Midterm Report

IMAGE SEGMENTATION USING CLUSTERING

Group 2

Our Member

ADINDA R.S.P
02/2141720158

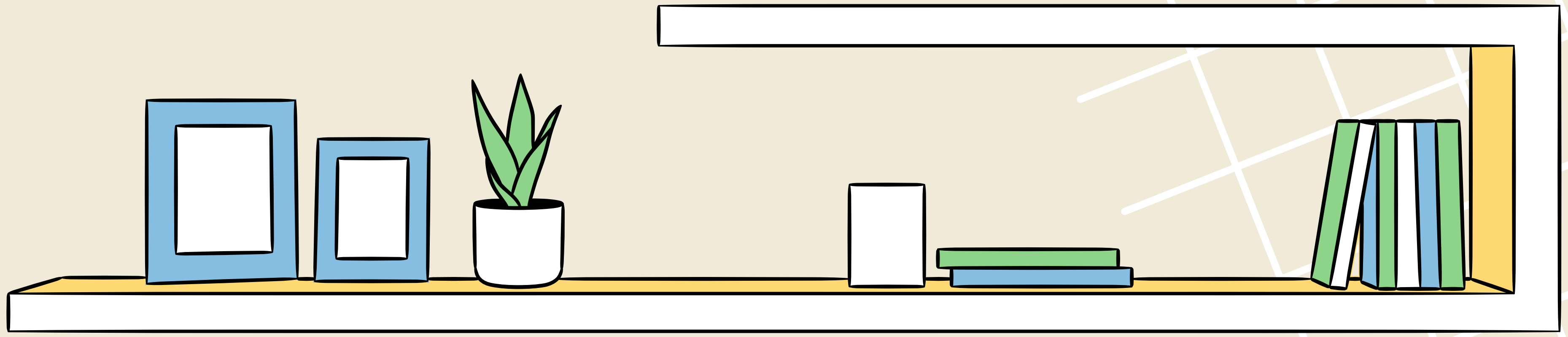
FAIZAL L
05/2141720246

KHAFILLAH A.S
11/2141720152

M. ADHIKA I. N
13/2141720267

M. FAHMI H
16/2141720153

Midterm Exam's Task & Rules



1. Select 5 license plate images for each group member from the provided dataset. [DOWNLOAD (https://storage.googleapis.com/kuliah_mah/dummy.zip)

