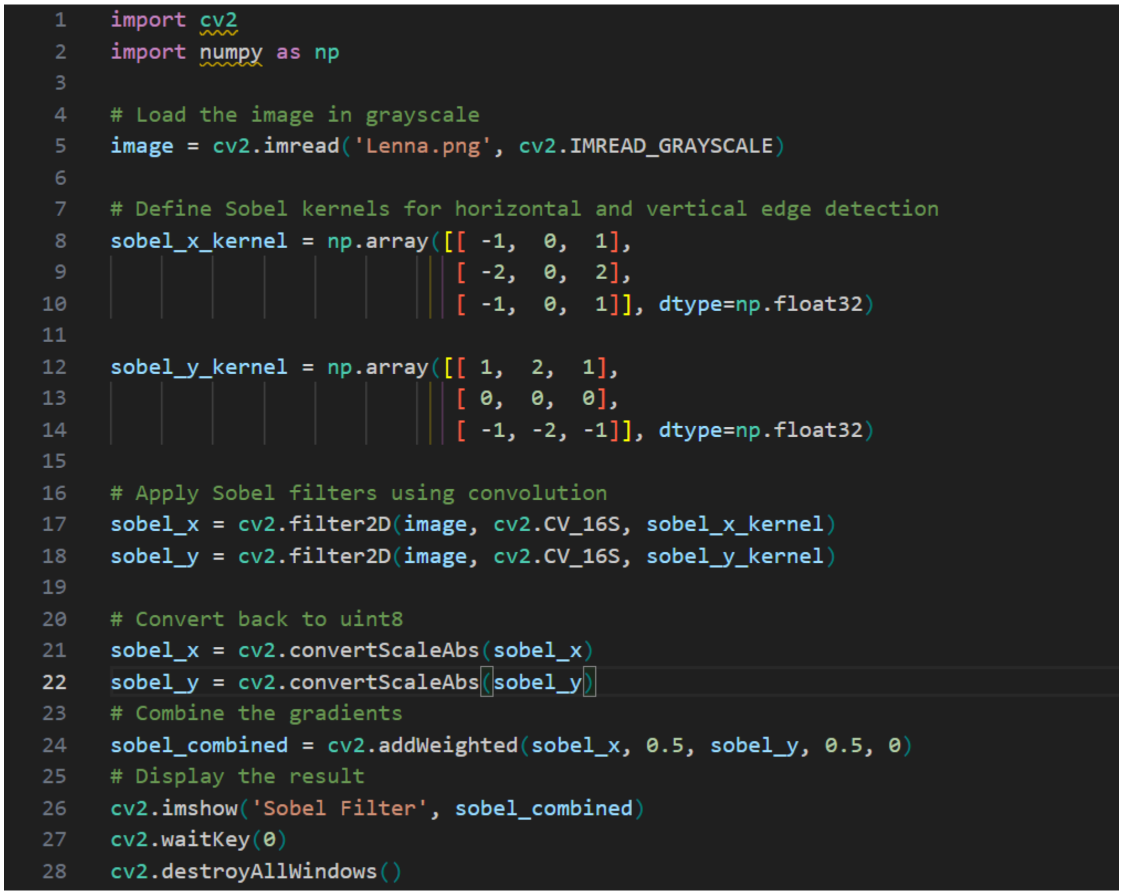
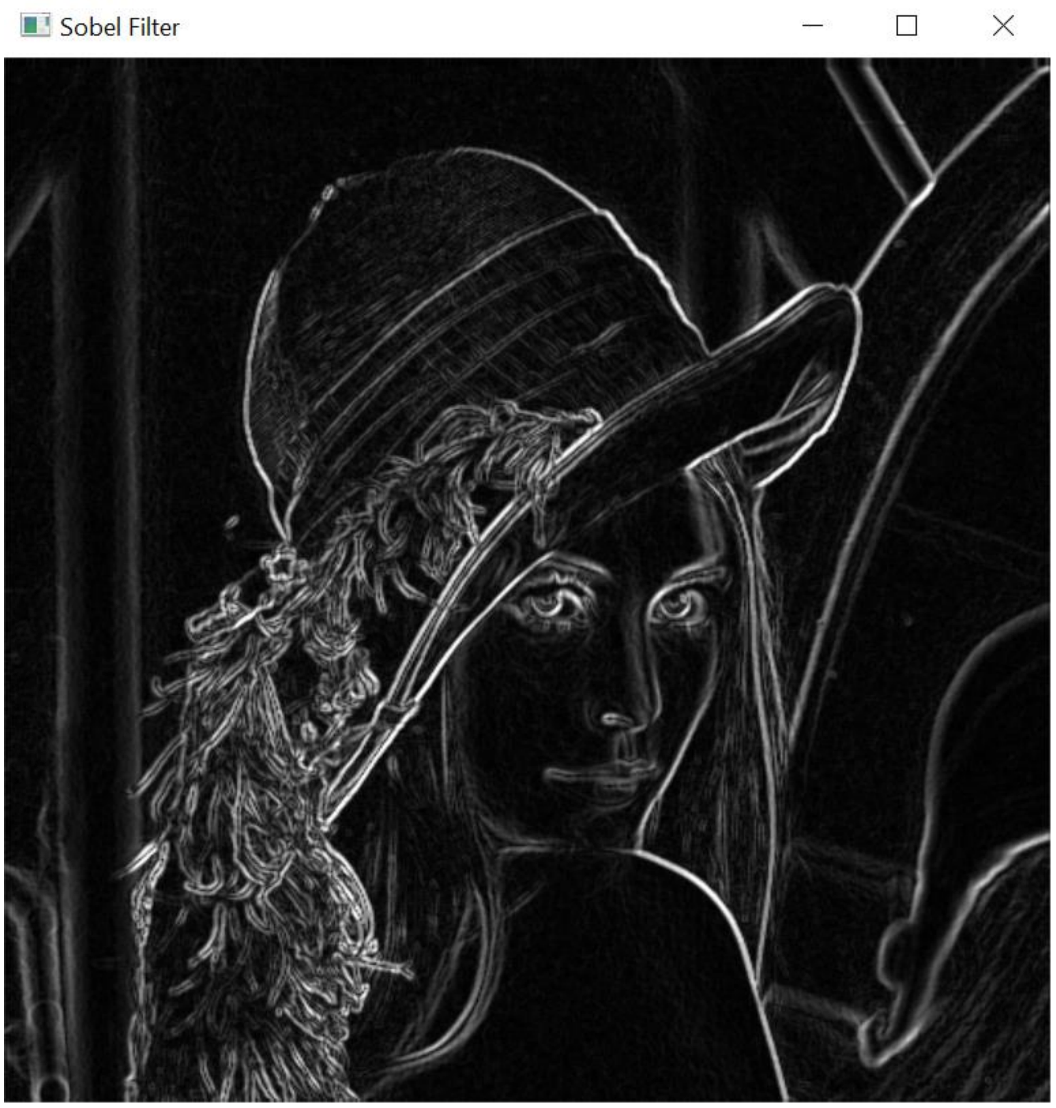
Filters and Edge Detection

