**EXPERIMENT:**

**AIM:** Write a program to apply canny edge detection on an image.

A. Perform Convolution operation on image using mask.

B. Implement pre-processing techniques on image.

**SOFTWARE:** Python(spyder)

**THEORY:**

**Canny edge detector has 5 stage-**

1. It is a multi-stage algorithm and we will go through each stages.

**2. Noise Reduction**

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter. We have already seen this in previous chapters.

**3. Finding Intensity Gradient of the Image**

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction ( Gx) and vertical direction ( Gy). From these two images, we can find edge gradient and direction for each pixel as follows:

Edge\_Gradient(G)=G2x+G2y−−−−−−−√Angle(θ)=tan−1(GyGx)

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

4. **Non-maximum Suppression**

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Point A is on the edge ( in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed ( put to zero).In short, the result you get is a binary image with "thin edges".

**5. Hysteresis Thresholding**

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded.



The edge A is above the maxVal, so considered as "sure-edge". Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select minVal and maxVal accordingly to get the correct result.

**CONVOLUTION:**

Convolution is a basic mathematical operation that several image processing operators use. Convolution is a method of multiplying two arrays of integers, often of different sizes but of the same dimensionality, to produce a third array of the same dimensionality.

**PROGRAM:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load image

image = cv2.imread(r"C:\Users\HP\Desktop\cv img.jpg", 1)

if image is None:

print("Error: Unable to open image.")

else:

# Convert to grayscale

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

# Apply GaussianBlur to reduce noise and improve edge detection

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

# Apply Canny edge detection

edges = cv2.Canny(blurred, 50, 150)

# Define a convolution kernel (e.g., a simple edge detection kernel)

kernel = np.array([[-1, -1, -1],

[-1, 8, -1],

[-1, -1, -1]])

# Apply convolution operation

convolution\_result = cv2.filter2D(image, -1, kernel)

# Preprocess image

preprocessed\_image = cv2.GaussianBlur(image, (5, 5), 0)

# Set up a 2x2 subplot grid

plt.subplot(2, 2, 1)

plt.imshow(image[...,::-1])

plt.title('Original Image')

plt.subplot(2, 2, 2)

plt.imshow(edges, cmap='gray')

plt.title('Canny Edge Detection')

plt.subplot(2, 2, 3)

plt.imshow(convolution\_result[...,::-1])

plt.title('Convolution Result')

plt.subplot(2, 2, 4)

plt.imshow(preprocessed\_image[...,::-1])

plt.title('Preprocessed Image')

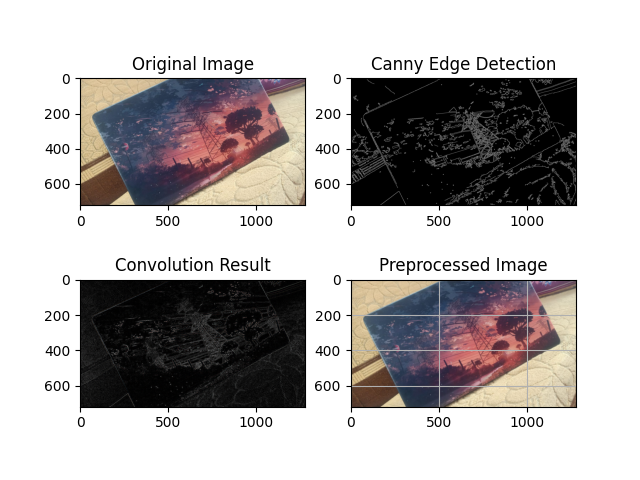
# Display the grid

plt.grid(True)

# Show the combined plot

plt.show()

**Output:**



**CONLUSION:** In this practical we perform convolutional different operation on images And detected edges by using canny edge detection.