Assignment 4: Image Segmentation

107062134 樊明膀

1. Explanation of how to use the function scipy.sparse.linalg.eig

   首先去找一下這個 function 怎麼 call

   scipy.sparse.linalg.**eigs**(*A, k=6, M=None, sigma=None, which='LM', v0=None, ncv=None, maxiter=None, tol=0, return_eigenvectors=True, Minv=None, OPinv=None, OPpart=None*)¶

   https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.eigs.html

   下面有解釋要如何去 call 每一個 data 代表什麼意義，我主要會 focus 在 Sparse array A:可以直接丟 COO format，以下這邊都可以用

   Sparse matrix classes

bsr_matrix(arg1[, shape, dtype, copy, blocksize])	Block Sparse Row matrix
coo_matrix(arg1[, shape, dtype, copy])	A sparse matrix in COOrdinate format.
csc_matrix(arg1[, shape, dtype, copy])	Compressed Sparse Column matrix
csr_matrix(arg1[, shape, dtype, copy])	Compressed Sparse Row matrix
dia_matrix(arg1[, shape, dtype, copy])	Sparse matrix with DIAgonal storage
dok_matrix(arg1[, shape, dtype, copy])	Dictionary Of Keys based sparse matrix.
lil_matrix(arg1[, shape, dtype, copy])	Row-based list of lists sparse matrix
spmatrix([maxprint])	This class provides a base class for all sparse matrices.

   the number of result ‘k’

   Which 有很多不同的

   我主要是比較 SR and SM，後來選擇了 SR

   `'SR'` : Eigenvalues with smallest real part (`eigs`).

   `'SM'` : Eigenvalues with smallest magnitude (`eigs`, `eigsh`), that is, smallest eigenvalues in the euclidean norm of complex numbers.

   tol 可以稍微調一點，這樣才不會跑那麼久

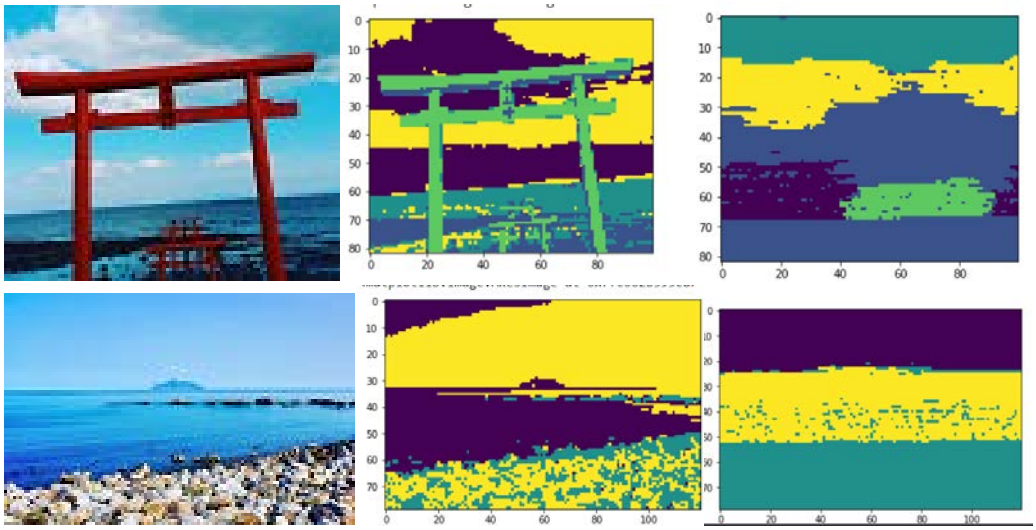
2. Explanation of how K-means algorithm works.

   k-means，這邊的 k 是你想分成幾群，means 就是每一群群心

   1. 我們先設定好要分成多少(k)群。
   2. 然後在 feature space 隨機給 k 個群心。
   3. 每個資料都會所有 k 個群心算距離(多數看到為:歐基李德距離 Euclidean distance)。
   4. 將每筆資料分類判給距離最近的那個群心。
   5. 每個群心內都會有被分類過來的資料，用這些資料更新一次新的群心。
   6. 一直重複 3 – 5，直到所有群心不在有太大的變動(收斂)，結束

