

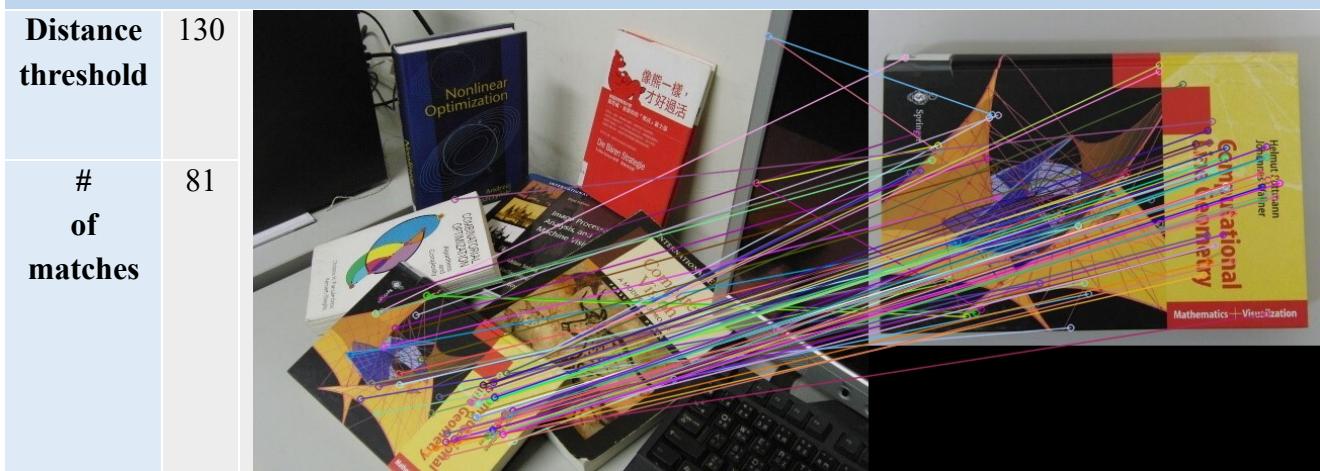
Computer Vision Homework 3

111062543 黃寶萱

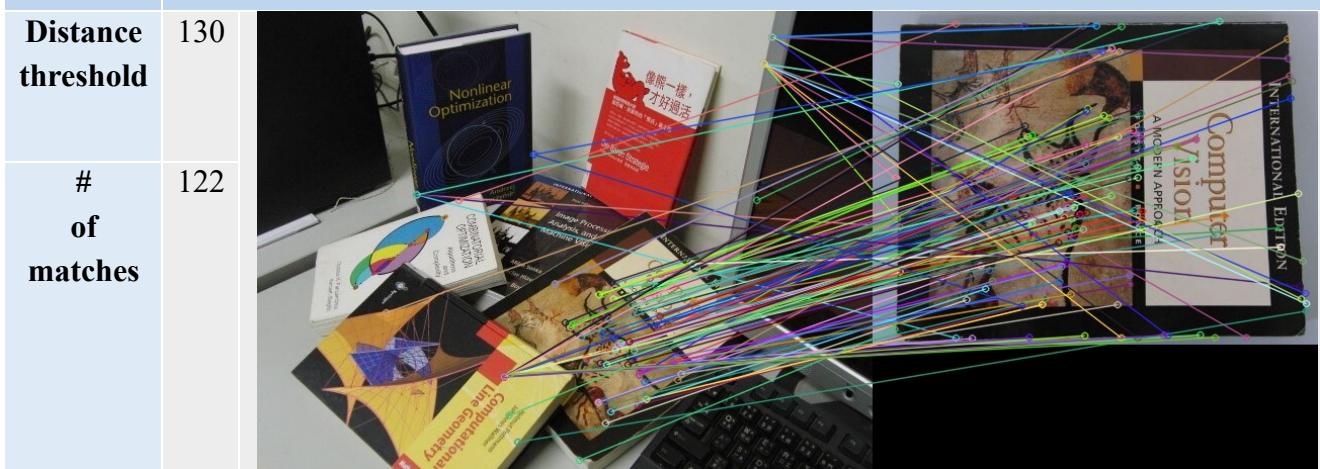
◆ Problem 1: Image Alignment with RANSAC

