第一章 OpenCV (Open Source Computer Vision)

I 常見範疇應用：

包含了電腦視覺 (Computer Vision)、機械學習及影像處理演算法的集合，共 2,500 多種演算法，為跨平台的免費商業化開放式授權原始碼 (BSD license)。電腦視覺演算法可以用在如偵測及辨識臉部，或是其他即時移動中的物體、分析人物動作行為、移動視訊裝置追蹤、移動物體追蹤、物件立體化及產生三維座標立體視覺、影像縫圖產生高解析度全場影像、藉由特徵資料庫找出相似圖形、移除相機紅眼效果、眼球追蹤、辨識場景及創立標記擴增實境等

OpenCV 現今超過 900 萬下載人次，並且具專屬的有 Yahoo Group 論壇及眾多文件。 OpenCV 被使用在 Google、Yahoo、Intel、IBM、Sony、Honda、Toyota 等公司，並被佈署於如街道圖、入侵偵測、監控設備(中國綠壩)、機器人導航及閃躲物體、游泳池溺水事件、視覺互動藝術、跑道檢查、工廠產品檢測及快速臉部偵測等

OpenCV 朝著即時 (Real-time) 的應用開發，使用 MMX、SSE 及 IPP 指令架構，由純 C++ Template 所發展的無縫式容器，由影像裝置所產生的原始資料建立矩陣類別及函式庫，支援 Python、Java 等等電腦語言、跨平台的使用如：Windows、Linux、Android、Mac OS 等等系統，演算法的最佳化提供了商業化的產品加速運算處理優勢，BSD license 更能使企業能更輕鬆容易的修改原始碼

