呈現黑色。
    - RGB 都一樣，則呈現灰色。
    - RGB 不一樣，則呈現各式各樣的彩色



圖像基礎知識

- 灰度
  - 灰度是表明圖像明暗的數值，即黑白圖像中點的顏色深度，範圍一般從0到255，白色為255 ，黑色為0，故黑白圖片也稱灰度圖像。
  - 灰度值指的是單個像素點的亮度，灰度值越大表示越亮



- 為什麼圖像識別要把圖片灰度化？
  - 包含色彩的話，特徵量，計算量成指數倍數增加
  - 例如一個點，灰度的話，頂多就256個維度，但假如算上RGB色彩的話，那就是1600萬以上維度。然後再相互組合

圖像基礎知識

- 圖像的灰度化
  - 灰度就是沒有色彩，RGB色彩分量全部相等。
  - 圖像的灰度化就是讓像素點矩陣中的每一個像素點都滿足關係：R=G=B，此時的這個值叫做灰度值。
  - 如RGB(100,100,100)就代表灰度值為100，RGB(50,50,50)代表灰度值為50。

- 算法
  - 假如原來某點的顏色為RGB(R,G,B)，可以通過下面幾種方法，將其轉換為灰度：
    1. 浮點算法：Gray=R*0.3+G*0.59+B*0.11
    2. 整數方法：Gray=(R*30+G*59+B*11)/100
    3. 移位方法：Gray =(R*77+G*151+B*28)>>8
    4. 平均值法：Gray=（R+G+B）/3
    5. 僅取綠色：Gray=G
  - 通過上述任一種方法求得Gray後，將原來的RGB(R,G,B)中的R,G,B統一用Gray替換，形成新的RGB(Gray,Gray,Gray)，用它替換原來的RGB(R,G,B)就是灰度化。

OpenCV- 簡介

OpenCV- 簡介

- OpenCV全名是Open Source Computer Vision Library，是一個影像處理函式庫，由Intel發起並參與開發，以BSD授權條款發行，可在商業和研究領域中免費使用，目前是非營利的基金組織OpenCV.org在維護

- 跨平台，可以執行在Linux、Windows、Android和Mac OS等系統上

- 由一系列C 、C++構成，同時提供了Python、Ruby、MATLAB等語言的介面，實現了影象處理和計算機視覺方面的很多通用演算法

OpenCV- 應用

OpenCV- 基本讀取圖片

- 首先引入 OpenCV 的 Python 模組

```
import cv2
```

- OpenCV 本身就有提供讀取圖片檔的函數可用，讀取一般的圖片檔，只要呼叫 cv2.imread 函數即可將圖片讀取進來：

```
img = cv2.imread('image/logo.png')
```

- OpenCV 的 cv2.imread 在讀取圖片時，可以在第二個參數指定圖片的格式，可用的選項有三種：
  - cv2.IMREAD_COLOR
    - 此為預設值，這種格式會讀取 RGB 三個 channels 的彩色圖片，而忽略透明度的 channel，對應的值為1。
  - cv2.IMREAD_GRAYSCALE
    - 以灰階的格式來讀取圖片，對應的值為0。
  - cv2.IMREAD_UNCHANGED
    - 讀取圖片中所有的 channels，包含透明度的 channel ，對應的值為 -1。

OpenCV- 顯示圖片

- 將圖片讀取進來之後，可以使用 OpenCV 所提供的 cv2.imshow 函數來顯示

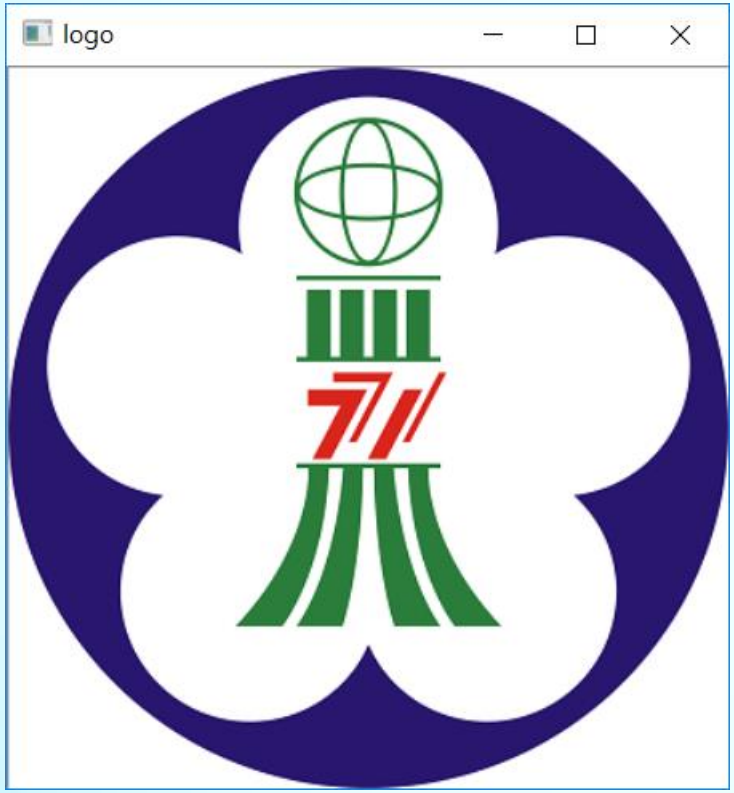
```
cv2.imshow('logo', img) #顯示圖片

cv2.waitKey(0) #持續等待至按下按鍵為止

cv2.destroyAllWindows() #關閉所有 OpenCV 的視窗
```

cv2.waitKey 函數是用來等待與讀取使用者按下的按鍵，而其參數是等待時間（單位為毫秒），若設定為 0 就表示持續等待至使用者按下按鍵為止，在此期間不斷刷新圖像。

這樣當我們按下任意按鍵之後，就會呼叫 cv2.destroyAllWindows 關閉所有 OpenCV 的視窗。

課堂練習

使用前面所敘述函數，開啟工作區data/image/logo.png圖片



OpenCV- 繪製矩形

- 首先引入 OpenCV 的 Python 模組

```
import cv2
```

- 引入模組numpy並以np作為別名：

```
import numpy as np
```

- 設定一個空的彩色圖片

```
img = np.zeros((431, 431, 3), np.uint8)
```

- 繪製矩形

```
cv2.rectangle(img, (20, 20), (411, 411), (55, 255, 155), 5)
```

左上角　右下角　顏色　線的類型(不設置就默認)

OpenCV- 繪製圓形

- 首先引入 OpenCV 的 Python 模組

```
import cv2
```

- 引入模組numpy並以np作為別名：

```
import numpy as np
```

- 設定一個空的彩色圖片

```
img = np.zeros((431, 431, 3), np.uint8)
```

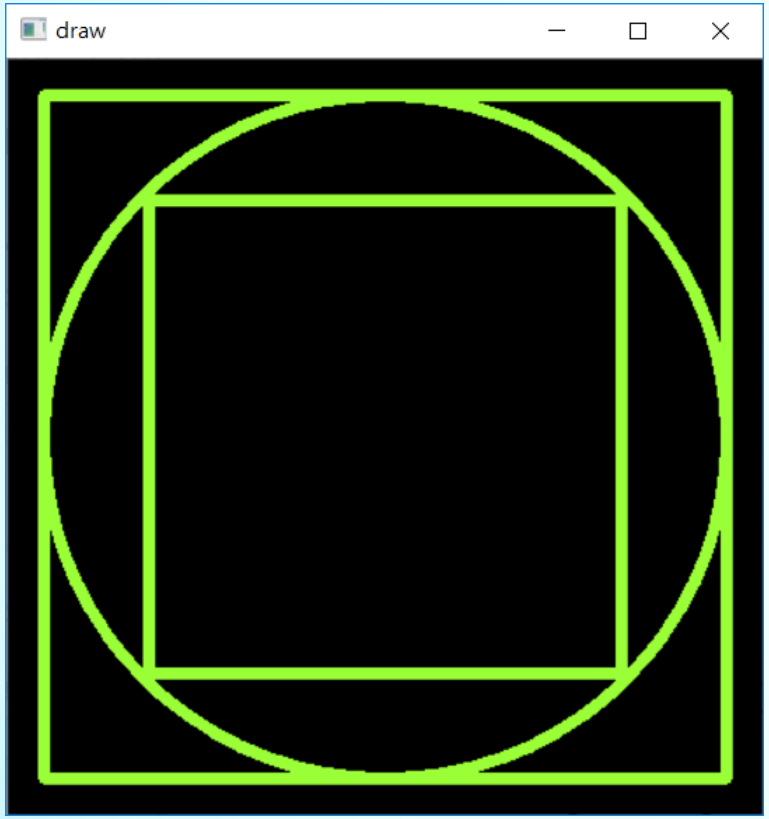
- 繪製圓形

```
cv2.circle(img, (200, 200), 50, (55, 255, 155), 1)
```

