電腦視覺與應用
Computer Vision and Applications

Midterm Project 1

指導教授: 林宗翰 教授

課程學生: M10907305 陳俊億

設計方法與結果:

首先一開始我先對於兩張照片進行 SIFT 特徵匹配，之所以使用 SIFT 去尋找特徵點，特徵匹配的效果如 Fig1.、Fig2.與 Fig3.所示。得到特徵點其實不盡然都是正確的，且必須做篩選，因為 SIFT 再找特徵點相似度是透過歐式距離去做選取，所以越接近相似度會越對應點也較為精準，得出來的特徵點，全部代入 opencv 之中 findHomography，且其中使用到的演算法為 RANSC，因為代入的特徵點比較多，使用 RANSC 可以隨機抽樣，比較有效的減少錯誤。

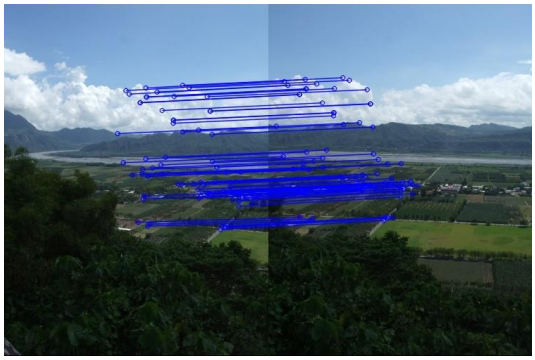

Fig1. SIFT 匹配(第一張&第二張)

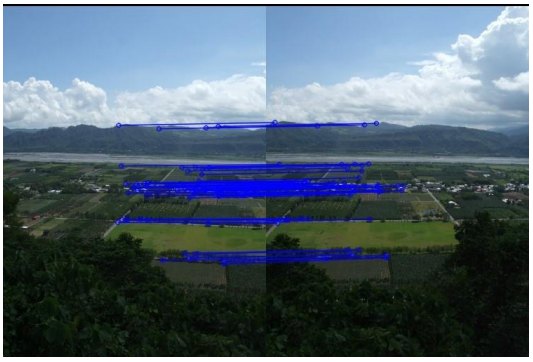

Fig2. SIFT 匹配(第二張&第三張)

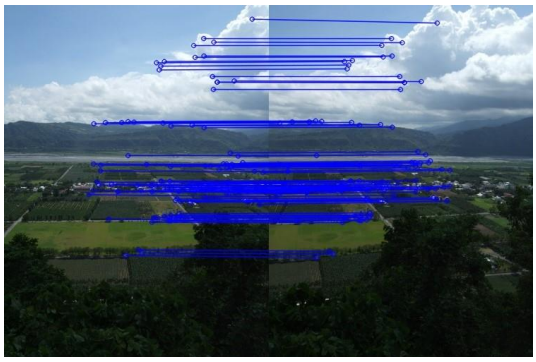

Fig3. SIFT 匹配(第三張&第四張)

因為使用 SIFT 對應點預期的多，無法全部放在 word 上，所以只有部分對應點放在 word 顯示，如 Fig5.所示，而其他的對應點放在 coordinate _values .xlsx 裡。