1. 人機互動
2. 物體識別
3. 圖像分割
4. 人臉識別
5. 動作識別

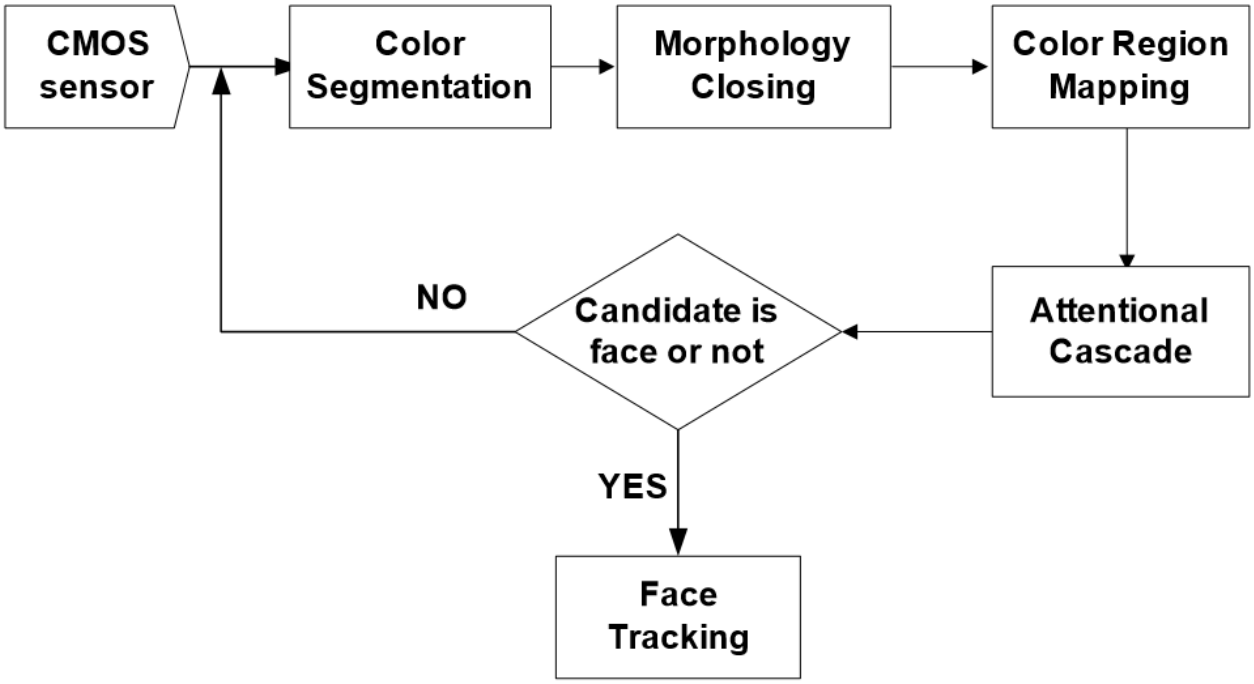
6. 運動跟蹤
7. 機器人

第二章 人臉辨識原理

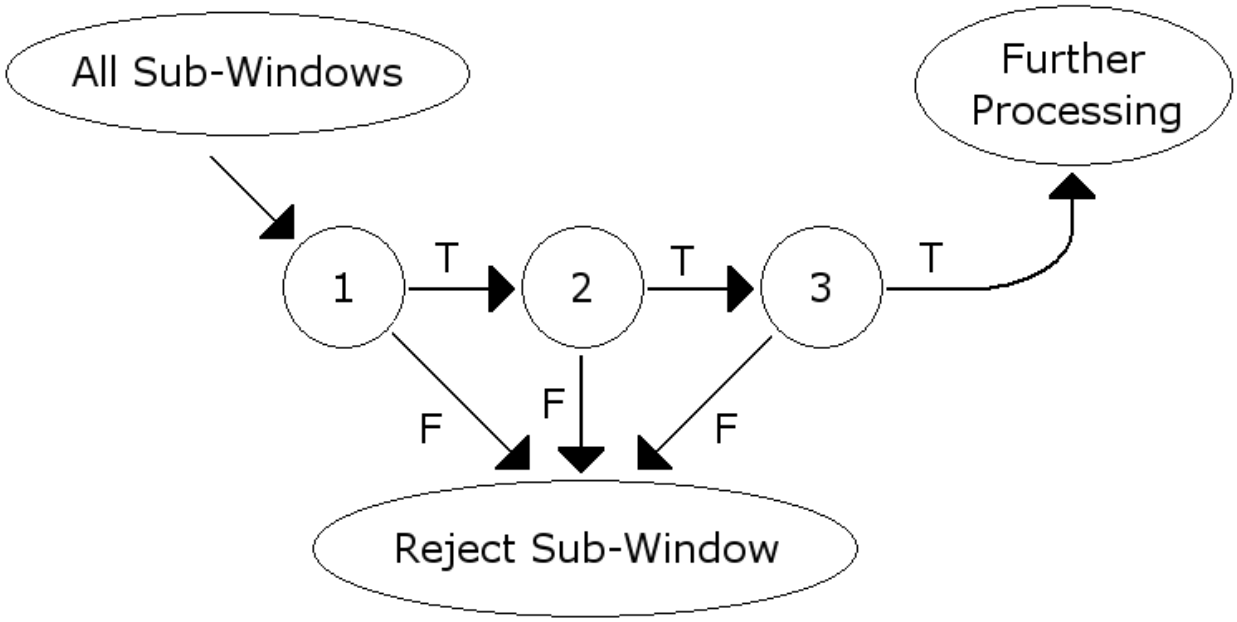
I 目前在眾多生物辨識方法(如虹膜辨識、指紋辨識、掌紋辨識及人臉辨識)中，其中以人臉辨識最廣為被企業和教學單位所使用和研究，因為不需要穿戴額外的裝置，更不需要和受測裝置有任何的接觸，只要透過簡單的攝影照相裝置，即可取得辨識所需的資料，因此可說是最方便的辨識方法之一。但是，由於影像辨識易受外界環境所干擾，例如：光源、角度和遮罩物等，加上所需處理的資料量大，因此辨識需要的時間長，所以如何排除這些外在因素，又能以最短的時間達到辨識的效果，即是人臉辨識的一大挑戰

