Authors: Michel and Neelanjan. Image clustering is the process of categorizing a collection of images into groups or clusters based on their visual similarities. Utilizing clustering algorithms, such as k-means or hierarchical clustering, images are grouped together based on shared features like color, texture, shape, or other visual attributes. The aim is to partition the image dataset into clusters where images within the same cluster exhibit higher similarity compared to those in other clusters. this lab was inspired from the opencv library source link :https://docs.opencv.org/3.4/d1/d5c/tutorial_py_kmeans_opencv.html

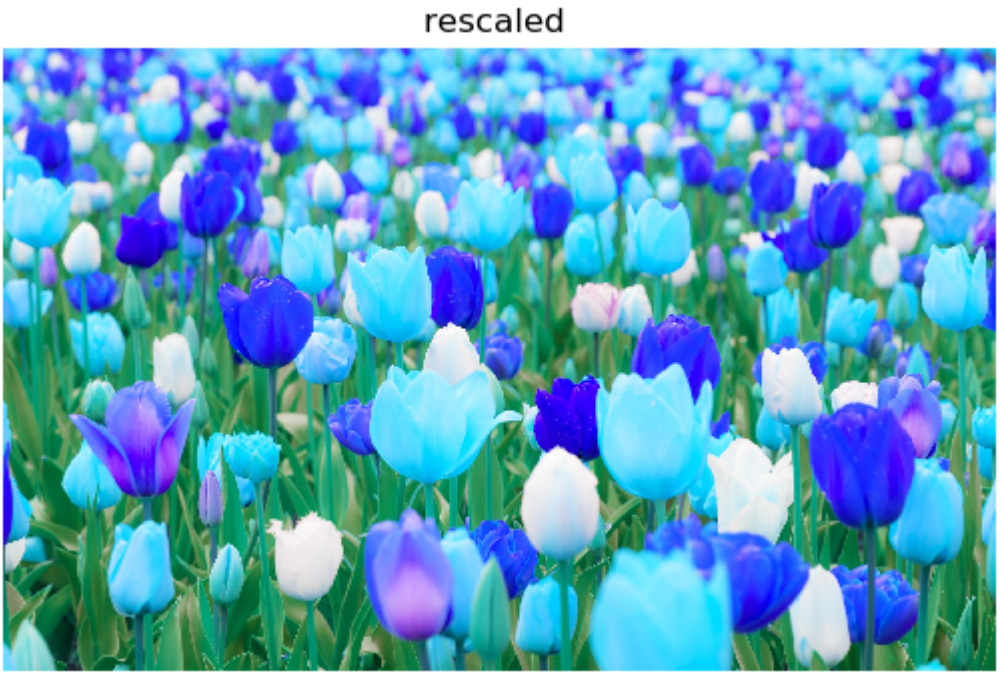
In [ ]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
import math
import cv2
```

We first import our image into our notebook for better visualization, and then we convert it to RGB format as requested in the lab

In [ ]:
```
img = cv2.imread('tulips.jpg')
imgrgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)


plt.imshow(imgrgb)
plt.axis('off')
plt.title('original')
plt.show()
```

https://scikit-image.org/docs/stable/user_guide/data_types.html, for converting image , since Matlab is different from Open Cv the rescale is not needed here for our image clustering operation.but still done as a requirement of the lab

In [ ]:
```
image_int = img.astype(np.float32)# convert image to data type
image_scaled = image_int / 255.0 #rescale of image