- SIFT feature matching

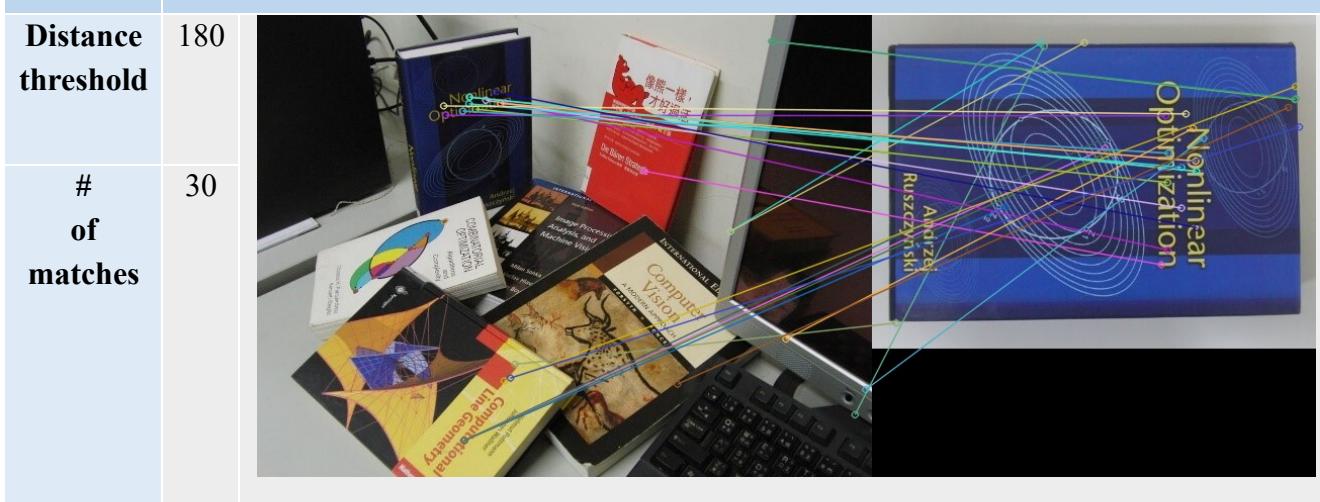
[Book 1] Matching result



[Book 2] Matching result



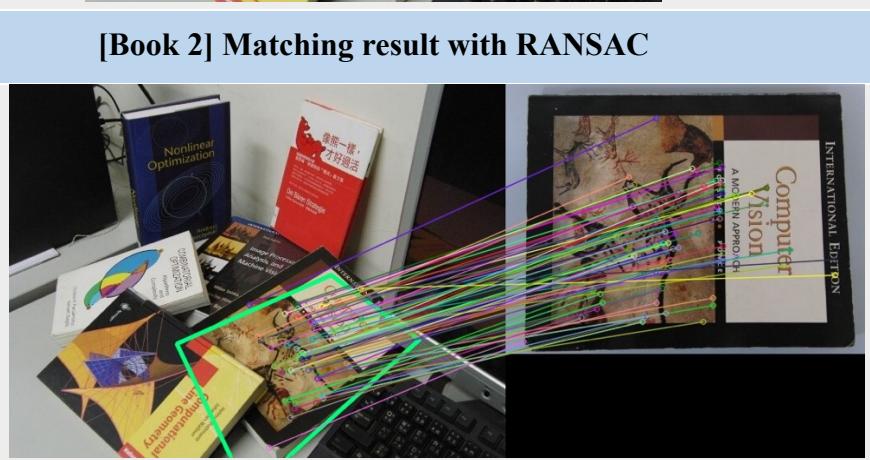
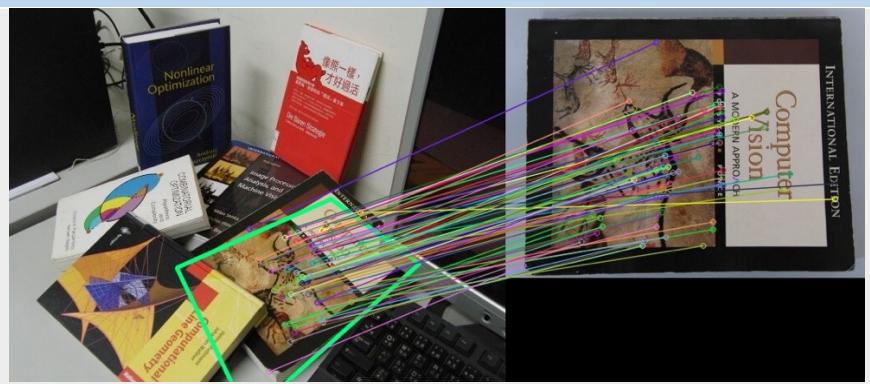
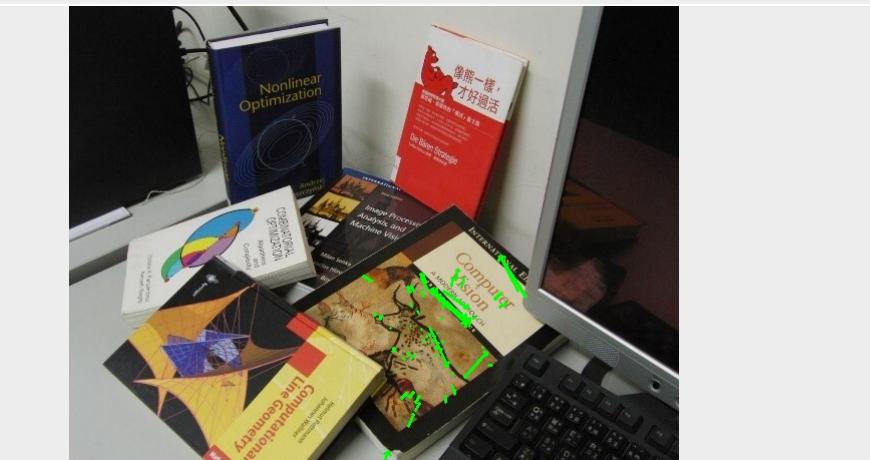
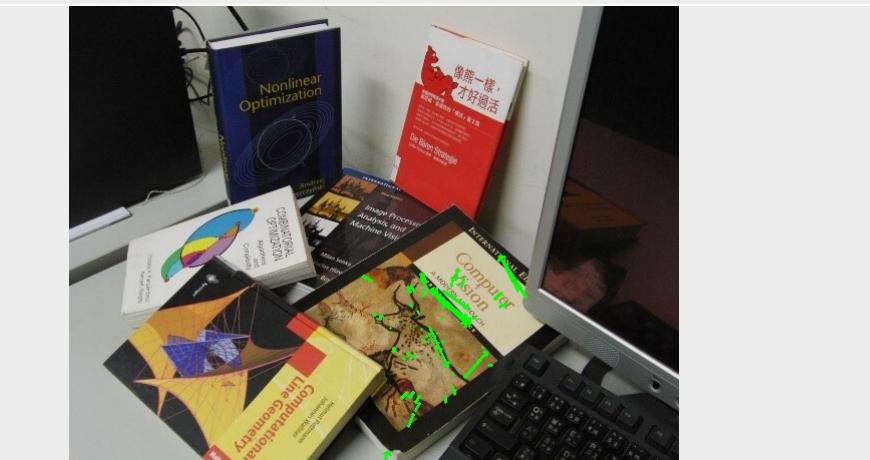
[Book 3] Matching result



