Computer Vision HW2 Report

310551079 梁友誠

Part1. Implementation

1. **Read Image:**

   讀取要拼接的兩張圖片並轉成灰階圖，讓 SIFT 可以使用灰階圖來計算 key points 以及 descriptors。

2. **Use SIFT to find key points and descriptors:**

   利用 SIFT 來計算上一個步驟所獲得的兩張灰階圖的 key points 以及 descriptors，其中 key points 為此圖片所包含的所有特徵點座標(x, y)，而 descriptors 為一個(key points 數*128)的矩陣

   (P.S: 此作業使用的 OpenCV 版本為 4.5.5，因此 SIFT 的函式為 cv2.SIFT_create())。

3. **Match key points between two images:**

   利用 KNN(K=2)來配對兩張圖片相同的特徵點，其做法為利用第一張圖片中每個特徵點 P 的 descriptor 去計算與第二張圖片中每個特徵點的 descriptor 之間的歐式距離，並記錄距離最短的 2 個 descriptor 為 D1 與 D2 (對應的 key point 為 P_1, P_2)，並判斷:

   $$D1 < D2 * threshold(0.7)$$

   若成立則代表 D1 所對應的 key point P_1 為 good match，因此可以配對 P 與 P_1。若不成立則代表 P 與 P1 跟 P2 的特徵都很接

近，因此不將此點列為配對點。此外，SIFT 所產生的 key points 中會有同一個特徵點有不同 descriptor 的情形，因此在使用 KNN 獲得配對完的特徵點後會有重複配對的情況，為了讓後續的 RANSAC 可以隨機挑選到不重複的配對特徵點，而在使用 KNN 完後要把相同配對的特徵點給刪除。

4. **Use RANSAC to get best Homography matrix:**

在獲得兩張圖片中配對好的關鍵點後可用 RANSAC 來獲得最佳的 Homography matrix，我所設的超參數為 iteration = 3000, threshold = 5，在每次的 iteration 中會隨機選取 4 組配對好的關鍵點，並計算 Homography matrix，其算法為利用 4 組 key point 的座標[($x_s^{(1)}$, $y_s^{(1)}$), ($x_d^{(1)}$, $y_d^{(1)}$)], [($x_s^{(2)}$, $y_s^{(2)}$), ($x_d^{(2)}$, $y_d^{(2)}$)], … , [($x_s^{(4)}$, $y_s^{(4)}$), ($x_d^{(4)}$, $y_d^{(4)}$)]建立矩陣 A 以及用 SVD decomposition 來求以下圖片的 h_{11}~h_{33}，SVD 可算出 A^TA 的 eigenvector 跟 eigenvalue，其中最小的 eigenvalue 所對應的 eigenvector 就是 h_{11}~h_{33}，並將 h_{33} normalize 到 1 即可得到 Homography matrix。

$$\begin{bmatrix} x_s^{(1)} & y_s^{(1)} & 1 & 0 & 0 & 0 & -x_d^{(1)}x_s^{(1)} & -x_d^{(1)}y_s^{(1)} & -x_d^{(1)} \\ 0 & 0 & 0 & x_s^{(1)} & y_s^{(1)} & 1 & -y_d^{(1)}x_s^{(1)} & -y_d^{(1)}y_s^{(1)} & -y_d^{(1)} \\ & & & & \vdots & & & & \\ x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)}x_s^{(i)} & -x_d^{(i)}y_s^{(i)} & -x_d^{(i)} \\ 0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -y_d^{(i)}x_s^{(i)} & -y_d^{(i)}y_s^{(i)} & -y_d^{(i)} \\ & & & & \vdots & & & & \\ x_s^{(n)} & y_s^{(n)} & 1 & 0 & 0 & 0 & -x_d^{(n)}x_s^{(n)} & -x_d^{(n)}y_s^{(n)} & -x_d^{(n)} \\ 0 & 0 & 0 & x_s^{(n)} & y_s^{(n)} & 1 & -y_d^{(n)}x_s^{(n)} & -y_d^{(n)}y_s^{(n)} & -y_d^{(n)} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

