

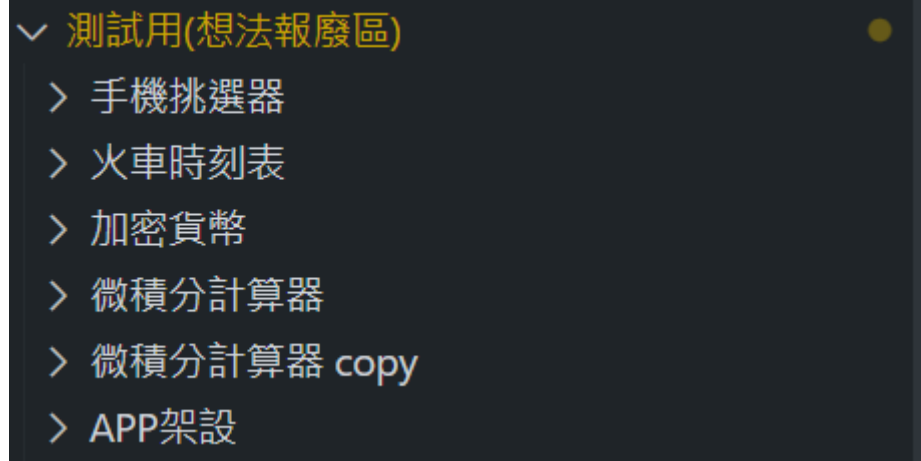
# 視訊畫家

Opencv的應用以及延伸學習

機械一A 112303522 王是鈞

# 主題發想

到要交報告的前一個禮拜才決定要做甚麼，理想很豐滿但現實很骨感，一開始覺得可以憑藉著之前爬蟲的經驗，做有關於蒐集網路上資料並加以分析及整理(手機的不同、虛擬貨幣的變動等等)，但又想完成有關自己的APP(計算機、火車時刻表)，但後續的整理資料都太過繁雜，且視覺化弄起來相對主觀。因此轉頭朝向opencv發想，想到之前視訊會議需要注意大畫面中的某一部份相對吃力，讓演講者辛勞，因此想說能夠在視訊畫面上畫圖一定很酷，就有了視訊畫家的誕生。

- 
- ✓ 測試用(想法報廢區)
  - > 手機挑選器
  - > 火車時刻表
  - > 加密貨幣
  - > 微積分計算器
  - > 微積分計算器 copy
  - > APP架設

圖一-嘗試後又放棄的想法

# 模組介紹

核心功能：圖像處理、輪廓檢測、運動分析、三圍檢測、機器學習.....

應用範圍：影像拼接與全景圖生成、臉部辨識與檢測、AR.....



