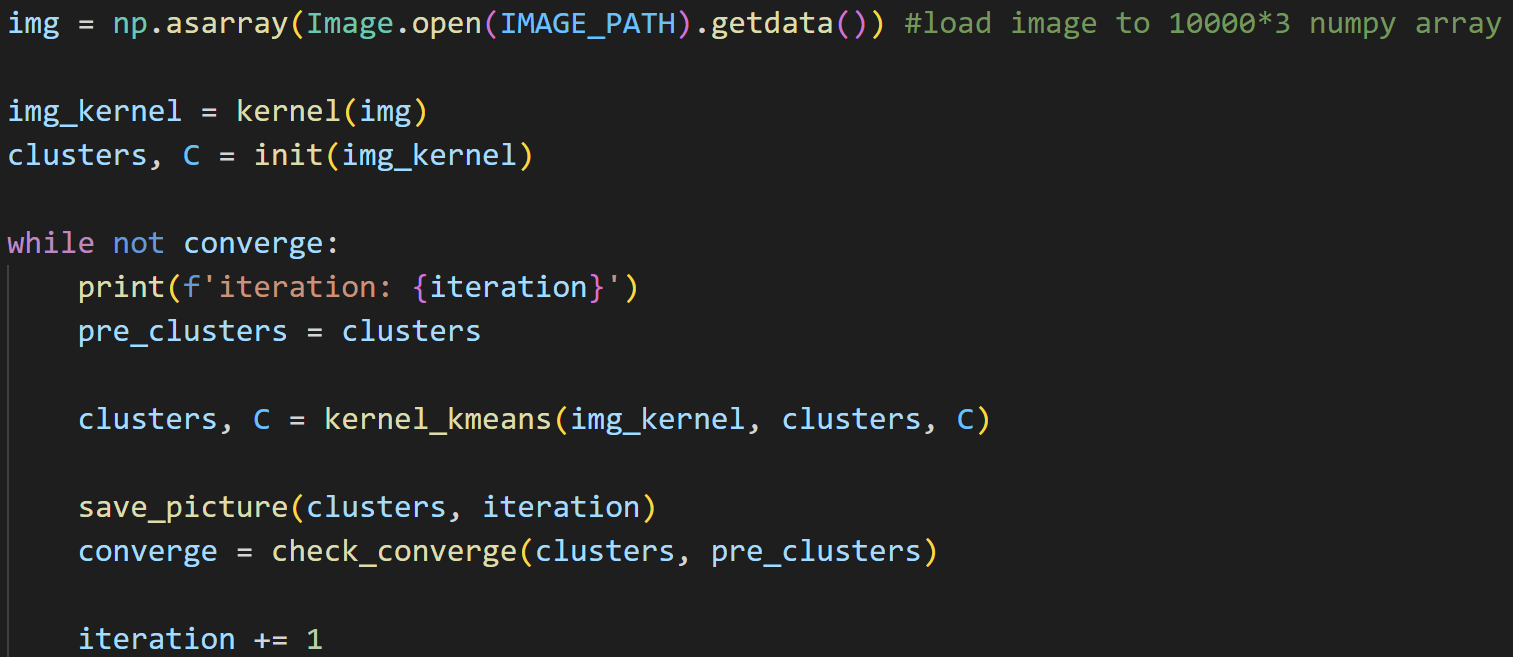
**Machine Learning HW6**

**311551094 資科工碩一 廖昱瑋**

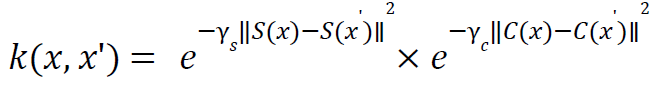
1. **Code**
2. Kernel k-means
3. Main function

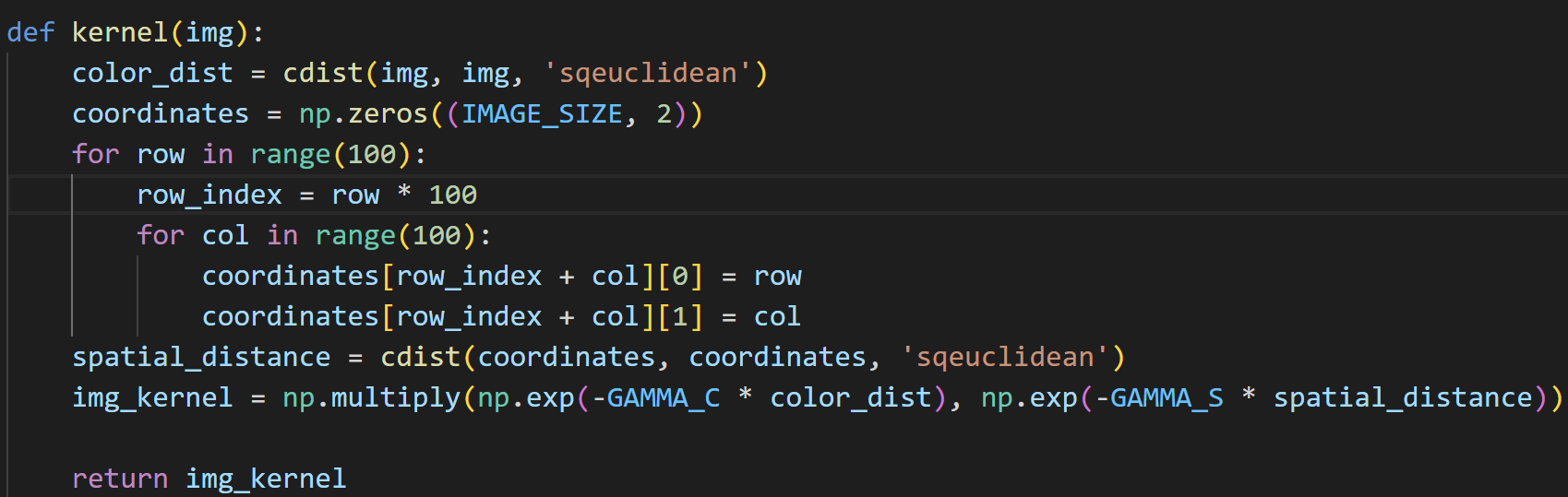
First, load the image. Then, computer kernel of the image and initialize clusters. After that, do kernel k-means in loop until converge.