2. Perform segmentation on the license plate images to enhance the characters on the license plates.

3. You can use the K-Means algorithm as explained in previous practical sessions or any other clustering algorithm.

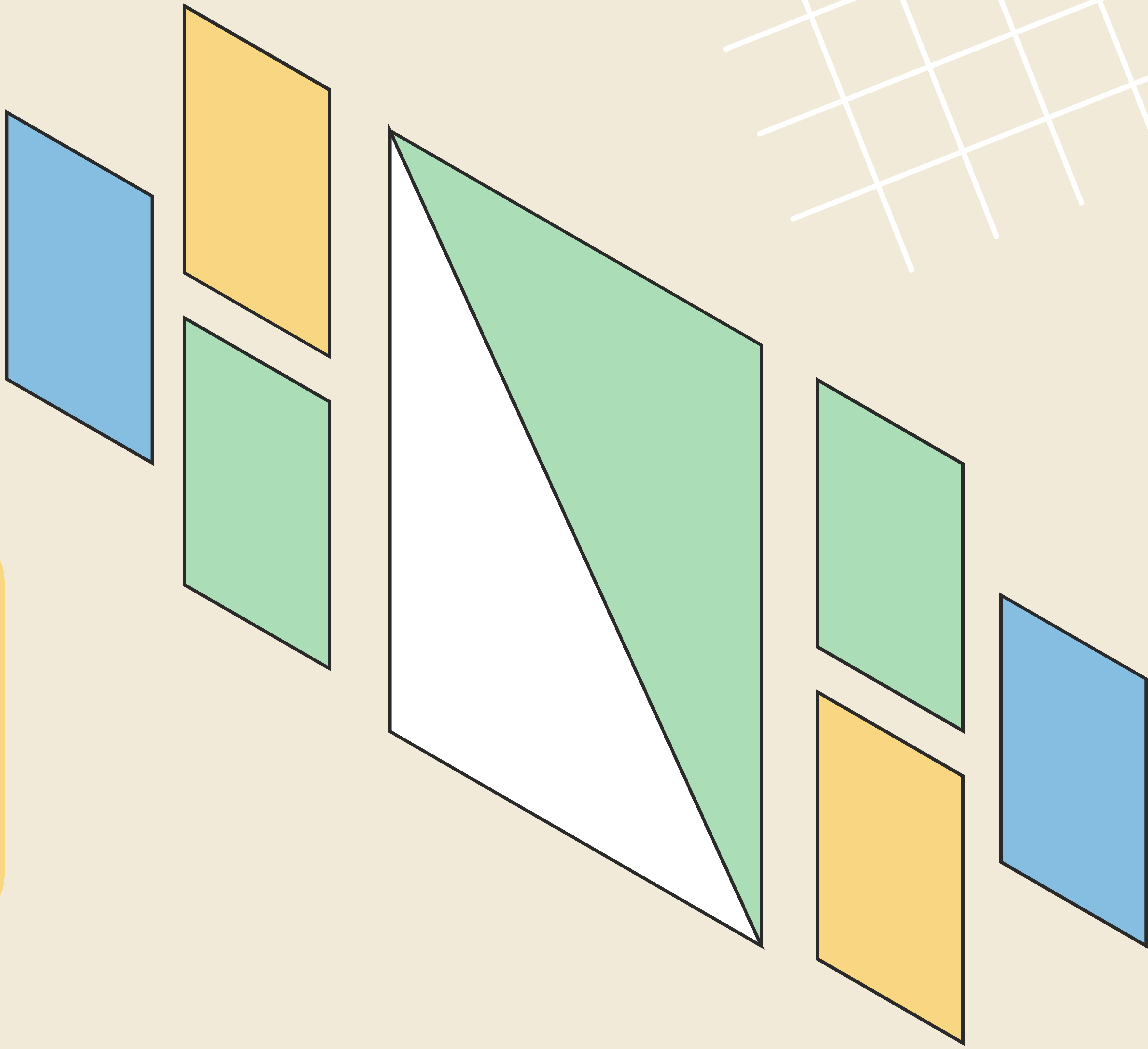
4. You are allowed to perform data preprocessing on the images, such as:
   - a Changing the color space
   - b Dimension reduction

5. Display a comparison of the images before and after segmentation.

What is Image segmentation?

Definition

Segmentation is one of the methods used to distinguish one object from another in an image. This can be done by grouping the pixel values in the image based on their color proximity.



Before start

step 1

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
```

step 2

```
pip install opencv-python
```

Installed the OpenCV-Python library for image processing

step 3

```
from google.colab import drive
drive.mount('/content/drive')
```

number 1, 2, 3, & 5

used for performing K-Means clustering

```
from sklearn.cluster import KMeans
```

used for performing K-Means clustering

```
def preprocess_image(image_path):
 image = cv2.imread(image_path)
```

```
blurred = cv2.GaussianBlur(image, (5, 5), 0)
 return image, blurred
```

Apply Gaussian blur to reduce noise

2. Specify the number of clusters for K-Means (enhancing characters)

Performing K-Means clustering for image segmentation

```
def kmeans_segmentation(image, k=2):
height, width, channels = image.shape
 pixels = image.reshape(-1, channels)
 kmeans = KMeans(n_clusters=k)
 kmeans.fit(pixels)
segmented_pixels = kmeans.cluster_centers_[kmeans.labels_].reshape((height, width, channels)).astype(np.uint8)
 return segmented_pixels
```

continue

```
# 5. Display images for comparison (before and after segmentation)
def display_images(original_image, segmented_image, title1, title2):
  plt.figure(figsize=(12, 6))
  plt.subplot(121), plt.imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)), plt.title(title1)
  plt.subplot(122), plt.imshow(cv2.cvtColor(segmented_image, cv2.COLOR_BGR2RGB)), plt.title(title2)
  plt.show()
```

→ Display images before and after segmentation

