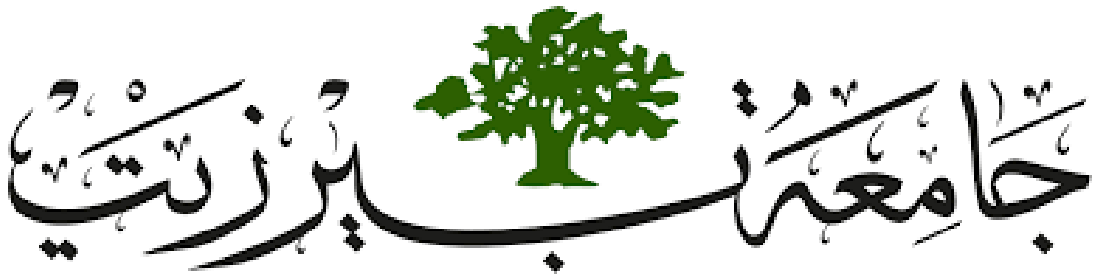


Birzeit University
Faculty of Engineering & Technology
Electrical & Computer Engineering Department
Computer Vision - ENCS5343
Assignment #1

Prepared by: Nsralla Hasan - 1200134

Instructor: Dr. Aziz Qaroush
Section: 1

Date: 10th November 2024

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

Image noise reduction is an important part of computer vision. It involves improving low-quality, noisy images to make them clear and sharp.
The main goal of this assignment is to perform a comparison of several simple smoothing filters, Box, Gaussian, and Median filters, with the more advanced ones like Adaptive Mean, Adaptive Median, and Bilateral filters, which have been applied for noisy image cleaning.
I will test these filters to find out which can give better performance regarding noise removal while keeping important details and edges in images. I am also going to look at the speed of each filter and at the impact of variation in filter size on its performance.
Later on, I will use the Mean Squared Error and Peak Signal-to-Noise Ratio metrics that are simple and describe how well each filter works. Furthermore, a Canny edge detector will be applied in order to see how well the filters can preserve edges and other important details. This would help in choosing the right filter for different image types and associated noise. Such techniques increase the reliability of computer vision applications: object recognition, driving autonomous cars, and analyzing medical images.

1.2 Filters Being Compared

In this assignment, I will present a comparison of six different filters for image denoising. Each filter decreases the noise of an image in its own way with an effort to keep the important details and edges. Below is a small description of each filter:

- **Box filter**: Also named average filter. Smooths a given image by replacing the value of every pixel with the mean value of neighboring pixels in a window specified around it.

- **Gaussian filter**: smooths an image by performing an average where the weighted pixel values are higher when nearer to the center of the kernel. Weights are distributed as a Gaussian, so the distribution of weights is bell-shaped.

- **Median filter**: replaces the value of every pixel by the median value of the neighboring pixels in the kernel.

- **Adaptive mean filter**: adapts the averaging process according to the local characteristics of the image. It either varies the size of the kernel or the weights used according to the values of the surrounding pixels.

- **Adaptive median filter**: : The Adaptive Median filter adapts its kernel size to the noise level approach locally. It keeps increasing the kernel size for a median free of noise.

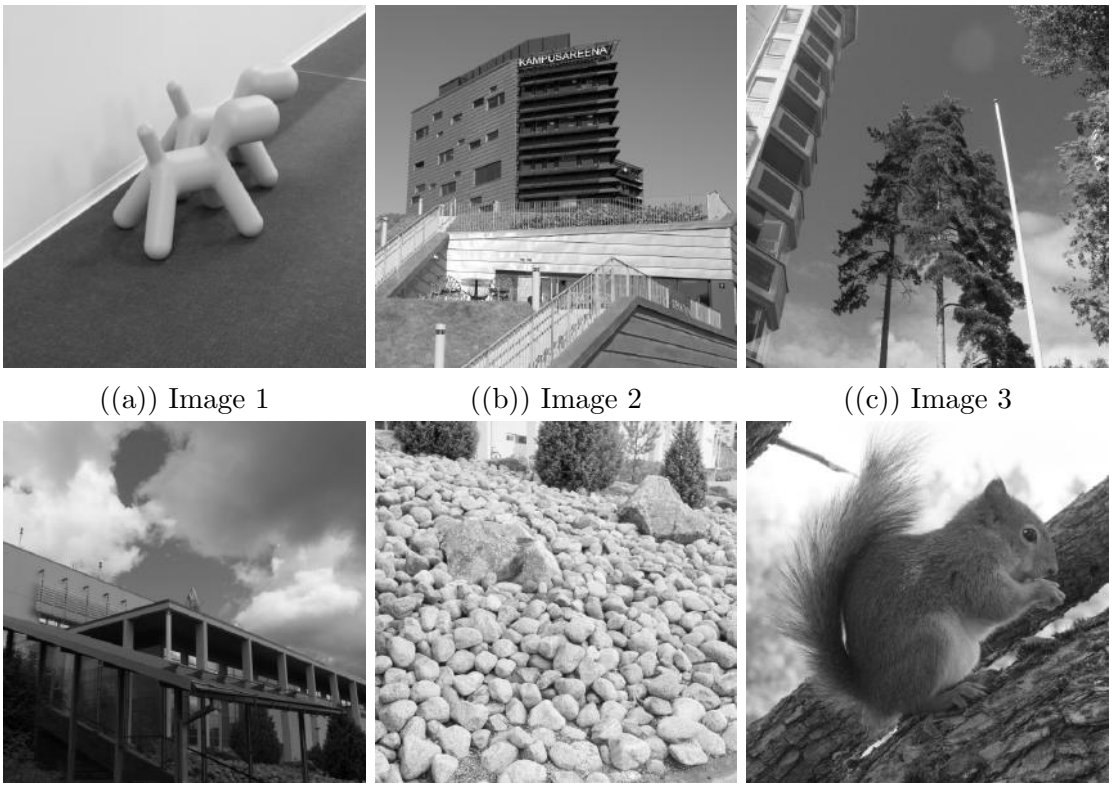
- **Bilateral filter**:his Bilateral filter smooths the images while preserving the edges, considering the spatial distance and the intensity difference between pixels. It gives higher weights meanwhile to those pixels that are closer in space and intensity.

Table 1: Summary of Filter Characteristics

Filter	Noise Reduction	Edge Preservation	Computational Efficiency	Kernel Size Effect
Box Filter	Moderate	Low	High	Larger kernels increase blurring
Gaussian Filter	High	Moderate	Moderate	Larger kernels improve smoothing
Median Filter	High (for Salt-and-Pepper)	High	Moderate	Larger kernels can handle more noise
Adaptive Mean Filter	High	High	Lower than simple filters	Adjusts to local noise levels
Adaptive Median Filter	High (especially for Salt-and-Pepper)	High	Lower than simple filters	Dynamically adjusts kernel size
Bilateral Filter	High	Very High	Low	Maintains edge details with larger kernels

2 Experiments

In this assignment, 6 main images (512 x 512)px were used.



((a)) Image 1 ((b)) Image 2 ((c)) Image 3

((d)) Image 4 ((e)) Image 5 ((f)) Image 6

Figure 1: Used Images

2.1 Noise Generation

I added two types of noise, Gaussian noise and Salt-and-Pepper noise. For each kind of noise, there are three different intensities: low, medium, and high. And each image has experienced different noise with different level. Each type of noise parametric description is given below.

2.1.1 Gaussian Noise

Gaussian noise was added to each image using the following parameters:

- **Mean (μ)**: 0
- **Standard Deviation**:
  - Low: $(\sigma) = 10$
  - Medium: $(\sigma) = 30$
  - High: $(\sigma) = 50$

To generate the Gaussian noise, a normal distribution with the specified mean and standard deviation was used. This noise is further added to the original image, and the resultant pixel values were clipped to ensure that they remain within the valid range of [0, 255].

2.1.2 Salt-and-Pepper Noise

Salt-and-Pepper noise was introduced with varying probabilities based on the chosen intensity level. The parameters for each intensity level are summarized in Table 2.

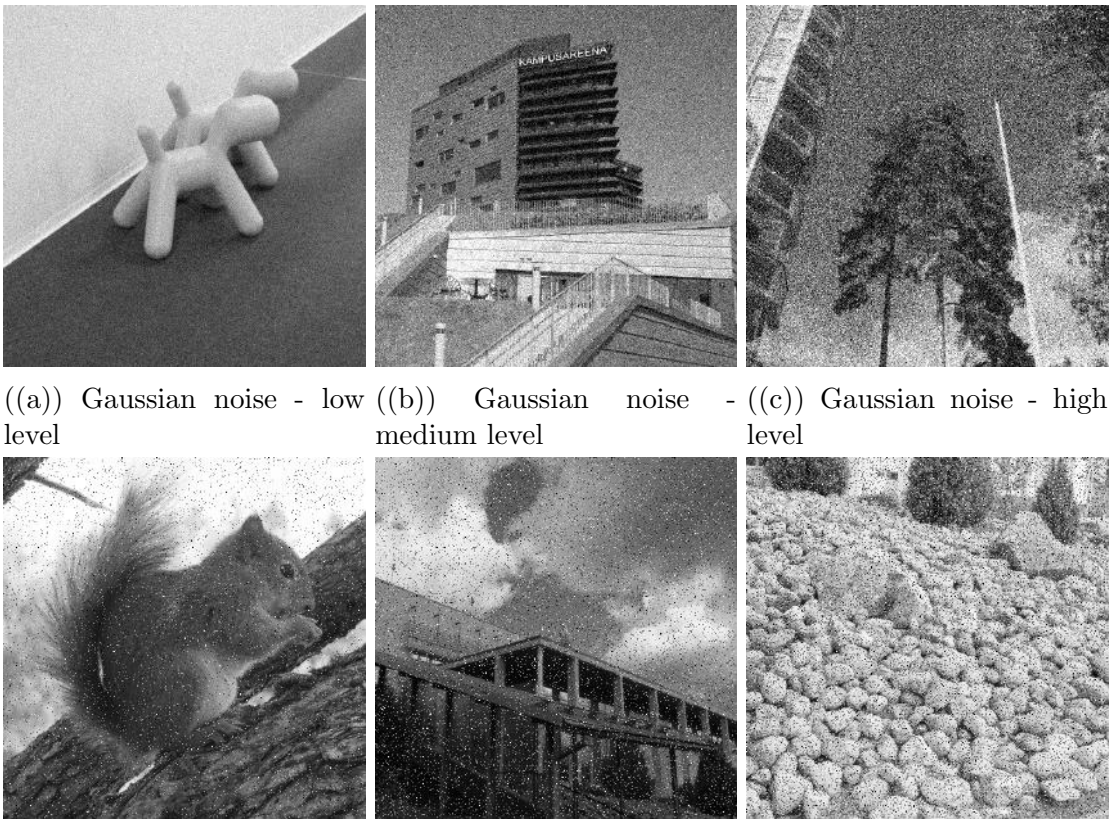
Table 2: Parameters for Salt-and-Pepper Noise at Different Intensity Levels

Intensity Level	**Noise Type**	**Salt Probability**	**Pepper Probability**
Low	Salt Paper	0.02	0.02
Medium	Salt Paper	0.04	0.04
High	Salt Paper	0.08	0.08

For each intensity level:

- **Low**: A small percentage of pixels are randomly selected to be altered, resulting in minimal noise.
- **Medium**: A moderate percentage of pixels are affected, introducing noticeable noise.
- **High**: A large percentage of pixels are altered, creating significant noise that heavily weaknes image quality.

The following grid shows the images after adding noise.



((a)) Gaussian noise - low level

((b)) Gaussian noise - medium level

((c)) Gaussian noise - high level

((d)) salt paper noise - low level

((e)) salt paper noise - medium level

((f)) salt paper noise - high level

Figure 2: Used Images after adding noise

2.2 Filter Implementation

Python was the language of implementation for applying and implementing filters. This language had several libraries used in image processing, numerical computation, and visualization. The most utilized library for image manipulation was OpenCV, Open Source Computer Vision Library, which provides efficient and optimized functions for filtering and edge detection. Other libraries included NumPy for numerical operations, Matplotlib for plotting and visualization, and scikit-image to calculate metrics of similarity between images.
In order to test how kernel size influences the performance of each filter, three different kernel sizes were taken into consideration:

- **Small Kernel**: 3×3
- **Medium Kernel**: 5×5
- **Large Kernel**: 7×7

2.2.1 Box Filter

Implementation: The Box filter has been implemented using the `cv2.filter2D` function from OpenCV. The normalizing kernel was created using NumPy, where each element value of the kernel is

$$\frac{1}{\text{kernel\_size} \times \text{kernel\_size}}$$

Specific Parameters:

- **Kernel Sizes**: 3, 5, 7
- **Application**: The filter was applied to all noisy images with different kernel sizes. To assess computational efficiency, the filtering execution time was measured using the `time` module.

2.2.2 Gaussian Filter

Implementation: The Gaussian filter was implemented using OpenCV's `cv2.GaussianBlur` function. This function convolves the image with a Gaussian kernel, where weights decrease with distance from the center pixel, following a Gaussian distribution.
Specific Parameters:

- **Kernel Sizes**: 3, 5, 7 (must be odd integers)
- **Sigma (σ)**: Set to 0, allowing OpenCV to calculate σ based on the kernel size.

Application: The Gaussian filter was applied in the same way as the Box filter, processing each noisy image with the specified kernel sizes. Computational time was also measured to evaluate performance.

2.2.3 Median Filter

Implementation: The Median filter was applied using the `cv2.medianBlur` function available in the OpenCV library. This filter replaces each pixel's value with the median value of neighboring pixels within the kernel window, effectively reducing salt-and-pepper noise.
Specific Parameters:

- **Kernel Sizes**: 3, 5, 7

Application: The noisy images were processed with the Median filter using different kernel sizes, and the time taken during filtering was measured to assess computational efficiency.

2.2.4 Adaptive Mean Filter

Implementation: The Adaptive Mean Filter was implemented using a function that calculates the local mean and variance within a specified kernel size. This filter adapts to local image statistics by comparing the local variance with the global variance of the entire image. If the local variance is less than the global variance, the pixel is replaced by the local mean; otherwise, the original pixel value is kept.
Specific Parameters:

- **Kernel Sizes**: 3, 5, 7
- **Global Variance**: Computed from the entire image to serve as a threshold.
- **Local Mean and Variance**: Calculated using the `uniform_filter` function from the `scipy.ndimage` module.
- **Data Types**: Images were converted to `float32` for greater precision in calculations.

Application: The filter was applied to each noisy image with the specified kernel sizes. Processing time was recorded using the `time` module to evaluate computational efficiency.

2.2.5 Adaptive Median Filter

Implementation: The Adaptive Median Filter uses a custom function that automatically adjusts the filter window size based on local image characteristics. The algorithm operates in two steps:

- **Level A**: Determines if the median value represents an impulse (noise) by checking if it lies within the minimum and maximum values in the window.
- **Level B**: If within range, the current pixel is replaced with the median value; otherwise, it remains unchanged. The window size iteratively expands from the initial size up to a maximum limit until the criteria are met.

Specific Parameters:

- **Initial Window Size**: 3×3
- **Maximum Window Size**: 7×7
- **Data Types**: Images were converted to `int16` to avoid overflow during calculations.

Application: The filter was applied to each noisy image, starting with the smallest window size and increasing it by 2 as necessary. This adaptive method effectively removes salt-and-pepper noise while preserving critical details and edges. Processing time and the final window size used for each pixel were recorded for analysis.

2.2.6 Bilateral Filter

Implementation: The Bilateral Filter was applied using OpenCV's `cv2.bilateralFilter` function. This filter reduces noise while preserving edges by considering both the spatial closeness and the intensity difference of neighboring pixels, replacing each pixel's intensity with a weighted average from neighboring intensities.

Specific Parameters:

- **Kernel Sizes (Diameter)**: 3, 5, 7
- **Sigma Color (σ Color)**: Controls the influence of intensity differences. Values were set to 75 for smaller kernels, adjusted for larger kernels (105 for a 7-size kernel).

- **Sigma Space (σ Space)**: Controls spatial distance influence(105 for a 7-size kernel).
- **Data Types**: Images remained in `uint8` format, as required by OpenCV functions.

Application: The filter was applied to each noisy image with different kernel sizes and corresponding sigma values. Processing time was measured to assess computational efficiency.

2.3 Evaluation Methods

2.3.1 Edge Detection and Performance Metrics

Edge Detection: The Canny edge detection algorithm was applied using `cv2.Canny` with low and high thresholds set at 100 and 200, respectively. This was performed on original, noisy, and filtered images to assess how well edges were preserved after filtering. **Visualization**: Matplotlib was used to display the original, noisy, and filtered images alongside their edge maps for qualitative analysis.