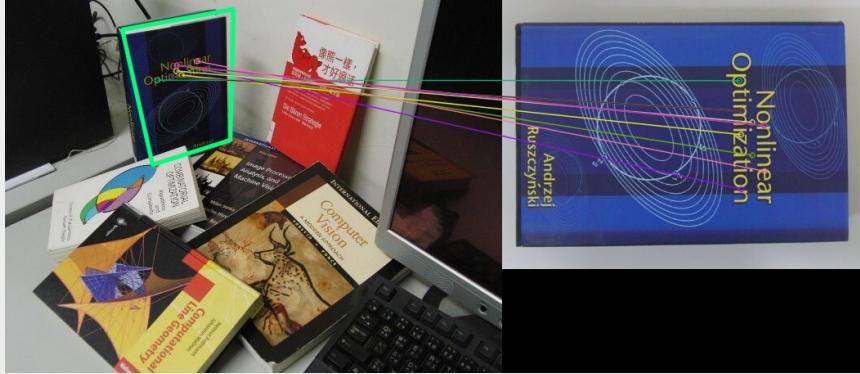
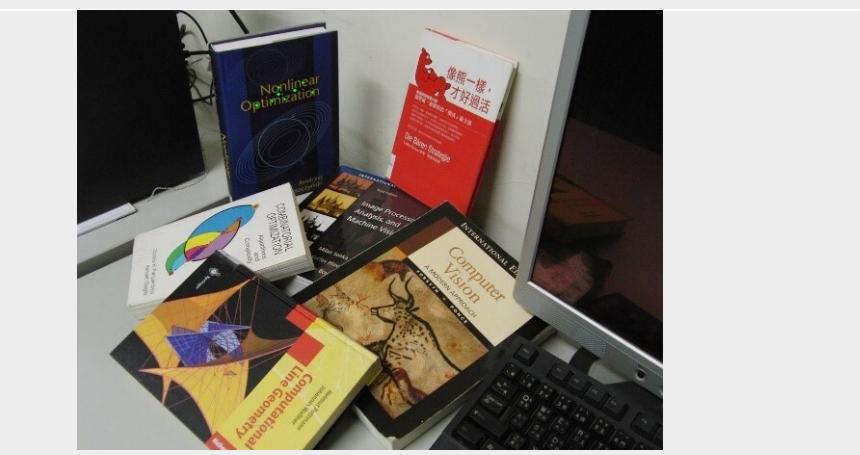
■ RANSAC algorithm