3. Your original images and the figures after running two segmentation methods.

左邊是 原圖 中間是RGB KMeans 右邊是Spectral Clustering



4. Discussion about what kinds of images prefer which segmentation methods.

上面的話比較偏好是 RGB KMeans 才可以看出鳥居的形狀

推斷說 比較精細的部分 或是顏色分的明顯的 可以用 RGB 的方式

下面的話 覺得說 RGB 不差 但右邊的有比較明顯的分層 岩石 海 跟 天空 這樣

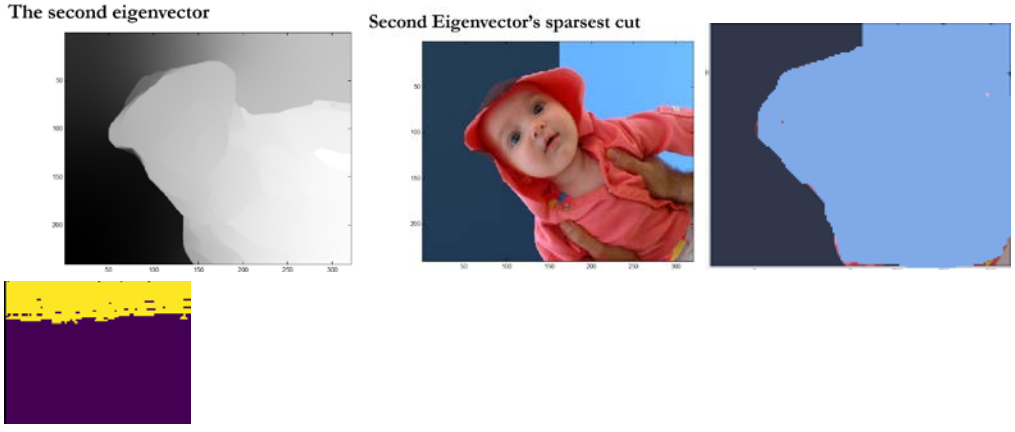
覺得比較大塊一點的分層 然後可能比較雜亂的話 可以用 Spectral cluster 的方式，但是這個方法的話 有很多細節要抓，可能還要先看哪幾層的 cut 比較符合效果，下一部分會介紹。

5. Figures of segmentation results for different number of eigenvectors and clusters and Discussion about the influence of different number of eigenvectors and clusters.

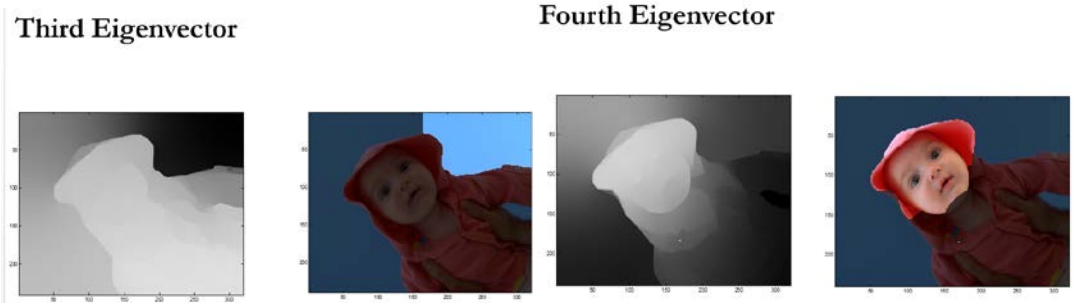
這個我還是拿桃子的圖比較適合 然後參考過 PPT 上面的方式

下面這個是取第二的 eig 可以看到說 比較黑的一定程度的就會被分開來 中間的圖算是先把它分好幾個顏色 但可以看出左跟右 我估計這張圖直接跑我們的那個算法用 cluster=2 會出現右邊的圖，寶寶跟右邊同色 或是 跟左邊深色同色，但很誇張的是他跑出的是

