Report

Mohammad Rishan Melethil

Abstract

This report summarizes my solutions to the four questions.Each question was implemented using Python, NumPy, OpenCV, and Matplotlib, without prebuilt filter functions. The report not only shows the outputs of the algorithms but also gives idea on the learning process, and key insights gained.

1 Introduction

The assignment was divided into four questions, each building upon core concepts of computer vision. The aim was to understand the working of image histograms, Gaussian noise, edge detection using Sobel and Laplacian filters, and implementing the Canny edge detector from scratch. goal was to learn both the mathematics and the programming logic behind image processing operations.

2 Question 1: RGB to Grayscale and Histograms

In this task, I read an RGB image, converted it to grayscale, and separated the red, green, and blue channels. For each channel, I constructed histograms from scratch without using built-in functions. This deepened my understanding of pixel intensity distributions.

Outputs

Figure 1: Original image, grayscale conversion, and separated channels.

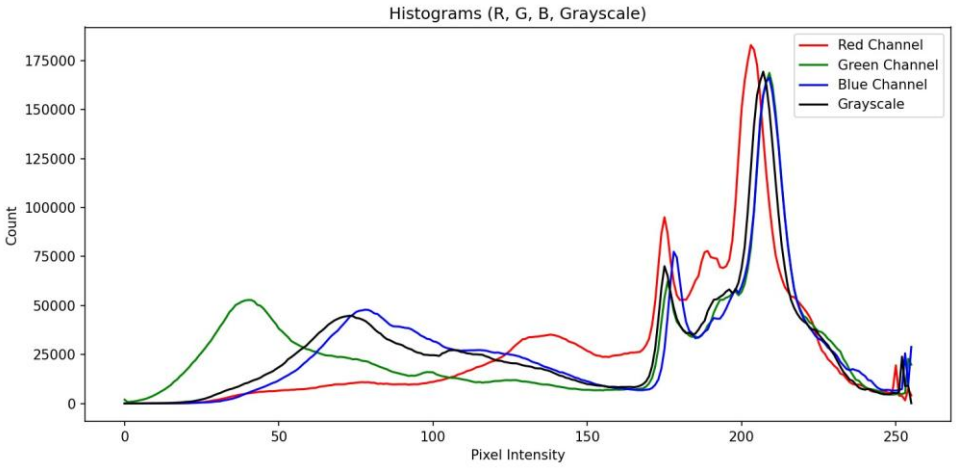

Figure 2: Histograms of RGB channels and grayscale image.

Observations

The grayscale image correctly represented intensity values. Histograms revealed how pixel intensities were distributed differently across channels, and how combining them into grayscale balances the visual information.

3 Question 2: Adding Gaussian Noise

The next step was to add Gaussian noise to the image and repeat the grayscale conversion, channel separation, and histogram analysis. Gaussian noise was generated using NumPy's np.random.normal function and added to the image using cv2.add.