1. Kernel function

spatial\_distance is , and color\_distance is .

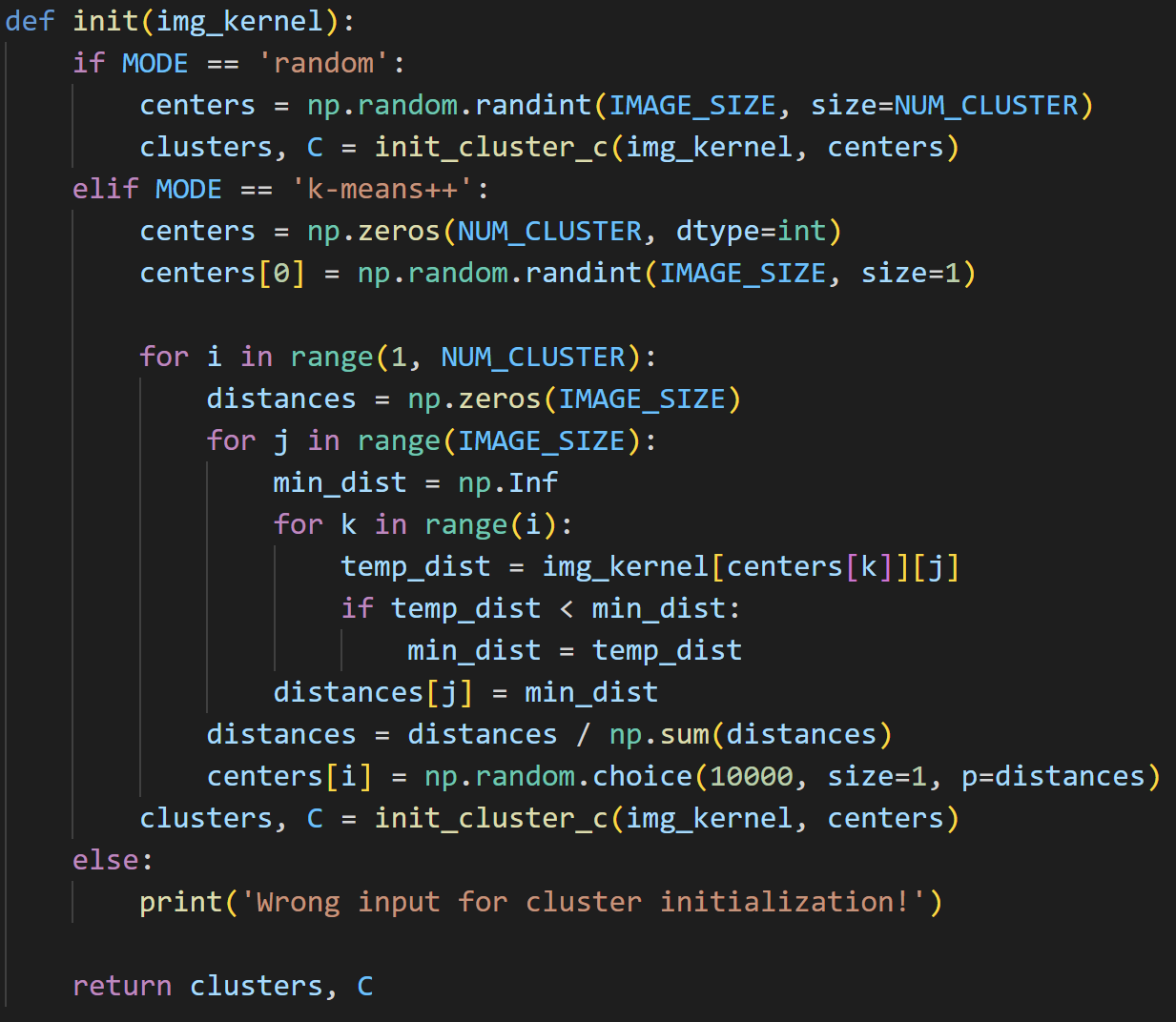
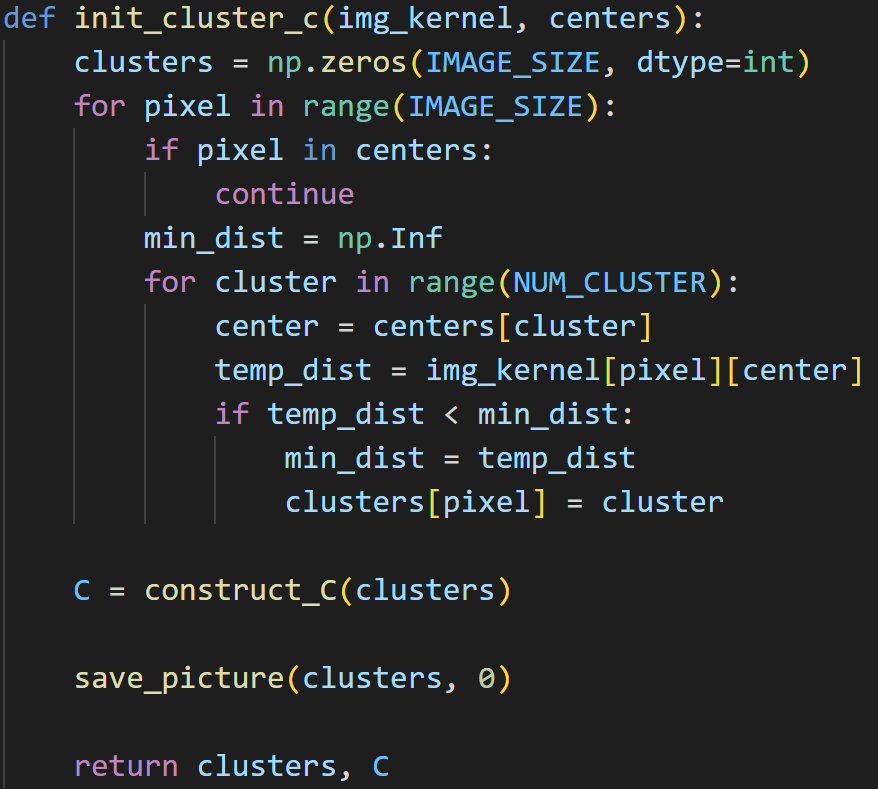
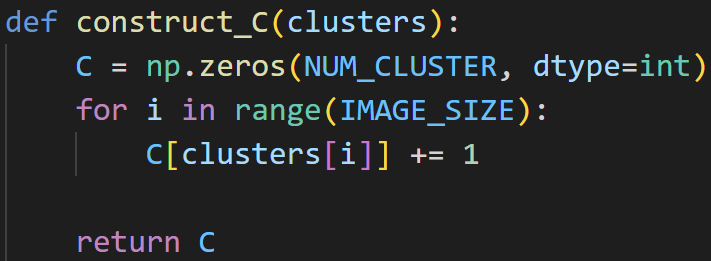




