影像處理 HW1

透視變形校正 Perspective Distortion Correction

資工碩一 7111056426 蘇亭云

STEP 1、讀取圖片

```
# 讀圖 opencv 讀進為 BGR 需轉換 成 RBG
in_bgr = cv2.imread('notebook.jpg',cv2.IMREAD_COLOR)
# 轉 BGR >> RGB
in_rgb = cv2.cvtColor(in_bgr,cv2.COLOR_BGR2RGB)
print(in_rgb.shape)
```

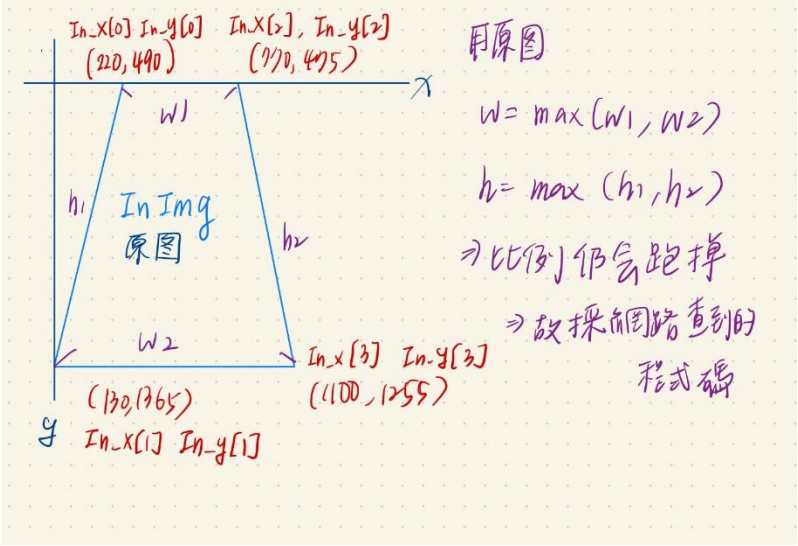
STEP 2、用小畫家找 4 點像素座標

```
# 用小畫家找4點像素座標
in_x=[220, 770, 130, 1100]
in_y=[490, 475, 1365, 1255]
```

STEP 3、計算 out_img： width and height

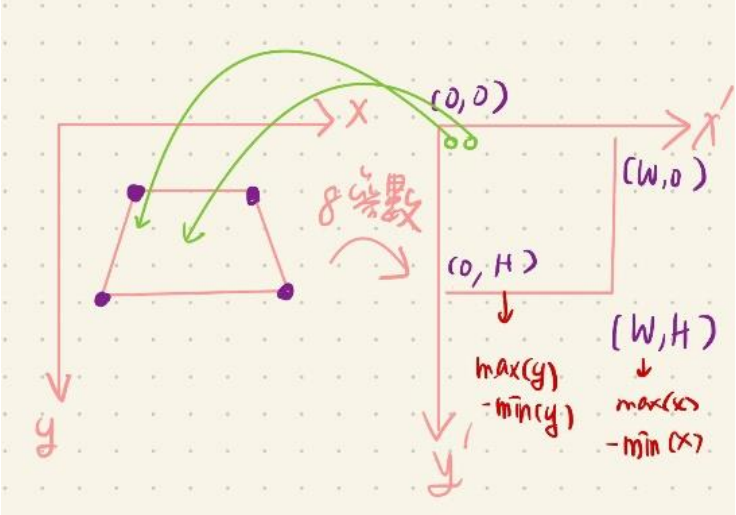
直接用歐式距離取大值或平均，比例仍易變形，故套網路上程式

```
w1=dis.euclidean([in_x[0],in_y[0]],[in_x[1],in_y[1]])
w2=dis.euclidean([in_x[2],in_y[2]],[in_x[3],in_y[3]])
w=int(np.around(max(w1,w2)))

h1=dis.euclidean([in_x[2],in_y[2]],[in_x[0],in_y[0]])
h2=dis.euclidean([in_x[3],in_y[3]],[in_x[1],in_y[1]])
h=int(np.around(max(h1,h2)))
```



```
# 計算新圖正確 width and height
w,h=realAspect_W_H(in_rgb, in_x, in_y)
print("w",w,"h",h)
```

