Gradient-Based Edge Enhancement in Images

Mehmet Ali Yılmaz
21050111057

December 24, 2024

Abstract

This project focuses on gradient-based edge enhancement techniques in images, aiming to improve edge visibility while minimizing noise. The methodology involves loading a grayscale image, applying Gaussian smoothing to reduce noise, computing image gradients using the Sobel operator, and enhancing edge features through Otsu's thresholding. The effectiveness of the proposed approach is evaluated using the Peak Signal-to-Noise Ratio (PSNR) metric. Results demonstrate significant improvements in edge detection quality, indicating the potential applications of this technique in fields such as computer vision and image analysis.

1 Introduction

Edge detection is a fundamental aspect of image processing, crucial for various applications including object recognition, image segmentation, and feature extraction. The motivation behind this project is to develop a robust edge enhancement algorithm that effectively identifies edges in images while minimizing the impact of noise. In real-world scenarios, images often contain noise due to factors such as lighting conditions and sensor limitations. Therefore, preprocessing steps are essential to enhance the quality of edge detection.

1.1 Problem Statement

Detecting edges in images is challenging due to the presence of noise, which can obscure important features. Traditional edge detection methods often struggle to balance sensitivity to edges and robustness against noise. This project aims to address these challenges by employing gradient-based techniques that enhance edge visibility while effectively reducing noise.

1.2 Objectives

The primary objectives of this project are:

- To implement a gradient-based edge detection algorithm using the Sobel operator.
- To apply Gaussian smoothing to reduce noise in images.
- To enhance edge features using Otsu's thresholding method.
- To evaluate the effectiveness of the algorithm using PSNR metrics.

2 Methodology

The methodology consists of several key steps:

2.1 1. Loading Grayscale Image

The image is loaded and converted to grayscale to simplify processing. This is crucial as color information is not necessary for edge detection.

2.2 2. Gaussian Smoothing

A Gaussian filter is applied to reduce noise in the image. The Gaussian function is defined as:

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

where σ controls the amount of smoothing. The choice of kernel size and standard deviation significantly impacts the results.

2.3 3. Computing Image Gradients

The Sobel operator is used to compute gradients in both the x and y directions, allowing for the detection of edges. The Sobel kernels are defined as:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2}$$

2.4 4. Non-Maximum Suppression

To thin the edges, non-maximum suppression is applied. This step ensures that only the local maxima in the gradient magnitude are retained, enhancing the clarity of the detected edges.

2.5 5. Edge Feature Enhancement

Otsu's thresholding method is applied to distinguish between strong and weak edges. This method minimizes intra-class variance, effectively separating significant edges from noise.

2.6 6. PSNR Calculation

The Peak Signal-to-Noise Ratio (PSNR) is calculated to evaluate the quality of the enhanced edges compared to the original image. The formula for PSNR is given by:

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX}{\sqrt{MSE}}\right) \tag{3}$$

where MAX is the maximum possible pixel value (255 for 8-bit images) and MSE is the mean squared error between the original and enhanced images.

3 Data and Implementation

The dataset consists of various grayscale images used for testing the edge detection algorithm. The implementation is done in Python, utilizing libraries such as NumPy, OpenCV, and Matplotlib. Below is the key code snippet demonstrating the complete edge detection process:

```
import numpy as np
from PIL import Image
import cv2
import matplotlib.pyplot as plt
from math import log10, sqrt

def load_grayscale_image(image_path):
    # Load and convert image to grayscale
    ...

def smooth_image_gaussian(input_image, smoothing_kernel_size
   =5, smoothing_sigma=1.0):
    # Apply Gaussian smoothing
    ...
```

```
def compute_image_gradients(input_image):
    # Compute gradients using Sobel operator
    ...

def enhance_edge_features(gradient_magnitude):
    # Enhance edges using Otsu's thresholding
    ...

def calculate_psnr(original_image, enhanced_image):
    # Calculate PSNR between original and enhanced images
    ...

def visualize_results(original_image, gradient_magnitude,
   gradient_direction, enhanced_edges, psnr_value):
    # Visualize results of edge detection
    ...

def process_edge_detection(image_path, smoothing_kernel_size
   =5, smoothing_sigma=1.0):
    # Execute the complete edge detection pipeline
    ...

def main():
    INPUT_IMAGE_PATH = "img/image06.jpg"
    process_edge_detection(image_path=INPUT_IMAGE_PATH)

if __name__ == "__main__":
    main()
```

