**Title:**

Apply edge detection (Sobel, Canny) and thresholding techniques on grayscale and color images.

**Aim:**

To apply Sobel and Canny edge detection techniques along with various thresholding operations on grayscale and color images using OpenCV.

**Description:**

**Edge detection** highlights boundaries by detecting sharp intensity changes in an image.

1. It reduces unnecessary details while retaining important structural features.
2. **Sobel operator** calculates gradients in horizontal (X) and vertical (Y) directions.
3. It uses 3x3 kernels and highlights edges aligned in those directions.
4. Sobel is efficient but sensitive to noise in complex or high-frequency images.
5. **Canny edge detection** is a multi-stage algorithm for accurate edge detection.
6. It starts with **Gaussian blur** to remove noise and smooth the image.
7. Then it computes **gradient magnitude and direction** using operators like Sobel.
8. **Non-maximum suppression** is used to thin edges to 1-pixel width.
9. **Double thresholding** classifies edges into strong and weak edges.
10. **Edge tracking by hysteresis** links weak edges to strong ones if connected.
11. Canny gives better results in terms of edge clarity and accuracy.
12. **Thresholding** converts images into binary form based on intensity values.
13. It can be applied to **grayscale** or each **RGB channel** of a color image.
14. Techniques like **global**, **adaptive**, and **Otsu’s thresholding** are used depending on the image type and lighting.

**Source Code:**

**NOTE:**

* The comments provided in this code are written only for easy understanding of each line.
* These comments are NOT mandatory to be written in the observation or record book.
* Use them only as a reference to understand what each part of the code does.

# Import the OpenCV library for image processing tasks like reading, transforming, and filtering images

import cv2

# Import NumPy, which is used for handling arrays, matrices, and numerical operations in image processing

import numpy as np

# Import pyplot from matplotlib, which is used to display images and visualizations in Python

from matplotlib import pyplot as plt

# ---------------------- Load the Image ----------------------

# Load the image file named 'loki.jpg' into a variable called 'image' using OpenCV's imread function

image = cv2.imread('loki.jpg')

# Check if the image is loaded successfully. If not, print an error and exit the program.

if image is None:

print("Image not found.") # If image is not found or path is incorrect

exit() # Exit the program

# ---------------------- Convert to Grayscale ----------------------

# Convert the image from BGR (default color format in OpenCV) to grayscale using cvtColor

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# ---------------------- Apply Sobel Edge Detection ----------------------

# Apply Sobel filter to detect edges in the X-direction (horizontal changes)

sobelx = cv2.Sobel(gray, cv2.CV\_64F, 1, 0, ksize=3)

# Apply Sobel filter to detect edges in the Y-direction (vertical changes)

sobely = cv2.Sobel(gray, cv2.CV\_64F, 0, 1, ksize=3)

# Combine the Sobel X and Y results to get a complete edge map using magnitude

sobel\_combined = cv2.magnitude(sobelx, sobely)

# ---------------------- Apply Canny Edge Detection ----------------------

# Apply Canny edge detection algorithm with threshold values 100 and 200

canny = cv2.Canny(gray, 100, 200)

# ---------------------- Apply Thresholding on Grayscale ----------------------

# Apply binary thresholding on the grayscale image.

# All pixel values above 120 become 255 (white), others become 0 (black)

\_, thresh\_gray = cv2.threshold(gray, 120, 255, cv2.THRESH\_BINARY)

# ---------------------- Apply Thresholding on Color Channels Separately ----------------------

# Split the color image into its three channels: Blue, Green, and Red

b, g, r = cv2.split(image)

# Apply binary thresholding on each channel separately

\_, thresh\_b = cv2.threshold(b, 120, 255, cv2.THRESH\_BINARY) # Threshold Blue

\_, thresh\_g = cv2.threshold(g, 120, 255, cv2.THRESH\_BINARY) # Threshold Green

\_, thresh\_r = cv2.threshold(r, 120, 255, cv2.THRESH\_BINARY) # Threshold Red

# Merge the thresholded channels back into a single color image

merged\_thresh = cv2.merge((thresh\_b, thresh\_g, thresh\_r))

# ---------------------- Display Results Using Matplotlib ----------------------

# Titles for each subplot image

titles = ['Original', 'Gray', 'Sobel X+Y', 'Canny', 'Threshold Gray', 'Threshold Color']

# Store all the images to be displayed in a list

images = [image, gray, sobel\_combined, canny, thresh\_gray, merged\_thresh]

# Create a figure with a specific size (width=12 inches, height=8 inches)

plt.figure(figsize=(12, 8))

# Loop through each image and display it in a subplot

for i in range(6):

plt.subplot(2, 3, i+1) # Arrange images in a 2x3 grid

if len(images[i].shape) == 2:

# If the image is grayscale (2D), use gray colormap

plt.imshow(images[i], cmap='gray')

else:

# If the image is color (3D), convert from BGR to RGB before displaying

plt.imshow(cv2.cvtColor(images[i], cv2.COLOR\_BGR2RGB))

plt.title(titles[i]) # Set the title for each subplot

plt.axis('off') # Hide axis ticks for clean view

# Adjust layout to prevent overlap of titles and plots

plt.tight\_layout()

# Display all the plotted images in a single window

plt.show()

**Output:**

The following outputs were generated after executing the program:

1. **Original Image** – Displays the input color image (loki.jpg) in RGB format.
2. **Grayscale Image** – Converted version of the original image in grayscale for further processing.
3. **Sobel Edge Detection (X + Y Combined)** – Displays the edges detected using the Sobel operator by combining horizontal and vertical gradients.
4. **Canny Edge Detection** – Displays precise and clean edges detected using the Canny algorithm with thresholds 100 and 200.
5. **Thresholded Grayscale Image** – Displays a binary version of the grayscale image where pixel values above 120 are set to 255 (white), and others to 0 (black).
6. **Thresholded Color Image** – Shows the thresholding applied individually to R, G, B channels and then merged into a single color output.

Each image was displayed using Matplotlib in a 2x3 grid layout for clear comparison and understanding.

**Result:**

The Python program was successfully executed using OpenCV and Matplotlib. Sobel and Canny edge detection techniques were applied on grayscale images, and thresholding was performed on both grayscale and color images to obtain binary outputs. All results were displayed effectively.