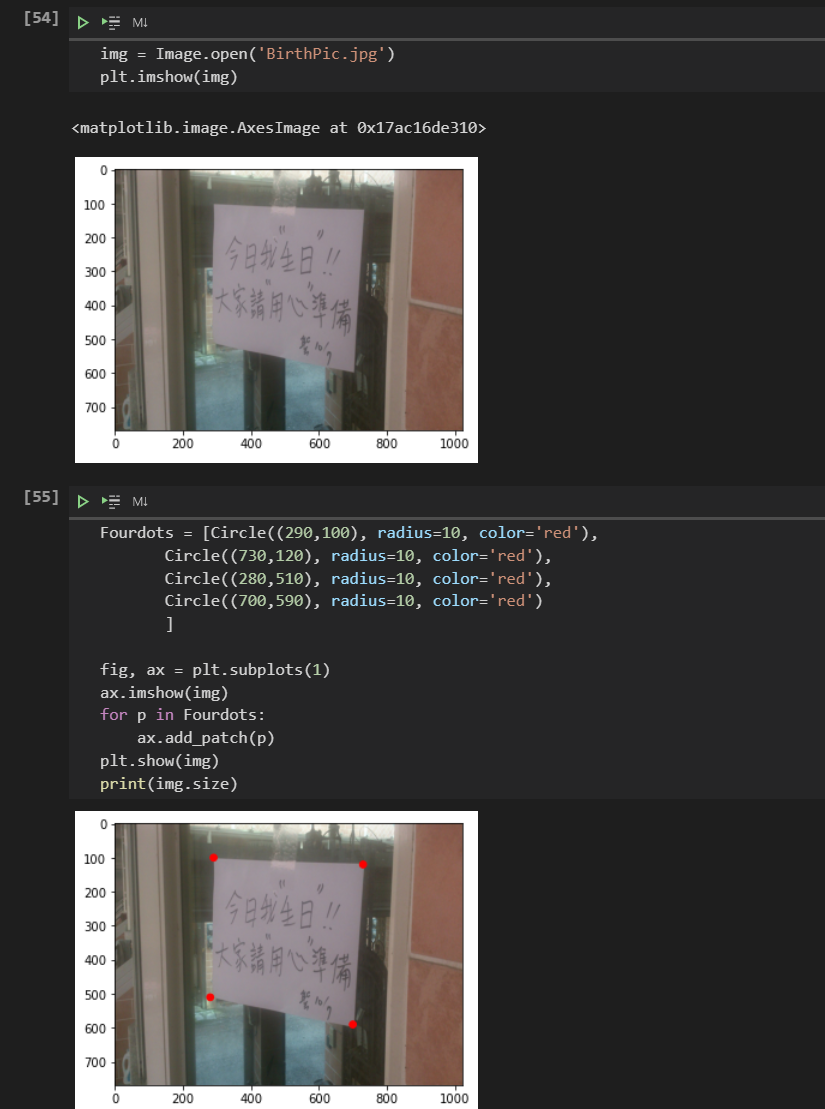
5109050016-鄭亦棋 影像辨識 作業一

作業環境 python

首先import 基本的參數庫後

匯入影像 大概抓出四個點

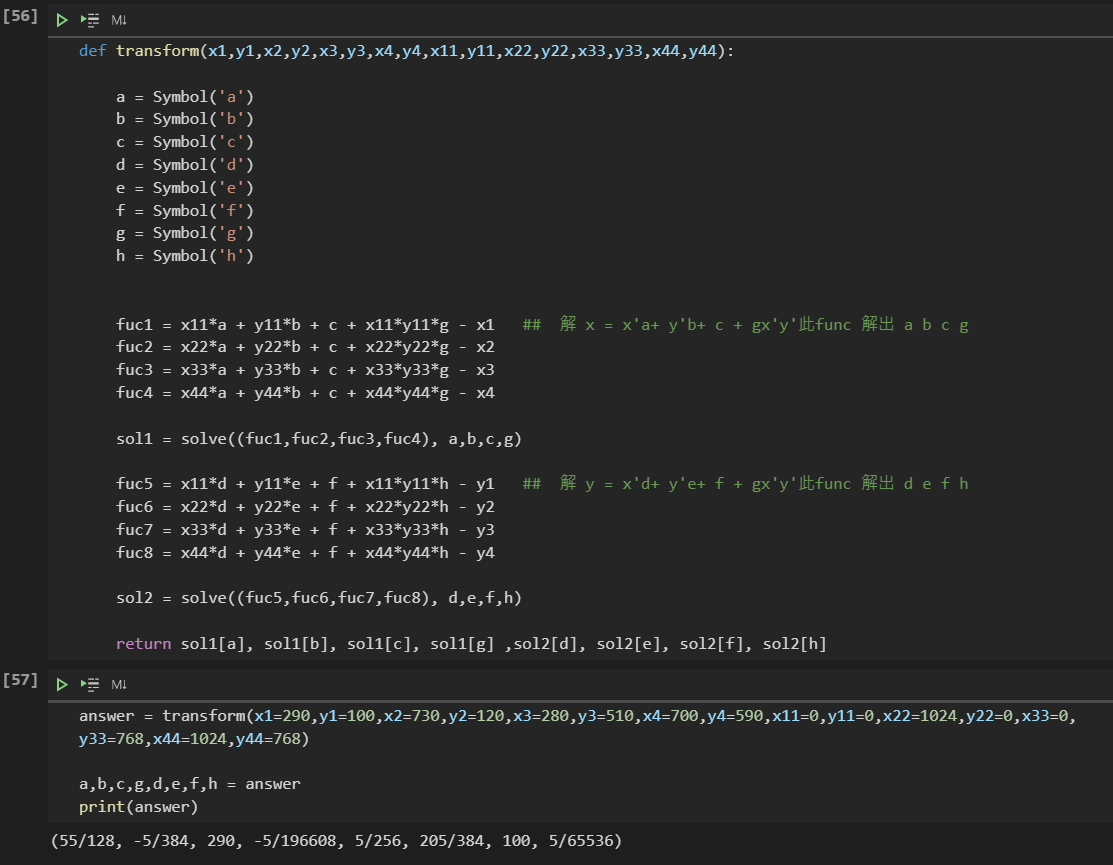
