投稿類別：資訊類

篇名：

防盜檢測系統-基於 Python 之 OpenCV 人臉偵測結合資料庫存儲

作者：
李鳳緯。台中市立西苑高中。高二 7 班

指導老師：
陳永錚 老師

壹、前言

一、研究動機

研究者在學校社團辦公室自主學習時，發現社辦雖然有許多貴重財產，卻沒有妥當的防盜系統，單單只有一道傳統的喇叭門鎖及門外只有一台的監視器。因為本來就有寫程式的興趣，故決定實作一套基於 Python 語言的 OpenCV 人臉辨識程式，在有人進入社團辦公室時，擷取當事人的臉並且記錄當下時間，傳到網頁上做紀錄，以此通知有人進入，達到比原來更完善的防盜系統。

二、研究目的

本研究目的是實作出一整套系統，此套系統能結合攝影機，實際安裝在門上且辨識人臉，並回傳人像照片及照相時間，達到真正意義上能監測是否有人進入社辦，且藉由時間判斷是否為正當使用此公共空間。

貳、文獻探討

本節將介紹此實驗所應用到軟、硬體之簡單介紹。

一、OpenCV

（一）OpenCV 簡介

「OPENCV 的全稱是 Open Source Computer Vision Library，是一個跨平台的電腦視覺資料庫。」(OpenCV，2021)，能利用常見的 C++、C、Python，Java 作為撰寫工具。

OpenCV 普遍被應用於影像辨識，動作追蹤上，在 OpenCV 官網上介紹主要應用並解決的領域則有如下：**「擴增實境、臉部辨識、手勢辨識、人機互動、動作辨識、動作跟蹤、物體辨識、圖像分割、機器人。」**（OpenCV，2021)

（二）人臉檢測原理

目前有兩種主要的人臉辨識方法，主要是分為：基於知識的辨識方法，是根據眼睛、眉毛、嘴巴、鼻子等器官及相互間的幾何關係來檢測人臉（Ello，2012），也就是說，基於知識的辨識方法，需要利用人的五官位置來作判斷，而另一種判斷方法，則是基於統計的方法：將人臉看作一個整體二維畫素矩陣，從統計的觀點通過大量人臉影像樣本構造，根據相似度來判斷人臉是否存在。（Ello，2012）

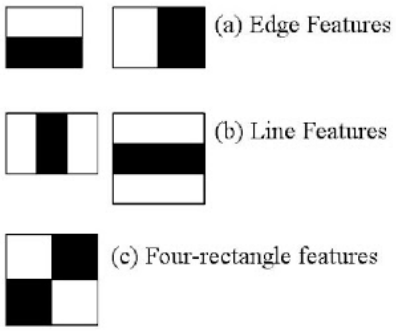
1、哈爾特徵檢測原理

是一種即時的人臉檢測演算法，運用大量的人類正臉圖像樣本訓練，以辨識速度快、硬體需求低，常常會被應用在行動裝置或是樹莓派(Raspberry Pi)等低階硬體裝置上（山姆大叔，2021），而這種演算法主要的辨識方式，則是透過小的方形矩陣，如同掃地機器人般由左上至右下，找尋人臉較明顯的特徵，像是鼻子與眼睛的明暗，眼睛的亮度通常會低於鼻頭。從而檢測出實際的臉部位置（山姆大叔，2021）。

（1）Haar-like 特徵

Haar-like 是哈爾特徵檢測下的演算方式，以明暗區塊圖作人臉特徵紀錄，範例如圖（一），如上所說的鼻子與眼睛的對比，再透過非人類的訓練圖像與人類的訓練圖像不斷的給予訓練，程式將會在訓練中學會如何分辨人臉與非人臉的特徵，也會有專門存儲錯誤與正確分類的資料庫，以此來作分析與判斷此訓練的數值是否需要持續作增加，再去更動不同類型圖片的權重，使正確率能夠伴隨訓練次數上升。(OpenCV，2022)