1. Initial clustering

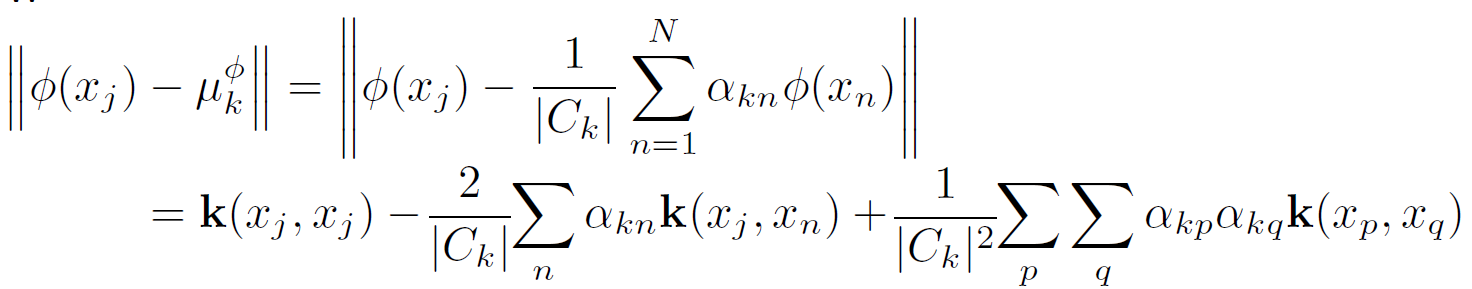
Provide 2 modes which are random and k-means++. The number of initial clusters(k) are defined by users.

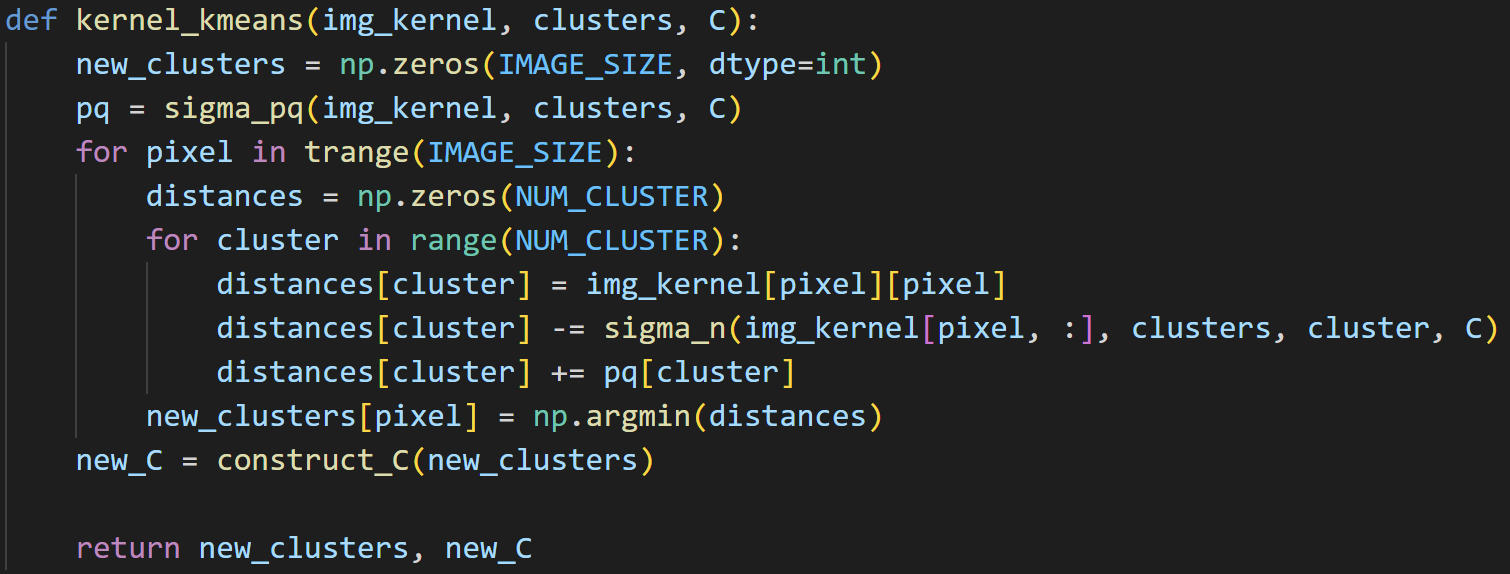
In random mode, randomly choose k data as centers. In k-means++, randomly choose 1 data as first center. Then, find farthest data from selected centers in the loop until we find k centers in total. After finding k centers, *init\_cluster\_c* and *construct\_c* help to do clustering for the initial centers.