依算出的 width and height，得到原圖 in_img 4 點對應到 out_img 的 4 點對應座標。



STEP4、解 AX=b

A 為 out_img 的 4 點座標代入得 8 個方程式，

b 為 in_img 的 4 點座標，

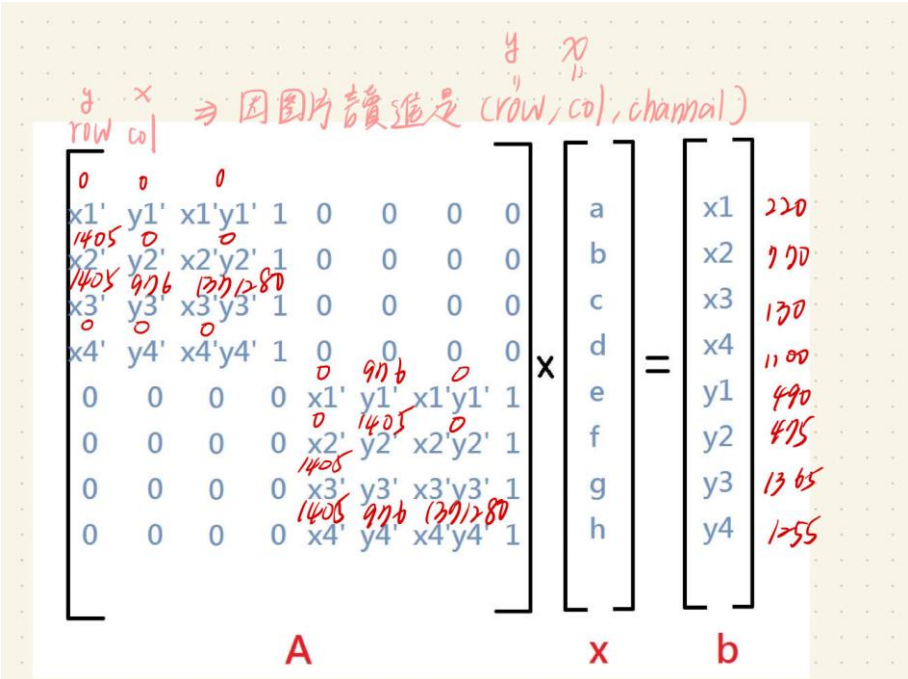
利用 python 程式庫 解出 a ~h 8 個參數 X = solve(A,b)。

因圖片是(row,col) 即 (y,x) 故先 y 後 x 去做計算

```
# 圖是(row,col) 即 (y,x) 故先 y 後 x 去做計算
A=np.array([
    [out_y[0] , out_x[0], out_x[0]*out_y[0], 1, 0, 0, 0, 0],
    [out_y[1] , out_x[1], out_x[1]*out_y[1], 1, 0, 0, 0, 0],
    [out_y[2] , out_x[2], out_x[2]*out_y[2], 1, 0, 0, 0, 0],
    [out_y[3] , out_x[3], out_x[3]*out_y[3], 1, 0, 0, 0, 0],
    [0, 0, 0, 0, out_y[0] , out_x[0], out_x[0]*out_y[0], 1],
    [0, 0, 0, 0, out_y[1] , out_x[1], out_x[1]*out_y[1], 1],
    [0, 0, 0, 0, out_y[2] , out_x[2], out_x[2]*out_y[2], 1],
    [0, 0, 0, 0, out_y[3] , out_x[3], out_x[3]*out_y[3], 1],
])
```

```
A [[      0       0       0       1       0       0       0       0]
 [      0     976       0       1       0       0       0       0]
 [   1405       0       0       1       0       0       0       0]
 [   1405     976 1371280       1       0       0       0       0]
 [      0       0       0       0       0       0       0       1]
 [      0       0       0       0       0     976       0       1]
 [      0       0       0       0    1405       0       0       1]
 [      0       0       0       0    1405     976 1371280       1]]
```

```
b=np.array([in_x[0], in_x[1], in_x[2], in_x[3], in_y[0], in_y[1], in_y[2], in_y[3]])
X = solve(A, b)
```



得 X (參數 a~h)

```
x[0]=-0.06405693950177936
x[1]=0.5635245901639344
x[2]=0.0003062831806779068
x[3]=220.0
x[4]=0.6227758007117438
x[5]=-0.015368852459016393
x[6]=-6.92783384866694e-05
x[7]=490.0
```

STEP5、Inverse mapping 算出轉置後的圖於原圖的位置並進行雙線性插值

```
# 做 inverse mapping + bilinear interpolation 雙線性插值
output_img=invert_mapping(X, in_rgb, out_rgb)
```

STEP5-1、轉置後的圖於原圖的位置

將 A 的四個點帶入 Pseudo Affine 公式

```
for i in range(rows):
    for j in range(cols):
        # 新座標點
        double_x = (X[0] * i) + (X[1] * j) + (X[2] * i * j) + X[3]
        double_y = (X[4] * i) + (X[5] * j) + (X[6] * i * j) + X[7]

        # 距離新座標點最近的整數點
        x = int(double_x)
        y = int(double_y)
```

X=row*x[0]+col*x[1]+row*col*x[2]+x[3]