Performance Metrics:

- **Mean Squared Error (MSE)**: Calculated to quantify the difference between the original and filtered images.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(I_{\text{original}}(i) - I_{\text{filtered}}(i)\right)^2$$

- **Peak Signal-to-Noise Ratio (PSNR)**: Computed based on the MSE to evaluate the quality of the filtered images.

$$\text{PSNR} = 20 \cdot \log_{10}\left(\frac{\text{MAX}_{\text{pixel}}}{\sqrt{\text{MSE}}}\right)$$

  where $\text{MAX}_{\text{pixel}} = 255$ for 8-bit images.

- **Edge Density**: Determined by calculating the proportion of edge pixels (pixels with a value of 255) to the total number of pixels in the edge-detected image.

$$\text{Edge Density} = \frac{\text{Number of Edge Pixels}}{\text{Total Number of Pixels}}$$

- **Structural Similarity Index (SSIM)**: Used to compare the similarity between the edges of the original image and the filtered image, providing a perceptual measure of image quality.

2.3.2 Computational Time Measurement Methodology

For each of the filters, computational time was measured using Python's built-in time module. Time was recorded immediately before and after the filtering operation, and the difference provided the elapsed time.

3 Results

In this section I will put each filter performance against each noise followed by canny edge detector applied for the original, noisy, filtered image for the 3 kernel sizes.

3.1 Box-Filter

3.1.1 First Image - Gaussian Noise - low Level

Edge Density

- **Original Image**: 0.0072
- **Noisy Image**: 0.1182

Table 3: Box Filter Results - Gaussian Noise (Low Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	30.6474	33.2669	0.0000	0.0064	0.2522	0.9664
5	19.8156	35.1607	0.0000	0.0045	0.2522	0.9811
7	17.8415	35.6165	0.0000	0.0027	0.2522	0.9594

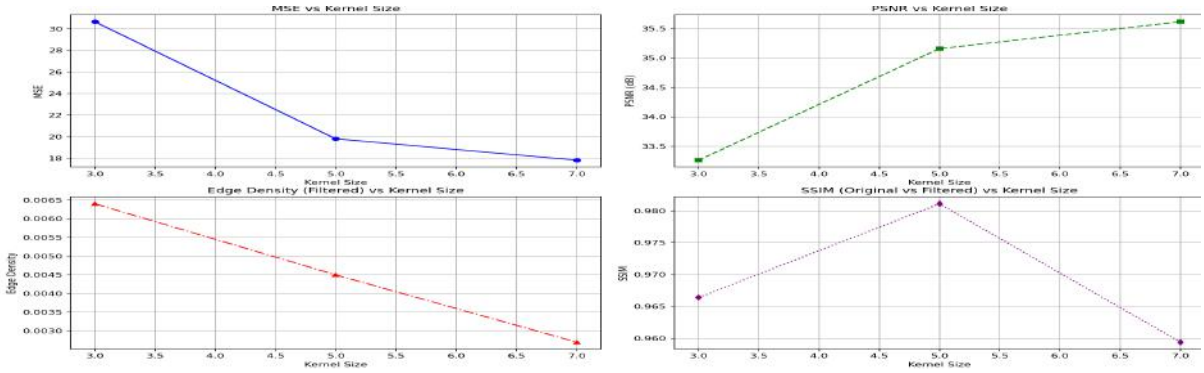
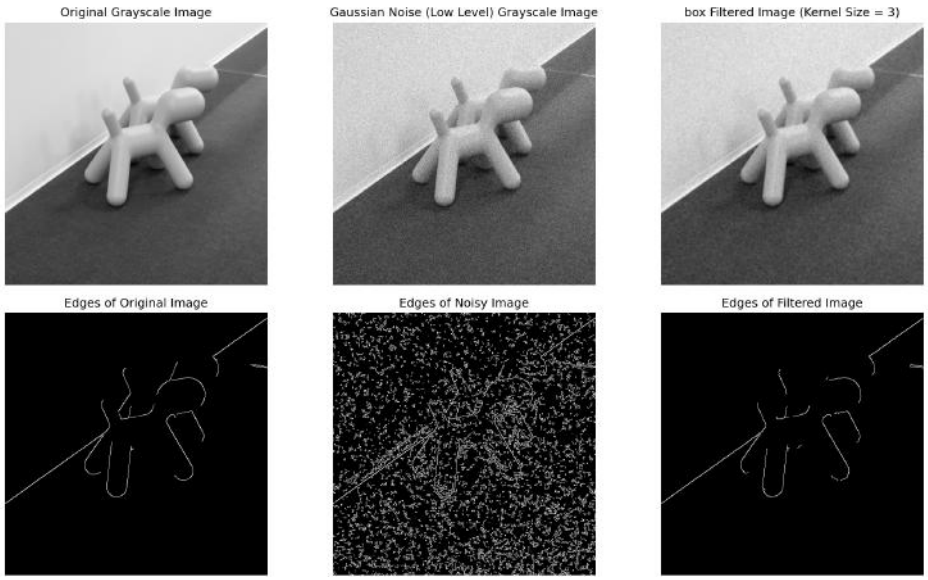
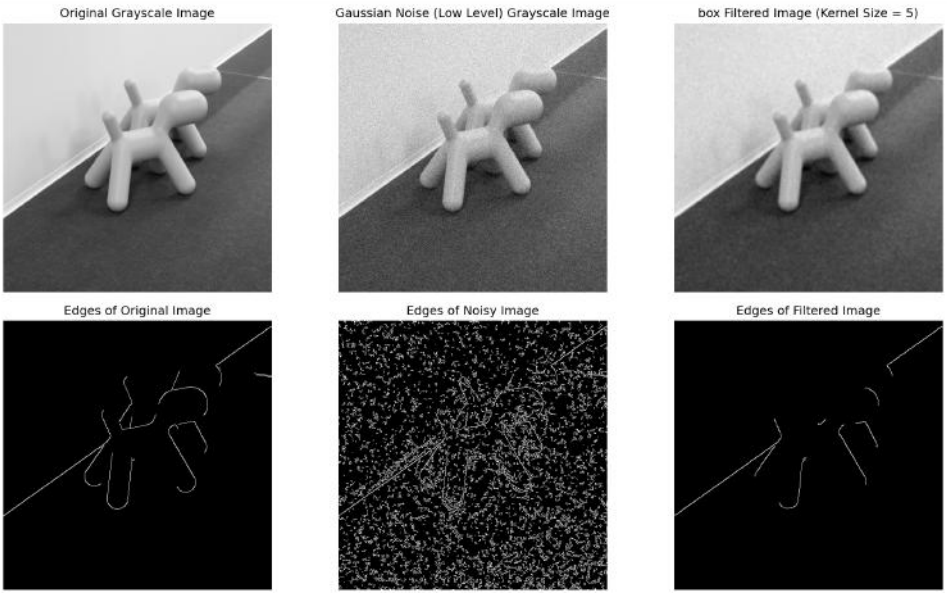


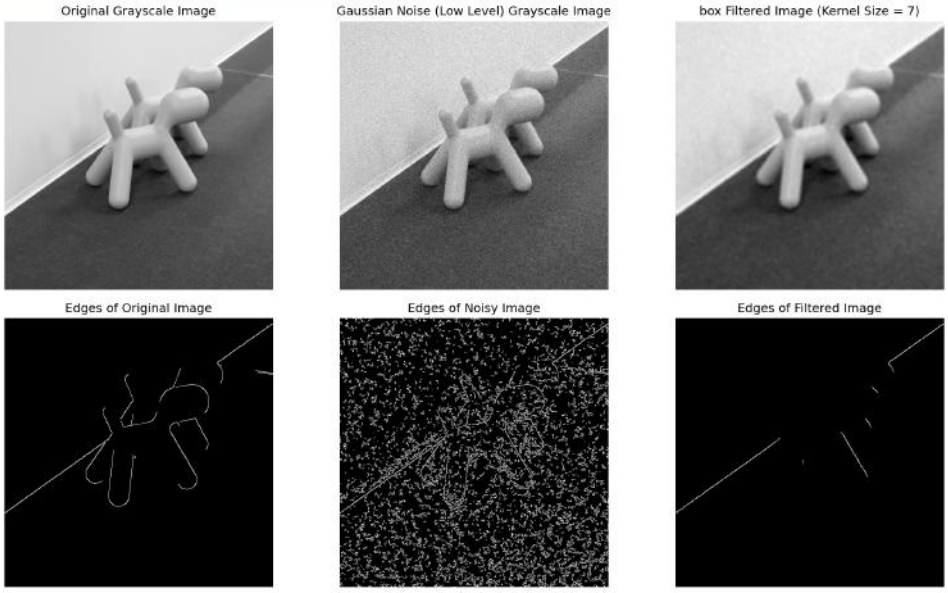
Figure 3: Performance Analysis of Box Filter Across Different Kernel Sizes



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 4: Applying Canny edge detector for the original, noisy, and filtered images with varying kernel sizes.

3.1.2 Second Image - Gaussian Noise - Medium Level

Edge Density

- **Original Image**: 0.1245
- **Noisy Image**: 0.3593

Table 4: Box Filter Results - Gaussian Noise (Medium Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	72.0904	29.5520	0.0162	0.1124	0.1347	0.5470
5	62.6016	30.1649	0.0010	0.0254	0.1347	0.4723
7	60.7431	30.2958	0.0029	0.0069	0.1347	0.4682

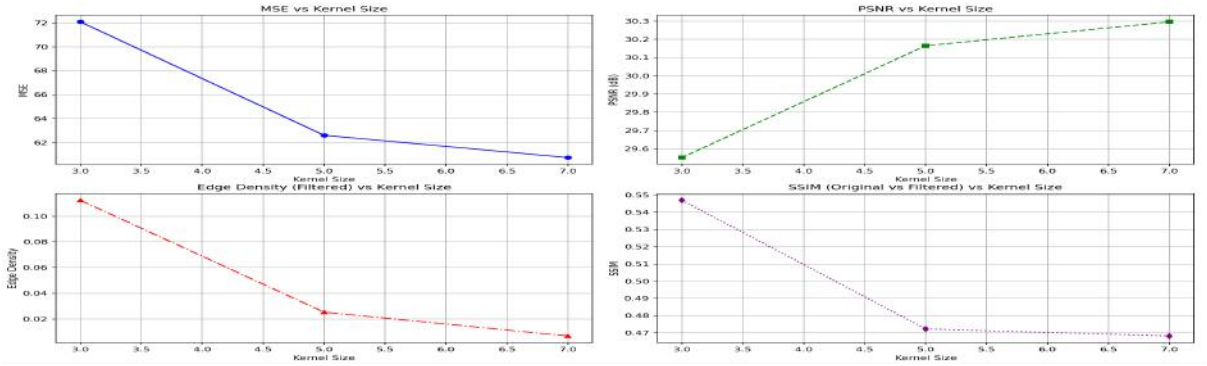
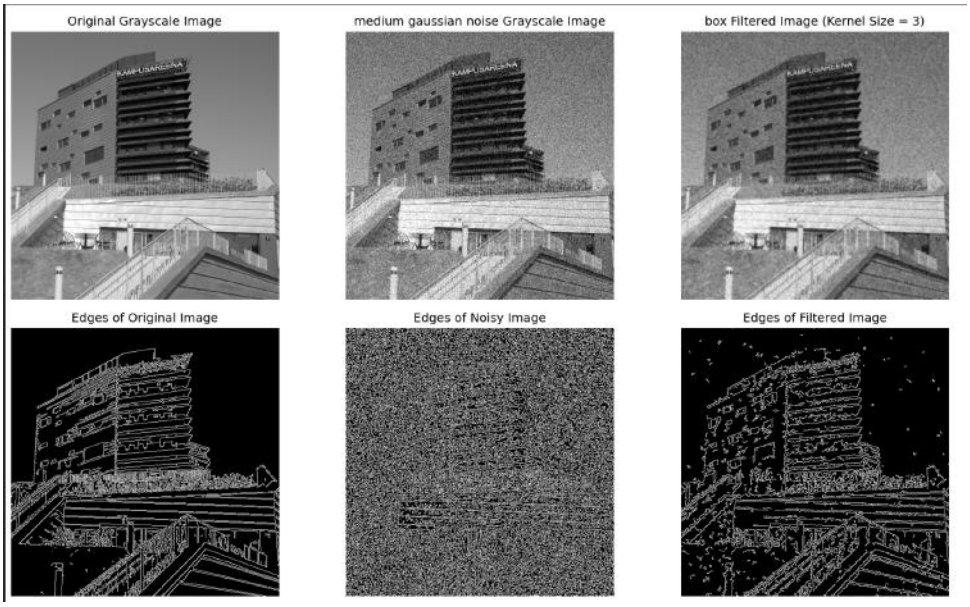
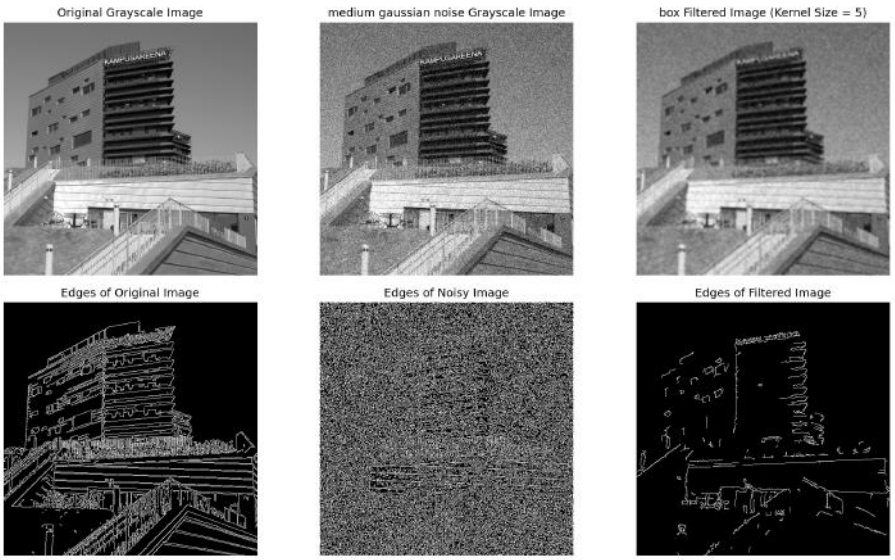


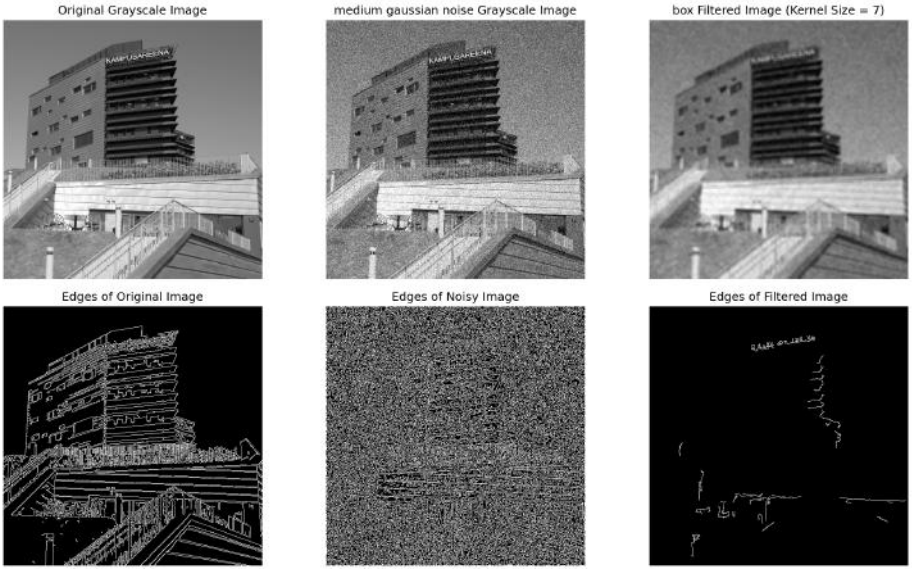
Figure 5: Performance Analysis of Box Filter Across Different Kernel Sizes 2



((a)) kernel 3



((b)) kernel 5



((c)) kernel 7

Figure 6: Canny edge detector applied across various configurations.

