BIRZEIT UNIVERSITY

**Faculty of Engineering and Technology**

**Electrical and Computer Engineering Department**

**ENCS5343 Computer Vision**

# Assignment # 1

**Student name: Islam Jihad**

**Student ID: 1191375**

---

**Notes:**

**1- Use this page as a cover for your home work.**

**2- Late home works will not be accepted (the system will not allow it).**

**3- Due date is December 18th, 2023 at 11:59 pm on ritaj.**

**4- Report including Input and Output images (Soft Copy). Include all the code you write to implement/solve this assignment with your submission.**

**5- Organize your output files (images) as well as used codes to be in a folder for each question (Q1, Q2, etc.). Then add the solution of all questions along with**

**this report in one folder named** Assign1**. Compress the** Assign1 **folder and name it as (**Assign1_LastName_FirstName_StudetnsID.Zip**).**

**6- Please write some lines about how to run your code.**

**7- You might need to use OpenCV Python in your implementation.**

## Table of Contents

Table of Figures ..... 3

Introduction: ..... 4

Procedure ..... 5

- Question 1: Image Acquisition and Preprocessing ..... 5
  - 1. Image Selection and Display ..... 5
  - 2. Power Law Transformation ..... 5
  - 3. Addition of Gaussian Noise ..... 6
  - 4. Mean Filtering ..... 7
  - 5. Salt and Pepper Noise and Median Filtering ..... 7
  - 6. Mean Filtering on Salt and Pepper Noise ..... 8
  - 7. Sobel Filtering ..... 9
- Question 2: Convolution and Filtering ..... 10
  - 1. Averaging Kernel Convolution ..... 10
  - 2. Gaussian Kernel Convolution ..... 10
  - 3. Sobel Edge Detection ..... 11
  - 4. Prewitt Edge Detection ..... 11
  - 5. Comparative Analysis ..... 12
- Question 3: Noise Reduction ..... 13
  - 1. Noise Analysis and Filter Application ..... 13
  - 2. Efficacy of Median Filtering ..... 13
- Question 4: Gradient Magnitude and Orientation ..... 14
  - 1. Gradient Magnitude Computation and Visualization ..... 14
  - 2. Histogram Analysis ..... 14
  - 3. Gradient Orientation Computation ..... 15
- Question 5: Image Differencing ..... 16
  - 1. Grayscale Conversion and Differencing ..... 16
  - 2. Analysis of Differences ..... 16
- Question 6: Edge Detection with Canny Algorithm ..... 17
  - 1. Application of Canny Edge Detector ..... 17
  - 2. Threshold Variation and Effects ..... 17

Conclusion: ..... 18

Instructions on Running the Code: ..... 19

Appendix: ..... 19

## Table of Figures

Figure 0-1 Original Image ........ 5
Figure 0-2 Power Law Transformed Image ........ 6
Figure 0-3 Image with Gaussian Noise ........ 6
Figure 0-4 Mean Filtered Image (5x5) ........ 7
Figure 0-5 Image with Salt and Pepper Noise (0.1) ........ 8
Figure 0-6 Median Filtered Image (7x7) after adding salt and pepper noise ........ 8
Figure 0-7 Mean Filtered Image on Salt and Pepper Noise (7x7) ........ 9
Figure 0-8 Sobel Filtered Image ........ 9
Figure 0-9 Averaging Kernel Outputs (1x1 and 2x2) ........ 10
Figure 0-10 Gaussian Kernel Outputs with varying σ values ........ 10
Figure 0-11 Sobel Filter Outputs and Gradient Magnitude ........ 11
Figure 0-12 Prewitt Filter Outputs and Gradient Magnitude ........ 12
Figure 0-13 Difference between Sobel and Prewitt Edge Detection ........ 12
Figure 0-14 Noise Reduction on Noisy Image 1 - Original, Averaging, and Median Filter Outputs ........ 13
Figure 0-15 Noise Reduction on Noisy Image 2 - Original, Averaging, and Median Filter Outputs ........ 13
Figure 0-16 Gradient Magnitude Analysis - Original Image and Stretched Gradient Magnitude ........ 14
Figure 0-17 Histogram Analysis - Gradient Magnitude and Orientation ........ 15
Figure 0-18 Histogram of Gradient Orientation (Radians) ........ 15
Figure 0-19 Gradient Orientation Visualization in HSV Color Space ........ 16
Figure 0-20 Image Differencing - 'Walk 1', 'Walk 2', and 'Difference' ........ 16
Figure 0-21 Canny Edge Detection with varying thresholds - Low, Medium, and High ........ 17

## Introduction:

In this report, we explore various image processing techniques using Python and OpenCV. Our journey begins with fundamental operations such as displaying images and extends to more complex procedures like convolution, noise reduction, and edge detection. These tasks are not merely academic exercises; they are fundamental to understanding the intricacies of digital image processing, a field pivotal in modern technology applications ranging from medical imaging to computer vision in autonomous vehicles.

The report is structured to provide a comprehensive overview of each task, starting from the rationale behind the technique, followed by a detailed explanation of the implementation, and concluding with an analysis of the results. The tools and libraries utilized, primarily Python and OpenCV, were chosen for their robustness and widespread usage in both academic and professional circles.

This report aims not only to fulfill the requirements of the assignment but also to serve as a learning document, illustrating the power and flexibility of image processing in solving real-world problems.

## Procedure

In this report, I'll talk about every task alone, one by one in sequence.

### Question 1: Image Acquisition and Preprocessing

#### 1. Image Selection and Display

The image that was chosen for this job is 256 by 256 pixels in size and meets the requirements of an 8-bit gray-level image. The following image processing methods are based on this image as their foundation. I used a cat image from the internet.

