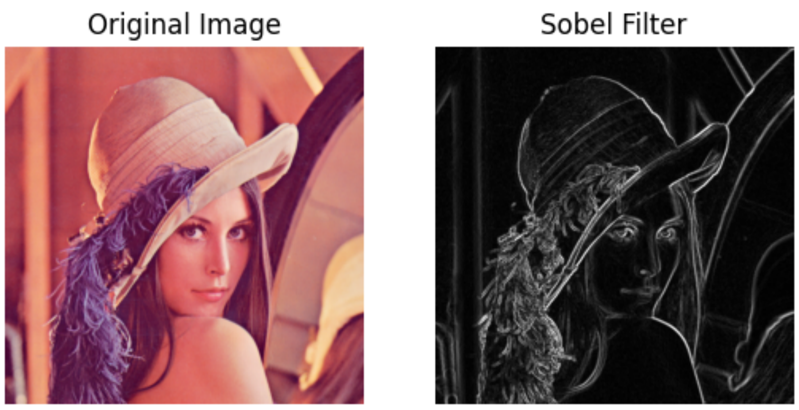
Image Processing Filters & Edge Detection Techniques

Sobel Filter



Sobel filter is used in image processing as an edge detection technique. It works by placing two kernels (matrices) on the image and shifting them pixel by pixel, until it computes the weighted sum for all of the pixels in the image. One of the kernels shifts in the horizontal direction, while the other shifts in the vertical direction. Each of these kernels is used to calculate the gradient intensity of pixels in the horizontal and vertical directions, showing us how the pixels' brightness changes throughout the image. The resulting gradients shows areas with sharp changes indicating edges.

Sobel filter is very useful when working with noisy images, as it performs smoothing during the process of edge detection. It also has a lot of applications in object recognition, and image segmentation tasks.

Python Code

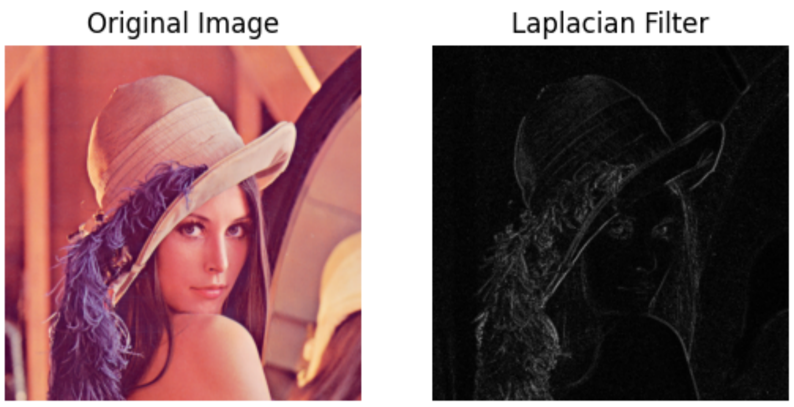
```python
# Apply Sobel filter in the x direction
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)  # x-gradient

# Apply Sobel filter in the y direction
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)  # y-gradient

# Convert gradients to absolute values
sobel_x = cv2.convertScaleAbs(sobel_x)
sobel_y = cv2.convertScaleAbs(sobel_y)

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

Laplacian Filter



This type of filter is also used as an edge detection technique. It works by summing the second derivatives of the image's intensity function in both the horizontal and vertical directions. This shows the rate of change of gradients across the image, helping us in highlighting areas where the intensity changes rapidly, indicating edges or boundaries in the image.

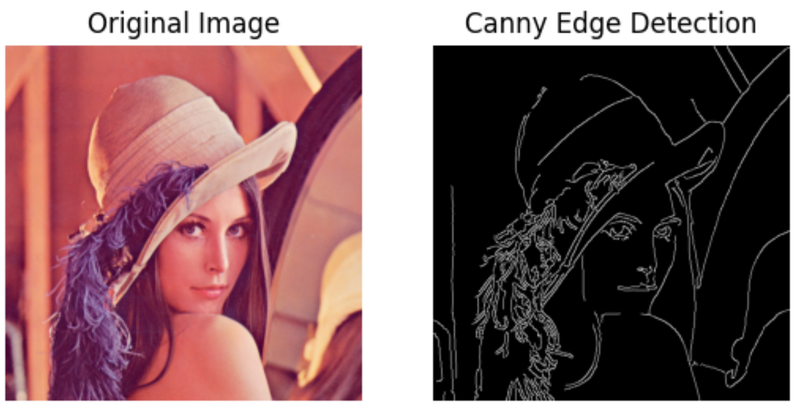
Laplacian filter is used in wide applications like enhancing edge details, and feature extraction. Although it may look like Sobel filter in calculating the gradient in both directions, Laplace filter is rotationally invariant, which means that it can detect edges regardless of orientation. Laplace filter is also sensitive to noise unlike the sobel filter.

Python Code

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

# Convert the result to a displayable format
laplacian = np.uint8(np.absolute(laplacian))
```

Canny Edge Detector



Canny Edge Detector is a very popular algorithm in image processing that is used for edge detection tasks. Its process involves several steps: first, in order to reduce noise, the image is smoothed using gaussian blur. Then, a gradient is computed using Sobel filter to determine the areas with sudden change in intensity, and direction of edges. Finally, the edges are classified as strong, weak, or non-edges using double thresholding, and edge tracking by hysteresis.

This algorithm is widely used for its effectiveness in detecting edges. It's applied in various computer vision fields like object recognition, and features extraction, where accuracy is essential for further analysis.

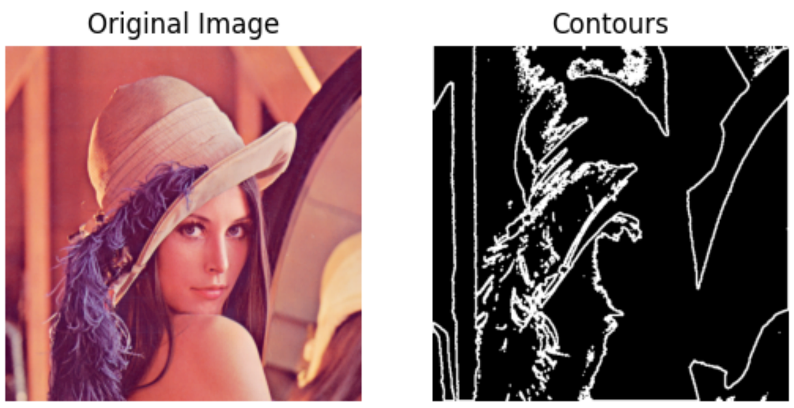
Python Code

```python
# Convert to grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply Gaussian Blur to reduce noise
blurred_image = cv2.GaussianBlur(gray_image, (5, 5), 1.4)

# Apply the Canny edge detector
edges = cv2.Canny(blurred_image, 50, 150)
```

Contours



Contours in image processing refers to the edges or boundaries of an object in an image. They can be identified by tracing continuous pixels with the same intensity or color. It is often performed on binary images where the object is segmented from the background.

Contours are very important in object detection, shape analysis, and recognition tasks. They can provide us with useful information about the object properties such as its area, parameter, and centroid.

Python Code

```python
# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply thresholding
_, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

# Find contours
contours, hierarchy = cv2.findContours(binary, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```