然候使用紅點來顯示要做的四個點的位置，一邊顯示一邊校正，直到抓出正確四個點的座標位置。



利用老師上課教的參數內容

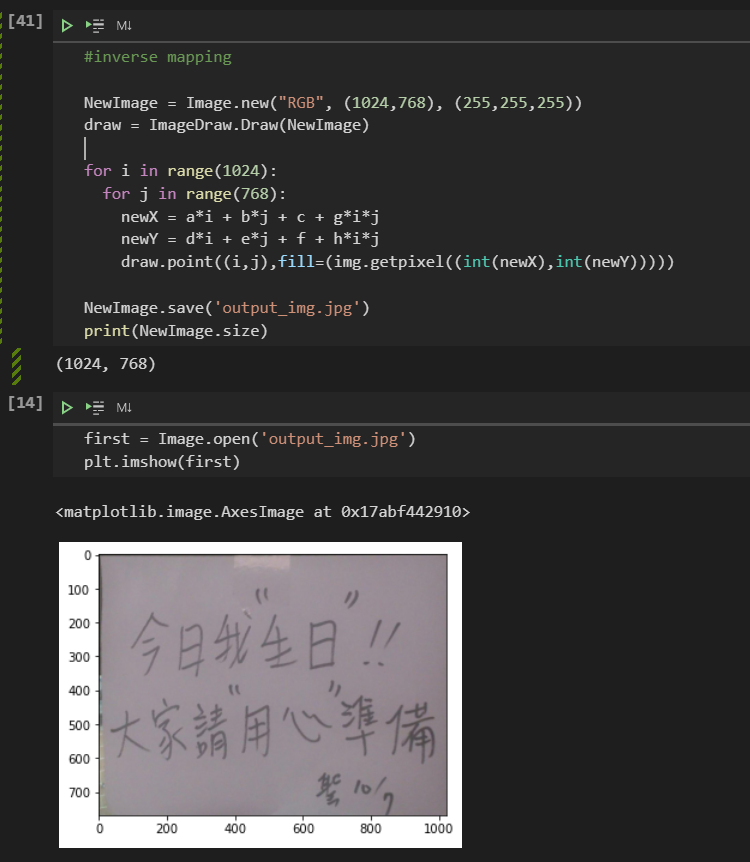
8個點可以解8個方程式 (這邊我用外部套件去解)

