HW1-Q2-KMC Code & Result fo beach.bmp with 3 Iterations

Import necessary modules

In [1]:
```
from PIL import Image
from matplotlib.pyplot import imshow
import numpy as np
```

Function to Read image and returns 3-D image array

In [2]:
```
def read_img(path):
    img = Image.open(path)
    img_arr = np.array(img, dtype='int32')
    img.close()
    return img_arr
```

Function to Displays image

In [3]:
```
def display_image(arr):
    arr = arr.astype(dtype='uint8')
    img = Image.fromarray(arr, 'RGB')
    imshow(np.asarray(img))
```

Read Image File and Print Image with Array Shape

In [4]:
```
img_arr = read_img("/Users/ankurdave/OneDrive - Georgia Institute of Technology/ISYE-6740/2021 /homework1/KP.bmp")
display_image(img_arr)
print("Shape of the matrix obtained by reading the image")
print(img_arr.shape)
```

```
Shape of the matrix obtained by reading the image
(410, 618, 3)
```

Reshape the matrix into "img_reshaped" by transforming "img_arr" from a "3-D" matrix to a flattened "2-D" matrix which has 3 columns corresponding to the RGB values for each pixel.

In [5]:
```
r, c, l = img_arr.shape
img_reshaped = np.reshape(img_arr, (r*c, l), order="C")
print("Shape of the reshaped matrix in 2D by multiplying 1st and 2nd element now indicating total number of pixels")
print(img_reshaped.shape)
```

```
Shape of the reshaped matrix in 2D by multiplying 1st and 2nd element now indicating total number of pixels
(253380, 3)
```

Function for Initializing the k centroids as a (X x k) numpy array where X will be the input from image_reshaped (r*c,l)

In [6]:
```
def init_centers(X, k):
    from numpy.random import choice
    start_centers = choice(X), size=k, replace=False)
    return X[start_centers, :]
```

Choosing 3 centroids by caling the init_centers function

In [7]:
```
starting_centers = init_centers(img_reshaped, 4)
print(starting_centers.shape)
print(starting_centers)
```

```
(4, 3)
[[ 93 101  62]
 [173 190 218]
 [ 78  92  43]
 [105 122  78]]
```

Function for Computing the distances of each pixel point with the 3 chosen centers

In [8]:
```
def compute_d2(X, centers):
    m = len(X)
    k = len(centers)
    S = np.empty((m, k))
    for i in range(m):
        d_i = np.linalg.norm(X[i, :] - centers, ord=2, axis=1)
        S[i, :] = d_i**2
    return S
```

Calling fucntion to calculate distances

In [9]:
```
distances = compute_d2(img_reshaped, starting_centers)
```

Function to assign cluster lables

In [10]:
```
def assign_cluster_labels(S):
    return np.argmin(S, axis=1)
```

Calling function to assign cluster lables

In [11]:
```
cluster_lables = assign_cluster_labels(distances)
```

Function to update centers based on the new cluster assignments

In [12]:
```
def update_centers(X, y):
    m, d = X.shape
    k = max(y) + 1
    assert m == len(y)
    assert (min(y) >= 0)
    centers = np.empty((k, d))
    for j in range(k):
        centers[j, :d] = np.mean(X[y == j, :], axis=0)
    return centers
```

Call function to update centers based on the new cluster assignments

In [13]:
```
updated_centers_array = update_centers(img_reshaped,cluster_lables )
print(updated_centers_array)
```

```
[[ 93.68375029 107.15674068  58.26508671]
 [201.86082111 212.03128527 220.52207957]
 [ 74.83546858  86.79652015  40.20565   ]
 [115.41605101 127.37228112  81.76711921]]
```

Function to return the within-cluster sum of squares.

In [14]:
```
def WCSS(S):
    return np.sum(np.amin(S, axis=1))
```

Function to check if centers have moved

In [15]:
```
def has_converged(old_centers, centers):
    return set([tuple(x) for x in old_centers]) == set([tuple(x) for x in centers])
```

Function to combine above steps and run KMC till the centers stop moving and are converged

In [16]:
```
def kmeans(X, k,
           starting_centers=None,
           max_steps=np.inf):
    if starting_centers is None:
        centers = init_centers(X, k)
    else:
        centers = starting_centers

    converged = False
    labels = np.zeros(len(X))
    i = 1
    while (not converged) and (i <= max_steps):
        old_centers = centers
        S = compute_d2(X, centers)
        labels = assign_cluster_labels(S)
        centers = update_centers(X, labels)
        converged = has_converged(old_centers, centers)
        print ("WCSS for this iteration" , i,"=", WCSS (S))
        i += 1
    return labels
```

Calling KMC function to run the iterations

Calculate final Class Lables for the output !!!!!