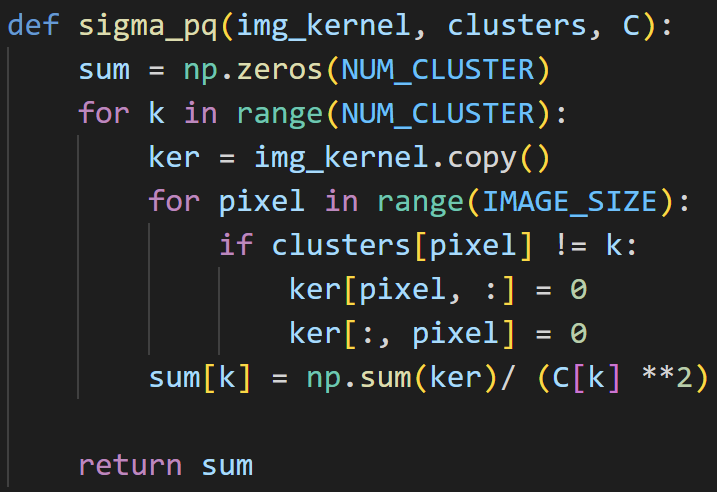
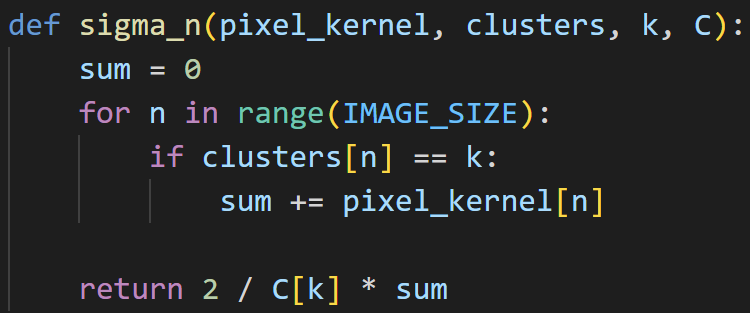
  

1. Do clustering by kernel k-means.

*sigma\_pq* is the second term, and *sigma\_n* is the third term of the following equation.

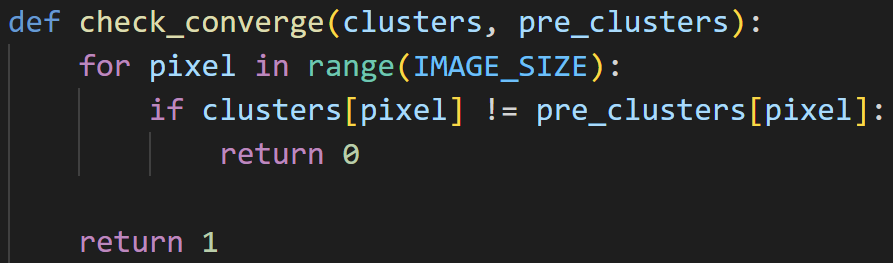




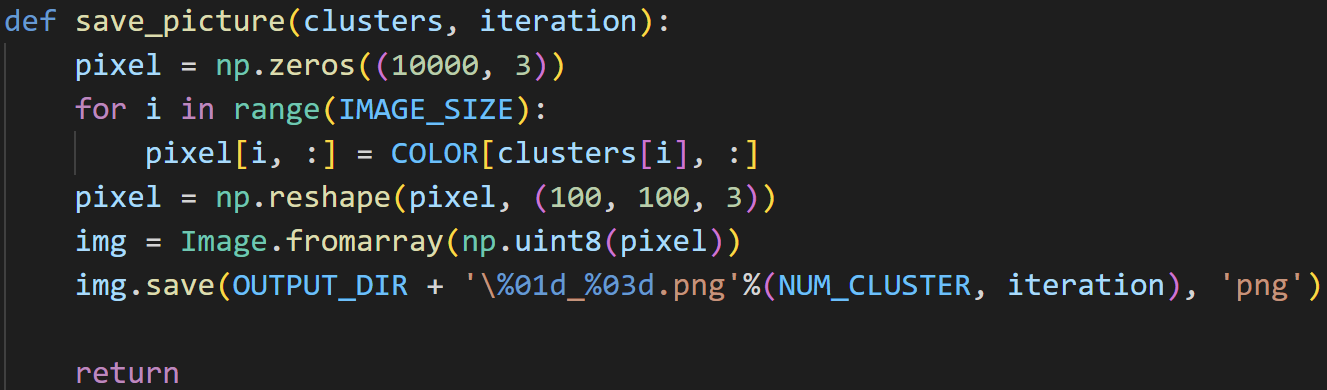
 

1. Check convergence

Define converge as no data changes its cluster in this run.

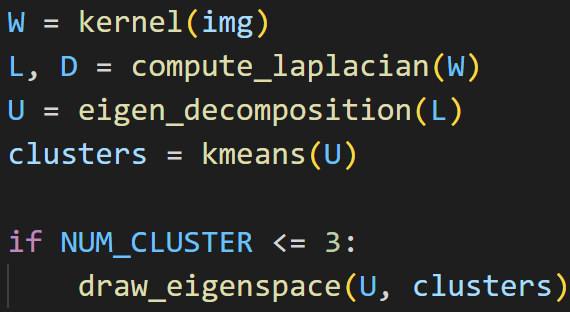
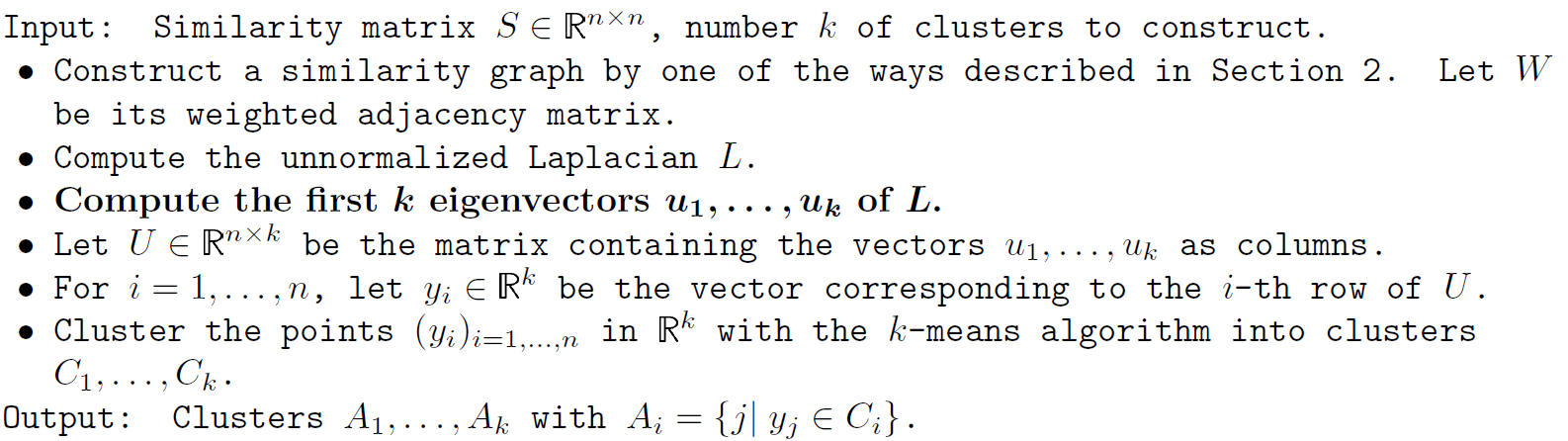


