**Assignment 4**

107062208 邱靖豪

1. **How to use the function “scipy.sparse.linalg.eigs.”?**

這個function的功能主要是用來算放進去的矩陣的eigenvectors跟對應的eigenvalues，以我的code當範例，spla是先從scipy.sparse.linalg引入的，接著，就使用.eigs的功能，L在這裡是那個稀疏矩陣，k = m，k是希望得到的eigenvectors數，而which = ‘SR’指的是找到前k小，同時也是實數的eigenvectors。而等號左邊的vals, vecs，指的是得到的eigenvalues跟eigenvectors。

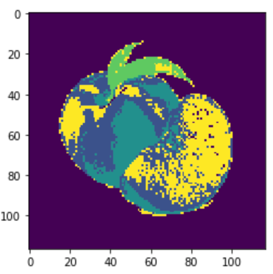
1. **How K-means algorithm works?**

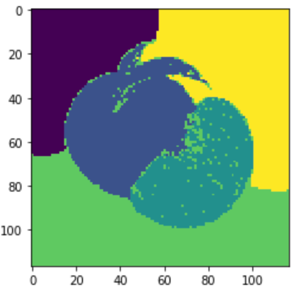
K-means algorithm是一種演算法，目標是將n個待觀察物分類成k個群集，K-Means是聚類演算法中的最常用的一種，演算法最大的特點是簡單，好理解，運算速度快。他分成三個步驟，首先，**(1) Initialisation**，先隨機舉出k個初始群集中心 (means或centroids)。接著是**(2)Assignment**，以「連結各個觀察物最近的centroids」，創造出k個clusters，方法是計算每筆資料到k個群心的歐式距離(歐基李德距離Euclidean distance)。再來是**(3)Update**，將每筆資料分類給距離最近的那個群心，接著，每個群心內都會有被分類過來的資料，用原本的資料加上被分類過來的資料，更新一次新的群心。反覆以上步驟，直到群心沒有太大變動，演算即結束。

1. **Original images and the figures after running two segmentation methods.**

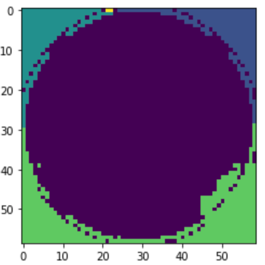
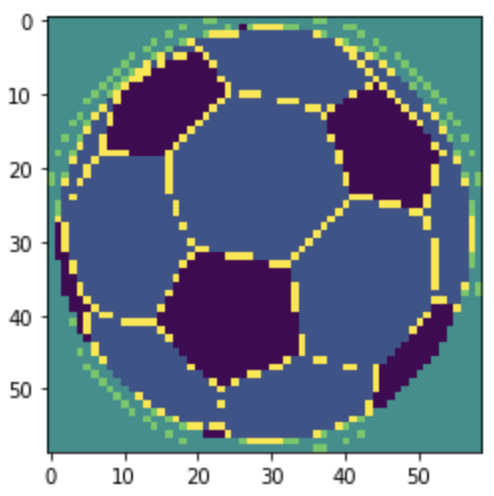
**Original image:**

**Original:** **RGB K-Means:**  **Spectral Clustering:**

****一張含有 水果, 室內 的圖片

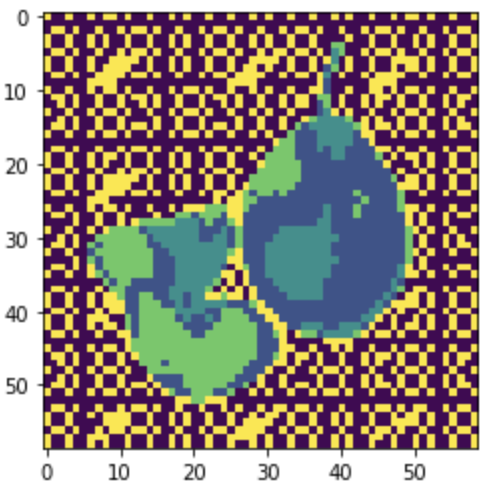
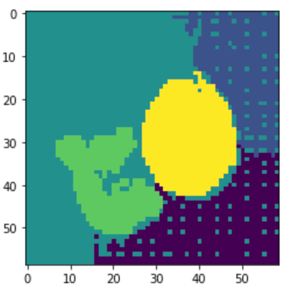
描述是以非常高的可信度產生****