圖一：Haar-like 特徵圖解析



（資料來源：OpenCV 官方網站）

（2）AdaBoost 演算法

AdaBoost 演算法(Adaptive Boosting)，是一種可將弱學習器轉為強學習器的演算法，而這個演算法的機制與原理，則是先從基礎訓練集訓練出一個基本學習器，再根據基礎學習器表現對樣本進行調整，之後不斷提高錯誤樣本的權重。，Adaboost 就是在過程中不斷的做增益(boosting)，增益原本的舊訓練器，以此達到高效且高準確的判斷。

（3）基於 Haar 特徵的級聯分類器

檢測圖片中，通常有一大部分都不是人臉的範圍，所以要是每張圖片都必須慢慢檢測如此大面積的範圍，效率和時間都會變得十分差勁，所以利用 Cascade of Classifiers，利用瀑布式演算法，將失敗的直接丟棄，而可能有人類臉孔的則是不斷做關注和增加訓練，將訓練的所有樣本分成數個階段，分類在不同等級的分類器中，失敗則放棄，而成功則進到更強的分類器做運算。(OpenCV，2022)

二、微處理器 – 樹莓派 Rasberry Pi

「一系列低成本、手掌大小的單板電腦。由英國 Rasberry Pi 基金會開發出來；以推廣電腦科學基礎教育為宗旨。」（王進德，2021），樹莓派簡單來說就是一台微型電腦，只不過需要額外下載作業系統，就能照正常電腦使用，幾乎所有應用於正常桌機上的功能都能在樹莓派上執行，能夠更便利於架設，也能隨時進行運算，不必到處裝上如桌機、機台等過大的系統，除了能夠更便利的進行部屬，樹莓派的耗電量也很低，因為其採用了 ARM 架構的處理晶片。

三、網頁架設

Python Flask 是一種輕量化的網頁架構，能夠利用 Python 快速做出前端與後端結合的輕量化網頁，也可以只開發需要的功能，避免版面太過雜亂。(Enoxs，2021)，從簡單的路由、網頁模板，都能利用 Flask 套件達成目的。

叁、研究方法

本節將介紹實驗中所使用的兩種方法，分為（一）文獻研究法、（二）程式編寫實作，研究架構則針對於實作部分做編排。

一、研究架構

研究架構著重於程式編寫實作區塊，分為四大區塊如下圖（二），文獻研究法為輔助實作而並行。

圖二：研究架構圖

資料來源：研究者自行繪製

二、研究方法

（一）文獻研究法

能夠根據一定的研究目的去做資料的統整與蒐集，從而全面且精準的濃縮所需研究問題的方法。而在單純針對人臉辨識、網頁架設及樹莓派時，也會有十分大量的資源，也不免利用此方法去做資料的蒐集，才能將這三樣物件了解透徹，進而將不同面向的物品結合在一個系統中。