```
# 1. Define the paths to the license plate images for each group member
image_paths = [
  '/content/drive/MyDrive/1I Class Informatics Engineering/POLINEMA CLASS TASK/CLASS Semester 5/(MACHLEARN_TI) Machine Learning/Meet_8 (UTS)/Dummy/dummy/B2520XK_PNG.rf.4e22939a8917f509074397176d67c5e5.jpg',
  '/content/drive/MyDrive/1I Class Informatics Engineering/POLINEMA CLASS TASK/CLASS Semester 5/(MACHLEARN_TI) Machine Learning/Meet_8 (UTS)/Dummy/dummy/25-E-2101-PAD-06-21_jpeg.rf.fb688472f5ba9c0c445f9d9330f39508.jpg',
  '/content/drive/MyDrive/1I Class Informatics Engineering/POLINEMA CLASS TASK/CLASS Semester 5/(MACHLEARN_TI) Machine Learning/Meet_8 (UTS)/Dummy/dummy/BM3452A_PNG.rf.0f77ba0375f050d4cdd9b8a36c7ab92b.jpg',
  '/content/drive/MyDrive/1I Class Informatics Engineering/POLINEMA CLASS TASK/CLASS Semester 5/(MACHLEARN_TI) Machine Learning/Meet_8 (UTS)/Dummy/dummy/BG1980A_png.rf.0144d9ab803a1ef7c66fad4c8178699f.jpg',
  '/content/drive/MyDrive/1I Class Informatics Engineering/POLINEMA CLASS TASK/CLASS Semester 5/(MACHLEARN_TI) Machine Learning/Meet_8 (UTS)/Dummy/dummy/019_jpg.rf.e73938ec62074bbde822f7d5a084bdef.jpg'
]
```

Load and preprocess
the license plate
images

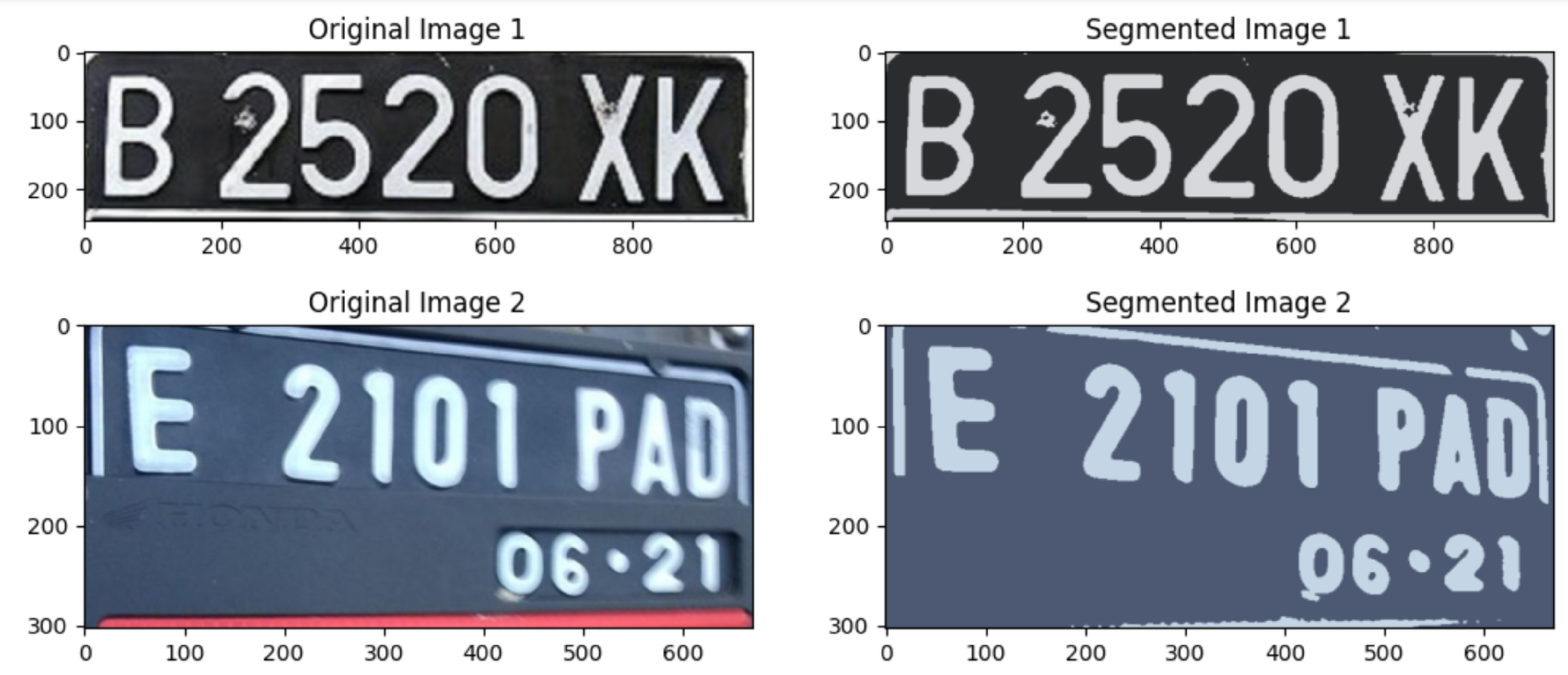
```
images = [preprocess_image(path) for path in image_paths]
k = 2 # Number of clusters for K-Means
segmented_images = [kmeans_segmentation(image, k) for _, image in images]
```

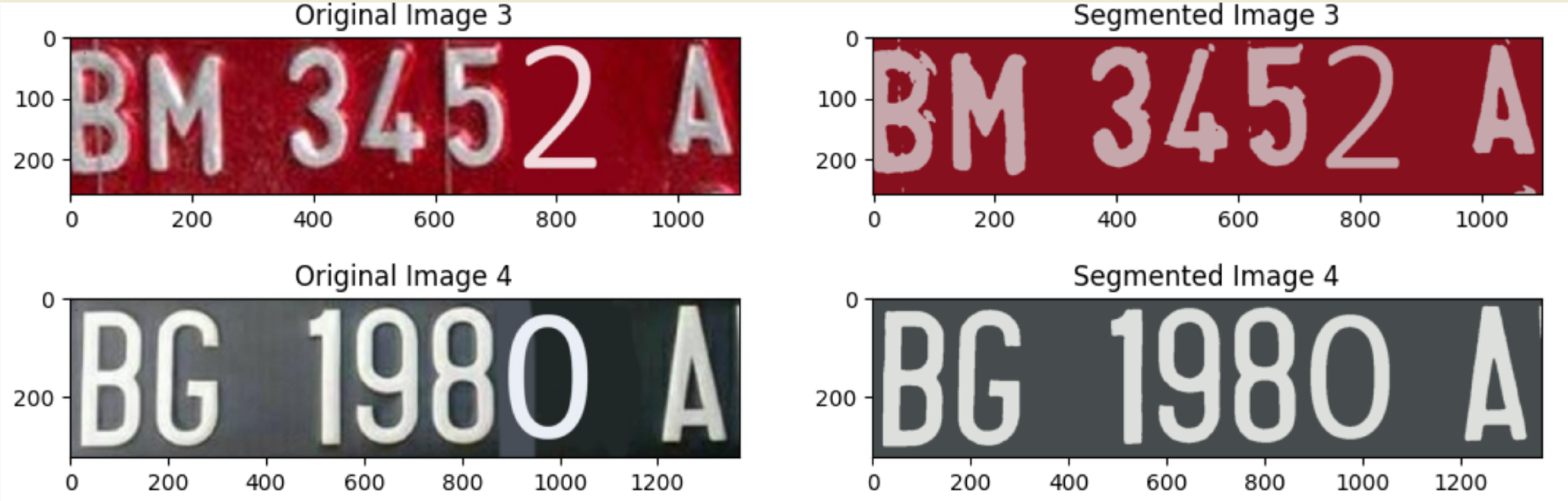
→ # 3. Load and preprocess the images, apply K-Means clustering, and display the results