1:image_left	1:image_right	2:image_left	2:image_right	3:image_left	3:image_right
[[[26.302 932.1706]]	[[[324.32132 949.779]]	[[[13.136763 1118.3807]]	[[[308.09146 1115.0109]]	[[[7.310509 1009.9193]]	[[[227.94395 1004.9571]]
[[46.55724 1015.1174]]	[[340.91577 1030.1102]]	[[8.817491 985.1729]]	[[309.3131 987.1132]]	[[8.05047 1015.5747]]	[[228.56947 1010.5551]]
[[50.0117 1031.906]]	[[343.51746 1046.4879]]	[[24.095469 1117.727]]	[[318.99255 1114.859]]	[[18.482319 1010.728]]	[[238.57082 1006.08563]]
[[56.46478 919.3138]]	[[352.82043 937.7779]]	[[16.611193 831.0221]]	[[321.44376 839.3956]]	[[20.895714 1042.872]]	[[239.95609 1037.0719]]
[[62.649868 1098.5952]]	[[353.35284 1111.5131]]	[[45.18925 1109.0874]]	[[338.98987 1107.6464]]	[[21.966848 1026.7812]]	[[241.43433 1021.6556]]
[[54.64001 730.912]]	[[354.6711 755.8935]]	[[42.131943 1012.9132]]	[[339.6629 1014.7607]]	[[21.966848 1026.7812]]	[[241.43433 1021.6556]]
[[62.81623 1018.55743]]	[[356.12875 1033.8618]]	[[39.85362 944.66516]]	[[339.9079 949.05774]]	[[25.24542 1118.9033]]	[[241.8391 1110.4811]]
[[66.883606 1101.5881]]	[[357.5246 1114.4601]]	[[44.440098 1017.5886]]	[[341.7145 1019.2161]]	[[32.46188 1219.1827]]	[[245.77217 1207.3339]]
[[63.058006 919.1142]]	[[358.84085 937.67926]]	[[41.486588 922.69476]]	[[342.23038 927.7781]]	[[30.12854 1116.2262]]	[[246.54764 1107.9894]]
[[70.49394 1012.9677]]	[[363.49744 1028.7882]]	[[49.576355 1017.1422]]	[[346.6644 1019.14734]]	[[27.449528 1022.66437]]	[[246.76234 1017.8487]]
[[66.041336 715.7589]]	[[366.0069 740.8418]]	[[52.281063 1003.3159]]	[[349.5066 1005.8666]]	[[31.682098 1119.8914]]	[[247.74059 1111.6875]]

Fig4. 部分的對應點

而這次的設計方法是以第一張不動的情況下，第二張和第一張進行 findhomography 找到 H 矩陣，在進行 warpPerspective 轉換過去，但在進行第三張合併時，發現合併後的第二張的點不是跟第三張對應的點一樣，透過推導公式發現，第一張與第二張得到的 H 矩陣，只要在乘上第二張與第三張的對應點(只需乘第二張的對應點)，就可以得到 wrap 過後的對應點，再與第三張做 findhomography，步驟與第一次合併一樣，接下來以此類推，如 Fig6.所示，而透過資料的呈現方式，我決定從 004.jpg > 003.jpg > 002.jpg > 001.jpg(從左到右看過去)。在第二張透過 warpPerspective 轉換過去之後，要如何跟第一張結合，如果直接相加會發現很明顯的重疊區域，而我使用一個簡單的方式，如 Fig5.所示，每次在要拼接之前，透過此方式可以解決重疊區域的問題，在影像矩陣運算中，相減完的數值，如果大於 255 會等於 255，而小於 0 時會等於 0，使用這個特性，相減完之後我可以得到沒有重疊的區域，且可能會說為甚麼被減的圖像沒重疊處也不見了，也是因為前面的特性小於 0 等於 0，最後我再把第一張圖跟相減完之後的結果相加，再重複動作直到圖片沒有為止。

Fig5. 重疊問題(左圖:第二張轉換之結果;中間圖:第一張圖;右圖:相減完之結果)

Fig6. 四張圖片拼接之結果

但是上述剛剛說的作法，是有存在缺陷的，還是有會一點點的縫合線出現，在某些情況下縫合線會看來得不連續，加上一開始的方法就只是第一張不動，第二張轉換完之後拼接上去，導致第二張照片要配合第一張，第一張照片不需要更動，看起來的效果也有點怪怪的，與 github 上面其他人寫出來的效果有差，這是我之後如果要繼續往拼接上專研要研究的地方，加上在演算法中解決縫合線屬於一個 DP 的問題，這也是未來要突破的一個方向。