3DCV homework 1

朱世耘 r09922115

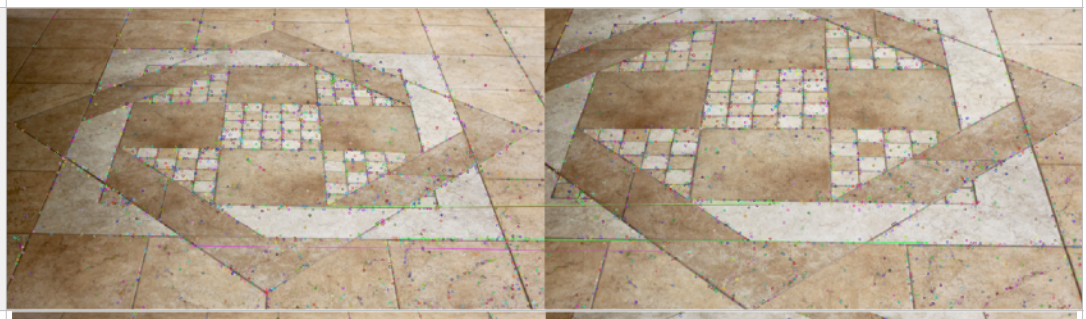
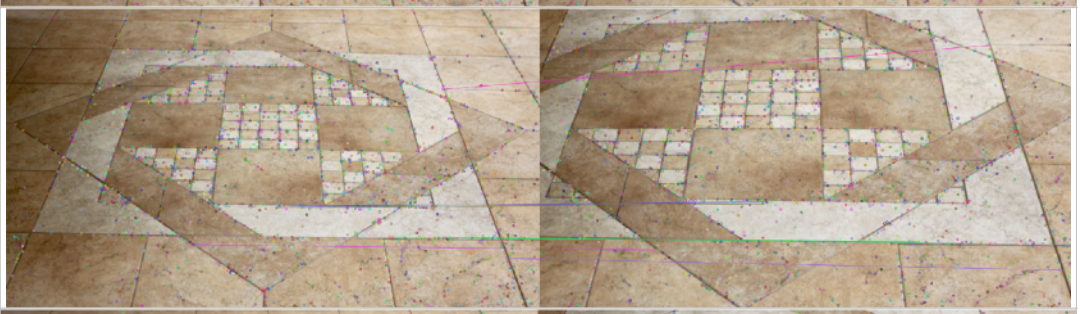
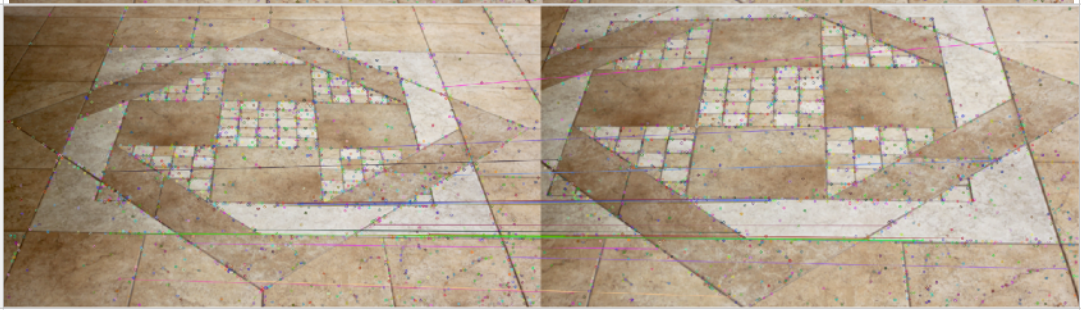
used library:

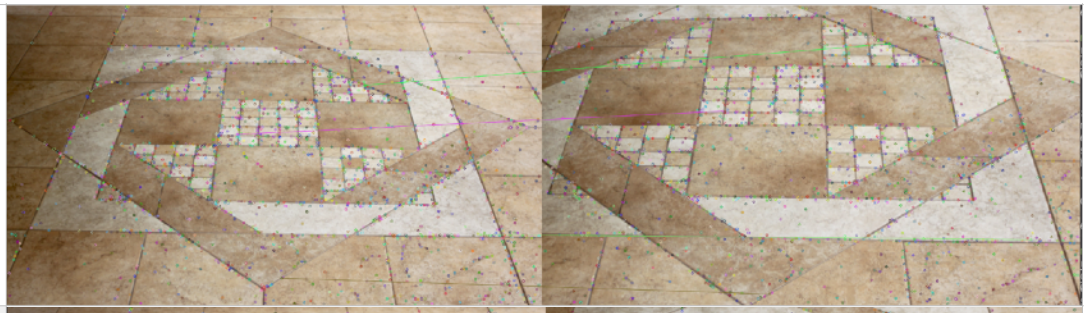
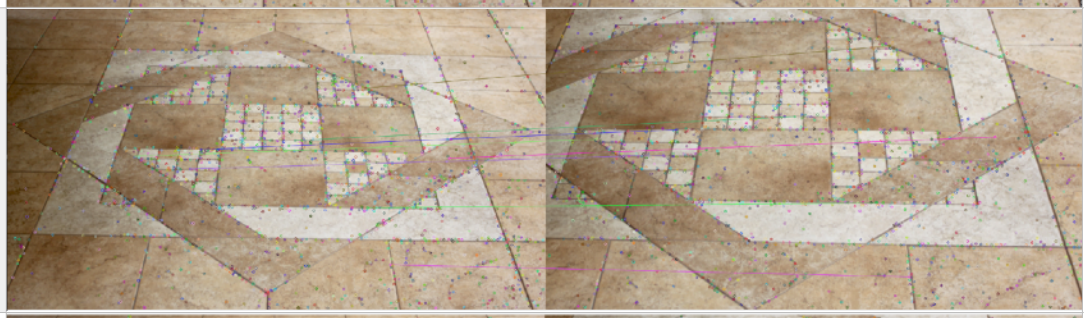
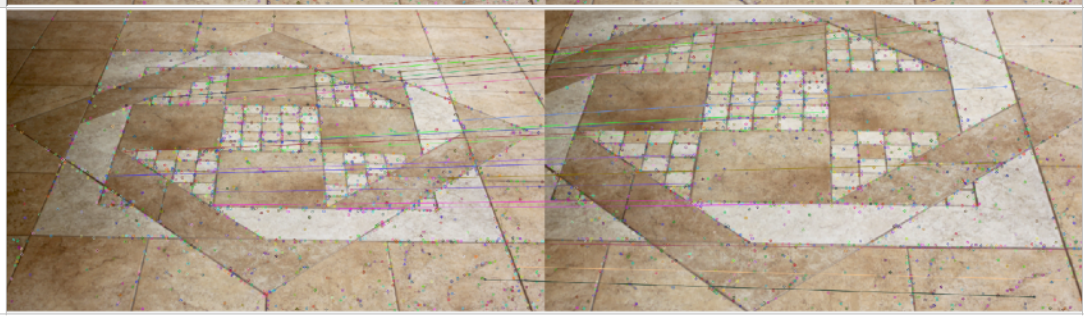
- numpy:處理陣列
- cv2: 讀寫檔、SIFT
- tqdm: 跑進度條看起來程式執行比較快
- copy: 複製陣列防止資料不必要的更動
- random: ransac時使用
- math: bilinear interpolation時使用