圓心　半徑　顏色　線的類型(不設置就默認)

課堂練習

使用前面所敘述函數，繪製出以下圖片



OpenCV-圖片上加上文字

- 若要在圖片上加上文字，可以使用 cv2.putText 函數：

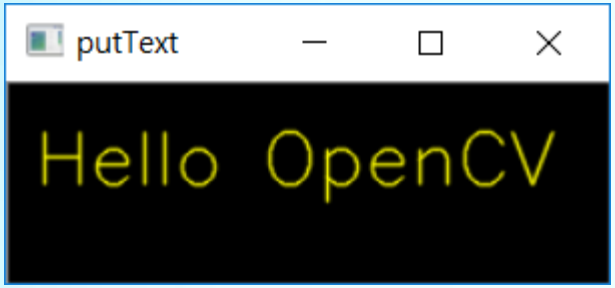
```
# cv2.putText(影像, 文字, 座標, 字型, 大小, 顏色, 線條寬度, 線條種類)
```

Enumerator	
FONT_HERSHEY_SIMPLEX Python: cv.FONT_HERSHEY_SIMPLEX	normal size sans-serif font
FONT_HERSHEY_PLAIN Python: cv.FONT_HERSHEY_PLAIN	small size sans-serif font
FONT_HERSHEY_DUPLEX Python: cv.FONT_HERSHEY_DUPLEX	normal size sans-serif font (more complex than FONT_HERSHEY_SIMPLEX)
FONT_HERSHEY_COMPLEX Python: cv.FONT_HERSHEY_COMPLEX	normal size serif font
FONT_HERSHEY_TRIPLEX Python: cv.FONT_HERSHEY_TRIPLEX	normal size serif font (more complex than FONT_HERSHEY_COMPLEX)
FONT_HERSHEY_COMPLEX_SMALL Python: cv.FONT_HERSHEY_COMPLEX_SMALL	smaller version of FONT_HERSHEY_COMPLEX
FONT_HERSHEY_SCRIPT_SIMPLEX Python: cv.FONT_HERSHEY_SCRIPT_SIMPLEX	hand-writing style font
FONT_HERSHEY_SCRIPT_COMPLEX Python: cv.FONT_HERSHEY_SCRIPT_COMPLEX	more complex variant of FONT_HERSHEY_SCRIPT_SIMPLEX
FONT_ITALIC Python: cv.FONT_ITALIC	flag for italic font

Enumerator	
FILLED Python: cv.FILLED	
LINE_4 Python: cv.LINE_4	4-connected line
LINE_8 Python: cv.LINE_8	8-connected line
LINE_AA Python: cv.LINE_AA	antialiased line

課堂練習

使用前面所敘述函數，繪製出以下圖片



OpenCV- 圖片顏色空間轉換

- 生活中大多數看到的彩色圖片都是RGB類型，但是在進行圖像處理時，需要用到灰度圖、二值圖、HSV、HSI等顏色制式，OpenCV提供了cvtColor()函數來實現這些功能。

```
#Convert to gray
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

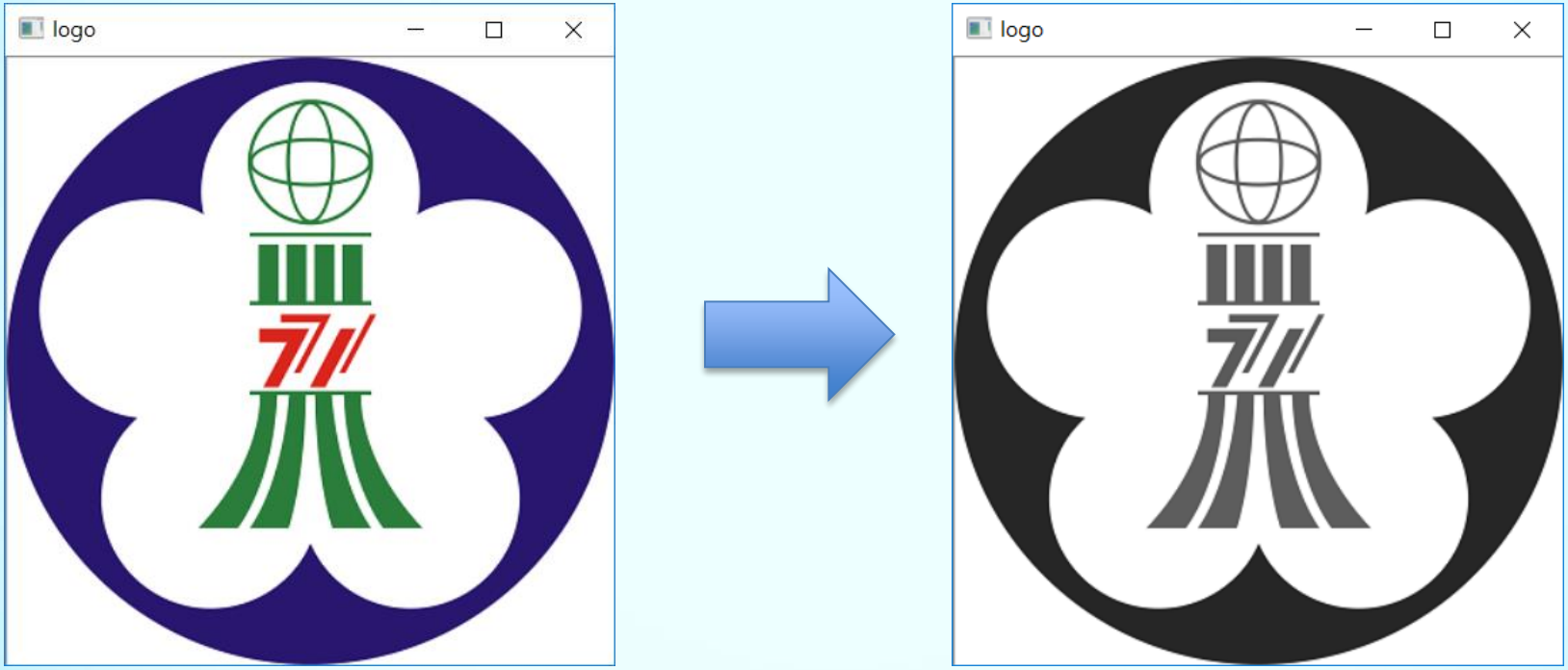
```
#Convert to hls
img = cv2.cvtColor(img, cv2.COLOR_BGR2HLS)
```

```
#Convert to hsv
img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

- 另外cvtColor()函數回傳的是原圖片的副本，也可以利用copy()函數複製圖片

課堂練習

使用cvtColor()函數，將工作區data/image/logo.png圖片轉為灰度圖



OpenCV- 寫入圖片

- 將圖片寫入檔案，可以使用 OpenCV 的 cv2.imwrite 函數

```
cv2.imwrite('image/output.png', img)
```

- cv2.imwrite 可透過圖片的副檔名來指定輸出的圖檔格式

```
cv2.imwrite('image/output.jpg', img)
```

```
cv2.imwrite('image/output.tiff', img)
```

- 輸出圖片檔案時，也可以調整圖片的品質或壓縮率

```
# 設定 JPEG 圖片品質為 90（可用值為 0 ~ 100）
cv2.imwrite('image/output.jpg', img, [cv2.IMWRITE_JPEG_QUALITY, 90])
```

```
# 設定 PNG 壓縮層級為 5（可用值為 0 ~ 9）
cv2.imwrite('image/output.png', img, [cv2.IMWRITE_PNG_COMPRESSION, 5])
```

