GNR607: Project

Edge Detection

Bhuvansh Goyal
(22B3908)

Instructor: Prof. B. Krishna Mohan

Introduction

Edge detection is a core technique in image processing, extensively used for image analysis. It plays a crucial role in identifying object boundaries, enabling the computation of parameters such as area, shape, size, and perimeter. This project focuses on implementing edge detection using the Laplacian of Gaussian (LoG) filter, where the user specifies the σ value. The detected edges are then refined by applying thresholding, with the threshold value also provided by the user, to produce the final edge map image.

Methodology

The following steps outline the process used in this project to generate the edge map:

1. **Input Image**: Load the user-specified image in grayscale mode and convert it to a numpy array for further processing.
2. **LoG Kernel**: Generate the Laplacian of Gaussian (LoG) kernel using the formula:

$$G(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$\begin{aligned} K(x,y) &= \left(\frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}\right) \\ &= \frac{1}{\sigma^2}\left(\frac{x^2+y^2}{\sigma^2} - 2\right) G(x,y) \end{aligned}$$

   The kernel's width is calculated as $2 \times \lceil 3 \times \sigma \rceil + 1$, ensuring a center pixel for assigning the convolution output. The kernel computation is optimized using the `np.meshgrid` function.

3. **Convolution**: Perform a 2D convolution of the image with the generated kernel using the `convolve2d` function from the `scipy` library.
4. **Thresholding**: Apply thresholding based on the following condition: a pixel is marked as an edge if:

$$\big(f''(x,y) > \text{threshold} \ \wedge \ f''(x+\Delta x, y+\Delta y) < -\text{threshold}\big)$$

   or

$$\big(f''(x,y) < -\text{threshold} \ \wedge \ f''(x+\Delta x, y+\Delta y) > \text{threshold}\big)$$

   where Δx and Δy correspond to the 8 possible directions.

Implementation

The edge detection is implemented in Python, with the main script in **main.py**. Supporting modules include:

- **kernel.py**: Generates the LoG kernel.
- **threshold.py**: Performs thresholding.

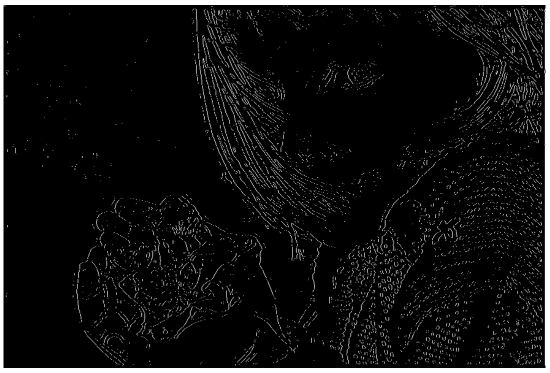
External libraries used:

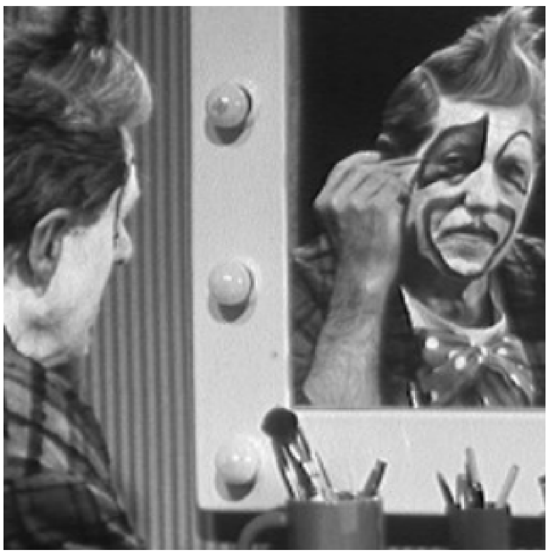
- **Pillow**: For loading images.
- **Matplotlib**: For displaying and saving results.
- **Scipy**: For 2D convolution.

Results

The following results were obtained using test images from the **images** folder and were saved in the **results** folder.