（二）程式編寫實作

1、OpenCV 程式編寫

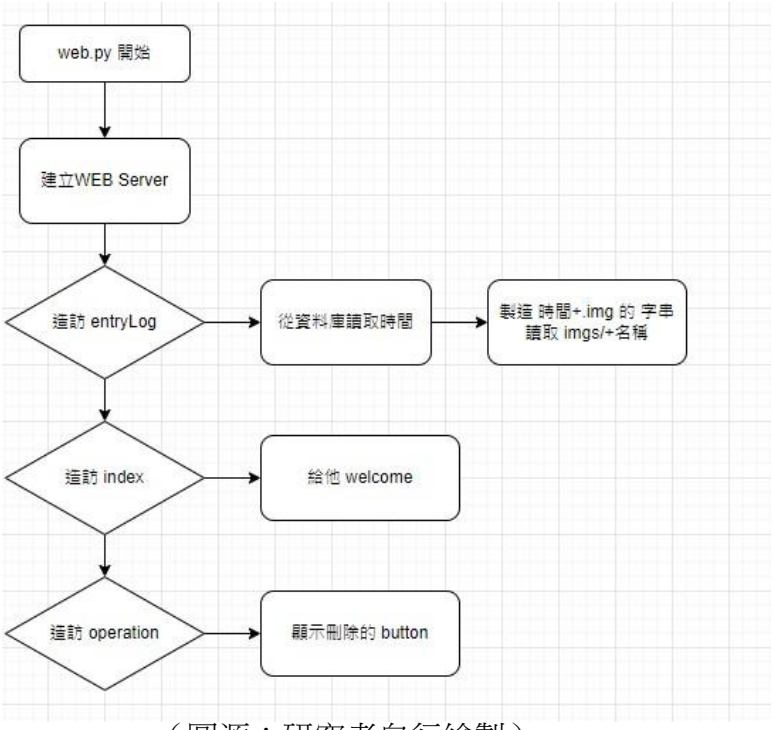
在研究過程中，OpenCV 的主要功能為讓我能夠對人臉做偵測的動作，所以我需要尋找或靠自行完成一個訓練好的人臉偵測模型，能夠在偵測到人臉之後以綠色方框圈起來，並且持續偵測，偵測成功數超過 20 次才會正式確定為人臉，且製作防呆，在三秒未偵測到物體時，取消偵測，來達到完全避免偵測到物體而意外佔到內存的情況，也避免在辨識中，誤判物體為人，也將偵測成功的照片以時間命名，存儲到 img 資料庫中，等待網頁主程式造訪。

2、網頁架設：Python Flask

程式流程圖如下圖（三），先建立一個 web server，在造訪 OpenCV 圖片存儲之資料庫，讀取本來設定的時間，也就是偵測成功 img 的檔名，以利同時

做時間的紀錄與顯示，介面也設計出能夠刪除資料庫所有檔案的功能，避免內存不足。

圖三：網頁程式之流程圖



（圖源：研究者自行繪製）

3、樹莓派：程式匯入與建置環境

使用樹莓派前，需要先將作業系統燒入電腦，之後再將程式碼經由樹莓派的環境下載下來，才能將我寫的程式匯入到微電腦，之後將攝影機，鍵盤，螢幕，滑鼠都連上並且測試（一）、（二）之系統是否正常運行。

4、實際安裝與測試

實際將建置好的樹莓派連接上攝影機，安裝上社辦門口，進行一定數量的偵測，並判斷分析偵測之圖片。

肆、研究分析與結果

本節將演示程式編寫實作，實際測試之結果。分為人臉辨識、網頁架設及樹莓派建置

一、OpenCV 人臉辨識

我利用 OpenCV 來編寫人臉辨識的程式碼，編寫基礎的程式使鏡頭能偵測到人臉輪

廓時，能夠將人臉用綠色的框框住，我採用的方法為利用 OpenCV 內建好的開源人臉資料庫，直接開始做編程。也有引入時間模組，以及創建 SQL 存儲到編寫好的網頁做顯示，範例如下圖（四）、（五）。

圖四：人臉辨識程式碼

```
while True:
    ret, frame = cap.read()
    if ret:
        frame = cv2.resize(frame, (0, 0), fx=1.2, fy=1.2)
        #cv2.imread(frame)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faceCascade = cv2.CascadeClassifier('face_detect.xml')
        faceRect = faceCascade.detectMultiScale(gray, 1.5, 3)
        for (x, y, w, h) in faceRect:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.imshow('video', frame)
```

圖片來源：研究者自行編寫

圖五：人臉辨識程式碼

```
cap = cv2.VideoCapture(0)
try:
    conn = sqlite3.connect('entry.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS logs
        (ID INTEGER PRIMARY KEY     AUTOINCREMENT,
        TIME           VARCHAR(150)    NOT NULL);''')
    print ("Successful")
    conn.commit()
except:
    print("Cannot Conect To Sqlite3 DB")
```