OpenCV- 獲取圖片屬性

- 要獲取圖片屬性，可以使用 OpenCV 的 shape 函數

```
img.shape
```

- 返回值，按照順序是一個包含行數，列數，通道數的元組

```
(360, 360, 3)
```
→ 代表是360 * 360 的三通道圖片

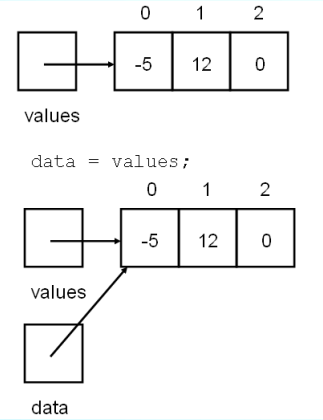
如果圖像是灰度圖，返回值僅有行數和列數，
所以通過檢查返回值可以判斷是灰度圖還是彩色圖

OpenCV- 複製圖片

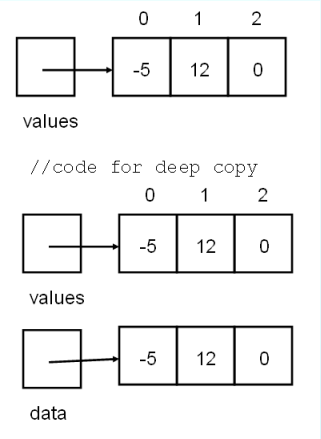
淺複製(Shallow copy)和深複製(Deep copy)

- 程式會為每個變數分配記憶體空間，當改變變數的值時，改變的是記憶體空間中的值，變數的地址是不改變的。
- 一般尋常意義的複制就是深複製，即將被複制對象完全再複制一遍作為獨立的新個體單獨存在，所以改變原有被複製對像不會對已經復制出來的新對象產生影響

淺複製(Shallow copy)



深複製(Deep copy)



OpenCV- 複製圖片

要深複製圖片的話，可以使用以下函數

- 使用copy函數直接複製

```
# 直接複製原來的圖片到一張新的圖片上
image2 = image.copy()
```

- 使用OpenCV庫的cvtColor函數

```
# 引入影象處理庫OpenCV
import cv2
```

```
# 使用cvtColor獲取原圖片的副本
image3 = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

OpenCV- 複製圖片

要深複製圖片的話，可以使用以下函數

- 使用numpy庫的zeros函數
  - 一張圖片的像素值範圍是[0,255], 因此默認類型是unit8

```
# 引入numpy庫，以np做別名
import numpy as np
```

```
# 使用numpy.zeros來返回一個新的數組，其中參數則是原圖片的屬性以及資料類型
image1 = np.zeros(image.shape, np.uint8)

# 獲取被複製圖片的rows、cols
rows = image.shape[0]
cols = image.shape[1]

# 遍尋複製好的數組，將被複製的數組，按照位置塞入
for now_row in range(rows):
    for now_col in range(cols):
        image1[now_row, now_col, 0] = image[now_row, now_col, 0]
        image1[now_row, now_col, 1] = image[now_row, now_col, 1]
        image1[now_row, now_col, 2] = image[now_row, now_col, 2]
```

OpenCV- 縮放圖片

要對圖片進行縮放的話，可以使用 OpenCV 的 resize函數

- 使用元組指定新圖片的尺寸

```
# 直接指定使用元組(tuple)指定新圖片的尺寸 (400, 400) > (圖片寬, 圖片高)
resImg1 = cv2.resize(img, (400, 400))
```

- 使用圖片的shape進行變換

```
# 使用圖片的shape進行變換
resImg2 = cv2.resize(img, (img.shape[1], img.shape[0]))
```

- 使用線性係數進行變換

```
'''
使用線性係數進行變換，
fx表示對坐標系中的x軸變換的係數，為水平方向，也就是長；
fy表示寬的係數。當係數大於1時，表示放大，小於1時表示縮小。
'''
resImg3 = cv2.resize(img, None, fx=0.1, fy=0.5)
```

課堂練習

將工作區目錄data/image下的logo.png ，進行複製

使用章節『OpenCV- 複製圖片』所教的三種複製圖片方式

- 使用copy函數直接複製
- 使用OpenCV庫的cvtColor函數
- 使用numpy庫的zeros函數

以及對複製好的圖片進行縮放操作

使用章節『OpenCV- 縮放圖片』所教的三種縮放圖片方式

- 使用元組指定新圖片的尺寸
- 使用圖片的shape進行變換
- 使用線性係數進行變換

OpenCV- 分割& 合併圖片通道

- 要將圖片的各個通道分離，可以使用 OpenCV 的 split 函數

```
B, G, R = cv2.split(image)
```

  - split函數的主要功能是把一個彩色圖像分割成3個通道，方便進一步的圖像處理
  - 在OpenCV中，影像排列方式是BGR，而非一般圖片的RGB 排列

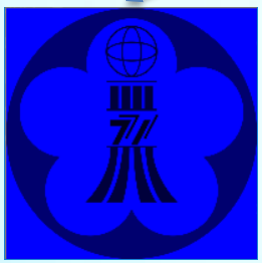
- 要將多個單通道圖像合併的話，可以使用 OpenCV 的 merge 函數

```
merged = cv2.merge([B, G, R])
```

課堂練習

將工作區目錄data/image下的logo.png 用前面章節講述的函數，進行通道分離，並按各顏色通道將其他顏色通道值設為0，將處理好的圖片，儲存至目錄data/image中，檔案格式為jpg

logo.png



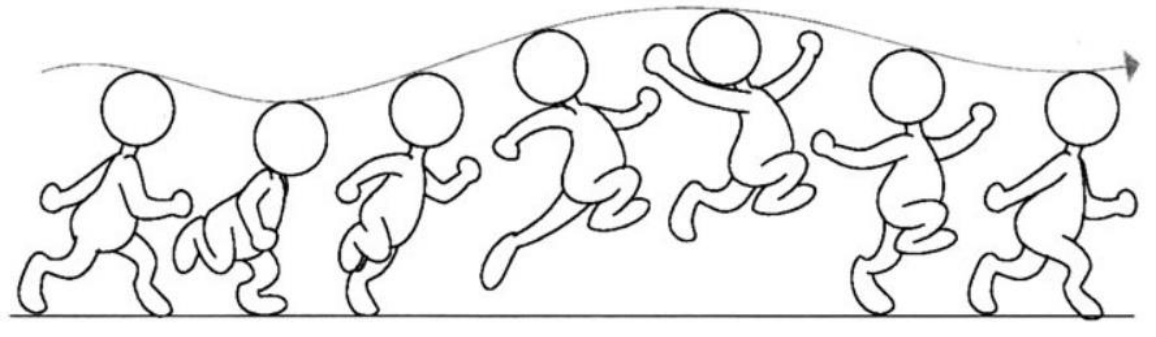
B.jpg

G.jpg

R.jpg

OpenCV- 影片

- 幀
  - 就是影像動畫中最小單位的單幅影像畫面，相當於電影膠片上的每一格鏡頭。一幀就是一副靜止的畫面，連續的幀就形成動畫，如電視圖像等。
  - 通常說幀數，就是在1秒鐘時間里傳輸的圖片的幀數，也可以理解為圖形處理器每秒鐘能夠刷新幾次，通常用fps（Frames Per Second）表示。
  - 人眼舒適放鬆時可視幀數約是每秒24幀。
- 影片處理
  - 在前面章節中，介紹了OpenCV圖片的輸入輸出以及相關操作，而本質上影片是由一幀幀的圖片組成，那麼之前所有的圖片處理的操作便能移植到影片上來。



OpenCV- 影片

- 影片輸入
  - 影片輸入有兩種模式，一種是讀取現有的影片文件，另一種是通過攝像頭實時獲取影片。

  - 在OpenCV中，這兩種模式用一個VideoCapture函數就可以全部搞定，唯一的區別是構造函數的參數不同。

```
# 選擇內建攝影機
cap = cv2.VideoCapture(0)

# 選擇影片檔
cap = cv2.VideoCapture('data/video/smile.mp4')
```

  - 當建立好 VideoCapture 物件之後，就可以使用它的 read 函數來擷取一張張連續的畫面影像了，第一個傳回值 ret代表成功與否（True 代表成功，False 代表失敗），而第二個傳回值 frame 就是攝影機的單張畫面

