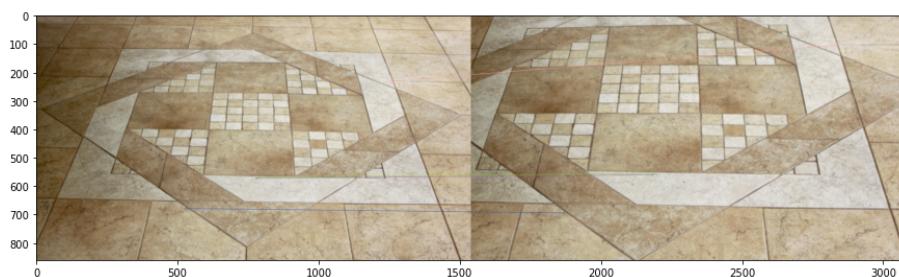


# TDCV HW1 Report

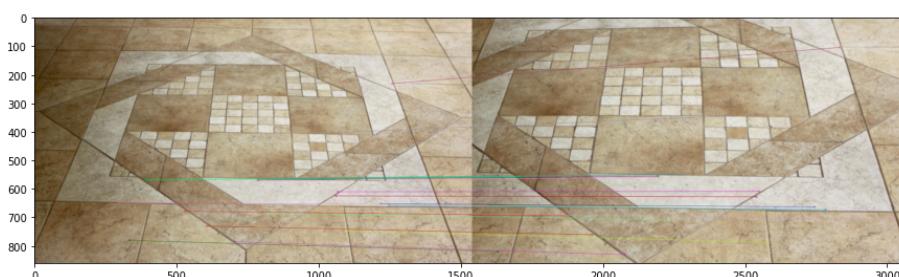
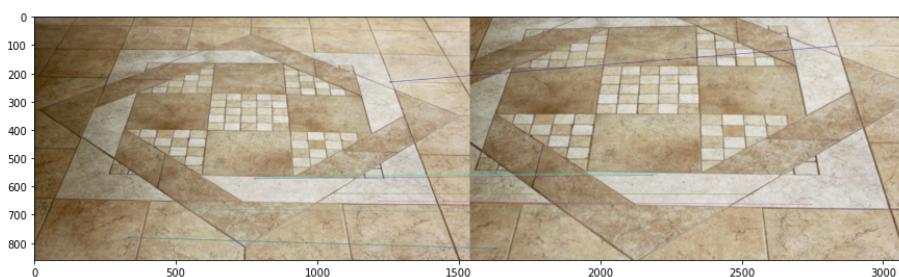
r10922061 資工所 王竑睿

## Problem 1: Homography estimation

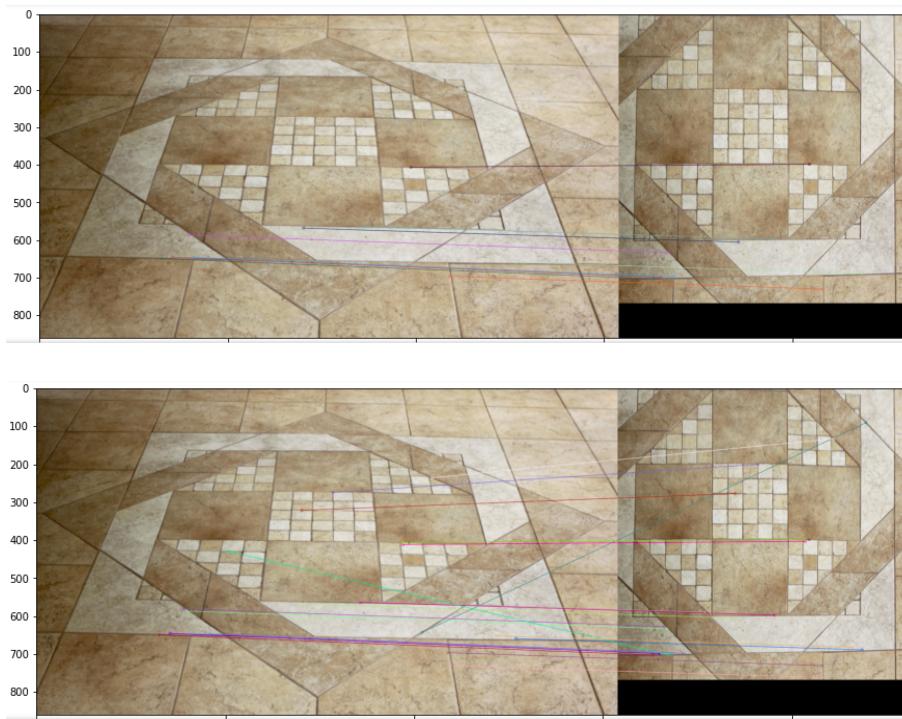
### Screenshots:



- 由上而下分別為  
 $k = 4, 8, 20$

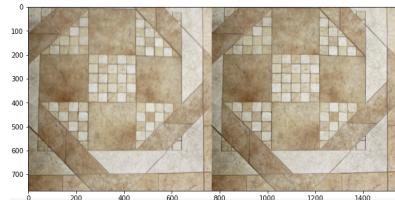


- 由上而下分別為  
 $k = 4, 8, 20$
- 可以看到  $k = 20$  時已經開始出現 outlier。如最下面的圖中亮綠色往右下的線條。



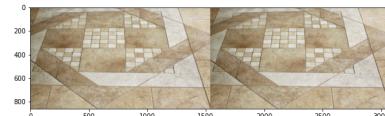
## Compare the errors:

	1-0 & 1-1.png		1-0 & 1-2.png	
k vs method	DLT	normalized DLT	DLT	normalized DLT
4	7028.8651	7028.8651	8.5157	8.5157
8	1.6692	1.3124	2.9786	3.3927
20	0.5074	0.4914	1048.3830	1130.1955



1. 圖為 1-0 與 1-1 的 warping, 於  
 $k = 20$

2. 右側為 ground truth



1. 圖為 1-0 與 1-2 的 warping, 於  $k = 8$

2. 右側為 ground truth

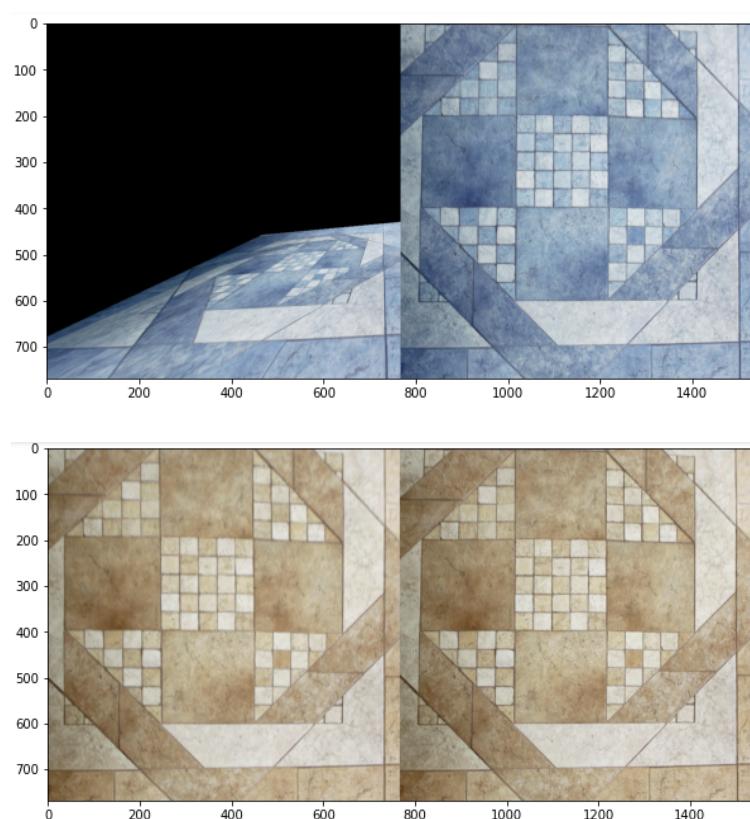
## Discussion

(interesting finding, difficulties you encountered, insights you observe)

1. 將 projective geometry 的  $(x, y, 1)$  經過 homography transformation 之後，產生的點可能為  $(x', y', c)$ 。必須要將點投影回  $(x'/c, y'/c, 1)$  才能得到正確結果。
2. 一般的 affine matrix  $\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$  會先對點進行旋轉與縮放，再進行平移。然而在進行 normalize 時，必須要先平移點集使其中心落到原點，再縮放使其平均距離為  $\sqrt{2}$ 。若要以矩陣運算來完成此操作，可以先產生一個 affine matrix，再取反矩陣，由於 affine matrix 的 inverse 會對點集合先平移再縮放旋轉，因此正好可以滿足 normalize 的需求。當然產生 affine matrix 時的 scale 必須與 normalize 需要的 ratio 呈倒數，平移的方向也要相反。
3. 使用較多 local features 時，較容易受到 outliers 影響，故在 1-0 與 1-2 這組測資上  $k = 20$  的表現較差。opencv 內則會以 RANSAC 去除掉 outlier 得到較好的結果。
4. Problem 2 的 warping 中，nearest interpolation 比 bilinear interpolation 的結果差，但執行效率較快。

