**EXPERIMENT NO. 11**

**EDGE DETECTION**

**EXPERIMENT NO. 11**

**AIM: -** To perform edge detection using Canny edge and sobel edge detection algorithm

**OBJECTIVES:**

1. To use and understand how the algorithm works.
2. To extract edges form image.
3. To check the accuracy of algorithm.

**EQUIPMENTS/SOFTWARE:** Python

**THEORY: -**

Sobel edge detection is a commonly used technique in computer vision and image processing to detect edges in an image. It works by convolving the image with a small kernel, typically a 3x3 or 5x5 matrix, called the Sobel operator.

The Sobel operator consists of two 3x3 or 5x5 matrices, one for horizontal changes and the other for vertical changes. These two matrices are used to compute the gradient magnitude and direction of the image pixels.

The gradient magnitude indicates how strong the edge is, and the gradient direction gives the direction of the edge. By thresholding the gradient magnitude, we can separate the edges from the non-edges in the image.

Sobel edge detection is a simple and effective method for detecting edges in images and is widely used in various applications such as object recognition, segmentation, and tracking.

Canny edge detection is an edge detection algorithm named after its inventor John F. Canny. It is a multi-stage algorithm that is widely used in computer vision and image processing applications to detect edges in an image while minimizing the detection of spurious responses.

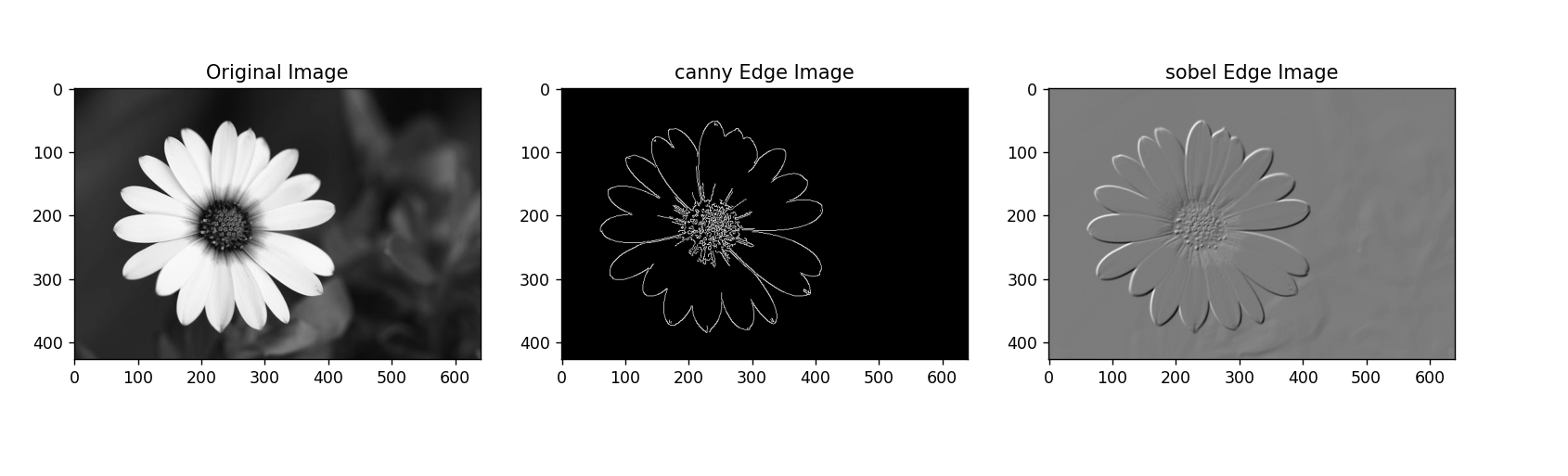
The Canny edge detection algorithm involves the following steps:

* Smoothing: The input image is first smoothed with a Gaussian filter to remove noise and small details that may not be relevant to the edges.
* Gradient calculation: The smoothed image is then convolved with the Sobel kernel to obtain the gradient of the image. The gradient provides the direction and magnitude of the image intensity changes.
* Non-maximum suppression: The gradient magnitude and direction are used to suppress the non-maximum edges, leaving only the strongest edges. This is done by comparing the gradient magnitude of each pixel with its neighboring pixels along the gradient direction. If the magnitude of the pixel is the highest, it is considered an edge pixel, otherwise, it is suppressed.
* Thresholding: The remaining edges are then thresholded to remove weak edges and noise. A high threshold is used to select strong edges, while a low threshold is used to connect these edges to form continuous contours.
* Hysteresis thresholding: This step is an extension of the thresholding step, where the remaining edges are classified into strong, weak, and non-edges based on their gradient magnitudes. The weak edges are then connected to the strong edges, forming continuous edges.
* The Canny edge detection algorithm is robust and produces high-quality edge maps with well-connected contours. It is widely used in various applications, including object detection, segmentation, and recognition, among others.

**Code:-**

import cv2  
import matplotlib.pyplot as plt  
  
img = cv2.imread('D:\\college related\\Third year\\sem6\\IPMV\\CODE\\flower.jpg',0)  
edges = cv2.Canny(img,100,200)  
  
sobelx = cv2.Sobel(img, cv2.CV\_64F, 1, 0, ksize=5)  
sobely = cv2.Sobel(img, cv2.CV\_64F, 0, 1, ksize=5)  
sobel = cv2.addWeighted(sobelx, 0.5, sobely, 0.5, 0)  
  
plt.subplot(1,3,1)  
plt.imshow(img,cmap = 'gray')  
plt.title('Original Image')  
  
plt.subplot(1,3,2)  
plt.imshow(edges,cmap = 'gray')  
plt.title('canny Edge Image')  
  
plt.subplot(1,3,3)  
plt.imshow(sobel,cmap = 'gray')  
plt.title('sobel Edge Image')  
plt.show()

**Output:-**

****

**CONCLUSION** :- Edge detection using Canny edge and sobel edge detection algorithm was performed successfully.