```
# lanjutan nomor 2 untuk mendisplay B & A Segmentation
for i in range(len(image_paths)):
  display_images(images[i][0], segmented_images[i], f'Original Image {i+1}', f'Segmented Image {i+1}')
```

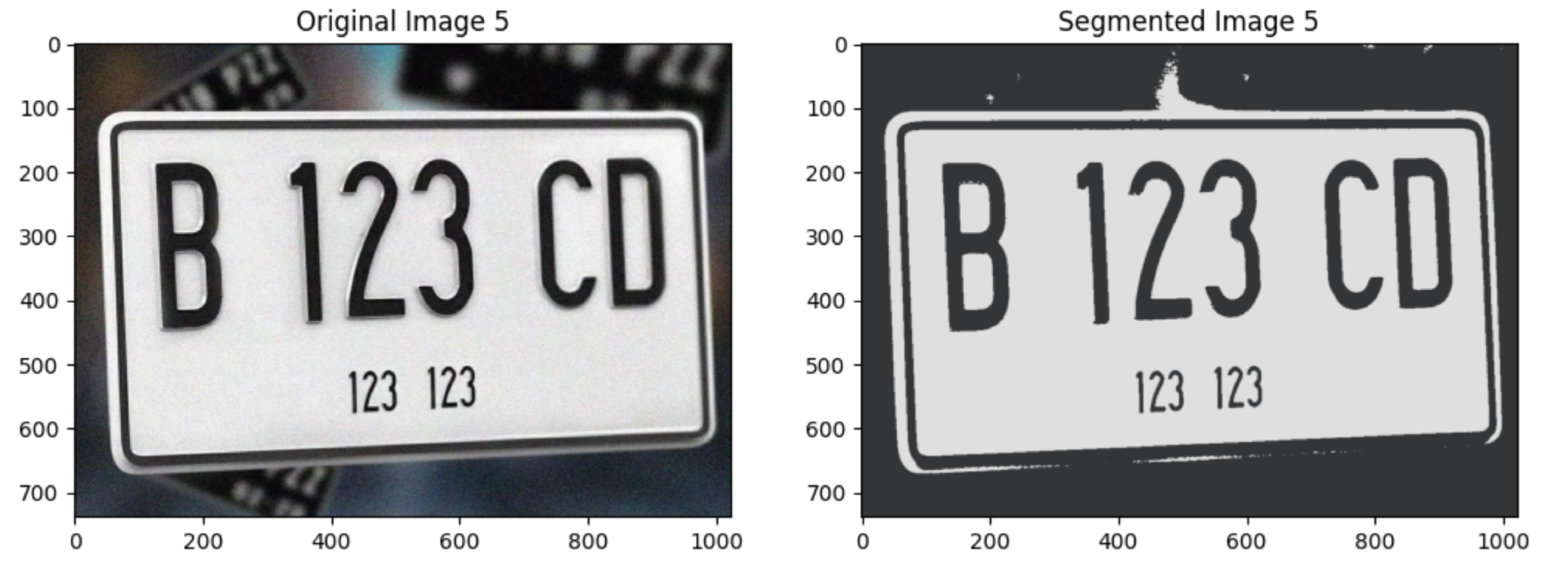
→ # Display images before and after segmentation

answer


Original Image 1
Segmented Image 1
B 2520 XK
0
100
200
0
200
400
600
800
Original Image 2
Segmented Image 2
E 2101 PAD
06·21
0
100