### (BONUS) Try other methods or tricks that may improve DLT or Normalized DLT

1. opencv 的 sift feature 受到 RGB 與 BGR 順序影響。在讀入 image 之後使用 cv.COLOR\_BGR2RGB 可以順利 transform 測資 2，若沒有使用，測資 2 將會無法順利 transform。此處猜測 opencv 內部對於 BGR convert 到 RGB 可能有對於3個 channel 不對稱的操作。
2. Visualize

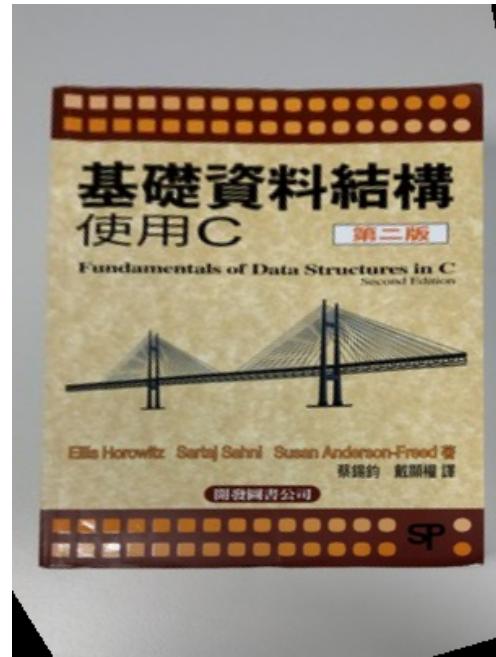
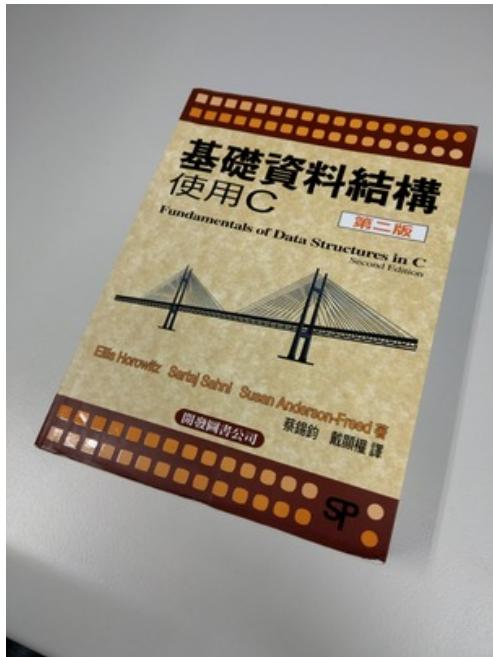


1. 上圖為未經過 cv.COLOR\_BGR2RGB 的 input，透過 matplotlib 來 visualize，因為藍色。
2. 下圖為經過 cv.COLOR\_BGR2RGB 的 input，透過 matplotlib 會呈現原圖的顏色。同時也因為 BGR convert 到 RGB，可以順利地完成 transform。

## Problem 2: Document rectification

### Input document image & Rectified results

- height: 403
- width: 302
- RGB image



**Briefly explain your method (how you choose the corners, warping efficiency)**

1. Manually choose the 4 corners of the book as initial points

The final points would be the 4 vertices of the rectangle composed of leftmost, rightmost, top, bottom edges of the book



## 2. warping efficiency



- nearest interpolate
- time spent
  - nearest interpolate: **0.0148 sec**
  - bilinear interpolate: **0.0874 sec**
  - bicubic interpolate: **0.0089 sec** (opencv may have implemented some speed up)  
(cv2.INTER\_CUBIC)

## how to execute

```
# Problem 1 Homograph Estimation
python 1.py -i1 images/1-0.png -i2 images/1-1.png -g groundtruth_correspondences/correspondence_01.npy --normalize
# Document Rectify
python 2.py --input_img images/smallBook.jpg --method_interpolate bilinear --output_img images/Rectified.png
```

## package

opencv==4.5.1.48

numpy == 1.22.3

## environment

python 3.8.6