

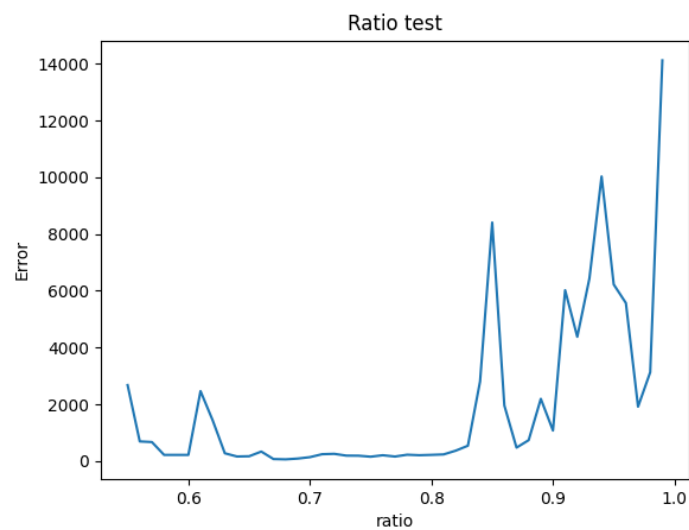
3DCV HW1 Report

電子所碩一 R10943117 陳昱仁

一. Problem 1

1. Outliers removal

運算 Homography 時，需先找出兩張圖的對應點，這邊使用的 detector 是 SIFT，但找出的點 outlier 還是太多，調整 Lowe's ratio test 的 ratio 來剔除多餘的點，依實驗結果，若使用所有點，0.65 - 0.85 間有較穩定之結果，最後使用最好的結果 $\text{ratio} = 0.68$ ，如下圖所示。


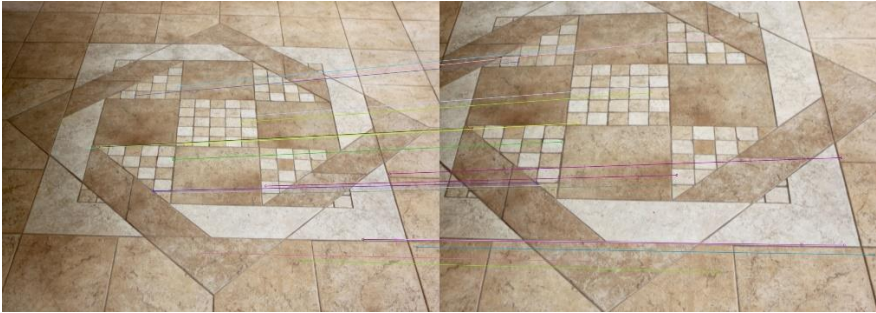
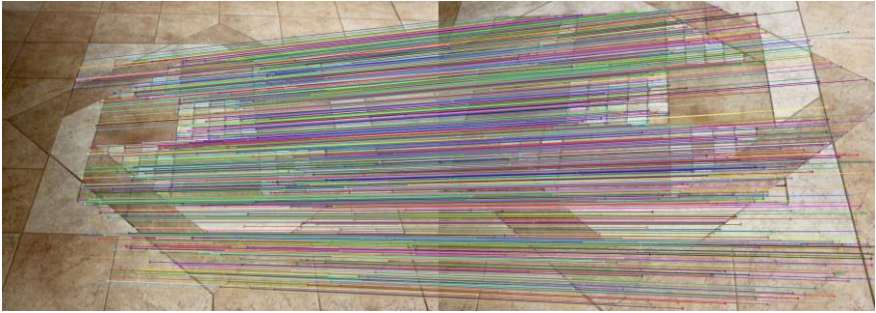




再來要剔除 outliers，來選出 k 個點，使用的演算法是 RANSAC，在有限疊帶次數，每次隨機選出 k 個點，若此組合在 SIFT 找出的對應點中有最多 inliers，則選此組合為最佳 k 個點，並另外使用所有 inliers 來和不同 k 做比較。