最後面把它的解print出來 確定值是正常的



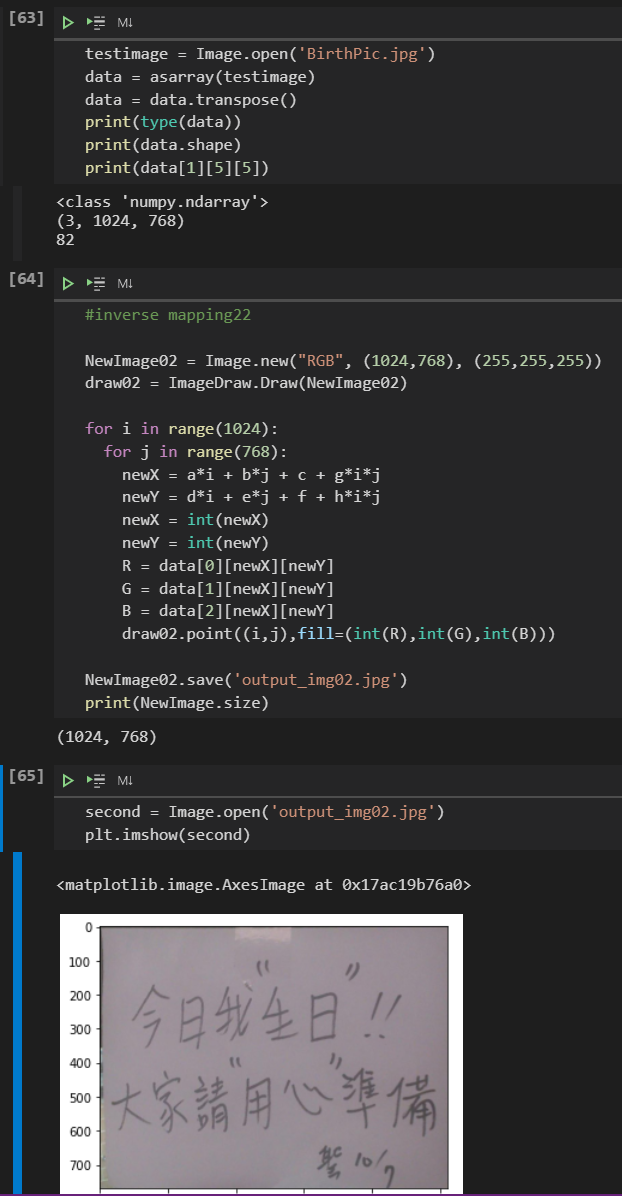
接著預備做 inverse mapping (先不用bilinear去取點 而是先用int抓像素位置)

這邊我做了兩次，一開始我用img.getpixle的方式做，有做出來



但是img.getpixle 莫名的速度很慢，我想說是不是因為呼叫函式所以比較慢，

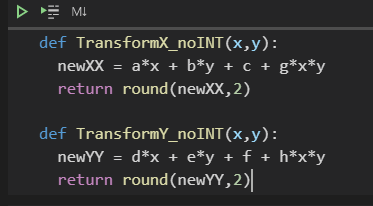
所以又另外寫了用np array的方式，內容都一樣。(只差在 要分RGB 然後轉一下矩陣的順序)