圖二-opencv

# 大致架構

先做一些小實驗來確認opencv的功用以及極限在哪裡，並把學習到的知識以及程式加以合併及使用，以達成目標。

想要將程式分成兩部分：

其一為確認畫面中的筆刷，可以利用hex色碼或是位置的捕捉來確認筆刷位置筆加以畫圖。

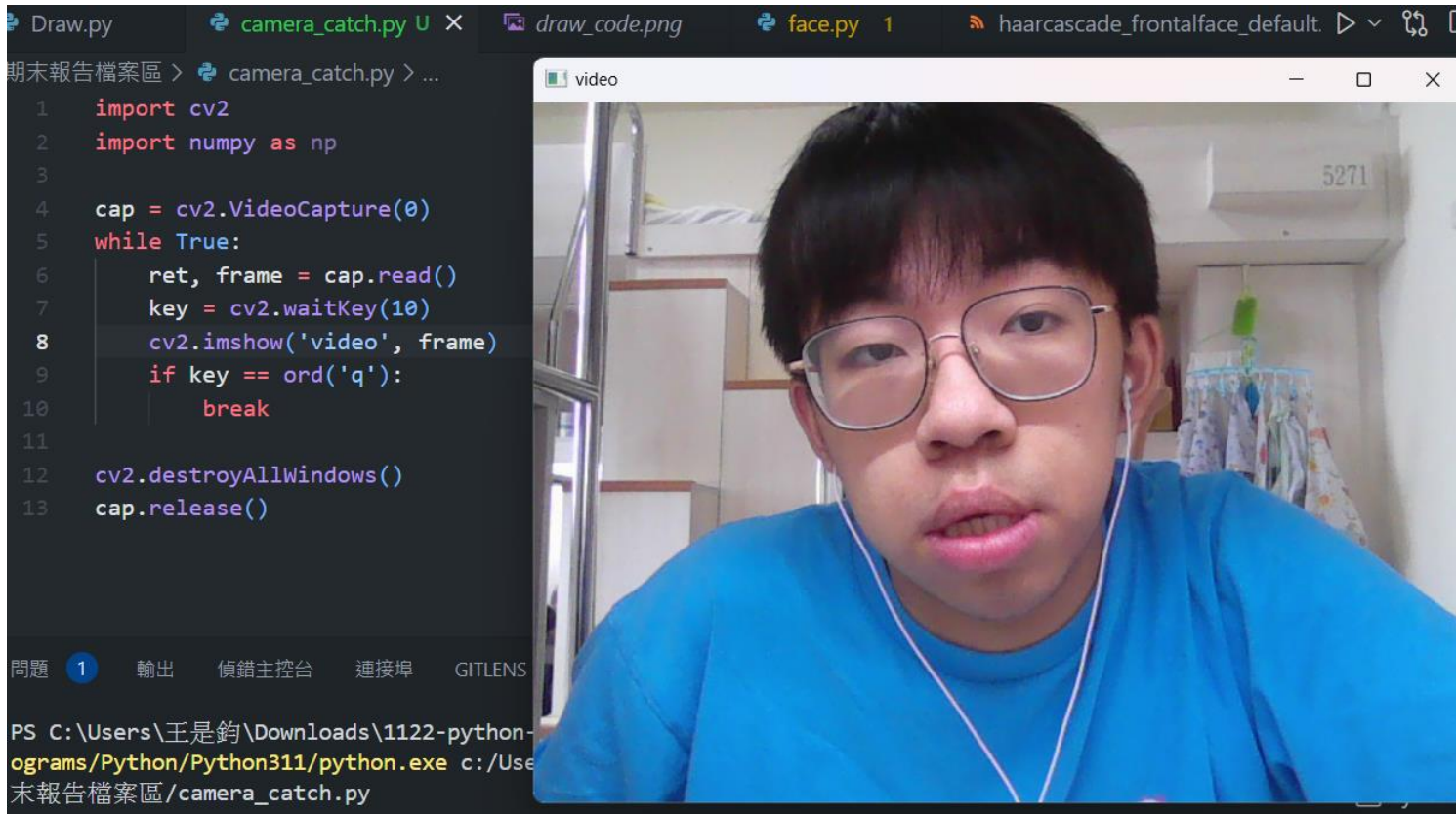
其二則是主程式的部分需要經由使用過後新增功能鍵來便利使用情境。



表一-程式流程大致設計

# 實驗過程-抓取視訊鏡頭

功用：透過opencv裡的函數可以方便我們直接獲得電腦上所搭配的視訊鏡頭，無須額外接線。



```
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    key = cv2.waitKey(10)
    cv2.imshow('video', frame)
    if key == ord('q'):
        break
```

VideoCapture可以讓我們抓取視訊鏡頭之外，Imshow()也可以讓我們有視窗可以看見鏡頭中的畫面，以利後繼續執行。

圖三-視訊畫面的捕捉

# 實驗過程-筆顏色的捕捉

功用：讀取視訊鏡頭後，將其轉換為HSV色彩空間，然後根據藍色的HSV值範圍創建一個遮罩(mask)，最後將遮罩應用到原始影像上，僅保留藍色部分。您可以根據需要調整藍色的HSV值範圍，以獲得想要的效果。

