HW9 Spectral Clustering

As always start with importing what we need and finding our data locations.

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from scipy.sparse import lil_matrix, csr_matrix
from sklearn.cluster import SpectralClustering
from scipy.sparse.linalg import svds
from sklearn.cluster import KMeans
import os

os.chdir("C:/Users/rique/Downloads/datasets")
```

Loading our image and normalizing it.

```
def load_dataset():

    img = Image.open("scene256.jpg")

    img = img.resize((256,164))

    img = np.array(img) / 256.0

    plt.imshow(img)
    plt.title('Original Image')
    plt.show()

    return img
```

Get the affinity matrix.

sigma = 0.03, check neighbors

```
def affinity_matrix(img, sigma=0.03):

   rows, cols, _ = img.shape
   pixels = rows * cols

   A = lil_matrix((pixels, pixels))

   for r in range(rows):
        for c in range(cols):
            index = to_index(row, col)

            if r + 1 < rows:
                bottom = r+1 * cols + c
                difference = img[r, c] - img[r + 1, c]
                A[index, bottom] = np.exp(-
np.linalg.norm(difference)*2 / (2 * (sigma*2)))

            if c + 1 < cols:
                right_side = r * cols + c+1
                difference = img[r, c] - img[r, col + 1]
                A[index, right_side] = np.exp(-
np.linalg.norm(difference)*2 / (2 * (sigma*2)))

   return A
```

Perform Spectral Clustering

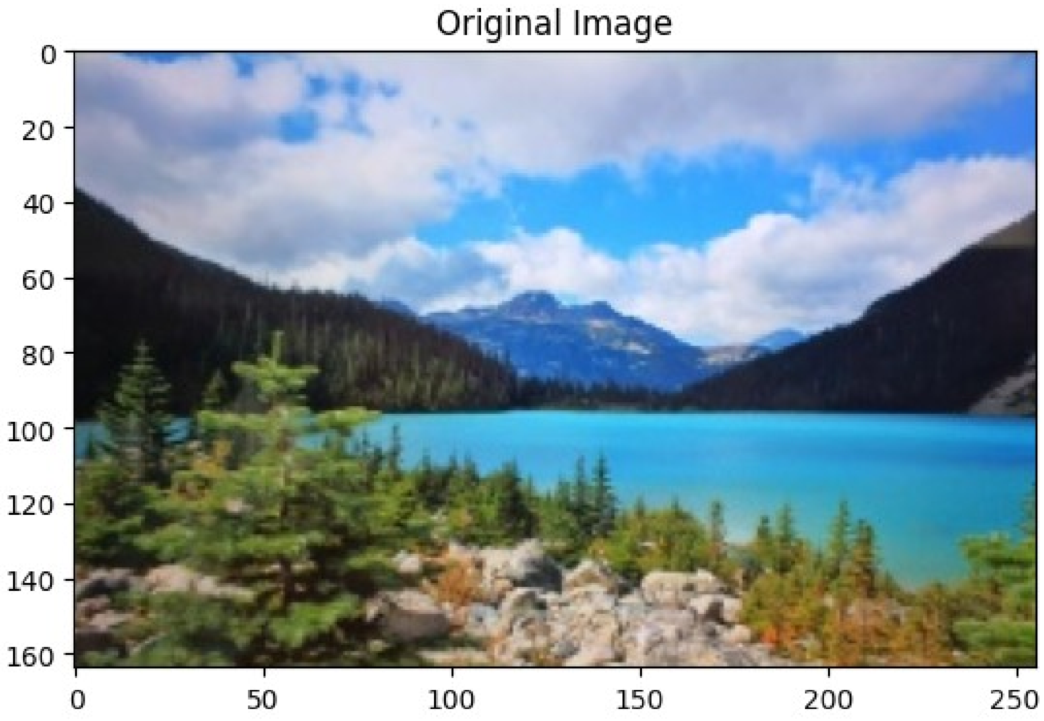
```
def SpectralCluster(A, num_clusters):

    sparse_affinity_matrix = csr_matrix(A)

    degree_matrix = sparse_affinity_matrix.sum(axis=1).A1
    degree_matrix = csr_matrix(np.diag(degree_matrix))
    laplacian_matrix = degree_matrix - sparse_affinity_matrix