在獲得 Homography matrix H 後即可將所有配對好的關鍵點 (P_1,P_2)利用 H 來將 P_1 投影到 P_2'，並計算投影誤差$||P_2 - P_2'||$，若此誤差小於 threshold(5)則可將此點列為 inlier，因此可計算用此 H 來得到的 inlier 數總合，RANSAC 就是要在 iteration(3000)內找出最大的 inlier 數所對應的 H 來當成是最佳的 Homography matrix，而為了加快計算速度，若找到的 H 符合 (inlier 數/match 數) > 0.8 也會停止 RANSAC，並將此 H 設為最佳 Homography matrix。

5. **Warp two images with homography matrix H:**

為了更好將第一張圖片(左圖)拼接到第二張圖片(右圖)，需先對齊兩張圖片的座標系並且讓這兩張的大小相等，而作法為先利用 H 將第一張圖片的四個角落座標系轉換到第二張圖片的座標系，也就是用 H 分別乘以(0, 0, 1), (0, w, 1), (h,0, 1), (h, w, 1)得到 (x1, y1, z1), (x2, y2, z2) …, (x4, y4, z4)，再把這些結果的 x 與 y 都除以 z 才會得到真正在第二張圖片的座標系，將上述除完 z 的結果分別取 x 的最小值 x_min 以及 y 的最小值 y_min 後取絕對值再加上第二張圖片的 w, h 就可當成是投影後的圖片大小。此外，轉換後的 x1~x4, y1~y4 可能會有負值出現，使得第一章圖片投影完後有些部分會消失，因此除了使用 H 外也要需額外使用 affine translation 來將消失的部分位移回來，同樣地對於第二

圖片也要使用 affine translation 來跟第一張圖片對齊。

6. **Use linear blending to concatenate two images smoothly:**

在獲得擁有相同座標系的兩張圖片後就可使用 Linear blending 的方法來拼接兩張圖片，讓中間重疊或者是邊界的部分看起來更平滑。其作法為先算出兩張圖片重疊的部分，並根據每個 row 重疊的 pixel 數目來分配左右兩張圖片的權重，越靠近右邊圖片的重疊部分左圖的權重會較小，而右圖權重較大，反之越靠近左邊圖片的重疊部分左圖的權重會較大，而右圖權重較小，並使得此全權重的總和為 1 來維持原本的像素值。舉例來說，img1 與 img2 在 h = 300, w = 11~20 的地方有 10 個 pixel 重疊，則左圖 w_{11}~w_{20} 的權重分別為 1~0.1，而右圖 w_{11}~w_{20} 的權重則為 0~0.9。下圖為左右兩張圖片透過 linear blending 後產生的結果，再把這兩張結果的每個像素值相加就可以得到最後的拼接圖。