圖片來源：研究者自行編寫

也在程式中設計了避免誤判到似人物體的防呆裝置，必須要在連續偵測成功 20 次以上才回傳資料給網頁，且在誤判後沒有持續輸入人臉裝置時，每三秒就會清除一次統計值，也是為了避免偵測到非人物體。程式如下圖（六）、（七），而實際測試則如圖（八）：

圖六：防呆部分起始部分程式碼

```
def runcam():
    import cv2, time, sqlite3, os
    detectTime = 0 ; detectLast = 0
    clearTrig = 3 #在未輸入幾秒重置計算張數
    confirmTrig = 20 #20張圖就算成功
    start = time.time() #設置起始時間
```

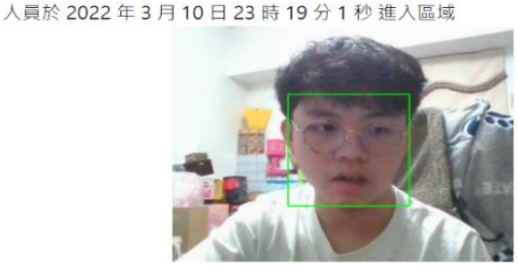
圖片來源：研究者自行編寫

圖七：防呆運算部分程式碼

```
        if detectTime > detectLast:
            detectLast = detectTime
    else:
        break
    print(detectTime)
    if detectTime > confirmTrig:
        print("座標: ",faceRect)
        nowTime = round(float(time.time()),4)
        cv2.imwrite(f"imgs/{str(nowTime)}.png", frame)
        c.execute(f"INSERT INTO logs (time) VALUES ({str(nowTime)})")
        conn.commit()
        detectTime = 0
        print("Got Your Face")
        start = time.time()
    if time.time()-start > clearTrig and detectTime!=detectLast:
        print(time.time()-start)
        detectTime = 0
        start = time.time()
        print("TIME OUT")
```

圖片來源：研究者自行編寫

圖八：研究者（本人）被偵測圖片在網頁的紀錄



圖片來源：研究者自行拍攝

二、網頁架設實作

網頁設計架構使用 Python 中的 Flask 套件，其中包含 Render_Template 以及 request 使用 Render_Template 模組產生動態內容提供給使用者，而 request 模組則可以判斷用戶是否有提交操作(POST)的動作。並搭配 SQLite 資料庫來儲存時間等紀錄，且未來可以做擴張。上述所使用套件將前端與後端處理整合在 Python 腳本裡，實際介面如圖（九）、（十）、（十一）、（十二）：

網頁具備以下三大功能:

（一）開啟 SQLite DB 檔案，並且進行讀取/寫入/刪除

1、讀取時間(格式為 1970/1/1 至今日期的秒數)並且轉換為人類可讀
2、以時間作為圖片之檔案名稱來提取圖片並遷入置網頁內容
3、可以透過網頁直接將所有紀錄重設

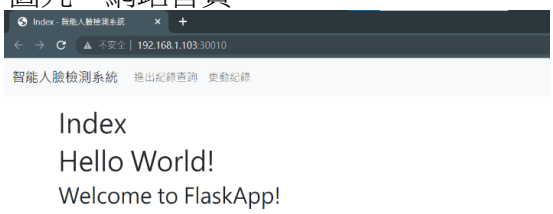
（二）令使用者以瀏覽器操作/檢視進出入紀錄

1、根據資料庫中的數量進行網頁內容動態生成
2、以 Flask Route 功能來制定路由（例：/ , /entryLog , /operations）

（三）簡單明瞭的操作介面

1、使用開源 css 作為網頁美化排版 (Bootstarp)