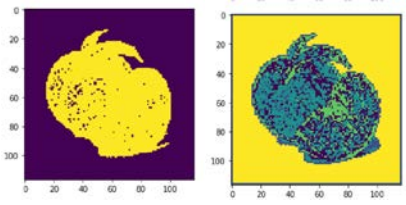
下面的結果(黃色紫色的)，我猜測這估計還是要跑了才知道



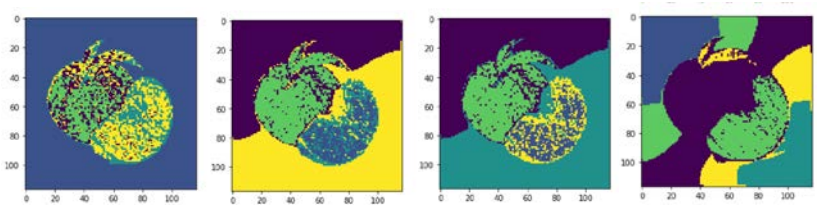
接下來分別是



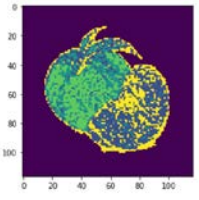
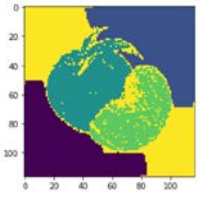
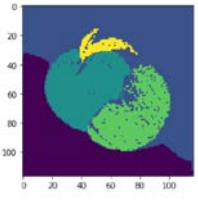
我個人會猜說 這個把臉 衣服 跟背景話分開來這這樣 如果取 2,3,4 下去做
接下來我會拿原本的 sample 做說明
First eigenvector 就可以看出很好的效果有把桃子跟 背景分出來，但如果把顏色提上去的話是沒有明顯的多色分層



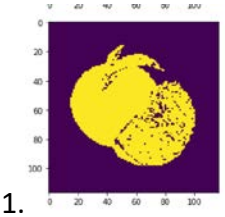
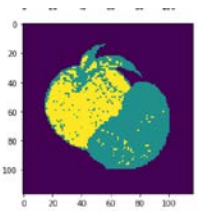
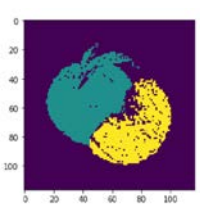
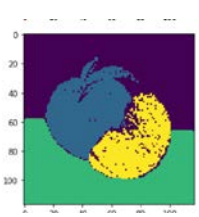
Second eigenvector 可以看到說 切片 跟完整一顆的分層，右邊一個是 third eigenvector 沒啥變化但是後面背景被切一半，再右一個 fourth 沒什麼差別，當做到 fifth cut 的時候就又出現了明顯的變化，而我要的最主要是說 葉子的那部分的變化，因此我可以直接說，如果有要葉子的出現的話，我的 eigenvector 要取到第五個。



接下來我做取 1:2, 2:4, 2:5 這個取這幾個的比較，可以看出說第一個 因為 first 跟 second，而 second 將切片的部分變明顯了(雖然只有取二也是差不多的效果)，第二個是 2 到 4，出現了果子跟切片明顯的分層 我覺得是因為 2,3,4 都有這個部分，然後背景開始被切割
第三個就是可以看出說，我的葉子出現了跟我上面所做的 eigenvector 要取到五才可以看出來一樣，但其實這個蠻容易有偏差的，因為 k_means 本來就沒有確定的分法，所以在歸類的時候。可能每次都不太一樣 如果不是很明顯的不同的話

1. 
2. 
3. 

下面圖這邊我做取 clusters 的變化，第一圖是取 vector 1:2 ,n_cluster 取 2 可以看到說 桃子跟背景分兩部分，第二圖 vector 沒變 cluster 取 3 就有切片的顏色，第三是取 vector 2:4 cluster 取 3 想跟上面圖的 2(cluster=5)比較說如果取 3 個色就可以拿掉 背景色
第四是取 vector 2:5 cluster=4 可以看到說 背景的分層先出來了 ，但其實這個做的偏差蠻大的跟我上面對圖 3 所說的一樣。

1. 
2. 
3. 
4. 

6. Discussion about the COO format.

COO format ((i,j),data)大致上是這樣的紀錄方式，而 code 中的就是用下面的這個方式 去創建的

```
sp.coo_matrix((lv, (li,lj)), shape=(N, N))
```

coo_matrix((data, (i, j)), [shape=(M, N)])

to construct from three arrays:

1. data[:] the entries of the matrix, in any order
2. i[:] the row indices of the matrix entries
3. j[:] the column indices of the matrix entries

Storage: 如果說是要像 rbg 要存 3 個資料的話 COO 總共需要五個空間，對於一個二維 data (x,y,r,g,b) 如果存滿一個 n*n 的矩陣 會需要 n*n*5 的 storage 而二維的話只要 n*n，但如果超過 4/5 的數字都是零的情況下，用 COO 就會比較省空間。
而 sparse matrix 就是大多數都是零的情況下 因此 COO 會比較省 storage。

