# Computer Vision HW3 Report

## 1. Image Alignment with RANSAC

a. SIFT Matching

- Brute Force SIFT Maching

  STEP 1 : 利用SIFT找出img1與img2的keypoints與discriptor

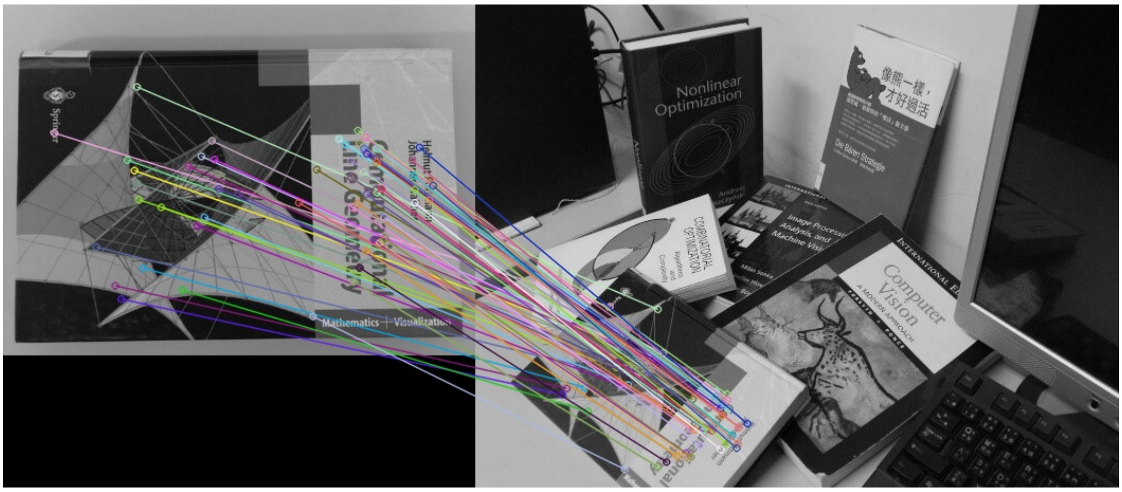
  STEP 2 : BruteForce計算img1與img2中任兩個discriptor之間的L1 Norm(distance)

  - 每次iteration取img1之其中一個discriptor，與img2中所有discriptor計算distance。取最小的兩個distance相除，若相除結果小於某個Threshold，則該次計算的img1之discriptor與img2中和img1 distance最小的keypoint為Match Keypoint。⇒ Neighbor Matching

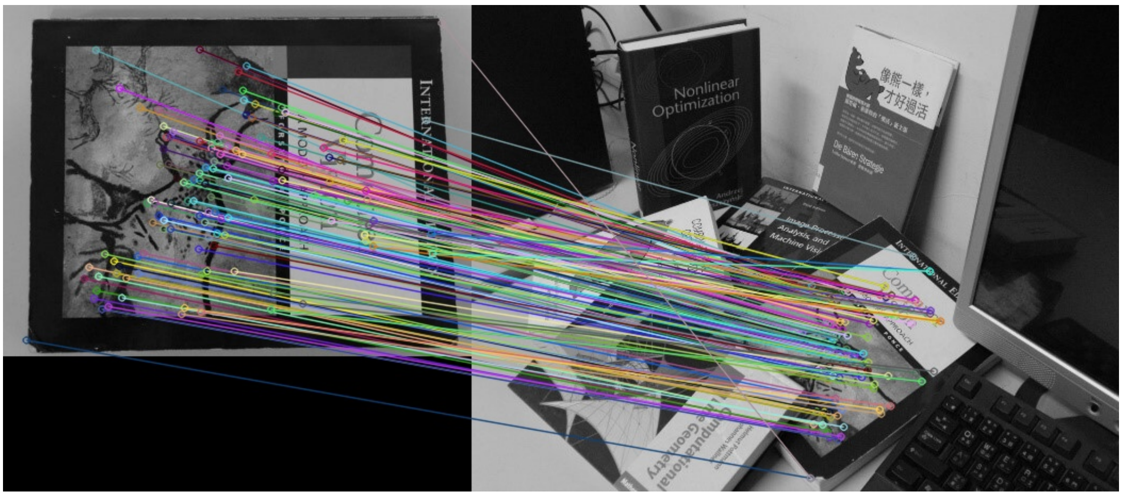
  STEP 3 : 將Match的keypoints紀錄下來，並畫圖

- Result

  Book1



  Book2



Book3

b. Using RANSAC To Find Best Homography

- Calculate Homography

The formula Homography transfomation is p’ = Hp

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Where p is the point before transformation,

p’ is the point after transformation,

H is the Homography Transformation Matrix

Then we can derive the following equation by p’ = Hp

$$\begin{aligned} x_i' &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y_i' &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Now, we will solve the Homography Equation by finding n points in original image, and n corrspondence points in new image

$$\underset{\substack{\mathbf{A}\\ \mathbf{2n \times 9}}}{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix}} \underset{\substack{\mathbf{h}\\ \mathbf{9}}}{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}} = \underset{\substack{\mathbf{0}\\ \mathbf{2n}}}{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}$$

Solving Homography Equation :

Step 1 : finding n points in original image, and n corrspondence points in new image

Note : n points in original image consist of the area we want to rectify in origin image.

