HW1\_透視變形校正 Perspective Distortion Correction

**一、透視變形**

透視變形是由相機與目標物體之間的距離與拍攝角度所造成，當相機與目標物體之間的距離有所變化時，會造成影像發生彎曲或變形，造成結果與所期待的相左，就是所謂的透視變形。

**二、如何校正**

在數學中，透視變形能透過以下聯立方程式表示：

其中為原始影像的座標，為校正後的座標，a、b、c、d、e、f、g、h為常數，表示變形的關係。

透過影像變形區域的四個角點代入聯立方程式，就可以計算出a、h共八個參數並得到變形關係。

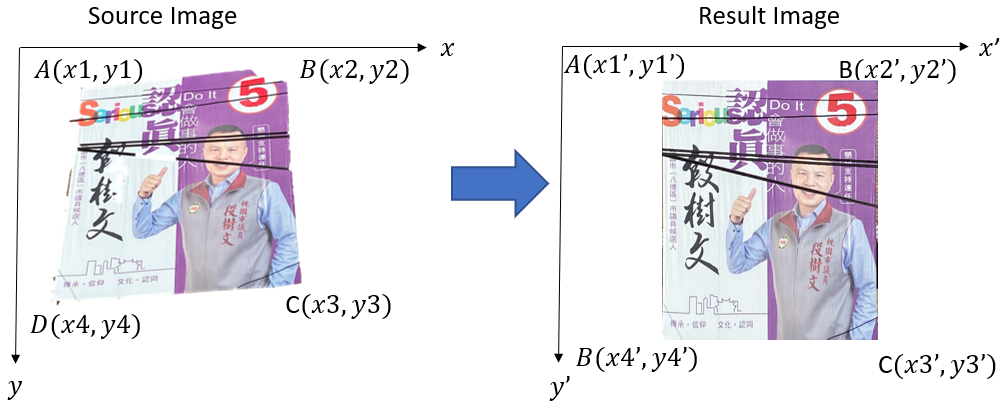


圖 一 座標點關係圖

**三、如何計算**

可以透過解多元一次方程組求出八參數，八個方程式八個未知數，必定有解。

可以將八參數寫成以下矩陣

計算出a、b、c、d、e、f、g、h後，就能套用Invert mapping +bilinear Interpolation將變形區的每一個點校正過去新的image。

Invert mapping + bilinear Interpolation：

**四、程式實作**

Step1：讀取圖片

首先設定圖片的路徑，透過search\_file函數將路徑內所有的圖片讀入，將圖片檔案名稱存進image陣列內



圖 二 讀取圖片程式碼

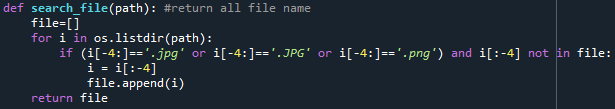


圖 三 搜尋檔案函數內容

此時陣列內容與資料夾內容顯示如下：



圖 四 資料夾圖片名稱

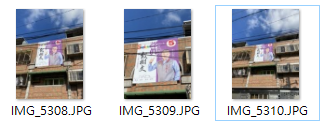


圖 五 資料夾內的圖片

就能使用迴圈將每張圖片讀入並做處理。

Step2：圖片前處理

進入迴圈後，使用者可以先設定預設的圖片大小，如本程式預設期望大小為1200\*1000，再使用cv2.imread讀取圖片img，並使用cv2.resize將圖片縮小成原本的3成，供使用者方便點選四個角點，否則圖片過大會超出螢幕範圍造成不方便點選。

再來會將img使用copy的方式給予img2，以免修改的時候動到原圖(img)，再使用cv2.imshow將img2展示出來，並將其與show\_xy涵式傳入setMouseCallback，供使用者可以點擊圖片的四個角點。

show\_xy函數會先判斷使用者是否點擊(if event ==1)，若使用者點擊，則會print出該點的x,y座標，並將x,y座標加入進point陣列內，代表此座標點為四角點之一，在點擊的同時會呼叫cv2.circle將使用者所點擊的點畫上紅圈，供使用者查看，當使用者點滿四個點，就將img關掉，並開始計算八參數。