3.1.3 Third Image - Gaussian Noise - High Level

Edge Density

- **Original Image**: 0.1320
- **Noisy Image**: 0.3726

Table 5: Box Filter Results - Gaussian Noise (High Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	89.3694	28.6189	0.0000	0.3064	0.0373	0.0599
5	79.8633	29.1073	0.0000	0.0436	0.0373	0.4652
7	74.6164	29.4025	0.0000	0.0090	0.0373	0.4652

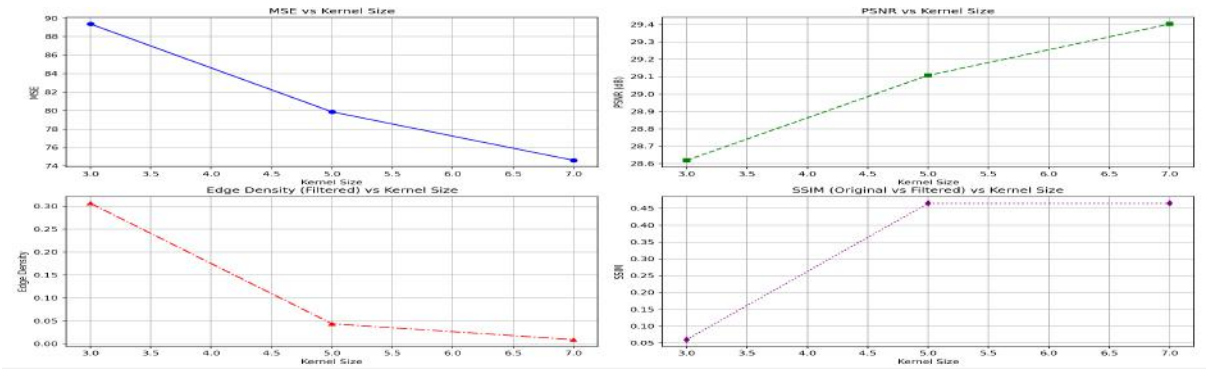
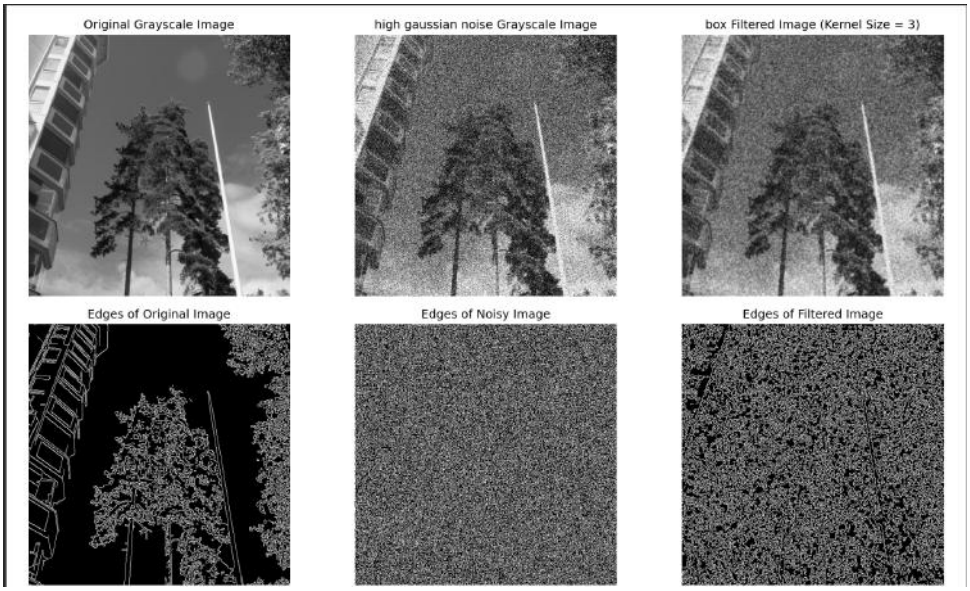
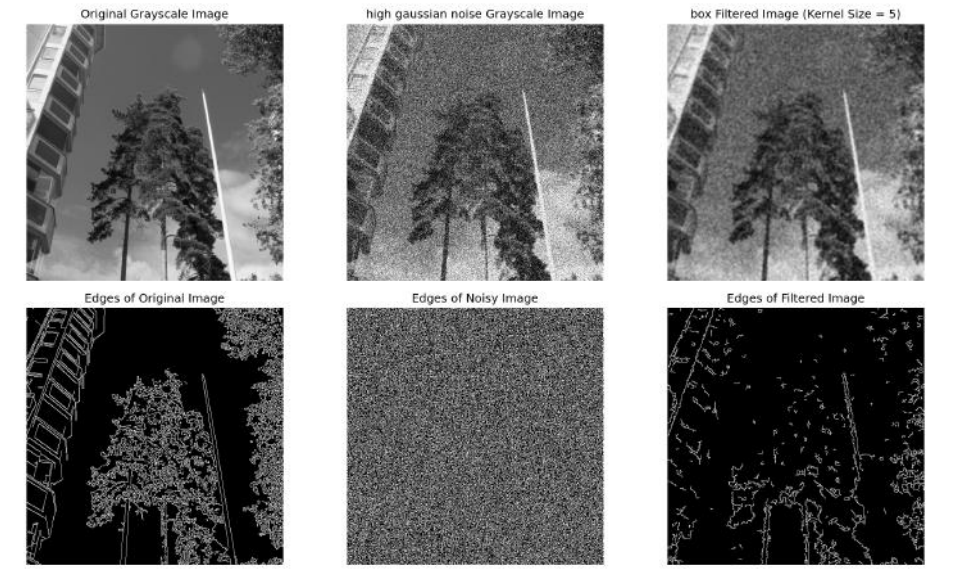


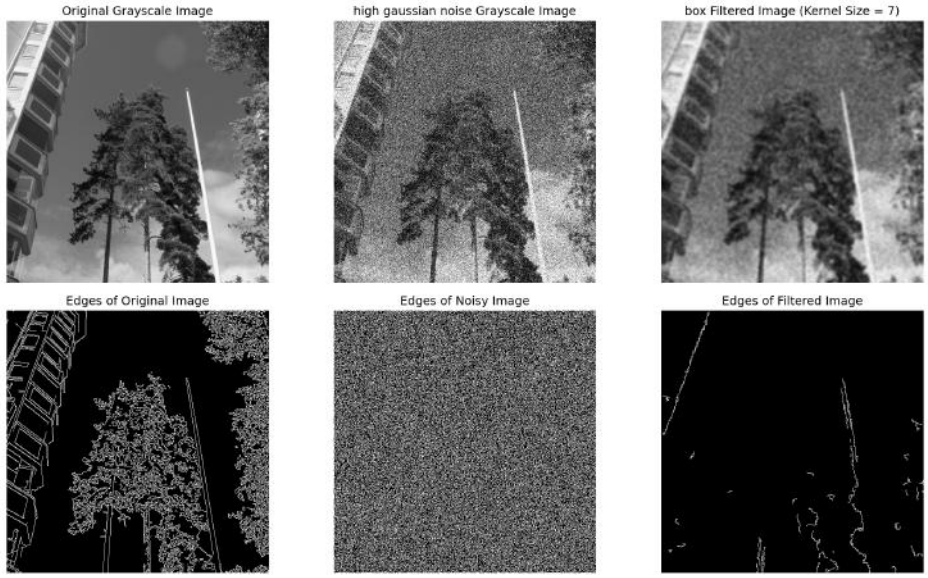
Figure 7: Performance Analysis of Box Filter Across Different Kernel Sizes 3



((a)) kernel 3



((b)) kernel 4



((c)) kernel 7

Figure 8: Application of Canny edge detector across various configurations.

3.1.4 Fourth Image - Salt-and-Pepper Noise - Low Level

Edge Density

- **Original Image**: 0.0907
- **Noisy Image**: 0.2342

Table 6: Box Filter Results - Salt-and-Pepper Noise (Low Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	45.6414	31.5372	0.0092	0.0883	0.2389	0.4562
5	58.7436	30.4412	0.0011	0.0053	0.2389	0.5699
7	59.6533	30.3745	0.0009	0.0059	0.2389	0.5699

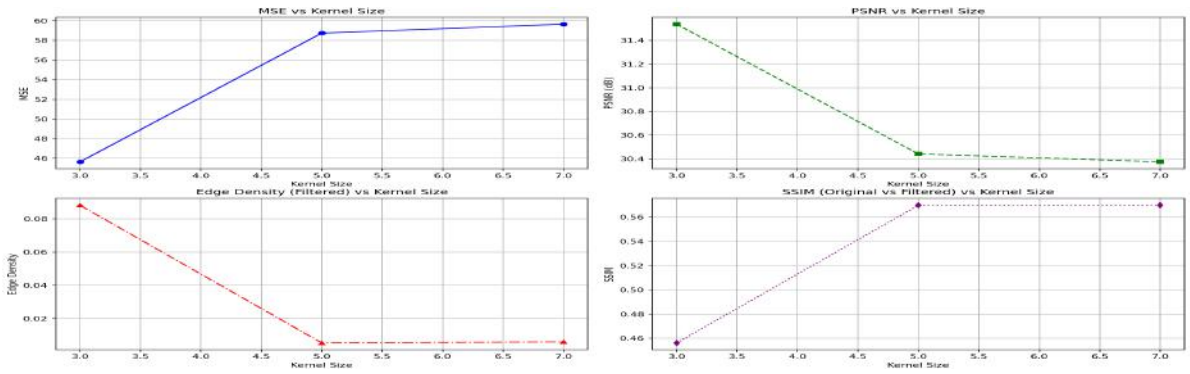
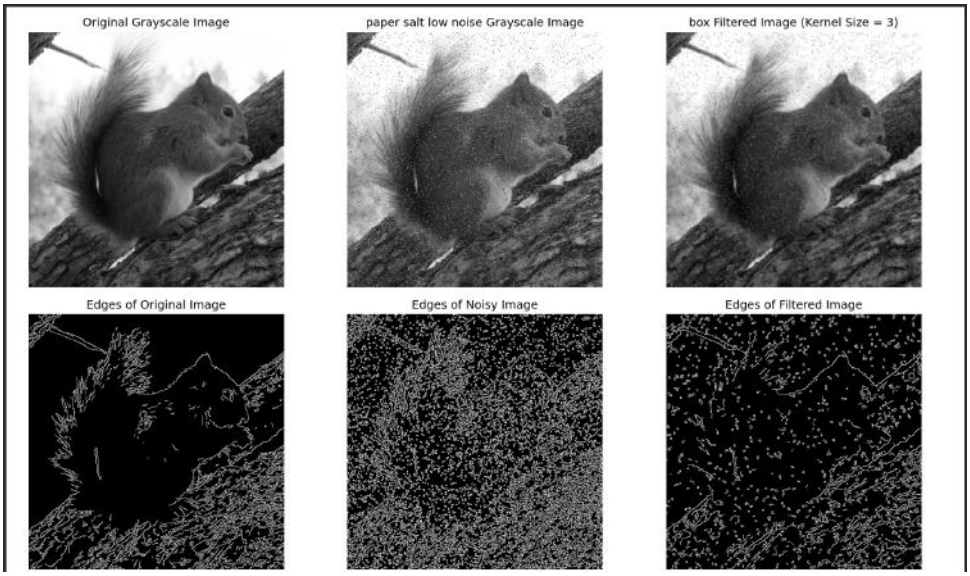
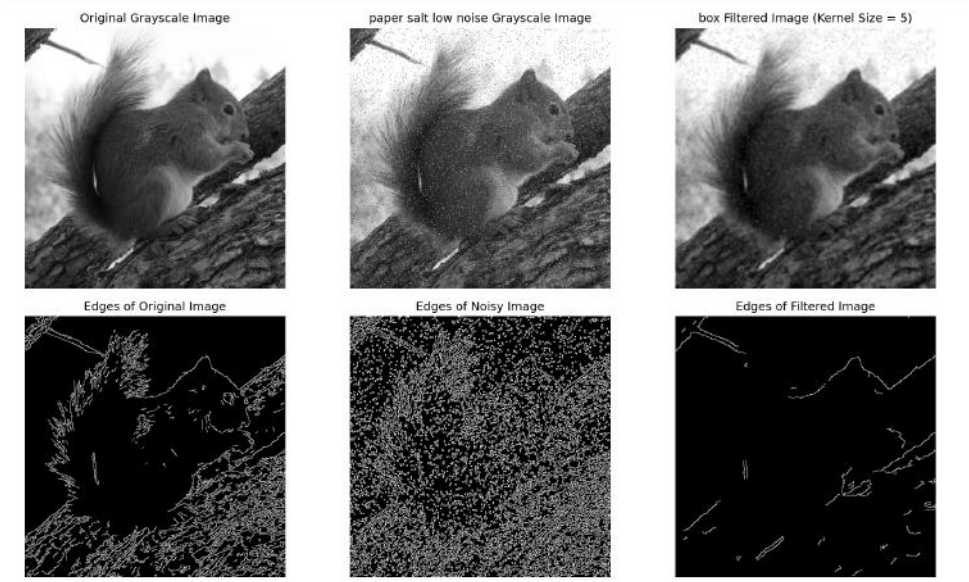


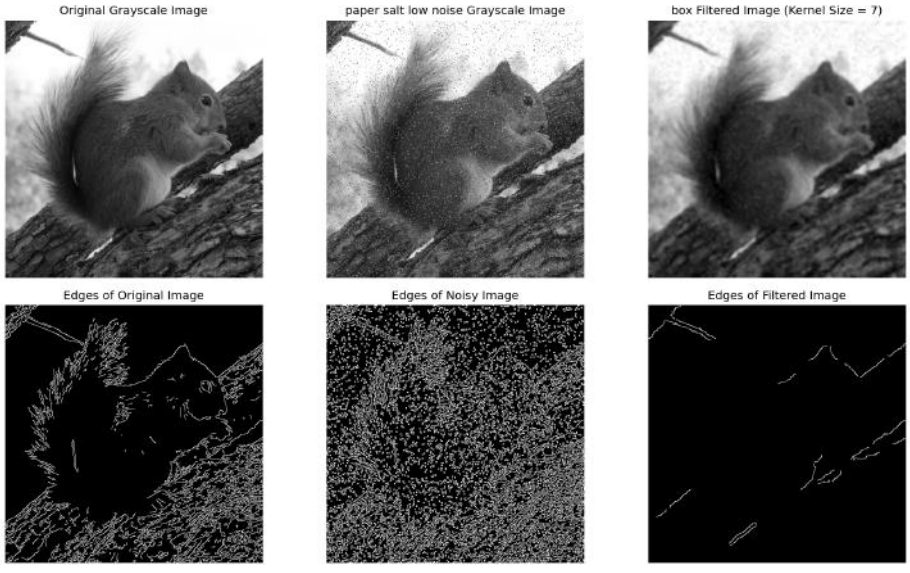
Figure 9: Performance Analysis of Box Filter Across Different Kernel Sizes 4



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 10: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

3.1.5 Fifth Image - Salt-and-Pepper Noise - Medium Level

Edge Density

- **Original Image**: 0.04199
- **Noisy Image**: 0.29196

Table 7: Box Filter Results - Salt-and-Pepper Noise (Medium Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	54.8809	30.7366	0.0010	0.1342	0.0725	0.3234
5	57.1850	30.5579	0.0000	0.0134	0.0725	0.7571
7	56.2773	30.6275	0.0010	0.0015	0.0725	0.7621