Step 2 : Solve h by least square method. ⇒ The approximation solution of h is the one of the EigenVector of A, which correspond to the minimum EigenValue

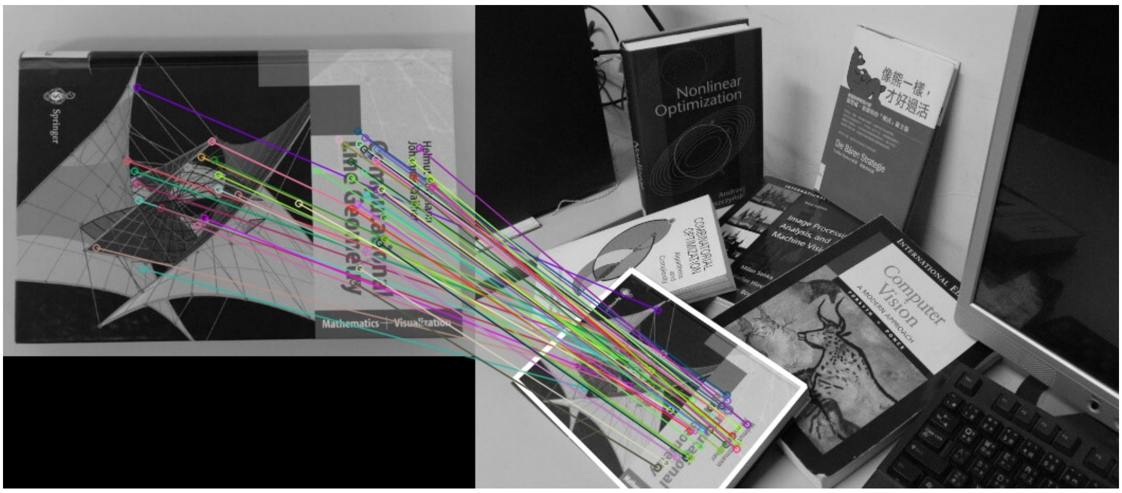
Step 3 : After solving Ah = 0, we get the vector h. By reshaping matrix h, we will get the Homography Transformation Matrix H.

- RANSAC Algorithm

```
REPEAT
  Select random 4 points (from mathicng points in SIFT) to calculate Homography Matrix
  Compute the set of inliers to this model from whole data set
UNTIL number of inlier / number of matching point > Threshold
```

- Result

Book1




Book2


Book3


- Compare the parameter settings in SIFT feature and RANSAC and discuss the result
    - SIFT Threshold : 用以決定Interest Point Brute Force Matching數量
        - 當SIFT Threshold設太高時，會導致有很多mismatch point ⇒ outlier很多 ⇒ RANSAC進行Homography效果差
        - 當SIFT Threshold設太低時，會導致matching point很少 ⇒ RANSAC進行Homography效果差
    - RANSAC Threshold: 用以決定容許計算完Homography以後驗證outlier的比例
        - 當RANSAC Threshold設太高時 ⇒ 容許有很多outlier ⇒ Homography效果差
    - 以Book3為例
        - RANSAC的Thresold為outlier/inlier < 0.12， SIFT的Brute Force Matching Threshold (1.a題Report的STEP2有提到) 為 <0.58 即為Matching。 ⇒ outlier少，因此matching效果佳



- RANSAC的Thresold為outlier/inlier < 0.3， SIFT的Brute Force Matching Threshold為 <0.75 即為Matching ⇒ outlier很多導致Homography被outlier影響


## 2. Image Segmentation

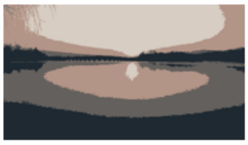
a. Kmeans

- Algorithm

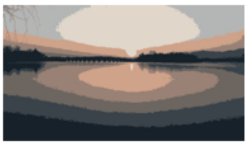
```
Randomly initialize the cluster centers, c1, ..., cK
REPEAT
  Given cluster centers, determine points in each cluster => For each point p, find the closest ci, and Put p int
  Given points in each cluster, update ci to be the mean of all points in cluster i
UNTIL ci not changing
```

- Discuss the difference between the results for different K’s.

  由下圖之結果可得知要將圖片完整地分群K的直不能太小。然而，當K過大時，對clustering不會有太大幫助(無法分出更多cluster)，因此我們需要透過實驗取得適當的K。



k = 4



k = 7

k = 11

K = 4

K = 7

K = 11

b. Kmeans++

- Algorithm

```
Randomly select the first centroid from the data points.
REPEAT k-1 Times
  For each data point compute its distance from the nearest, previously chosen centroid.
  Select the next centroid from the data points such that the probability of choosing a point as centroid is dire
```

- Discuss the difference between Kmeans and Kmeans++

K means易受初始random的中心點影響，而kmeans++解決kmeans的缺點 ⇒ kmeans++的center是以所有點距離最近ceneter中最遠的點做為下一個center