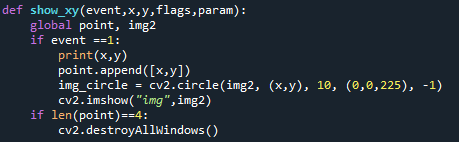


圖 六 show\_xy函數



圖 七 圖片點擊後顯示紅點示意圖

Step3：計算八參數

當使用者點選完四個角點後，會將 1.圖片 2.期望大小 3.四個角點 傳入calculate函數中，此函數中會先判斷四個角點的座標值，並將四個角點重新排序為[左上、右上、右下、左下]之順序，以免產出的效果不如預期，排序原理如下圖：

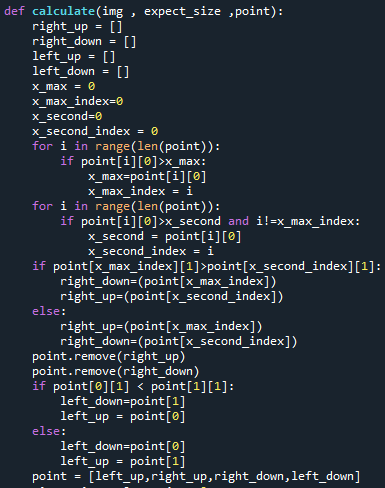


圖 八 防呆機制

程式首先判斷x的最大值與其index，再來判斷第二大的x值與其index，藉此可以判斷這兩個座標點為右邊，再來判斷這兩個座標點的y值，y值較大的為右下角，反之則為右上角。

之後將這兩個已經確定為右上及右下角的座標點從陣列point中移除，此時point陣列剩下兩個座標點，我們只需要再比較y值，若y值較大的則為左下，反之為左上，當左上、右上、右下、左下皆確定後，再將這四個點依序放進point陣列中，以此達到防呆的效果，因此使用者點選的順序不會造成程式出現錯誤。

待四個座標點順序確定後，即可開始計算八參數，首先我建立代表我們所預期的圖片的四個座標(x1\_,y1\_)、(x2\_,y2\_)、(x3\_,y3\_)、(x4\_,y4\_)

並由程式自動依據使用者所期望的圖片大小建立前述的八參數矩陣(此範例A矩陣為期望大小1200\*1000所建立出的矩陣，b矩陣為使用者所點選的四個角點)：

有了A矩陣與b矩陣確切的值後，即可透過np.linalg.solve套件替我們計算出x的值，此時x的值為

即可將x值回傳至主程式，程式碼如下：

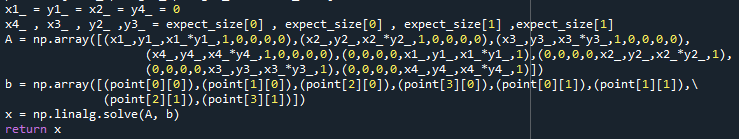


圖 九 計算八參數

Step4：Invert mapping + bilinear Interpolation

前項步驟所算的x回傳至主程式後，會建立一個全為0的圖片output\_img，大小為我們所期望的大小，也就是1200\*1000，建立後會將我們所算出來的x、原圖img及output\_img傳入inverse\_mapping函數，此函數會透過兩個迴圈將output\_img的長寬跑一輪，並且將每個output\_img的座標點依據以下公式將原圖img的座標點像素值傳入output\_img相對應的座標點