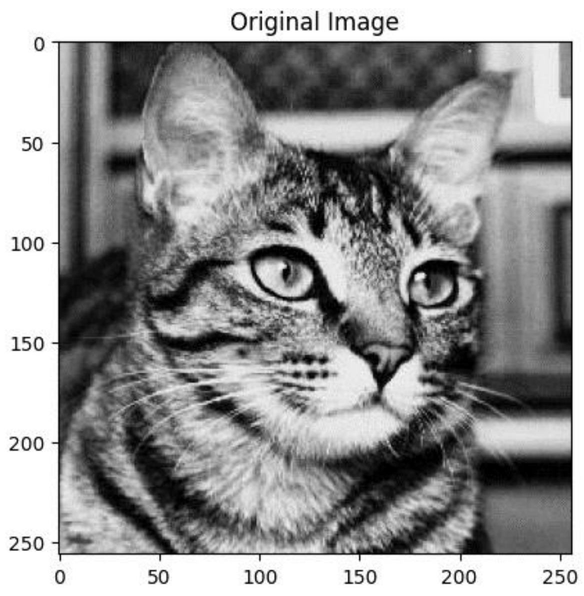

*Figure 0-1 Original Image*

#### 2. Power Law Transformation

The original image has been given a power law transformation with gamma adjusted to 0.4 to improve its visual dynamism. The final image shows how brightness and contrast are affected by gamma correction.

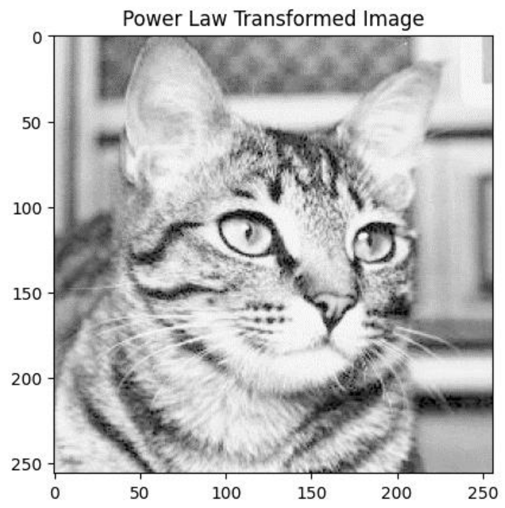

*Figure 0-2 Power Law Transformed Image*

We gave the original image a gamma correction with a value of 0.4 in the "Power Law Transformation" portion of our image processing assignment. With minimal impact on the image's brighter portions, this non-linear modification raised the brightness levels to enhance the overall visibility of the darker areas. A crucial method for boosting picture contrast for additional analysis or visual presentation is demonstrated by the transformation, which highlights mid-tone textures while maintaining the integrity of the original composition. As a result, characteristics that were previously muted in the original grayscale values may now be distinguished more clearly, as seen in Figure 2.

## 3. Addition of Gaussian Noise

To simulate typical real-world picture disruptions, 40 gray-level variance of zero-mean Gaussian noise was added to the original image.

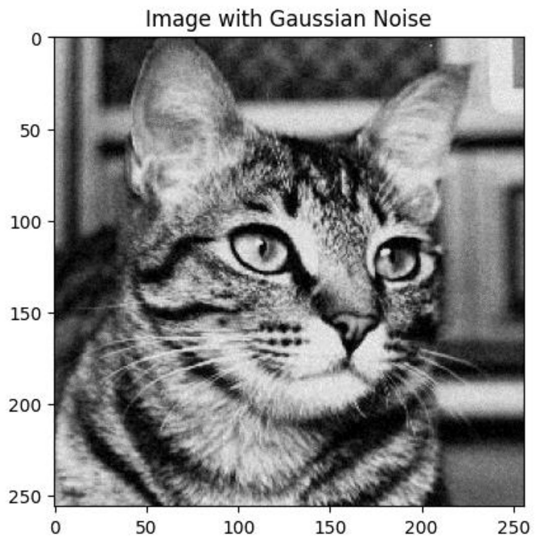

*Figure 0-3 Image with Gaussian Noise*

Zero-mean Gaussian noise adds a degree of complexity that imitates real-world situations where pictures are frequently crooked. We mimic the impression of random variations in pixel intensity, similar to sensor noise in digital photography, by introducing noise with a variance of 40 gray levels. The noise distorts the sharpness of the image, serving as a concrete illustration of typical image processing difficulties and laying the groundwork for later noise reduction methods.

## 4. Mean Filtering

The noise-damaged picture was then subjected to a 5x5 mean filter. This stage demonstrates mean filtering's capacity to reduce noise, however it tends to blur the image.

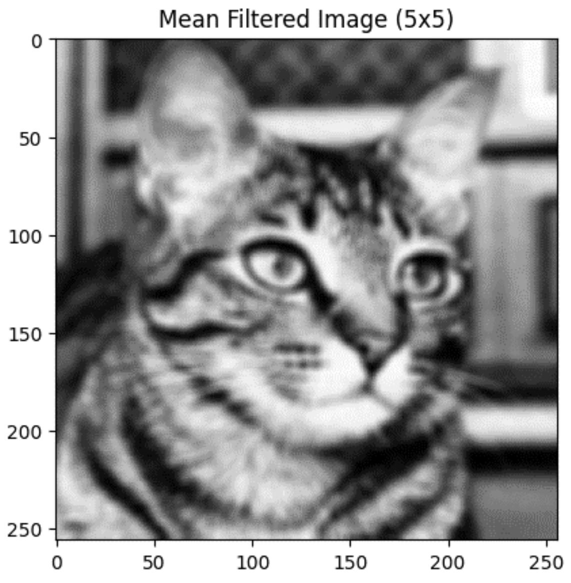

*Figure 0-4 Mean Filtered Image (5x5)*

By using a 5x5 mean filter, the Gaussian noise is reduced, indicating the filter's usefulness in reducing irregularities. The final image is less crisp but has a noticeable decrease in noise, demonstrating the trade-off between mean filtering's intrinsic loss of information and noise suppression.

## 5. Salt and Pepper Noise and Median Filtering

The original picture was processed by adding noise and then using a 7x7 median filter. The result clearly shows how good the median filter is at reducing noise, especially when it comes to maintaining edges while eliminating "salt and pepper" noise.