```
# 從VideoCapture擷取一張影像
ret, frame = cap.read()
```

課堂練習

使用VideoCapture庫，讀取位於工作區data/video目錄下的影片 ，將每一幀影像轉換為灰度，並播放於OpenCV視窗

人臉檢測 & 辨識- 簡介

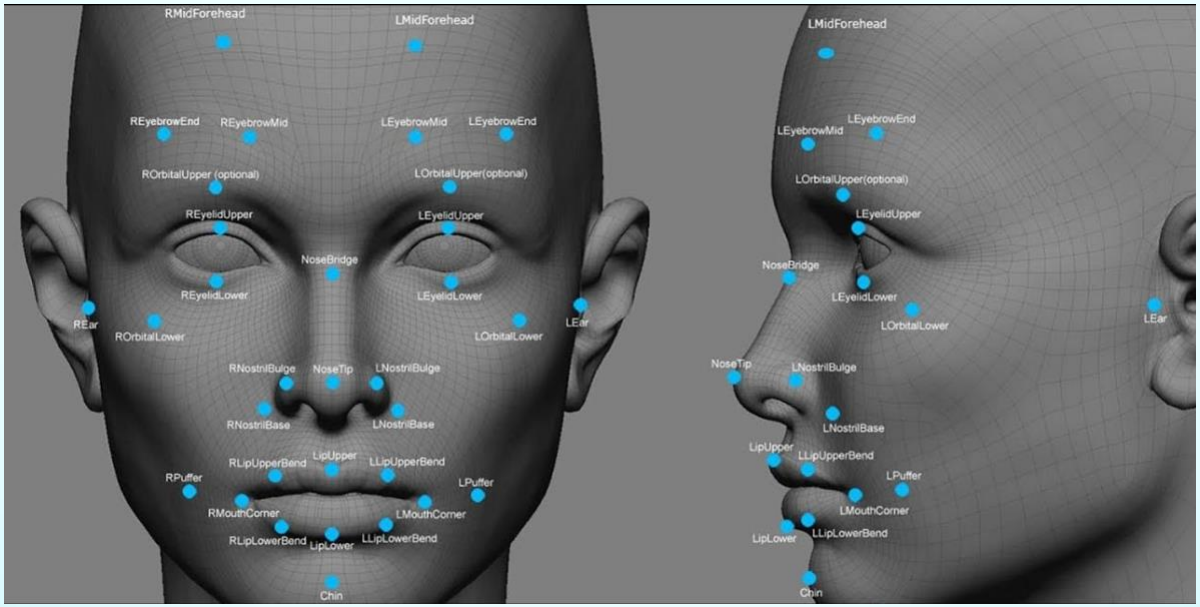
人臉檢測 & 辨識- 基礎

- 人臉辨識是指基於人的臉部特徵資訊進行身分辨識的一種生物辨識技術。
- 使用含有臉部的圖片或視訊流，並自動在圖像中檢測和跟蹤臉部，進而對檢測到的臉部進行臉部辨識的一系列相關技術。

人臉檢測 & 辨識- Classifier(分類器)

- 對於像臉一樣複雜的東西，沒有一個簡單的方法可以直接告訴你，它是否找到了一張臉。

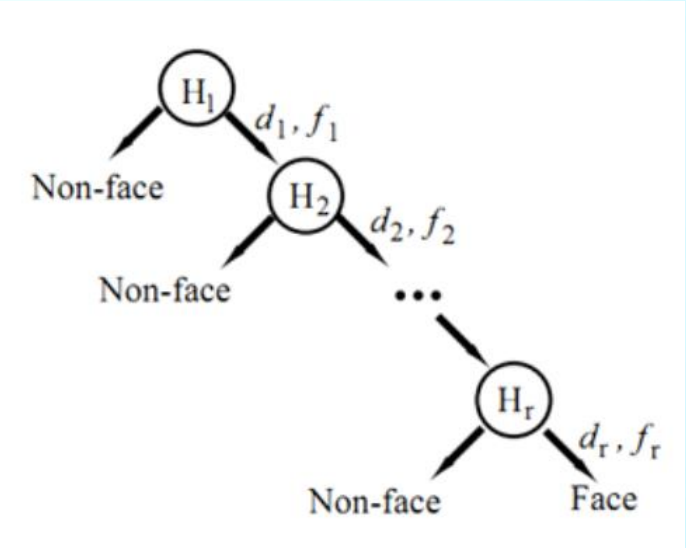
- 相反，必須匹配數以千計的小圖案/特徵。 將識別面部的任務分解為數千個較小的算法，每個任務都很容易解決。 這些任務稱為分類器。



人臉檢測- OpenCV Haar Cascades

- 一張臉，會有超過 6000 個 classifiers，那如果要辨識一張圖片裡面有幾張人臉，不就等於是要把圖裡面的全部區塊，都分別跑完 6000 個 classifiers 比對，這會需要很多很多的運算資源與時間。

- 而OpenCV中的Cascades (瀑布) 是把人臉辨識的 6000 個 classifiers 分成數個階段，比對圖區塊時，都從第一個階段做辨識，若沒通過就淘汰不用往下做，有通過才繼續做下階段的辨識，直到通過全部的辨識，才判斷為人臉，這種類似瀑布一段一段往下做法，好處是節省運算資源與時間，不用每個圖區塊都得做足 6000 個辨識。

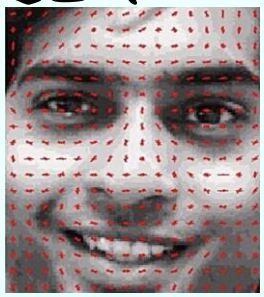


人臉辨識- dlib Hog + SVM

- 使用Haar級聯分級器進行眼睛檢測，形成幾何臉部模型；而鼻子的檢測則被用作眼睛檢測的再確認機制。
- 之後，從大量臉部影像中擷取「定向梯度直方圖」(Histogram of Oriented Gradients；HOG)特徵，作為辨識機制的一部份。
- 然後將這些HOG特徵一併標記為某一個人臉/使用者，並且訓練支援向量機(Support Vector Machines ; SVM)模型以預測饋送到系統中的臉部

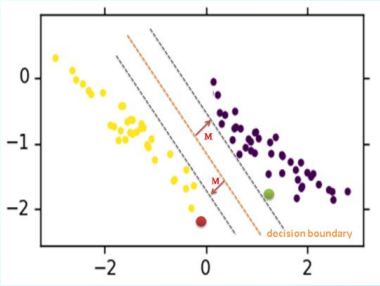
什麼是定向梯度直方圖(HOG)

首先將圖像分成小的連通區域，我們把它叫細胞單元。然後採集細胞單元中各像素點的梯度的或邊緣的方向直方圖。最後把這些直方圖組合起來就可以構成特徵描述器



什麼是支援向量機(SVM)

是一種二分類模型，它的基本模型是定義在特徵空間上的間隔最大的線性分類器



人臉辨識- 應用情景

人臉辨識- 應用情景

- **情緒偵測**

人臉辨識技術透過分析臉部各種不同的點，並追蹤這些點之間的關係來偵測情緒。這與所謂的『微表情』一樣，說明軟體決定常見的情緒，如驚喜、快樂、憤怒、悲傷、厭惡等

漱口水品牌 – 李施德霖®發起的一項廣告活動，
該公司開發了一款App，可藉由嗡嗡聲通知盲胞有人正在對他們微笑

https://adage.com/creativity/work/power-smile/43122?

人臉辨識- 應用情景

- **情緒偵測**

在中國杭州，第十一中學正在嘗試使用該技術，名為智能教室行為管理系統，由放置在黑板上方的三個攝像頭通過每隔30秒掃描一次，分析學生，並由處理器進行分類統計、打分，識別學生情緒、確認出缺勤情況。

系統將課堂行為分為六種類型的學生行為：閱讀、書寫、聽講、起立、舉手和趴桌子，而每種行為就代表一個分值

https://www.techspot.com/news/74719-chinese-school-using-facial-recognition-analyze-students-emotions.html

人臉辨識- 應用情景

- **人臉偵測**

東京奧運組織委員會2018年8月公開最新式臉部辨識系統，屆時包括選手、志工及媒體等約 30 萬名相關人士入場時，就可以由機器來確認是否符合入場資格，比傳統人工判斷快約 2.5 倍。

https://www.youtube.com/watch?time_continue=15&v=nIa7EUExUUo

人臉辨識-特徵點抓取與比對

人臉辨識-特徵點抓取與比對

- Haar特徵分類器就是一個XML文件，該文件中會描述人體各個部位的Haar特徵值，包括人臉、眼睛、嘴唇等等。

- 加載分類器(人臉)

