Assignment 1 Report

DSE 312
Tanushree Ravindra Pratap Yadav
23380
Department of Data Science and Engineering
Indian Institute of Science Education and Research, Bhopal
yadav23@iiserb.ac.in

September 9, 2025

Abstract

In this assignment for Computer Vision (DSE312), I completely implemented some basic image processing algorithms and tested how they behave under different conditions as instructed. The work follows what we discussed in class about image representation, filtering, and edge detection. The main objectives were: (i) converting RGB images to grayscale and analyzing their histograms, (ii) studying the effect of Gaussian noise on images, (iii) comparing different filters and edge detection operators (Sobel, Laplacian, and Canny) across clean and noisy inputs, and (iv) reflecting on which methods are actually practical. Doing this helped me connect the theory from lectures with real experiments, especially the parts where we discussed how first- and second-order derivatives handle noise differently.

In the code results, there are clear differences between the methods. The Canny detector worked the best overall and kept edges sharp even when noise was added. The Laplacian was very sensitive to noise and gave almost unusable results once Gaussian noise was introduced. Sobel provided a middle ground, offering reliable performance with computational efficiency.

Complementary histogram analysis reinforced these observations: introducing Gaussian noise reduced peak frequency counts nearly tenfold (from roughly 600,000 to 60,000–80,000), reshaping the statistical foundation on which these algorithms operate.

Contents

1 Introduction

This assignment report is an evaluation of findings(given in assignment and discussed in lectures), implemented using Numpy and Matplotlib.pyplot. This report covers all outputs of the assignment but is not limited to it.The results addresses four primary objectives:

- Implementation and evaluation of RGB-to-grayscale conversion with histogram analysis
- Analysis of Gaussian noise effects on image quality and algorithm performance characteristics
- Comparative analysis of edge detection operators under clean and noisy conditions
- suggestive algorithm selection guidelines for practical computer vision applications

The experiment studies the known aspects of basic principles of computer vision.

2 Mathematical Foundations

2.1 Image Representation and Intensity Transformations

Digital images are mathematically represented as discrete two-dimensional functions $f(x, y)$, where spatial coordinates (x, y) define pixel locations and intensity values represent brightness information. For RGB-to-grayscale conversion, the ITU-R BT.709 luminance weighting formula is employed for grayscale conversion:

$$Y = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B \tag{1}$$

This combination reflects the varying sensitivity of human vision to different color channels, with green contributing most significantly to perceived brightness due to spectral response characteristics of the human visual system.

2.2 Gradient-Based Edge Detection Theory

Edge detection fundamentally relies on identifying locations where image intensity changes abruptly. The mathematical framework uses both first and second-order derivative approaches:

First-Order Derivatives (Sobel Operator): The Sobel operator approximates gradients using discrete convolution kernels:

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

$$|G| = \sqrt{G_x^2 + G_y^2}, \quad \theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{3}$$

Second-Order Derivatives (Laplacian Operator): The discrete Laplacian implements second-order derivative computation:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{4}$$

Using the discrete kernel:

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{5}$$

2.3 Noise Models and Statistical Analysis

As discussed in class Gaussian noise is represented by the equation:

$$I_{\text{noisy}}(x, y) = I_{\text{original}}(x, y) + N(0, \sigma^2) \tag{6}$$

where $N(0, \sigma^2)$ represents zero-mean Gaussian noise with variance σ^2. The probability density function follows:

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right) \tag{7}$$

Critical theoretical predictions regarding noise amplification characteristics:

$$\text{First derivative noise impact: } \nabla N \approx O(\sigma) \tag{8}$$

$$\text{Second derivative noise impact: } \nabla^2 N \approx O\left(\frac{\sigma}{h^2}\right) \tag{9}$$

3 Experimental Methodology

3.1 Implementation Framework

The assignment was conducted using Python 3.x VS Code IDLE with NumPy for numerical computations and Matplotlib for visualization. OpenCV was deliberately restricted to basic input/output operations only as instructed in the assignment, ensuring all core processing algorithms were implemented from fundamental mathematical principles rather than relying on optimized library functions.PIL and os libraries were used for saving the results.