plt.imshow(image_scaled)
plt.axis('off')
plt.title('rescaled ')
plt.show()
```



we reshape our image as learn in the very first labs . and display data of the 2d image

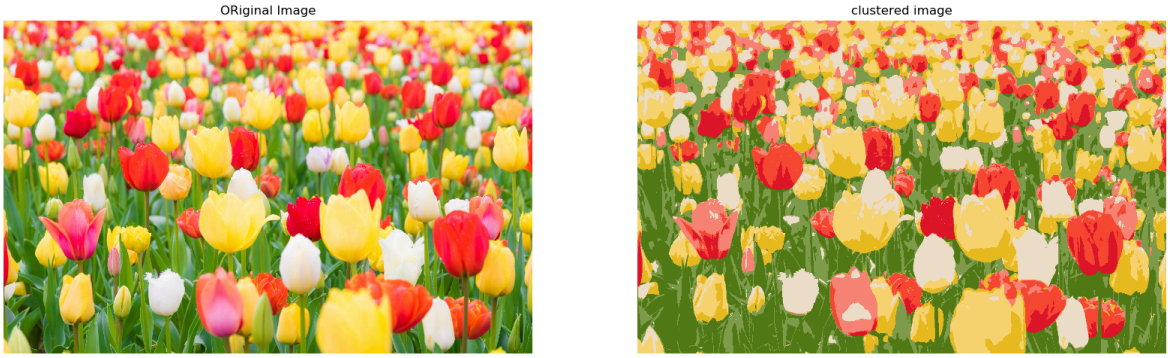
In [ ]:
```
image_data = np.array(imgrgb)
flat_img = image_data.reshape(-1, 3) # 3 to specifies that each row shoul

print (flat_img)
```

```
[[233 184 126]
 [233 184 126]
 [232 186 126]
 ...
 [ 52 104   6]
 [ 45  95   0]
 [ 45  94   2]]
```

Here is the main operation is happening for the image clustering . i pick 8 for the K means represented as k .

In [ ]:
```
K = 8

flat_img = np.float32(flat_img)

# Perform K-means clustering
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)

# Set flags (Just to avoid line break in the code)
flags = cv2.KMEANS_RANDOM_CENTERS

ret, label, center = cv2.kmeans(flat_img,K,None,criteria,10,flags)
```

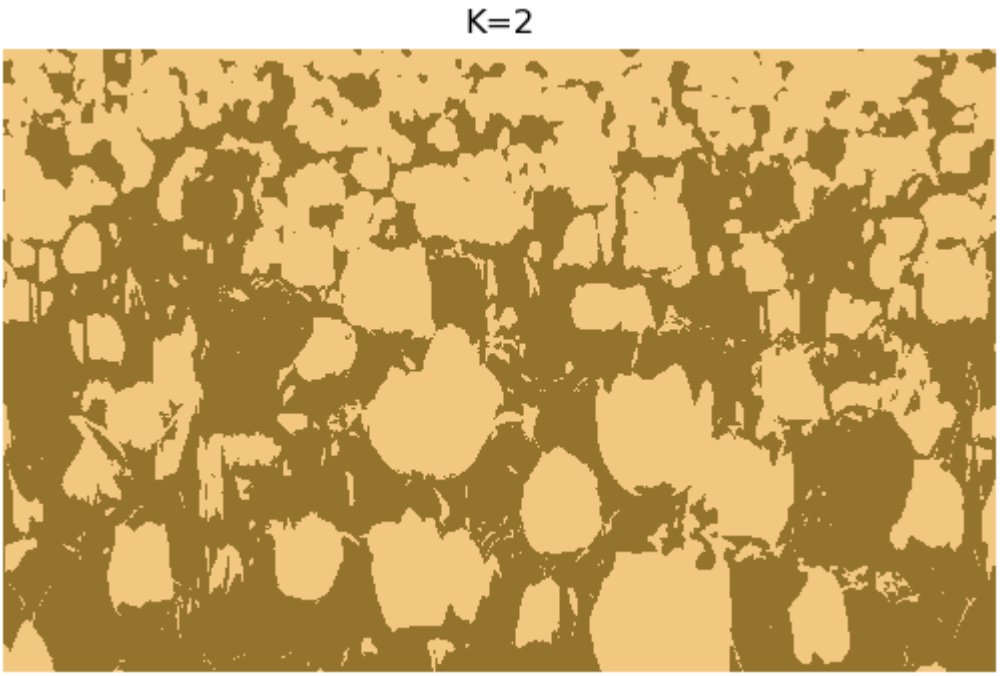
```
# Labels will contain the cluster assignments for each pixel
# Centers will contain the centroid values for each cluster


# convert it back to normal for display

center = np.uint8(center)
res = center[label.flatten()]
res1 = res.reshape((img.shape))




