COM4504

IMAGE SEGMENTATION USING KMEANS CLUSTERING FOR DOMINANT COLORS

by

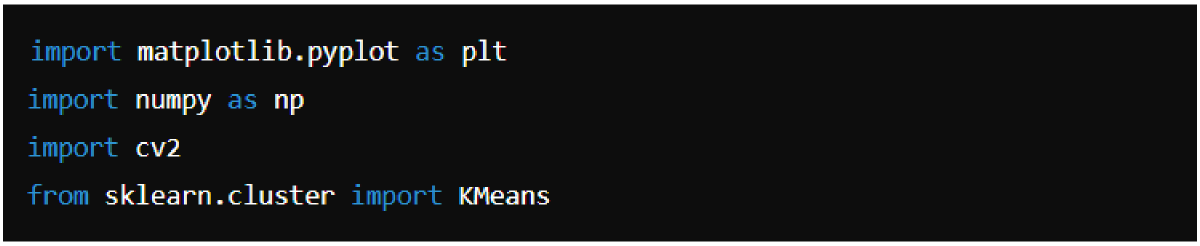
YUSUF ÖZCAN

INTRODUCTION

This report explains how to perform image segmentation using KMeans clustering algorithm. The provided Python code loads an image, converts it to RGB format, clusters the pixels into a specified number of color groups (clusters), and generates a segmented image where each segment is represented by one of the dominant colors.

STEP-BY-STEP EXPLANATION

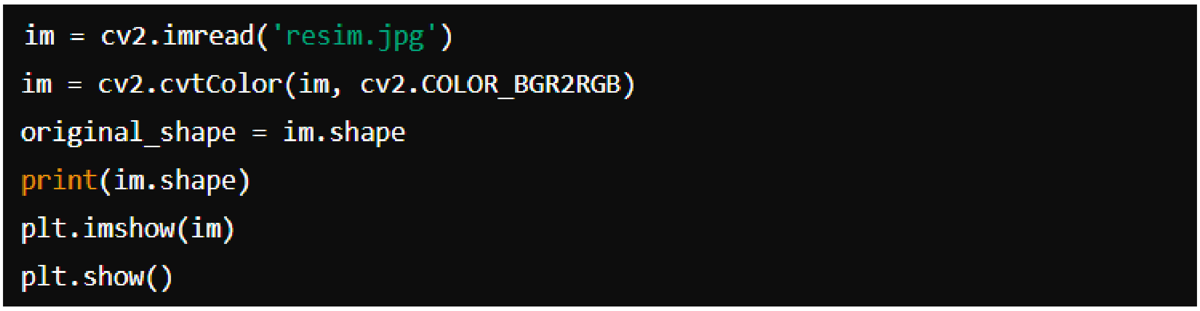
1. Importing Necessary Libraries



```python
import matplotlib.pyplot as plt
import numpy as np
import cv2
from sklearn.cluster import KMeans
```

At the beginning of the code, the necessary libraries for image processing (cv2), data processing (numpy), visualization (matplotlib), and clustering (sklearn.cluster.KMeans) are imported.

2. Loading and Converting The Image to RGB Format



```python
im = cv2.imread('resim.jpg')
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
original_shape = im.shape
print(im.shape)
plt.imshow(im)
plt.show()
```

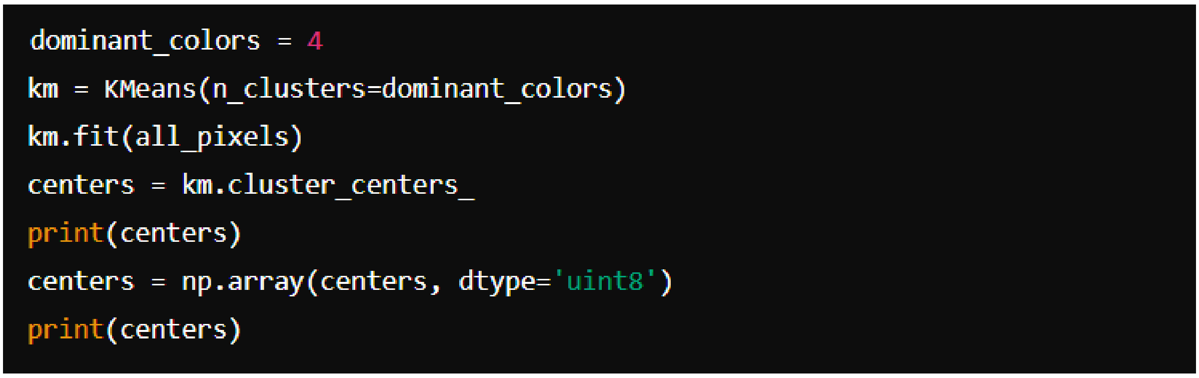
- The image file ‘**resim.jpg**’ is read using ‘**cv2.imread**’.