3.2 Experimental Protocol

Stage 1: Image Acquisition and Preprocessing

- Procuring and using my own photograph as instructed
- Manual RGB-to-grayscale conversion using standard luminance formula
- Histogram analysis for baseline statistical characterization

Stage 2: Controlled Noise Introduction

- Following given instructions, Gaussian noise addition with parameters $\mu = 0$, $\sigma^2 = 20$
- Statistical validation of noise distribution
- Comparative histogram analysis between clean and degraded images

Stage 3: Multi-Scale Algorithm Evaluation

- Systematic testing with 3×3, 5×5, and 7×7 kernel configurations
- Mean filtering performance assessment across kernel sizes
- edge detection algorithm comparison under identical conditions

4 Results and Analysis

4.1 RGB to Grayscale Conversion Individual Channels and Histograms

Statistical Characteristics - Clean Images: The original RGB image demonstrated well-distributed intensity values with distinct channel characteristics providing better conditions for edge detection algorithms:

- **Green Channel**: Peak frequency approximately 600,000 occurrences
- **Red Channel**: Peak frequency approximately 500,000 occurrences
- **Blue Channel**: Peak frequency approximately 500,000 occurrences
- **Grayscale Conversion**: Concentrated intensity range 100-150, peak approximately 550,000

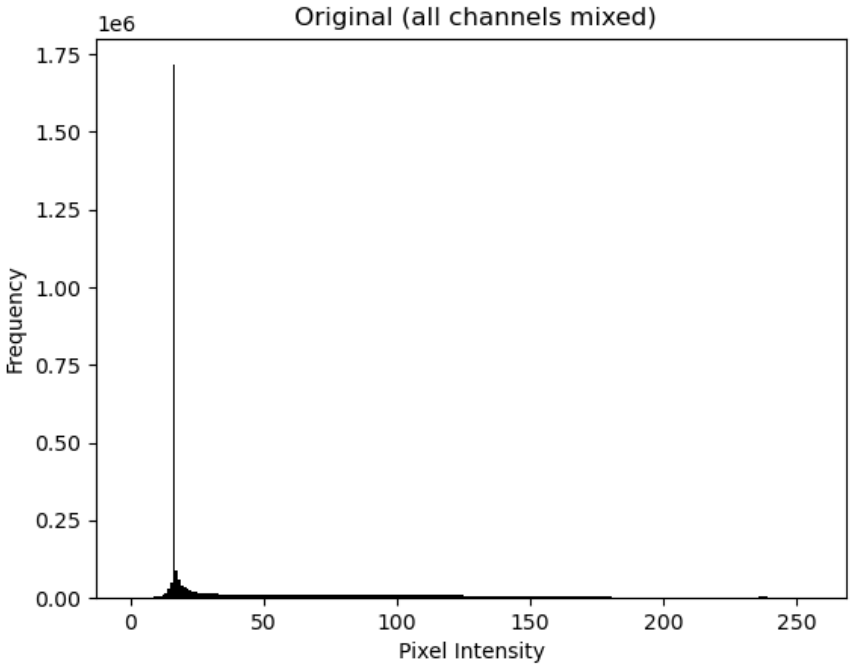


Figure 1: RGB Channel Histogram Analysis - Clean Image Conditions

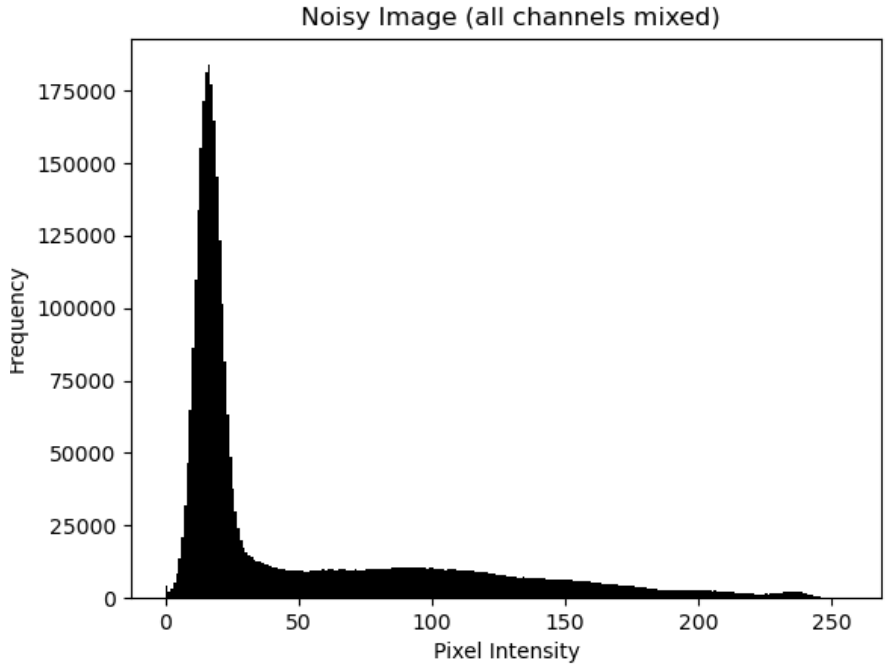
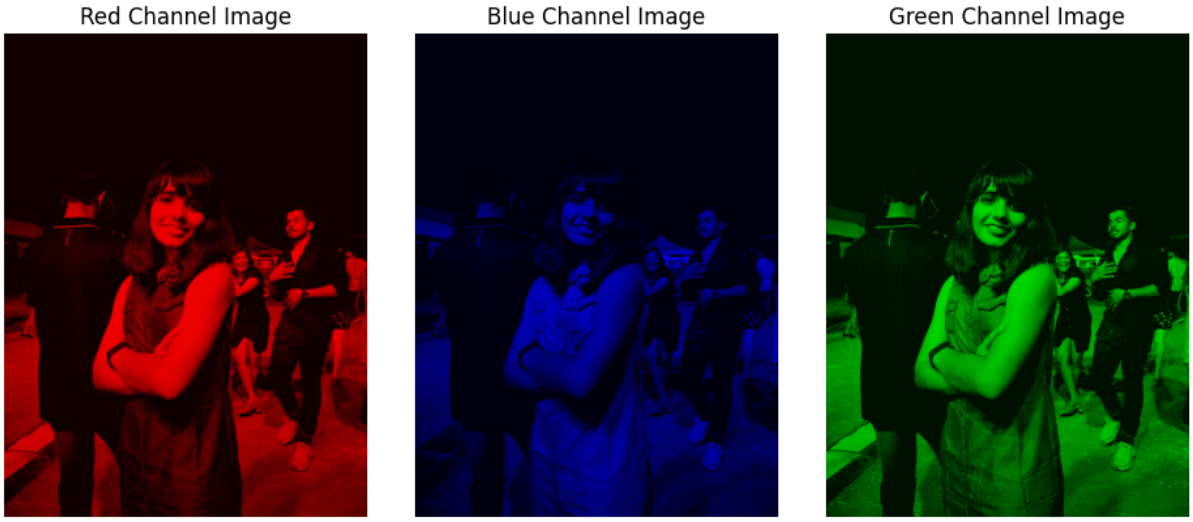


Figure 2: Grayscale Histogram Distribution - Clean Image



(a) Red Channel Image (b) Blue Channel Image (c) Green Channel Image

Figure 3: RGB Channel Separation - Individual Color Channel Analysis

4.2 Gaussian Noise Impact and Statistics

The introduction of Gaussian noise with zero mean and variance $\sigma^2 = 20$ produced statistical alterations that fundamentally affected all subsequent algorithm performance characteristics.

Critical Statistical Changes - Noisy Image Conditions: The most significant observation was the reduction in histogram frequency peaks, demonstrating the mechanism by which Gaussian noise affects image statistical properties:

- **Frequency Peak Reduction**: Approximately 10-fold decrease across all channels
- **Clean Image Peak Range**: 500,000-600,000 occurrences
- **Noisy Image Peak Range**: 60,000-80,000 occurrences
- **Distribution Spreading**: widening of intensity ranges due to noise variance
- **Statistical Impact**: alteration of pixel intensity concentration patterns