In [17]:
```
class_labels = kmeans(img_reshaped, 4)
print(class_labels.shape)
```

```
WCSS for this iteration 1 = 1428264070.0
WCSS for this iteration 2 = 523697011.48171425
WCSS for this iteration 3 = 223880189.1960477
WCSS for this iteration 4 = 179883902.31759885
WCSS for this iteration 5 = 172332697.66990656
WCSS for this iteration 6 = 169613236.98012513
WCSS for this iteration 7 = 168029834.0617551
WCSS for this iteration 8 = 166943957.56275874
WCSS for this iteration 9 = 166109607.39875853
WCSS for this iteration 10 = 165467904.88481134
WCSS for this iteration 11 = 164943753.11700824
WCSS for this iteration 12 = 164496545.00249213
WCSS for this iteration 13 = 164148595.9242062
WCSS for this iteration 14 = 163849014.51752135
WCSS for this iteration 15 = 163571291.95352045
WCSS for this iteration 16 = 163361861.11801404
WCSS for this iteration 17 = 163159220.2226887
WCSS for this iteration 18 = 162985721.89447835
WCSS for this iteration 19 = 162828987.7191753
WCSS for this iteration 20 = 162714050.47713983
WCSS for this iteration 21 = 162596549.31988382
WCSS for this iteration 22 = 162468003.72179365
WCSS for this iteration 23 = 162359975.1548586
WCSS for this iteration 24 = 162254169.88904005
WCSS for this iteration 25 = 162131356.17727023
WCSS for this iteration 26 = 162016728.27542132
WCSS for this iteration 27 = 161896195.94890183
WCSS for this iteration 28 = 161790658.6923191
WCSS for this iteration 29 = 161693557.55508196
WCSS for this iteration 30 = 161584550.39254192
WCSS for this iteration 31 = 161469946.73182687
WCSS for this iteration 32 = 161336244.46751946
WCSS for this iteration 33 = 161208263.93622127
WCSS for this iteration 34 = 161053744.72764462
WCSS for this iteration 35 = 160899397.9087784
WCSS for this iteration 36 = 160707421.92157415
WCSS for this iteration 37 = 160479895.05909663
WCSS for this iteration 38 = 160259217.0988935
WCSS for this iteration 39 = 160003628.28156197
WCSS for this iteration 40 = 159681409.91363358
WCSS for this iteration 41 = 159364818.17443758
WCSS for this iteration 42 = 159015166.19624475
WCSS for this iteration 43 = 158651803.68004084
WCSS for this iteration 44 = 158282771.42947623
WCSS for this iteration 45 = 157974430.22353813
WCSS for this iteration 46 = 157706492.94827256
WCSS for this iteration 47 = 157481969.5062016
WCSS for this iteration 48 = 157289736.63677762
WCSS for this iteration 49 = 157165128.55661675
WCSS for this iteration 50 = 157061708.75894588
WCSS for this iteration 51 = 156959816.4081241
WCSS for this iteration 52 = 156945029.33443618
WCSS for this iteration 53 = 156905581.98887813
WCSS for this iteration 54 = 156887694.37532637
WCSS for this iteration 55 = 156874679.5419912
WCSS for this iteration 56 = 156866720.7949487
WCSS for this iteration 57 = 156864421.6573583
WCSS for this iteration 58 = 156862905.2444199
WCSS for this iteration 59 = 156861790.80739585
WCSS for this iteration 60 = 156861141.62237707
WCSS for this iteration 61 = 156860286.7217244
WCSS for this iteration 62 = 156859875.92660248
WCSS for this iteration 63 = 156859108.91304712
WCSS for this iteration 64 = 156854403.825438
WCSS for this iteration 65 = 156846430.68627587
WCSS for this iteration 66 = 156844623.96584573
WCSS for this iteration 67 = 156844318.9239065
WCSS for this iteration 68 = 156844227.17021734
WCSS for this iteration 69 = 156844191.756389
WCSS for this iteration 70 = 156844169.9286336
WCSS for this iteration 71 = 156844159.89931163
WCSS for this iteration 72 = 156844156.99290836
(253380,)
```

Calculate the new cluster centers based on new Class labels for the output !!! Storage structure -- label:array(cluster_center)

In [18]:
```
ind = np.column_stack((img_reshaped, class_labels))
centroid = {}
for i in set(class_labels):
    c = ind[ind[:,3] == i].mean(axis=0)
    centroid[i] = c[:3]
print(centroid)
```

```
{0: array([111.22019691, 124.50817231,  77.23104077]), 1: array([213.24508313, 228.49369758,
241.39599784]), 2: array([165.23659903, 158.29834897, 148.3299149 ]), 3: array([83.3900986 ,
96.23893375, 48.40274658])}
```

Generating a matrix of the same dimensions as img_reshaped, where each pixel is replaced by the cluster center to which it belongs

In [19]:
```
img_new = np.array([centroid[i] for i in class_labels])
print(img_new.shape)
```

```
(253380, 3)
```

Displaying the clustered image

In [20]:
```
r, c, l = img_arr.shape
img_display = np.reshape(img_new, (r, c, l), order="C")
display_image(img_display)
```

Above is the updated image with new class_lablers and centroids !!!!!