CANNY EDGE DETECTION

COMPUTER VISION

AD23402

SUBMITTED BY:

MENAKA.J

2023510025

***INTRODUCTION:***

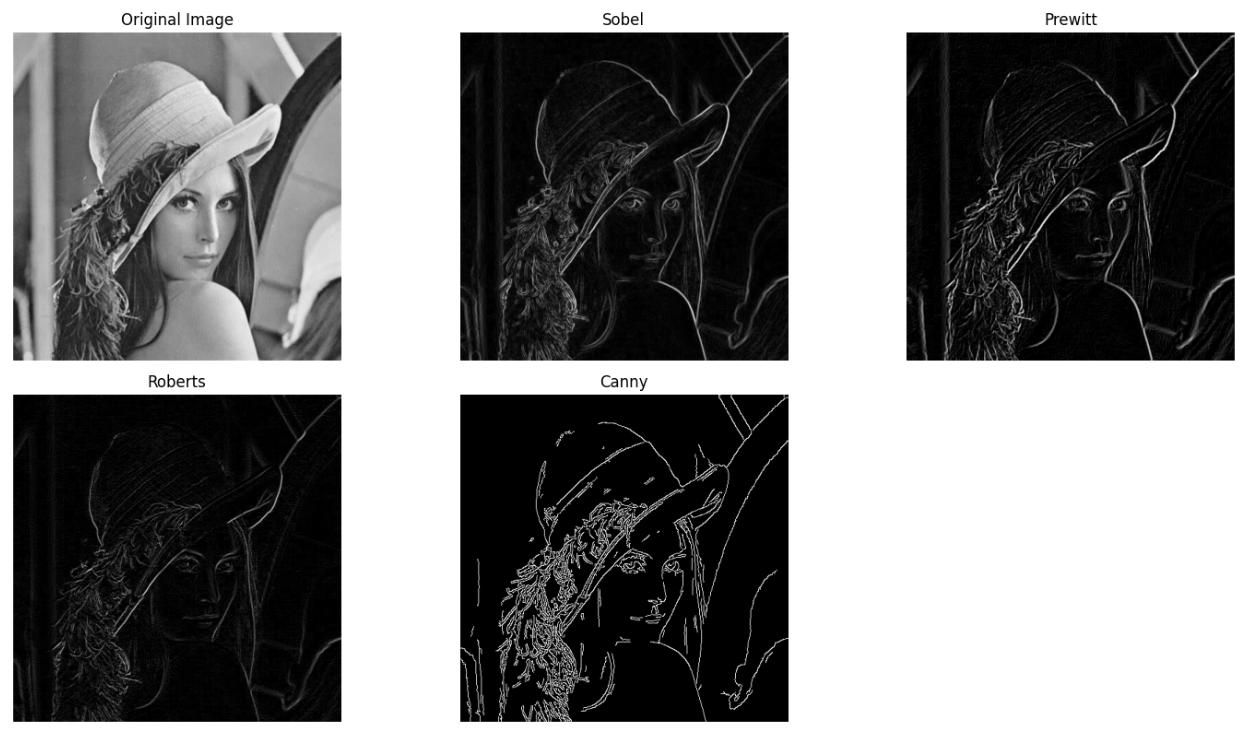
* Edge detection is a fundamental technique in image processing and computer vision.
* It aims to identify points in a digital image where brightness changes sharply, often corresponding to object boundaries.
* Canny Edge Detection, developed by John F. Canny in 1986, is one of the most widely used edge detection algorithms due to its optimal accuracy and reliability.
* It is a set of several procedures to find almost all the edges that could be potentially present in the image in a very precise manner.
* It includes several stages, starting from smoothing using a Gaussian filter, then computing the gradients, thinning the responses via a method called non-maximum suppression, and lastly hysteresis thresholding to connect the edges properly and eliminate the weak ones.

***NEED FOR CANNY:***

* We can detect edge in the image using many edge detection technique
  + 1. Robert’s edge detection
    2. Sobel edge detection
    3. Prewitt edge detection
* These traditional methods often suffer from issues such as noise sensitivity, poor localization, and multiple responses to a single edge.
* Although we have many techniques to find edge in an image, Canny gave better edges by ignoring week edges and consider only strong edges and edges connected to it.
* Canny also has an advantage since it use the concept of “Thresholding” in image.

