

---

## *PCV PRACTICAL - 9*

---

Id:- 190030059

Name:- A.R.Snehita

### **PRE-LAB:-**

#### **1)\_What Is Color Quantization?**

Color quantization is applied when the color information of an image is to be reduced. The most common case is when a 24-bit color image is transformed into an 8-bit color image. In computer graphics, color quantization or color image quantization is quantization applied to color spaces; it is a process that reduces the number of distinct colors used in an image, usually with the intention that the new image should be as visually similar as possible to the original image. Computer algorithms to perform color quantization on bitmaps have been studied since the 1970s. Color quantization is critical for displaying images with many colors on devices that can only display a limited number of colors, usually due to memory limitations, and enables efficient compression of certain types of images.

#### **2) State the main reasons for performing color quantization?**

Color quantization is critical for displaying images with many colors on devices that can only display a limited number of colors, usually due to memory limitations, and enables efficient compression of certain types of images. Nowadays, color quantization is mainly used in GIF and PNG images. GIF, for a long time the most popular lossless and animated bitmap format on the World Wide Web, only supports up to 256 colors, necessitating quantization for many images. Some early web browsers constrained images to use a specific palette known as the web colors, leading to severe degradation in quality compared to optimized palettes. PNG images support 24-bit color, but can often be made much smaller in filesize without much visual degradation by application of color quantization, since PNG files use fewer bits per pixel for palettized images.

### 3) Why do we use K-Means Clustering for color quantization?

Performs a pixel-wise Vector Quantization (VQ) of an image of the summer palace, reducing the number of colors required to show the image from 96,615 unique colors to 64, while preserving the overall appearance quality. In this example, pixels are represented in a 3D-space and K-means is used to find 64 color clusters. In the image processing literature, the codebook obtained from K-means (the cluster centers) is called the color palette. Using a single byte, up to 256 colors can be addressed, whereas an RGB encoding requires 3 bytes per pixel. The GIF file format, for example, uses such a palette.

### 4) Write the formula for Euclidean distance for two RGB colors?

Euclidean Distance in n-space is generally defined by the following formula, where **p** and **q** are cartesian coordinates.

$$\begin{aligned}d(\mathbf{p}, \mathbf{q}) &= d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}.\end{aligned}$$

Since RGB Color space is has three dimensions (R, G, B), its Euclidean Distance would be defined as the following:

$$\text{distance}_e = \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}$$

### IN-LAB TASK:

Two Best Friends Named Ramesh and Rajesh were thinking to reduce the number of colors in an image but they were not familiar with that. Help them to implement the Color Quantization with OPENCV Using K means clustering algorithm for the given image where K value is 3?

Code:

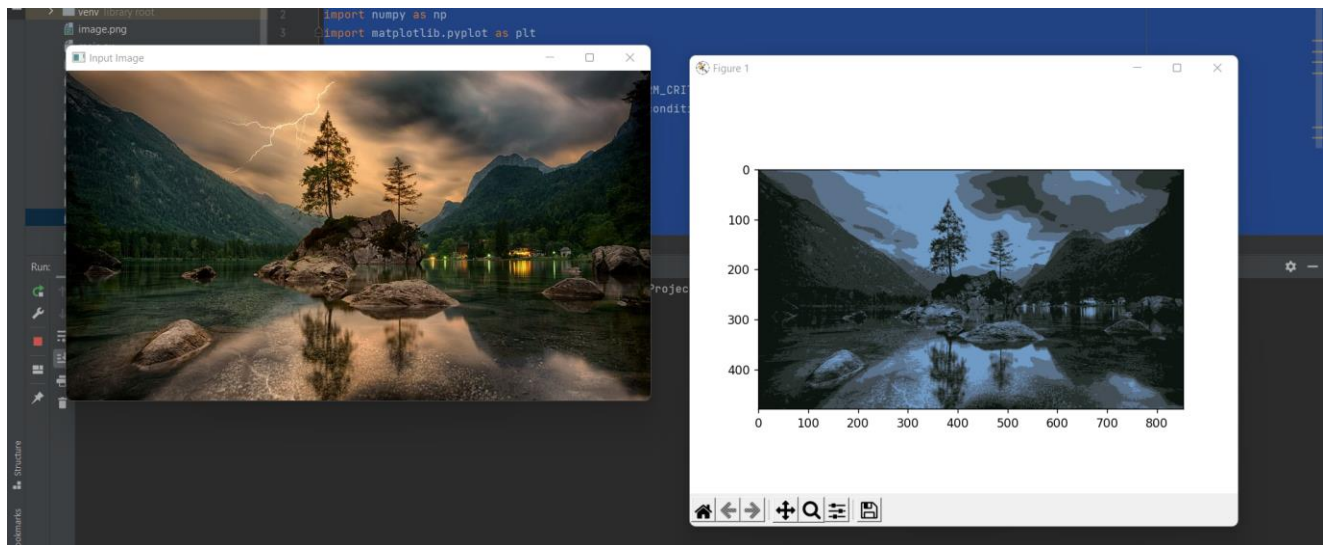
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
def quantimage(image,k):
    i = np.float32(image).reshape(-1,3)
    condition = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,20,1.0)
```