圖九：網站首頁



圖片來源：研究者自行截圖

圖十：點擊進出紀錄查詢

圖片來源：研究者自行截圖

圖十一：點擊更動紀錄

圖片來源：研究者自行截圖

圖十二：刪除資料庫畫面

圖片來源：研究者自行截圖

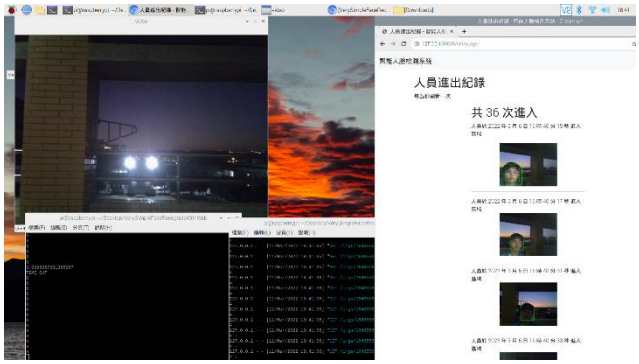
三、樹莓派：程式匯入與建置環境

在使用樹莓派之前，需要先購買 SD 卡，以及準備好螢幕、滑鼠、鍵盤，攝影機，以及足以提供充足瓦數的電供，先利用原桌機搜尋樹莓派官網，將作業系統燒錄至 SD 卡中，之後直接放入樹莓派之插槽，就可以開機做使用。然後利用樹莓派作業系統，直接將程式碼下載做使用。

四、實際安裝與測試

實際將樹莓派，攝影機安置好之後，就能夠在社辦門外出入，給攝影機偵測做辨識的測試，樹莓派以及開啟程式後介面如下圖（十三）：

圖十三：樹莓派開啟程式後介面



圖片來源：研究者自行截圖

伍、研究結論與建議

本節會將研究之結果做統整，以及分析對於此裝置未來改進之建議。

一、研究結論

研究結論將由機構，以及判斷時的正確率作探討。

（一）連動機構

在持續偵測約一小時後，儘管沒有人進入鏡頭被偵測，微處理器也會因程式過度運轉而使運行逐漸緩慢，在緩慢下雖然檢測資料的載入變慢，但偵測效率依然還在，不用擔心會有空檔而辨識不出來。

（二）偵測時有無口罩之差別

測試近百筆資料後，發現戴口罩完全辨識不出來，故沒有實驗記錄，而不戴口罩的成功率則為 100%，唯一失敗的因素僅為場地的燈光過亮，導致攝影機反光無法辨識。

（三）程式對於誤判的結果處理

因程式碼設定了防呆機制，故在 36 筆實驗資料中，沒有任何一次誤判，包含了側臉，過黑都成功辨識，圖例如下圖（十四）、（十五）：

圖十四：模糊之圖例

圖片來源：研究者自行截圖

圖十五：過黑之圖例

圖片來源：研究者自行截圖

二、研究建議

本節提供之建議分別針對機構面與程式未來之發展走向。

（一）因為實驗在亮度太高時，會無法判斷，所以可以考慮製造遮光罩，來避免只要開手電筒就能無效化此攝影機。

（二）因目前的設計，只有相同網路才能連接資料庫，也只有連上網站才能做資料查看，所以我認為可以連動手機，只要有人進入就會收到訊息，做到即時的監測。

陸、參考文獻

1. Ello（2012 年 4 月 28 日）。淺析人臉檢測之 Haar 分類器方法。
   https://www.cnblogs.com/ello/archive/2012/04/28/2475419.html

2. 山姆大叔（2021 年 9 月 22 日）。[Day9] Face Detection - 使用 OpenCV & Dlib：Haar cascades。
   https://ithelp.ithome.com.tw/articles/10263258

3. Enoxs（2021 年 9 月 26 日）。Python Flask 入門指南 ： 輕量級網頁框架教學。
   https://devs.tw/post/448

4. 王進德（2016）。**CP20 Raspberry Pi 樹莓派初學指引**。啟芳出版社。

5. OpenCV（2022 年 3 月 12 日）。OpenCV Tutorials。
   https://docs.opencv.org/4.x/d9/df8/tutorial_root.html