***REASONS FOR CANNY OVER OTHER METHODS:***

* Good detection (low error rate)
* Good localization (accurate position)
* Minimal response (one response per edge)

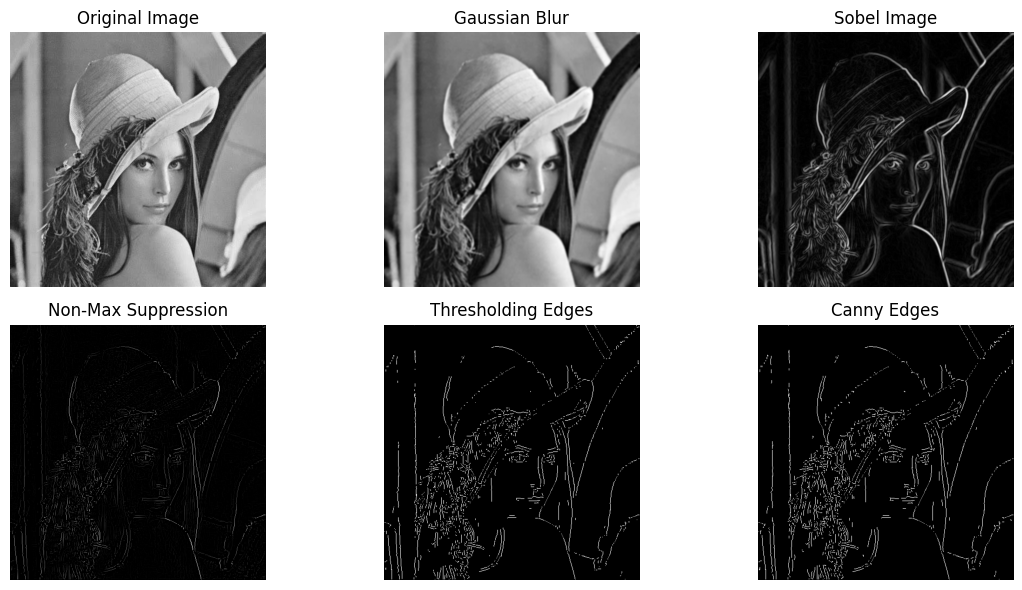


From this image, we can observe that Canny gave better edges than all other edge detection techniques.

***STEPS IN CANNY EDGE DETECTION:***

1. **Gaussian Smoothing** – reduces noise.
2. **Gradient Magnitude[Sobel Filter]** – detects intensity changes.
3. **Non-Maximal Suppression** – thins edges.
4. **Double Thresholding** – classifies strong and weak edges.
5. **Final Canny Edges** – tracks edges by hysteresis to finalize.

Before applying gaussian blur , convert the image to Grayscale image.