II 人臉辨識系統是由人臉偵測與人臉辨識兩部分組成。首先，人臉偵測 為人臉辨識系統的首要步驟，目的就是對輸入影像做人臉偵測，如果在人臉偵測的步驟發生錯誤，那將會影響後續的人臉辨識步驟。因此如何在複雜的背景中，正確擷取出人臉影像是非常重要。

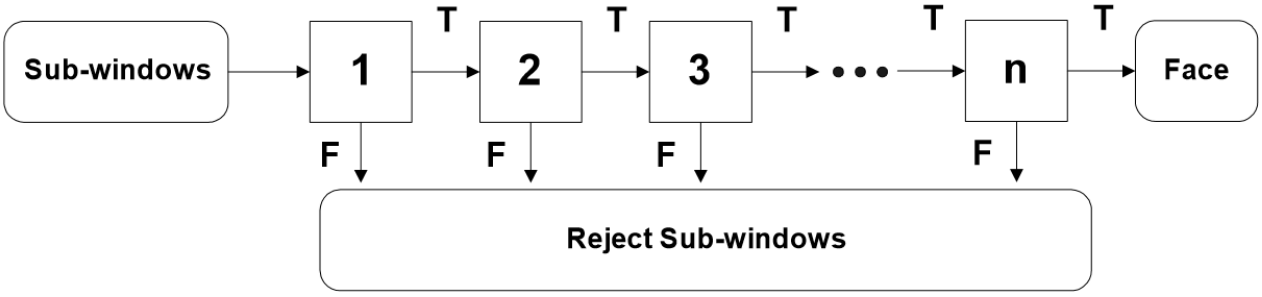
III 首先，就是要可以偵測出人臉，在環境中找到人臉所在的位置，當偵測到人臉之後再交由人臉追蹤系統進行持續地追蹤(認得是否人臉)；或是剪裁出人臉區域圖片再交由顏面辨識系統進行檢驗(檢驗身份)



人臉偵測之流程圖如圖所示。在人臉偵測的過程當中，系統會先將輸入的影像進行膚色的色彩分割，並且透過形態學的閉合運算（Closing）填滿膚色區域內空洞及不連續之處。接著我們透過膚色區域投影將屬於膚色的區域框選出來，並且利用投影的長寬比先排除可能不屬於人臉的區域，再將可能屬於人臉的區域交由基於特徵法（Feature-based methods）來判別此區域是否為正面的人臉



Face Detection 基本作法就是先做 Feature Extraction，接著做 Cascade Detection。特徵擷取(Feature Extraction)沒有問題，那麼瀑布偵測(Cascade Detection)是什麼呢？其實就只是一連串的檢查動作，通過第一關才能進入第二關，通過第二關才能進入地三關.....直到通過最後一關，才會將此特徵辨識為人臉。在偵測階段時，因為大部分的 window 沒有包含人臉，於是提早拒絕負面的 window，來加快偵測速度



也就是將輸入的影像進行全域的掃描（Full Searching），此方式是先在影像中取出一小區塊的影像視窗（Sub-window），接著利用各種特徵或者訓練過的類神經網路對此影像視窗進行人臉特徵的檢驗，若不是人臉的話，則再取下一塊影像視窗。為了準確地偵測出人臉，通常會包含許多特徵及判斷條件，雖然有一定的準確率，但是相對來說，也會消耗比較多的運算量在處理非人臉的影像視窗

IV 人臉偵測的研究已經發 展多年，且也已有相當多的論文發表，但近年的人臉偵測方法可劃分為 四種類型：Knowledge-Based、Feature invariant、Template matching 和 Appearance-Based

1 **Knowledge-Based 的方法主要是將一般人臉特徵的關係，轉成一些規則。例如，人臉影像通常會有一雙對稱的眼睛、一個鼻子、一個嘴巴，因此可以利用它們間的距離與位置來訂出規則。然而 Knowledge-Based 的**

缺點是，規則的制定是否符合所有人臉的五官特性。如果要求過高，勢必會降低準確度。反之也會降低準確度

2 **Feature invariant** 方法主要是以邊緣偵測的方式擷取臉部特徵如眉毛、眼睛、鼻子、嘴巴及膚色，以統計的方式來描述它們之間的關係。此方法的缺點是，這些人臉特徵會受到亮度、雜訊及遮蔽物的影響