```

ret,label,center = cv2.kmeans(i, k , None,
condition,10,cv2.KMEANS_RANDOM_CENTERS)
center = np.uint8(center)
final_img = center[label.flatten()]
final_img = final_img.reshape(image.shape)
return final_img
image = cv2.imread('pcvimage.jpg')
plt.imshow(quantimage(image,5))
cv2.imshow("Input Image", image)
plt.show()

```

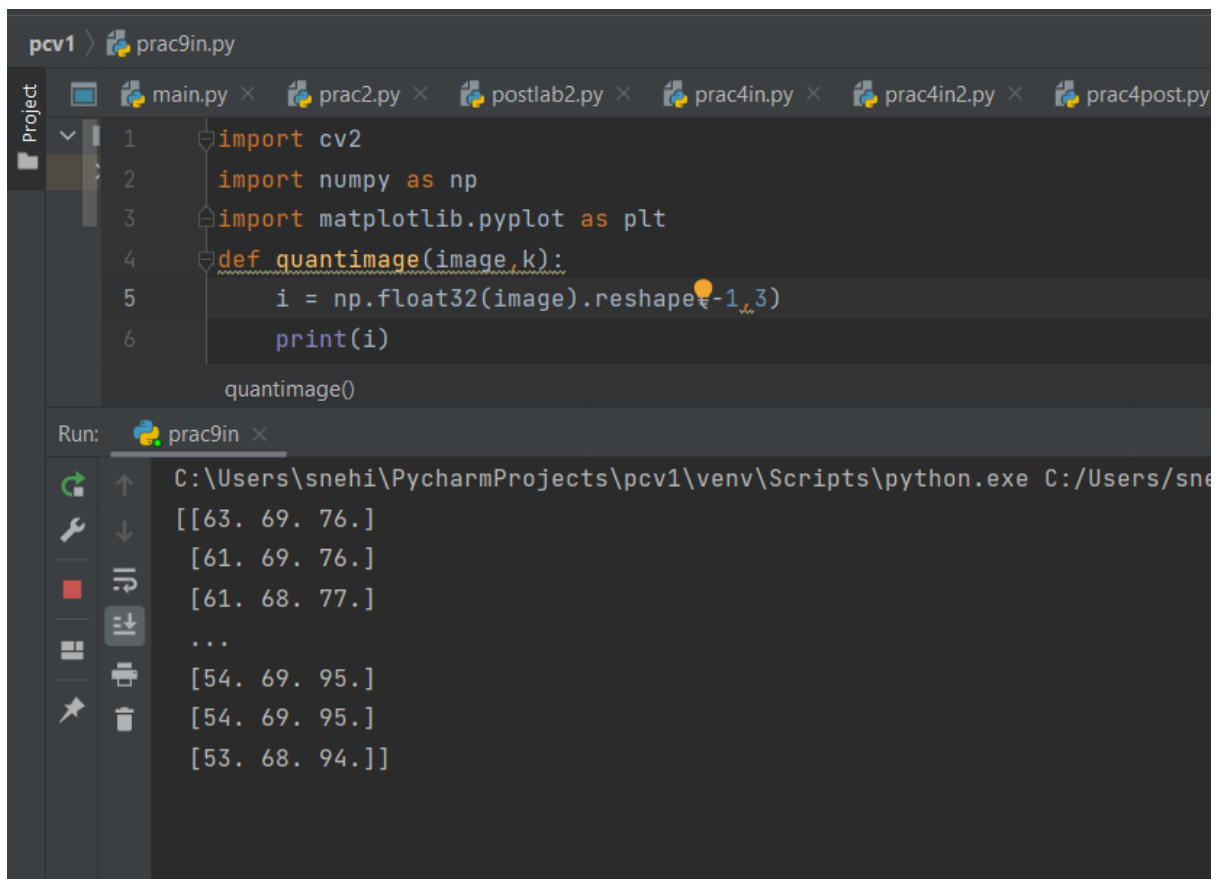
## OUTPUT:



## POSTLAB:

The two friends were satisfied for the Color Quantization which you had performed but they had doubt regarding how it was actually implemented. Demonstrate to them by doing this manually

- In the first step we shall try to choose the number of clusters for our image. Here number of cluster basically represents the number of colors we want to represent our image with. In our problem we would see how to represent the image with 3 and 50 colors.

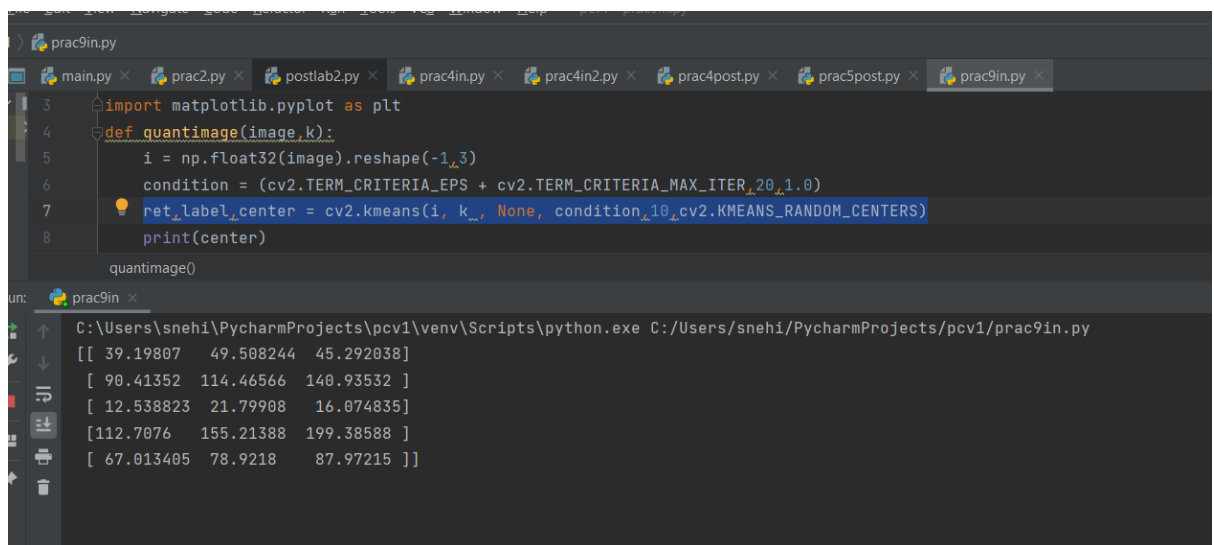