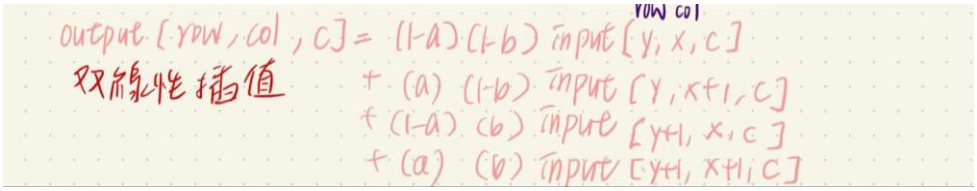
即 新 X=原圖 y*a+原圖 x*b+原圖 x*y*c+d

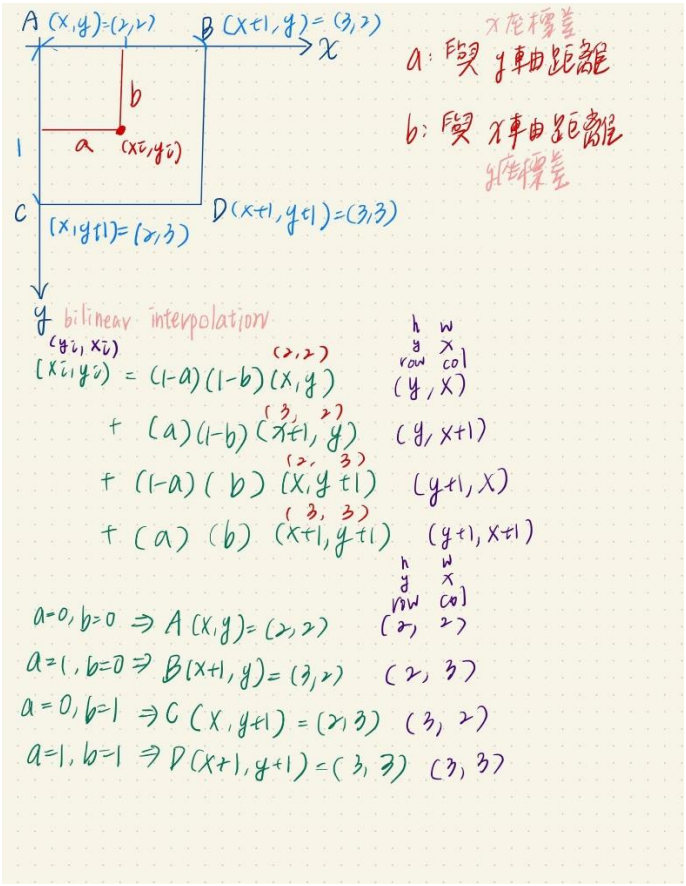
Y=row*x[4]+col*x[5]+row*col*x[6]+x[7]

即 新 Y=原圖 y*e+原圖 x*f+原圖 x*y*g+h

STEP5-2、進行雙線性插值

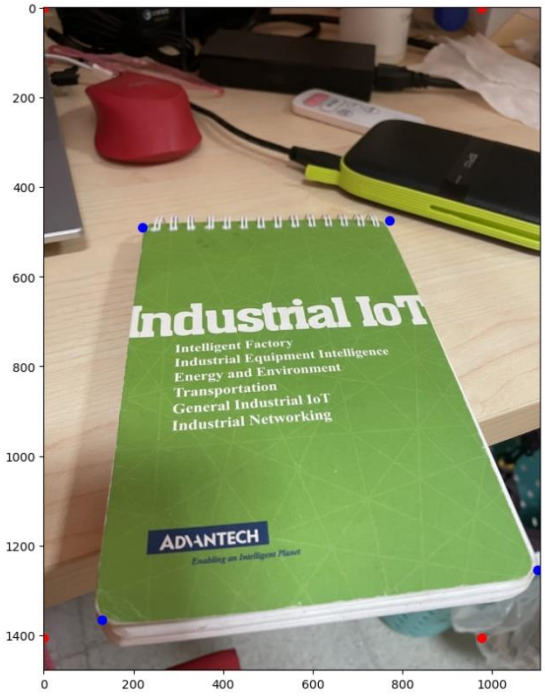
```
# a:y軸距離(x座標差) b:x軸距離(y座標差)
a = (double_x - x)
b = (double_y - y)

# 做 bilinear interpolation 雙線性插值
# 圖是(row,col)即 (y,x)
for c in range(3):
    output_img[i, j, c] = (1 - a) * (1 - b) * input_img[y, x, c] + \
                          a * (1 - b) * input_img[y, x+1, c] + \
                          (1 - a) * b * input_img[y+1, x, c] + \
                          a * b * input_img[y + 1, x + 1, c]
```





STEP6、結果：

原圖



校正後

透視校正後圖