[Book 1] Matching result with RANSAC			
error threshold	1.0		
# of inliers	54		
deviation vector result			
[Book 1] Matching result with RANSAC			
error threshold	25.0		
# of inliers	61		
deviation vector result			

[Book 2] Matching result with RANSAC

error threshold	1.0	
# of inliers	11	
deviation vector result		
[Book 2] Matching result with RANSAC		
error threshold	50.0	
# of inliers	76	
deviation vector result		

[Book 3] Matching result with RANSAC

error threshold	1.0	
# of inliers	11	
deviation vector result		

[Book 3] Matching result with RANSAC

error threshold	15.0	
# of inliers	14	
deviation vector result		

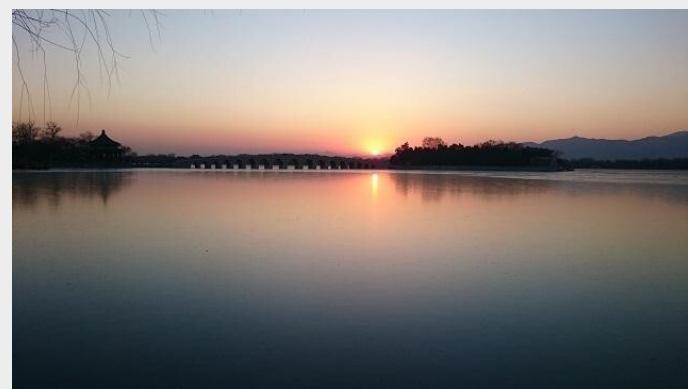
Discussion:

在 SIFT feature matching 的部分，分別用不同的 distance threshold 篩選 matching correspondences 的數量，threshold 越大可以保留越多的 correspondences，但同時也有較多的 miss matching，由於討論區上有提到不一定要超過 500 個 correspondences，因此為了在後續 RANSAC algorithm 中能有較好的 performance 以及縮短計算時間，我利用 threshold 分別在 Book1, Book2, Book3 保留 81, 122, 30 個 correspondences，每張圖皆有 matching 以及 miss matching 的情況，因而可以在後續的 RANSAC 中看出使用 RANSAC 做 feature matching 的差異。在 Book3 只保留 30 個 correspondences 是因為經過多次的 threshold 測試，會發現 threshold 越大反而得到越多的 miss matching，好的 matching 數量並未增加，導致後續無法找到好的 homography matrix，因此最終只保留 30 個 correspondences。

在實作 RANSAC algorithm 的部分，三張圖統一都是執行 2000 個 iterations，並保留有最多 inliers 的結果。在每個 iteration 中，隨機選擇四對 correspondences 得出 homography matrix H，並利用 matrix H 將書本上的 correspondent feature point 轉到另一張圖片上，計算 transformed 的點與原本對應點的距離，若距離小於 threshold 則為 inlier。Threshold 越大代表 miss matching 有可能被視為 inlier，並且得到的 homography matrix 較不精準，可以從 bounding box 以及 deviation vector 看出當 threshold 越大書本的 bounding box 無法準確的匡住書本，deviation vector 的長度也會較大，因而影響的 RANSAC algorithm 的 performance，反之，較小的 threshold 則可以有效地避免掉 miss matching 並且可以有很好的 object detection performance。

