Computer Vision

Name: Abhishek Choudhary

Enrollment No: 04619051622

Lab - 3 : Image Segmentation using K Means

OBJECTIVE

To perform Image segmentation using K means Clustering

Reading and Displaying Image

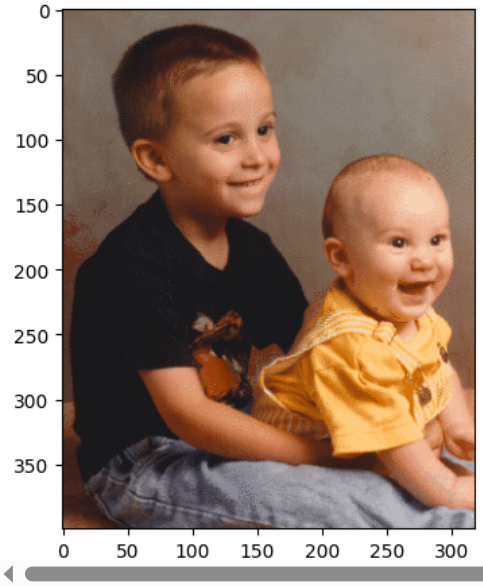
Reading and displaying an image is the first step before applying any modifications or analysis .To read and display an image in python we first load an image into Python using OpenCV and display it using Matplotlib. We convert the image from OpenCV's default BGR color format to the standard RGB format so that colors display correctly in matplotlib or other libraries.

```
import numpy as np
import cv2
import matplotlib.pyplot as plt


from google.colab import files

uploaded = files.upload()
```

```
Choose Files kids.png
• kids.png(image/png) - 134131 bytes, last modified: 4/14/2025 - 100% done
Saving kids.png to kids (1).png
```

```
img = cv2.imread("kids.png")
img=cv2.cvtColor(img ,cv2.COLOR_BGR2RGB)
plt.figure(figsize=(6.5,5))
plt.imshow(img)
img.shape
```

```
(400, 318, 3)
```



Function for Image Segmentation

THEORITICAL CONCEPTS

What is Image Segmentation?

1. Image segmentation is the process of dividing an image into parts or regions.
2. Pixels in the same region look similar in colour, brightness, or texture.

3. Segmentation makes it easier to understand or analyze the image (e.g., find objects, backgrounds, etc).
4. Used For:
    - Object detection (finding people, cars, etc.)
    - Medical imaging (finding tumors in scans)
    - Background removal
    - Robot vision
    - Image compression

What is Vectorization?

Vectorization refers to the flattening of pixels. We convert the image into a 2D array where each row is one pixel's RGB values. Shape becomes: (total_pixels, 3)

Image is a grid of pixels. Normally, an image is a 3D array: height × width × color channels (e.g 600×480×3).

We vectorize the image because algorithms like K-Means expect data as vectors (rows of features). So we reshape the image to feed it into the model.

Example:

A 4×4 image → 16 pixels

Vectorized shape → 16 × 3

Each row = [R, G, B] values of 1 pixel

What is K-Means Clustering?

K-Means is an unsupervised machine learning algorithm that groups data into K clusters based on similarity.

What is K?

K is the number of clusters (or groups) you want.

In images, K is number of dominant colors/regions we want to separate.

How K-Means Works (In Image Segmentation)

1. Input = Pixels as Points
2. Each pixel is a 3D point: [R, G, B]
3. Initialize K Cluster Centers randomly.
4. Start with K random colors as "centers" (called centroids).
5. Assign pixels to Nearest center.
6. For every pixel, find the closest center (based on color similarity, i.e., Euclidean distance).
7. Recalculate centers by updating each center to be the average color of all pixels assigned to it.
8. Repeat Steps 5–7.
9. Keep adjusting until pixels stop switching clusters or max iterations hit.
10. Final Result = Segmented Image
11. Replace each pixel's colour with the colour of its cluster's center. This simplifies the image into K colors (regions).

Example:

Consider an original Image: Choose K = 3. K-Means clusters all pixels into 3 color groups. Output image looks like it's made of 3 main colors = segmented image.

UNDERSTANDING THE FUNCTION

1. Image as a Matrix

- An image is stored as a 3D NumPy array: Height x Width x Channels (RGB).
- Each pixel has three values (Red, Green, Blue), ranging from 0 to 255.

