三維電腦視覺與深度學習應用 作業一

姓名: 葉冠宏 學號:R11943113 系級:台大電子所一年級

1.執行環境

程式語言:Python

Package: numpy, cv2

如何執行:

第一題:把 1.py 和 1-0.png, 1-1.png, 1-2.png, correspondence_01.npy, correspondence_02.npy 放置於同一個資料夾之後，移動到該資料夾。並執行 python 1.py。

第二題:把 2.py 和 withPoint.png 放置於同一個資料夾後，移動到該資料夾。並執行 python 2.py。

2. Problem 1: Homography Estimation

在 get correspondence 這個步驟中，我是採用了 SIFT 演算法及助教 github 給的範例程式來做實作。然而在參數調整上我在 cv.SIFT_create() 的 function sigma 是用 1.8。而在 ratio test 上我的 ratio 則是用 0.71。對於所採用的 match points，我只採用了最佳解的 distance 和第二佳解的比值是小於 0.71 的配對組合。

homography 的矩陣我則是採用了老師上課講義所提供的公式，並用 svd 的方式取得解。

以下為各個不同的 k 和 image 組合時，採用 SIFT 演算法所得出的配對結果:

Image A to Image B(k=4):

Image A to Image B(k=8):

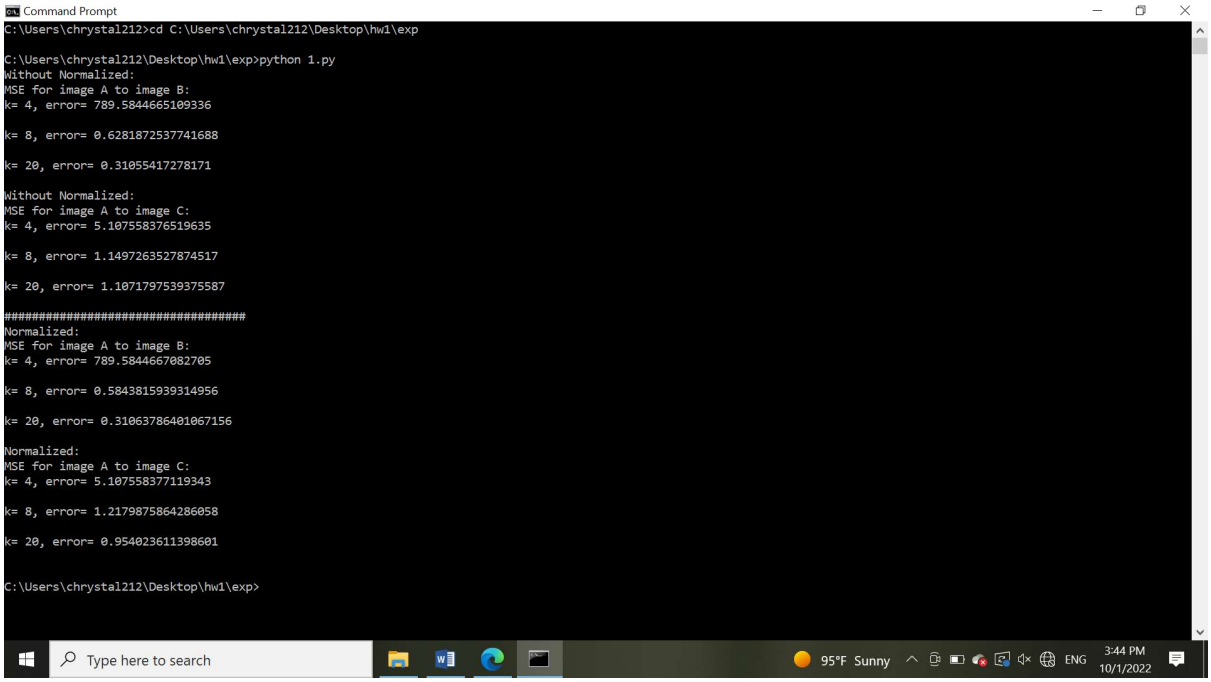
Image A to Image B(k=20):

Image A to Image C(k=4):

Image A to Image C(k=8):

Image A to Image C(k=20):

以下為我針對 image A 到 image B ，和 image A 到 image C 做實驗所得到的實驗結果。其中，我也做了如果對 image 做了 normalize 的話，其所得到的結果是如何。以下為我所得到實驗結果的截圖:



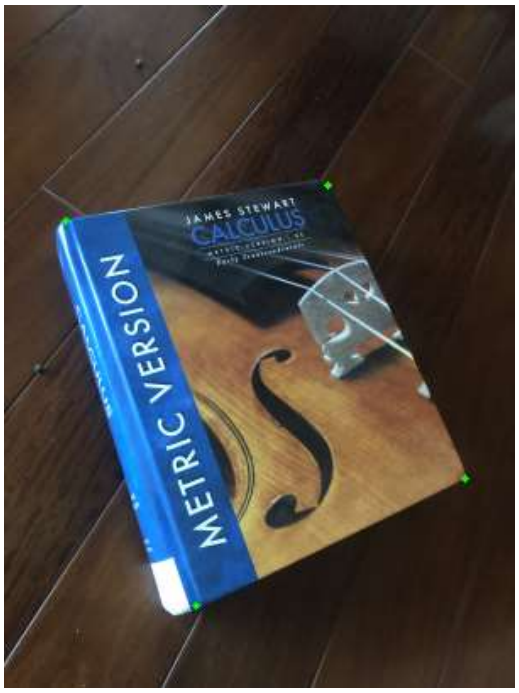
我們可以看到的是，在沒有 normalize 的狀態下，image A to image B 當 k=20 個點時候，error 平均來說可以只有 0.31055417278171。然而當 k 下降至 8 時，error 是 0.6281872537741688，而當 k=4 時，error 則達到 789.5844665109336，效果非常的不好。但這也符合實驗的預期，所採樣的點越少，所求出的解越不精準。而在 image A 到 image C 的實驗中，我們也可以看到類似的實驗預期。當 k=20 時，效果很好，平均來說 error 只有 1.1071797539375587。而當 k 下降至 8 的時候，error 來到 1.1497263527874517，最差的 k=4 error 則有 5.107558376519635。

對於已經 normalize 的點來說，我們可以看到效果有比較好，原因是因為可能有些座標系因為某些數學矩陣關係會造成一些偏頗的現象，當我們先執行 normalize 的動作後，可以去除一些座標系扭曲的其他因素。我們可以觀察到，在 image A 到 image B 中，k=8 的時候，error 有從 0.6281872537741688 下降到了 0.5843815939314956。而在 image A 到 image C 中，k=20 時 error 從 1.1071797539375587 下降到了 0.954023611398601。其他情況則是變化不大。

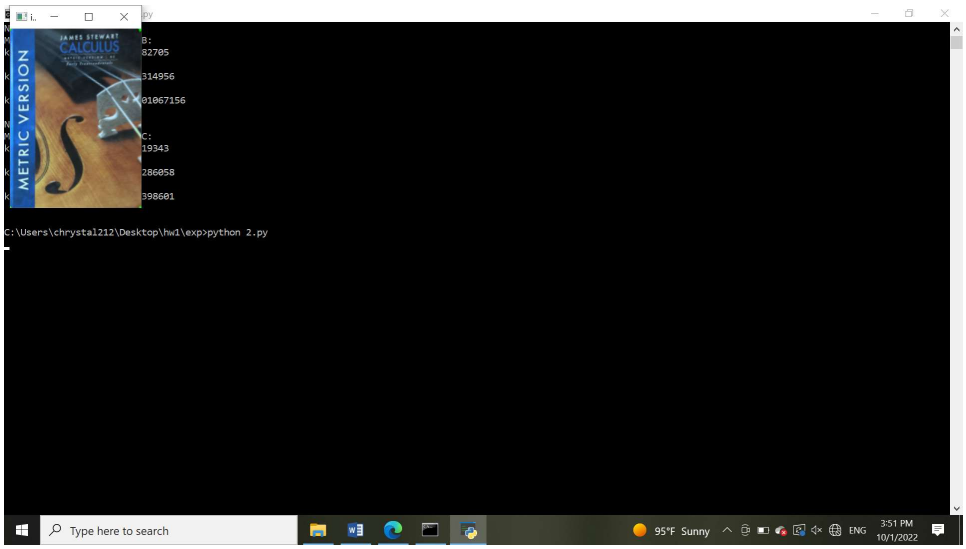
在過程中我遇到最大的挑戰是 SIFT 參數值的選取和 ratio test 中 ratio 的訂定。根據 open cv 的官方網頁，SIFT 的參數有 nfeatures, nOctaveLayers, contrastThreshold, edgeThreshold 和 sigma 等。經過各種排列組合的實驗後，我發現 nfeatures 維持在預設 0, nOctaveLayers 維持在預設 3, contrastThreshold 維持在預設 0.04, edgeThreshold 維持在預設 10, 而 sigma 則經過各種數值的嘗試後設定在 1.8 效果會最好。同樣的 ratio test 中 ratio 的訂定也是經過各種排列組合後決定設定在 0.71。我發現在參數的設定上有許多 local maximum，即兩個還不錯的參數設定之間可能隔了許多的值，而中間的值都造成了不好的效果。

3. Problem 2: Document Rectification

在做 problem 2 的時候，我先用 mouse_click_example.py 對 original.JPG 的書依序從左至右，從上到下畫了四個角點。取完四個邊角點後，校正實驗前的 image 如下:



實驗校正後的 image 如下:



取得一開始書的四個角點的座標後，我用歐式距離去計算書的寬和高是多少，並求得投影之後正面的書的四個角點的座標應該是如何。接下來我去計算投影的 homography 矩陣是什麼及其 inverse 矩陣。接著我就可以從預計呈現出的視窗的 image 的 pixel 座標，採用 backward warping 的方式去計算反推回來後的座標及周圍的 pixel 位置及強度是如何，加權算出投影後 image pixel 的強度。最後結果所呈現出的 warping efficiency 可以看到如上圖所呈現，達到了不錯的效果。