    _, _, eigenvectors = svds(laplacian_matrix, k=num_clusters)

    kmeans = KMeans(n_clusters=num_clusters, n_init=10)

    symmetric_matrix = 0.5 * (sparse_affinity_matrix +
sparse_affinity_matrix.T)

    spectral_model = SpectralClustering(n_clusters=num_clusters,
affinity='precomputed')
    labels = spectral_model.fit_predict(symmetric_matrix)

    labels_image = labels.reshape(164, 256)

    plt.imshow(labels_image, cmap='viridis')
    plt.title('Spectral Clustering Result')
    plt.colorbar()
    plt.show()

    return labels
```

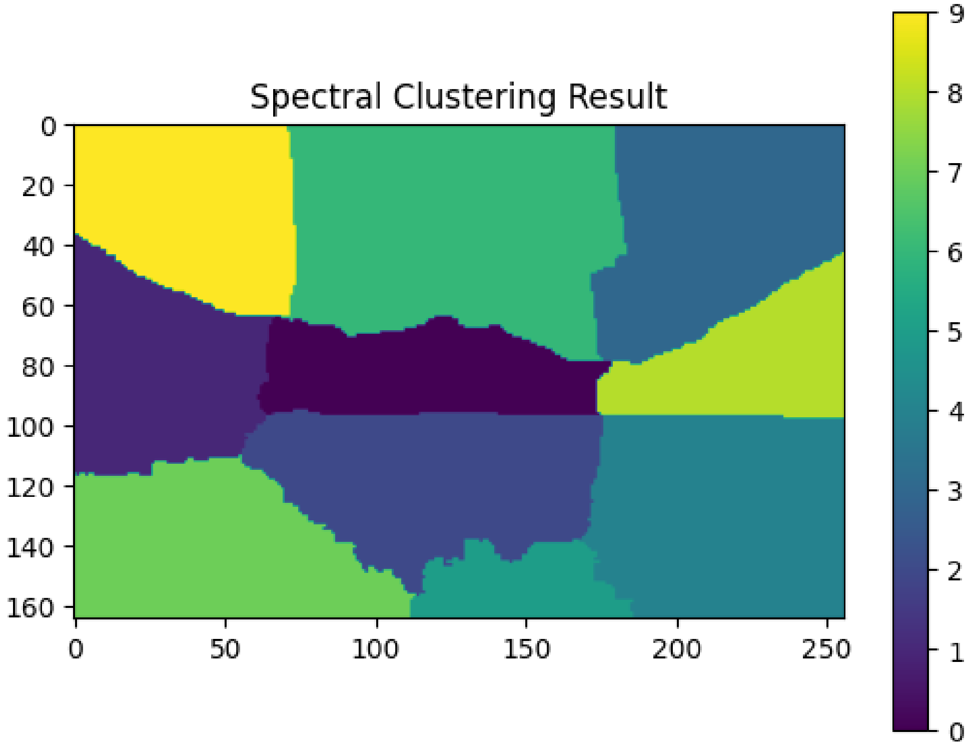
Color our image with the means of RGB pixel values

```
def coloring(img, num_clusters, labels):
    original_image = np.array(img)

    new_img = np.zeros_like(original_image)

    for i in range(num_clusters):
        cluster_pixels = original_image[labels.reshape(164, 256) == i]

        mean_color = np.mean(cluster_pixels, axis=0)