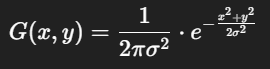
***WORKFLOW:***

***STEP 1 – GAUSSIAN SMOOTHING:***

**Purpose:**  
To reduce noise and minor variations in the image to prevent false edge detection.

**How:**  
Convolve the image with a Gaussian filter.

**Formula:**

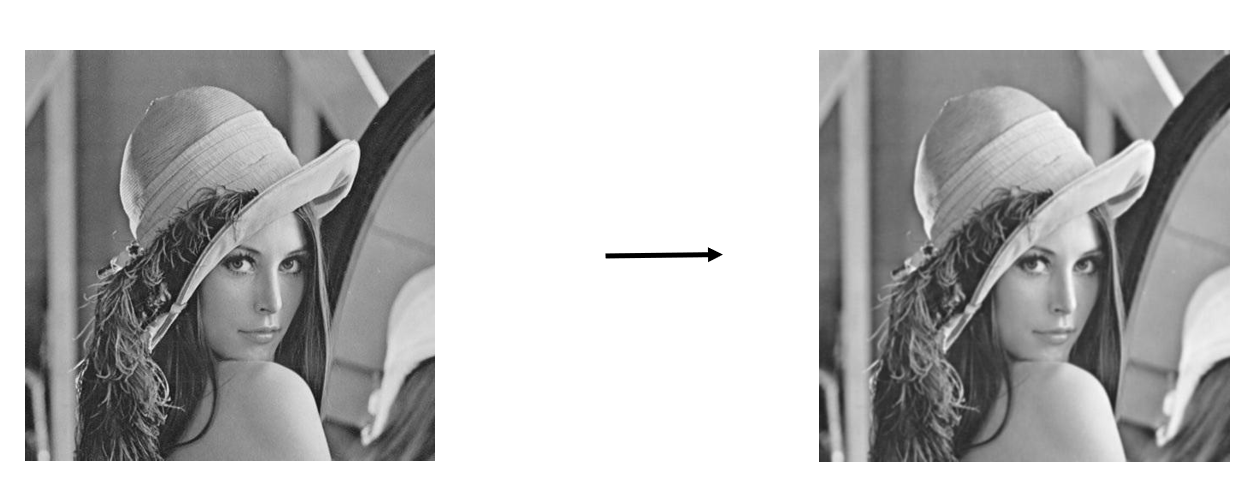


Where:

* G( x, y ) is the Gaussian kernel.
* σ is the standard deviation (controls smoothing).

**Key Point:**

* Removes high-frequency noise.
* Larger σ → more smoothing, but can blur edges.

******

***STEP 2 – GRADIENT MAGNITUDE AND DIRECTION:***

**Purpose:**  
To find regions with high intensity changes in both X and Y direction (edges).

**What Are Gradients?**

* A gradient measures the rate of change in pixel intensity.
* In edge detection, a large gradient means a sudden change — indicating a potential edge.

**Why Gradients?**

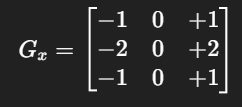
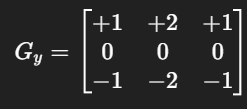
Edges occur where there is a sharp change in brightness, which corresponds to high gradients in the image.

**Visual Interpretation:**

* Gx detects vertical edges (intensity changes along x-axis).
* Gy detects horizontal edges (intensity changes along y-axis).
* The combination of Gx and Gy gives us the complete edge structure.

**How It’s Done:**

**1.Apply derivative filters** (**Sobel** operators): to detect intensity change in both horizontal (x) and vertical (y) directions.

**2.Compute horizontal and vertical gradients**:



Where

* ‘I’ is the image
* \* Is the convolution

**3.Calculate Gradient Magnitude**:



​​ This gives the **strength** of the edge at each pixel.

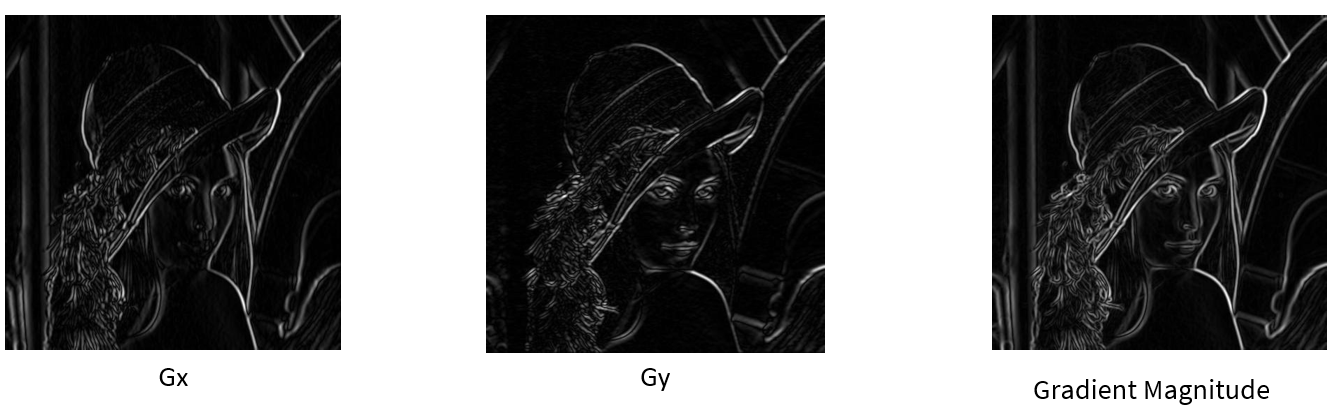
**4.Calculate Gradient Direction**:



This gives the **orientation** of the edge (used in Step 3 for Non-Maximum Suppression).

