作業1: Perspective-Distortion-Correction

班級: 碩專班

學生:5108056016張仲威

教授: 吳俊霖 教授

目錄

[一. 主要演算法 2](#_Toc59982619)

[二. 程式片斷 2](#_Toc59982620)

[三. 測試資料 4](#_Toc59982621)

[四. 結果呈現 4](#_Toc59982622)

[五. 討論與結論 5](#_Toc59982623)

# 主要演算法

主要一般我們在做透視變形校正，會希望將在圖像中變形的物體，如：大樓、門牌、石碑、佈告欄、公文等等的東西，把它們校正到"正視"該物體時的樣子

在數學上，這樣的變形我們可以用下列的聯立方程式來表示：

​x = ax' + by' + cx'y' + d

y = ex' + fy' + gx'y' + h

其中**x、y為原始影像的座標**，**x'、y'為校正後的座標**，

a、b、c、d、e、f、g、h為常數，表示變形關係。

而這個時候我們只要找出**變形區域的四個點**，再把這個四點**代入解聯立方程式**，即可得出 a~h 八個參數，也代表知道他們彼此之間的**變形關係**

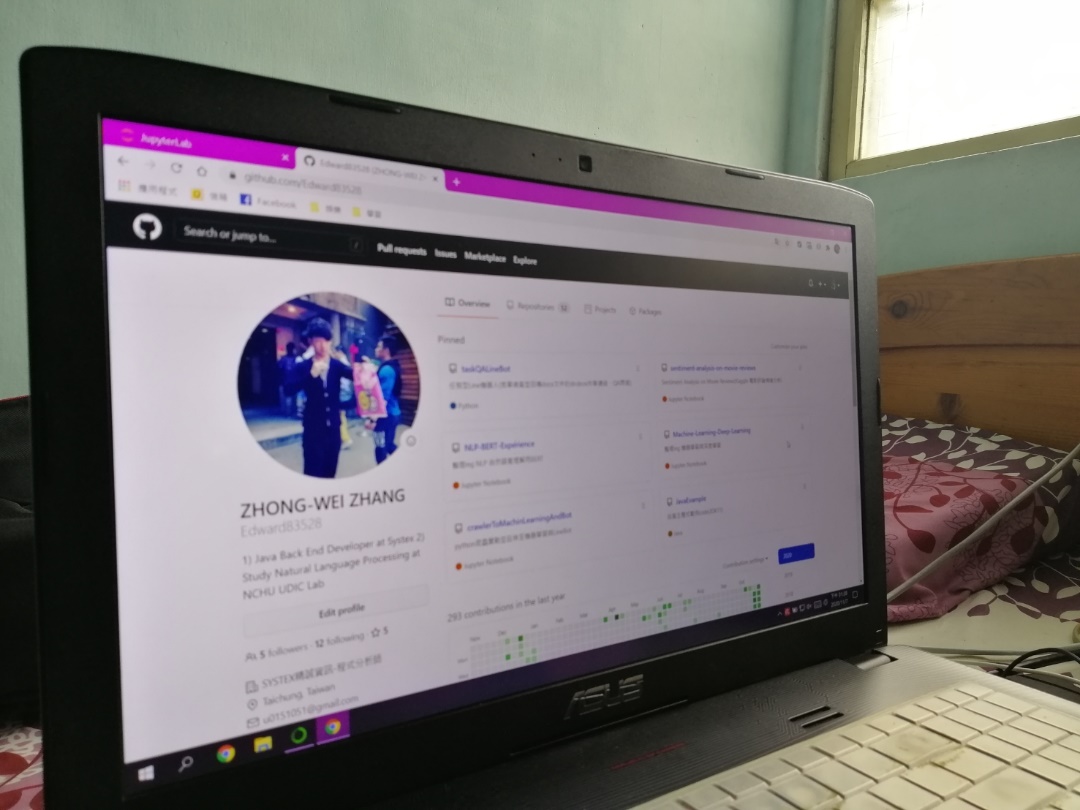
# 程式片斷

1. **def** perspectiveLogo(image, point1, point2):
2. h, w = image.shape[:2] # 獲取圖像的大小，shape返回的是一個tuple元組，第一個元素表示圖像的高度，第二個表示圖像的寬度，第三個表示像素的通道數。
3. **print**(h, w) # (1108,1477)取高度和寬度就好
4. #print(point2[2][0]) # 300.0
5. #print(point2[2][1]) # 300.0
6. img1 = np.zeros((int(point2[2][0]), int(point2[2][1]), 3), dtype=np.uint8) # 給定300\*300 長度3 shape形狀>用0填充數組
7. M = cv2.getPerspectiveTransform(point1, point2) # 計算得到轉換的矩陣
8. '''''
9. M
10. [[ 1.55112036e-01 -6.37446723e-03 -2.02176852e+01]
11. [-4.56013848e-02  3.22669109e-01 -4.60250059e+01]
12. [-4.08228948e-04  7.07633919e-06  1.00000000e+00]]
13. '''
14. # 自己做透視變換
15. **for** i **in** range(h):
16. **for** j **in** range(w):