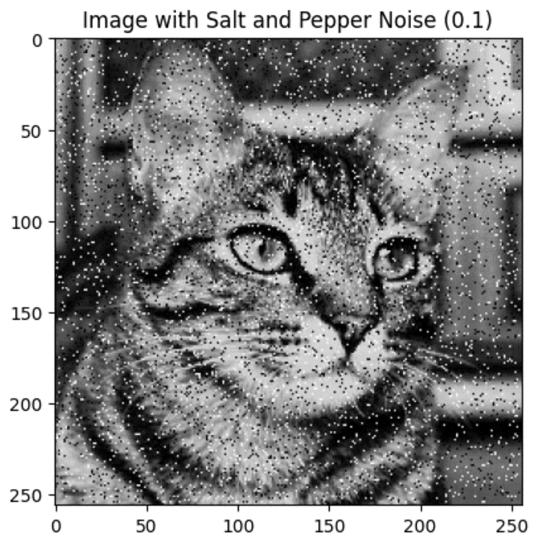

*Figure 0-5 Image with Salt and Pepper Noise (0.1)*

A new sort of difficulty is presented when black and white pixels are contrasted sharply by adding salt and pepper noise. The effectiveness of the 7x7 median filter in eliminating this kind of noise emphasizes its ability to preserve edges, which makes it particularly useful for retaining sharpness in digital photos.

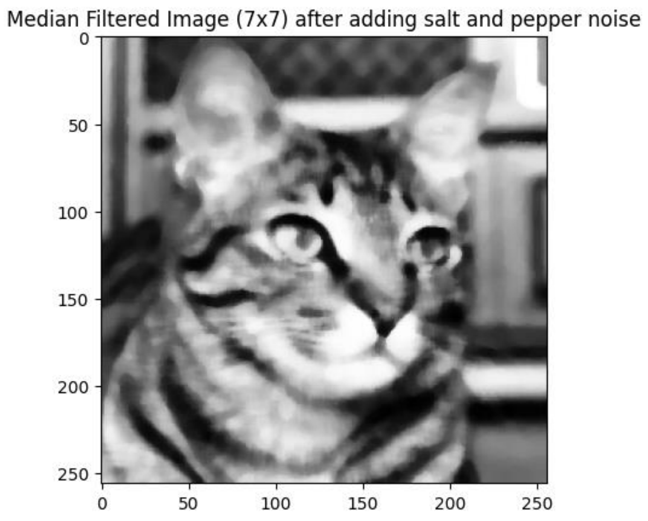
Median Filtered Image (7x7) after adding salt and pepper noise
0
50
100
150
200
250

*Figure 0-6 Median Filtered Image (7x7) after adding salt and pepper noise*

## 6. Mean Filtering on Salt and Pepper Noise

The variations in filtering approaches and their suitability in different noise circumstances are demonstrated by applying a 7x7 mean filter to the picture tainted by salt and pepper noise.

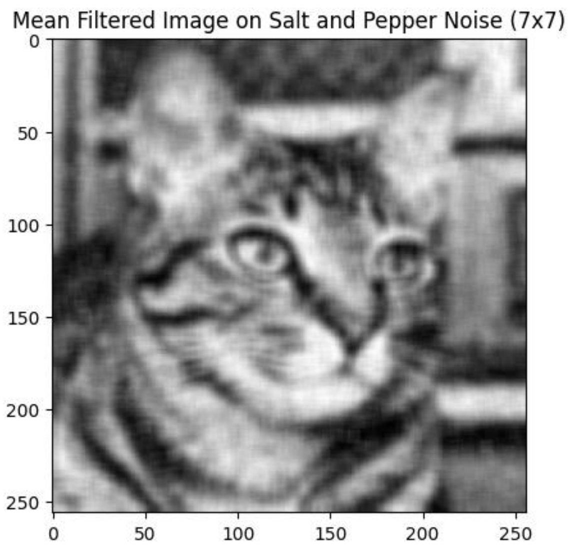

*Figure 0-7 Mean Filtered Image on Salt and Pepper Noise (7x7)*

The limits of mean filtering when dealing with noise of this kind are demonstrated by applying a 7x7 mean filter to the picture containing salt and pepper noise. It tends to distort the picture, which validates the superiority of the median filter in these circumstances.

## 7. Sobel Filtering

The original image's borders and textures are visible when the Sobel filter is applied without the use of ready functions, emphasizing the filter's sensitivity to changes in spatial gradients.

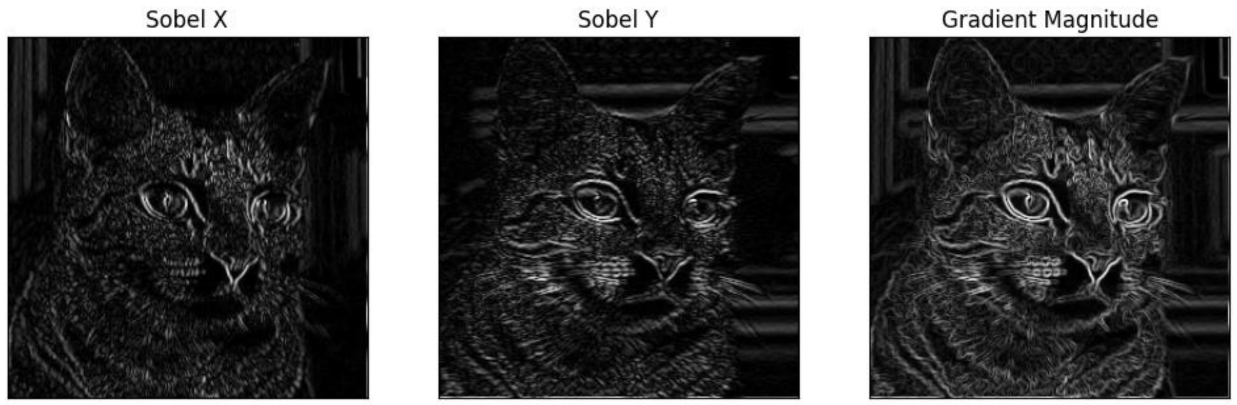

*Figure 0-8 Sobel Filtered Image*

By highlighting edges, the Sobel filter draws attention to the image's structural elements. In order to do more complex image processing operations like segmentation and pattern recognition, this technique is essential for feature extraction and object delineation.

## Question 2: Convolution and Filtering

In order to examine the impact of convolution operations on picture quality and edge recognition skills, we examine how they are applied to images using different filters in this section.