- The image is converted from BGR to RGB format using **‘cv2.cvtColor’**.
- The dimensions of the image (height, width, color channels) are obtained using **‘im.shape’** and printed.
- The original image is displayed using **‘plt.imshow’** and **‘plt.show’**.

3. Reshaping the Image Pixels

- The image pixels are reshaped into a 2D array where each pixel is represented by a triple (RGB). This is necessary for the KMeans clustering algorithm.
- The new shape of the pixel data is printed using **‘all_pixels.shape’**.

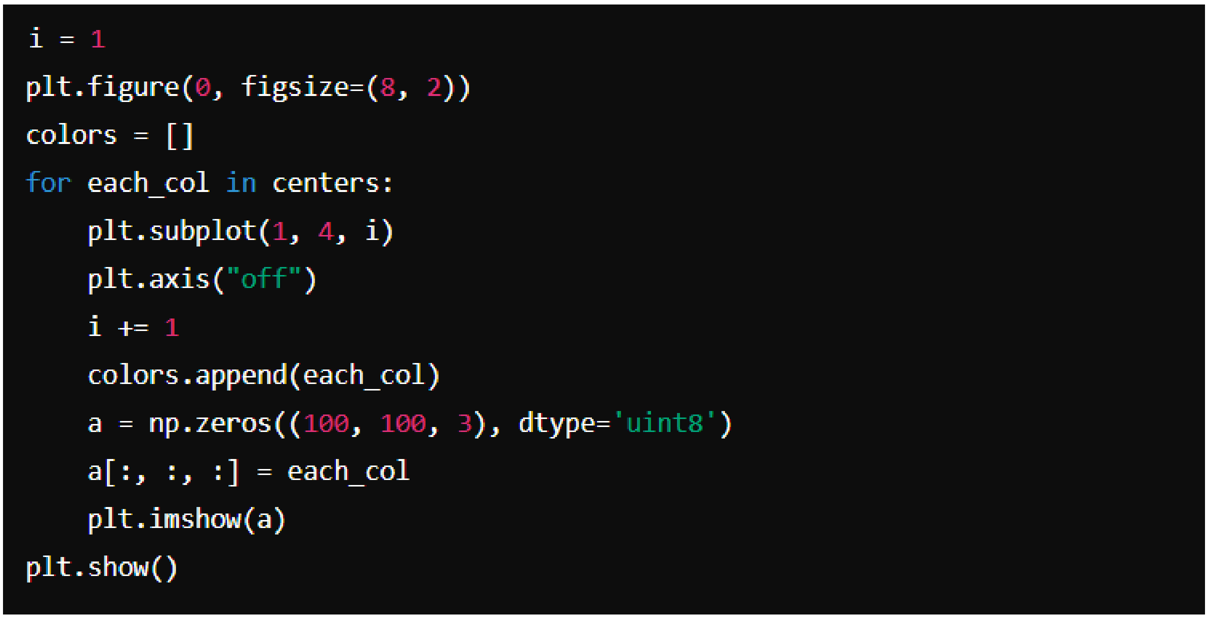
4. Determining Dominant Colors Using KMeans Clustering



```
dominant_colors = 4
km = KMeans(n_clusters=dominant_colors)
km.fit(all_pixels)
centers = km.cluster_centers_
print(centers)
centers = np.array(centers, dtype='uint8')
print(centers)
```

- The number of dominant colors is specified by the **‘dominant_colors’** variable (in this case,4).
- A **‘KMeans’** object is created and fitted to the pixel data (**‘km.fit(all_pixels)’**).
- The cluster center (dominant colors) are obtained using **‘km.cluster_centers_’** and converted to **‘uint8’** type before being printed.