17. x = (M[0][0]\*j + M[0][1]\*i + M[0][2]) / (M[2][0]\*j + M[2][1]\*i + M[2][2]) + 0.5
18. y = (M[1][0]\*j + M[1][1]\*i + M[1][2]) / (M[2][0]\*j + M[2][1]\*i + M[2][2]) + 0.5
19. x, y = int(x), int(y)
20. #print(x, y)
21. **if** 1 <= x < point2[2][0]-1 **and** 1 <= y < point2[2][1]-1:
22. img1[y, x, :] = image[i, j, :]
23. img1[y, x-1, :] = image[i, j, :]
24. img1[y, x+1, :] = image[i, j, :]
25. img1[y-1, x, :] = image[i, j, :]
26. img1[y+1, x, :] = image[i, j, :]
27. img2 = cv2.warpPerspective(image, M, (300, 300)) # 利用CV2透視變換成是相對於原圖image是以M變換後的圖像
29. img = np.hstack((img1, img2)) # 水平(按列順序)把陣列給堆疊起來(兩者可以做比較)
30. show(img)
31. cv2.imwrite('output.jpg', img) # 寫入圖檔

**主函式我自行標了4個點**

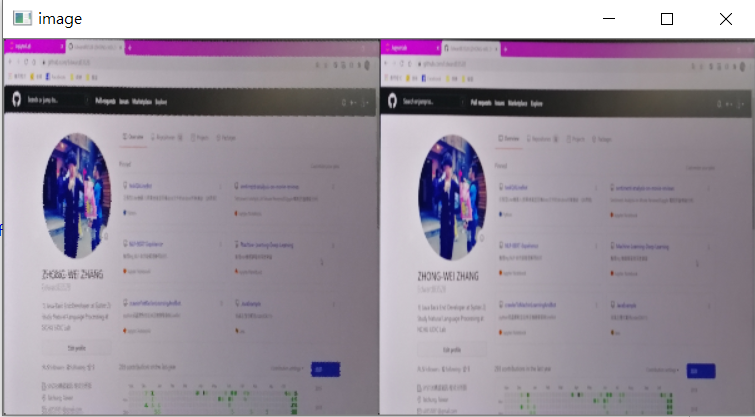
1. **def** main():
2. image = cv2.imread(path) # 加載圖像
3. img = image.copy() # 複製圖片
5. point1 = np.float32([[137, 162], [1163, 307], [1178, 797], [173, 1038]]) # 標出四個點
6. #print(point1)
7. point2 = np.float32([[0, 0], [300, 0], [300, 300], [0, 300]]) # 輸出的四個點
8. #print(point2)
9. perspectiveLogo(image, point1, point2)
11. img = draw\_line(img, point1)
12. show(img) # show 出我標的點的輪廓圖

# 測試資料



# 結果呈現

左邊是利用CV2透視變換的結果,**右邊**是**自行用解聯立方程式的方法**自行實現,兩者做比較



# 討論與結論

已我們實驗結果的圖,我們自行實現透視變化效果也是跟cv2效果相當,也就代表我們的邏輯方法正確