**My image1:**

**Original:** **RGB K-Means:**  **Spectral Clustering:**

**My image2:**

**Original:** **RGB K-Means:**  **Spectral Clustering:**



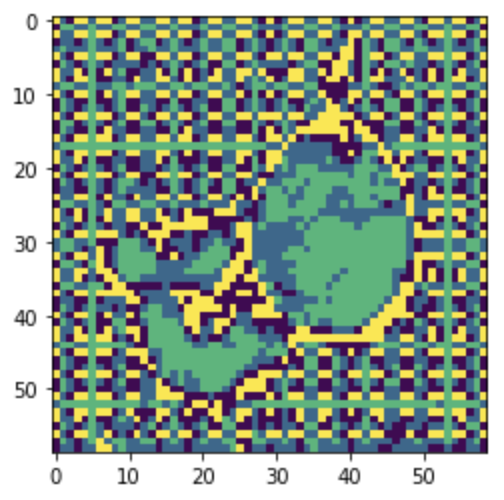
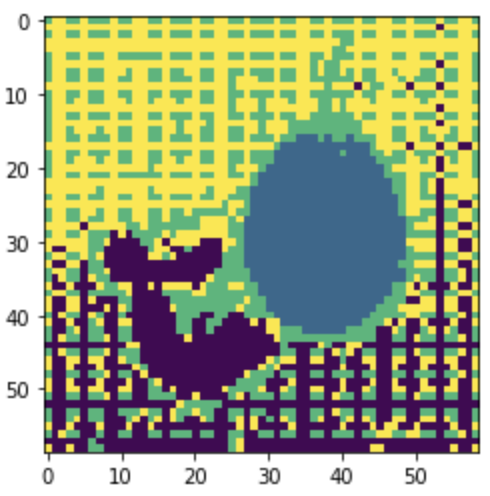
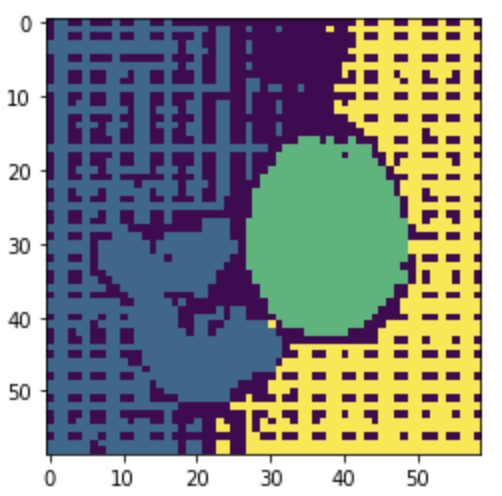
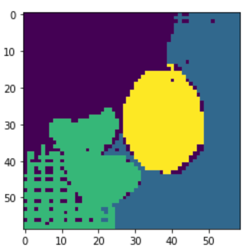
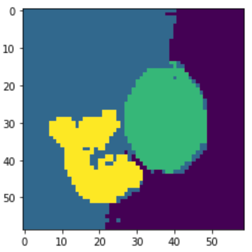
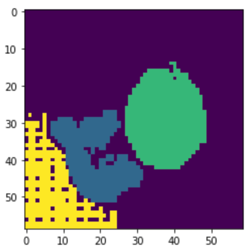
1. **Discussion about what kinds of images prefer which segmentation methods**

從我自己挑的兩張圖看來，只要是圖形顏色單調，或者是形狀沒那麼複雜的，**RGB K-Means**的效果較好，反之，比較複雜的圖案，跟顏色比較有陰影對比的圖片，用**Spectral Clustering**會比較好。