Part2. Result of stitching 2 images

1. Stitch m1.jpg and m2.jpg

2. Stitch m3.jpg and m4.jpg

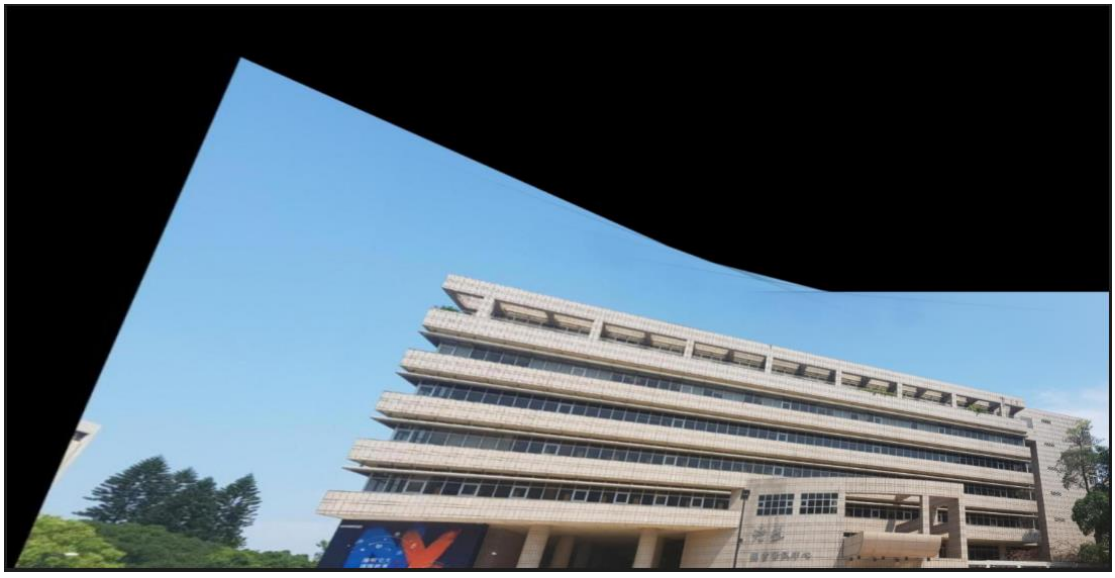
3. Stitch m1.jpg and m4.jpg

Part3. Stitch more images

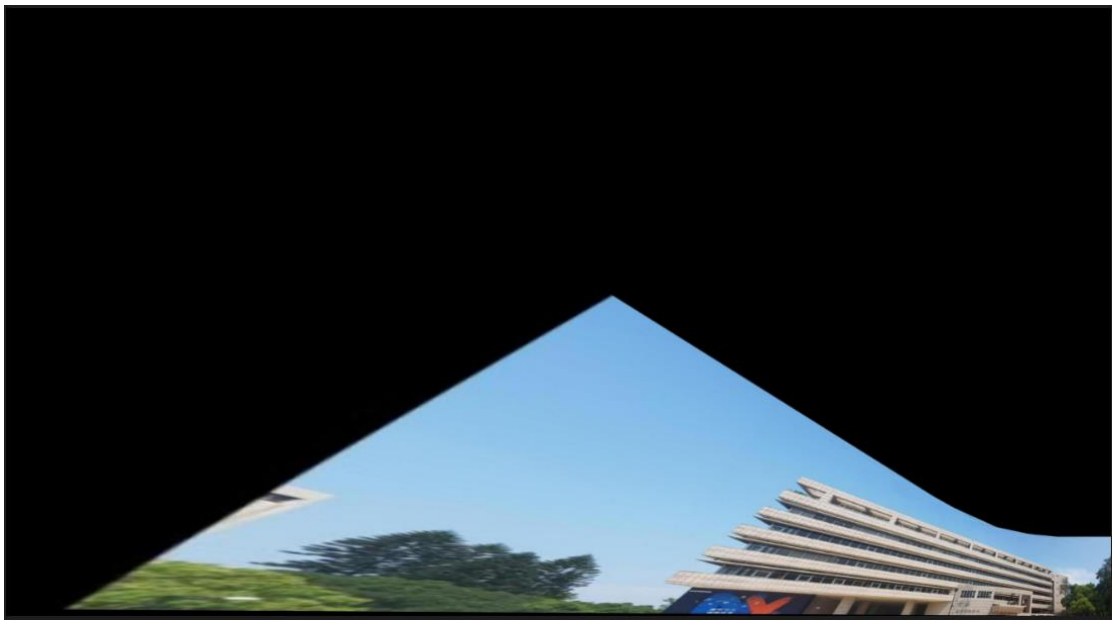
在此部分中，我採用先拼接 m1 與 m2 來得到 m12.jpg，再用 m12 與 m3 拼接得到 m123.jpg，也就是先把拚接兩張圖片的結果存下來後再依序拼接，以下是拼接多張圖片的結果:

1. Stitch m1~m4:

2. Stitch m1~m6:



3. Stitch m1~m8:



從以上結果可看出，拼接越多張照片後，照片的尺寸會變大許多(m1234 的尺寸為 1994*1052，而 m12345678 的尺寸為 10553*5793)，不僅會增加拼接圖片時的運算時間，而且產生的結果也會變得更扭曲，實際物體在圖片中的比率也變小許多。