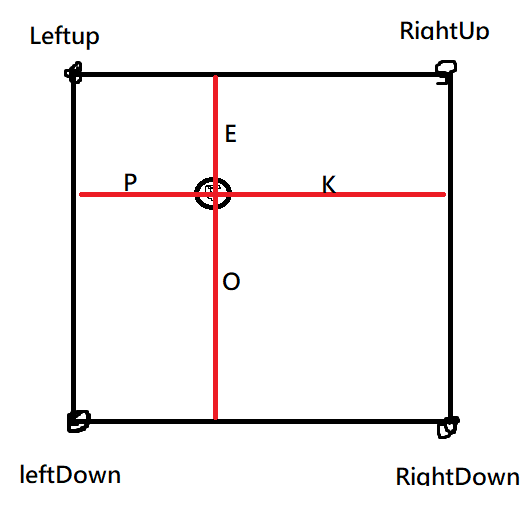
確定上面可行了，接下來就是把原本用int抓像素的這段，加上bilinear去得到最終pixel

因為這邊失敗了很多次，所以我後來先把transform的X跟Y都寫成函式，確認不是這段有問題

後來又因為算很慢，我想說是不是因為小數點太多，所以我直接取round倒第2位



接著寫bilinear 四邊的角是要取像素的最近四點 peko分別是來四邊的距離(0~1)