200
300
0
100
200
300
400
500
600


Original Image 3
Segmented Image 3
BM 3452 A
0
100
200
0
200
400
600
800
1000
Original Image 4
Segmented Image 4
BG 1980 A
0
200
0
200
400
600
800
1000
1200

answer



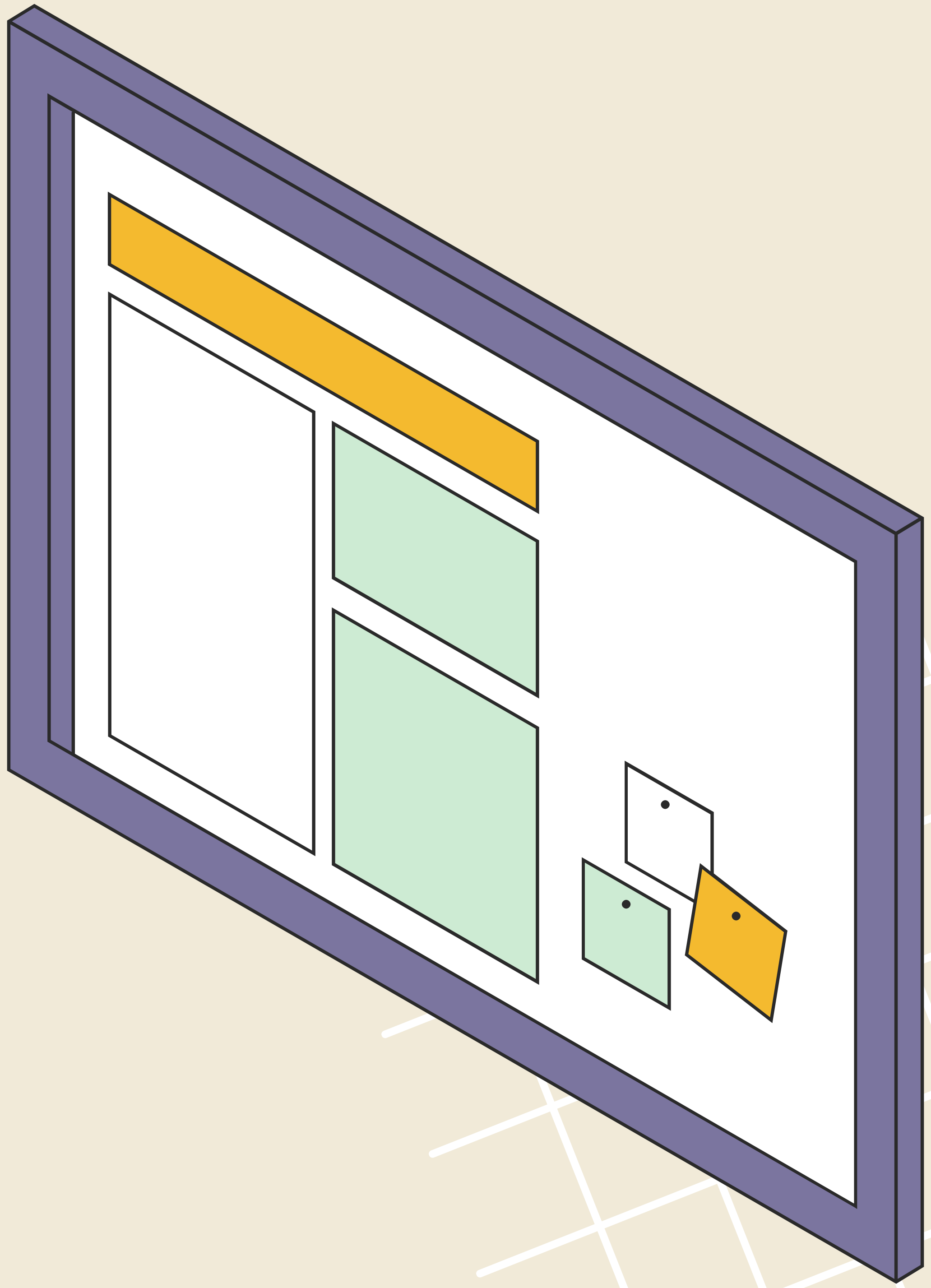
Original Image 5

Segmented Image 5

Number 4A

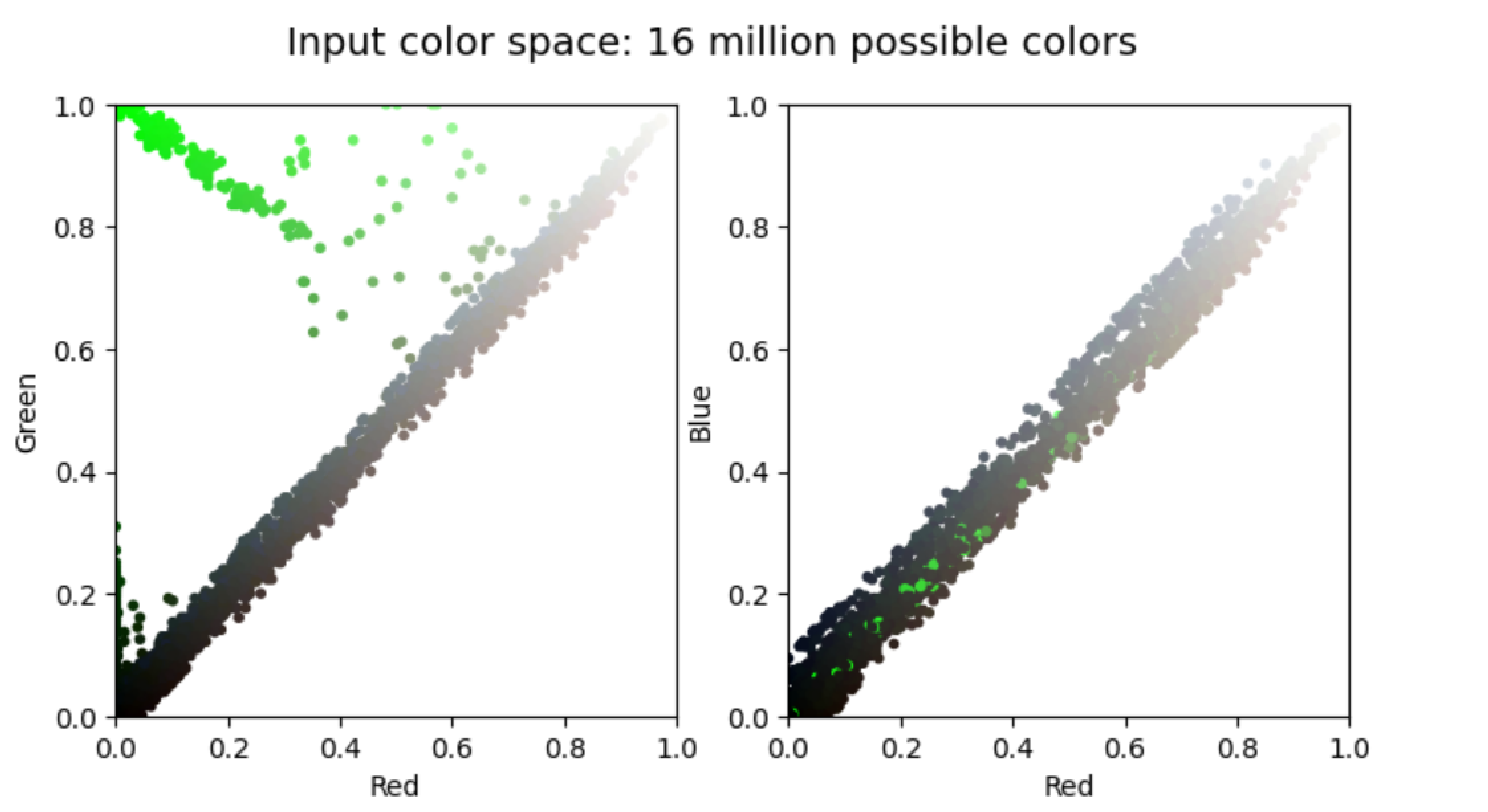
```
def plot_pixels(data, title, colors=None, N=10000):
    if colors is None:
        colors = data