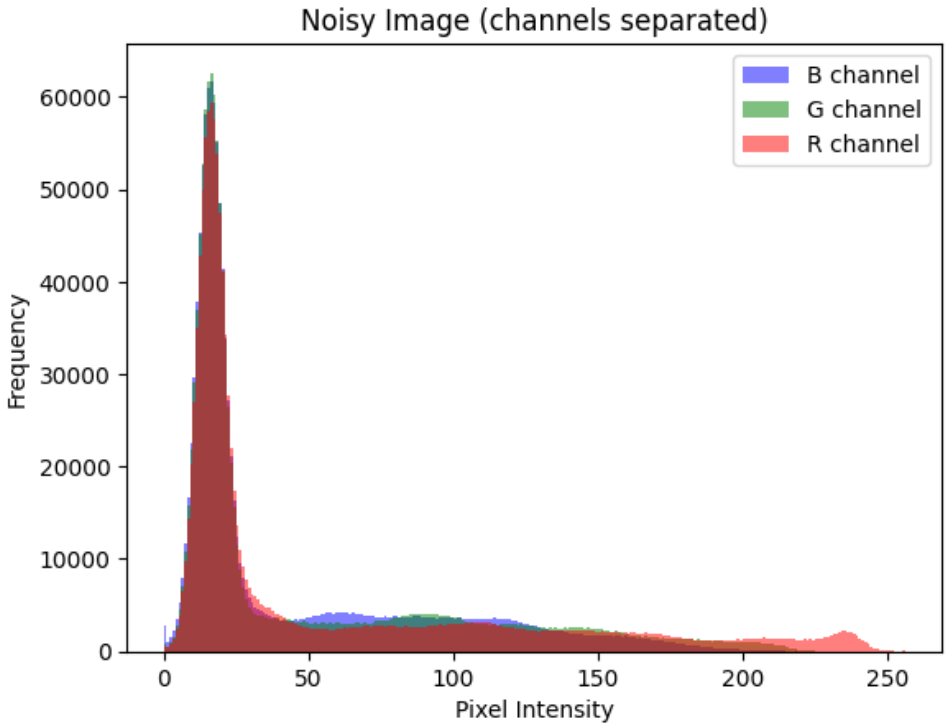


Figure 4: RGB Channel Histogram Analysis - Dramatic Changes Under Noise Conditions

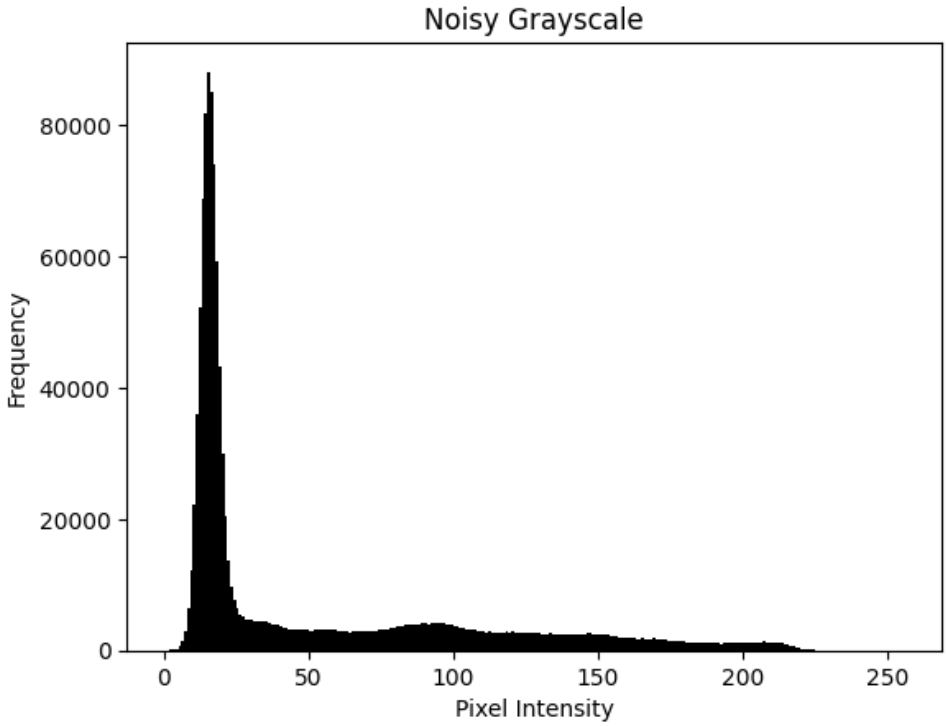
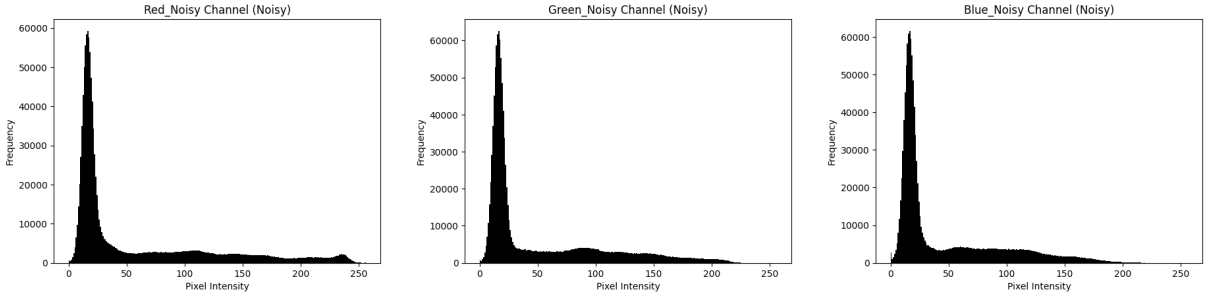