1. **Figures of segmentation results for different number of eigenvectors and clusters**

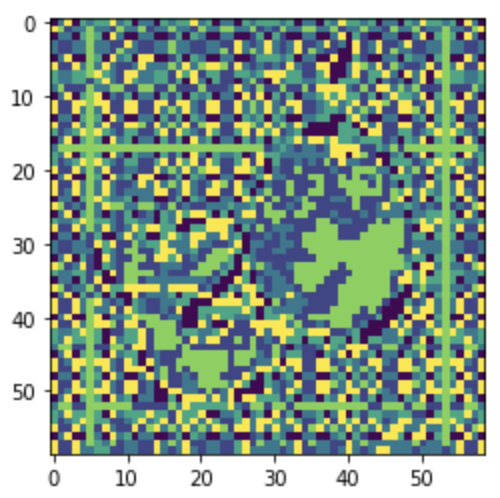
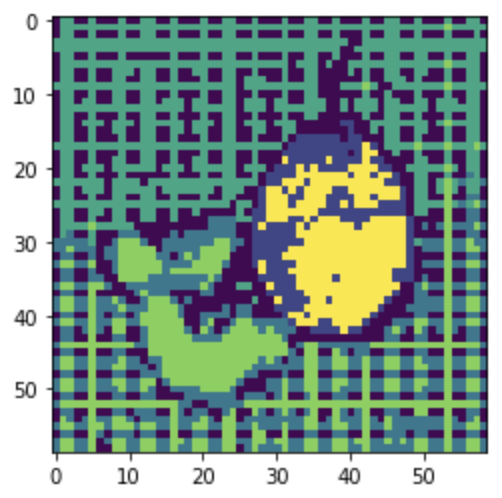
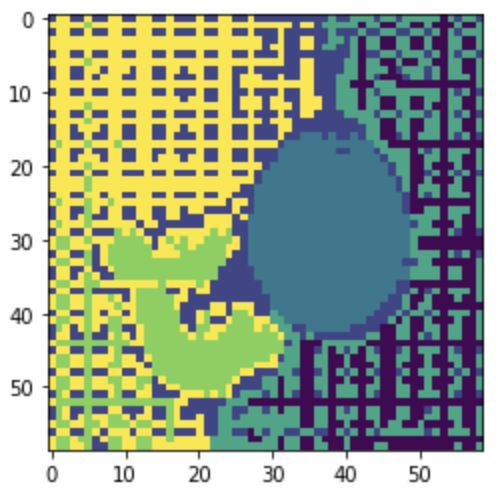
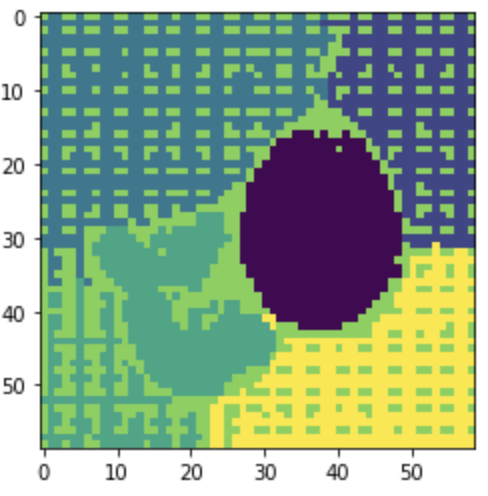
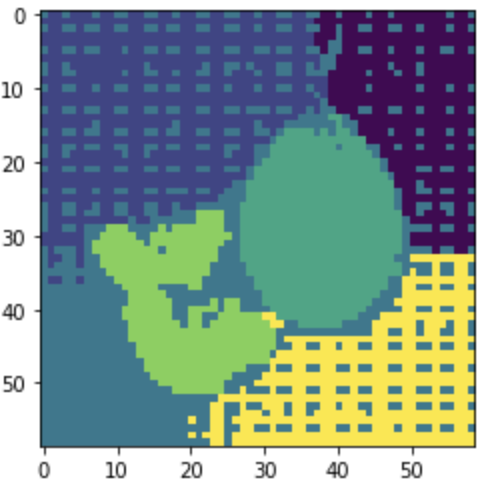
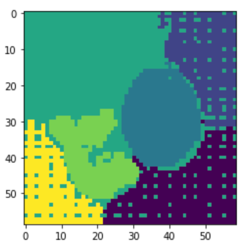
**Different numbers of eigenvectors**