    rng = np.random.RandomState(0)
    i = rng.permutation(data.shape[0])[:N]
    colors = colors[i]
    R, G, B = data[i].T

    fig, ax = plt.subplots(1, 2, figsize=(8, 4))

    ax[0].scatter(R, G, color=colors, marker='.')
    ax[0].set(xlabel='Red', ylabel='Green', xlim=(0, 1), ylim=(0, 1))

    ax[1].scatter(R, B, color=colors, marker='.')
    ax[1].set(xlabel='Red', ylabel='Blue', xlim=(0, 1), ylim=(0, 1))

    fig.suptitle(title, size=14)
```

```
plot_pixels(data, title='Input color space: 16 million possible colors')
```



This function plot_pixels creates a scatter plot using pixel data from an image or dataset in two separate color spaces, Red-Green and Red-Blue. The function enables for the amount of displayed pixels, color data, and title to be customized, and it shows its usage by viewing the "Input color space" with 16 million potential colors in a subplot with two scatter plots.

Answer for 4A



Input color space: 16 million possible colors

Number 4B

```
from sklearn.decomposition import PCA
from google.colab.patches import cv2_imshow # Import cv2_imshow for Colab
```

```
original_image = cv2.imread('/content/drive/MyDrive/1I Class Informatics Engineering/POLINEMA
CLASS TASK/CLASS Semester 5/(MACHLEARN_TI) Machine Learning/Meet_8
(UTS)/Dummy/dummy/BM9098V_PNG.rf.36b28183ada2d698e331cfffcfe2f1d0.jpg')
```

Read original image
using OpenCV

```
if original_image is not None:
  # Konversi gambar ke Grayscale
  grayscale_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)
```

Check if the image is loaded
successfully

```
  n_components = 10
 pca = PCA(n_components=n_components)
```

Create a PCA object
with the desired
number of components
(for example, 10)

```
reduced_image = pca.fit_transform(grayscale_image)
```

Apply PCA to image data