### 1. Averaging Kernel Convolution

First, we processed the original image using averaging kernels of 3x3 and 5x5 sizes. These filters contributed to a smoother look by lowering noise and detail in the image.

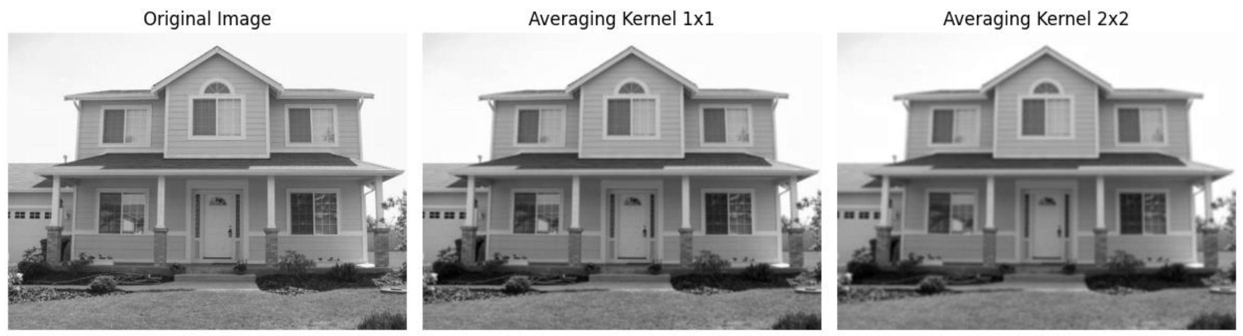

*Figure 0-9 Averaging Kernel Outputs (1x1 and 2x2)*

### 2. Gaussian Kernel Convolution

The Gaussian filter, applied with different standard deviations (σ), effectively blurs the image while preserving edge boundaries. The larger the σ, the more pronounced the blurring effect.

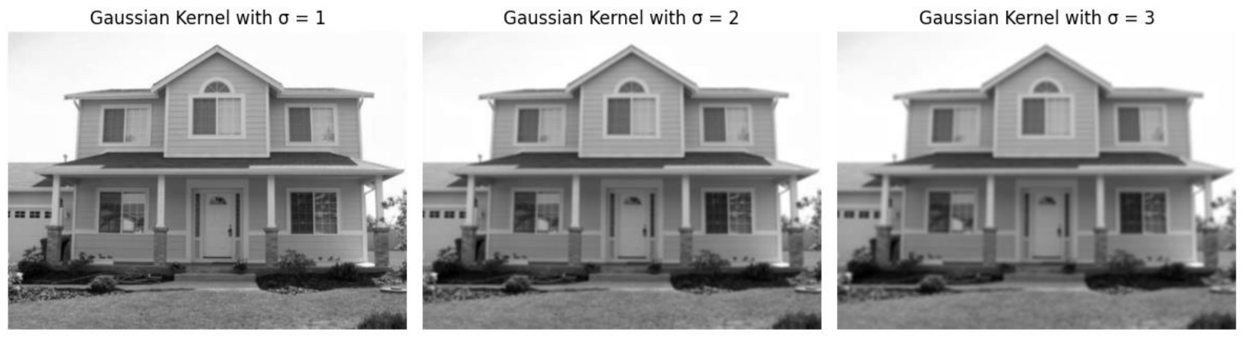
Gaussian Kernel with σ = 1
Gaussian Kernel with σ = 2
Gaussian Kernel with σ = 3

*Figure 0-10 Gaussian Kernel Outputs with varying σ values*

Our examination of convolution and filtering reveals that, while averaging kernels may be helpful for some smoothing applications, it can also blur the image and reduce noise and detail. Gaussian filters are perfect for situations where edge preservation matters since they produce a smoother output while better retaining edges, especially when the σ value is larger.

## 3. Sobel Edge Detection

The Sobel operator was then employed to highlight the edges in both horizontal and vertical directions, as well as the overall gradient magnitude.

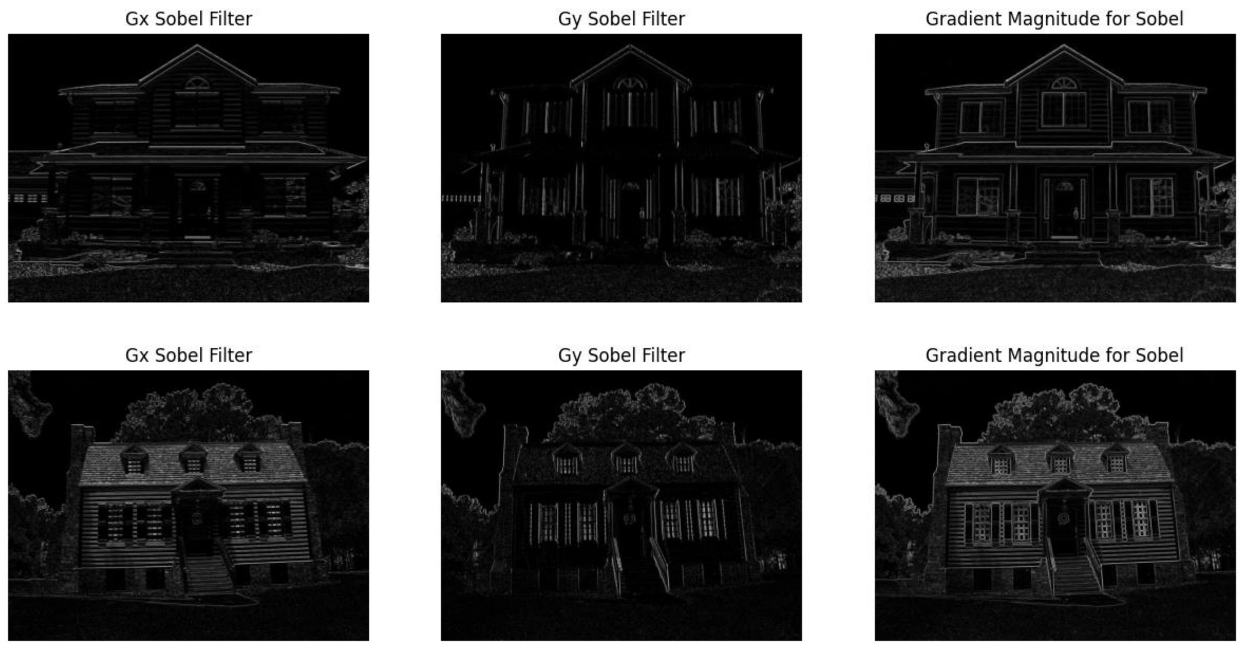

*Figure 0-11 Sobel Filter Outputs and Gradient Magnitude*

## 4. Prewitt Edge Detection

Similar to the Sobel filter, the Prewitt operator was used for edge detection. It has a different convolution kernel that affects the edge detection sensitivity.