待output\_img全部的座標點都有像素值後，會使用cv2.imshow將透視變形校正後的圖片show出來給使用者查看。

程式碼如下：

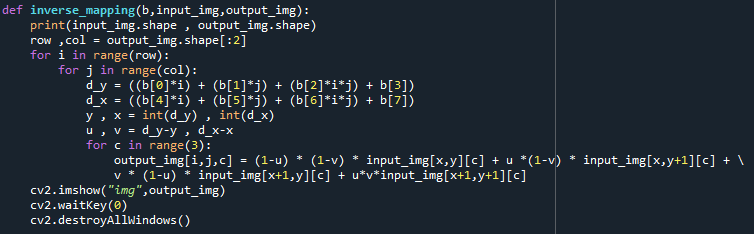
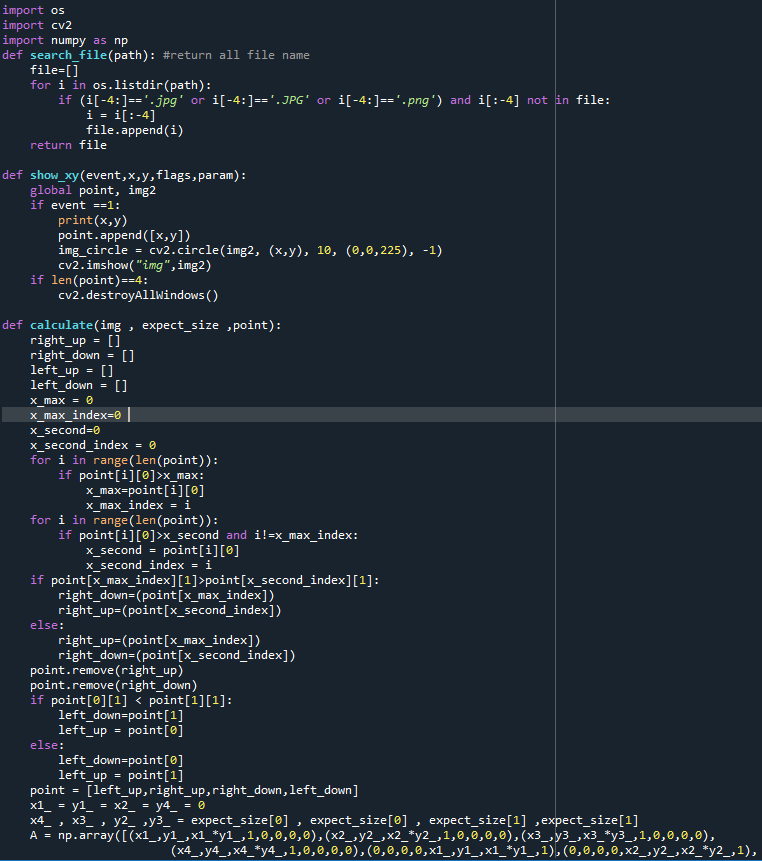


圖 十 inverse\_mapping函數內容

最終結果如下圖：

|  |  |
| --- | --- |
| 圖 十一 原始圖片點選四個角點 | 圖 十二 透視變形校正後結果 |

整體程式碼如下：



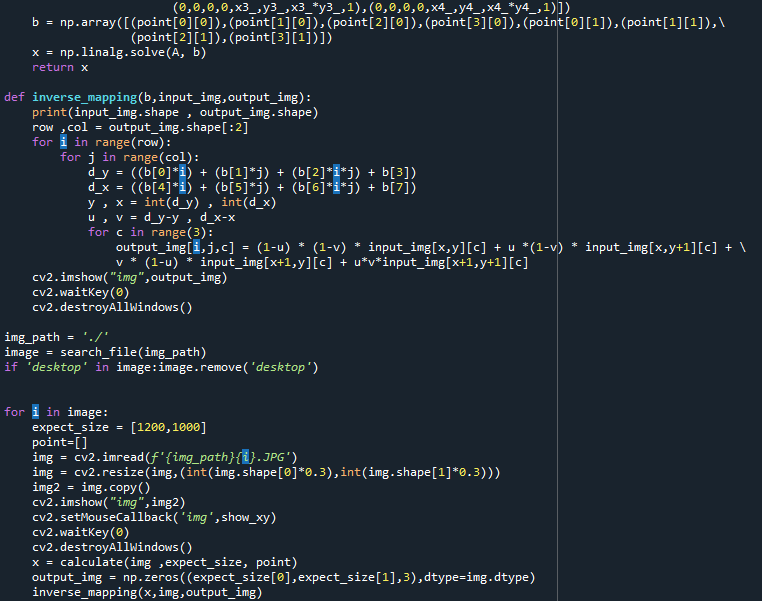


圖 十三 整體程式碼

心得：

我本身做圖片的處理通常都是直接呼叫opencv套件，第一次自己將完整影像處理的code寫完，蠻有成就感的，其中我遇到最大的困難是opencv與numpy的座標系統是相反的，導致我在寫程式時出現了些問題，但好在後來debug過後結果已經是正確的了，相信經過這次作業對於我未來在影像處理實作上會有很大的幫助。