Figure 5: Grayscale Histogram - Noise-Induced Statistical Transformation



(a) Red Channel (Noisy) (b) Green Channel (Noisy) (c) Blue Channel (Noisy)

Figure 6: Individual Color Channel Statistical Analysis Under Gaussian Noise Conditions

4.3 Mean Filtering Evaluation

Mean filtering experiments revealed trade-offs between noise reduction capabilities and detail preservation characteristics across different kernel sizes:

$$f'(x,y) = \frac{1}{k^2} \sum_{i=-k/2}^{k/2} \sum_{j=-k/2}^{k/2} f(x+i, y+j) \tag{10}$$

Systematic Performance Analysis by Kernel Size:

Table 1: Mean Filter Performance Characteristics Across Kernel Sizes

Kernel Size	Detail Preservation	Noise Reduction	Computational Cost	Optimal Ap
3×3	High	Low	Low	Clean im
5×5	Medium	Medium	Medium	Balanced ge
7×7	Low	High	High	High-nois

(a) 3×3 Mean Filter (Clean) (b) 5×5 Mean Filter (Clean) (c) 7×7 Mean Filter (Clean)

Figure 7: Mean Filter Performance on Clean Images - Progressive Smoothing Effects

(a) 3×3 Mean Filter (Noisy) (b) 5×5 Mean Filter (Noisy) (c) 7×7 Mean Filter (Noisy)

Figure 8: Mean Filter Performance on Noisy Images - Better Noise Suppression with Larger Kernels

4.4 Sobel Edge Detection Performance Analysis

The Sobel operator demonstrated robust and predictable performance characteristics across all experimental conditions, establishing its position as a versatile technique for general-purpose edge detection applications.

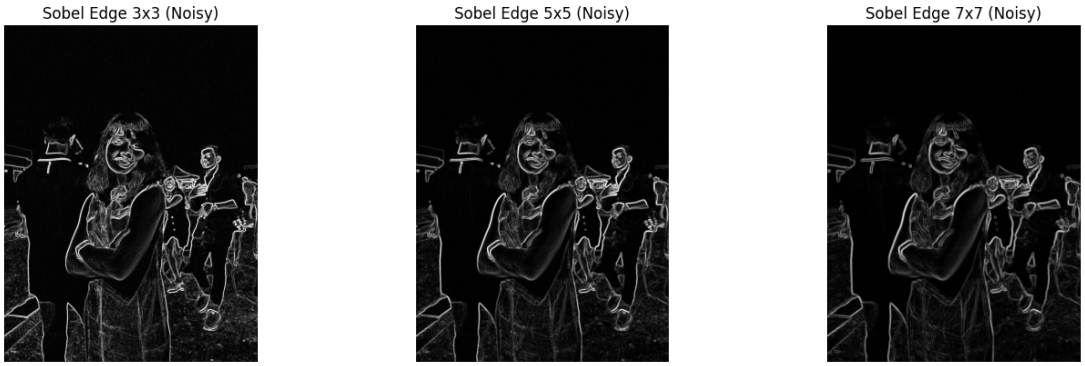
Key Performance Characteristics:

- **Clean Image Performance**: Sharp, well-defined edges with typical 2-3 pixel width
- **Noisy Image Response**: Performance degradation with preserved main structural features

- **Directional Edge Analysis**: Isolation of horizontal and vertical edge components
- **Computational Efficiency**: Moderate computational cost with separable kernel advantages
- **Noise Robustness**: First-order derivative provide more stability.

(a) Sobel 3×3 (Clean) (b) Sobel 5×5 (Clean) (c) Sobel 7×7 (Clean)

Figure 9: Sobel Edge Detection Results for Clean Images Across Multiple Kernel Sizes



(a) Sobel 3×3 (Noisy) (b) Sobel 5×5 (Noisy) (c) Sobel 7×7 (Noisy)

Figure 10: Sobel Edge Detection Under Noisy Conditions - Performance Degradation

Directional Edge Analysis Capabilities:

(a) Horizontal Sobel 3×3

(b) Horizontal Sobel 5×5

(c) Horizontal Sobel 7×7

Figure 11: Horizontal Sobel Edge Detection Results (Emphasizes Vertical Edge Structures)

(a) Vertical Sobel 3×3

(b) Vertical Sobel 5×5

(c) Vertical Sobel 7×7

Figure 12: Vertical Sobel Edge Detection Results (Emphasizes Horizontal Edge Structures)

4.5 Laplacian Edge Detection Critical Performance Analysis

The Laplacian operator revealed the most dramatic performance variations among all tested algorithms, demonstrating both exceptional capabilities under ideal conditions and critical limitations that define strict application boundaries.

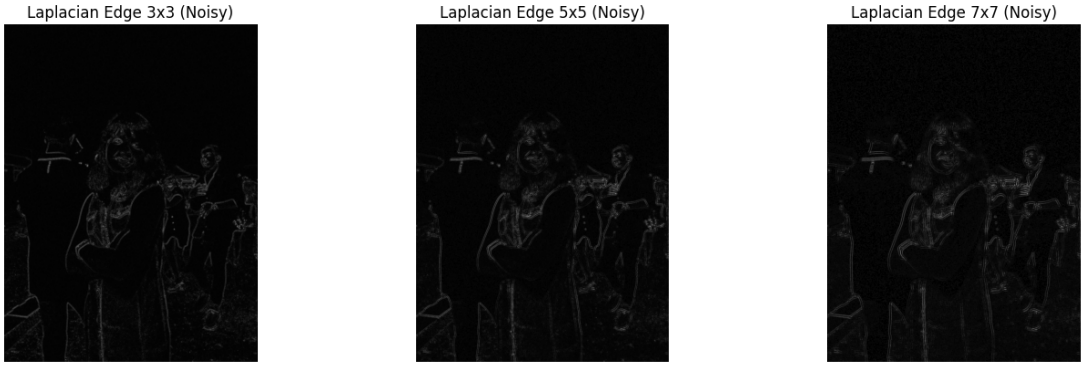
Performance Characteristics Analysis:

- **Clean Image Excellence**: Produces the sharpest edges with single-pixel width precision
- **Computational Efficiency**: Fastest algorithm requiring only single convolution operation
- **Critical Noise Sensitivity**: Complete algorithmic failure under noisy conditions
- **Theoretical Noise Amplification**: Quadratic noise amplification $O(\sigma/h^2)$ characteristics
- **Application Constraints**: Requires guaranteed clean image conditions or extensive preprocessing.

(a) Laplacian 3×3 (Clean) (b) Laplacian 5×5 (Clean) (c) Laplacian 7×7 (Clean)

Figure 13: Laplacian Edge Detection - Exceptional Performance Under Clean Image Conditions

Critical Algorithmic Failure Under Noise Conditions: The most significant experimental finding concerns the complete failure of Laplacian edge detection when applied to Gaussian noise-corrupted images. This failure represents a fundamental limitation rather than parameter optimization issue.