**Cluster = 4**

**1 2 3 4 5 6**

**Different numbers of eigenvectors**

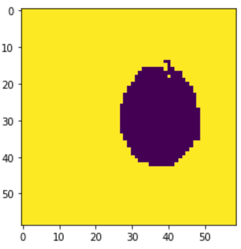
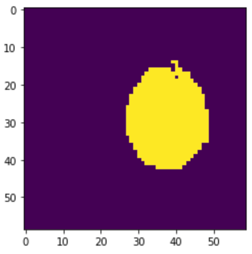
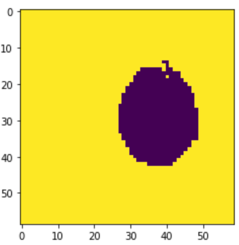
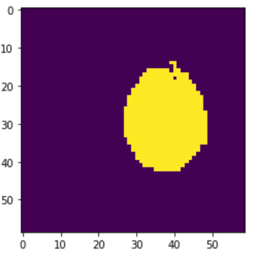
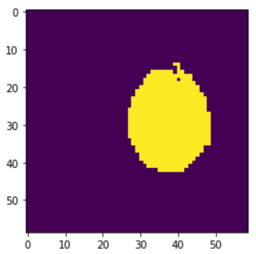
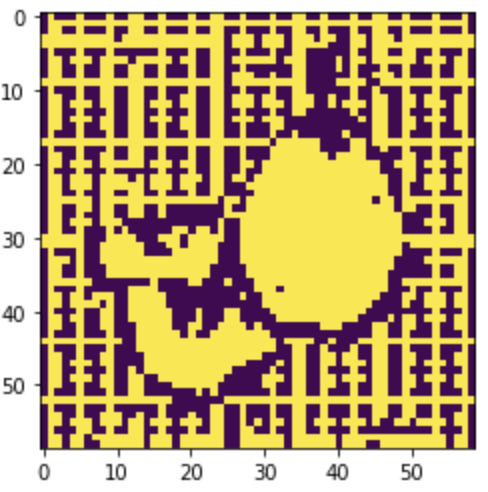
**Cluster = 6**

1. ** 2 3 4 5 6**
2. **Discussion about the influence of different number of eigenvectors and clusters**

Clusters在上面提過，在這個演算法裡的功能是代表分出類群，因此，cluster=n代表把圖片分成n個色塊，由上題的figures可以發現，同一張圖片當clusters的數值較大時，需要較多的eigenvectors才能清楚的顯示圖片，以eigenvectors=4作為對照，cluster=4時能比較好看出輪廓了。但假如cluster數過小，分成的色塊也就比較少，假使eigenvectors數多，還是沒辦法讓輪廓較清楚。以cluster=2為例，甚至當eigenvector變大時，沒辦法將完整的輪廓顯示出。