1. Save the result as image in each run.

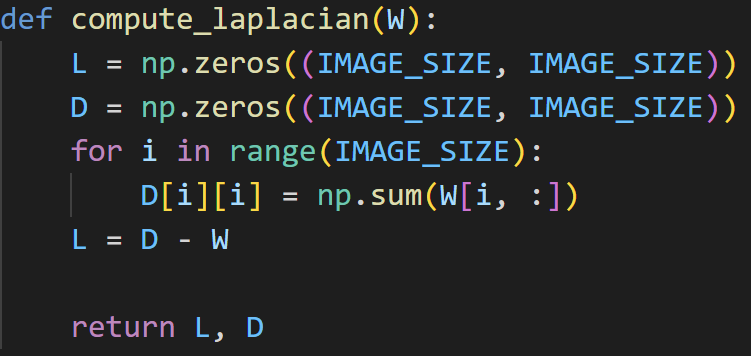
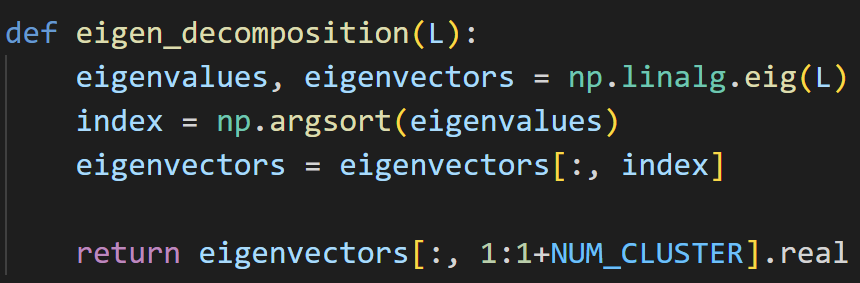


1. Spectral clustering – ratio cut
2. Main function

First compute the similarity matrix W by *kernel*. Then, compute the unnormalized Laplacian L. After that, do eigen decomposition of L and construct U by the number of clusters defined by users. Use k-means algorithm to cluster rows of U. In the last, draw coordinates in the eigenspace if number of clusters are equal or less than 3.