```
# Cascade path
cascPath = "data/haarcascades/haarcascade_frontalface_default.xml"
```

```
# Create the haar cascade
faceCascade = cv2.CascadeClassifier(cascPath)
```

- 目前人臉識別使用的是 detectMultiScale函數，它可以檢測出圖片中所有的人臉，並保存各個人臉的坐標、大小（用矩形表示）

```
# Detect faces in the image
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1, #表示在前後兩次相繼的掃描中，搜索窗口的比例係數。默認為1.1即每次搜索窗口依次擴大10%
    minNeighbors=5, #表示構成檢測目標的相鄰矩形的最小個數(默認為3個)
    minSize=(30, 30),#限制得到的目標區域的最小範圍
)
```

人臉辨識-特徵點抓取與比對

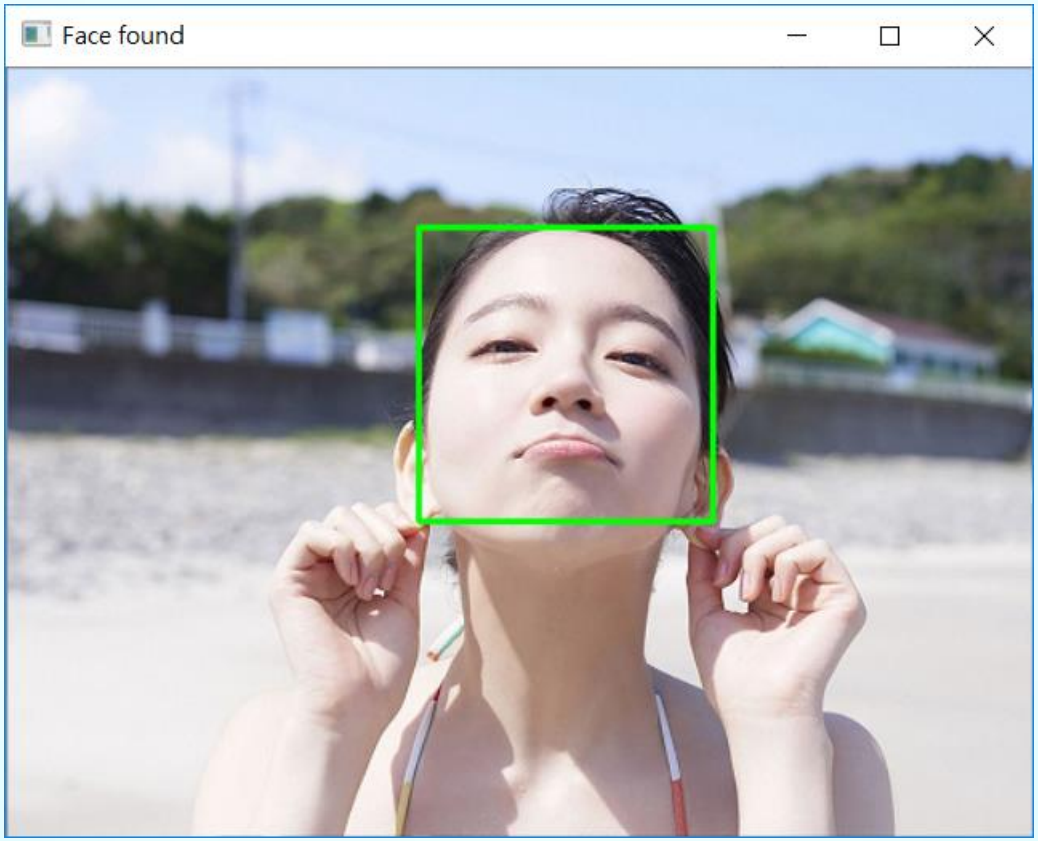
- 將找到的人臉，繪製標記起來

```
# Draw a rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

- 利用OpenCV的Python介面，實現人臉識別的基本流程，大致如下
  1. 讀取圖片
  2. 將圖片轉換為灰度模式，便於人臉檢測
  3. 利用特徵分類器檢測圖片中的人臉
  4. 繪製人臉的矩形區域
  5. 顯示人臉識別後的圖片

課堂練習

使用detectMultiScale()函數，將工作區data/image/girl.jpg ，用矩形標記其人臉



人臉辨識-特徵點抓取與比對

笑臉檢測

- 是在人臉檢測之後得到的人臉區域中進行的

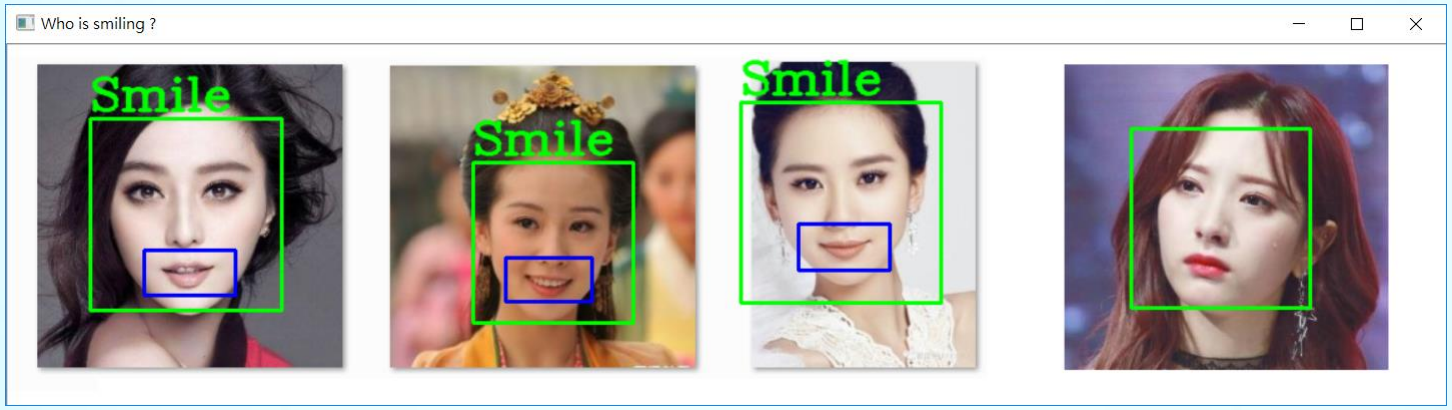
```
# Cascade path
cascPath = "data/haarcascades/haarcascade_frontalface_default.xml"
smilePath = "data/haarcascades/haarcascade_smile.xml"
```

- 加載人臉、笑臉分類器

```
# Create the haar cascade
faceCascade = cv2.CascadeClassifier(cascPath)
smileCascade = cv2.CascadeClassifier(smilePath)
```

課堂練習

使用detectMultiScale()函數，將工作區data/image/smile.jpg ，用矩形標記其人臉，且在標記的人臉中，找出哪張人臉是微笑表情，並用矩形標記，在其上標記文字



dlib 人臉檢測

- 引入會使用到的庫

```
# 引入人臉識別庫dlib
import dlib
```

```
# 引入影象處理庫OpenCV
import cv2
```

- 使用人臉提取器從灰度圖中取出臉的資訊

```
# 使用dlib庫提供的人臉提取器
detector = dlib.get_frontal_face_detector()
```