**Key Point:**

* Strong edges → high gradient magnitude.
* Direction used in the next step.



***STEP 3 – NON-MAXIMAL SUPPRESSION:***

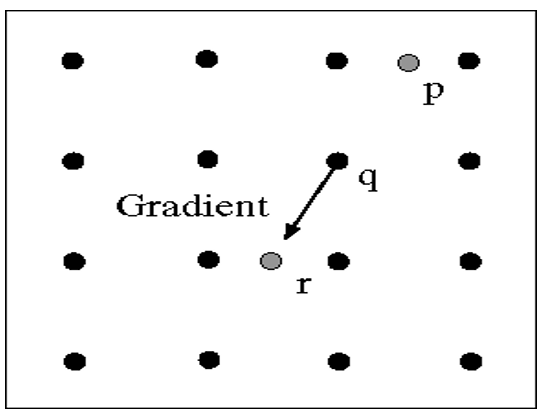
**Purpose:**

To thin out the wide edges detected in the gradient step, keeping only the pixels that are local maxima in the gradient direction.

**How:**

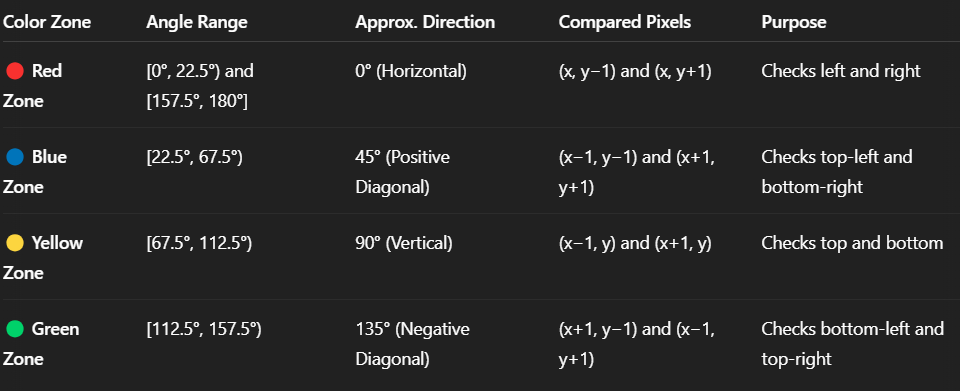
A pixel is retained as an edge only if it has the highest intensity compared to its two neighboring pixels along the gradient direction.

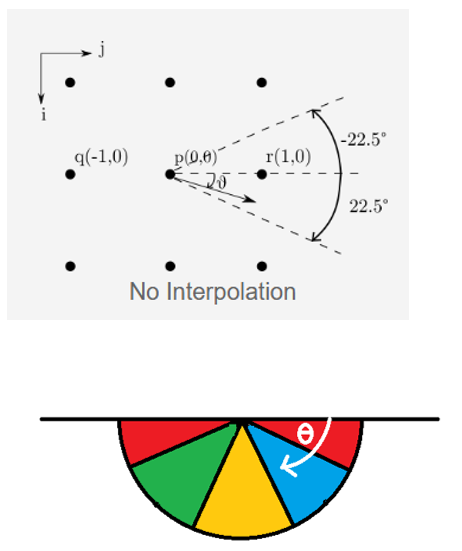
* 1. If the center pixel 𝑞 is greater than both neighbors 𝑝 and 𝑟, it is kept.
  2. Otherwise, it is suppressed (set to zero or black).



**Gradient Directions:**

Since the gradient direction is a continuous angle (0°–180°), it's approximated to **four main directions**, based on angle intervals:

******

******

**Algorithm Steps:**

1. For each pixel ‘q’ with gradient magnitude and direction:
2. Identify the direction group (0°, 45°, 90°, or 135°) from the gradient angle.
3. Compare ‘*q’* with neighboring pixels ‘*p*’ and *‘r*’ in that direction.
4. If *q>p* and *q>r* → **keep pixel** (possible edge).
5. Else → **suppress pixel** (set to 0 or black).