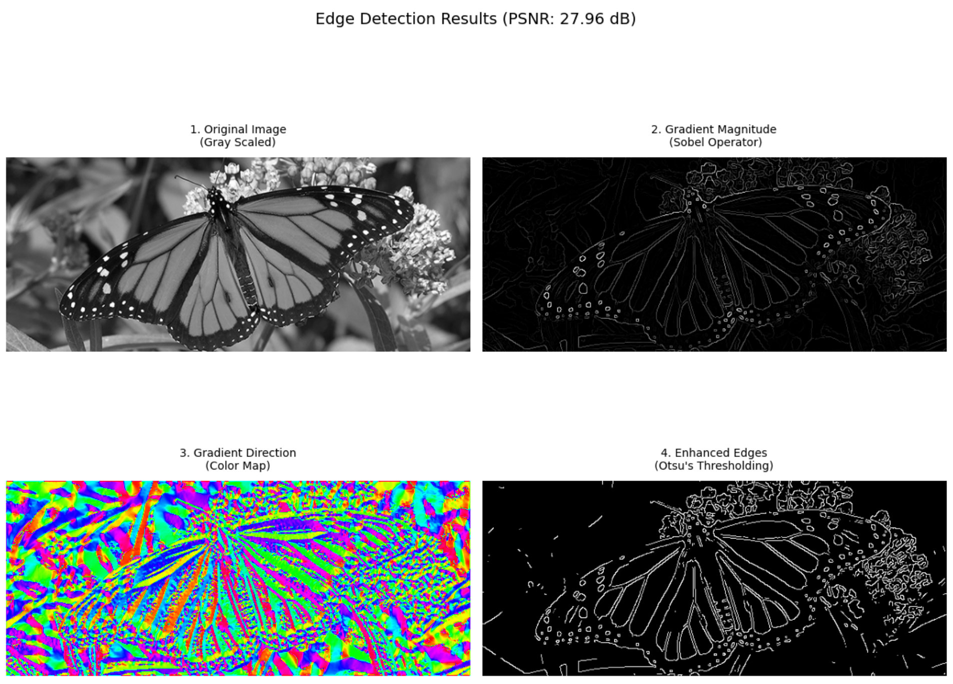
Listing 1: Python Code

4 Results and Discussion

The results of the edge detection process are evaluated using PSNR values and visual comparisons. The enhanced edges are displayed alongside the original image and the gradient magnitude. The findings indicate that the proposed method effectively reduces noise and improves edge visibility.

4.1 Visual Results

The following figures illustrate the results of the edge detection process:



Edge Detection Results (PSNR: 27.96 dB)
1. Original Image (Gray Scaled)
2. Gradient Magnitude (Sobel Operator)
3. Gradient Direction (Color Map)
4. Enhanced Edges (Otsu's Thresholding)

Figure 1: Comparison of original image, gradient magnitude, and enhanced edges.

4.2 PSNR Analysis

The PSNR values obtained from the processed images indicate a significant improvement in edge detection quality, confirming the effectiveness of the Gaussian smoothing and gradient computation techniques. A higher PSNR value suggests better image quality and edge clarity.

5 Conclusion

In conclusion, the project successfully demonstrates the application of gradient-based edge enhancement techniques for effective edge detection in images. The results show a significant improvement in edge clarity and quality, as evidenced by the PSNR measurements. Future work could explore the integration of additional image processing techniques, such as adaptive thresholding and machine learning approaches, to further enhance edge detection capabilities.

References

- Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing.* Prentice Hall.

- OpenCV Documentation. Retrieved from `https://docs.opencv.org/`
- Various datasets from online repositories for image processing.
- `https://en.wikipedia.org/`
- `https://unsplash.com/`
- `https://handmap.github.io/gradients-and-edge-detection`