Feature Extraction

#write a code for merging this following thing at one image and seperable results too line detection, canny edge detection, texture analysis, image segmentation and color histogram

```
!pip install opencv-python-headless numpy matplotlib
```

```
Requirement already satisfied: opencv-python-headless in
/usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (1.26.4)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib) (1.16.0)
```

```
#13
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Function to perform different image processing techniques
def process_image(image_path):
    # Read the input image
    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Line Detection using Probabilistic Hough Line Transform
    edges = cv2.Canny(gray, 50, 150, apertureSize=3)

    # Adjust parameters for clearer line detection
```

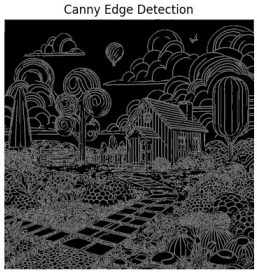
```
    lines = cv2.HoughLinesP(edges, 1, np.pi / 180, threshold=100,
minLineLength=50, maxLineGap=10)
    line_img = img.copy()

    if lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line[0]
            cv2.line(line_img, (x1, y1), (x2, y2), (0, 255, 0), 3)  #
Use green color and thicker lines

    # Canny Edge Detection
    canny = cv2.Canny(gray, 100, 200)

    # Texture Analysis using Local Binary Pattern (LBP)
    def lbp_texture_analysis(image):
        height, width = image.shape
        lbp_image = np.zeros_like(image)

        for y in range(1, height - 1):
            for x in range(1, width - 1):
                center = image[y, x]
                code = 0
                code |= (image[y - 1, x - 1] > center) << 7
                code |= (image[y - 1, x] > center) << 6
                code |= (image[y - 1, x + 1] > center) << 5
                code |= (image[y, x + 1] > center) << 4
                code |= (image[y + 1, x + 1] > center) << 3
                code |= (image[y + 1, x] > center) << 2
                code |= (image[y + 1, x - 1] > center) << 1
                code |= (image[y, x - 1] > center) << 0
                lbp_image[y, x] = code
        return lbp_image

    texture_analysis = lbp_texture_analysis(gray)

    # Image Segmentation using K-means clustering
    Z = img.reshape((-1, 3))
    Z = np.float32(Z)

    # Criteria and number of clusters
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,
10, 1.0)
    K = 3  # Number of clusters
    _, labels, centers = cv2.kmeans(Z, K, None, criteria, 10,
cv2.KMEANS_RANDOM_CENTERS)

    centers = np.uint8(centers)
    segmented_image = centers[labels.flatten()].reshape((img.shape))

    # Color Histogram
```

```
    hist_color = cv2.calcHist([img], [0], None, [256], [0, 256])
    hist_r = cv2.calcHist([img], [2], None, [256], [0, 256])
    hist_g = cv2.calcHist([img], [1], None, [256], [0, 256])
    hist_b = cv2.calcHist([img], [0], None, [256], [0, 256])

    # Plotting results
    fig, axs = plt.subplots(3, 3, figsize=(18, 12))

    axs[0, 0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    axs[0, 0].set_title('Original Image')
    axs[0, 0].axis('off')

    axs[0, 1].imshow(cv2.cvtColor(line_img, cv2.COLOR_BGR2RGB))
    axs[0, 1].set_title('Line Detection')
    axs[0, 1].axis('off')

    axs[0, 2].imshow(canny, cmap='gray')
    axs[0, 2].set_title('Canny Edge Detection')
    axs[0, 2].axis('off')

    axs[1, 0].imshow(texture_analysis, cmap='gray')
    axs[1, 0].set_title('Texture Analysis (LBP)')
    axs[1, 0].axis('off')

    axs[1, 1].imshow(cv2.cvtColor(segmented_image, cv2.COLOR_BGR2RGB))
    axs[1, 1].set_title('Image Segmentation (K-means)')
    axs[1, 1].axis('off')

    axs[1, 2].plot(hist_r, color='r')
    axs[1, 2].plot(hist_g, color='g')
    axs[1, 2].plot(hist_b, color='b')
    axs[1, 2].set_title('Color Histogram')
    axs[1, 2].set_xlim([0, 256])

    axs[2, 0].axis('off')
    axs[2, 1].axis('off')
    axs[2, 2].axis('off')

    plt.tight_layout()
    plt.show()

# Example usage
image_path = '/content/DALL·E 2024-11-14 14.41.04 - Create a high-
quality image of a colorful outdoor scene with diverse elements.
Include a small house or cabin with a wooden texture, surrounded by
gre.webp'  # Replace with your image path
process_image(image_path)
```

Original Image

Line Detection

Canny Edge Detection



Texture Analysis (LBP)

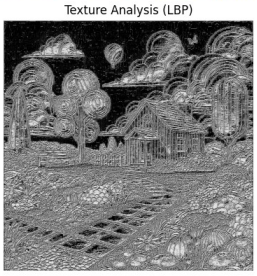


Image Segmentation (K-means)

Color Histogram