***STEP 4 – DOUBLE THRESHOLDING:***

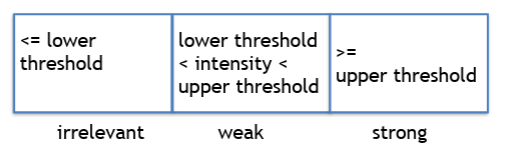
**Purpose:**

To classify edge pixels into strong, weak, or non-relevant edges, based on gradient magnitude, so that we can later determine which ones are true edges.

**How:**

Use two thresholds:

* High threshold (T \_ high): Pixels > T \_ high → Strong edges.
* Low threshold (T \_ low): Pixels between T \_ low and T \_ high → Weak edges.
* Below T \_ low → Non-edges.



**Key Point:**

* Helps distinguish actual edges from noise.
* Weak edges may be real if connected to strong edges.

**How It Works:**

Double Thresholding applies ***two thresholds***to the gradient magnitude image:



**Points to remember:**

* **High threshold** ensures only clear, prominent edges are marked as **strong**.
* **Low threshold** helps include weak edges that might be part of a real edge.
* This step alone **does not confirm** whether weak edges are valid — that's handled in the next step (Edge Tracking by Hysteresis).

***STEP 5 – FINAL EDGES BY HYSTERIS:***

**Purpose:**

To determine which **weak edges** from the previous step are part of **real edges**, by checking if they are **connected to strong edges**.

**Key Principle:**

* **Strong edges** are always retained (they are definitely edges).
* **Weak edges** are kept **only if** they are **connected** to strong edges through  **neighborhoods**.
* **Unconnected weak edges** are discarded (converted to black).

**Need:**

* Hysteresis here means that “weak edges follow strong edges” — they are only preserved if they are linked to strong ones.
* This avoids breaking apart edges while removing stray noise pixels.

**Step-by-Step Breakdown:**

1. Traverse the entire image pixel-by-pixel
2. For each pixel that is labeled as a weak edge:
   1. Check its neighbors (surrounding pixels).
   2. If any neighbor is a strong edge, upgrade the current weak pixel to a strong edge.
3. If no neighbors are strong:
   1. Set the current weak pixel to zero (discard it — not an edge).
4. Continue this process for all pixels.
5. The result is a clean binary image where:
   1. Strong edges are white (255),
   2. All others (weak & non-edges) are black (0).



***ADVANTAGES:***

* **High Accuracy:** Detects a wide range of edges with minimal error.
* **Noise Reduction:** Uses Gaussian filter to smooth out noise before detecting edges.
* **Edge Localization:** Finds edge positions with high precision.
* **Thin Edges:** Non-Maximum Suppression ensures edges are one-pixel wide.
* **Selective Edge Detection:** Double thresholding helps distinguish between strong and weak edges.
* **Connectivity:** Hysteresis ensures only meaningful edges (connected to strong edges) are retained.
* **Low False Detections:** Reduces chances of detecting irrelevant or noise-induced edges.

***DISADVANTAGES:***

* **Computationally Intensive:** Involves multiple complex steps, making it slower than simpler methods.
* **Parameter Sensitive:** Requires careful tuning of low and high threshold values for optimal results.
* **May Miss Fine Edges:** Over-smoothing or poor thresholding may lead to loss of weak but important edges.
* **No Color Edge Detection:** Operates on grayscale images; cannot detect color-specific edges.
* **Fixed Kernel:** Uses predefined Sobel kernels; not adaptive to content-specific edge patterns.

***APPLICATIONS:***

* Object recognition
* Lane detection in self-driving cars
* Medical image analysis
* OCR (Optical Character Recognition)
* Image segmentation and preprocessing

***PYTHON IMPLEMENTATION:***

import cv2

import matplotlib.pyplot as plt

# Load the image in grayscale

image = cv2.imread('your\_image.jpg', cv2.IMREAD\_GRAYSCALE)

# Apply Gaussian Blur to reduce noise (Step 1)

blurred = cv2.GaussianBlur(image, (5, 5), 1.4)

# Apply Canny edge detection (Steps 2–5 are internal)

edges = cv2.Canny(blurred, threshold1=50, threshold2=150)

# Display original and edge-detected images

plt.figure(figsize=(10,5))

plt.subplot(1, 2, 1)