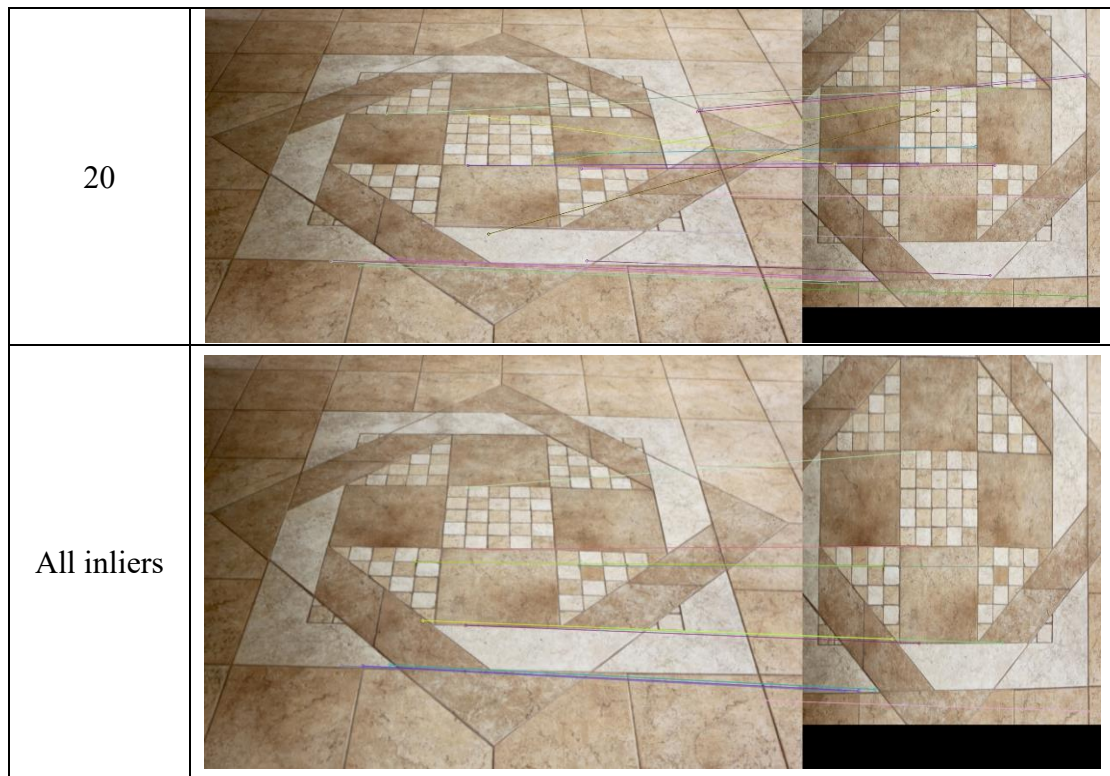
2. Matching result

下方為剔除 outlier 後，找出之對應點結果。

k	1-0 and 1-1
4	

8	
20	
All inliers	

k	1-0 and 1-2
4	
8	



3. Normalization

為了進一步得到更好的結果，將座標 normalize 到 1，座標會在 3 度空間的球面附近，normalize 方法使用的是 Hartley algorithm，計算方式及矩陣如下。