# mathlab plot
plt.figure(figsize=(20, 10))
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.title('ORiginal Image')
plt.subplot(1, 2, 2)
plt.imshow(res1)
plt.title('clustered image')
plt.axis('off')
plt.show()
```

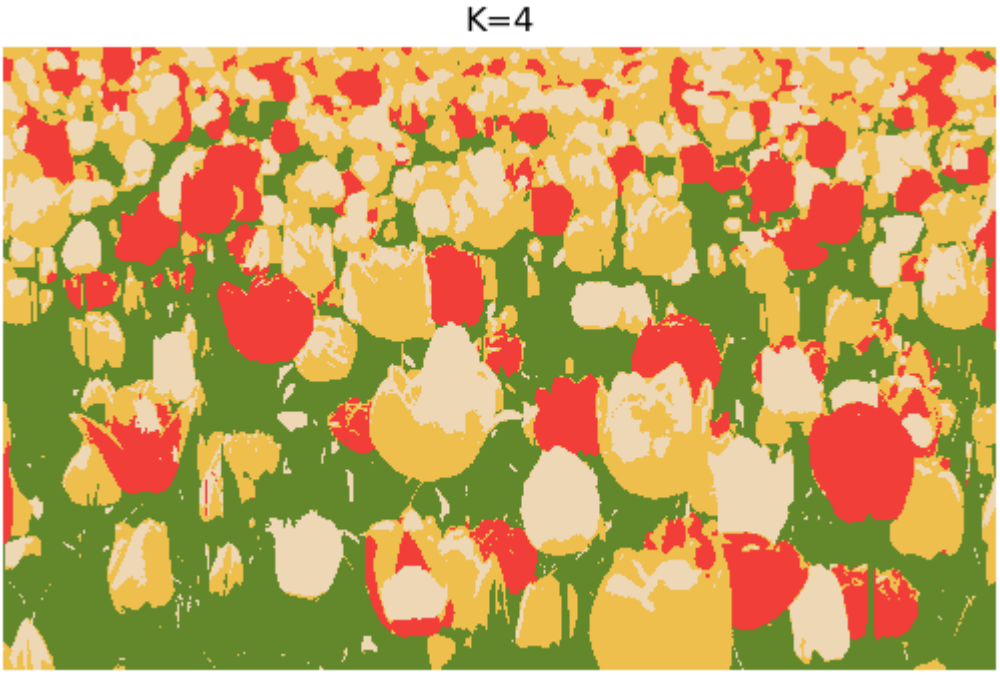


Getting the rest of the images clustering by a different K means values

In [ ]:
```
# Perform K-means clustering for different k values

for K in [2, 4, 8]:
    flat_img = np.float32(flat_img)

    # Perform K-means clustering
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1
    flags = cv2.KMEANS_RANDOM_CENTERS
    ret, label, center = cv2.kmeans(flat_img, K, None, criteria, 10, flag

    # Convert center to uint8
    center = np.uint8(center)

    # Map the labels to the center values
    res = center[label.flatten()]
    res1 = res.reshape(img.shape)

    # Display the result
    plt.figure()
    plt.imshow(res1)
```

```
    plt.title(f'K={K}')
    plt.axis('off')