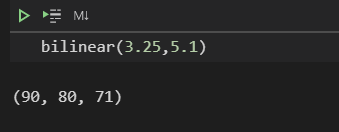


按照上面的概念寫出以下方法



然後這邊好像就一定要分RGB來取值了，如果直接 getpixel不管怎麼做，結果好像都會是全白的。

隨便抓一個點，確定bilinear沒有問題，可以有正常的output



最終正式跑圖，用bilinear來做inverse mapping的取點。

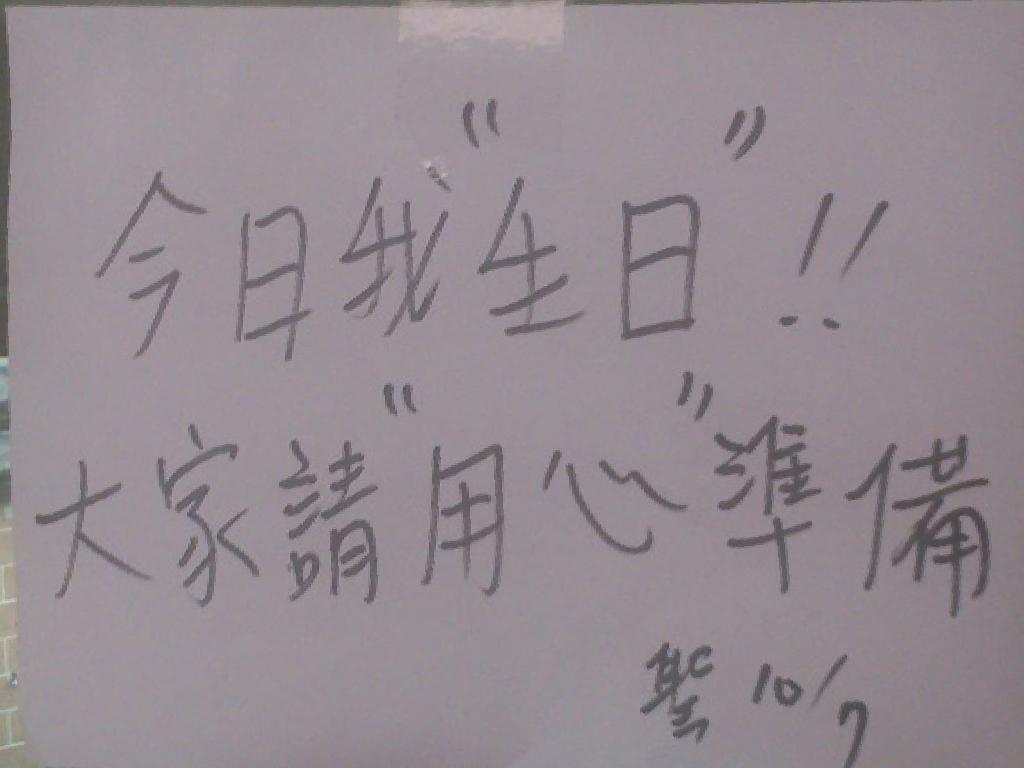
方法就是用上面第一次的，只是把int這段換成bilinear

(我跑超級久，最後還是不確定為什麼可以這麼慢。我跑了30分鐘)

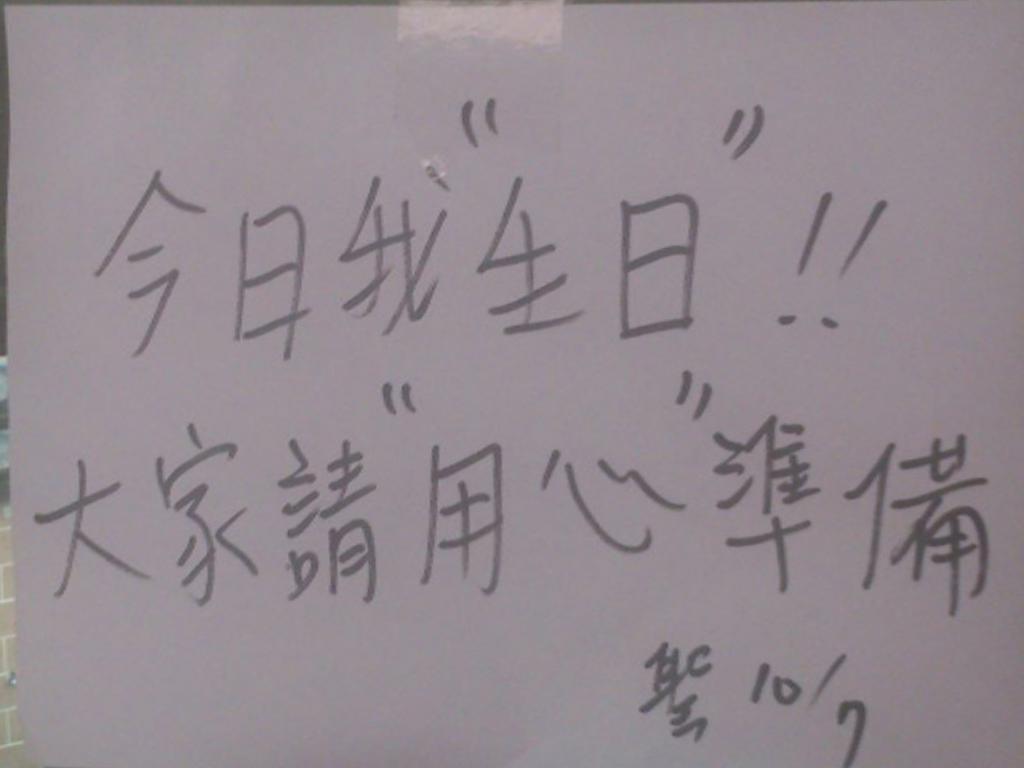


最終成果比較

取int的↓

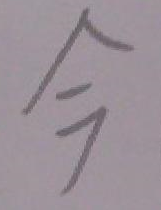


用bilinear的↓

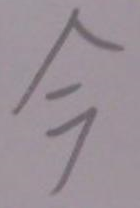


放大

取int的↓



用bilinear的↓



結論

bilinear 好棒棒 比較不會有鋸齒。