3 **Template matching** 方式是先人工定義出一個標準樣本，計算出此標準樣本的相關數值，如人臉輪廓、眼睛、鼻子及嘴巴。然後將輸入影像基於此標主樣的相關數值做計算及比較。雖然此方法較簡單且易實現，但缺點是只適用於與標準影像相同大小且位置及的人影像

4 **Appearance-Based** 的方法一般是透過統計分析及機器學習的方式找出人臉與非人臉間的特性。因此必須使用到分佈模型或鑑別函數，同時須使用降低維度來減少計算的複雜度。此方法的缺點是較複雜且需要大量的樣本學習，才可能達到較高的效能

5 **Lin** 提出結合膚色及五官距離的人臉偵測方法。人臉偵測架構主要分成兩個部分，第一部分是使用膚色及基於三角形分割來搜尋可能的人臉區域。第二部分是以前饋式多層感知類神經網路來進行人臉識別。此方法的優點是在膚色偵測中，將光線因素考慮進去，因此可大大減少不同光線下的膚色偵測錯誤，且可適用於不同人臉大小及不同角度及表情。尤其，在複雜背景的情況下，可以有效地提高人臉偵測的速度。缺點是如果背景與膚色相近，就無法排除背景雜訊，且會在三角形偵測步驟中花費過多時間，進而導致整體系統速率的降低

6 **Huang** 提出將三種人臉偵測方法以串連及並聯的方式做結合。這三種不同的人臉偵測，各自以不同的人臉特徵來加強人臉偵測，包括 **2D Haar wavelet**、梯度方向**(gradient direction)**及賈柏濾波器**(Gabor filter)**。此方法的優點是三種特徵皆經由主成份分析**(Principle Component Analysis, PCA)**降低維度後，以多項式類神經網路**(Polynomial neural network, PNN)**為分類器來做分類。接著將效率較差且簡單的偵測方法放在前面，而效率較高且複雜的偵測方法放置後方，以多重步驟的方式來提高偵測速度。同時，每一個步驟的輸出都有不同的輸出權重並且與前一步驟的輸出

權重作加總來提高偵測率。此方法的缺點是雖然方法與 Viola 所提的方法有點類似，皆以層疊的方式串接分類器，但此方法使用三種特徵來進行人臉偵測，由於三種不同特徵都不是以簡單的方式取得特徵，所以所花的時間成本也相對的比較高，因此並不適合於即時人臉偵測與辨識系統。

V 然而人臉不是固定不變，因此不管是哪一類型的人臉偵測方法都會受到下列因素的影響：(1)姿勢(2)形狀(3)表情(4)亮度(5)複雜背景的影響。近年來人臉偵測研究已有相當多的研究成果發表，其層疊(cascade)分類器在人臉偵測方面速度快且準確度
https://github.com/opencv/opencv/tree/master/data/haarcascades

VI box.py

```
1   import cv2
    import config
    import face
    import hardware
2   if __name__ == '__main__':
3       print 'Loading training data...'
4       model =
    cv2.face.createEigenFaceRecognizer(config.COMPONENT_NUM
    BER, config.POSITIVE_THRESHOLD)
5       model.load(config.TRAINING_FILE)
6       print 'Training data loaded!'
7       camera = config.get_camera()
8       box = hardware.Box()
9       box.lock()
10      print 'Running box...'
11      print 'Press button to lock (if unlocked), or unlock if the correct
    face is detected.'
12      print 'Press Ctrl-C to quit.'
13      while True:
14          if box.is_button_up():
15              if not box.is_locked:
16                  box.lock()
17                  print 'Box is now locked.'
18              else:
19                  print 'Button pressed, looking for face...'
20                  box.starttest()
21                  image = camera.read()
22                  image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
23                  result = face.detect_single(image)
24                  box.endtest()
```

```
25                 if result is None:
26                     print 'Could not detect single face!  Check the image
    in capture.pgm' \
27                         ' to see what was captured and try again with only
    one face visible.'
28                     continue
29                 x, y, w, h = result
30                 crop = face.resize(face.crop(image, x, y, w, h))
31                 label = model.predict(crop)
32                 if label == config.POSITIVE_LABEL:
33                     print 'Recognized face!'
34                     box.unlock()
35                 else:
36                     print 'Did not recognize face!'
```