plt.show()
```

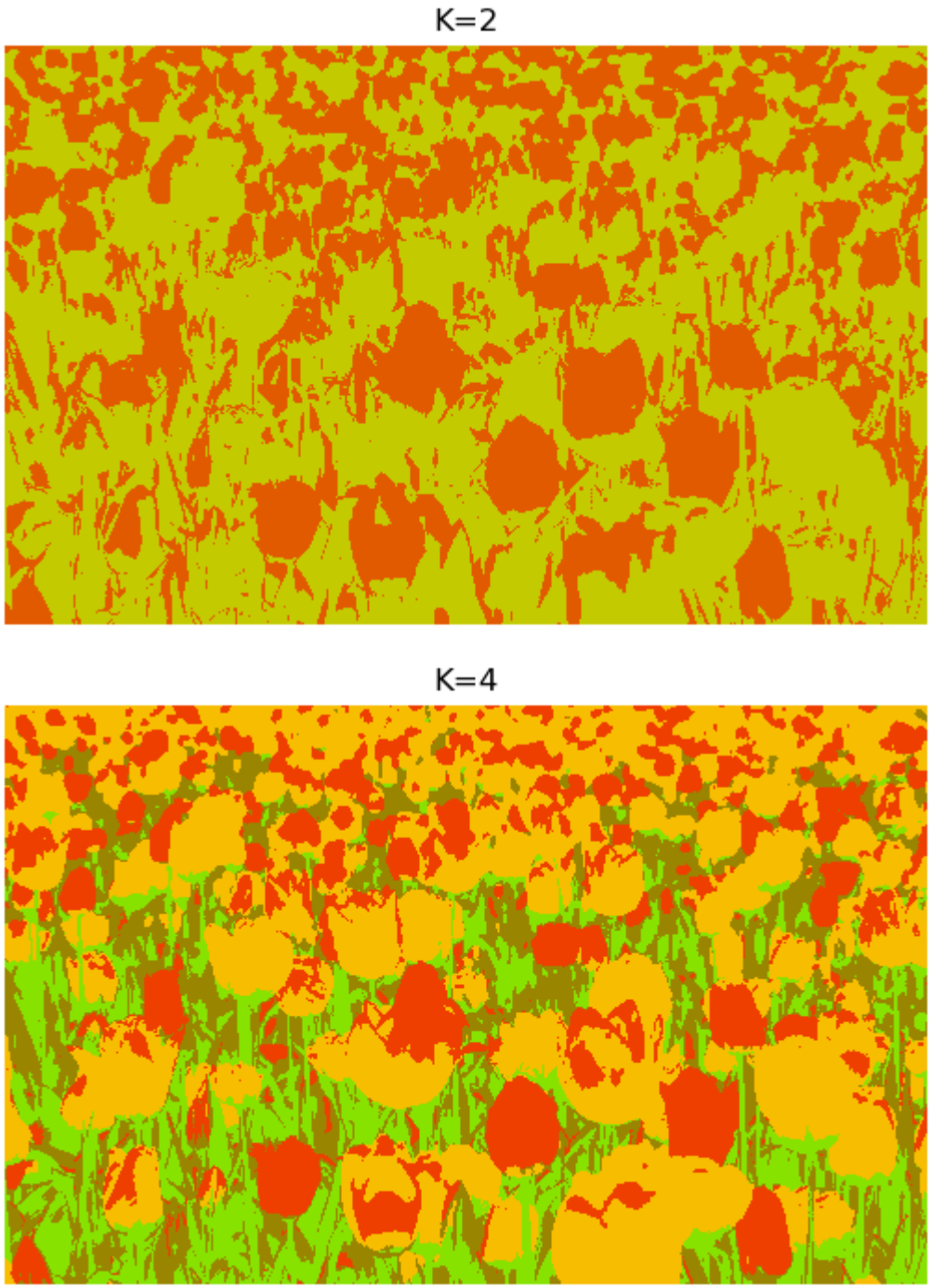




Remap hsv value and applying the clustering to the hsv images to do the same operation but while remap the hsv the values to the cylindrincal coordinates

In [ ]:
```
# Convert BGR to HSV
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Remap to cylindrical coordinates
hsv_mapped = np.zeros_like(hsv, dtype=np.float32)
hsv_mapped[..., 0] = hsv[..., 0] / 180.0 * np.pi  # Hue
hsv_mapped[..., 1] = hsv[..., 1]  # Saturation
hsv_mapped[..., 2] = hsv[..., 2]  # Value

# Reshape to 2D array
flat_img2 = hsv_mapped.reshape(-1, 3)

# Perform K-means clustering
for K2 in [2, 4, 8]:
    flat_img2 = np.float32(flat_img2)

    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1
    flags = cv2.KMEANS_RANDOM_CENTERS
    ret, label, center = cv2.kmeans(flat_img2, K2, None, criteria, 10, fl

    # Convert center to uint8
    center = np.uint8(center)

    # Map the labels to the center values
    res = center[label.flatten()]
    res1 = res.reshape(img.shape)

    # Convert back to BGR
    res1_hsv = cv2.cvtColor(res1, cv2.COLOR_BGR2HSV)

    # Display the result
    plt.figure()
    plt.imshow(cv2.cvtColor(res1_hsv, cv2.COLOR_HSV2RGB))
    plt.title(f'K={K2}')
```

```
    plt.axis('off')

plt.show()
```



K=8