(a) Input Image 1


(b) Edge Detection Output 1


(c) Input Image 2

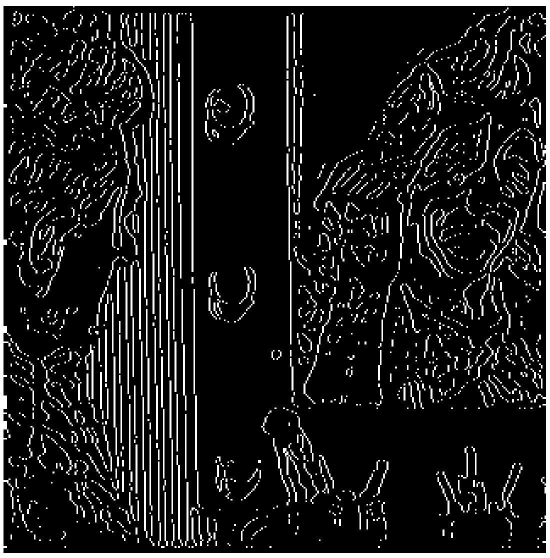
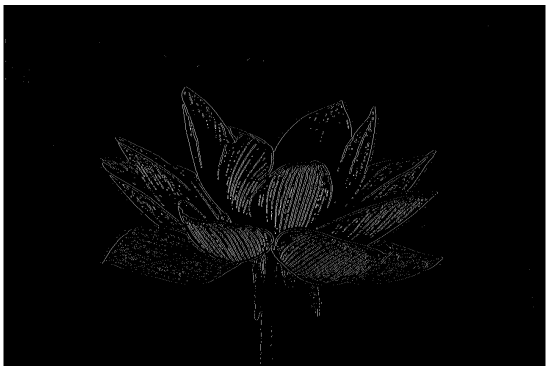
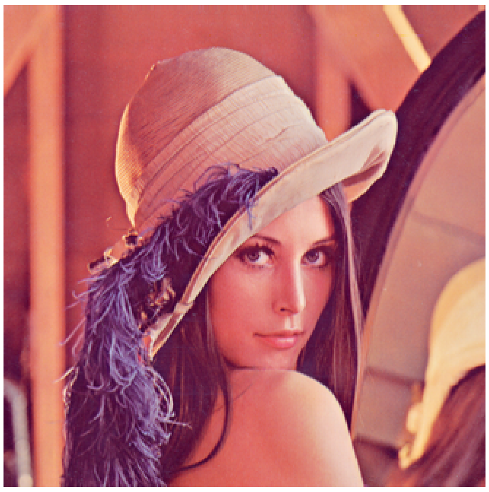

(d) Edge Detection Output 2

Figure 1: Edge Detection Results for Various Input Images

(a) Input Image 3


(b) Edge Detection Output 3


(c) Input Image 4

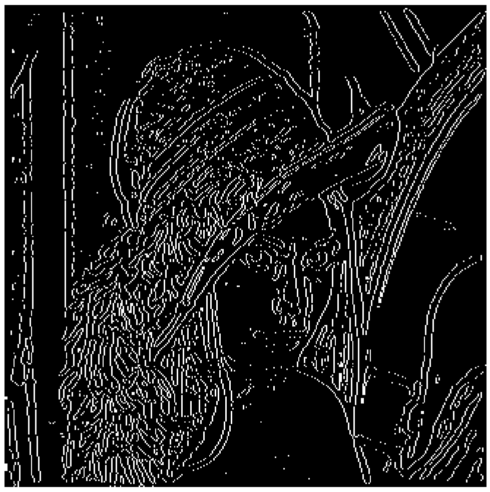

(d) Edge Detection Output 4

Figure 2: Edge Detection Results for Various Input Images

Drawbacks

Some of the drawbacks of using this approach are:

- The edges detected are often not continuous and may contain breaks, making it challenging to identify complete object boundaries.
- The performance of the method is sensitive to the choice of the σ value in the LoG filter, which can affect the detection of fine details or larger structures.
- Thresholding requires careful selection of the threshold value, as inappropriate values can result in missing edges or excessive noise.
- The method is computationally intensive due to the convolution with the LoG kernel, especially for large images or high σ values.

Conclusion

This project demonstrates the use of the Laplacian of Gaussian filter for edge detection, providing flexibility to the user by allowing custom sigma and threshold values.