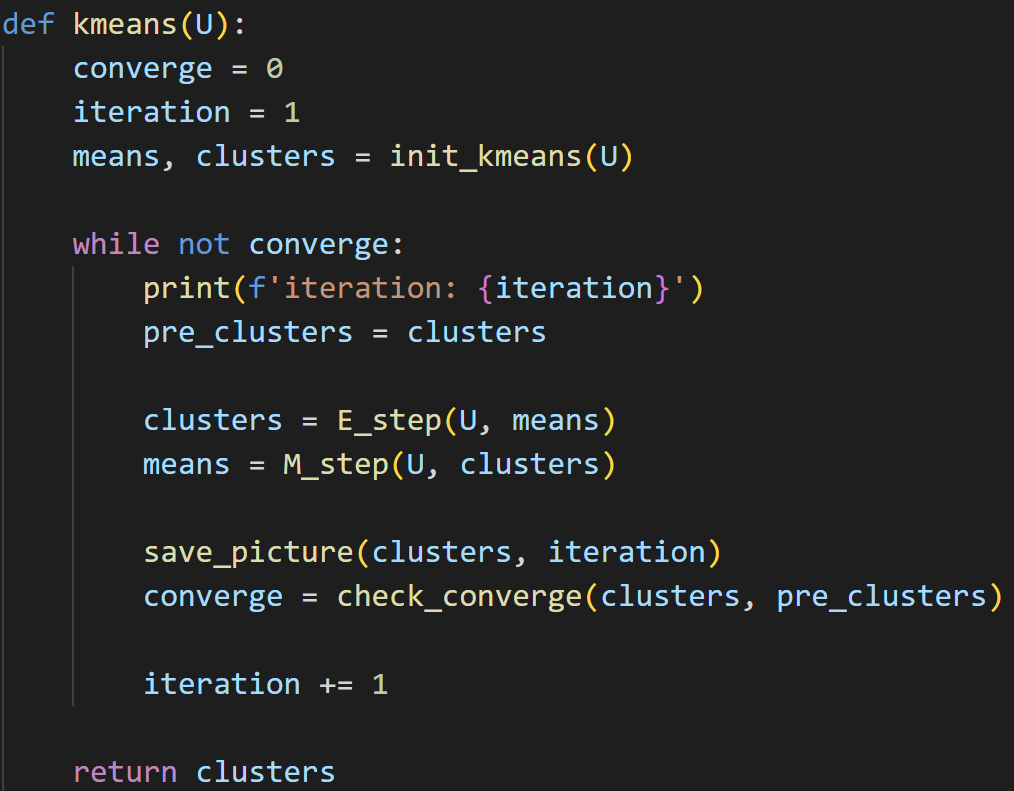
 

1. Compute the unnormalized Laplacian, and do eigen decomposition

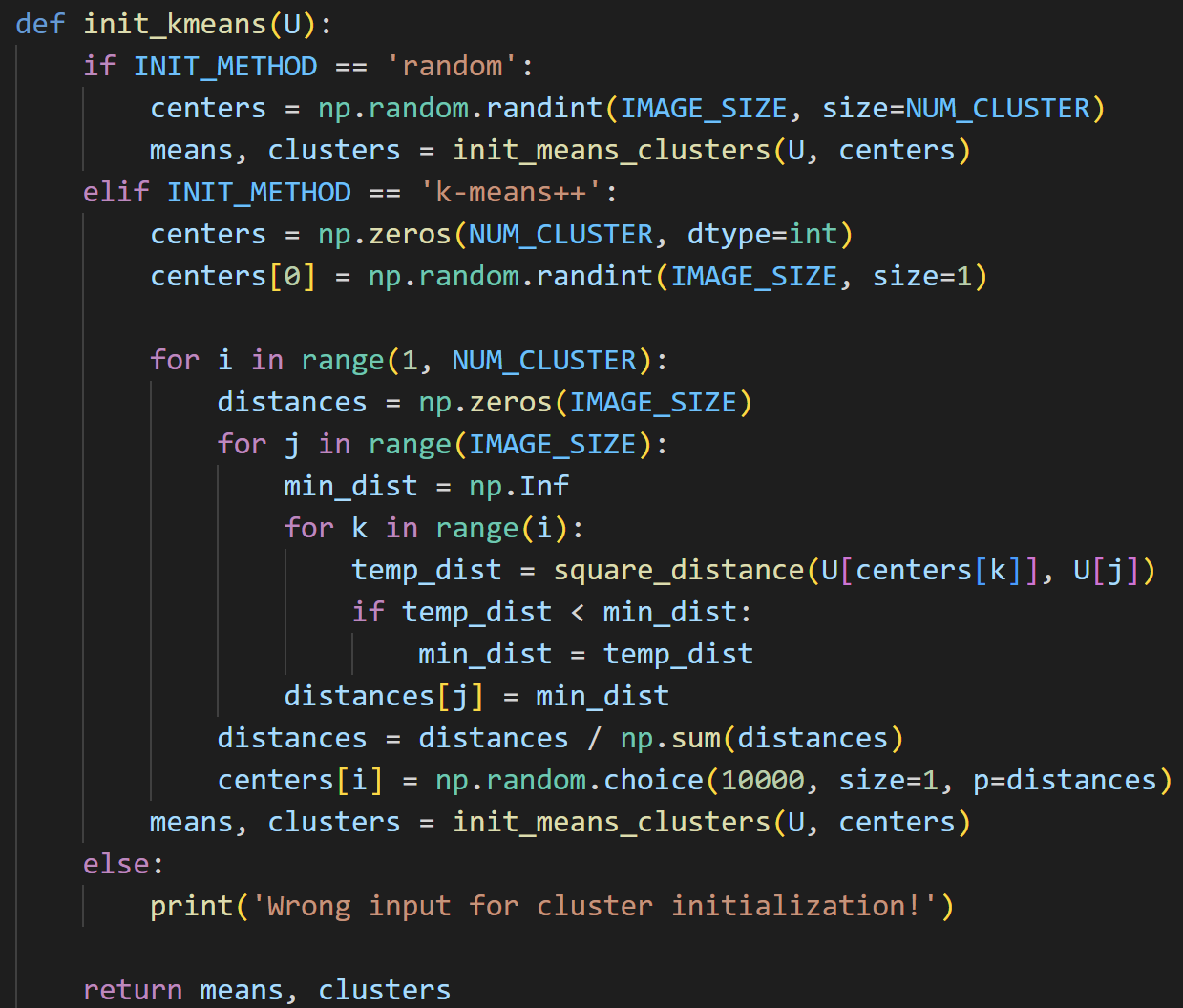
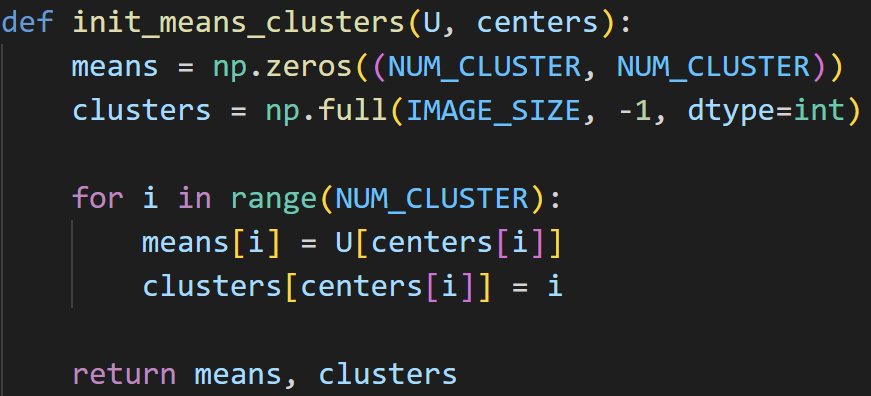
1. k-means algorithm

Initialize the clustering by *init\_kmeans*. In E-step, assign cluster for each data. In M-step, compute mean for each cluster. Do k-means algorithm in loop until converge.