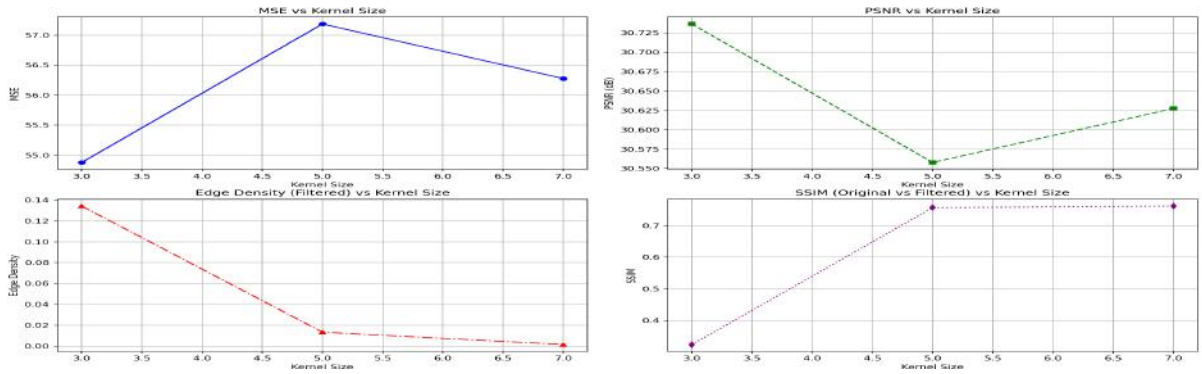
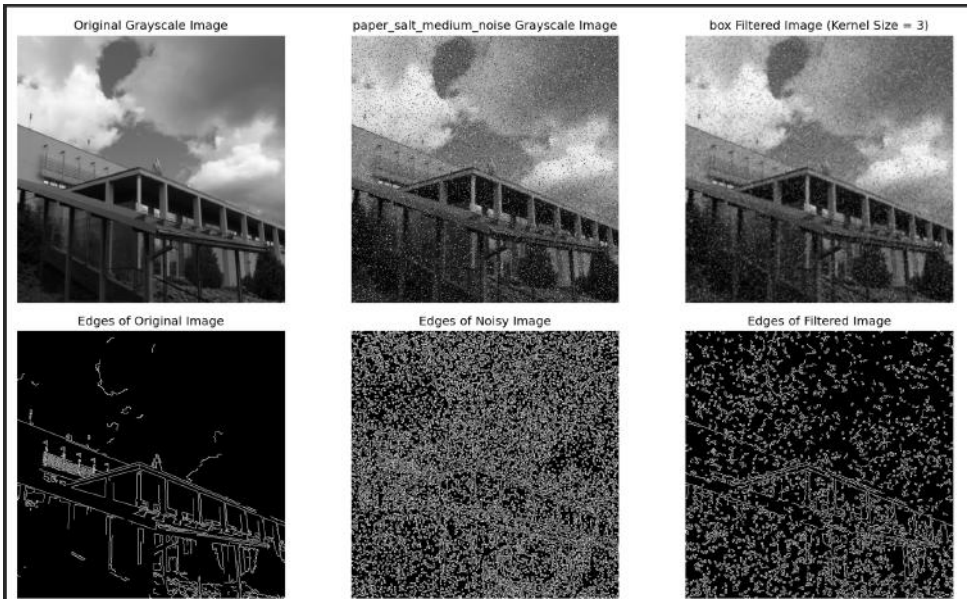
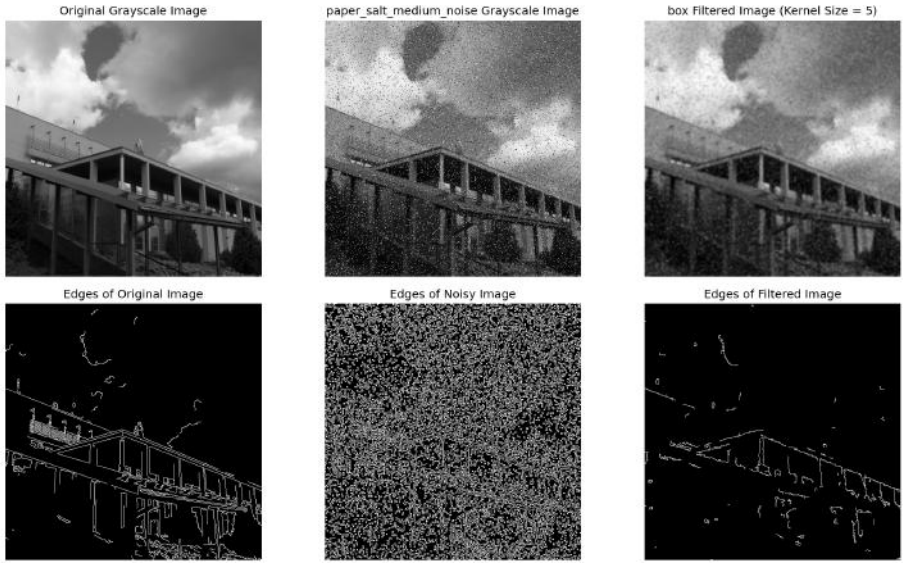


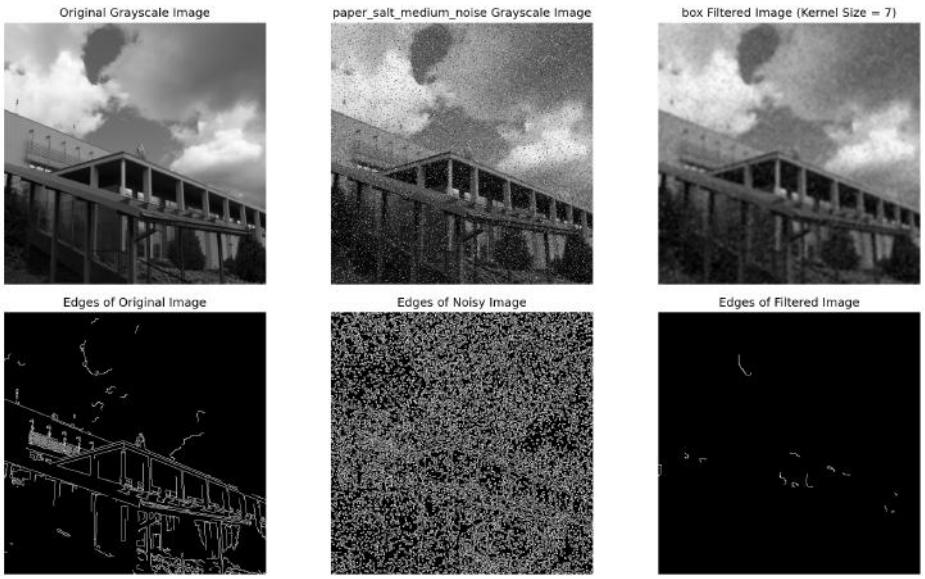
Figure 11: Performance Analysis of Box Filter Across Different Kernel Sizes 5



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 12: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

3.1.6 Sixth Image - Salt-and-Pepper Noise - High Level

Edge Density

- **Original Image**: 0.2108
- **Noisy Image**: 0.3739

Table 8: Box Filter Results - Salt-and-Pepper Noise (High Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	85.9307	28.7893	0.0000	0.3052	0.1175	0.1219
5	91.6422	28.5099	0.0000	0.1041	0.1175	0.1391
7	95.2886	28.3404	0.0000	0.0286	0.1175	0.1336

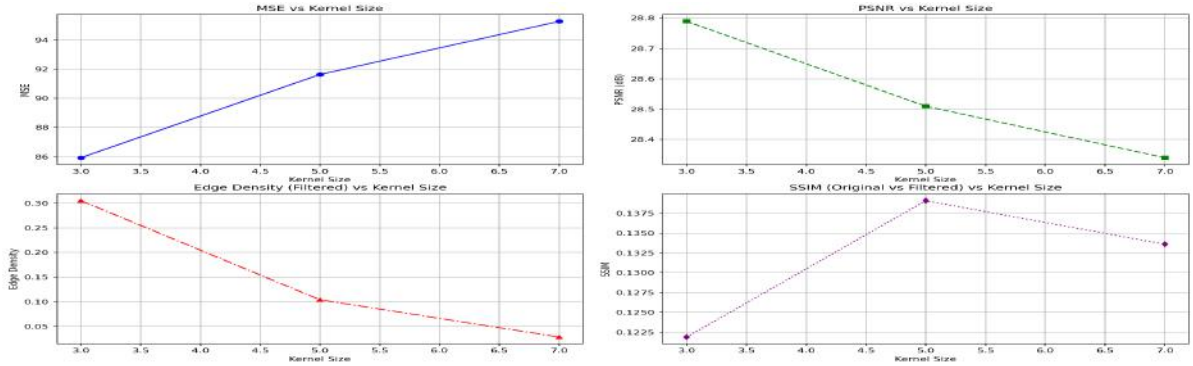
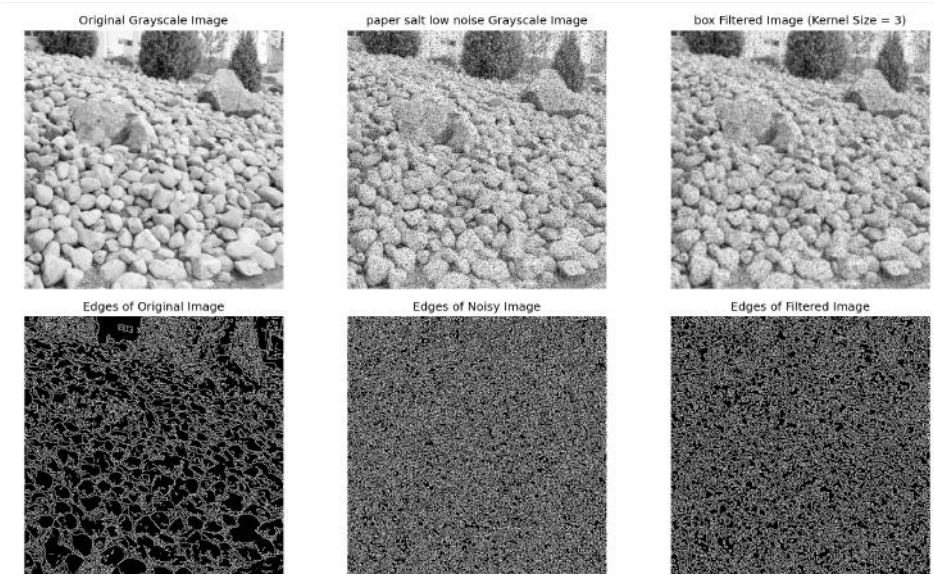
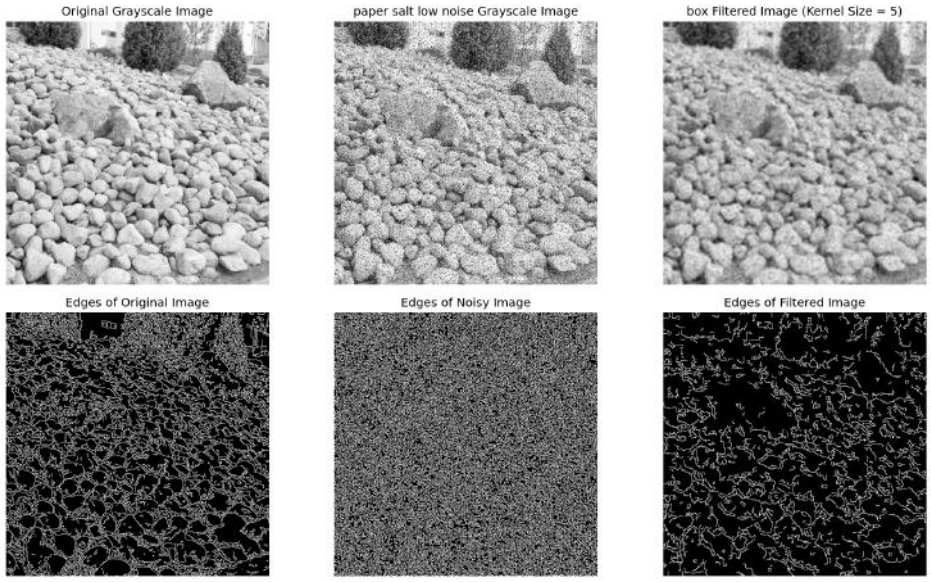


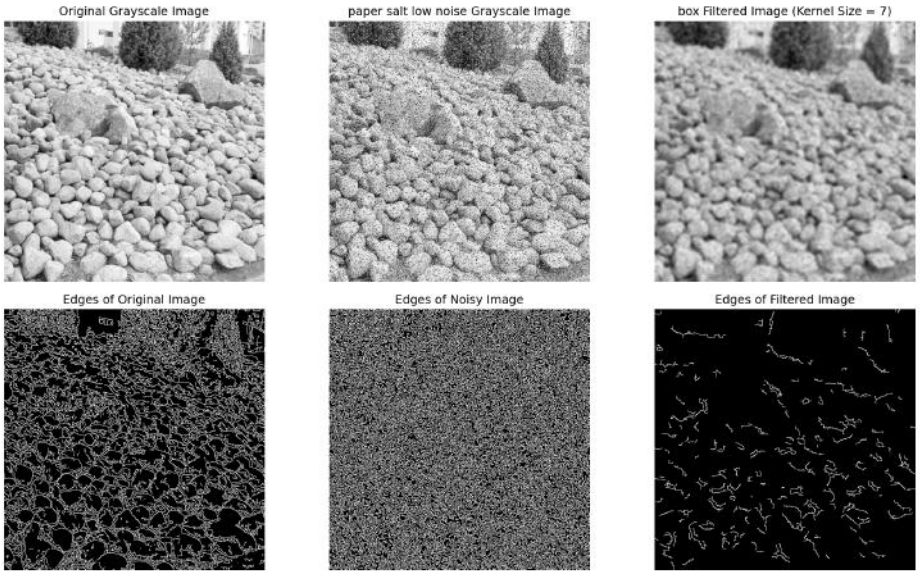
Figure 13: Performance Analysis of Box Filter Across Different Kernel Sizes 6



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 14: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

3.1.7 Summary of Results

Table 9: Average Performance Metrics for Box Filter Applications with Different Noise Types and Levels

Noise Type & Level	Kernel Size	Average MSE	Average PSNR (dB)	Average Filtering Time (s)	Average Edge Density - Filtered	Average SSIM (Original vs Noisy)	Average SSIM (Original vs Filtered)
Gaussian Noise - Low Level	3	30.6474	33.2669	0.0000	0.0064	0.2522	0.9664
	5	19.8156	35.1607	0.0000	0.0045	0.2522	0.9811
	7	17.8415	35.6165	0.0000	0.0027	0.2522	0.9594
Gaussian Noise - Medium Level	3	72.0904	29.5520	0.0162	0.1124	0.1347	0.5470
	5	62.6016	30.1649	0.0010	0.0254	0.1347	0.4723
	7	60.7431	30.2958	0.0029	0.0069	0.1347	0.4682
Gaussian Noise - High Level	3	89.3694	28.6189	0.0000	0.3064	0.0373	0.0599
	5	79.8633	29.1073	0.0000	0.0436	0.0373	0.4652
	7	74.6164	29.4025	0.0000	0.0090	0.0373	0.4652
Salt-and-Pepper Noise - Low Level	3	45.6414	31.5372	0.0092	0.0883	0.2389	0.4562
	5	58.7436	30.4412	0.0011	0.0053	0.2389	0.5699
	7	59.6533	30.3745	0.0009	0.0059	0.2389	0.5699
Salt-and-Pepper Noise - Medium Level	3	54.8809	30.7366	0.0010	0.1342	0.0725	0.3234
	5	57.1850	30.5579	0.0000	0.0134	0.0725	0.7571
	7	56.2773	30.6275	0.0010	0.0015	0.0725	0.7621
Salt-and-Pepper Noise - High Level	3	85.9307	28.7893	0.0000	0.3052	0.1175	0.1219
	5	91.6422	28.5099	0.0000	0.1041	0.1175	0.1391
	7	95.2886	28.3404	0.0000	0.0286	0.1175	0.1336

3.2 Gaussian-Filter

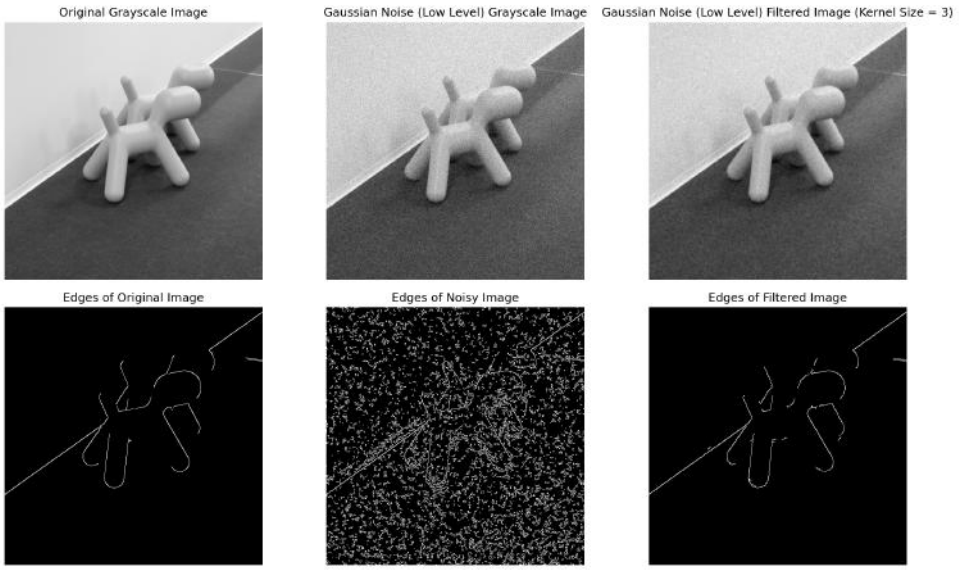
3.2.1 First Image - Gaussian Noise - Low Level

Edge Density:

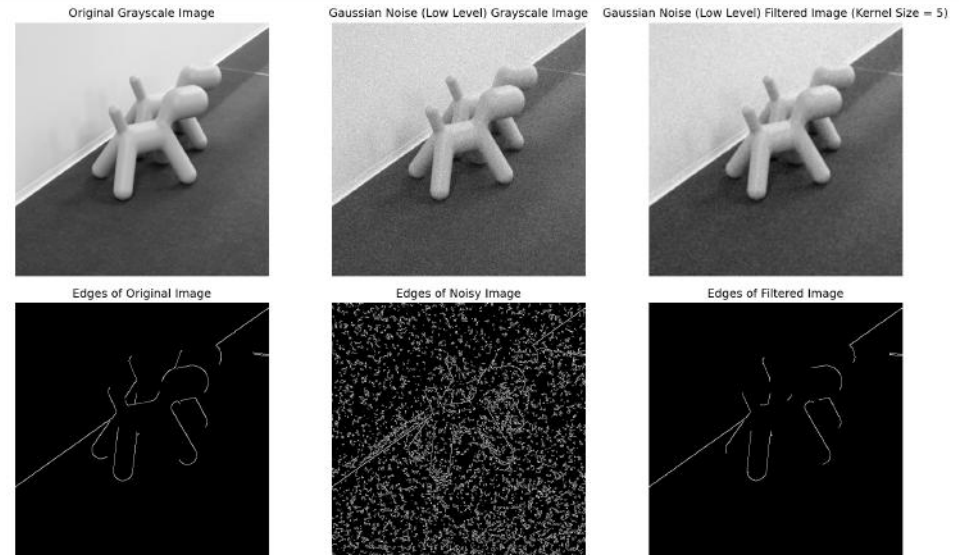
- **Original Image**: 0.0072
- **Noisy Image**: 0.1182

Table 10: Gaussian Filter for Gaussian Noise (Low Level)

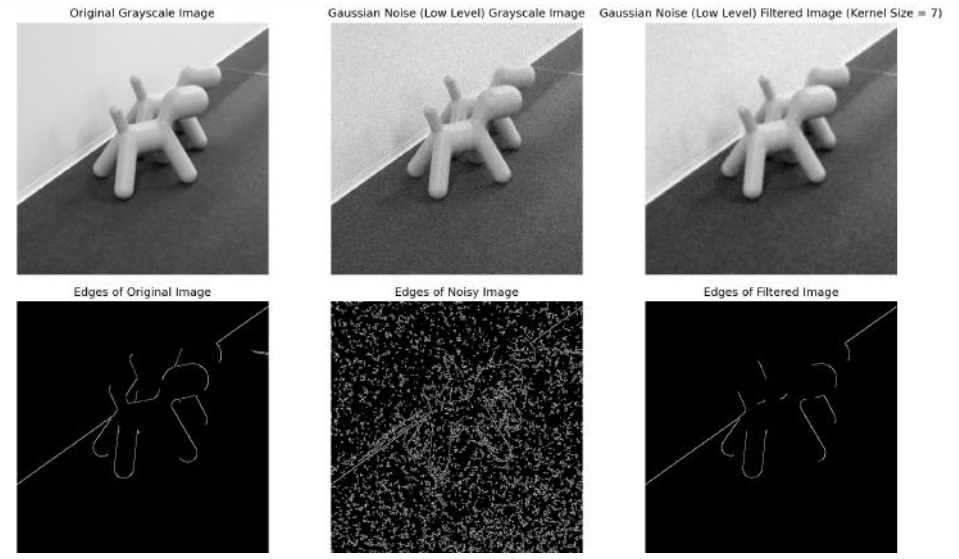
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	34.2016	32.7903	0.0042	0.0072	0.2522	0.9813
5	34.2016	32.7903	0.0042	0.0072	0.2522	0.9813
7	18.7286	35.4057	0.0040	0.0056	0.2522	0.9783



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 15: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

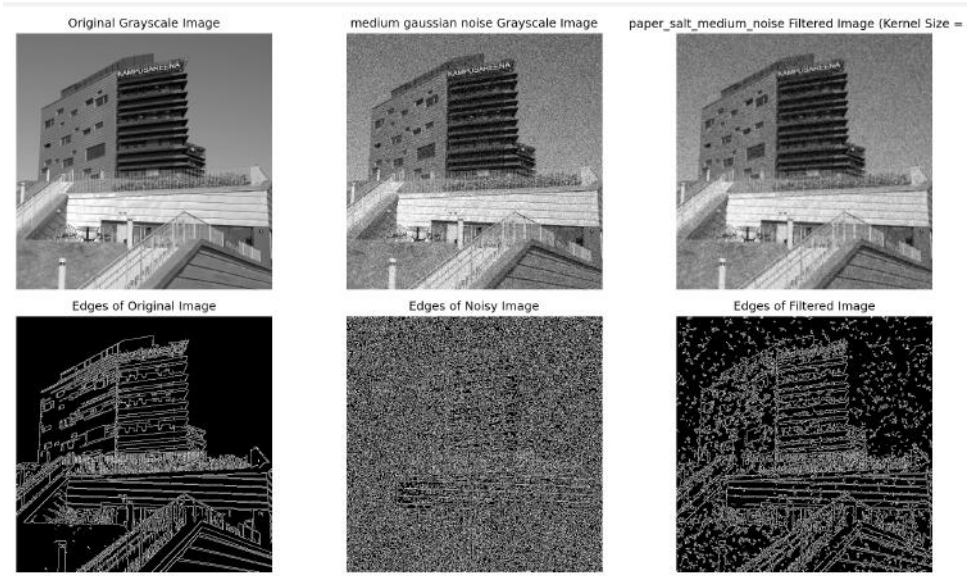
3.2.2 Second - Image - Gaussian Noise - Medium Level

Edge Density:

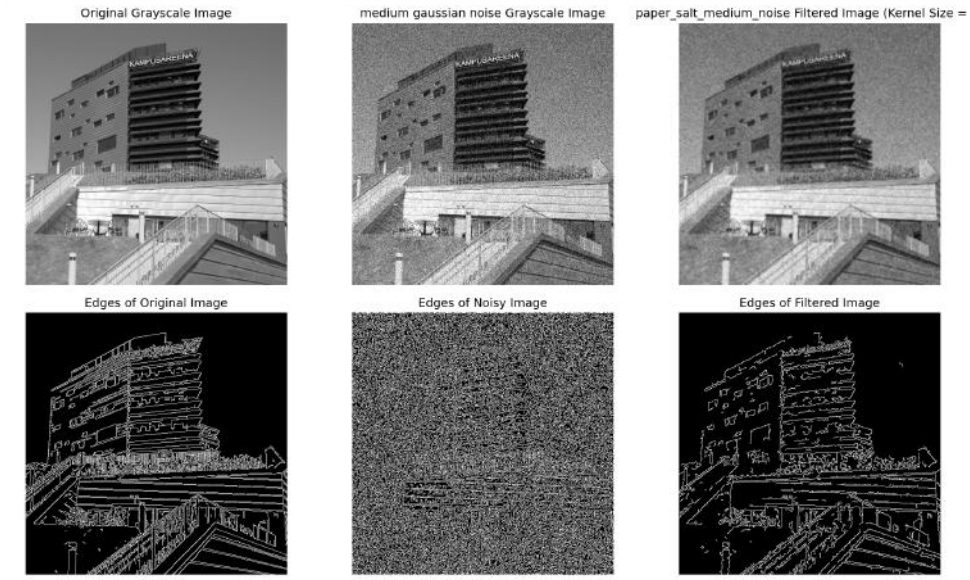
- **Original Image**: 0.1245
- **Noisy Image**: 0.3593

Table 11: Gaussian Filter for Gaussian Noise (Medium Level)

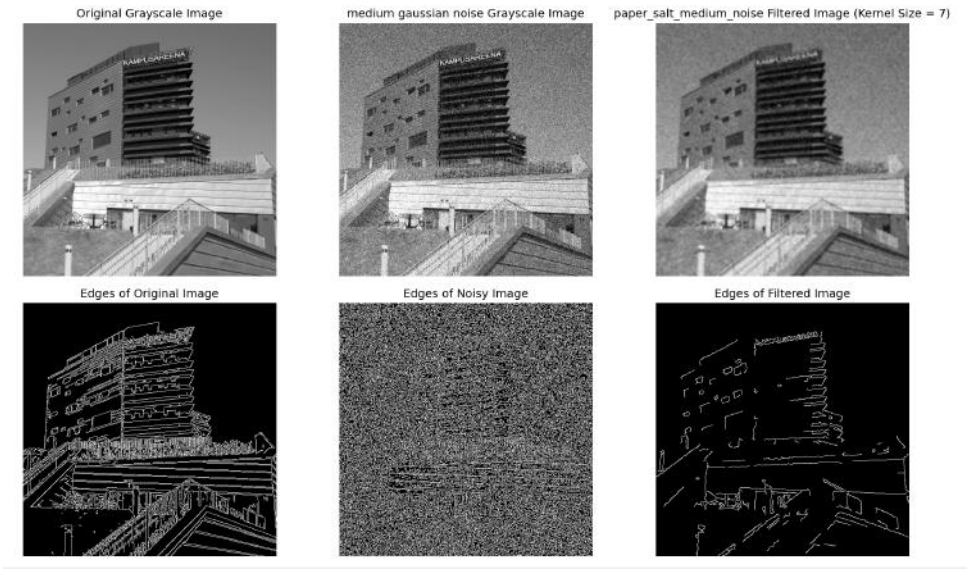
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	73.4635	29.4701	0.0155	0.1721	0.1347	0.3943
5	66.7580	29.8858	0.0079	0.0802	0.1347	0.6042
7	61.6406	30.2321	0.0041	0.0279	0.1347	0.5285



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 16: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

3.2.3 Third Image - Gaussian Filter - High Level Noise

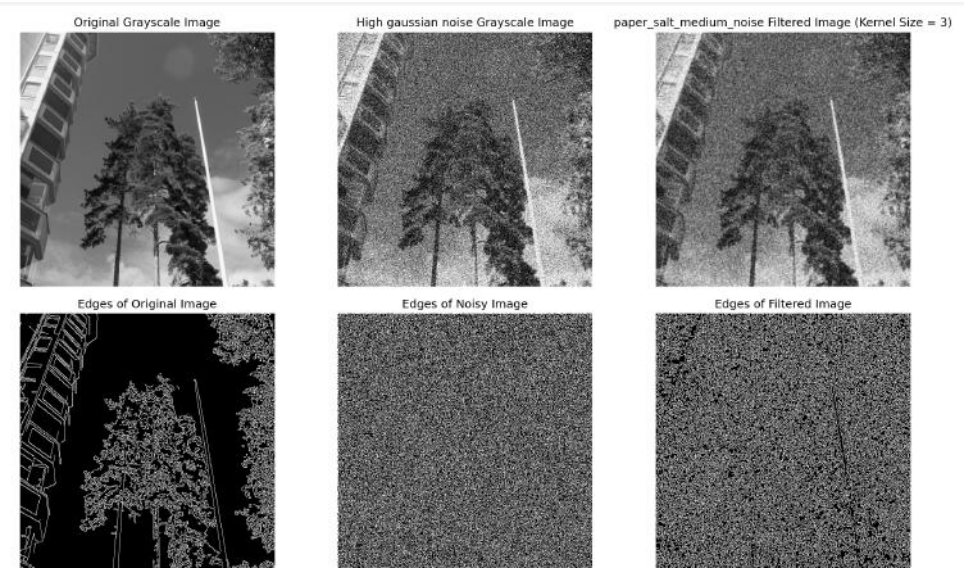
Edge Density:

- **Original Image**: 0.1320
- **Noisy Image**: 0.3726

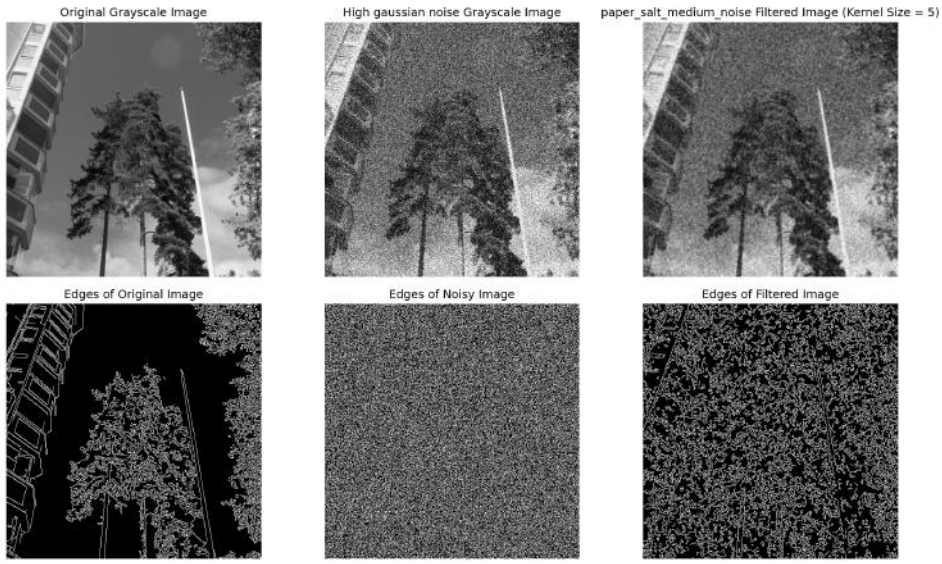
Table 12: Gaussian Filter Performance Metrics - High Level Noise

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	90.9879	28.5410	0.0000	0.3582	0.0373	0.0524
5	84.9952	28.8369	0.0000	0.2209	0.0373	0.1218
7	79.1100	29.1485	0.0011	0.0421	0.0373	0.4927

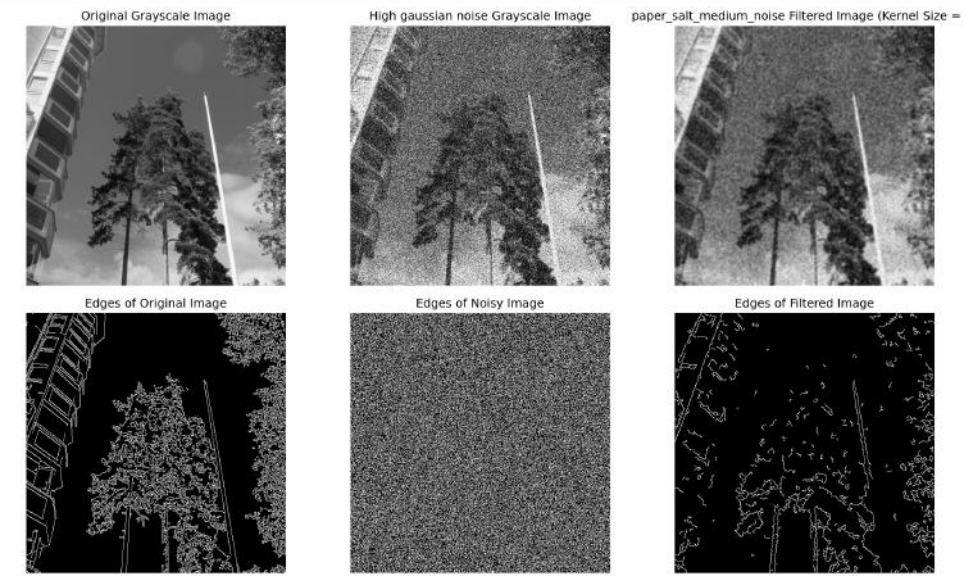
please note that the naming for the third column is wrong, it must be Gaussian filter instead of 'paper salt medium noise'



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 17: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

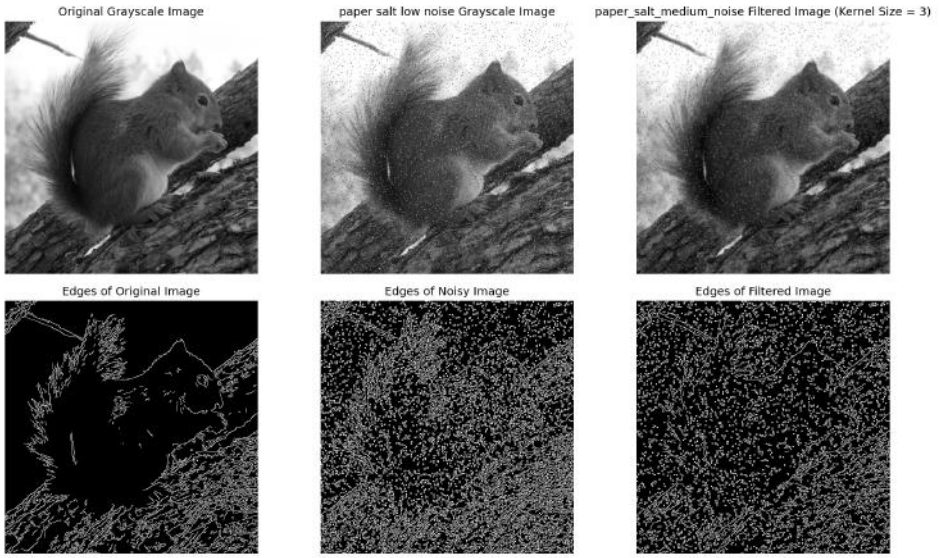
3.2.4 Fourth Image - Salt-and-Pepper Noise - Low Level

Edge Density:

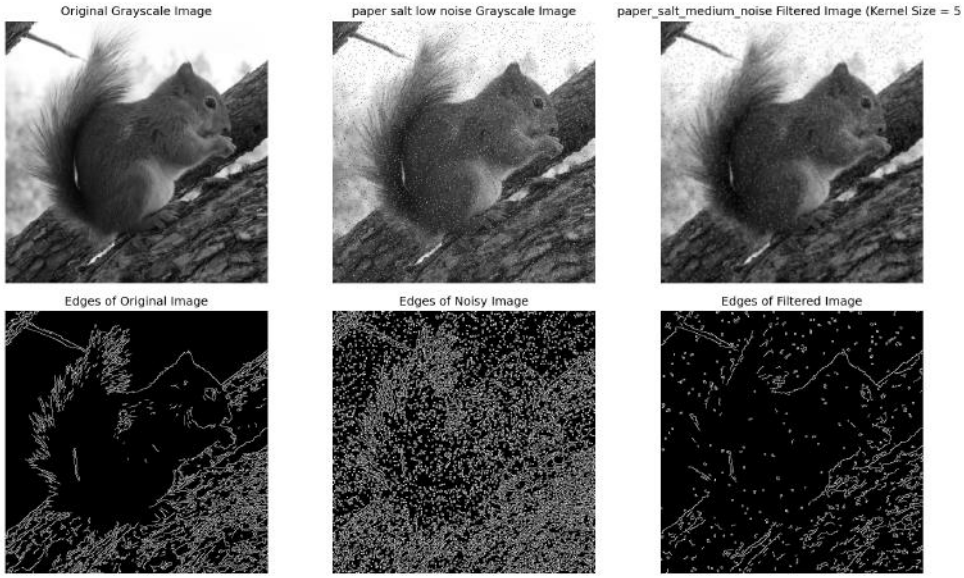
- **Original Image**: 0.0907
- **Noisy Image**: 0.2342

Table 13: Gaussian-Filter -Salt-and-Pepper Noise Filter Performance Metrics - Low Level

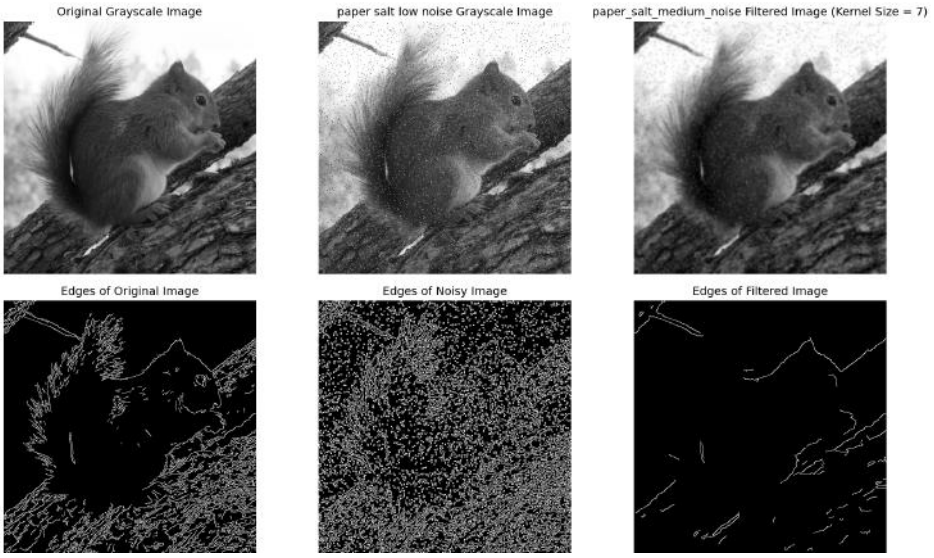
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	39.1752	32.2007	0.0106	0.1617	0.2389	0.2469
5	46.5186	31.4545	0.0029	0.0481	0.2389	0.5562
7	53.3710	30.8578	0.0040	0.0115	0.2389	0.5914



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 18: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

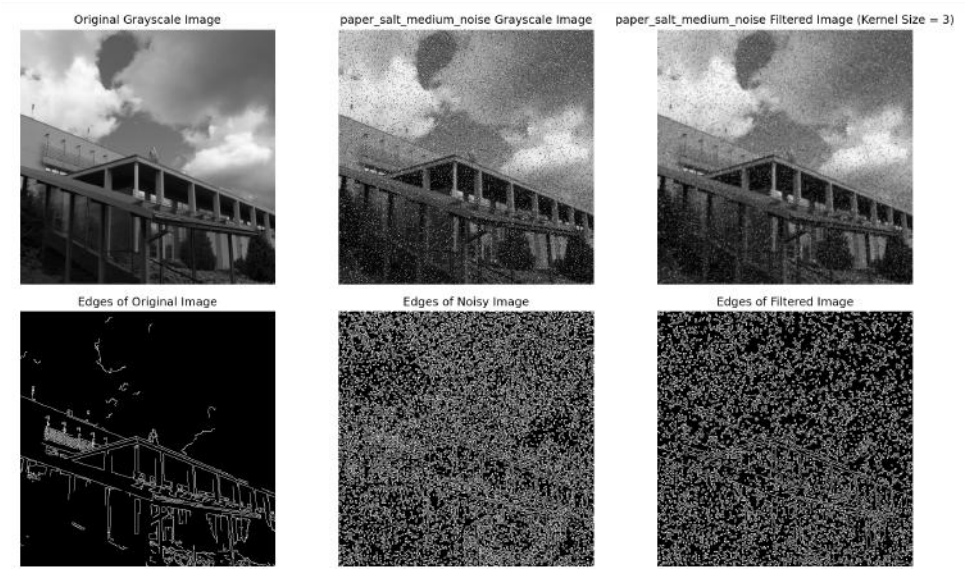
3.2.5 Fifth Image - Salt-and-Pepper Noise - Medium Level

Edge Density:

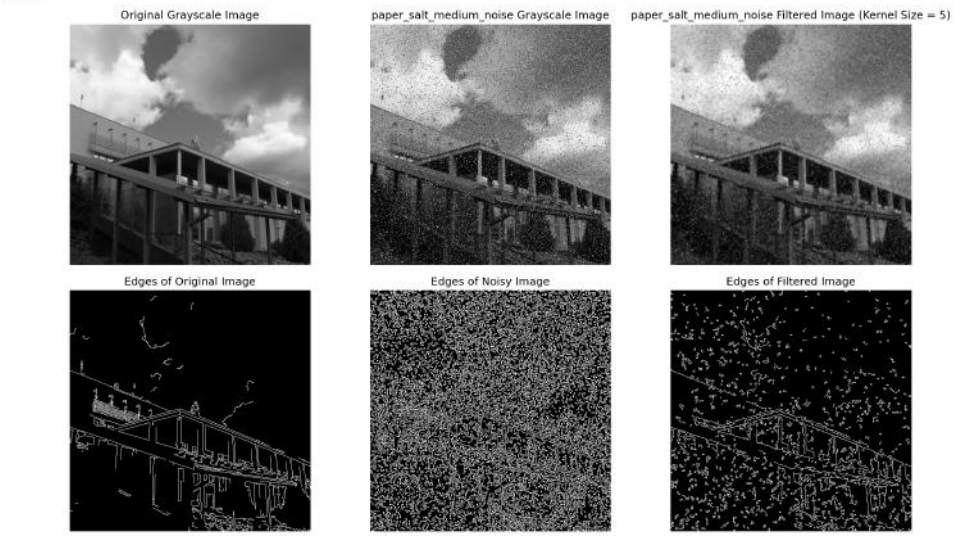
- **Original Image**: 0.04199
- **Noisy Image**: 0.29196

Table 14: Gaussian-Filter -Salt-and-Pepper Noise Filter Performance Metrics - Medium Level

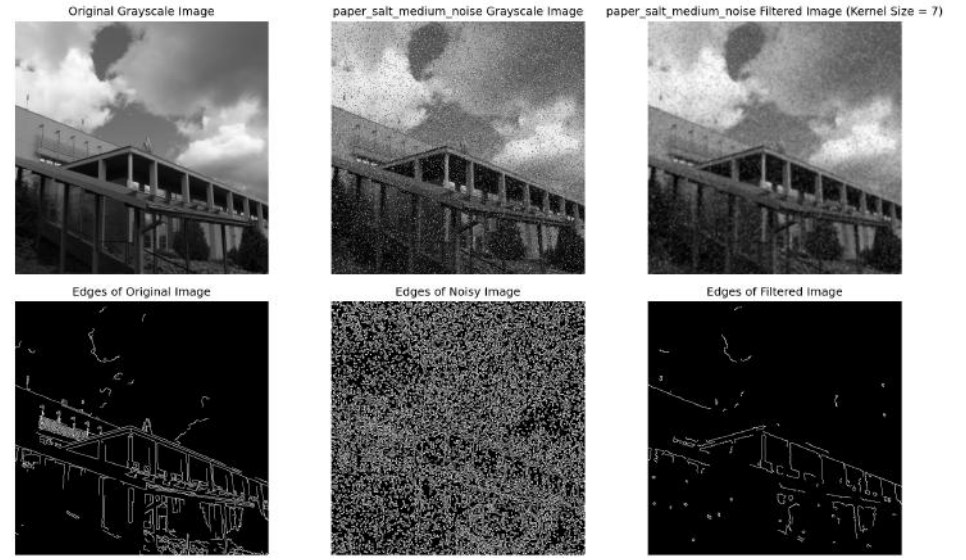
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	50.2201	31.1220	0.0020	0.2192	0.0725	0.1168
5	53.5030	30.8470	0.0021	0.0754	0.0725	0.5537
7	54.9885	30.7281	0.0029	0.0143	0.0725	0.7854



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 19: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

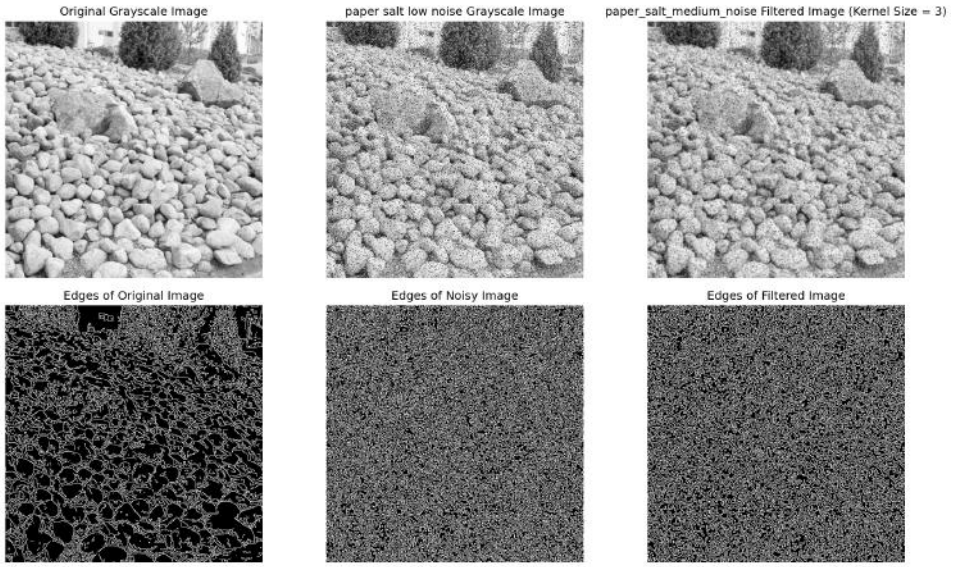
3.2.6 Sixth - Salt-and-Pepper Noise - High Level

Edge Density:

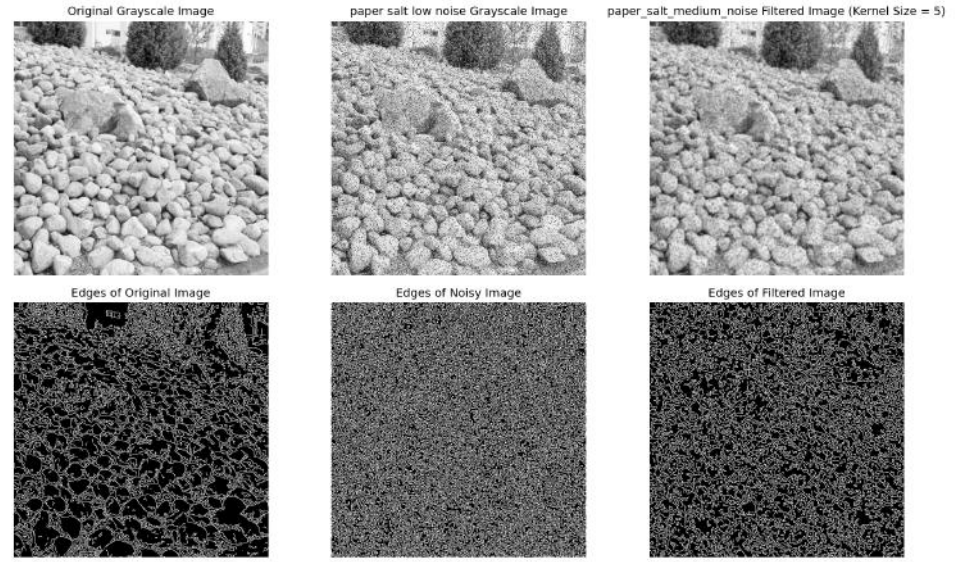
- **Original Image**: 0.2108
- **Noisy Image**: 0.3739

Table 15: Gaussian-Filter - Salt-and-Pepper Noise Filter Performance Metrics - High Level

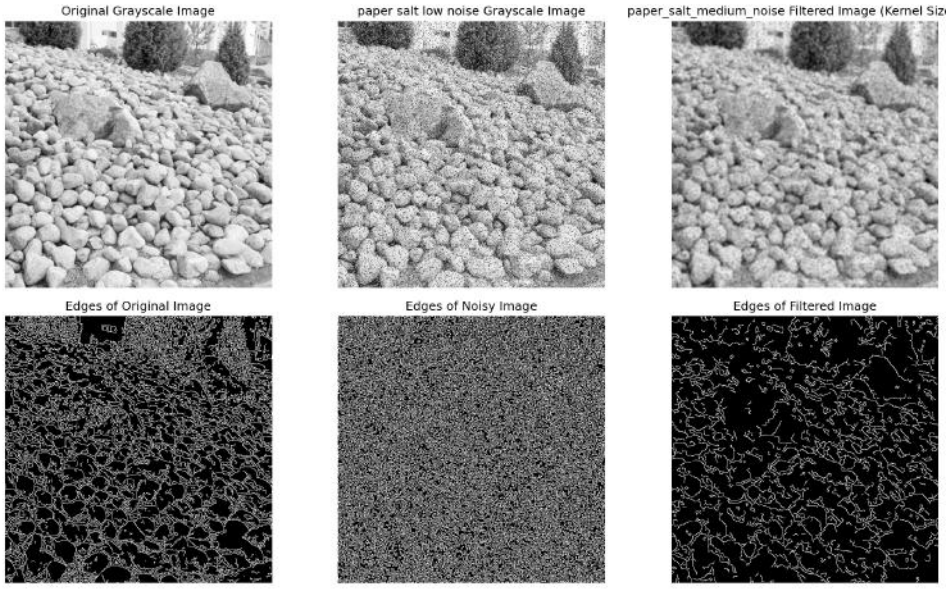
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	80.7747	29.0581	0.0000	0.3451	0.1175	0.1225
5	85.6049	28.8058	0.0000	0.2478	0.1175	0.1518
7	90.1118	28.5829	0.0000	0.0941	0.1175	0.2137



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 20: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

3.2.7 Summary of Results

Average Performance Metrics Across All Gaussian Filter Applications

Table 16: Average Performance Metrics for Gaussian and Salt-and-Pepper Noise at Various Levels

Noise Type & Level	Kernel Size	Average MSE	Average PSNR (dB)	Average Filtering Time (s)	Average Edge Density - Filtered	Average SSIM (Original vs Noisy)	Average SSIM (Original vs Filtered)
Gaussian Noise - Low Level	3	34.2016	32.7903	0.0042	0.0072	0.2522	0.9813
	5	34.2016	32.7903	0.0042	0.0072	0.2522	0.9813
	7	18.7286	35.4057	0.0040	0.0056	0.2522	0.9783
Gaussian Noise - Medium Level	3	73.4635	29.4701	0.0155	0.1721	0.1347	0.3943
	5	66.7580	29.8858	0.0079	0.0802	0.1347	0.6042
	7	61.6406	30.2321	0.0041	0.0279	0.1347	0.5285
Gaussian Noise - High Level	3	90.9879	28.5410	0.0000	0.3582	0.0373	0.0524
	5	84.9952	28.8369	0.0000	0.2209	0.0373	0.1218
	7	79.1100	29.1485	0.0011	0.0421	0.0373	0.4927
Salt-and-Pepper Noise - Low Level	3	39.1752	32.2007	0.0106	0.1617	0.2389	0.2469
	5	46.5186	31.4545	0.0029	0.0481	0.2389	0.5562
	7	53.3710	30.8578	0.0040	0.0115	0.2389	0.5914
Salt-and-Pepper Noise - Medium Level	3	50.2201	31.1220	0.0020	0.2192	0.0725	0.1168
	5	53.5030	30.8470	0.0021	0.0754	0.0725	0.5537
	7	54.9885	30.7281	0.0029	0.0143	0.0725	0.7854
Salt-and-Pepper Noise - High Level	3	80.7747	29.0581	0.0000	0.3451	0.1175	0.1225
	5	85.6049	28.8058	0.0000	0.2478	0.1175	0.1518
	7	90.1118	28.5829	0.0000	0.0941	0.1175	0.2137