Computation: 計算的話二維的要 n 才得到一個乘法資料，整張 n*n 需要時間複雜度 n^3 的複雜度，才可以算完。
如果用 COO 的情況下，一個資料最差也是 n，也是要 n^3 才可以解決，甚至在計算上可能會因為比較長的串裡面找到你要的行列相乘反而更是麻煩，如果對 w*h 長度的做直線搜索的話，會更浪費時間，但如果先對 row 做好排序，在對 col 做好排序的話，這樣乘法可以做到上面 n^3 的方式，相較之下，兩個排序的時間就微不足道了，而大多數是 0 的情況下，就更會有優勢，因為可以省掉很多項不用做。

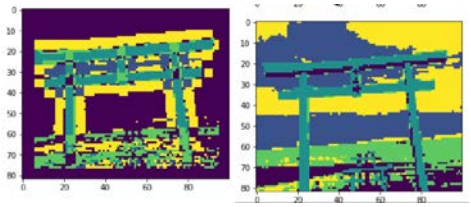
7. (option) Result and discussion on the bonus question.

我有設計一個加強圖中物件的分隔的方式，就是把邊的部分特別的挑出來

我做的是將我的圖的 pixel 做處理跟附近的點做平均的值，相減再做 kmeans ，我覺得這樣比較不會有整個圖混在一起的狀況。

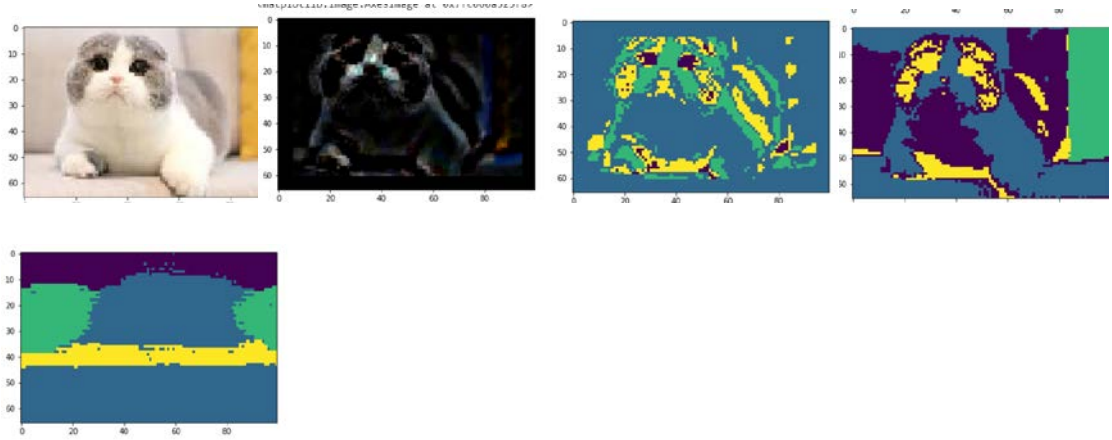
kmean 出來跟 RGB 的方式有點像，但有時候細節會比較多，或是說有把東西特別的挑出來，並把背景給去掉，但如果圖片太複雜的話可能會不太適合。

下圖這個鳥居的就是把鳥居給獨立出來了，左圖是我的算法 右是 RGB

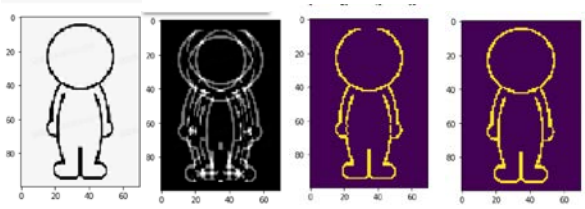


貓的話，最左是原圖，第二張經過我處理，第三張 是跑我的 kmeans ，第四張是 RGB，第五張是 Spectral cluster (直接爆掉)

我個人是覺得說我的貓保留了更多的細節 就算是兩個顏色的話也比較可以看得出來，而且不會跟背景混在一起。



這個是顯示一個邊線的圖順序一樣，左二跟左一差不多，可以看到就算我轉過後有點混雜，但 kmeans 出來的方式其實還可以



這是裝置藝術的那個香蕉，可以看到說我做出來的那個有把背景給分出來，但 RGB 的反而就整個抓出來了，然後 spectral 還是爆掉了