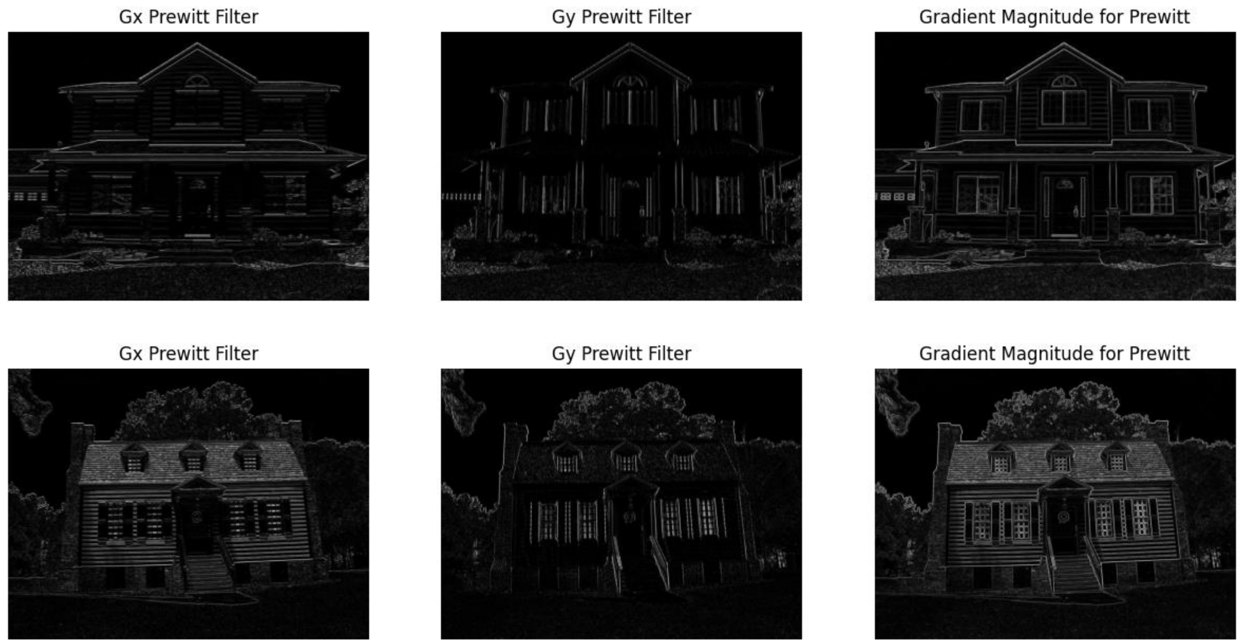

*Figure 0-12 Prewitt Filter Outputs and Gradient Magnitude*

## 5. Comparative Analysis

Finally, a comparative analysis was conducted between the Sobel and Prewitt operators by examining the difference in their edge detection results on the test images.

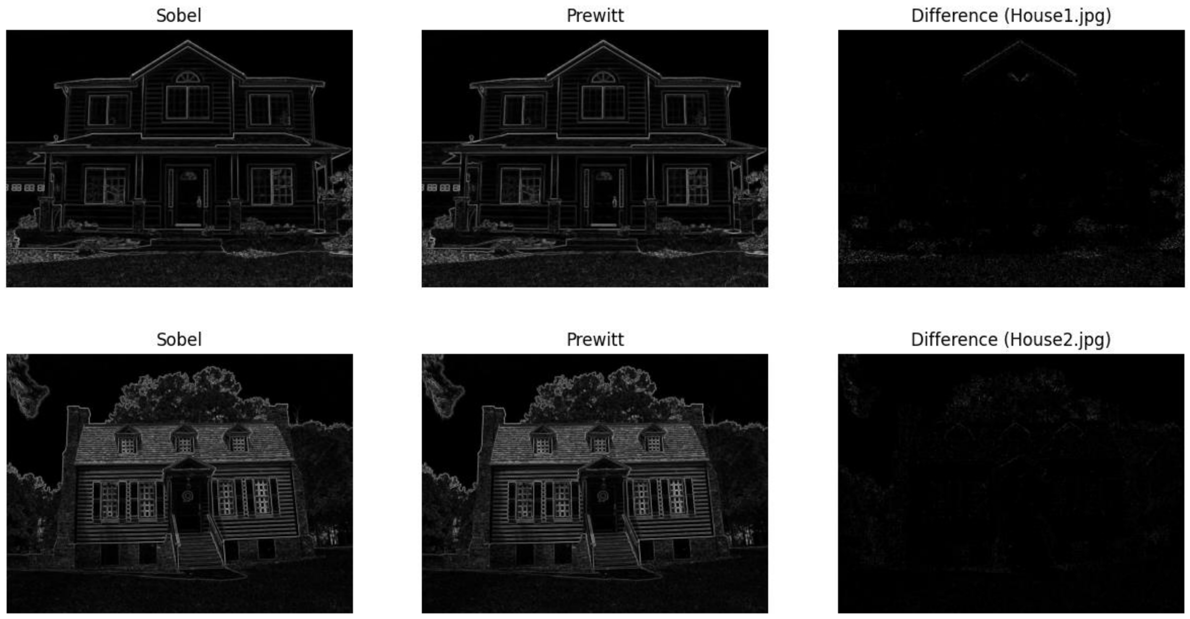

*Figure 0-13 Difference between Sobel and Prewitt Edge Detection*

Sobel and Prewitt filters are more focused on edge detection, with Sobel providing slightly more emphasis on diagonal edges. The comparative analysis underscores that while both detect edges effectively, the choice between them may depend on the specific characteristics of the image and the desired sharpness of edges.

