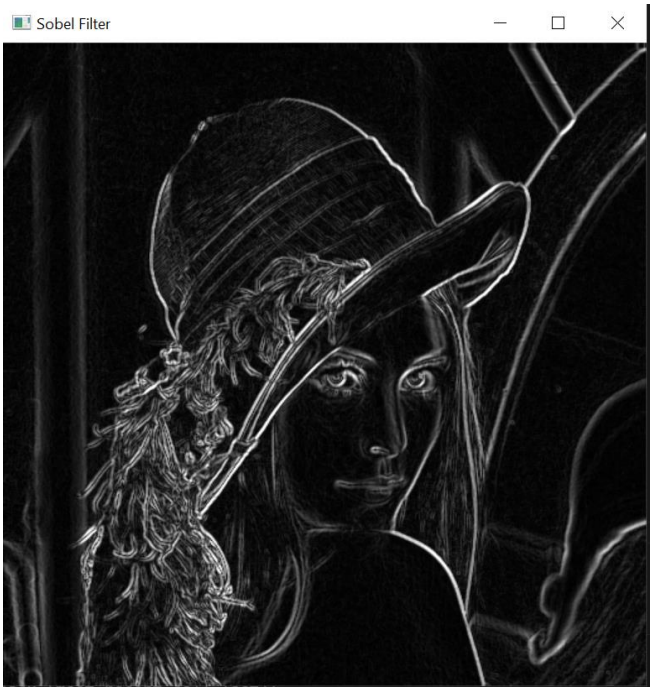
Sobel Filter

The Sobel filter is commonly used to find edges by approximating the gradient of the image intensity. The gradient is calculated using convolution with Sobel kernels, and the results are combined to produce the final edge map. This filter highlights the edges in both horizontal and vertical directions.

Here's a different approach using OpenCV for applying the Sobel filter:

```python
import cv2
import numpy as np

# Load the image in grayscale
image = cv2.imread('Lenna.png', cv2.IMREAD_GRAYSCALE)

# Apply Sobel filter in x and y directions
sobel_x = cv2.Sobel(image, cv2.CV_16S, 1, 0, ksize=3)
sobel_y = cv2.Sobel(image, cv2.CV_16S, 0, 1, ksize=3)

# Convert back to uint8
sobel_x = cv2.convertScaleAbs(sobel_x)
sobel_y = cv2.convertScaleAbs(sobel_y)

# Combine the gradients
sobel_combined = cv2.addWeighted(sobel_x, 0.5, sobel_y, 0.5, 0)

# Display the result
cv2.imshow('Sobel Filter', sobel_combined)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

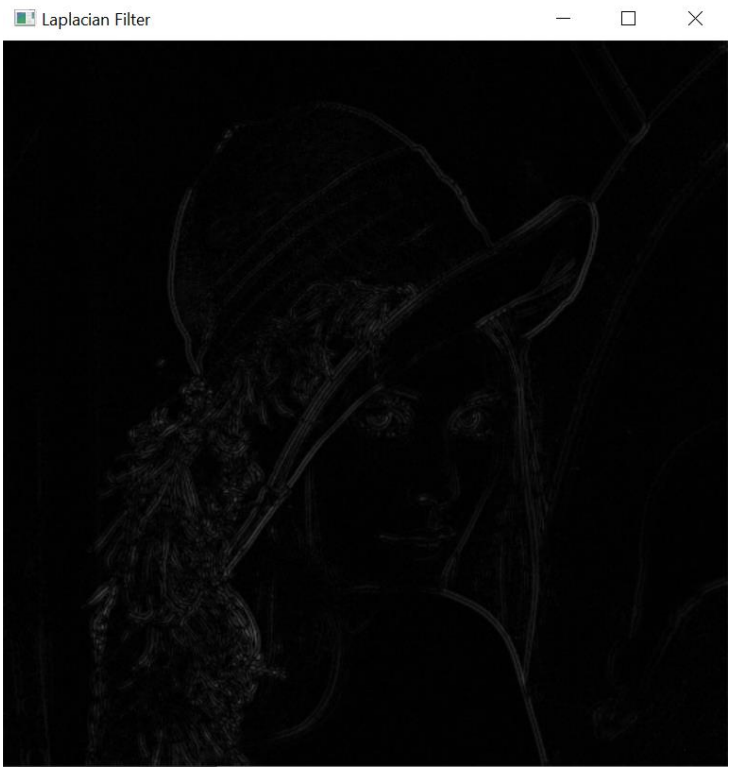


Laplacian Filter

The Laplacian filter is used to find edges by computing the second derivative of the image. It emphasizes regions with rapid intensity changes and is useful for edge detection and feature enhancement.

Here's an alternative implementation using OpenCV:

```python
import cv2

# Load the image in grayscale
image = cv2.imread('Lenna.png', cv2.IMREAD_GRAYSCALE)

# Apply Gaussian blur to reduce noise
blurred_image = cv2.GaussianBlur(image, (3, 3), 0)

# Apply Laplacian filter
laplacian = cv2.Laplacian(blurred_image, cv2.CV_64F)
laplacian = cv2.convertScaleAbs(laplacian)

# Display the result
cv2.imshow('Laplacian Filter', laplacian)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

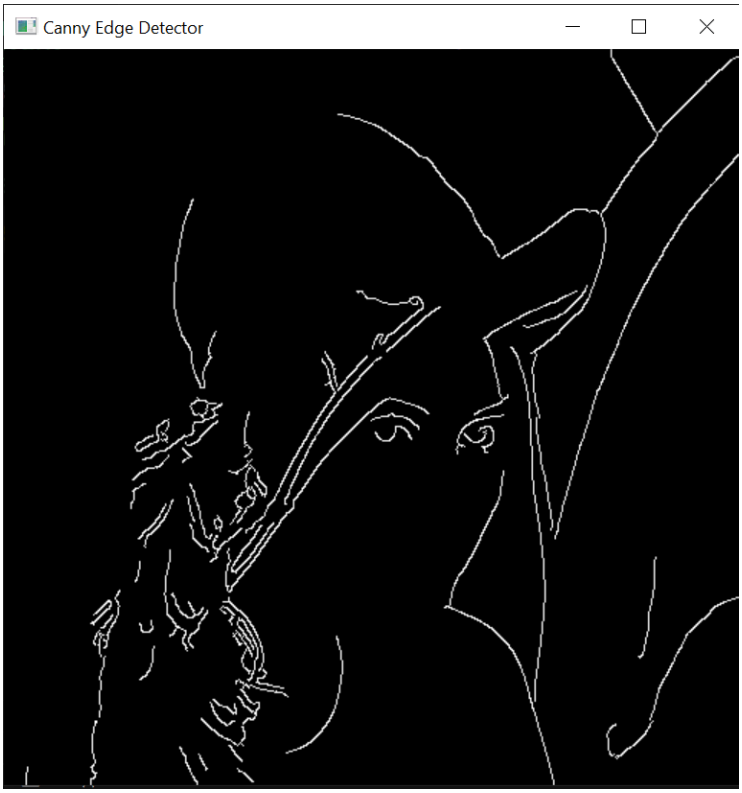


Canny Edge Detector

The Canny Edge Detector is a multi-step algorithm that detects edges by finding the intensity gradients and applying non-maximum suppression and edge tracking with hysteresis.

Here's another way to apply the Canny edge detector using OpenCV:

```python
import cv2

# Load the image in grayscale
image = cv2.imread('Lenna.png', cv2.IMREAD_GRAYSCALE)

# Apply Gaussian blur to smooth the image
blurred_image = cv2.GaussianBlur(image, (5, 5), 1.5)

# Apply Canny edge detector
edges = cv2.Canny(blurred_image, 100, 200)

# Display the result
cv2.imshow('Canny Edge Detector', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Contours in Image Processing

Contours are used to detect and analyze shapes and boundaries in an image. They are particularly useful for object detection and feature extraction.

Here's an alternative method to detect and draw contours using OpenCV:

```python
import cv2

# Load the image in grayscale
image = cv2.imread('Lenna.png', cv2.IMREAD_GRAYSCALE)

# Apply thresholding to binarize the image
_, binary_image = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)

# Find contours
contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Draw contours on the original image
contour_image = cv2.drawContours(image.copy(), contours, -1, (0, 255, 0), 2)

# Display the result
cv2.imshow('Contours', contour_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```