Sobel Filter

The Sobel filter is an edge detection filter used to find the gradient of image intensity at each pixel, highlighting regions with high spatial frequency. It operates by convolving the image with Sobel kernels, which are specifically designed to detect edges in horizontal and vertical directions. This helps in identifying areas of the image where there is a significant change in intensity. The Sobel filter is commonly used in image processing for feature detection and image segmentation.



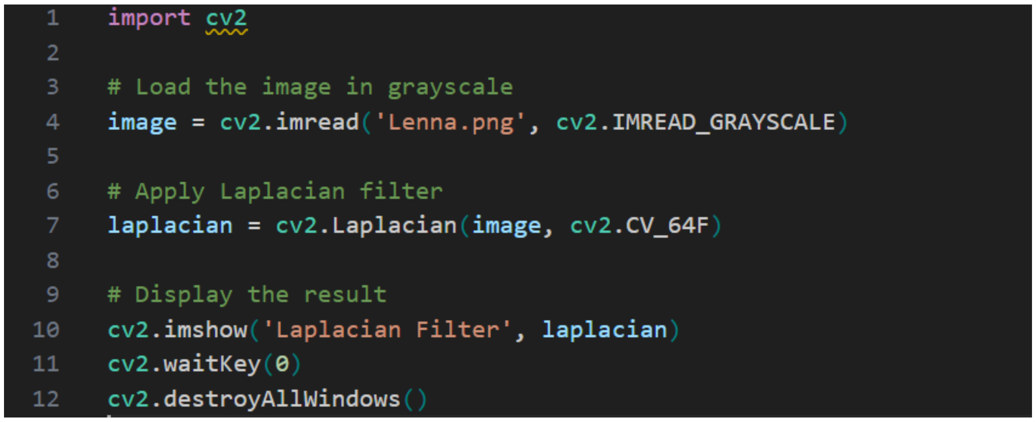
```python
import cv2
import numpy as np

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

# Define Sobel kernels for horizontal and vertical edge detection
sobel_x_kernel = np.array([[ -1,  0,  1],
                           [ -2,  0,  2],
                           [ -1,  0,  1]], dtype=np.float32)

sobel_y_kernel = np.array([[ 1,  2,  1],
                           [ 0,  0,  0],
                           [ -1, -2, -1]], dtype=np.float32)

# Apply Sobel filters using convolution
sobel_x = cv2.filter2D(image, cv2.CV_16S, sobel_x_kernel)
sobel_y = cv2.filter2D(image, cv2.CV_16S, sobel_y_kernel)

# 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()
```