The gradient magnitude images are crucial for visualizing the strength of edges and can play a significant role in subsequent image processing tasks such as object detection and segmentation.

## Question 3: Noise Reduction

### 1. Noise Analysis and Filter Application

With this job, we dealt with a typical problem in digital imaging: photos tainted by noise from salt and pepper. We evaluated the noise reduction efficacy of the 5x5 averaging and median filters on two noisy photos.

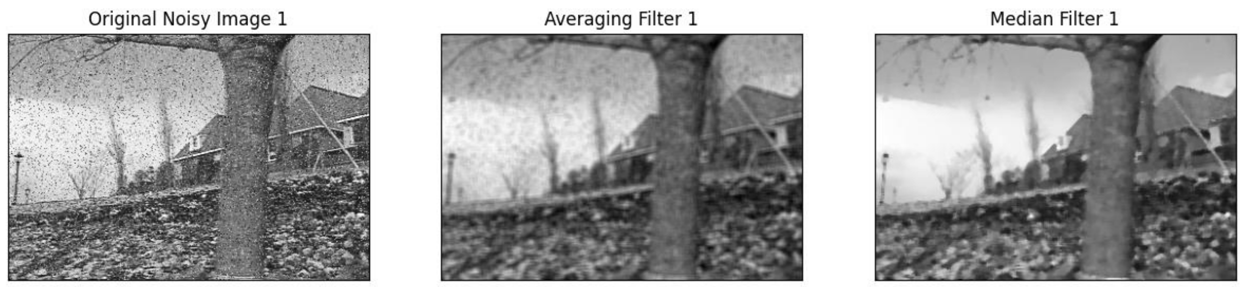

*Figure 0-14 Noise Reduction on Noisy Image 1 - Original, Averaging, and Median Filter Outputs*

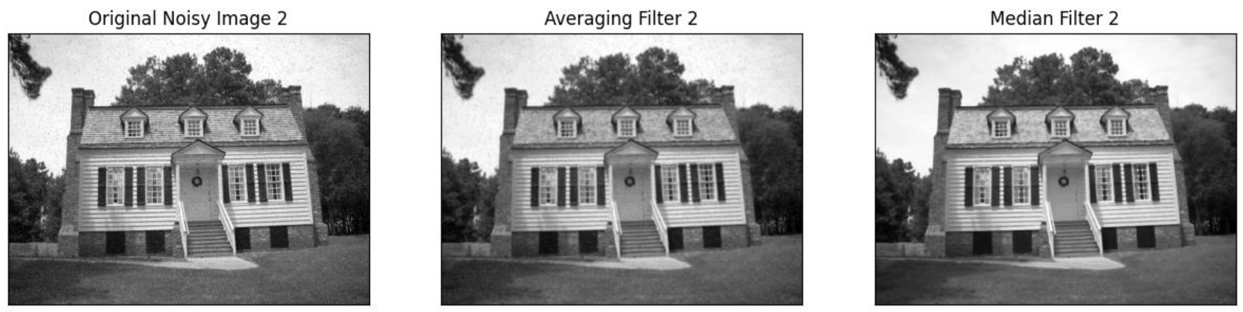

*Figure 0-15 Noise Reduction on Noisy Image 2 - Original, Averaging, and Median Filter Outputs*

### 2. Efficacy of Median Filtering

In these situations, the median filter works better than the averaging filter. Because the high noise values continue to affect the average, the averaging filter distorts the image and is unable to effectively eliminate the salt and pepper noise. The borders and features of the image are preserved by the median filter, which performs well by rejecting the extreme values and determining the median of the surrounding pixels.

The visual contrast presented in Figures 14 and 15 amply illustrates how well the median filter performs in terms of preserving picture quality and efficiently eliminating noise. The median filter is recommended for image processing jobs that involve reducing noise in the salted and pepper domain. This example demonstrates why.

## Question 4: Gradient Magnitude and Orientation

### 1. Gradient Magnitude Computation and Visualization

Using Sobel operators, we calculated the image's gradient magnitude. The gradient magnitudes were expanded to cover the whole range of 0 to 255 in order to improve viewing.

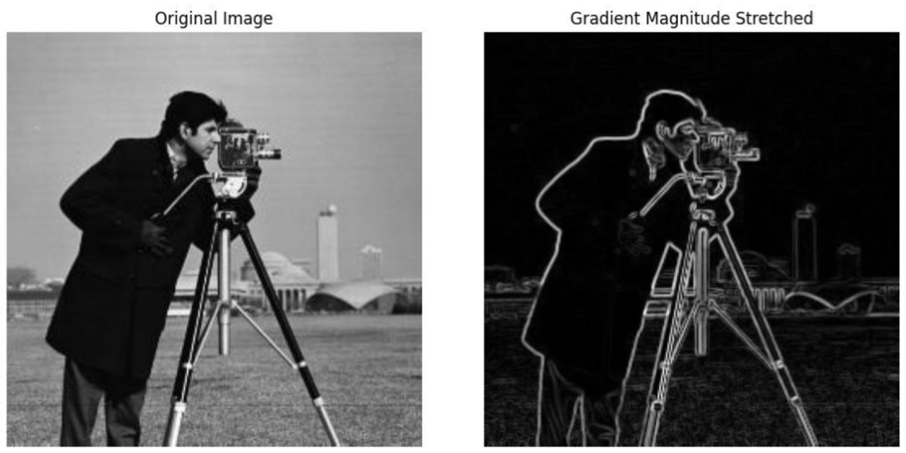

*Figure 0-16 Gradient Magnitude Analysis - Original Image and Stretched Gradient Magnitude*

### 2. Histogram Analysis

The distribution and orientation of the edges in the picture may be deduced from the gradient magnitude and orientation histograms.