        new_img[labels.reshape(164, 256) == i] = mean_color

    plt.imshow(new_img)
    plt.title('IMAGE WITH CLUSTER MEAN COLORS')
    plt.show()
```

1a

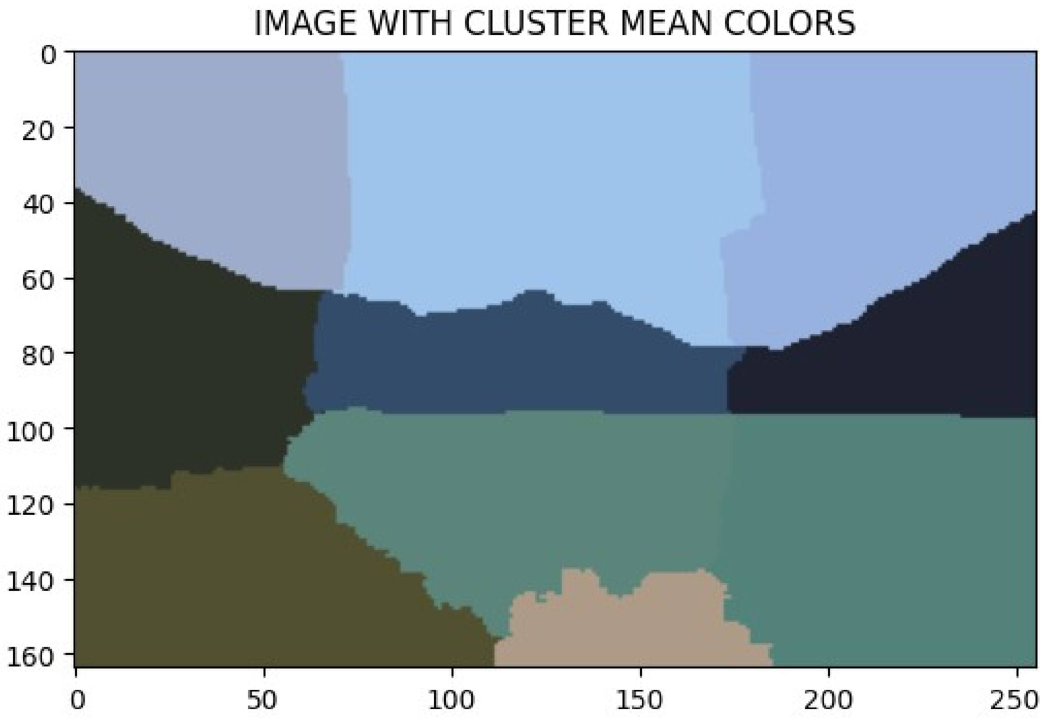
```
x = load_dataset()
A = affinity_matrix(x)
labels = SpectralCluster(A, 10)
```





1b

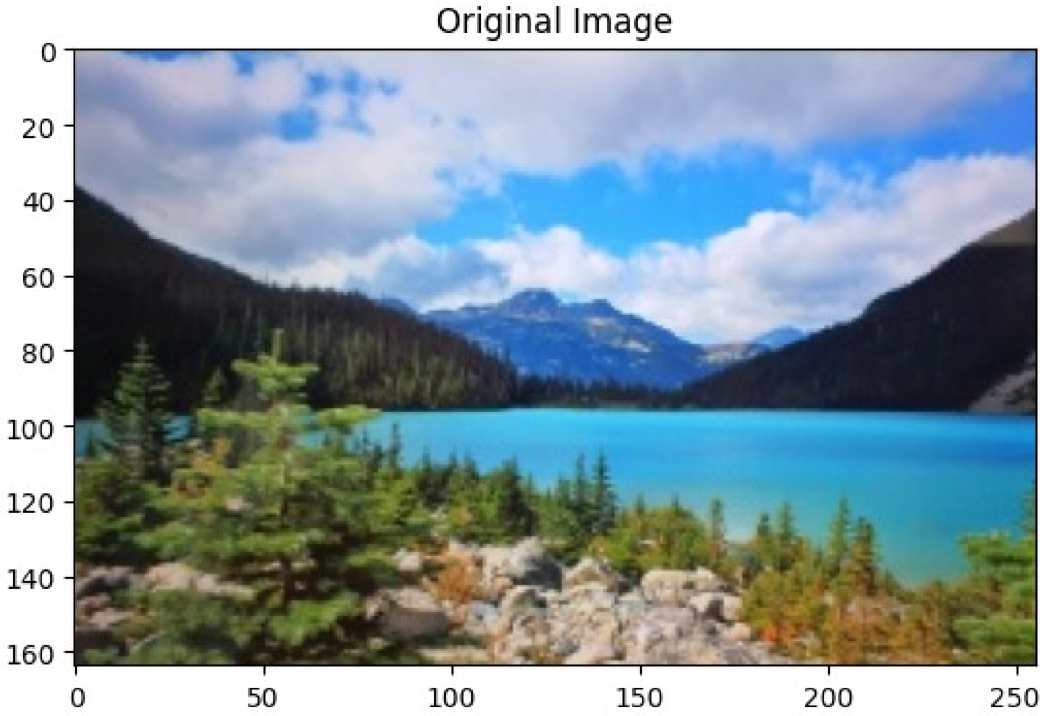
```
coloring(x, 10, labels)
```



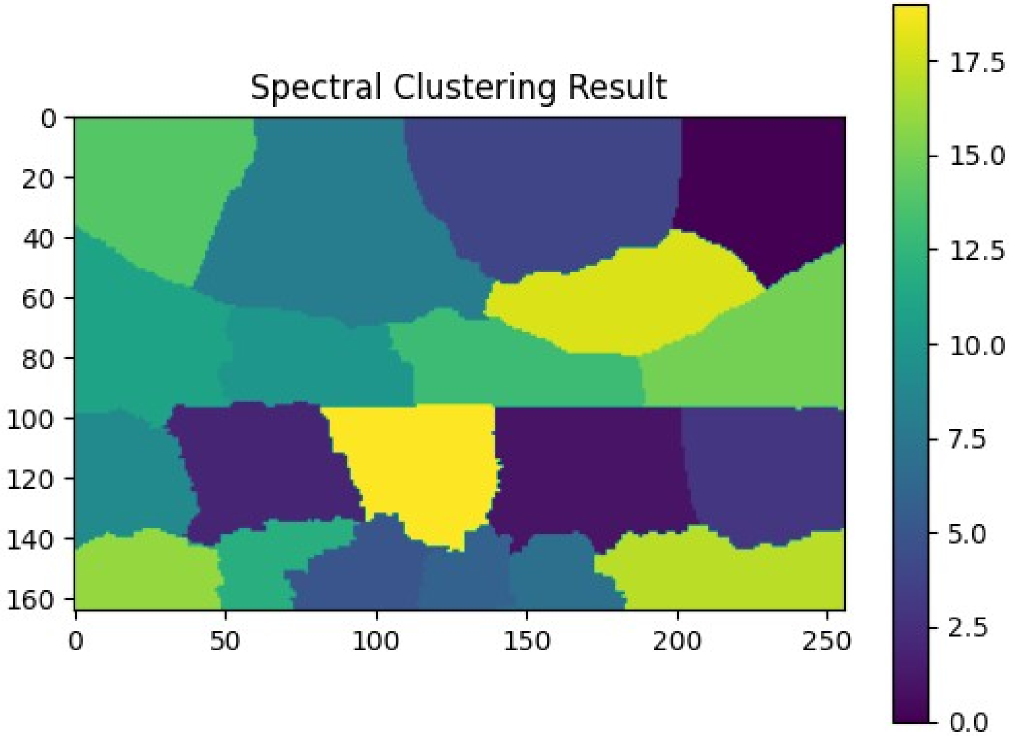
2a (c)

20 clusters