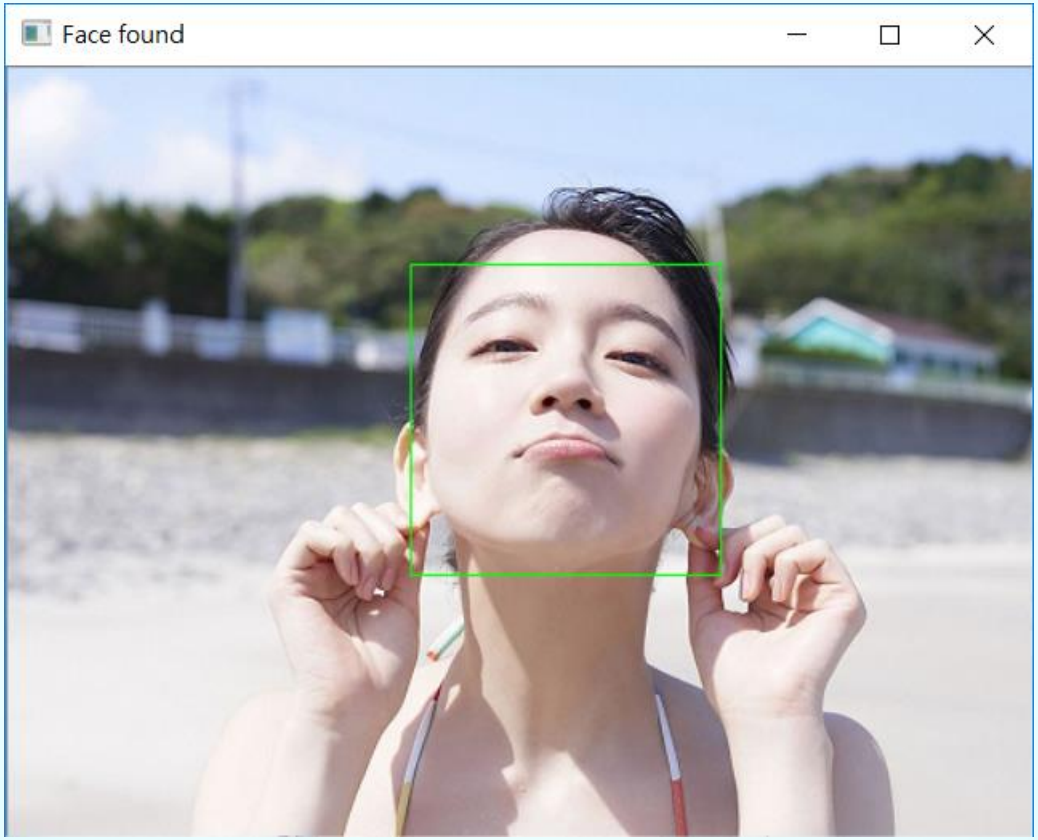
```
# 返回臉的資訊
faces = detector(gray)
```

- 繪製矩形框住臉

```
# 框出每張臉
for face in faces:
    cv2.rectangle(img, (face.left(), face.top()), (face.right(), face.bottom()), (0, 255, 0), 1)
```

課堂練習

使用OpenCV以及dlib庫，將工作區data/image/girl.jpg ，用矩形標記其人臉

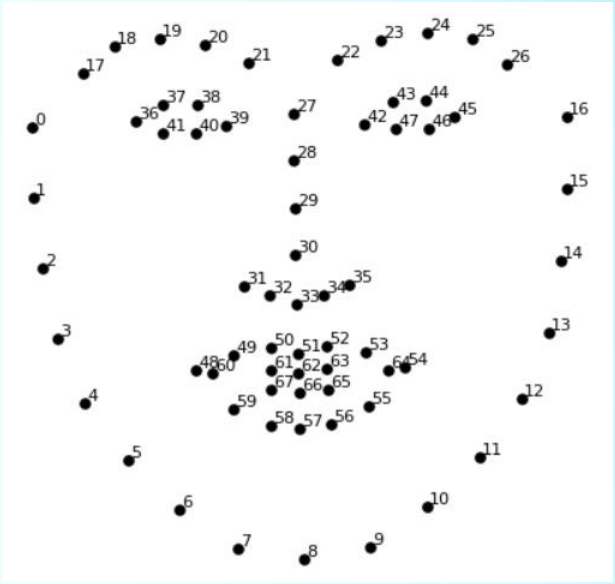


OpenCV + dlib 偵測人臉68個特徵點

- 引入會使用到的庫

```python
# 引入人臉識別庫dlib
import dlib
```

```python
# 引入影象處理庫OpenCV、imutils
import cv2
from imutils import face_utils
```



- 加載提取器

```python
# 使用dlib庫提供的人臉提取器
detector = dlib.get_frontal_face_detector()
```

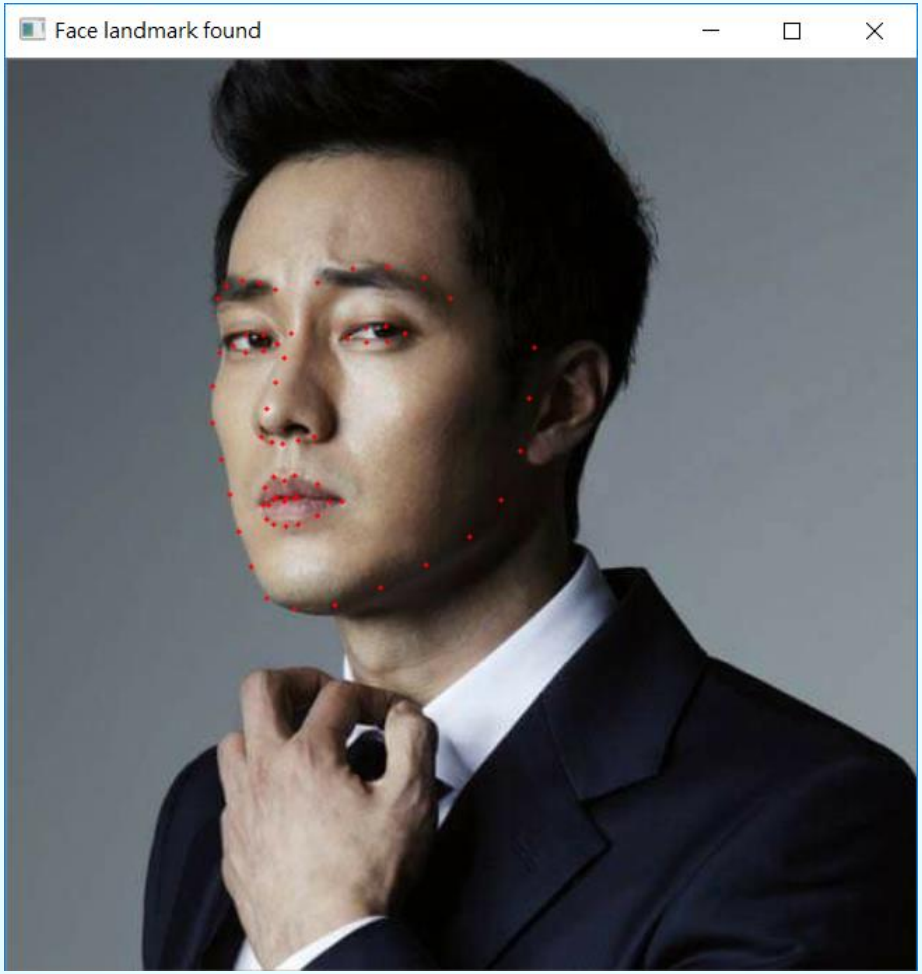
```python
# 使用官方提供的模型構建特徵提取器
predictor = dlib.shape_predictor("data/dlib/landmarks.dat")
```

- 使用特徵提取器，從臉檢測出特徵點，並使用影像處理庫(imutils)將其轉為坐標矩陣，以便繪製圖形

```python
# 標記人臉中的68個特徵點
shape = predictor(gray, face)

# shape轉換成68個坐標點矩陣
shape = face_utils.shape_to_np(shape)

# 繪製圖形標誌
for (x, y) in shape:
    cv2.circle(img, (x, y), 1, (0, 0, 255), -1)
```