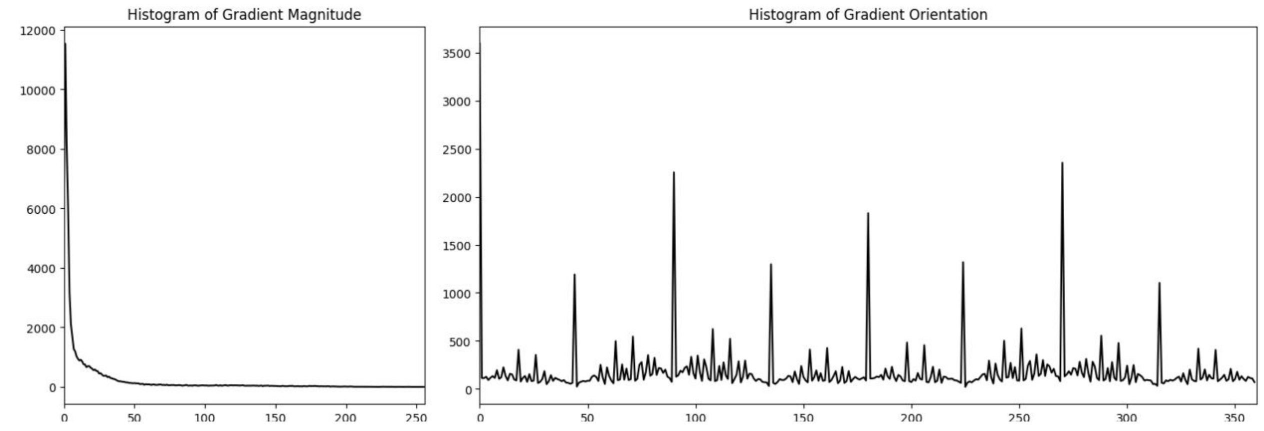

*Figure 0-17 Histogram Analysis - Gradient Magnitude and Orientation*

It's important to note that the gradient orientation histogram displays an uneven distribution of edge directions, which suggests that the texture of the picture is not uniform. The image's dominating edge orientations are shown as peaks in the histogram, which most likely indicate prominent structural lines or features. Man-made settings are characterized by the existence of strong vertical and horizontal lines, which appear as peaks at 0, π/2, and π radians. For tasks like object identification, scene interpretation, and even directing feature extraction for machine learning models, it is essential to comprehend edge orientation. The extra histogram, which gives a numerical representation of the frequency of edges at each orientation, supports these findings.

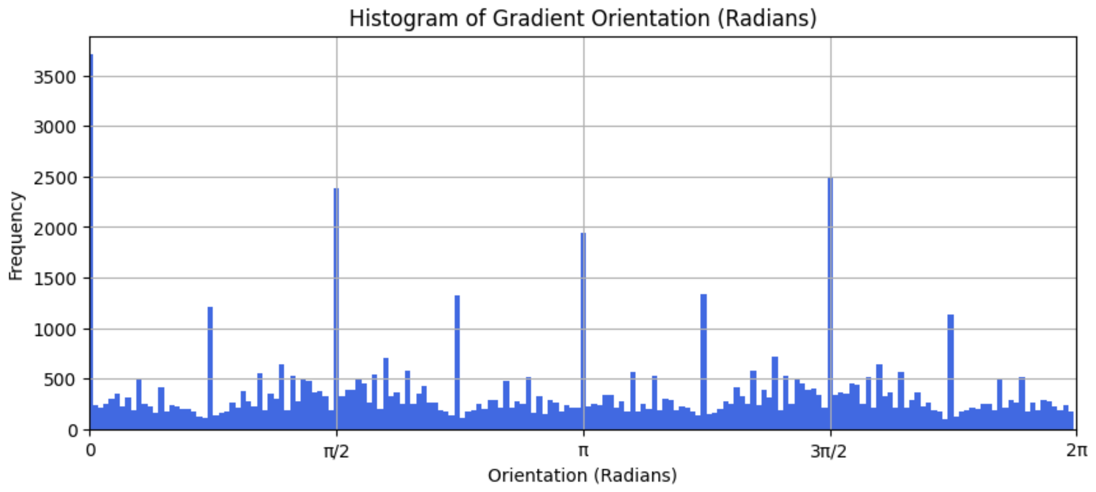

*Figure 0-18 Histogram of Gradient Orientation (Radians)*

## 3. Gradient Orientation Computation

Understanding the texture and structure of the topic requires knowing the gradient orientation, which was computed to disclose the angle of the gradient vectors across the image.

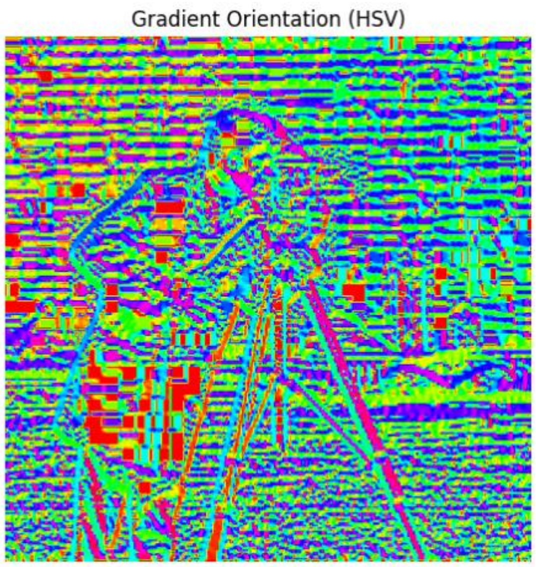

*Figure 0-19 Gradient Orientation Visualization in HSV Color Space*

The histograms and visualizations collectively offer a detailed portrayal of the image's edge and texture characteristics, essential for advanced image processing applications.

## Question 5: Image Differencing

### 1. Grayscale Conversion and Differencing

Grayscale conversion was applied to both the "walk 1.jpg" and "walk 2.jpg" photos in order to equalize color fluctuations and highlight intensity discrepancies. 'walk 2.jpg' is subtracted from 'walk 1.jpg' to show the differences between the two pictures.