$$s = \left(\frac{1}{2n} \sum_{i=1}^n \|m_i - \bar{m}\|^2 \right)^{1/2}$$

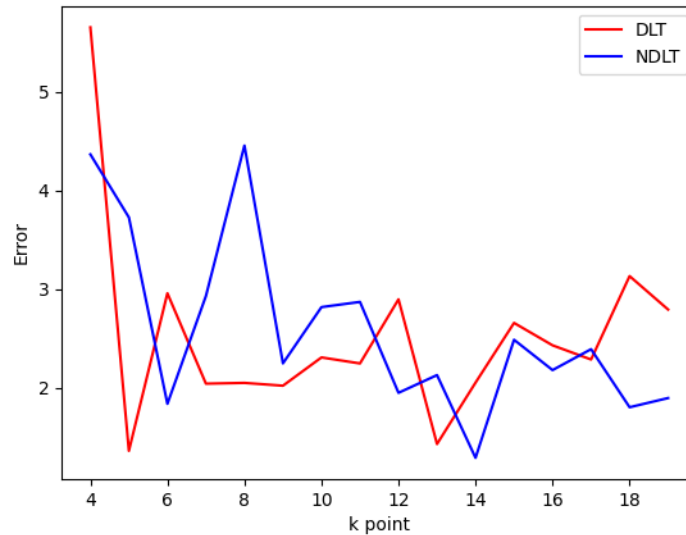
$$= \left(\frac{1}{2n} \sum_{i=1}^n (m_{1,i} - \bar{m}_1)^2 + (m_{2,i} - \bar{m}_2)^2 \right)^{1/2} \quad \mathbf{T} = \begin{bmatrix} s^{-1} & 0 & -s^{-1}\bar{m}_1 \\ 0 & s^{-1} & -s^{-1}\bar{m}_2 \\ 0 & 0 & 1 \end{bmatrix}$$

評估平均 error，左側為沒 normalize 的 error，右側為 normalize 的 error。

k	1-0 and 1-1	1-0 and 1-2
4	0.234 / 0.211	3.458 / 1.865
8	0.158 / 0.162	2.915 / 1.651
20	0.179 / 0.108	1.503 / 3.075
All inliers	0.098 / 0.092	4.829 / 2.188

由於 RANAC 有隨機性，除了使用 all inliers 以外，其他結果為十次的平均誤差，可以看到 k 越大，運算 H 矩陣需滿足的方程式越多，大致上誤差越小。有 normalize 也會比沒 normalize 效果好一點，使用所有 inliers 大略也會比其他狀況好。

1-1 和 1-0 相似，所以能找到較佳之對應點，error 也較小，1-2 和 1-0 相機角度落差較大，error 較大，也相較沒那麼穩定，大致上誤差都在個位數。下方圖為 k 在 4-20 間，圖 1-0 和 1-2 沒 normalize 和 normalize 之比較。