```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4 def quantimage(image,k):
5     i = np.float32(image).reshape(-1,3)
6     print(i)
7
8 quantimage()
```

Run: prac9in

```
C:\Users\snehi\PycharmProjects\pcv1\venv\Scripts\python.exe C:/Users/snehi/PycharmProjects/pcv1/prac9in.py
[[63. 69. 76.]
 [61. 69. 76.]
 [61. 68. 77.]
 ...
 [54. 69. 95.]
 [54. 69. 95.]
 [53. 68. 94.]]
```

- After choosing the number of colors now it is the time to choose the cluster centroids which would be the color representative of the clusters. For example for 3 colors let  $C_1 = (120, 140, 180)$ ,  $C_2 = (125, 180, 170)$ ,  $C_3 = (122, 145, 182)$  be the three cluster centers.



```
3 import matplotlib.pyplot as plt
4 def quantimage(image,k):
5     i = np.float32(image).reshape(-1,3)
6     condition = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 20, 1.0)
7     ret,label,center = cv2.kmeans(i, k, None, condition, 10, cv2.KMEANS_RANDOM_CENTERS)
8     print(center)
9
10 quantimage()
```

Run: prac9in

```
C:\Users\snehi\PycharmProjects\pcv1\venv\Scripts\python.exe C:/Users/snehi/PycharmProjects/pcv1/prac9in.py
[[ 39.19807   49.508244  45.292038]
 [ 90.41352   114.46566   140.93532 ]
 [ 12.538823   21.79908   16.074835]
 [112.7076   155.21388   199.38588 ]
 [ 67.013405   78.9218    87.97215 ]]
```

- In the next step we shall compute the distance of each point from the cluster centroids. And based on the distance we shall assign each point to the center with shortest distance. To compute the distance we shall consider the euclidean distance. We would continue this process till the maximum number of iteration is reached or the center doesn't change at all.