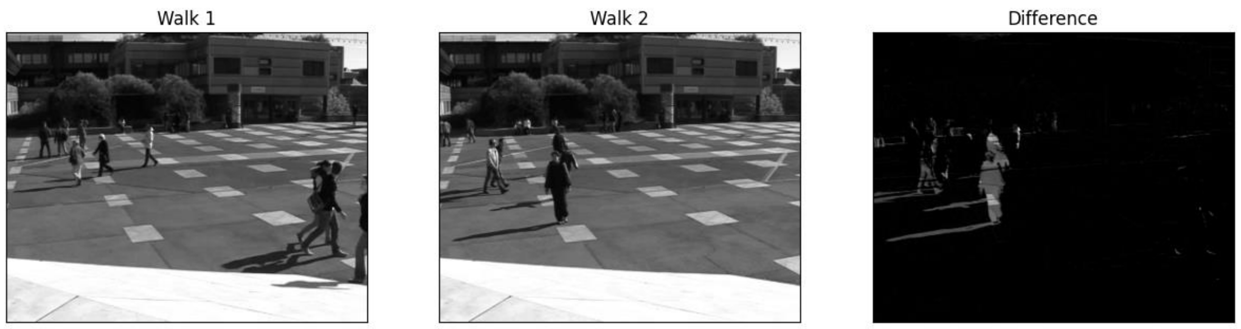

*Figure 0-20 Image Differencing - 'Walk 1', 'Walk 2', and 'Difference'*

### 2. Analysis of Differences

The differential picture that results highlights regions of change, especially when people are walking or moving items are present. The moving subjects seem bright, displaying their pathways and motions between the two frames, while the static background stays black, suggesting little to no change.

This method is essential for motion detection and is frequently applied in dynamic imaging applications such as sports analysis, video surveillance, and others.

## Question 6: Edge Detection with Canny Algorithm

### 1. Application of Canny Edge Detector

Using OpenCV's Canny function, the Canny edge detection technique was applied to 'Q 4.jpg'. This method uses a multi-stage procedure to determine the most significant edges in photos, allowing it to detect a wide variety of edges.

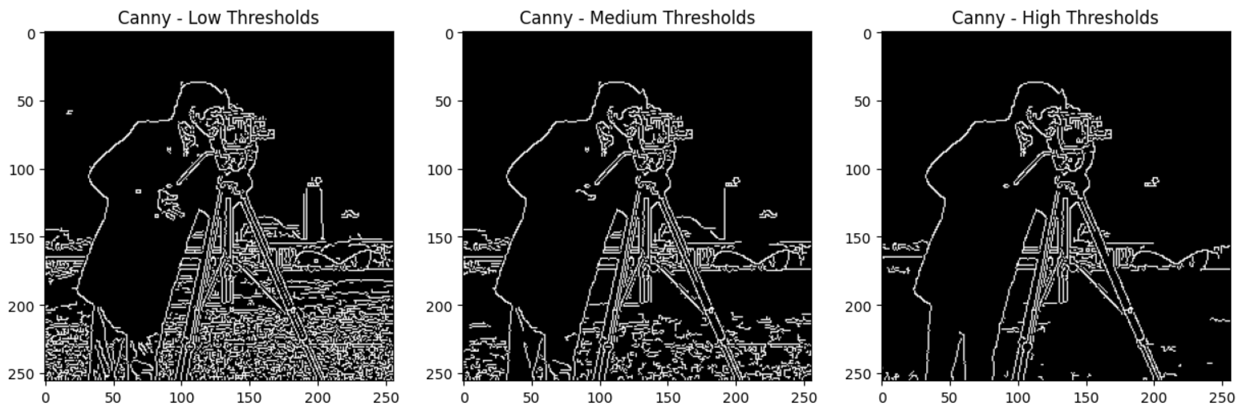

*Figure 0-21 Canny Edge Detection with varying thresholds - Low, Medium, and High*

For low I used threshold1=50, threshold2=100

For mid I used threshold1=100, threshold2=200

For high I used threshold1=150, threshold2=300

### 2. Threshold Variation and Effects

Variations in threshold settings govern the edge detection's sensitivity. While high thresholds produce fewer but more noticeable and important edges, low thresholds often catch more edges, including finer details. The algorithm's capacity to distinguish between real edges and noise is greatly impacted by the threshold selection, which also has a significant effect on the edge detection result.

The significance of choosing suitable threshold values for edge detection tasks is emphasized in this section. The goal is to strike a balance between detecting too many edges (including possible noise) and missing important edges.

## Conclusion:

This report's conclusion summarizes the wide variety of image processing methods examined. Every job, from the basic one of displaying images to the intricate one of detecting edges, has offered important insights into the difficulties and factors involved in manipulating digital images. The project has been a useful exercise in comprehending the impacts of different filters and transformations, illuminating the subtleties of edge recognition, noise reduction, and picture enhancement. When considering these learning objectives, it becomes clear that the careful implementation of these methods is essential in the field of computer vision, potentially having an impact on a variety of real-world situations.

## Instructions on Running the Code:

The simulation analyses in this assignment are divided into six Jupyter Notebook files, each of which corresponds to a distinct study component. To guarantee smooth operation and precise outcomes, kindly adhere to following guidelines:

1. **Environment Setup**: Set up a Python environment identical to the one being used for this project first. Installing Jupyter Notebook and Python (version 3.11.4) are required for this. Install the necessary libraries as well, following the instructions in the'requirements.txt' file located in the project repository. In your Python environment, you can usually accomplish this by using the command **'pip install -r requirements.txt'**.

2. **Opening the Notebooks**: Once the environment is prepared, run Jupyter Notebook for each notebook by going to the project directory. Using '**jupyter notebook**' in your terminal or command prompt will allow you to accomplish this. This will launch a window in your browser where you may access and open each notebook file.

3. **Running the Code**: Each notebook's code is broken up into cells. Sort these cells in order of top to bottom in each notebook. Certain cells could take longer to complete than others, particularly if they need intricate data processing or computations. Make sure that before going on to the next cell, every cell has finished its execution.

4. **Viewing the Results**: The code that produces the results is shown right after it in the notebook cells. This might involve tables, graphical representations, or textual outputs. See the markdown texts and comments in each notebook for a thorough analysis that includes context and justifications for the findings.

You should be able to duplicate the analysis and findings in this report by following these steps. Please contact me if you have any questions or need any clarification.

## Appendix:

The full code are in the attached files as they are long to append here.