



HW3 Report

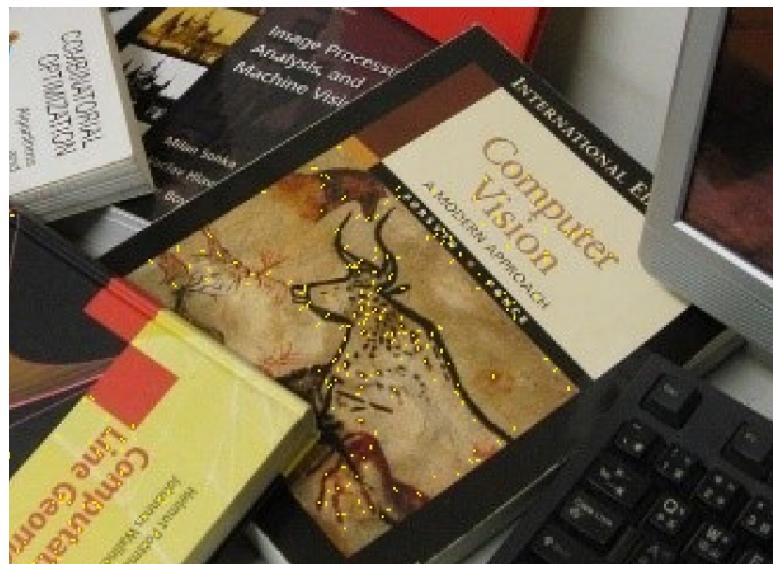
| 111062566 劉緒紳

Part 1 - Implementation

Deviation vectors

Deviation vector 為實際上的點以及透過 homography 轉換完的點之間的差距。

觀察下圖可以發現，deviation vector 的長度很小、不易觀察，這是因為 RANSAC 能將 outlier 有效地去除，而留下的都是 inlier，因此被 homography 轉換完的座標會跟實際上的座標很相近。

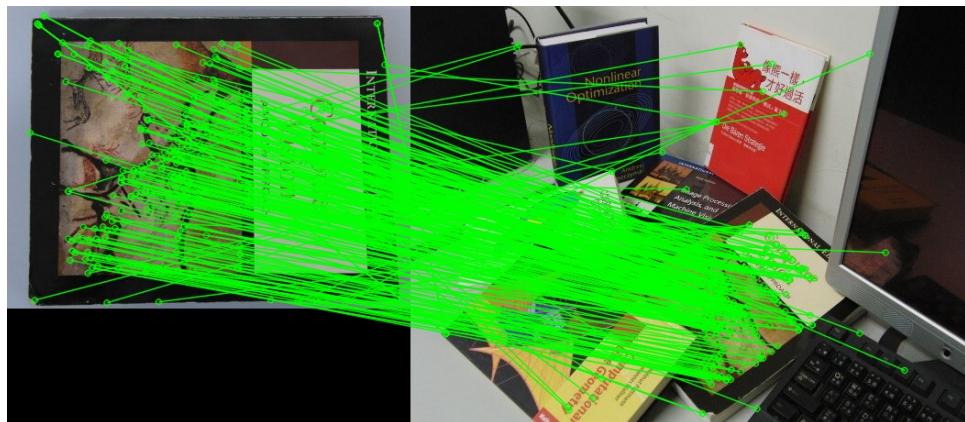


Deviation vectors (截圖)

Comparison between SIFT and RANSAC

首先，單純的 SIFT matching 會有很多 outlier，需要初步的篩選，而我是選用 ratio test，threshold 大約設定在 0.7 至 0.8 之間（每張圖不太一樣），threshold 越低 outlier 會越少（但同時也會剔除一部分的 inlier）。

SIFT + ratio test (threshold = 0.8) 的結果如下所示：

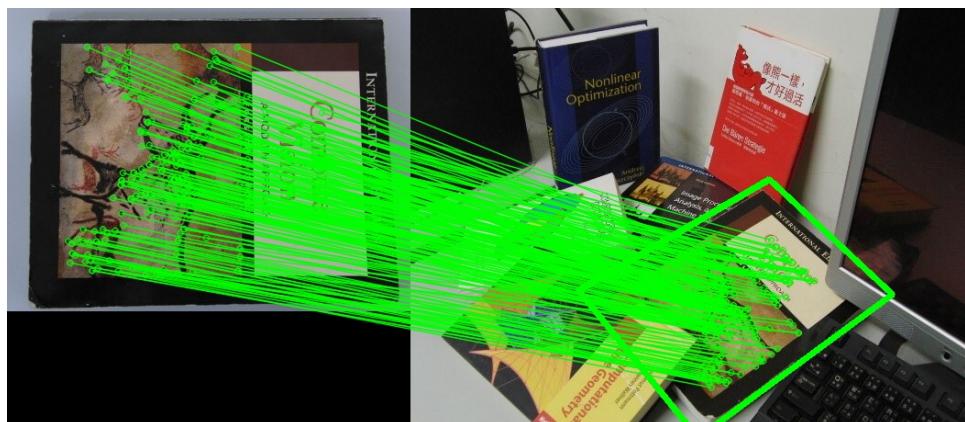


SIFT + ratio test (0.8)

可以看到大部分的 matching 還算正確，但有些誤差較大的 outlier。

而我們可以用 RANSAC 進一步改善這個結果，RANSAC 需要設定一個誤差（也就是 deviation vector 的長度）的容許值，在此容許值內都會被當作 inlier，而誤差容許值越大則 outlier 越多。

