

Computer Vision Homework 2 Report

311554057 陳宇呈

1. SIFT in OpenCV :

這部分是直接呼叫 OpenCV 中的 function `kp, des = SIFT_Detector.detectAndCompute(img, None)` 能夠得到圖片的所有 keypoints 和 descriptors 用這兩個資訊去進行下一步的特徵檢測。

2. Feature matching - KNN :

這部分是使用的 Brute-force KNN 去找到最近的兩個 descriptor，會遍歷第一張圖片的每個 descriptor，然後計算與第二張圖片中所有 descriptor 的 Euclidean 距離，然後根據 Euclidean 距離找出最好與次好的 match，之後根據 Lowe's Ratio 來篩選出好的 match，這邊 threshold 是設為 0.75，並且我在最後的部分求出兩張圖片對應為好的 match 的 keypoint 座標，方便後續的計算。

3. Homography :

要求出 Homography 矩陣需要建構出線性系統為： $P_2 = H \cdot P_1$, $P_2 = (x_2, y_2, 1)$, $P_1 = (x_1, y_1, 1)$ ， P_1 和 P_2 為 correspondence points， H 為 homography 矩陣，若想解出 Homography 矩陣至少需要 4 對 match 才能解出矩陣中的 8 個未知數，那 4 對 match 會從 RANSAC 中隨機挑選出來，然後我是照課本令方程式為 $A \cdot h = 0$ ，如下圖：

$$\begin{array}{ccccccccc} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -x_3Y_3 & -x_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -y_3X_3 & -y_3Y_3 & -y_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -x_4Y_4 & -x_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -y_4X_4 & -y_4Y_4 & -y_4 \end{array} * \begin{bmatrix} A_{11} \\ A_{12} \\ A_{13} \\ A_{21} \\ A_{22} \\ A_{23} \\ A_{31} \\ A_{32} \\ A_{33} \end{bmatrix} = 0$$

大寫為第一張圖片的座標，小寫為第二張的座標。之後使用 SVD 分解找到方程式的最小平方誤差解，**Homography** 矩陣會是 V 的最後一列，然後根據 H 中最後一個元素對 H 做 normalize，我們就可以得到 **Homography** 矩陣。

4. RANSAC :

這部分是為了找到最好的 H ，他會重複 N （設定為 2000）遍，每次 loop 會先去隨機挑選出 4 對 correspondence points，使用這些點來計算出 **Homography** 矩陣，並且根據計算出來的 **Homography** 矩陣對於所有好的 match 來計算出 $p2'$ ， $p2'$ 的計算公式為 $p2' = p1 * H$ ，算出 $p2'$ 後去與實際的 correspondence points $p2$ 計算兩個的距離若小於 threshold（設為 5）的話，inliers 的數量就遞增，然後檢查這次的 loop 的 inliers 數量有沒有大於之前任何時候的 inliers 數量，如果有就更新 inliers 數量和將 **Homography** 矩陣更新為這次 loop 的 H ，最後就會得到最好的 H 。

5. Stitching image :

這部分是要將兩張圖片拼接起來，首先使用 **Homography** 矩陣算出左邊圖的 4 個角落在右邊圖 perspective 的位置，之後我們要計算出將兩個圖像組合在一起的大小 size 的計算方式是投影片上的方法，寬度為右邊圖的寬度加上計算出的四個角落座標中 x 座標的最小值 (x') 若大於 0 則取 0，高度為右邊圖的高度加上計算出的四個角落座標中 y 座標的最小值 (y') 若大於 0 則取 0。對於左邊圖，我們要計算出一個新的 **Homography** 矩陣，新的 **Homography** 矩陣是原始 **Homography** 矩陣乘上 Affine translation 矩陣

$A \left(\begin{bmatrix} 1 & 0 & -x' \\ 0 & 1 & -y' \\ 0 & 0 & 1 \end{bmatrix} \right)$ 而得到的，然後對左邊圖和右邊圖分別進行 warp

perspective，左邊使用的是新的 **Homography** 矩陣而右邊則是使用 Affine translation 矩陣 A 。做完 warp perspective 後我們會得到兩張 warp 後的圖，接下來是將他們拼接起來。

6. Blending

而如果直接拼接會有明顯的界線，所以要進行 blending，這部分會去遍歷所有 pixel 然後去比較在這個 pixel 是左邊 pixel 的 RGB 總和大還是右邊 pixel 的 RGB 總和大，之後選擇 RGB 總和大的當作這個 pixel 的結果，最後就會建構出一張拼接完成的圖片。

Result :

Result of baseline :

	baseline
m1~m2	 A photograph of a modern swimming pool building. The building has a light grey textured facade on the left and a blue-painted section on the right containing large windows. The word "游泳館" (Swimming Pool) is written in white on the blue part. The building is set against a clear blue sky. In the foreground, there is a paved area with a red curb and some greenery.

m1~m3



m1~m4



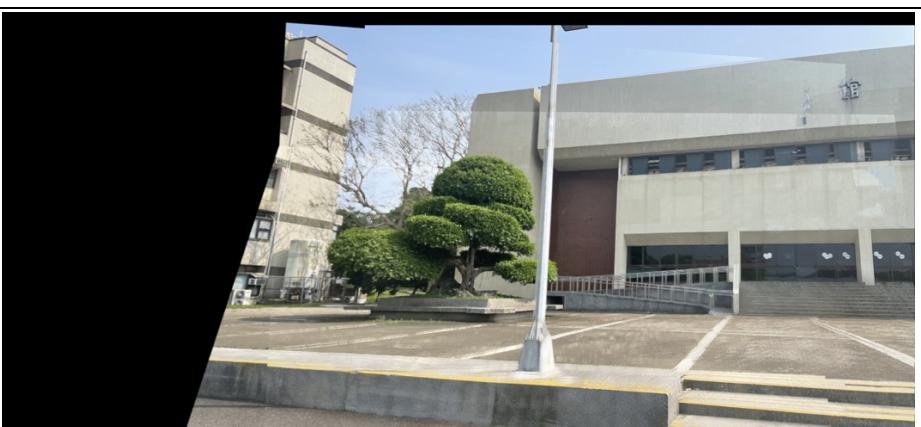
m1~m5



m1~m6



Result of bonus :

	bonus
m1~m2	
m1~m3	
m1~m4	

Conclusion :

可以看到在圖片的接縫處還是會有一點畫素不均勻的部分，而且隨著多張圖片的拼接，拼接的圖片會變形的越來越嚴重，而從 **baseline** 的結果看起來會像是從左邊視角看過去，但投影片上的結果視角卻是中間看過去，這部分是需要進一步處理的部分。而 **bonus** 的部分則是他消除了 **m1** 的最底部箭頭的部分，可能是因為經過 **warp perspective** 後箭頭的部分超出範圍被截掉了。結論是從整體上來看還是有將圖片完整的拼接起來，只是需要再進一步將一些細節完善。