Sobel Filter

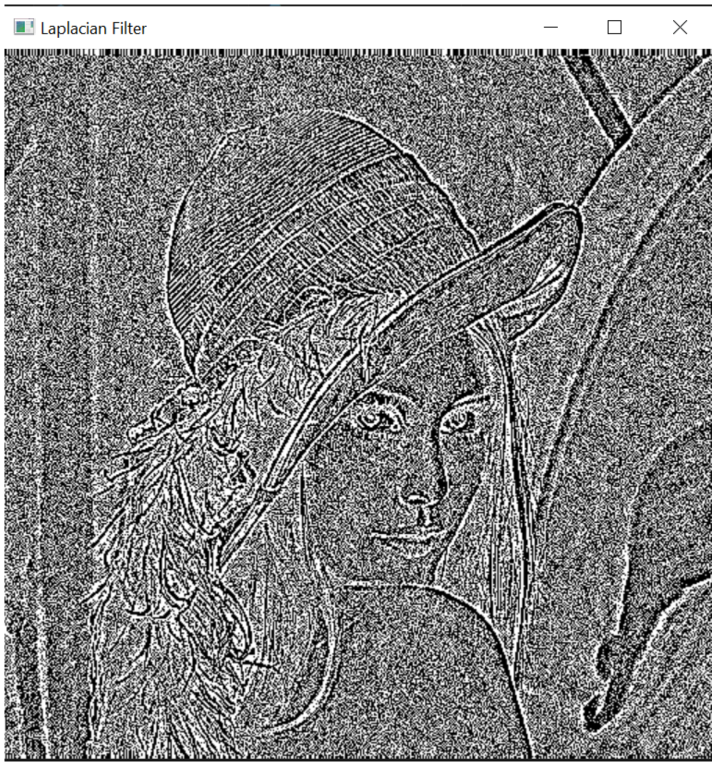
Laplacian Filter

The Laplacian filter is a second-order derivative filter used to detect edges by calculating the Laplacian of the image. It highlights regions of rapid intensity change and is particularly useful for detecting edges and textures. The Laplacian operator is applied to the image to find areas where the intensity of the pixel changes abruptly, which often corresponds to edges in the image.

Here's a Python code example to apply a Laplacian filter:



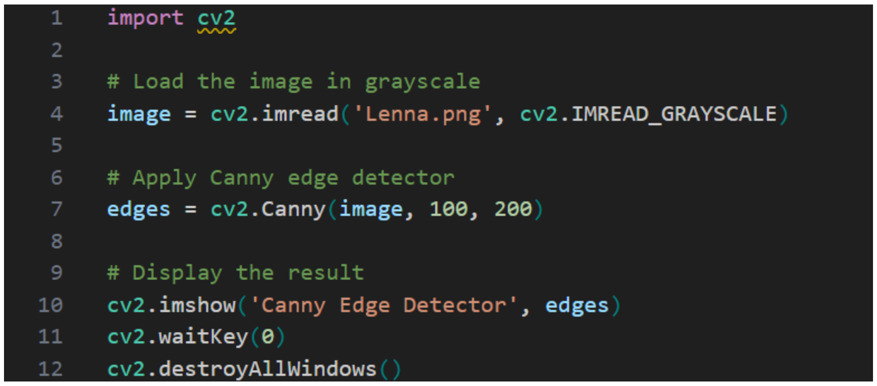
```python
import cv2

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

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

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



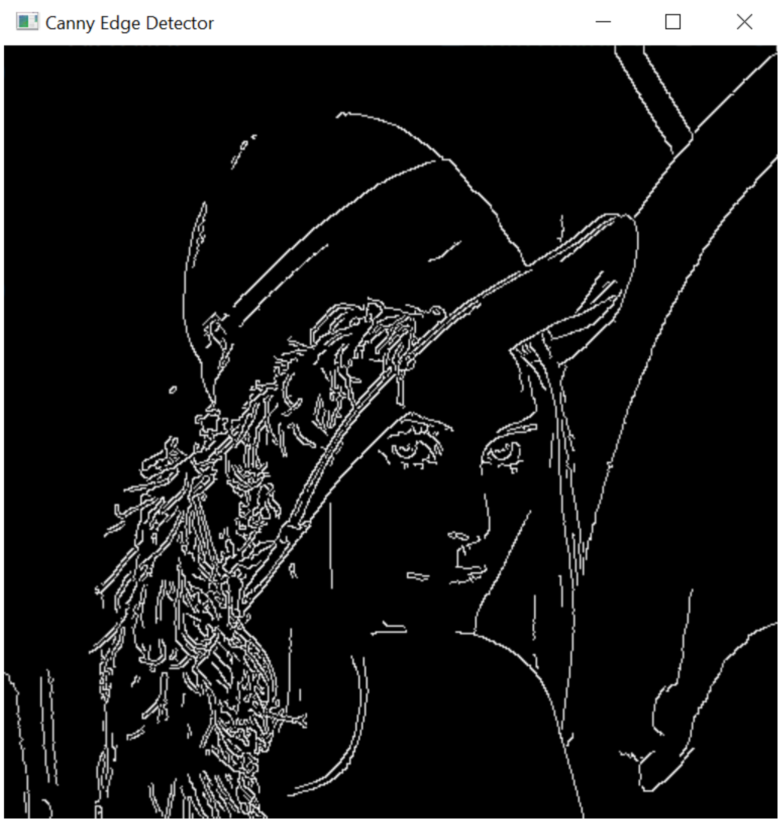
Canny Edge Detector

The Canny Edge Detector is a multi-stage algorithm designed to detect a wide range of edges in images. It uses a combination of Gaussian smoothing, gradient computation, non-maximum suppression, and edge tracking by hysteresis. The Canny algorithm is effective in detecting edges with precise localization and is widely used in various image analysis tasks.

Here is an example of how to apply the Canny Edge Detector in Python:



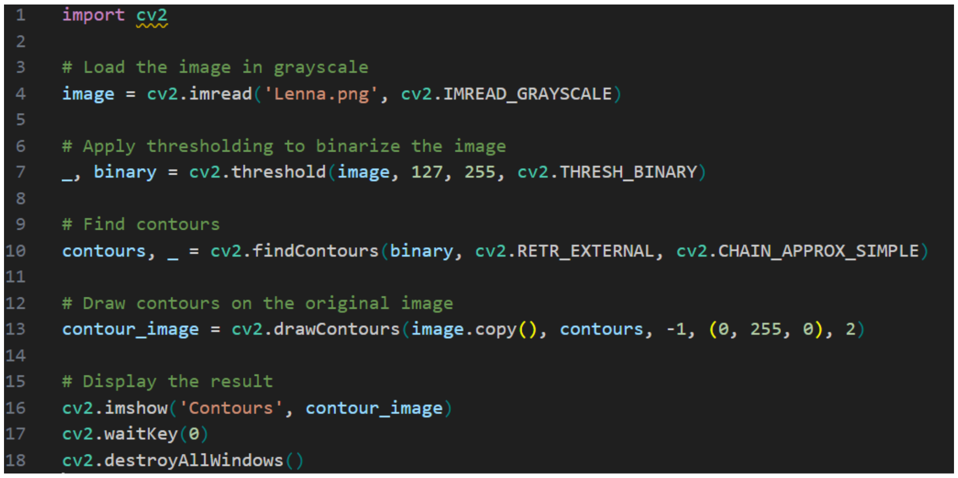
```python
import cv2

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

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

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



Contours in Image Processing

Contours are curves that join all the continuous points along a boundary that have the same color or intensity. In image processing, contour detection is used for shape analysis, object detection, and recognition. The algorithm identifies the boundaries of objects in an image, which can then be used for further analysis or processing.

Here's a Python example of detecting 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 = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)

# Find contours
contours, _ = cv2.findContours(binary, 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()
```