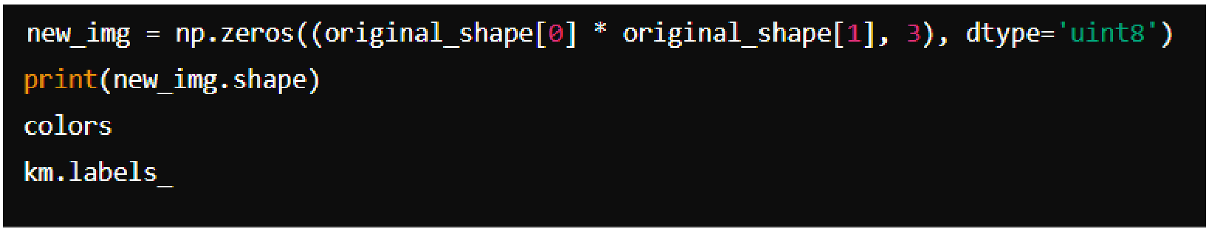
5. Visualizing Dominant Colors



```
i = 1
plt.figure(0, figsize=(8, 2))
colors = []
for each_col in centers:
    plt.subplot(1, 4, i)
    plt.axis("off")
    i += 1
    colors.append(each_col)
    a = np.zeros((100, 100, 3), dtype='uint8')
    a[:, :, :] = each_col
    plt.imshow(a)
plt.show()
```

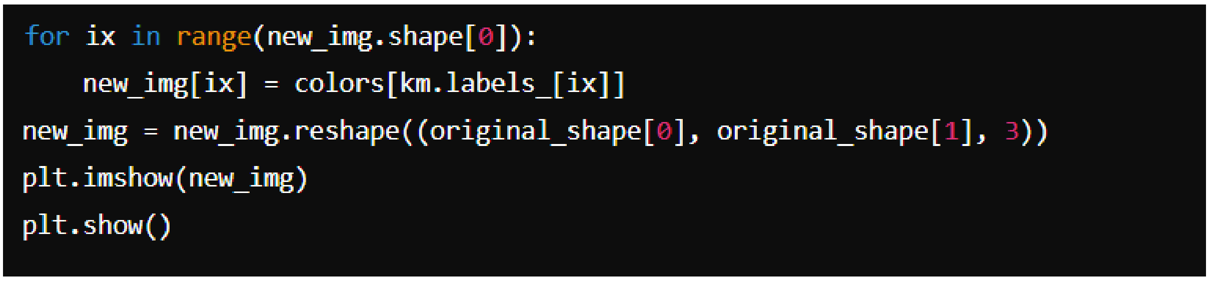
- The dominant colors are displayed as individual squares. **‘plt.subplot’** is used to show each color in a subplot.
- The dominant colors are visualized using **‘plt.imshow’** and **‘plt.show’ .**

6. Creating the New Segmented Image



```
new_img = np.zeros((original_shape[0] * original_shape[1], 3), dtype='uint8')
print(new_img.shape)
colors
km.labels_
```

- A new blank image (‘**new_img**’) is created, matching the original image dimensions.
- Using the **‘colors’** variable and **‘km_labels_’**, each pixel is assigned a new color.



```
for ix in range(new_img.shape[0]):
    new_img[ix] = colors[km.labels_[ix]]
new_img = new_img.reshape((original_shape[0], original_shape[1], 3))
plt.imshow(new_img)
plt.show()
```

- The new image is filled with the new colors for each pixel.

- The new image is reshaped to its original dimensions using **‘new_img.reshape’** and displayed using **‘plt.imshow’** and **‘plt.show’**.

CONCLUSION

This code performs image segmentation by using the KMeans clustering algorithm to divide an image into dominant colors. As a result, each pixel in the image is assigned to one of the dominant colors, creating a segmented image. This method is an effective approach for simple segmentation and identifying dominant colors in images.