```
reconstructed_image = pca.inverse_transform(reduced_image)
```

Restore data to
original form

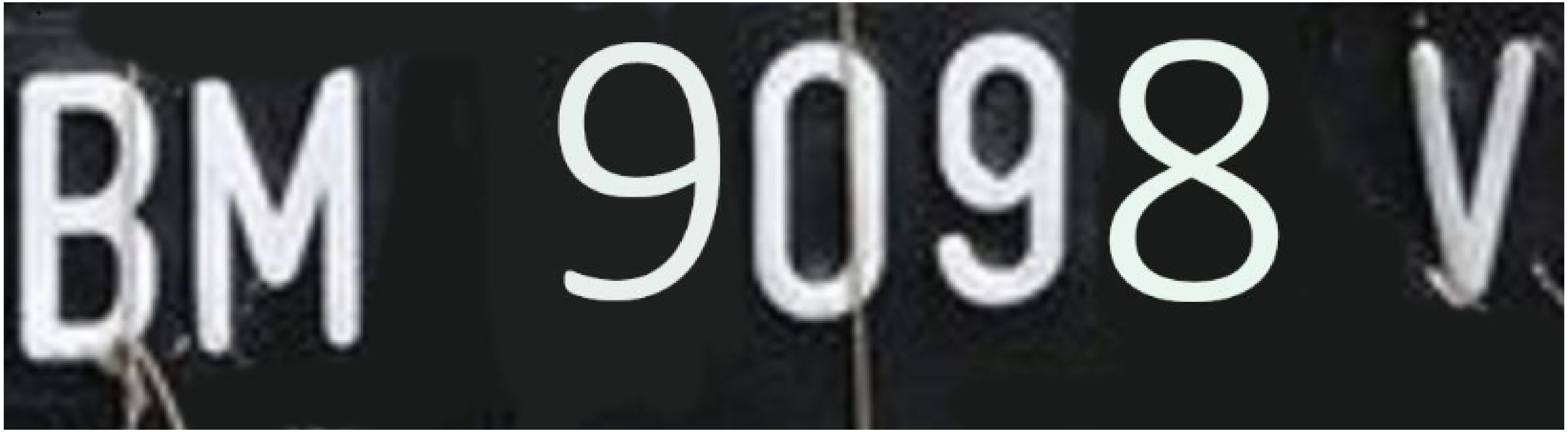
```
reco_image = reconstructed_image.astype(np.uint8)

# Tampilkan gambar asli
cv2_imshow(original_image)
print("Original Picture")
```

Change it back to image form

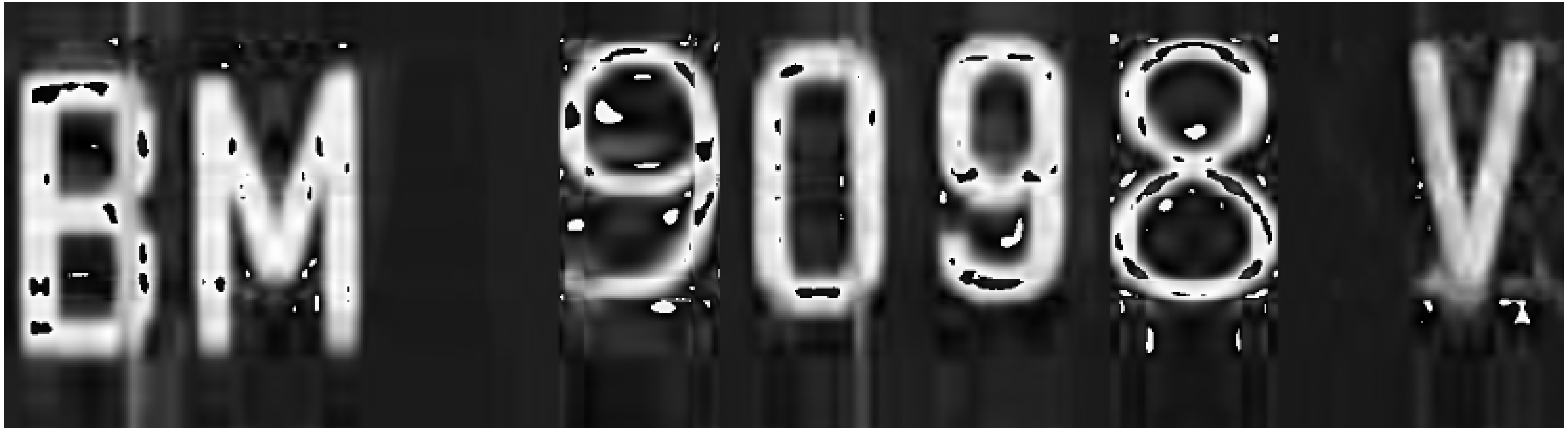
```
  cv2_imshow(cv2.cvtColor(reco_image, cv2.COLOR_GRAY2BGR))
  print("Picture that has been already get dimension reduction")
else:
  print("Fail to read picture!!.")
```

Display an image that has
been reduced in dimensions

Answer



Original Picture



Picture that has been already get dimension reduction

Conclusion

This report provides a clear definition of segmentation and code snippets for importing the necessary libraries, loading and preprocessing images, applying K-Means clustering, and showing a comparison of images before and after segmentation. The report also explains the use of PCA for dimensionality reduction. Overall, a comprehensive and well-implemented approach to image segmentation using clustering.

Thank you