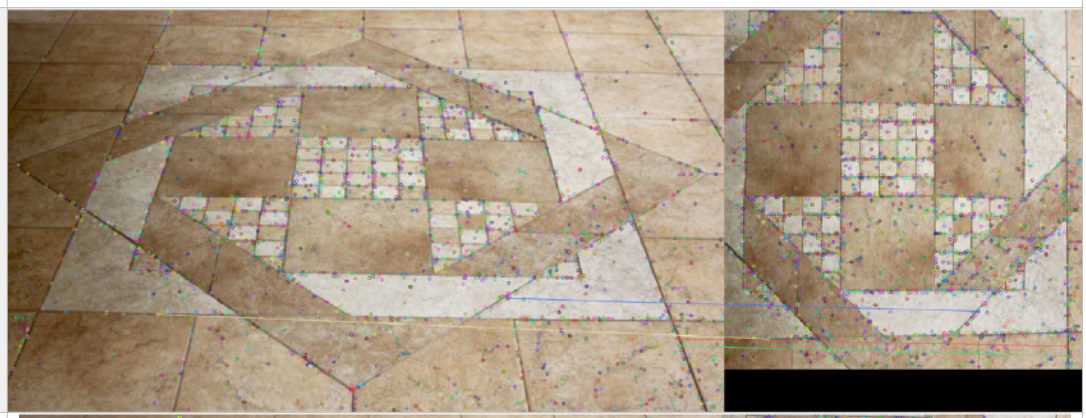
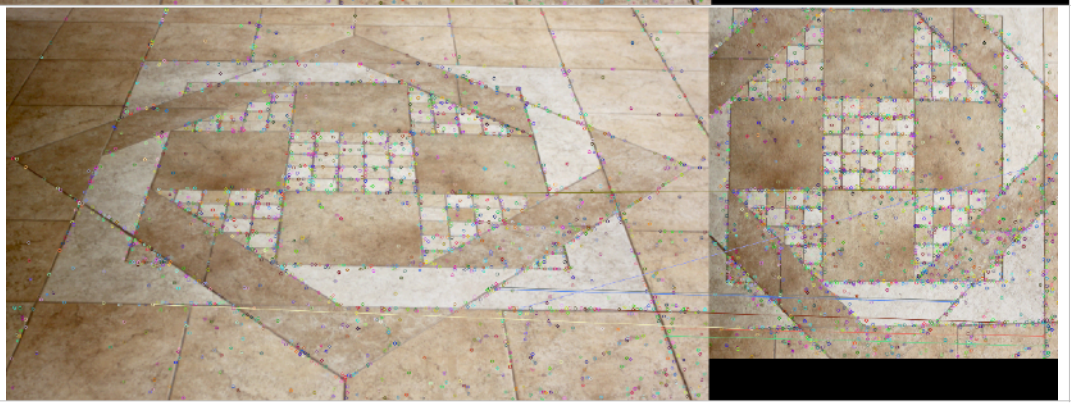
Problem 1:

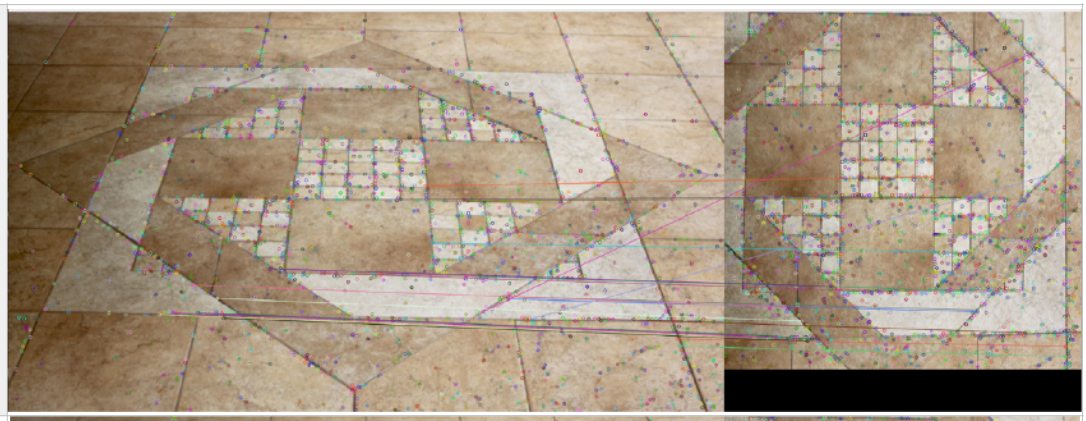
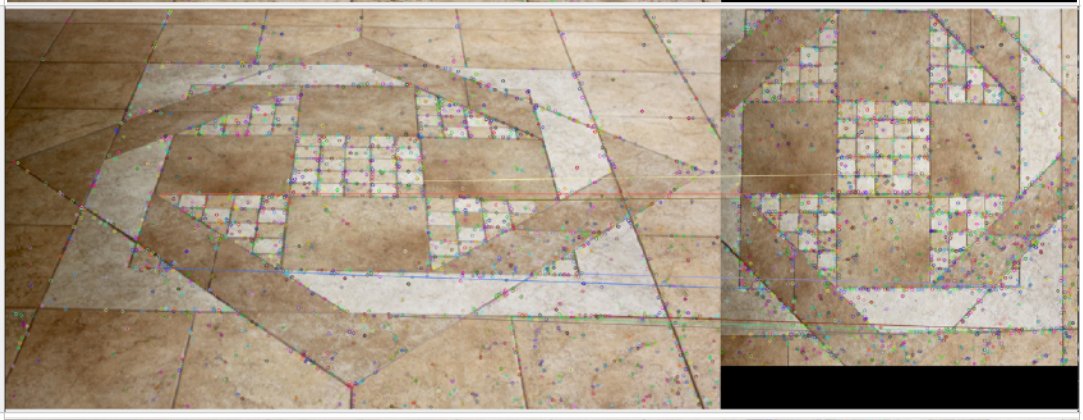
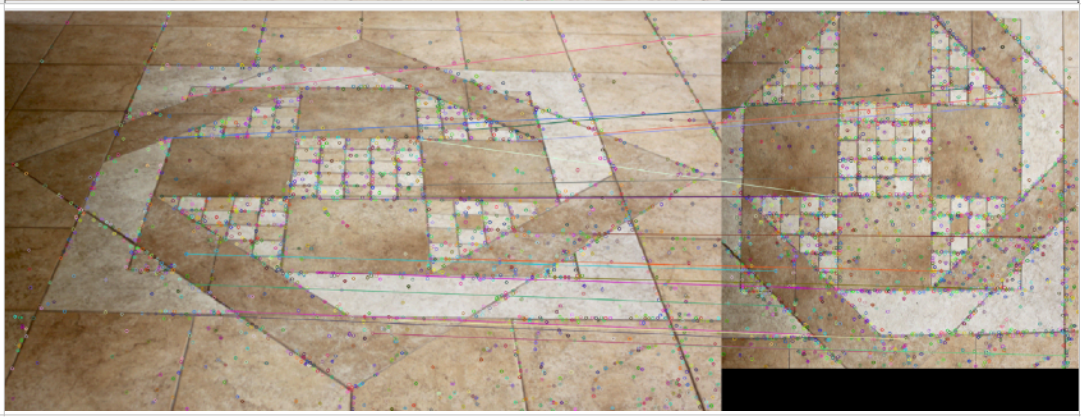
(a),(b)

使用ransac來挑出outlier，先挑1000個matching points，再隨機挑出k個點來當特徵點做轉換，轉完之後把1000個點做轉換來算error，重複操作300次之後挑出error最小的組合。

number of corresponding points	1-0 and 1-1
4 points	
8 points	
20 points	

4 points (reject outlier)	
8 points (reject outlier)	
20 points (reject outlier)	

number of corresponding points	1-0 and 1-2
4 points	
8 points	

20 points	
4 points (reject outlier)	
8 points (reject outlier)	
20 points (reject outlier)	

discussion:
可以看到第二張圖用SIFT容易出現許多outlier，所以一定要用RANSAC來做排除outlier，而在不容易出現outlier的圖一，取越多點會更穩定做得越好，但是圖二是取越多點就容易取到outlier，反而造成偏差，下表中的error就可以看出來。

(c)(d)