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

plt.title('Original Grayscale Image')

plt.axis('off')

plt.subplot(1, 2, 2)

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

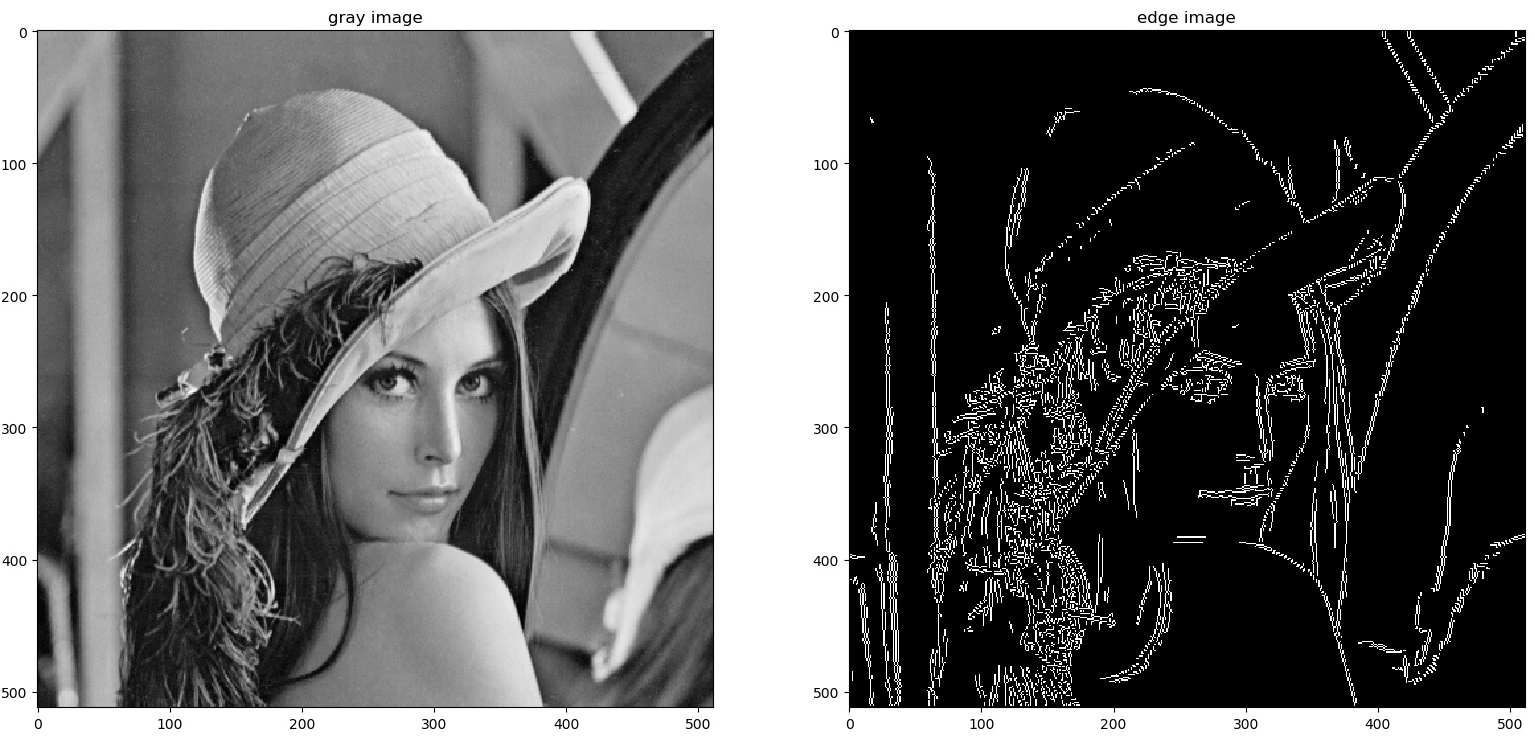
plt.title('Canny Edge Detection')

plt.axis('off')

plt.tight\_layout()

plt.show()

***SAMPLE OUTPUT:***



***CONCLUSION:***

Canny Edge Detection remains a cornerstone in edge detection methods due to its optimal performance in terms of detection, localization, and minimal response. Its well-defined steps and robustness make it suitable for a wide range of computer vision tasks.