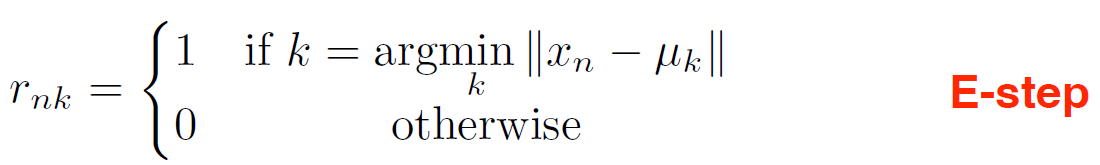
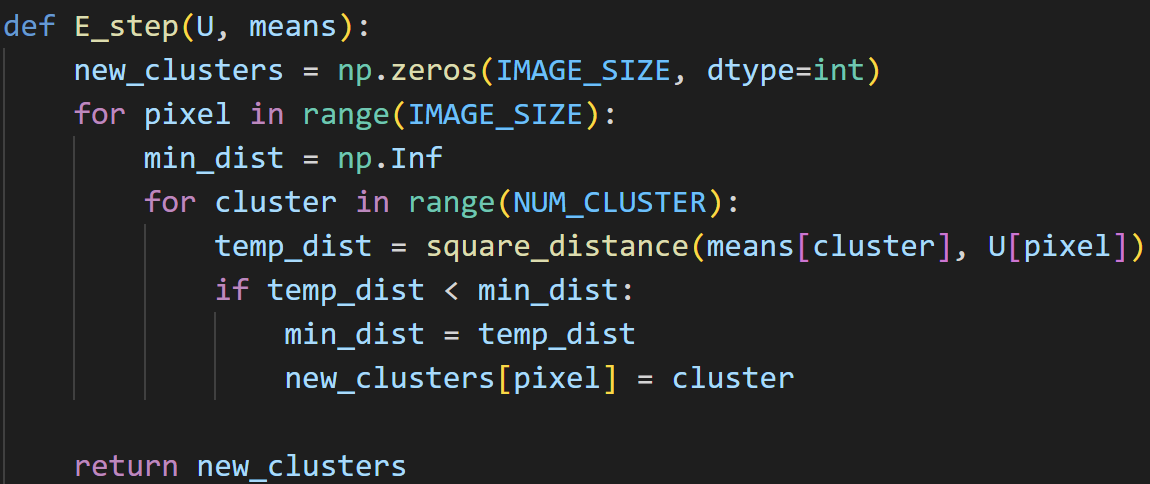


1. Initialize the clusters

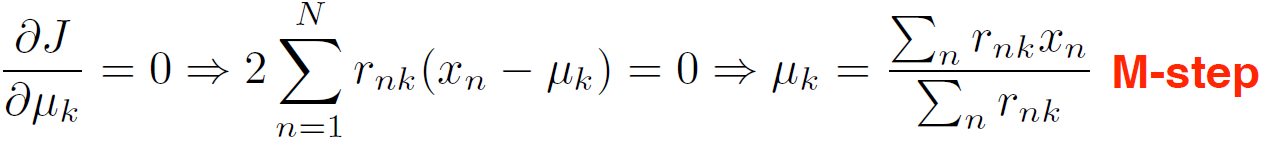
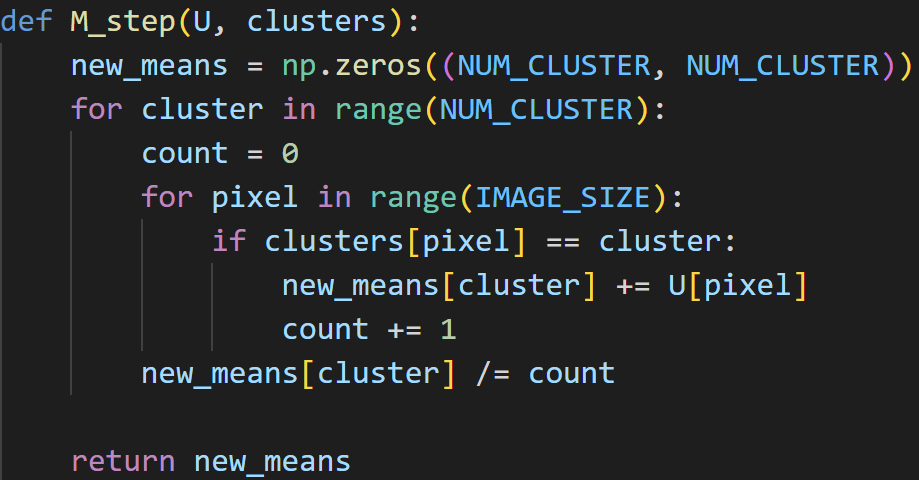
Provide 2 modes which are random and k-means++. The number of initial clusters(k) are defined by users.

1. E-step

1. M-step

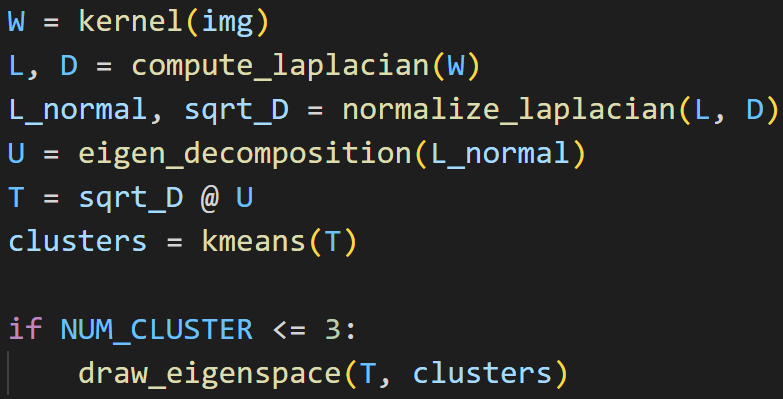
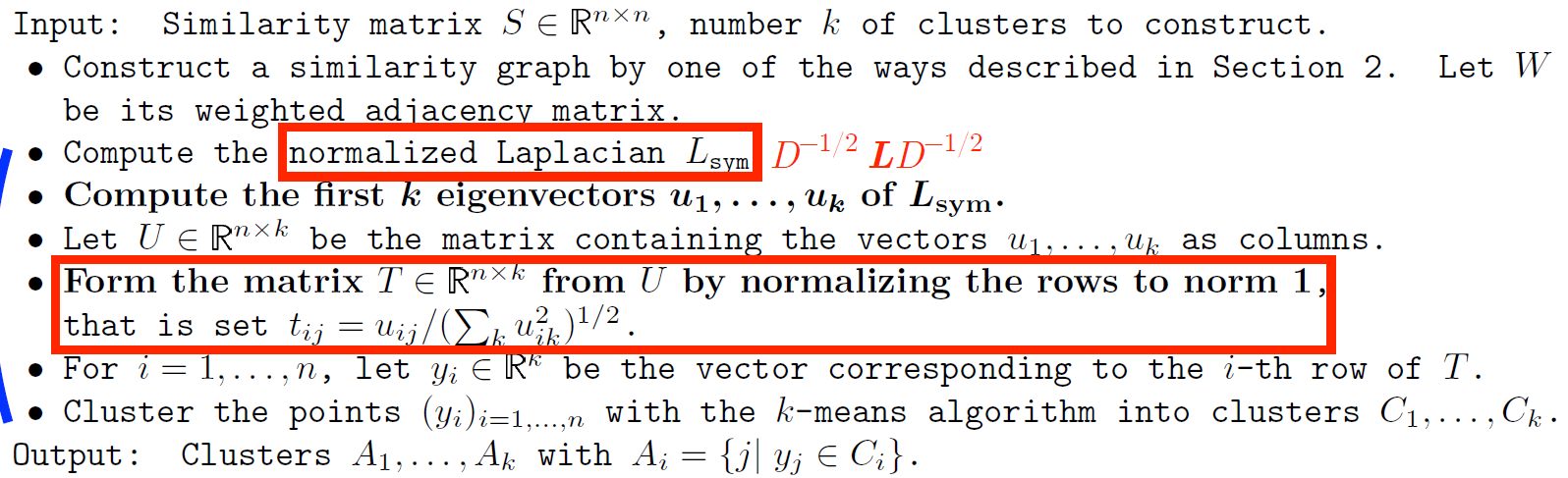
 