Outputs

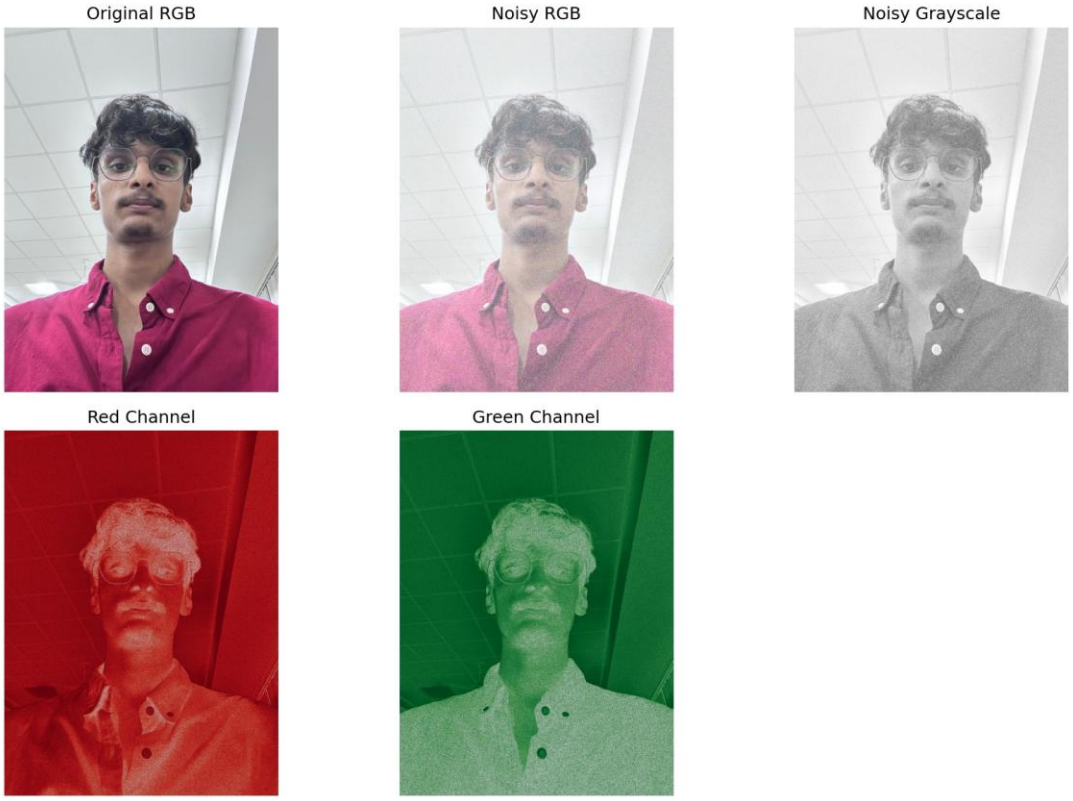

Figure 3: Original image, noisy RGB image, noisy grayscale, and noisy channels.

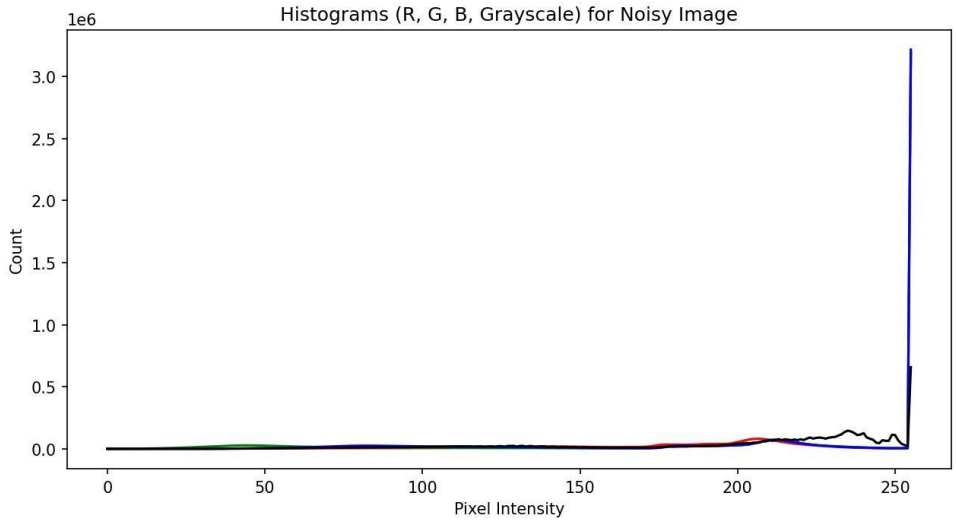

Figure 4: Histograms of noisy RGB channels and noisy grayscale image.

Observations

Adding Gaussian noise caused pixel intensity variations, which can be seen as grainy patterns in the image. Histograms spread out more, especially in brighter regions, indicating higher intensity variance due to noise.

4 Question 3: Sobel and Laplacian Filters

For edge detection, I implemented convolution manually. I applied Sobel (for directional edges) and Laplacian (for all-direction edges) filters of sizes 3 × 3, 5 × 5, and 7 × 7 to both clean and noisy grayscale images.

Outputs

Figure 5: Clean vs noisy grayscale images.