HSV代表色相（Hue）、飽和度（Saturation）和明度（Value）

```
while True:
    h_min = cv2.getTrackbarPos('Hue Min', 'TrackBar')
    h_max = cv2.getTrackbarPos('Hue Max', 'TrackBar')
    s_min = cv2.getTrackbarPos('Sat Min', 'TrackBar')
    s_max = cv2.getTrackbarPos('Sat Max', 'TrackBar')
    v_min = cv2.getTrackbarPos('Val Min', 'TrackBar')
    v_max = cv2.getTrackbarPos('Val Max', 'TrackBar')
    print(h_min, h_max, s_min, s_max, v_min, v_max)

    ret, img = cap.read()
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower = np.array([h_min, s_min, v_min])
    upper = np.array([h_max, s_max, v_max])

    mask = cv2.inRange(hsv, lower, upper)
    result = cv2.bitwise_and(img, img, mask=mask)

    cv2.imshow('img', img)
    cv2.imshow('hsv', hsv)
    cv2.imshow('mask', mask)
    cv2.imshow('result', result)
    cv2.waitKey(1)
```

圖四-創建遮罩顯示顏色



# 實驗過程-筆跡的繪製



圖五-原先的視訊畫面

程式執行後



圖六-處理後的視訊畫面

# 實際流程

最後設計出來的結果與先前的架構並無太大的差異，一樣沿用先前構想的方式去設計程式，大多時間會遇到的BUG也用了中央機房的GPT一同解決。



表二-使用流程圖



# 成果展示(一)-程式碼

Code Link:

1.Draw:

<https://ideone.com/a9YTD9>

2.Setting:

<https://ideone.com/MEUKOM>

功能:

1.Draw:

運行程式、開啟視訊畫面，按下  
不同按鍵有不同功能:

q:結束視訊

e:清除筆跡

d:開啟繪圖

2.Setting:

運行程式調整色筆、確認係數

```
1 import cv2
2 import numpy as np
3
4 def empty(v):
5     pass
6
7 #img = cv2.imread('XiWinnie.jpg')
8 #img = cv2.resize(img, (0, 0), fx=0.5, fy=0.5)
9
10 cap = cv2.VideoCapture(0)
11
12 cv2.namedWindow('TrackBar')
13 cv2.resizeWindow('TrackBar', 640, 320)
14
15 cv2.createTrackbar('Hue Min', 'TrackBar', 0, 179, empty)
16 cv2.createTrackbar('Hue Max', 'TrackBar', 179, 179, empty)
17 cv2.createTrackbar('Sat Min', 'TrackBar', 0, 255, empty)
18 cv2.createTrackbar('Sat Max', 'TrackBar', 255, 255, empty)
19 cv2.createTrackbar('Val Min', 'TrackBar', 0, 255, empty)
20 cv2.createTrackbar('Val Max', 'TrackBar', 255, 255, empty)
21
22 #hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
23
24 while True:
25     h_min = cv2.getTrackbarPos('Hue Min', 'TrackBar')
26     h_max = cv2.getTrackbarPos('Hue Max', 'TrackBar')
27     s_min = cv2.getTrackbarPos('Sat Min', 'TrackBar')
28     s_max = cv2.getTrackbarPos('Sat Max', 'TrackBar')
29     v_min = cv2.getTrackbarPos('Val Min', 'TrackBar')
30     v_max = cv2.getTrackbarPos('Val Max', 'TrackBar')
31     print(h_min, h_max, s_min, s_max, v_min, v_max)
32
33     ret, img = cap.read()
34     hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
35     lower = np.array([h_min, s_min, v_min])
36     upper = np.array([h_max, s_max, v_max])
37
38     mask = cv2.inRange(hsv, lower, upper)
39     result = cv2.bitwise_and(img, img, mask=mask)
40
41     cv2.imshow('img', img)
42     cv2.imshow('hsv', hsv)
43     cv2.imshow('mask', mask)
44     cv2.imshow('result', result)
45     cv2.waitKey(1)
46
47
```