1-0 to 1-1

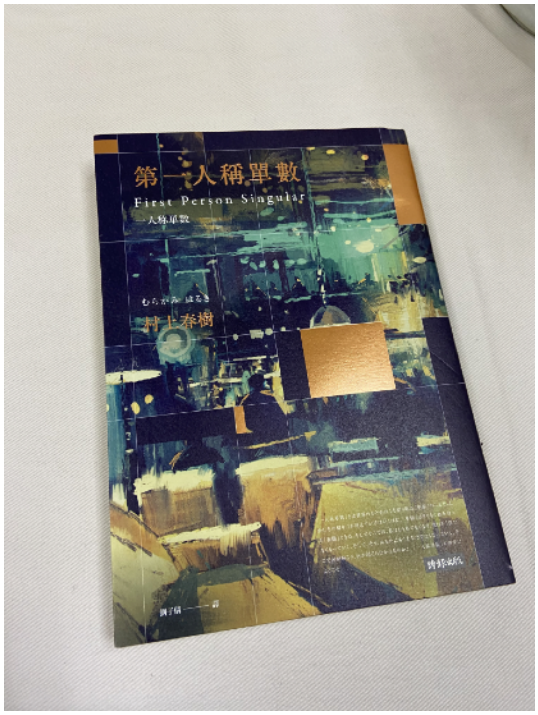
	transform	normalized transform
4 points (reject outlier)		
reconstruct error	0.918	0.918
20 points (reject outlier)		
reconstruct error	0.726	0.732

1-0 to 1-2

	transform	normalized transform
4 points (reject outlier)		
reconstruct error	2.021	2.012
20 points (reject outlier)		
reconstruct error	8.777	8.273

(e)
I have implemented RANSAC when finding outlier, and bilinear interpolation when doing image transform

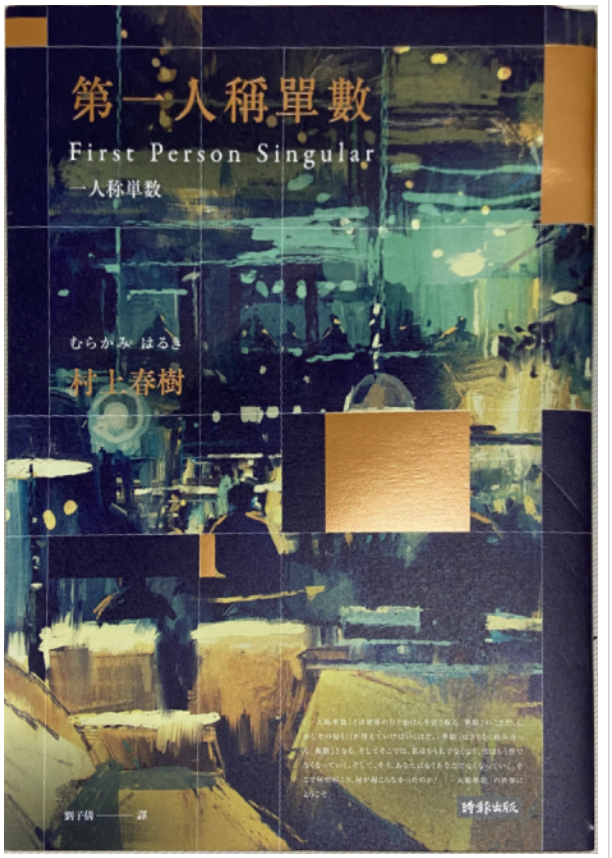
Problem 2
(a)
original image:
choose 4 corners as feature points, try to transform 4 points into 4 points respectively.



(b)
using bilinear interpolation as textbook

```
def interpolation(img, x, y):
    #print(img)
    x1 = math.floor(x)
    x2 = math.floor(x+1)
    y1 = math.floor(y)
    y2 = math.floor(y+1)
    x_d1 = x-float(x1)
    x_d2 = float(x2)-x
    y_d1 = y-float(y1)
    y_d2 = float(y2)-y
    img=np.array(img)
    if x1 < 0 or x2 >= img.shape[0] or y1 < 0 or y2 >= img.shape[1]:
        return 0
    else:
        result=0
        result+=x_d1*y_d1*(img[x2][y2].astype(float))
        result+=x_d1*y_d2*(img[x2][y1].astype(float))
        result+=x_d2*y_d1*(img[x1][y2].astype(float))
        result+=x_d2*y_d2*(img[x1][y1].astype(float))
    result.astype(int)
    #print(result)
    return result
```

(c)
result:



discussion:
基本上做法跟第一題一樣，而且自己做完了找對應點的工作，這題中比較麻煩的地方是做bilinear interpolation，如果沒有想辦法加速的話，要跑好幾分鐘，然後型別的地方要注意轉成float來計算。