1 匯入所需程式庫

2 **if __name__ == '__main__':**

判斷是否是相同的 Python 檔案

3 **print 'Loading training data...'**

列印訓練集資料載入中訊息

4 **model = cv2.face.createEigenFaceRecognizer(config.COMPONENT_NUMBER, config.POSITIVE_THRESHOLD)**

```
    依 config.COMPONENT_NUMBER 指定的特徵數量、
    config.POSITIVE_THRESHOLD 允收距離，來建立特徵臉鑑別器
5       model.load(config.TRAINING_FILE)
    載入 config.TRAINING_FILE 指定的訓練集檔案
6       print 'Training data loaded!'
    列印訓練集資料完成載入訊息
7       camera = config.get_camera()
    取得拍攝模組
8       box = hardware.Box()
    建立電子鎖物件
9       box.lock()
    將電子鎖上鎖
10      print 'Running box...'
11      print 'Press button to lock (if unlocked), or unlock if the correct
    face is detected.'
12      print 'Press Ctrl-C to quit.'
    10~12 列印電子鎖已啟動、上鎖與提示訊息
13      while True:
    無窮迴圈
14          if box.is_button_up():
    假如 S1 開關按下又放開
15              if not box.is_locked:
    假如電子鎖式解鎖狀態
16                  box.lock()
    將電子鎖上鎖
17                  print 'Box is now locked.'
    列印電子鎖已上鎖訊息
18              else:
    否則
19                  print 'Button pressed, looking for face...'
    列印電子鎖 S1 開關按下又放開，正進行拍攝、檢驗訊息
20                  box.starttest()
    電子鎖開始檢驗
21                  image = camera.read()
    拍攝一張相片
```

```
22              image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
```
將圖片灰階化
```
23              result = face.detect_single(image)
```
判斷是否為單一人臉的圖片
```
24              box.endtest()
```
電子鎖結束檢驗
```
25              if result is None:
```
假如無回傳值
```
26                  print 'Could not detect single face!  Check the image in
capture.pgm' \
27                      ' to see what was captured and try again with only one
face visible.'
```
26~26 列印無法偵測到人臉訊息
```
28                  Continue
```
回到迴圈起始位置
```
29              x, y, w, h = result
```
取得欲裁切的資料
```
30              crop = face.resize(face.crop(image, x, y, w, h))
```
進行裁切、鎖放人臉圖片
```
31              label = model.predict(crop)
```
進行比對檢驗評估
```
32              if label == config.POSITIVE_LABEL:
```
假如回傳值為 config.POSITIVE_LABEL 時
```
33                  print 'Recognized face!'
```
列印人臉驗證成功訊息
```
34                  box.unlock()
```
電子鎖解鎖
```
35              else:
```
否則
```
36                  print 'Did not recognize face!'
```
列印不認得此人臉訊息

VII 常見問與答

1 為何不使用 Python 3.X？

因為目前顏面辨識仍有部分程式庫尚未修改為適用於 3.X 版

2 如何提高辨識穩定度？

信心水準越高，信賴區間的範圍越大，誤差範圍越大，也就是說辨識越寬鬆

3 為什麼我已經靠很近了，還是無法辨識？

太接近鏡頭反而造成難以取得全臉影像；保持約 **20~30cm** 距離較易取樣

4 顏面辨識誤差大且容易偽造，因此不建議使用於保全要求高的需求，可作為輔助識別機制

5 樹莓派主電源為 **3.3V** 輸入端嚴禁超過 **3.3V** 的電壓輸入，以免燒毀樹莓派