圖七-setting.py的程式碼

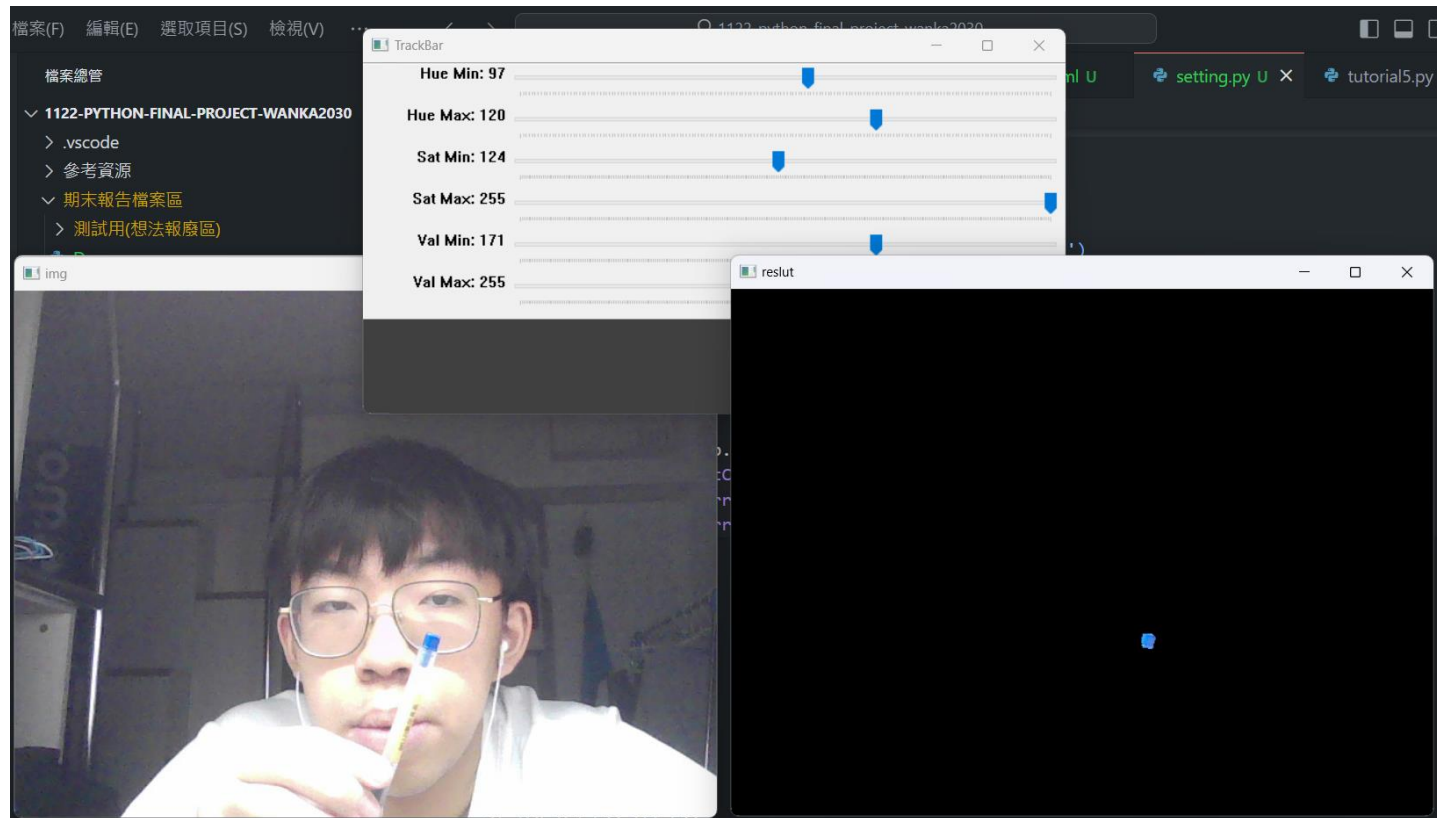
```
1 import cv2
2 import numpy as np
3
4 cap = cv2.VideoCapture(0)
5 penColorHSV = [[100, 50, 50, 140, 255, 255]]
6 penColorBGR = [[255, 0, 0]]
7 drawPoints = []
8
9 def findpen(img):
10     hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
11     for i in range(len(penColorHSV)):
12         lower = np.array(penColorHSV[i][0:3])
13         upper = np.array(penColorHSV[i][3:6])
14
15         mask = cv2.inRange(hsv, lower, upper)
16         result = cv2.bitwise_and(img, img, mask=mask)
17         penx, peny = findc(mask)
18         cv2.circle(imgContour, (penx, peny), 10, penColorBGR[i], cv2.FILLED)
19         if peny != -1:
20             drawPoints.append([penx, peny, i])
21     #cv2.imshow('result', result)
22
23 def findc(img):
24     contours, hierarchy = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
25     x, y, w, h = -1, -1, -1, -1
26     for cnt in contours:
27         #cv2.drawContours(imgContour, cnt, -1, (255, 0, 0), 4)
28         area = cv2.contourArea(cnt)
29         if area > 500:
30             peri = cv2.arcLength(cnt, True)
31             vertices = cv2.approxPolyDP(cnt, peri * 0.02, True)
32             x, y, w, h = cv2.boundingRect(vertices)
33
34     return x+w//2, y
35
36 def draw(drawpoints):
37     for point in drawpoints:
38         cv2.circle(imgContour, (point[0], point[1]), 10, penColorBGR[point[2]], cv2.FILLED)
39 drawEnabled = False
40 while True:
41     ret, frame = cap.read()
42     if ret:
43         imgContour = frame.copy()
44         findpen(frame)
45
46         if drawEnabled:
47             draw(drawPoints)
48
49         for point in drawPoints:
50             cv2.circle(imgContour, (point[0], point[1]), 10, penColorBGR[point[2]], cv2.FILLED)
51
52         cv2.imshow('video', frame)
53         cv2.imshow('contour', imgContour)
54     else:
55         break
56
57 key = cv2.waitKey(10)
58 if key == ord('q'):
59     break
60 elif key == ord('e'):
61     imgContour = frame.copy() # 清除筆跡
62     drawPoints = [] # 清空筆跡列表
63 elif key == ord('d'):
64     drawEnabled = not drawEnabled # 切換繪圖開關
65
66 cv2.destroyAllWindows()
67 cap.release()
68
```