課堂練習

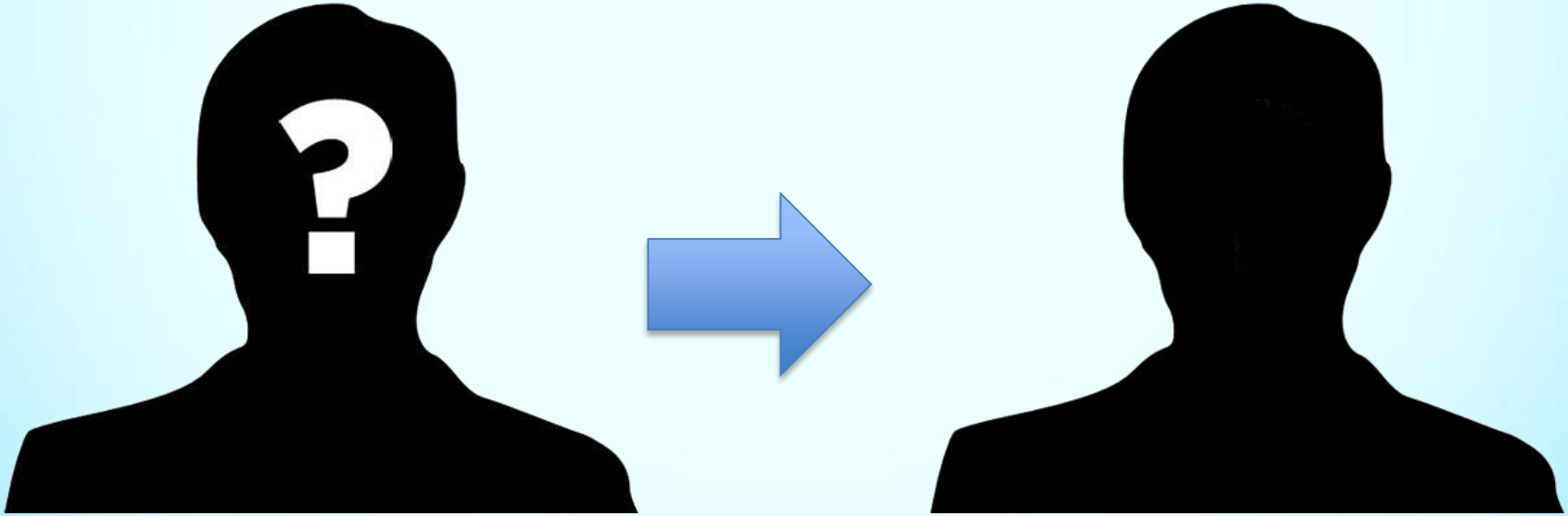
使用OpenCV以及dlib庫，將工作區data/image/men.jpg ，將其人臉特徵標記出來



人臉辨識-人臉比對

利用OpenCV庫進行人臉識別的步驟：

1. 人臉採集：對樣本圖片進行人臉採集
2. 生產標籤文件：生產對應樣本圖片的標籤文件
3. 訓練模型資料：對樣本圖片進行訓練，產生相應的xml文件
4. 人臉識別：利用訓練好的模型資料和分類器進行人臉識別



人臉辨識-人臉採集(課堂練習)

利用前面章節所教的函數，
將工作區目錄data/video/下的兩部影片
(TT_TZUYU.mp4、Obama.mp4)，讀取其影片，

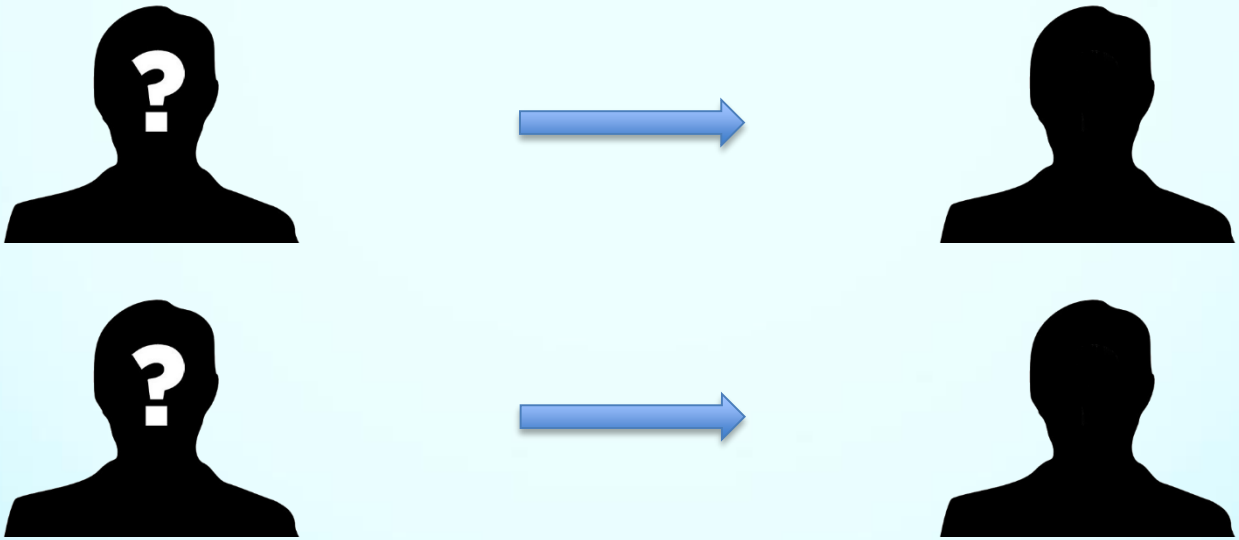
並將影片中人物的人臉部位截取下來(起碼十張以上)，
轉為400*400的灰度圖，檔案格式為pgm，
儲存位置分別為工作區目錄image/train/Obama以及image/train/TZUYU

人臉辨識-生產標籤文件

當人臉採集完成後，我們就有了基本的訓練數據集，
但光是蒐集生成人臉數據並沒有用，

還得寫個映射表，進行分類的動作，讓機器知道人臉數據對應那個人物

標籤文件格式 ：	圖片路徑	分隔符號	標籤記號



人臉辨識-生產標籤文件

1. 引用os庫: 該庫主要是與操作系統作互動

```
# 引入os庫
import os
```

2. 宣告會使用到的變數

```
# 指定根目錄
root_path ="data/image/train"

# 標籤文件位置路徑
label_path= "%s/label.txt" % (root_path)

# 分隔符號
separator=";"

# 宣告標籤
label = 0
```

```
# 創建標籤文件
#
#   預期的樹狀目錄圖
#   .
#   |-- README
#   |-- s1
#   |    |-- 1.pgm
#   |    |-- ...
#   |    |-- 10.pgm
#   |-- s2
#   |    |-- 1.pgm
#   |    |-- ...
#   |    |-- 10.pgm
#   ...
#   |-- s40
#   |    |-- 1.pgm
#   |    |-- ...
#   |    |-- 10.pgm
#
```

3. 打開文件，等等用來寫入標籤訊息

```
'''
打開一個文件用於寫入，
後面w代表是: 如果該文件已存在則打開文件，並從開頭開始編輯，
即原有內容會被刪除，反之則創建新文件
'''
fh = open(label_path,'w')
```

人臉辨識-生產標籤文件

4. 去根目錄下遍尋檔案，依照映照表格式，將蒐集的人臉圖與標籤寫入文件

```
# 遞迴列出所有子目錄與檔案，並將檔案路徑以及分隔符號、標籤寫入文件
for root, dirs, files in os.walk(root_path):
    for dir in dirs:
        # 當前目錄
        current_dir_path = "%s/%s" % (root, dir)

        # 當前檔案
        for file in os.listdir(current_dir_path):

            # 檔案路徑
            file_path = "%s/%s" % (current_dir_path, file)
            print ("%s%s%d" % (file_path, separator, label))

            # 寫入檔案
            fh.write(file_path)
            fh.write(separator)
            fh.write(str(label))
            fh.write("\n")

        # 每當處理過一個目錄時，標籤值遞增
        label = label + 1
```

5. 最後關閉使用到的文件資源

```
# 關閉文件資源
fh.close()
```

人臉辨識-生產標籤文件(課堂練習)

利用前面章節『人臉辨識-生產標籤文件』講解的步驟，

將工作區目錄 data/image/train 中採集到人臉資料(Obama、TZUYU)
產生出對應的映射表(標籤文件)

人臉辨識-訓練模型數據

1. 引用會使用到的庫

```
# 引入影象處理庫OpenCV
import cv2
```

```
# 引入numpy庫，並用np當別名
import numpy as np
```

2. 宣告兩個空陣列，用來放置圖片位置以及對應的標籤號

```
# 放置圖片位置
images = []
```

```
# 放置標籤號
labels = []
```

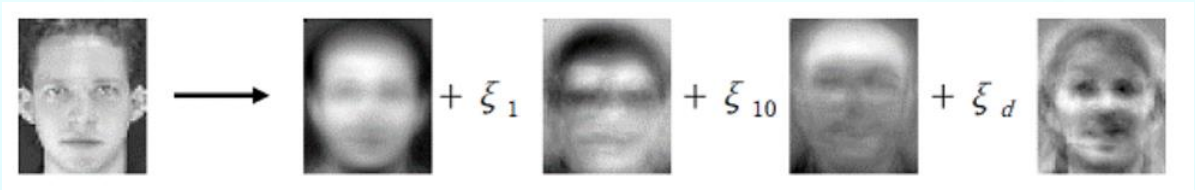
3. 打開建立好的映照表(標籤文件)，並放置內容於宣告陣列中

```
# 打開對應的標籤文件，並放置內容
with open('data/image/train/label.txt') as fh:
    for line in fh:
        arrLine = line.split(';')
        image = cv2.imread(arrLine[0], 0)
        images.append(image)
        labels.append(int(arrLine[1]))
```