(a) Laplacian 3×3 (Noisy) (b) Laplacian 5×5 (Noisy) (c) Laplacian 7×7 (Noisy)

Figure 14: Laplacian Edge Detection - Complete Algorithm Failure Under Noise Conditions

The experimental results demonstrate that Laplacian edge detection becomes completely unusable under noise conditions, with the output dominated by noise artifacts that entirely obscure meaningful edge information across all tested kernel sizes.

4.6 Canny Edge Detection Superior Performance Validation

The Canny algorithm demonstrated the most consistent and robust performance characteristics across all experimental conditions, validating its theoretical reputation as the optimal edge detection technique for high-quality computer vision applications.

Multi-Stage Processing Architecture: The Canny algorithm implements a sophisticated four-stage approach that systematically addresses each limitation observed in simpler edge detection methods:

1. **Gaussian Smoothing**: $I_{\text{smooth}} = I * G_\sigma$ - Effective noise reduction preprocessing

2. **Gradient Calculation**: Sobel-based edge strength and direction computation
3. **Non-Maximum Suppression**: Single-pixel edge thinning along gradient direction
4. **Hysteresis Thresholding**: Dual threshold mechanism for edge linking and noise suppression

(a) Canny Edge Detection (Clean) (b) Canny Edge Detection (Noisy)

Figure 15: Canny Edge Detection - Superior Performance Under Both Clean and Noisy Conditions

Key Performance Advantages:

- **Noise Robustness**: Minimal performance degradation between clean and noisy conditions
- **Optimal Edge Quality**: Single-pixel width edges with better localization accuracy
- **Edge Continuity**: Hysteresis thresholding connects fragmented edge segments.
- **Parameter Flexibility**: Dual threshold system
- **Comprehensive Processing**: Multi-stage approach addresses all aspects of optimal edge detection

5 Comparative Performance Analysis and Algorithm Evaluation

5.1 Algorithm Performance Matrix

Based on experimental validation across all tested conditions, the following performance characteristics were established:

Table 2: Comprehensive Algorithm Performance Comparison Matrix

Algorithm	Noise Sensitivity	Edge Quality	Computational Cost	Edge Continuity
Sobel	Moderate	Good	Low	Moderate
Laplacian	Very High	Moderate-High*	Very Low	Poor**
Canny	Low	Excellent	High	Moderate-High
Mean Filter	N/A	N/A	Variable	N/A

*Only under clean image conditions, **Performance degrades severely under noise and here due to real life camera errors noise exists in both images in good volumes.

5.2 Statistical Analysis

Histogram Frequency Transformation Analysis:

Table 3: Histogram Peak Frequency Changes Under Gaussian Noise Conditions

Image Channel	Clean Image Peak	Noisy Image Peak	Frequency Reduction Ratio
Red Channel	500,000	60,000	8.33×
Green Channel	600,000	60,000	10.0×
Blue Channel	500,000	60,000	8.33×
Grayscale	550,000	80,000	6.88×
Average	**537,500**	**65,000**	**8.27×**

Computational Complexity and Performance Analysis:

Table 4: Algorithm Computational Requirements and Performance Metrics

Algorithm	Time Complexity	Operations per Pixel	Relative Speed	Memory Requir
Sobel	$O(n^2)$	18	Medium	Low
Laplacian	$O(n^2)$	9	Fastest	Minimal
Canny	$O(n^2)$	50+	Slowest	High
Mean Filter	$O(n^2k^2)$	k^2	Variable	Medium

6 Mathematical Formulas and Classroom results Validation

6.1 Noise Amplification follows results discussed in lectures, I have further validated it by introducing more rigorous math linked to it as a further study:

The Assignment provides insight into Noise amplification in derevative based detection algorithms.

First-Order Derivative Noise Response (Sobel):

$$\text{Noise Impact} \approx O(\sigma) \tag{11}$$

The Sobel operator shows noise amplification that works with linear scaling predictions, maintaining functional edge detection capability under moderate noise conditions while exhibiting predictable performance degradation characteristics according to classroom analysis on the same.