3.3 Median-Filter

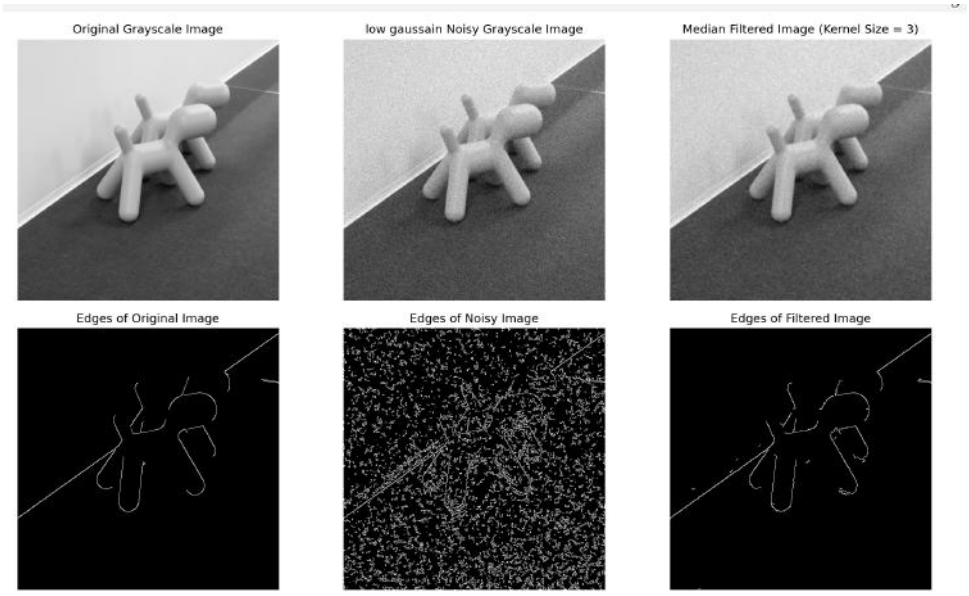
3.3.1 First Image - Gaussian noise - low level

Edge Density:

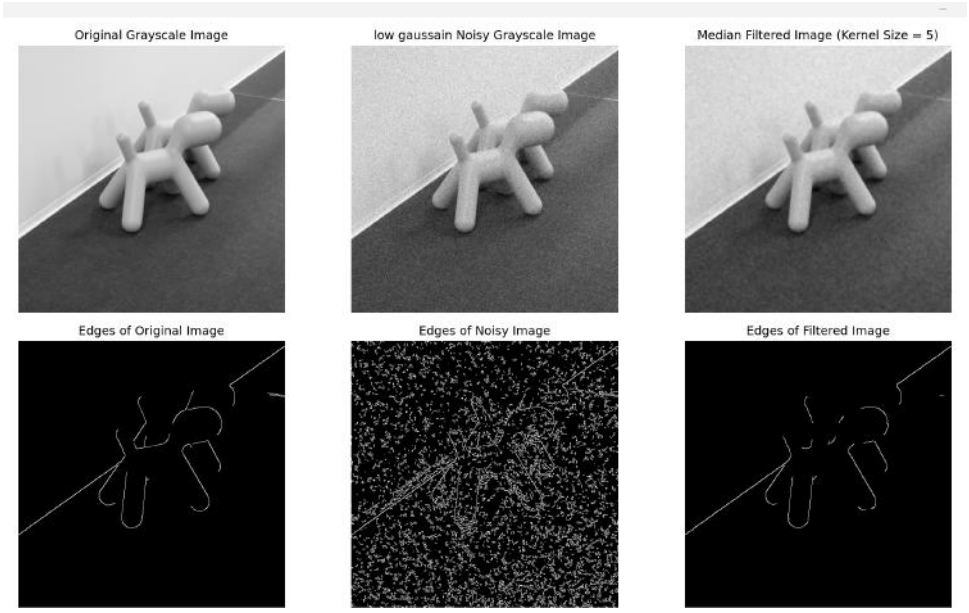
- **Original Image**: 0.0072
- **Noisy Image**: 0.1182

Table 17: Median Filter Performance Metrics - Gaussian Noise (Low Level)

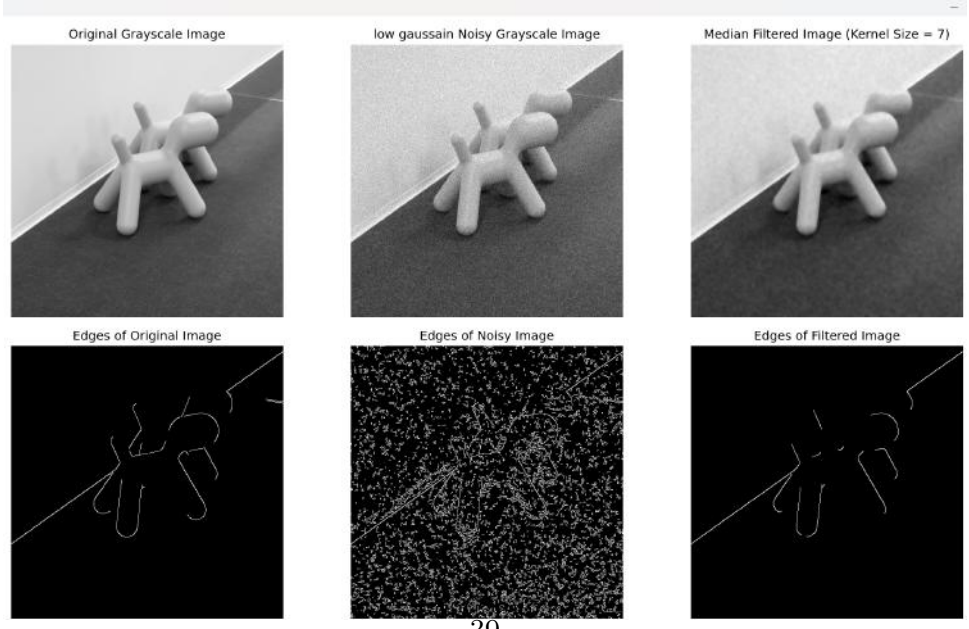
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	39.3593	32.1803	0.0000	0.0078	0.2522	0.9762
5	23.6599	34.3907	0.0040	0.0060	0.2522	0.9784
7	19.1871	35.3007	0.0042	0.0054	0.2522	0.9768



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 21: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

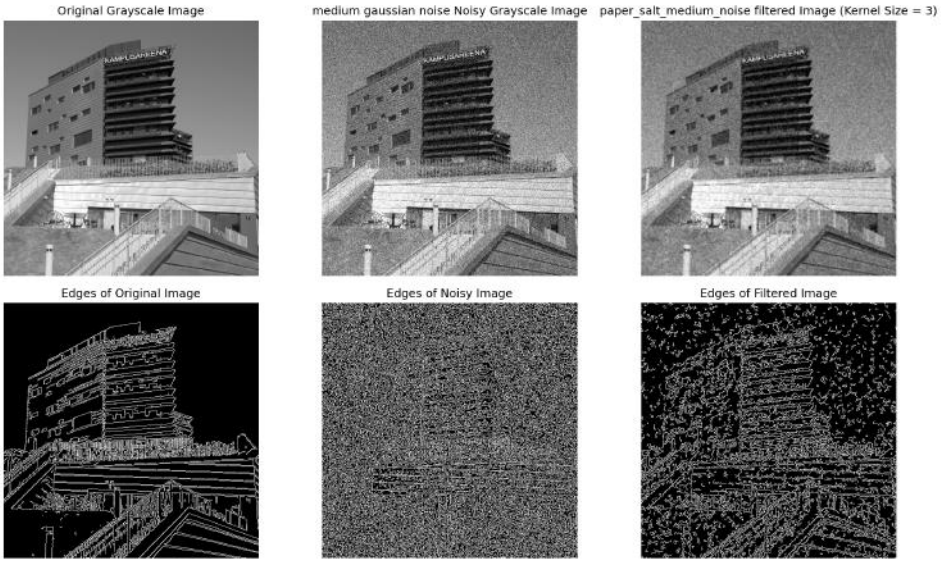
3.3.2 Second Image - Gaussian Noise - Medium Level

Edge Density:

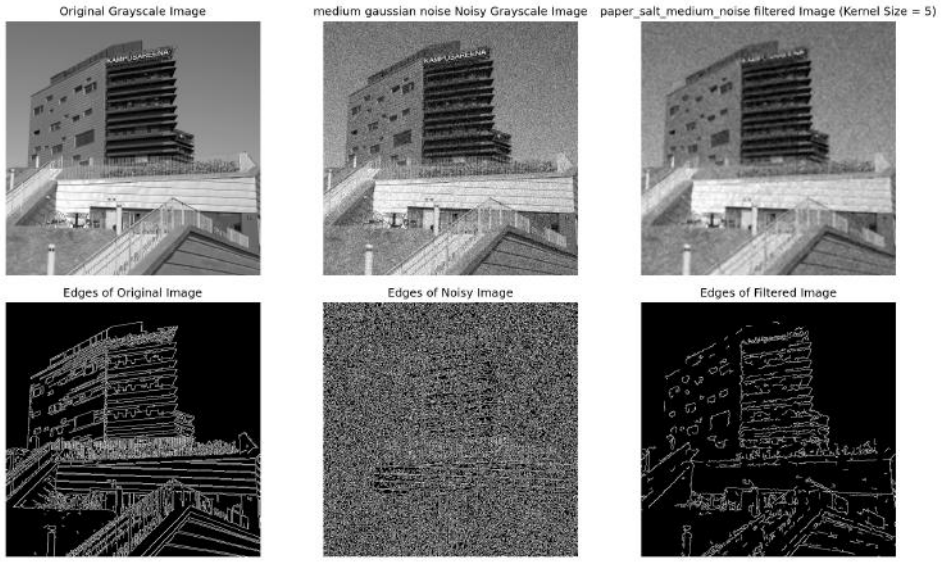
- **Original Image**: 0.1245
- **Noisy Image**: 0.3593

Table 18: Median Filter Performance Metrics - Gaussian Noise (Medium Level)

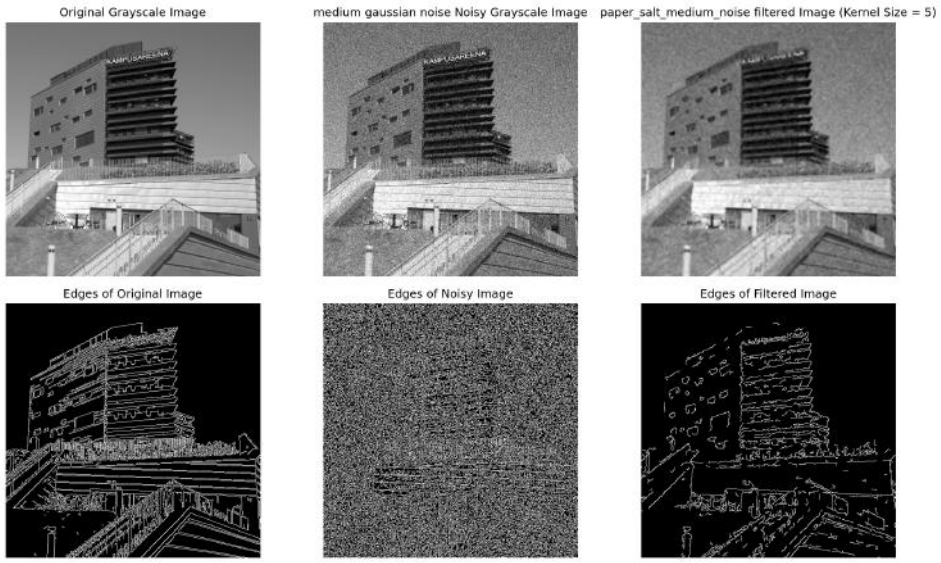
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	76.7794	29.2784	0.0085	0.1727	0.1347	0.2791
5	65.7728	29.9503	0.0032	0.0540	0.1347	0.5102
7	60.9982	30.2776	0.0119	0.0187	0.1347	0.4872



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 22: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

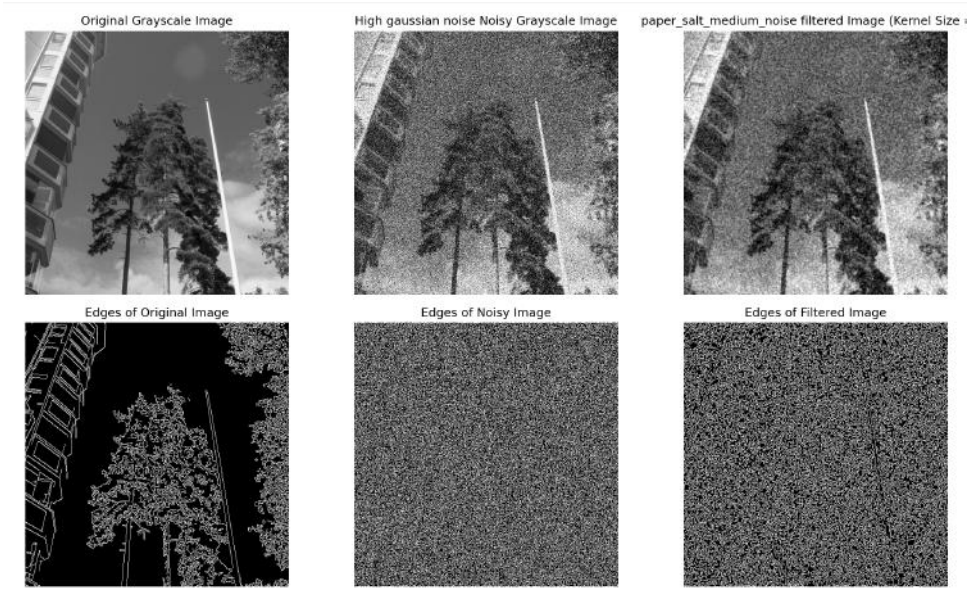
3.3.3 Third Image - Gaussian Noise - High Level

Edge Density:

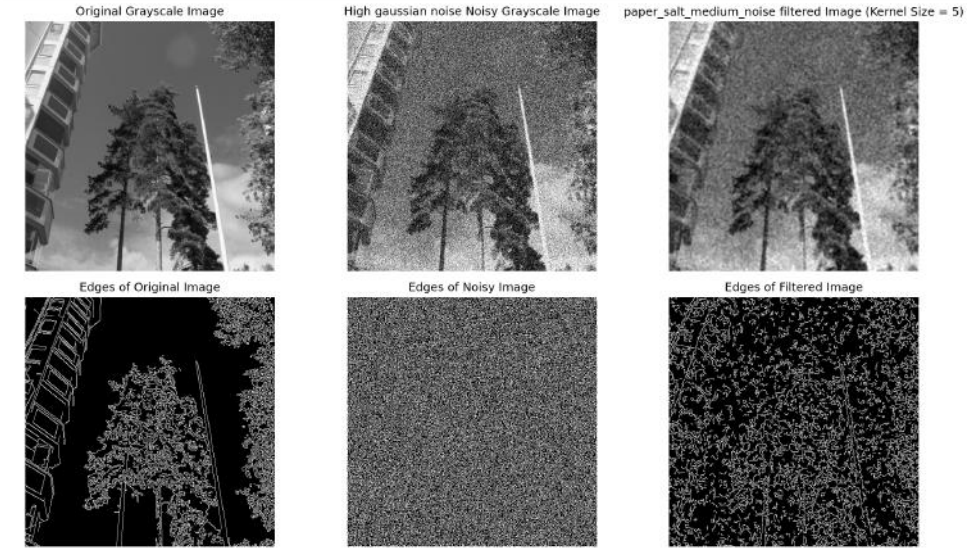
- **Original Image**: 0.1320
- **Noisy Image**: 0.3726

Table 19: Median Filter Performance Metrics - Gaussian Noise (High Level)