2. Vectorization

- The image is reshaped into a 2D array: Number of Pixels x 3.
- This flattens the image so that each row is a pixel and each column is a color channel.
- Required for applying clustering on pixel colors.

3. Type Conversion

- KMeans in OpenCV expects data in float32, not integers.
- So pixel values (0–255) are converted from uint8 to float32.

4. KMeans Clustering

- KMeans groups the pixel colors into K clusters.
- Each cluster represents a dominant color in the image.
- Returns:
  - center: the RGB value of each cluster center (i.e., dominant colors).
  - label: tells you which cluster each pixel belongs to.

5. Clustering Criteria

- Sets the stopping condition for the algorithm:
  - Stop after 10 iterations or if the improvement is less than 1.0 (epsilon).
- Prevents infinite loops and ensures convergence.

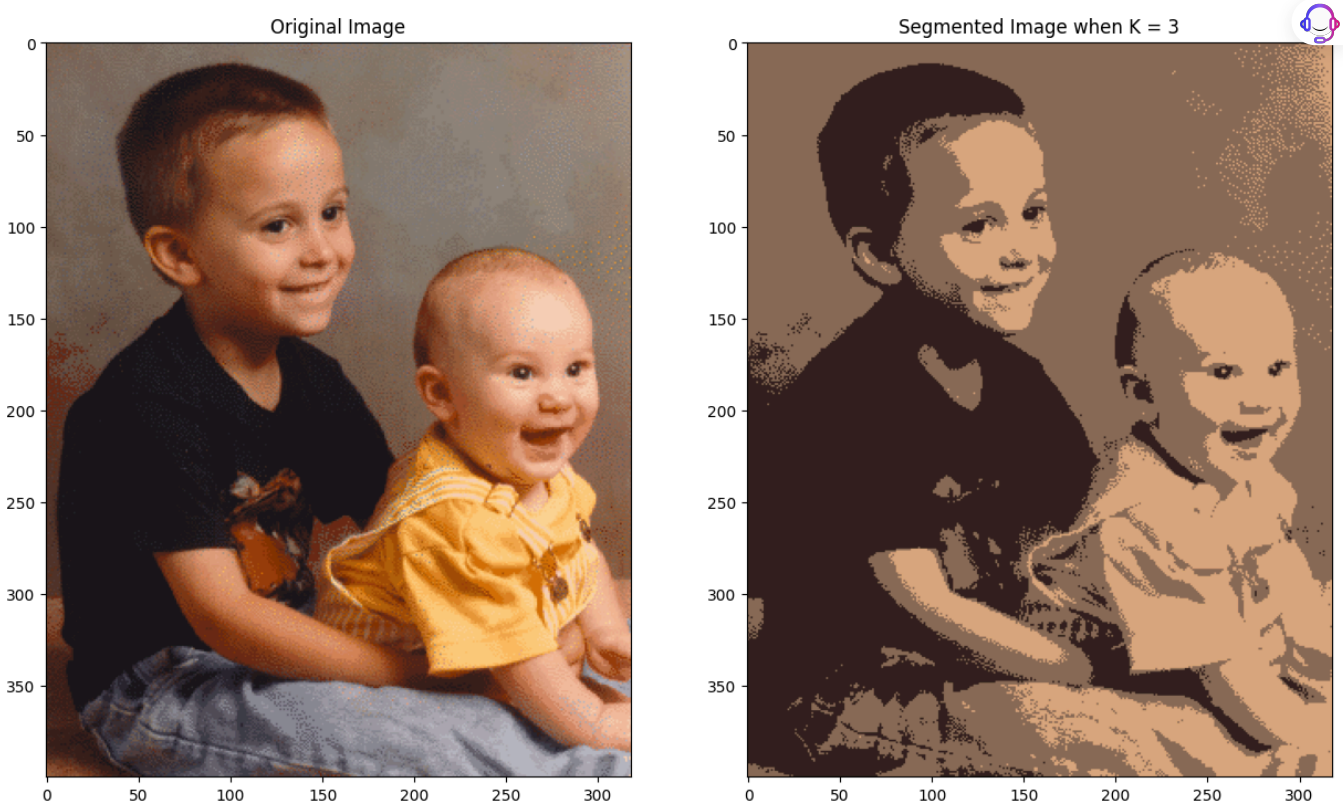
6. Pixel Replacement

- Each pixel is now assigned the RGB value of its cluster center.
- This effectively repaints the image using only K colors.
- The result is an image with simplified color regions - this is image segmentation.