◆ Problem 2: Image Segmentation

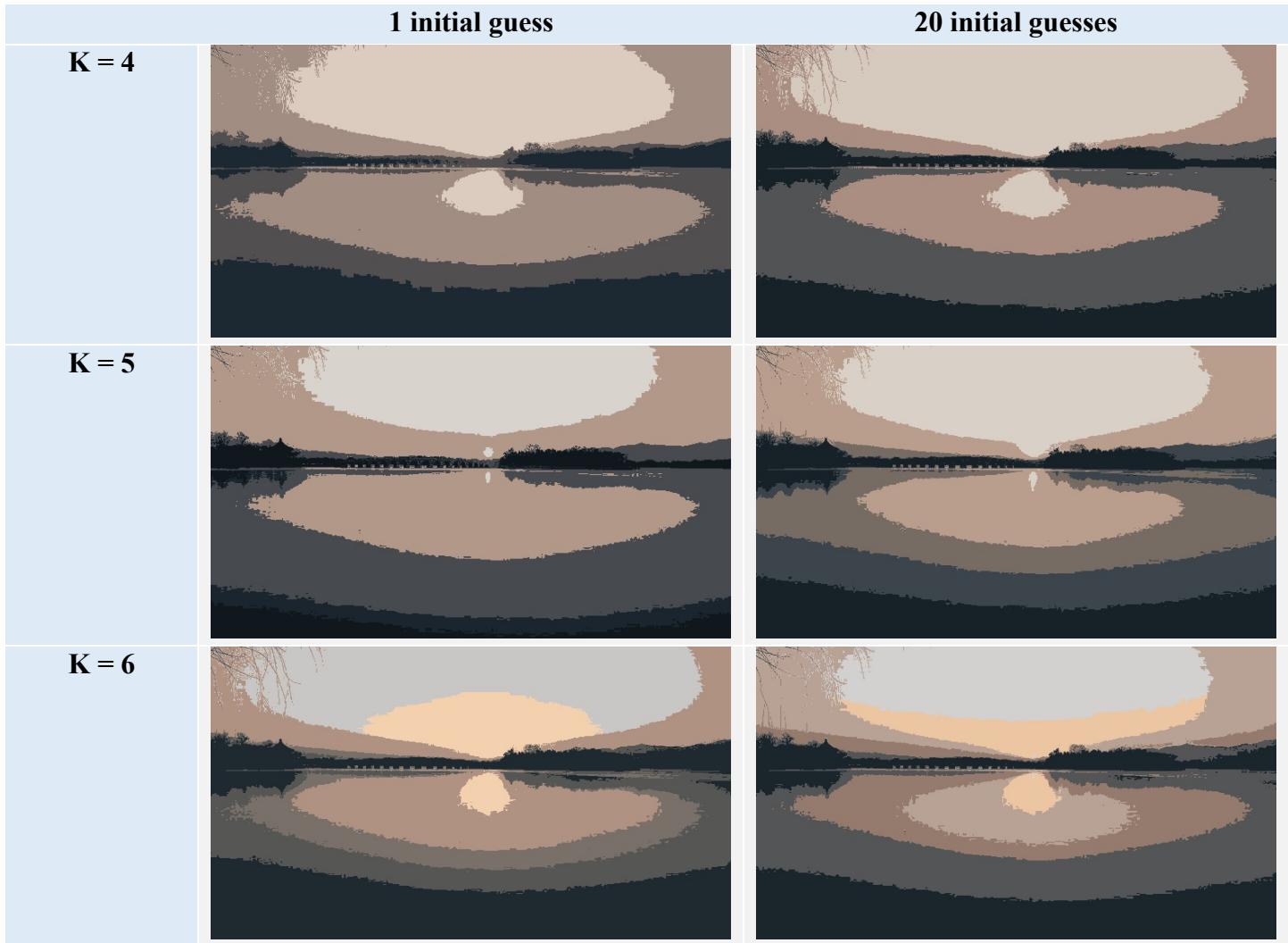
2-image.jpg



2-masterpiece.jpg



■ A: K-Means





Discussion:

1. Different values of K

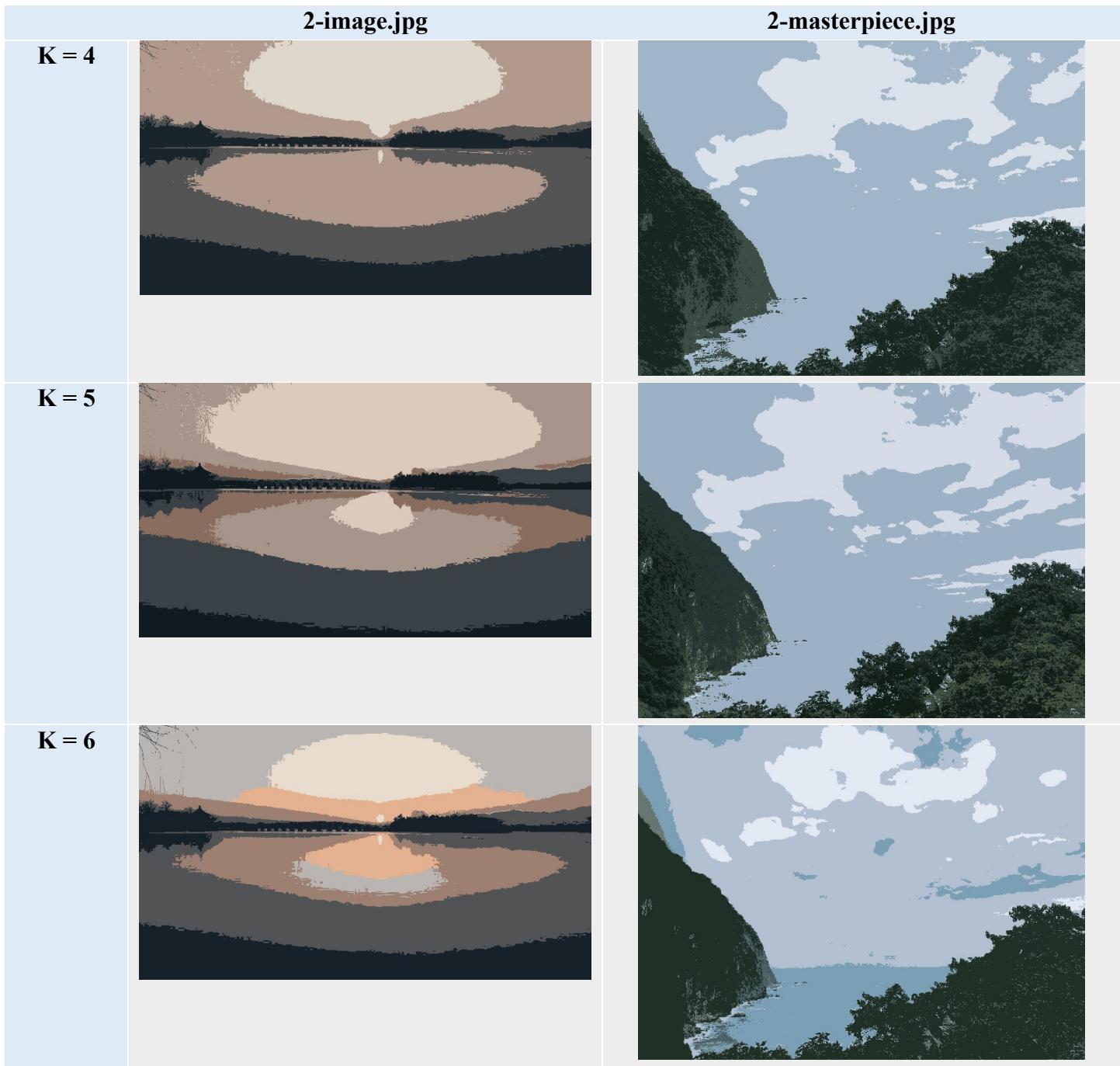
從上表中可以看出，不論是 1 initial guess 或 20 initial guesses 的 segmentation 結果，當 K 值越大(K=6)可以有更好的 segmentation 結果，在 2-image.jpg 中可以看出當 K=6 便可以判斷出天空中有夕陽的顏色，而不是將整片天空歸類成同一個 cluster；

在 2-masterpiece.jpg 中可以看出當 K=6 時山的部分會更立體，因為可以將山分成不同的 cluster，而不是整座山都是同一個顏色。

2. 1 initial guess vs. 20 initial guesses

基於 K-Means algorithm 是 random 選擇 K 個 initial centers，segmentation 的結果容易受到 initial centers 好壞的影響，因此這部分我嘗試讓他重複做 20 次 initial guesses，並計算每次最後 cluster 結果與 center 之間的距離統計，保留最好的結果(distance 總和最小)，可以明顯看出重複執行 20 次的 segmentation 結果有較好的表現，特別是 2-materpiece.jpg 中大海的部分。

■ B: K-Means++



Discussion:

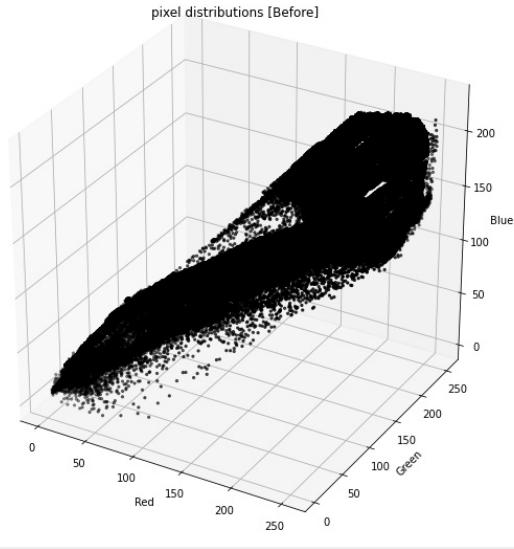
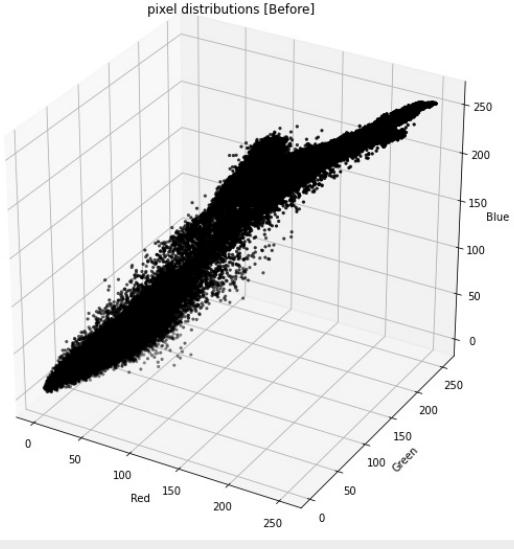
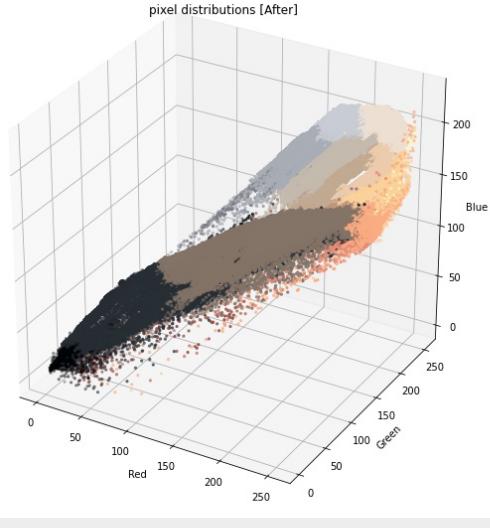
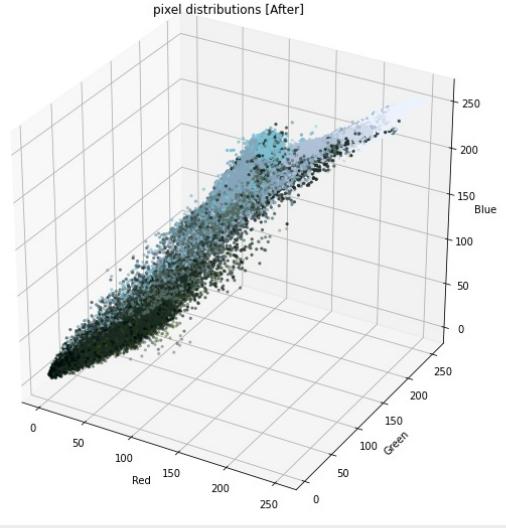
1. Compared with 1 initial guess

當 K 較小時(K=4)，2-image.jpg 的 segmentation 結果差異不大，但在 2-masterpiece.jpg 中則較可以看出 K-Means++會有較好的 segmentation 表現（尤其是山的部分），因為 K-Means++可以藉由計算 distance 選擇 K 個 initial centers，不是 random 選擇。

2. Compared with 20 initial guesses

整理而言，不論是 2-image.jpg 或 2-masterpiece.jpg 使用 K-Means++與使用 K-Means (20 initial guesses) 的 segmentation 結果差異不大，因為在 20 次的 iteration 過程中已保留最好的 segmentation 結果，也代表 K-Means++選擇 initial centers 的做法能有效提升 segmentation 結果。

■ C: Mean Shift algorithm (RGB feature space)

	2-image.jpg	2-masterpiece.jpg
Segmented result		
Pixel distribution [before]		
Pixel distribution [after]		
Image size	528 x 300	500 x 375
# of clusters	32	29
kernel	Gaussian Kernel with bandwidth = 3	

■ D: Mean Shift algorithm (RGB & spatial information)

