Edge Detection Using Laplacian of Gaussian: A Mathematical and Coding Perspective

Aditi Bharadwaj, Dakota Chang

April 10, 2024

Github Repo: https://github.com/dcoder0111/QEA2_LaplacianOverGaussianOperator.git

1 Introduction

This document explores the concept of edge detection using the Laplacian of Gaussian (LoG) method, both from a mathematical standpoint and through its implementation in Python. Edge detection is a fundamental task in image processing, helping in object detection and image segmentation.

2 Gaussian Kernel

The Gaussian Kernel smooths an image by blurring it, reducing noise and detail. Mathematically, it is represented as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1}$$

where σ is the standard deviation, controlling the extent of the smoothing. In Python, we can generate a Gaussian Kernel using the following code:

```
def gaussian_kernel(size, sigma=1.0):
    size = int(size) // 2
    x, y = np.mgrid[-size:size+1, -size:size+1]
    normal = 1 / (2.0 * np.pi * sigma**2)
    g = np.exp(-((x**2 + y**2) / (2.0*sigma**2))) * normal
    return g
```

3 Laplacian of Gaussian (LoG)

The LoG operator enhances regions of rapid intensity change and is thus used for edge detection. It is the Laplacian ∇^2 of the Gaussian function G:

$$\nabla^2 G(x, y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

This formula detects edges by identifying zero crossings after filtering the image with this kernel. The corresponding Python implementation is as follows:

```
def laplacian_of_gaussian_kernel(size, sigma=1.0):
    size = int(size) // 2
    x, y = np.mgrid[-size:size+1, -size:size+1]
    normal = 1 / (np.pi * sigma**4)
    lo_g = ((x**2 + y**2 - (2.0 * sigma^2)) / (sigma^4)) * np.exp(-((x**2 + y^2) / (2.0*
    sigma^2))) * normal
    return lo_g
```

4 Zero Crossing

Zero crossing is a method to detect edges. An edge is presumed to be present where the sign of the Laplacian of the Gaussian changes, i.e., where it crosses zero. The zero-crossing method is applied to the image processed by the LoG kernel:

```
def zero_crossing(img):
    img[img > 0] = 1
    img[img < 0] = 0
    out_img = np.zeros(img.shape)
    for i in range(1, img.shape[0]-1):
        for j in range(1, img.shape[1]-1):
            if img[i, j] > 0 and any_neighbor_zero(img, i, j):
                out_img[i, j] = 255
    return out_img
```

This function scans the image for pixels that are positive but adjacent to at least one negative pixel, marking these as edges.

5 Conclusion

Through the mathematical formulations and Python code provided, we gain a comprehensive understanding of how the Laplacian of Gaussian method functions for edge detection in images. This method, blending theory with practical application, showcases the power of mathematical concepts in solving real-world problems in image processing.