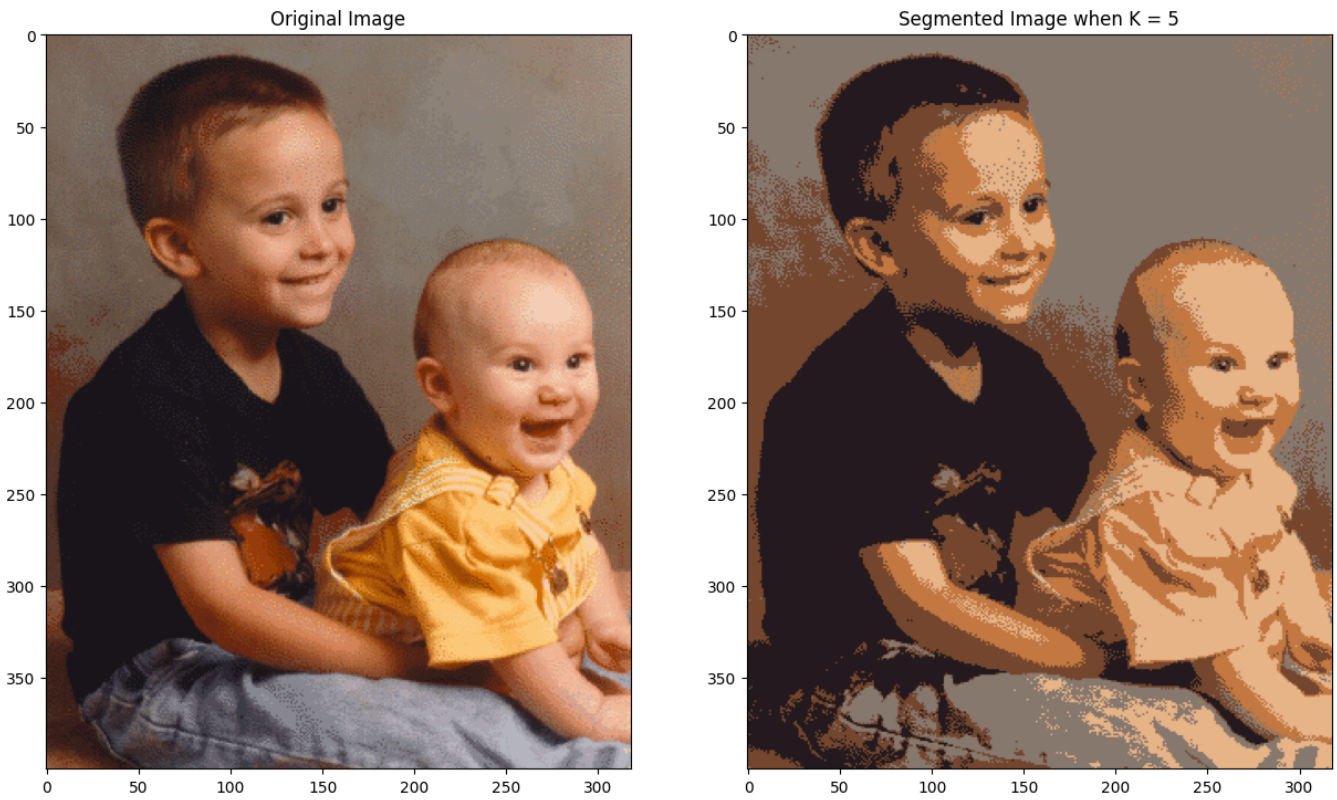
7. Visualization

- matplotlib.pyplot is used to show:
  - Original image.
  - Segmented image using K dominant colors.
- Helps visually understand the segmentation effect.