```
x = load_dataset()
A = affinity_matrix(x)
labels = SpectralCluster(A, 20)
```



Original Image



Spectral Clustering Result

2b (c)

20 clusters

```
coloring(x, 20, labels)
```

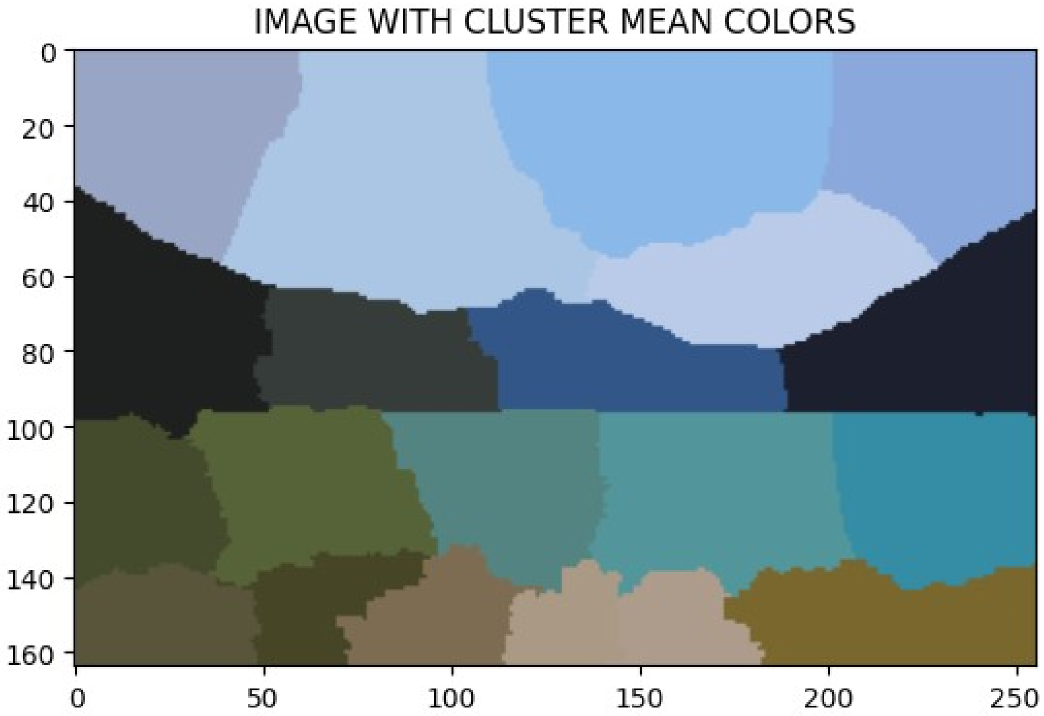


IMAGE WITH CLUSTER MEAN COLORS