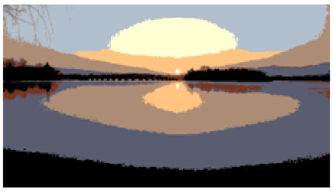
從實作以後output的圖片可以看出kmeans++得到的圖片較kmeans得到的圖篇鮮明

k means

kmeans++

- result

k = 4



k = 7

k = 11

K = 4

K = 7

K = 11

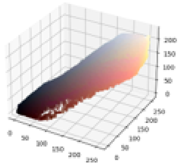
c. MeanShift (Color)

- Algorithm

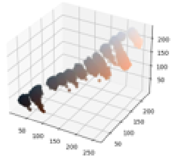
```
REPEAT
  p = random select point from unvisited Points
  centroid =  RGB Value of a p
  mark p as visited point

  REPEAT
    prev_centroid = centroid
    針對所有與centroid之距離小於一個bandwith的點 :
      1. cntroid = 與centroid之距離小於一個bandwith的點之RGB總和之平均
      2. mark這些點為visited point
  UNTIL dis(centroid, prev_centroid) < Threshold

UNTIL All Points are Visited
```

- result :

Bandwidth = 20

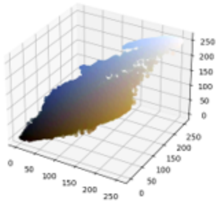


Scatter Plot Before segmentation

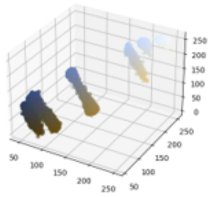


Scatter Plot After segmentation

Bandwidth = 20



Scatter Plot Before segmentation



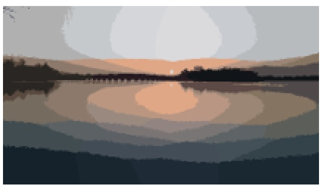
Scatter Plot After segmentation

d. MeanShift (Spatial + Color)

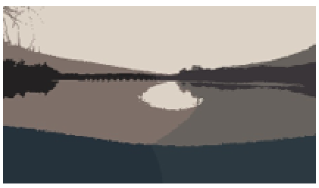
- Algorithm

```
REPEAT
  p = random select point from unvisited Points
  centroid =  RGB Value of a p
  mark p as visited point

  REPEAT
    prev_centroid = centroid
    針對所有與centroid之距離小於一個bandwith的點 :
      1. cntroid = 與centroid之距離小於一個bandwith的點之RGB + (x, y)總和之平均
      2. mark這些點為visited point
  UNTIL dis(centroid, prev_centroid) < Threshold

UNTIL All Points are Visited
```

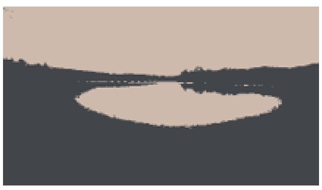
- result



Bandwidth = .15



Bandwidth = .35



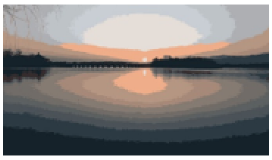
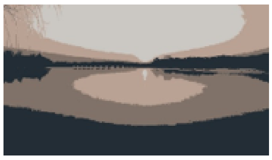
Bandwidth = .50

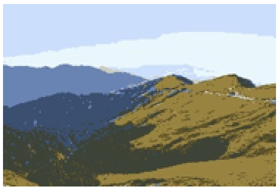
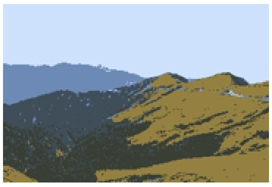
Bandwidth = .15　Bandwidth = .35　Bandwidth = .50

e. MeanShift with different BandWidth

- Discuss the segmentation results for different bandwidth parameters
    - 只使用 Color 進行 Image Segmentation
        - Bandwidth越小 ⇒ Cluster效果越佳

Bandwidth = 20　Bandwidth = 35　Bandwidth = 50

Bandwidth = 20　Bandwidth = 35　Bandwidth = 50

- 使用 Color + Spatial 進行 Image Segmentation
    - Bandwidth越小 ⇒ Cluster效果越佳

Bandwidth = .15↵　Bandwidth = .35↵　Bandwidth = .50↵

Bandwidth = .15　Bandwidth = .35　Bandwidth = .50

- 註 : 由於在計算Color + Spatial之segmentation時，我將RGB與XY的值都normailized到0~1，因此bandwidth之設定會介於0~1之間。

f. Compare the segmentation results by using K-means and mean-shift algorithms and their computational cost.

| | Kmeans | Meanshift |
|---|---|---|
| Computational Cost | O(n^2) | O(n^2) |
| Spatial Information | Cannot consider spatial information during image segmentation | Can consider spatial information during image segmentation |
| Feature Space | Can Only Handle RGB Feature Space | Can Handle Arbitary Feature Space |
| parameter | k (don’t have physical meaning) | bandwidth (has physical meaning) |

- 以下兩張圖可以看出meanshift相較於kmeans可以考慮spatial location進行clustering

k means

meanshift