人臉辨識-訓練模型數據

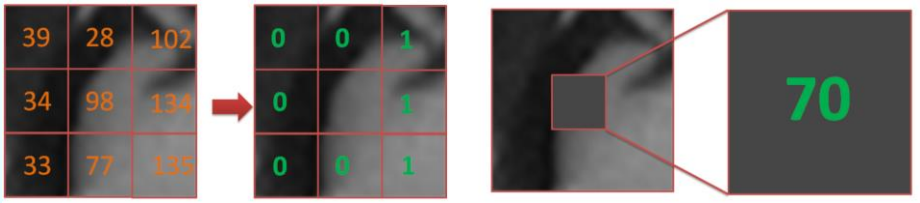
OpenCV 3有三種人臉識別的方法，它們分別基於不同的三種算法

- **Eigenfaces(特徵臉)**
  - 在人臉識別歷史上應該是具有里程碑式意義的，因為該方法被認為是第一種有效的人臉識別方法
  - 是一種基於統計特徵的方法，將人臉圖像視為隨機向量，並用統計方法辨別不同人臉特徵模式
  - 任意一張人臉圖像都可以被認為是這些標準臉的組合，例如，一張人臉圖像可能是特徵臉1的20%，加上特徵臉2的37%，在減去特徵臉3的9%



人臉辨識-訓練模型數據

- **Fisherfaces**
  - 在一個人臉集合上求得k個特徵向量
  - 將人臉在k個特徵向量上做投影，得到k維的列向量或者行向量，然後和已有的投影求得歐式距離，根據閾值來判斷是否匹配
- **Local Binary Pattern Histogram(LBPH)**
  - 將各個像素與其附近的像素進行比較，並把結果保存為二進位數
    - 假如將中心點像素為閾值，與其相鄰的8個像素的灰度值進行比較，若這8個相鄰點其值大於中心像素值，那麼該相鄰點像素的位置被標記為1，否則為0。
    - 比較完之後，按著順時鐘從最左上角排列其值，可得到一組二進位值00111000，換算為10進位則為70，因此該cell的灰度值將重新定義為70



    - 有了各個區域影像的LBP值，我們可以將其轉換為Histogram直方圖(X軸為像素強度，Y軸為像素數目)，每一個影像區域會得出一個直方圖，若將這些直方圖合併，則得到整張影像的直方圖。
  - 當有直方圖，就可以描述這幅圖片了，之後利用各種相似性度量函數，就可以判斷兩幅圖像之間的相似性

人臉辨識-訓練模型資料

4. 定義人臉識別模型以及要使用的算法

```
# 定義人臉識別模型
model = cv2.face.EigenFaceRecognizer_create()
```

5. 進行模型訓練

```
# 進行模型訓練
model.train(np.array(images), np.array(labels))
```

6. 訓練好之後保存結果

```
# 保存訓練模型資料
model.save('data/selfcascades/predict_face.xml')
```

人臉辨識-訓練模型資料(課堂練習)

將前面採集的人臉資料以及建立好的映射表(標籤文件)，

利用前面章節『人臉辨識-訓練模型資料』講解的步驟，
訓練模型，並將模型保存於工作區目錄 data/selfcascades 中

人臉辨識-人臉識別

1. 引用會使用到的庫

```
# 引入影象處理庫OpenCV
import cv2
```

```
# 引入人臉識別庫OpenCV
import dlib
```

2. 分類名稱

```
# 分類名稱
name = ['Obama','TZUYU']
```

3. 載入訓練好的模型數據

```
# 載入訓練後的模型
model = cv2.face.EigenFaceRecognizer_create()
model.read('data/selfcascades/predict_face.xml')
```

4. 指定要辨識的圖片檔路徑

```
# 選擇要辨識的圖片檔
imagePath = "data/image/who.jpg"
```

人臉辨識-人臉識別

5. 將影像轉換成灰度圖，並使用dlib庫提供的人臉提取器，提取灰度圖中的人臉資訊

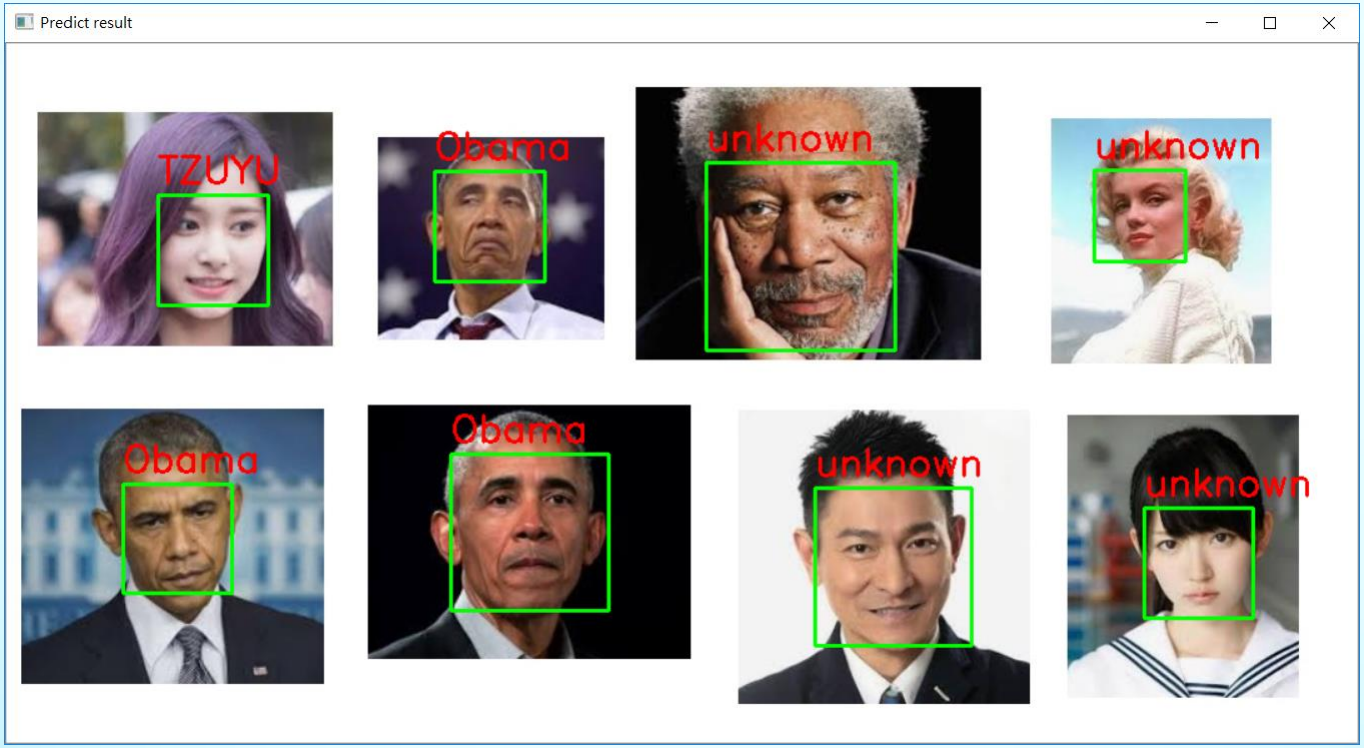
6. 遍尋人臉資訊，使用模型預測人臉

```
# 遍尋每張人臉
for face in faces:

    x = face.left()
    y = face.top()
    w = face.width()
    h = face.height()

    # 繪製矩形框住人臉
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # 擷取圖片中，人臉的部位圖片
    face = cv2.resize(gray[y:y+h, x:x+w], (400,400))

    # 使用模型去預測臉
    params = model.predict(face)

    # params[0] : 表示對應的標籤 / params[1] : 表示閾值
    if(params[1] < 16500):
        who = name[params[0]]
    else:
        who = 'unknown'
        cv2.imshow(('%d.unknown face' % num), face)

    # 繪製文字於矩形上
    cv2.putText(image, who, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
```

人臉辨識-人臉識別(課堂練習)

利用前面章節『人臉辨識-人臉識別』講解的步驟，

讀取工作區目錄 data/image 中who.jpg圖檔，判斷出圖片中人物



問題與討論