以下為 RANSAC 的結果（容許值 = 3）：



RANSAC (error threshold = 3)

Part 2 - Implementation

Comparison between different K's in K-means



k=4



k=7



k=10



以結果來看，當 K 越大時，segmentation 後的顏色會越接近原本的顏色；另外，若以執行時間來比較，則 K 越大執行時間越長。

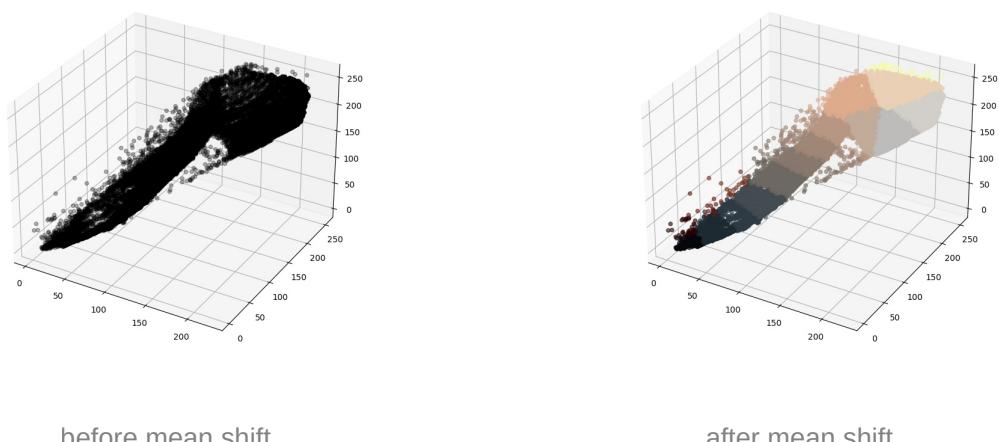
Comparison between K-means and K-means++

K-means 很容易受到初始化時挑選的 cluster center 影響，若挑的點不好，則可能最後會收斂在 local optimum，因此需要重複執行多次 K-means 並選出最好的結果。

而 K-means++ 在初始化時會盡量挑距離目前已經選出的 center 較遠的點，如此一來能有效避免收斂在 local optimum 的情況，不過初始化的時間會較原始的 K-means 久。

另外，在我的實作中，K-means 和 K-means++ 皆有嘗試 50 組不同的起始組合，效果相差不遠。

Pixel distribution before/after Mean Shift



從圖中可以觀察到，在做完 Mean shift 後，在 RGB 空間中相近的 pixel 會被分類到同個 cluster 中。

Comparison between difference bandwidths in Mean Shift



bandwidth=15



bandwidth=30



bandwidth=45



bandwidth=15



bandwidth=30



bandwidth=45

可以發現，bandwidth 越小，segmentation 中包含的顏色就越多，因為 bandwidth 小的 kernel 會考慮的範圍較小，進而較容易分辨靠得很近的 cluster。

Combine the color and spatial information to fit Mean Shift



with spatial information



without spatial information

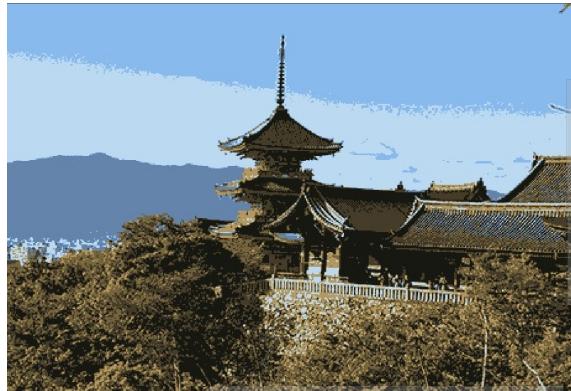
加入了空間資訊的 Mean Shift segmentation 結果中有很明顯的色塊感，原先在沒有空間資訊時屬於同個 cluster 的 pixel，在加入了空間資訊後就被分到了不同的 cluster。

[Bonus] Comparison between K-means and Mean Shift

先從 segmentation 的結果來比較兩種方法的效果：



原圖



K-means



Mean Shift

可以很明顯地發現，Mean Shift 的結果中保留了清水寺的紅色部分，而 K-means 則沒有，這是因為 K-means 會考慮較 global 的資訊，容易受到較大的 cluster 影響而忽略較小的 cluster，例如圖中的紅色；而 Mean Shift 中的 kernel 則能考慮 local 的資訊，可以透過調整 bandwidth 來決定偵測到的 cluster 大小。

接著以時間複雜度來比較兩者。K-means 的時間複雜度為 $O(MNkd)$ ，其中 M 為最多執行幾個 iteration， N 為資料數量， k 為 cluster 數量，而 d 為每筆資料的維度。

而 Mean Shift 的時間複雜度則是 $O(MN^2d)$ ，其中 M 為每個 kernel 最多執行幾次 mean shift iteration，而 N 表示有多少點會作為 kernel 中心執行 mean shift， d 一樣是每筆資料的維度。

可以看到 Mean Shift 的時間明顯較 K-means 高出許多，因此當資料量龐大時，需要使用一些技巧來降低所需要的時間，例如將原圖縮小，或是對原圖隨機採樣出較少量的點做為 kernel，如此一來可以降低複雜度中的 N ，而且最終的結果也不會相差太多。

我在作業中採用的是將原圖縮小的方式，避免隨機採樣時可能會漏掉一些較小的 cluster。