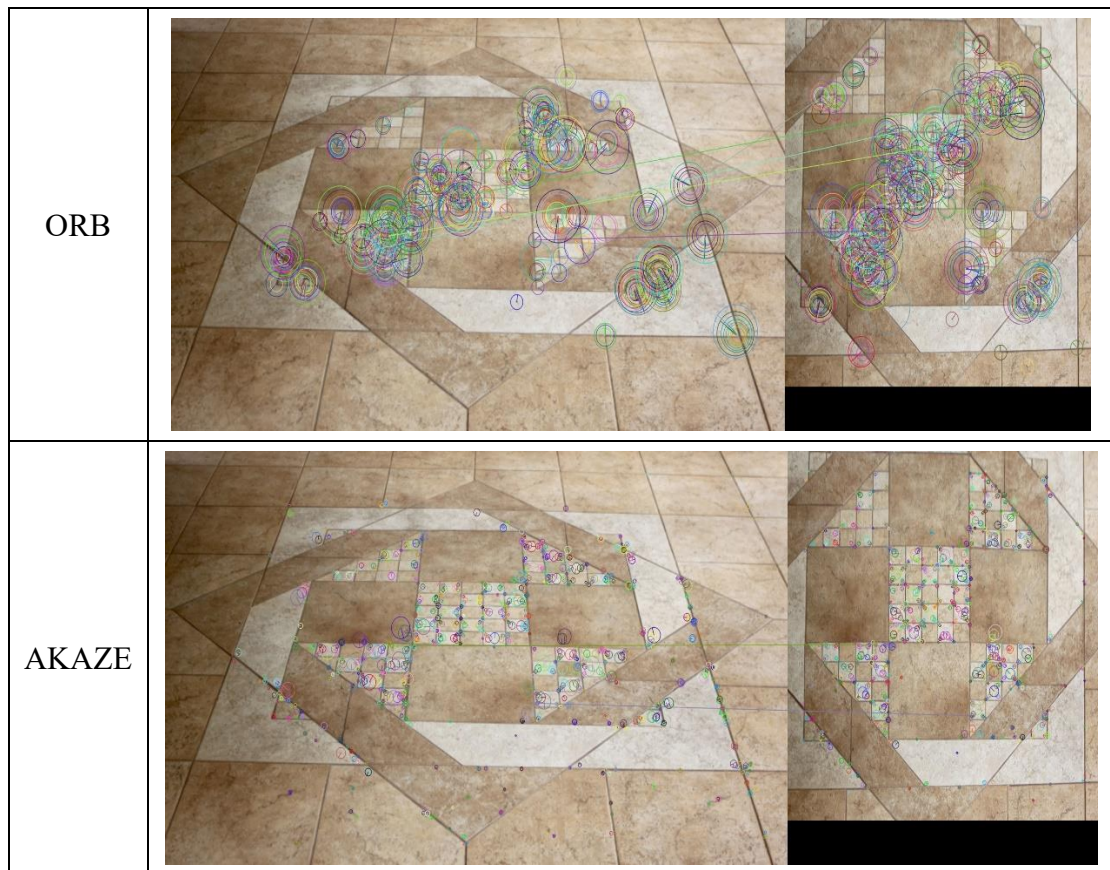


由上圖可知，除了少數有 noise 的 case，大多數情況在 k 較多的情形，有 normalize 能達到較好的效果。

4. Different detector

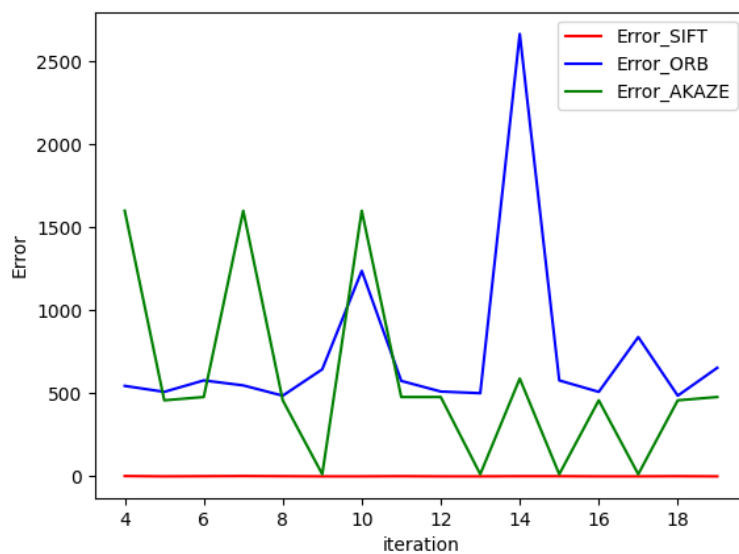
另外探討不同 feature detector 對準確度之比較，使用的是 SIFT、ORB、以及 AKAZE，從演算法的原理可知道，ORB 以及 AKAZE 皆比 SIFT 運算快，找到的點較少，如下圖所示。

Detector	1-0 and 1-2
SIFT	



從上圖還可以觀察到，SIFT 找的点除了較多以外，分布也較為均勻，ORB 以及 AKAZE 大多分布在 1-0 圖片中心梯度較高區域，在這個紋理大量重複的例子，很容易找到配對錯的点，點數不足的情形更難以算出好的 H 矩陣。

實驗並觀察在使用所有匹配点的情形，SIFT 還是相較另外兩者還來的準確穩定許多，換言之代表另外兩者是由準確度來換取速度。



5. Slope selection

為了使效能提升，觀察 1-0 到 1-2 之間的轉換，相機角度不同，不同位置紋理相近，配對較為困難。觀察其 $k=20$ 的 matching 圖，如下方紅圈處所圈出之配對點之連線，明顯不對，且相較於其他配對連線，斜率來的高。