1. Draw the coordinates in the eigenspace

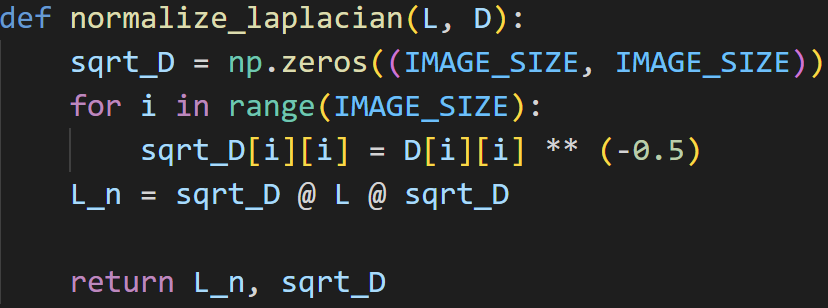


1. Spectral clustering – normalized cut
2. Main function

First compute the similarity matrix W by *kernel*. Then, compute the Laplacian matrix L. Normalized L by *normalize\_laplacian*. After that, do eigen decomposition of L\_normal and construct U by the number of clusters defined by users. Re-substitute T from U by normalizing the rows. Use k-means algorithm to cluster rows of T. In the last, draw coordinates in the eigenspace if number of clusters are equal or less than 3.

1. Normalize the Laplacian matrix



1. **Result**
2. Kernel k-means (、 )
3. random

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 clusters | 3 clusters | 4 clusters |
| Image 1 |  |  |  |
| Image 2 |  |  |  |

1. k-means++

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 clusters | 3 clusters | 4 clusters |
| Image 1 |  |  |  |
| Image 2 |  |  |  |

In the previous results, surprisingly, we can see that k-means++ does not have better result than random and both initial states turn out not too bad results. The reason may be the bad hyperparameters. However, k-mean++ for 4 clusters in image 1 has the better result than random.

1. Spectral clustering – ratio cut (、 )
2. random

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 clusters | 3 clusters | 4 clusters |
| Image 1 |  |  |  |
| Image 2 |  |  |  |

The results are quite good. For image 1, 4 clusters is fairly equal to 3 clusters. This is reasonable because of the origin pattern of the image. However, for image 2, 4 clusters is needed to imply more information because the complexity of the image.

|  |  |  |
| --- | --- | --- |
|  | 2 clusters | 3 clusters |
| Image 1 |  |  |
| Image 2 |  |  |

In eigenspace, we can see that it separates quite well which confirms the above clustering results. The summation in each dimension of eigenvector is 0.

1. k-means++

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 clusters | 3 clusters | 4 clusters |
| Image 1 |  |  |  |
| Image 2 |  |  |  |

The k-means++ results look likely to random results.

|  |  |  |
| --- | --- | --- |
|  | 2 clusters | 3 clusters |
| Image 1 |  |  |
| Image 2 |  |  |

In eigenspace, we can see that it separates quite well which confirms the above clustering results.

1. Spectral clustering – normalized cut (、 )
2. random

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 clusters | 3 clusters | 4 clusters |
| Image 1 |  |  |  |
| Image 2 |  |  |  |

The result is even better than ratio cut. For image 1, 4 clusters can show more information than 3 clusters. It provides mountains in island.

|  |  |  |
| --- | --- | --- |
|  | 2 clusters | 3 clusters |
| Image 1 |  |  |
| Image 2 |  |  |

For image 1, we can see that it has discontinuous clustering in grey part. This also shows in the above image results. Some blue dots are in the island, and some red dots are in the ocean.

1. k-means++

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 clusters | 3 clusters | 4 clusters |
| Image 1 |  |  |  |
| Image 2 |  |  |  |

The clustering results are quite good instead of 2 clusters. It’s because the number of clusters is too small.

|  |  |  |
| --- | --- | --- |
|  | 2 clusters | 3 clusters |
| Image 1 |  |  |
| Image 2 |  |  |

Unlike random initialization, for image 1, the cluster result in eigenspace is great. The above image results also show very good clustering.

1. **Discussion**
2. Spectral clustering has better result than kernel k-means. It’s because we don’t update the means of clusters in kernel k-means. It relies more on the initial state but no matter in which initial method, it contains random properties.
3. It is likely to have different clustering results every time we run the program, even if all the hyperparameters are the same. It’s because of different initial states.
4. The hyperparameters are very important. If we set bad hyperparameters, the result may be nonsense. If γ in RBF kernel is large, the decision boundary will be tight. If γ in RBF kernel is small, the decision boundary will be loose.
5. The hyperparameters are even important than initial method. If we set bad hyperparameters, k-means++ still has bad result.
6. K-mean++ has better time efficiency than random because k-means++ mostly converges in fewer runs.
7. It’s hard to set number of clusters because it doesn’t have correct answer to choose k in unsupervised learning. Therefore, we need to observe the results to determine good k. For image 1, it’s good to choose k as 3. For image 2, it’s good to choose k as 4.