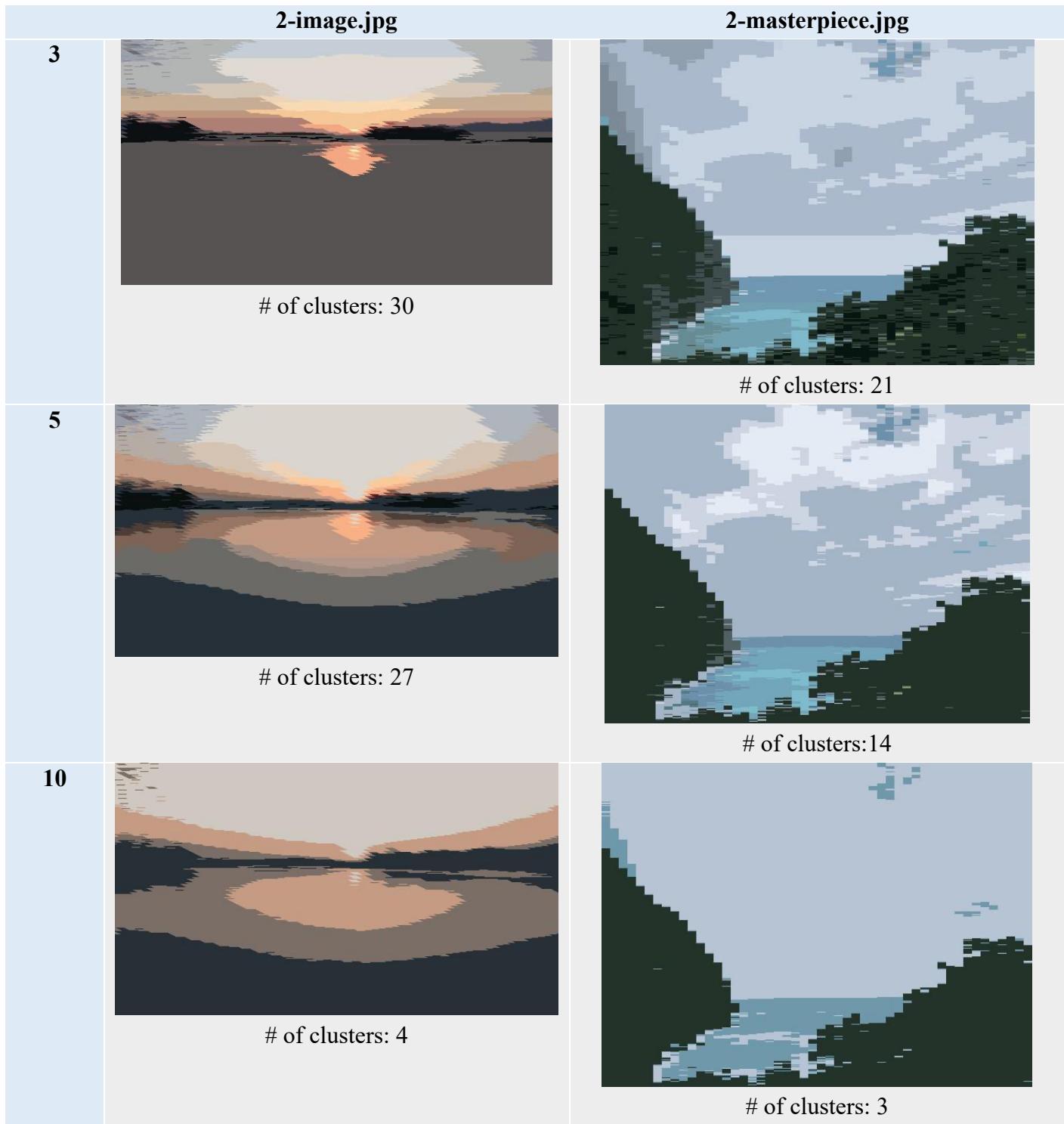
	2-image.jpg	2-masterpiece.jpg
result		
Image size	528 x 300	500 x 375
# of clusters	170	73
kernel	Gaussian Kernel with bandwidth = 3	

Discussion:

在 Question(C)與 Question(D)中同樣都是使用 Gaussian kernel (bandwidth=3)來做 Mean-Shift clustering 運算；(GROUP_DISTANCE_THRESHOLD=20)計算每個點與 cluster center 的距離，若該點與目前的 cluster center 距離皆大於 20 則將該點加入新的 cluster；(MIN_DISTANCE=0.001)計算點每次 shift 的距離，若該點移動的距離小於 0.001 則下一個 iteration 不會再做該點的 cluster 運算。

經過測試後，若使用原圖大小的每個 pixel 去做 clustering 執行時間會超過 5 個小時且仍未得出 segmentation 結果，因此為了縮短程式執行時間，先利用 cv2.resize() 將圖片縮小:2-image.jpg ($640 \times 360 \rightarrow 528 \times 300$), 2-masterpiece.jpg ($640 \times 480 \rightarrow 500 \times 375$)，並每間隔 10 個 pixel 才做運算，可以將執行時間降低至大約 3 個小時，但因為合併 10 個 pixel 導致 segmentation 的圖片結果會出現類似一格一格的現象，但依舊可以看出 segmentation 的結果，這部分還可以再嘗試其他的優化方式，使程式執行時間可以有效地縮短並得到更好的 segmentation image。也嘗試過直接將原圖縮小 16 分之一（長寬各縮短為原本的 $1/4$ ），如此便不用合併 pixels 也可以在合理的時間內得到結果，但因為 image 實在太小了導致看不清楚結果，因此最終選擇以目前的方式做 Mean-Shift 運算。

■ E: Mean-Shift algorithm with different bandwidth value



Discussion:

由於在 Question(C)(D)的執行時間依舊有點長，Question(E)我將變數 MIN_DISTANCE 統一改為 0.01 並用不同的 kernel bandwidth 做運算，可以看出當 kernel bandwidth 較大時他的分類結果會分成較少的 clusters，就 2-image.jpg 而言可以分成較少的 cluster 得出好的 segmentation 結果，但在 2-masterpiece.jpg 中則需要分成較多類別在山與天空的部分才可以更符合原圖的結果。