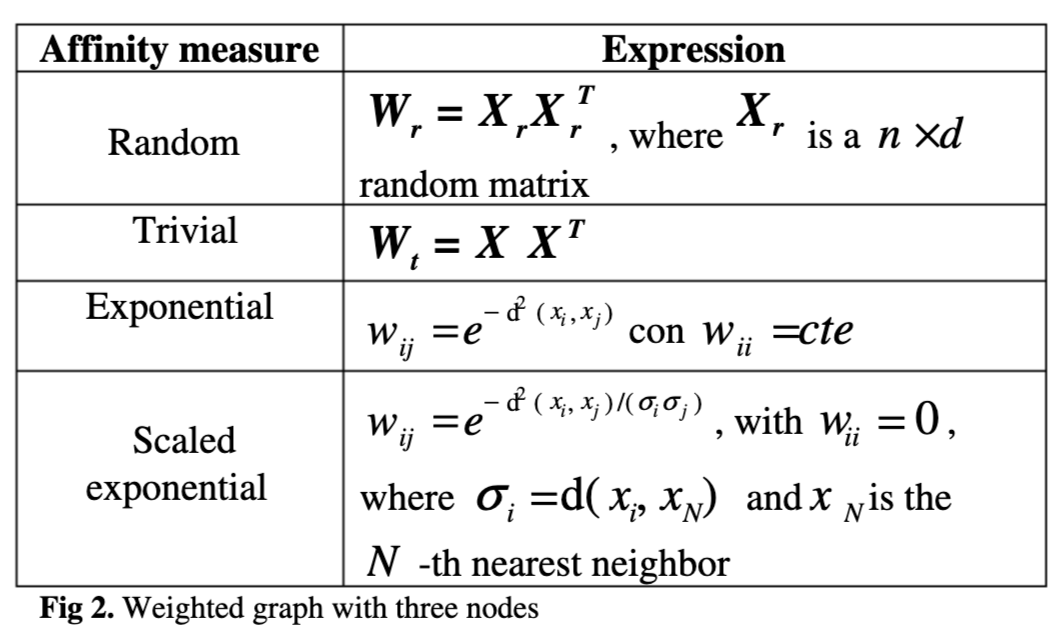
**Different numbers of eigenvectors**

**Cluster = 2**

1. ******** **2 3 4 5 6**
2. **Discussion about the COO format**

它是一種儲存sparse matrix的方式，因為在sparse matrix中value為0的數值很多，用一般的二維矩陣來記錄很佔空間，因為只會紀錄很分散的非零值，所以，COO format就是用來優化此問題的方法。考慮原本的A是N element大小的sparse matrix，但經過COO format後，matrix的大小就會變為sparse matrix中不為0的element個數(k)\*3(row index、column index、value)，分別用來記錄那些非零元素得row值、column值以及數值大小，這樣子存取空間是3k，考慮到k<<N，所以3k<<N，存取空間來說，應該是縮小不少。但是，COO不支援儲存和刪減element，除非轉換為其它格式的矩陣，否則很難對其做操作或是矩陣運算，因此在計算eigenvalues上，所需的時間比較長，因為還要轉換成可運算的矩陣，這是coo format較不方便的地方。

**(h)** **Result and discussion on the bonus question**



這是在網路上搜尋到的論文，所做的比較。根據它的公式去解讀的話，我們使用scaled exponential affinity的話，切割出來的照片會比較少鋸齒，物件跟背景的切割會比較圓滑。我們能了解到，pixel之間的weight如果是用作業中的公式，就會發現到如果兩個顏色差異是一樣的，他們的weight就會是一樣的，比如說一組是紅（1,0,0）黑（0,0,0），另一組是藍（0,1,0）跟黑，量來說差異是一樣的，然而卻有不同的顏色。因此，如果使用scaled exponential affinity，每個weight在運算時，他的pixel都有不同的值，所以即使他們在顏色編碼上的差異一樣，他們的weight會不一樣，這樣就能分辨出顏色差距一樣，但實際顏色不一樣的組合，另外，這篇論文的算法是用距離，但是，我覺得因為pixel利用到的編碼概念，跟算距離的概念是可以相通的，所以這個方法在此也適用。

以下是example，(a)為原圖，(b)是trivial affinity， (c)是exponential affinity (d)是scaled exponential affinity，可以看出scaled exponential affinity的確比較光滑，符合原圖輪廓一些。

參考資料及例子來源：<http://diegopeluffo.com/publicaciones/STSIVA2010.pdf>