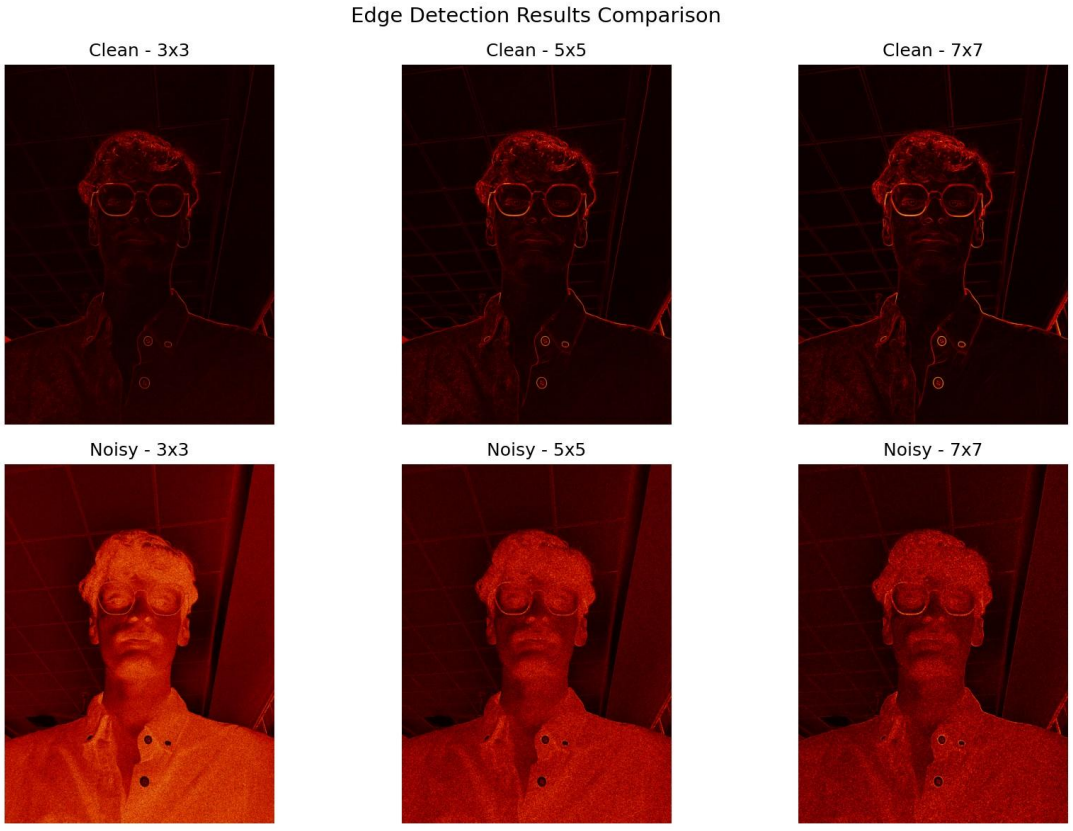

Figure 6: Sobel edge detection results for clean and noisy images with different kernel sizes.

Observations

Smaller kernels (3 × 3) captured more details but were more sensitive to noise. Larger kernels (7×7) reduced noise but blurred edges. The 5×5 kernel kept the balance between sharpness and noise reduction.

5 Question 4: Canny Edge Detection

Finally, I implemented the Canny edge detector step by step: Gaussian smoothing, gradient computation with Sobel filters, non-maximum suppression, and hysteresis thresholding. Both clean and noisy grayscale images were processed.

Outputs

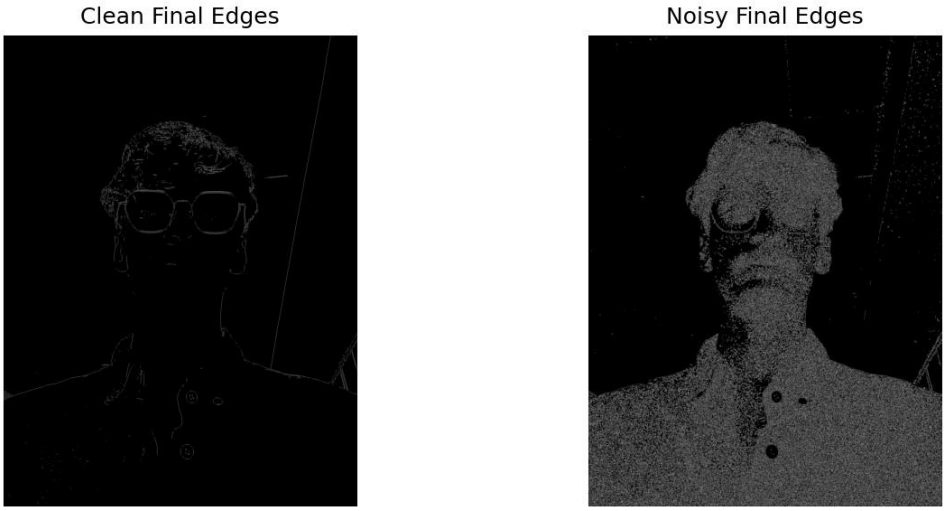

Figure 7: Final edges detected using the Canny algorithm on clean and noisy images.

Observations

The Canny detector handled noise pretty well, though noisy images still had extra edges. This exercise showed the importance of each step in the pipeline (smoothing, suppression, thresholding).

6 Reflections and Learning

Through this assignment, I learned:

- Histograms are a simple but powerful way to see how pixel intensities are spread, and converting to grayscale makes analysis much easier.
- How Gaussian noise affects images and why filtering is necessary.

- The working of Sobel and Laplacian operators, and how kernel size impacts noise and edge sharpness.
- The step-by-step process of the Canny algorithm and why it is considered one of the most reliable edge detectors.
- The importance of implementing algorithms manually to gain a deeper understanding, rather than just calling built-in functions.

7 Conclusion

This assignment gave me good experience with basic yet powerful image processing techniques. By implementing everything manually, I not only practiced Python coding but also developed a strong conceptual understanding of noise, histograms, filtering, and edge detection. These concepts are fundamental in computer vision and provide the foundation for more advanced techniques.