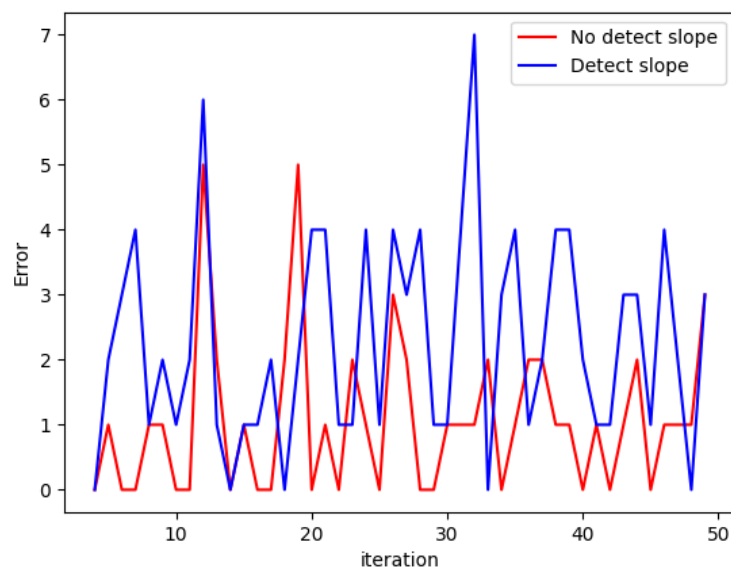
因此提出一個假設，在相機拍攝尺度差異不大的情形下，配對點的連線斜率，彼此之間應該差異不會太大，若斜率差異太大甚至正負號顛倒，判定此配對為 outlier，此法命名為 "Slope selection"。



演算法如下

1. 求出所有線斜率並求出平均值
2. 若該線和平均斜率差值之絕對值大於 threshold，判定為 outlier
3. Return inliers

觀察 Slope selection 的結果，絕大多數情形，誤差都比沒做此法來的高，原因應該是有些不錯的配對點也會被刪除掉，此法應有其他未考慮到的改善空間。



二. Problem 2

這部分將照片轉正，為了避免有沒填到的 pixel，使用的是 backward wapping，將輸出圖片座標位置轉回原圖找到其對應位置，值得注意的是，為了方便 backward wapping，anchor_img 角點是輸出圖的四個角落，target_img 對應角點是使用 mouse_click_example.py 在原圖找到的點，如此一來便不用再求 H 的反矩陣便能直接使用。結果如下。



在執行 backward wapping 時，若使用兩層迴圈，一個一個 pixel 找對應點速度會太慢，應使用 numpy 中 vector computation 的方式，降低 for loop 使用，加速程式運行。如下程式碼所示，pt 是矩陣，所有運算皆不需用到 loop。

```
def interpolation(anc_img, pt):
    h, w = pt[:, 0], pt[:, 1]
    h0, w0 = np.floor(h).astype(int), np.floor(w).astype(int)
    h1, w1 = h0 + 1, w0 + 1

    a, b = anc_img[h0, w0], anc_img[h0, w1]
    c, d = anc_img[h1, w0], anc_img[h1, w1]

    h1_h, h_h0 = h1 - h, h - h0
    w1_w, w_w0 = w1 - w, w - w0
    wa = np.expand_dims((h1_h * w1_w), axis = -1)
    wb = np.expand_dims((h1_h * w_w0), axis = -1)
    wc = np.expand_dims((h_h0 * w_w0), axis = -1)
    wd = np.expand_dims((h_h0 * w1_w), axis = -1)

    x = (wa*a + wb*b + wc*c + wd*d).astype(np.uint8).reshape(anc_img.shape)

    return x
```

三.結果與討論

Part1 中，算出 H 矩陣難度不高，但紋理重複的照片，outliers 的剔除方法需要多加考量，這裡採用的方法有 ratio test，RANSAC，對其再分別比較有 normalize 以及沒 normalize 之結果。並額外比較不同 feature 的準確度，以及觀察提出”Slope selection”，做了許多實驗並比較數據。

Part2 中，原先是使用雙層 loop 來找對應點，但由於圖片畫質高，要算太久，改成運用 vector computation 來運算加速。

另外還有一未實現的演算法為”cross line detection”，若某 matching line 和其他 matching line 相交數量過多，則為 outlier 可能性較高，想法上和”Slope selection”類似，相交數量多代表斜率和其他線差異大，須將其剔除掉，希望未來有機會實現這個方法。