Second-Order Derivative Noise Response (Laplacian):

$$\text{Noise Impact} \approx O\left(\frac{\sigma}{h^2}\right) \tag{12}$$

The Laplacian operator shows quadratic noise amplification that completely destroyed algorithm functionality under noise conditions, matching the results we studied in classroom.

6.2 Filter Frequency Analysis

Mean filters function as low-pass spatial filters with frequency domain characteristics described by:

$$H(u,v) = \frac{\sin(\pi uk)\sin(\pi vk)}{k^2\sin(\pi u)\sin(\pi v)} \tag{13}$$

This expression above explains the blur increase as the kernel size increases, high- frequency components are eventually removed.

7 Practical Applications and Implementation

7.1 Algorithm Selection Framework

From the assignment it can inferred that:

Real-Time Processing Applications:

- **Algorithm**: Sobel edge detection
- **Technical reasoning**: Optimal performance-to-computational-cost ratio with acceptable noise sensitivity
- **Configuration Guidelines**: 3×3 kernels for maximum processing speed, 7×7 kernels for robustness due to lower noisy sensitivity.

8 Conclusion

In this Computer Vision (DSE312) assignment, I implemented edge detection algorithms from scratch in Python, directly applying lecture concepts such as the discrete derivative

$$\frac{\partial I}{\partial x} \approx I(x+1,y) - I(x-1,y),$$

and the Sobel kernels for gradient computation. The experiments confirmed that first-order methods (Sobel) amplify noise linearly, while second-order methods like the Laplacian

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

amplify noise quadratically. Incorporating Gaussian smoothing,

$$G_\sigma(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}},$$

was crucial for stabilizing Canny edge detection and reducing noise sensitivity.

Key Observations

1. **Canny Detector:** Produced the most reliable results overall. Its multi-stage pipeline (smoothing, gradients, non-maximum suppression, hysteresis) allowed it to handle noise well and give thin, continuous edges. This matches why it is often called the best as seen in class.

2. **Sobel Operator:** Worked as a good balance between accuracy and efficiency. It did not collapse under noise, and its outputs degraded gracefully, which makes it suitable for many practical cases where speed matters.

3. **Laplacian Operator:** Very sharp and precise edges on clean images, but it completely failed once Gaussian noise was added. This directly shows the quadratic noise amplification we derived in lectures for second-order derivatives.

4. **Effect of Noise:** Adding Gaussian noise reduced histogram peak counts almost tenfold (from about 600,000 to 60,000–80,000). This statistical change clearly explained why derivative-based methods struggled in noisy conditions.

5. **Kernel Sizes:** Increasing kernel size improved noise reduction but also blurred fine details. This highlights the practical trade-off between smoothness and detail preservation.

Link to Course Concepts

This assignment turned abstract lecture concepts into real understanding. Implementing Sobel, Laplacian, and Canny as per lecture 6 and 7 made filtering and edge detection (lecture 9 10 and 11) tangible. I clearly saw how derivatives amplify noise, Gaussian smoothing reduces it, and kernel sizes trade localization for robustness. Coding the math deepened appreciation for computer vision's theoretical foundations.

Future Work

For future assignments, I would like to explore hybrid approaches (e.g., combining classical filters with machine learning models) and test these methods on more diverse images. This would help see where the simple methods break down and where they can still compete.

Code Used

The complete implementation for this assignment can be found in the attached Python file submitted with this report:

File: (`Assigment1.py`, embedded)

References

[1] Oppenheim, A. V., & Schafer, R. W. (2010). *Discrete-Time Signal Processing*. Pearson.

[2] Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing* (3rd Edition). Pearson.

[3] OpenCV Documentation. Sobel and Laplacian Edge Detection. `https://docs.opencv.org/4.x/d2/d2c/tutorial_sobel_derivatives.html`

[4] OpenCV Documentation. Canny Edge Detector. `https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html`

[5] Course Lecture Notes. DSE 312 - Computer Vision, IISER Bhopal.

[6] Stanford CS229. *Lecture Notes on Image Processing*. Stanford University Computer Science Department. `https://web.stanford.edu/class/ee368/handouts.html`