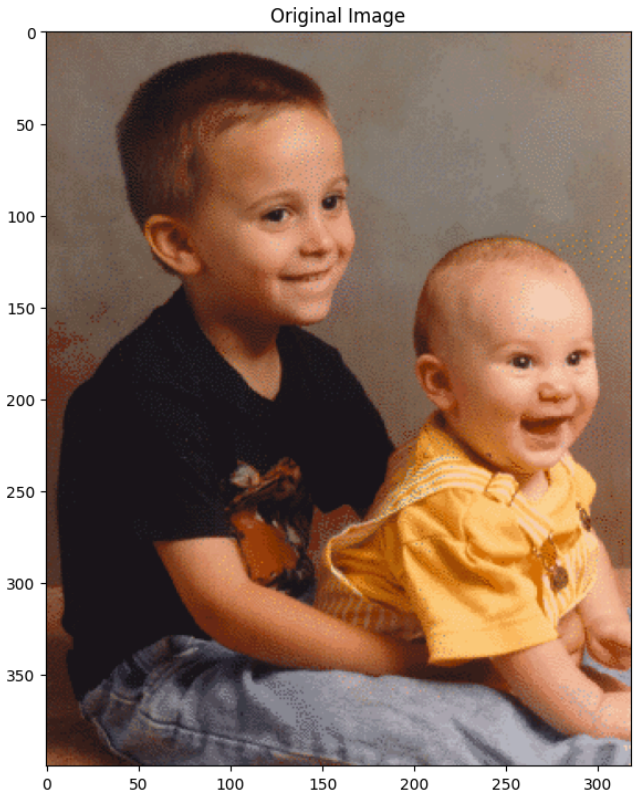
```
def image_segmentation(img, k, attempts=10):

    # Reshape image to 2D array of pixels and 3 color values (RGB)
    vectorized_img = img.reshape((-1, 3))
    vectorized_img = np.float32(vectorized_img)

    # Define criteria = (type, max_iter, epsilon)
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)

    # Apply KMeans
    ret, label, center = cv2.kmeans(vectorized_img, k, None, criteria, attempts, cv2.KMEANS_PP_CENTERS)

    # Convert back into uint8 and get result
    center = np.uint8(center)
    res = center[label.flatten()]
    result_image = res.reshape((img.shape))

    # Plot
    plt.figure(figsize=(15, 12))
    plt.subplot(1, 2, 1)
    plt.imshow(img)
    plt.title('Original Image')

    plt.subplot(1, 2, 2)
    plt.imshow(result_image)
    plt.title(f'Segmented Image when K = {k}')
    plt.show()


# k=3
image_segmentation(img, 3)
```

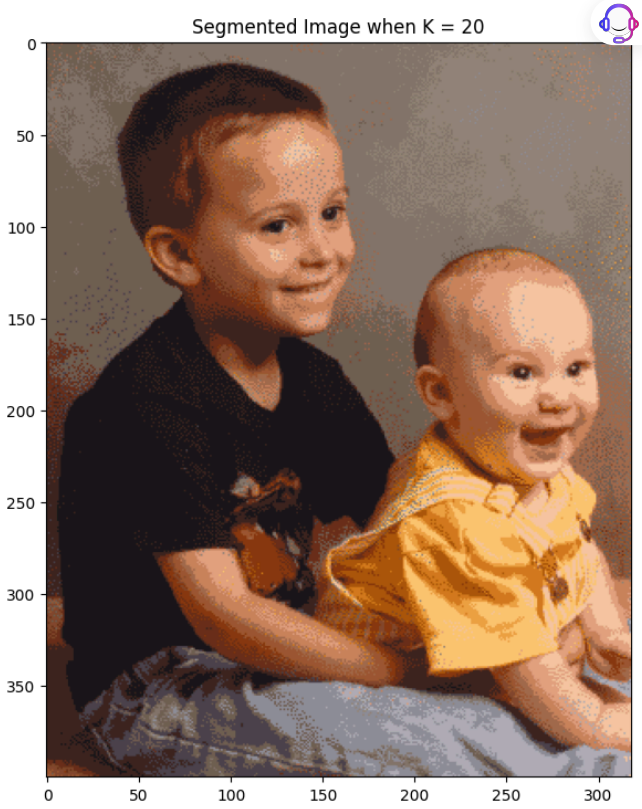



```
# k=5
image_segmentation(img, 5)
```




```
# k=20
image_segmentation(img, 20)
```


Original Image
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300


Segmented Image when K = 20
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300

CONCLUSION