圖八-draw.py的程式碼

# 成果展示(二)-實際運用

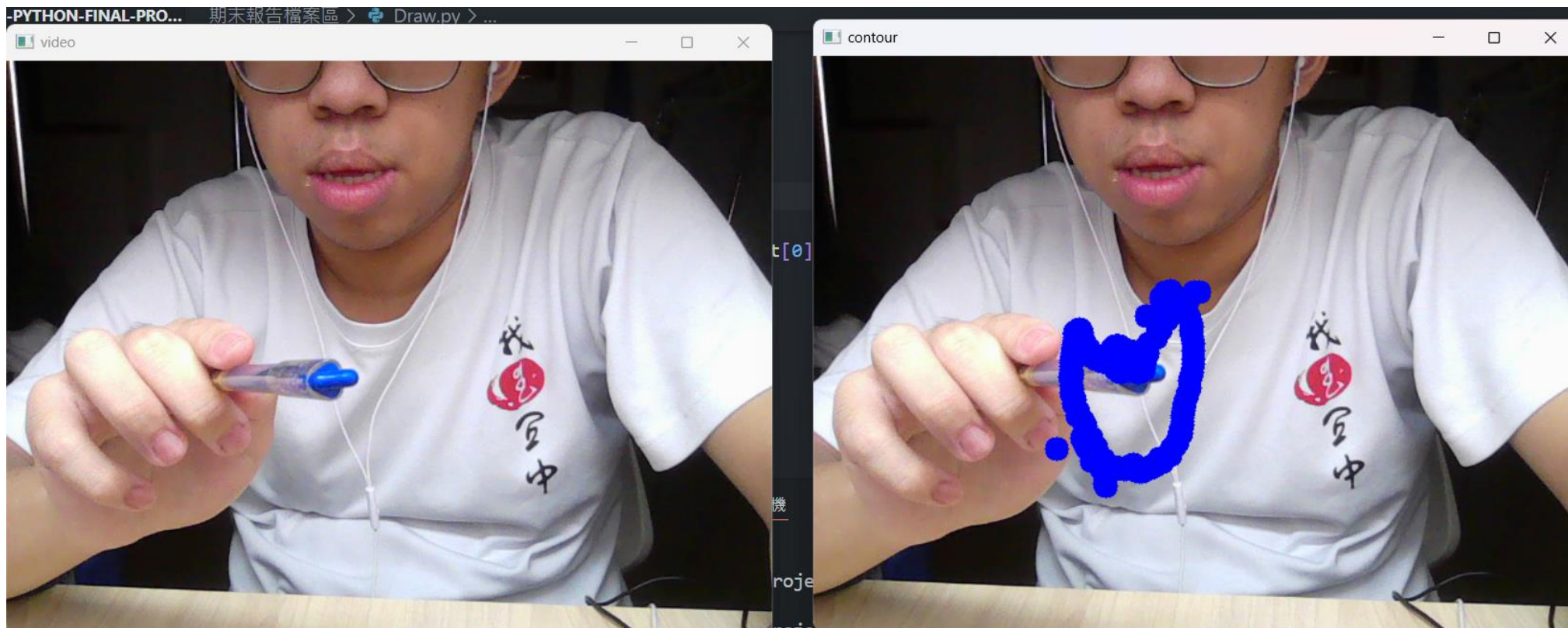
程式大致上分為兩個部分，其一為需要校正色筆的setting檔：

需要調整係數，以確認我們的筆頭的顏色。(這邊採用hex色碼)



圖九-左邊為鏡頭所拍攝的畫面，上方為調整係數的調桿，右邊則為經由程式運行後的圖片

## 成果展示(二)-實際運用



圖十-左邊為鏡頭所拍攝的畫面，右邊則為經由程式運行後的圖片  
根據我們所拿的筆，可以在視訊畫面上繪製出根據筆跡的圖案。

# 日後改善(自我評估)

問題	解決方案
筆刷會因為鏡頭中出現類似顏色的物體導致筆刷偏移，有無想法解決？	可以由顏色確定位置改成由點去跟隨筆刷的移動，不僅能穩定筆刷，也能跟隨筆跡。
是否能夠隨著我筆的不同，而有不同的顏色？	能將多樣顏色放入陣列之中，讓顏色的切換更為順暢，但大多數顏色還是需要經由手動確認。
能否多支援多支筆在同一畫面中作圖，已達成娛樂或教學效果？	可以新增函數來讓程式同時支援雙色筆或是多色筆，並新增畫面已確認筆機的位置。
給自己的作品打幾分？滿分十分	7/10分，有些地方還可以做得更好，也可以是這支援線上的視訊軟體來使用，現在整理的問題都是日後前進的目標。

表三-自我評估



# 學習心得

Opencv 在上述的機器視覺具有著一定的成分，除了是能利用在圖像辨識、影像處理以及近幾年來的人臉辨識都能做的到。不單單是因為它是 Python 延伸出來的模組庫之外，也因為它的便利性之外，有許多國內外的專家會將已經訓練且學習過的辨識模型丟到 Github 上面供人下載使用。而在課堂實際體會過後，Python 簡易的語法讓版面變的簡略之外，更多的是 Opencv 提供的語法更可以讓編撰者能夠有更簡易的編寫程式。



圖十一-人臉辨識的效果

# 參考資料

- **【python】opencv 2小時初學者教學** | 影像辨識 | 影像處理 | 人臉辨識 | 電腦視覺  
=>opencv的學習、程式主架構
- NCUEDU中央機房  
=>程式問題、與AI討論哪些可以改進
- [OpenCV]基礎教學筆記：影像讀取、前處理(with python)-001  
=>視訊畫面的處理、讀取圖片及影片