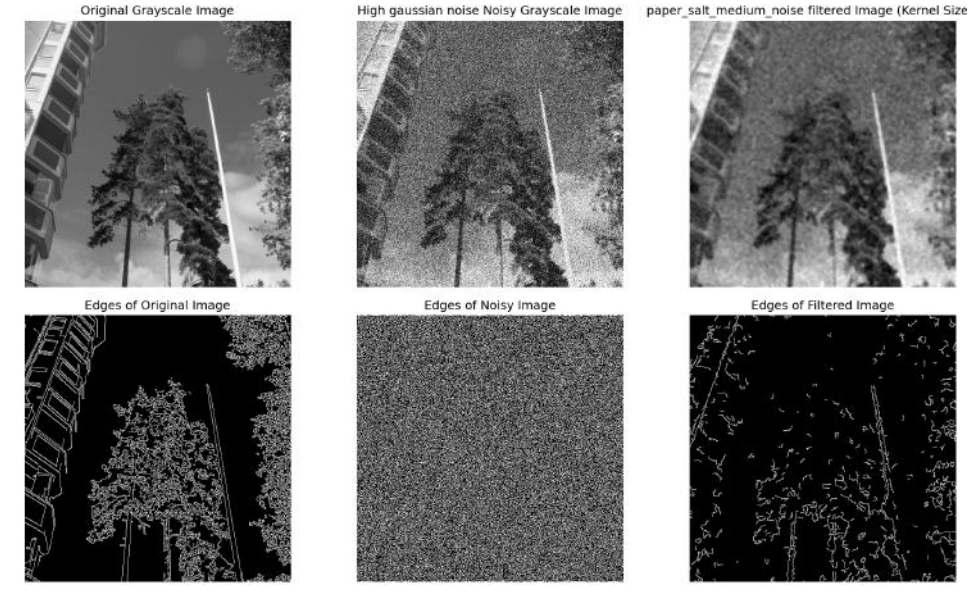
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	95.0296	28.3522	0.0000	0.3407	0.0373	0.0353
5	84.5429	28.8600	0.0000	0.1567	0.0373	0.1643
7	77.9243	29.2141	0.0035	0.0407	0.0373	0.4471



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 23: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

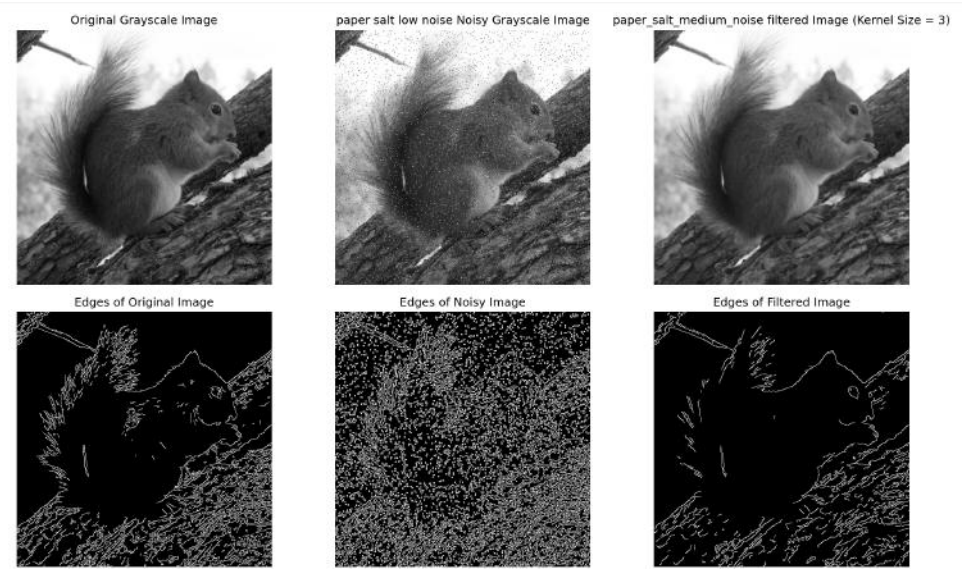
3.3.4 Fourth - Image - Salt-and-Pepper Noise - Low Level

Edge Density:

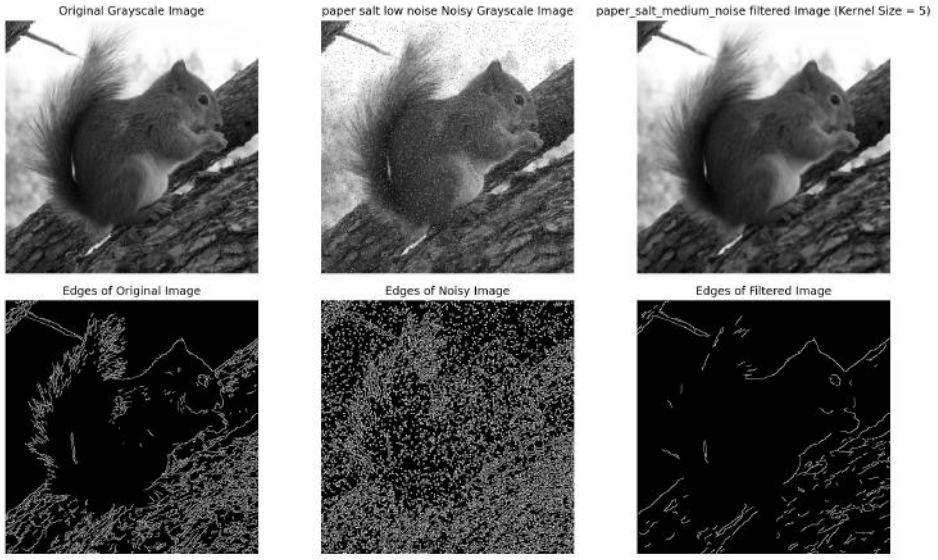
- **Original Image**: 0.0907
- **Noisy Image**: 0.2342

Table 20: Median Filter Performance Metrics - Salt-and-Pepper Noise (Low Level)

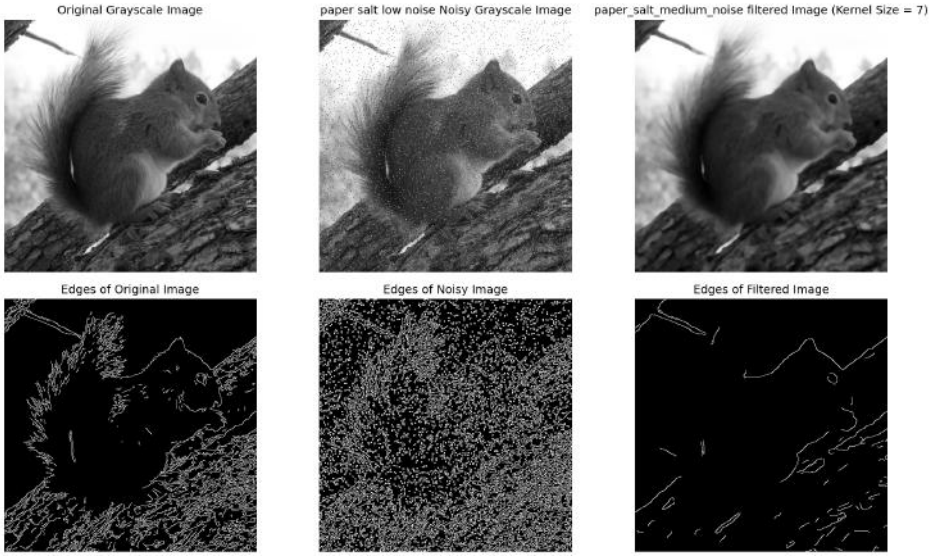
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	24.6156	34.2187	0.0354	0.0480	0.2389	0.7081
5	35.3798	32.6432	0.0010	0.0229	0.2389	0.6265
7	41.0205	32.0008	0.0060	0.0133	0.2389	0.5995



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 24: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

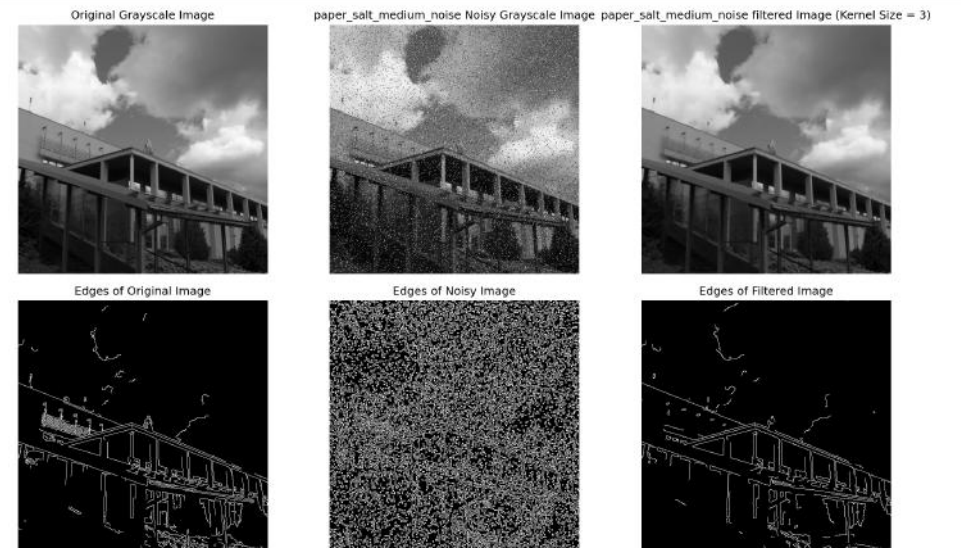
3.3.5 Fifth - Image - Salt-and-Pepper Noise - Medium Level

Edge Density:

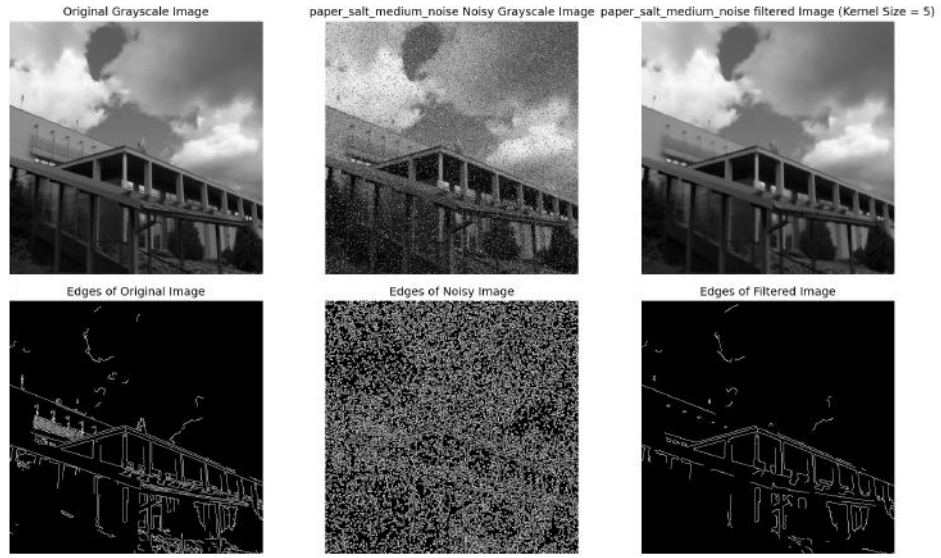
- **Original Image**: 0.04199
- **Noisy Image**: 0.29196

Table 21: Median Filter Performance Metrics - Salt-and-Pepper Noise (Medium Level)

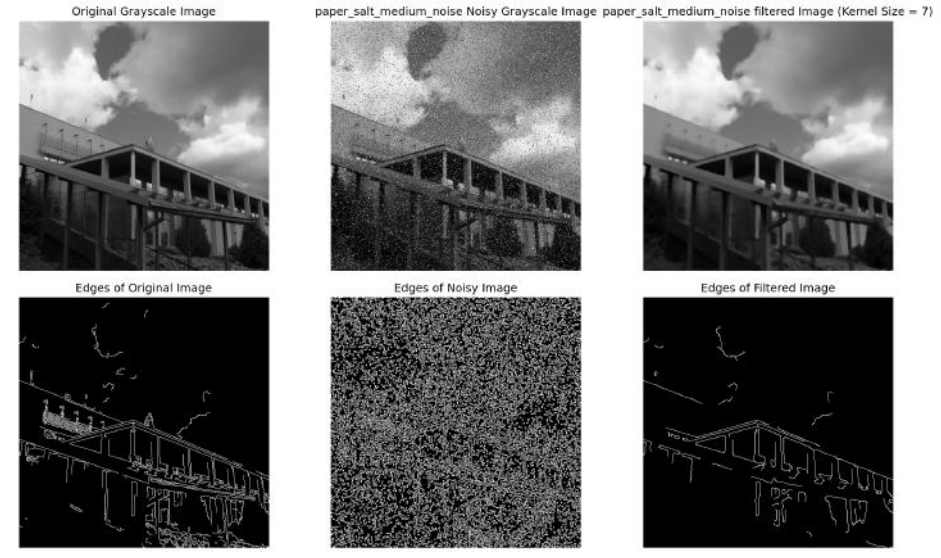
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	8.3013	38.9394	0.0000	0.0295	0.0725	0.8789
5	12.8869	37.0293	0.0020	0.0221	0.0725	0.8472
7	16.4162	35.9781	0.0060	0.0177	0.0725	0.8241



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 25: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

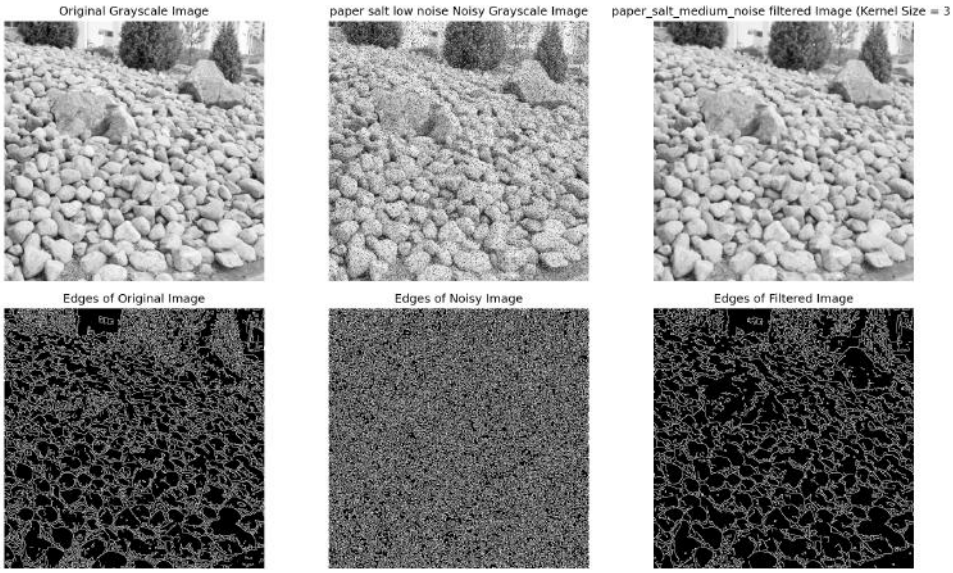
3.3.6 Sixth - Image - Salt-and-Pepper Noise - High Level

Edge Density:

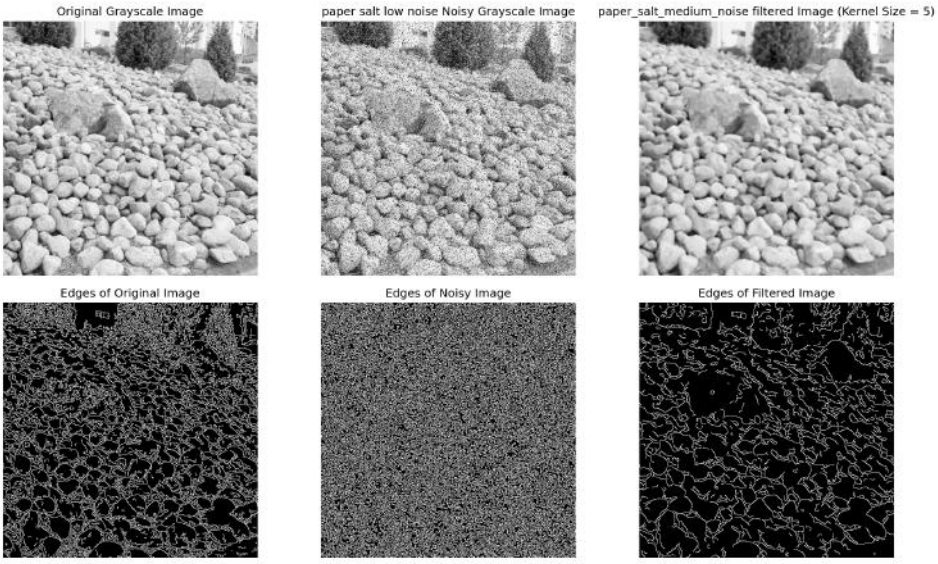
- **Original Image**: 0.2108
- **Noisy Image**: 0.3739

Table 22: Median Filter Performance Metrics - Salt-and-Pepper Noise (High Level)

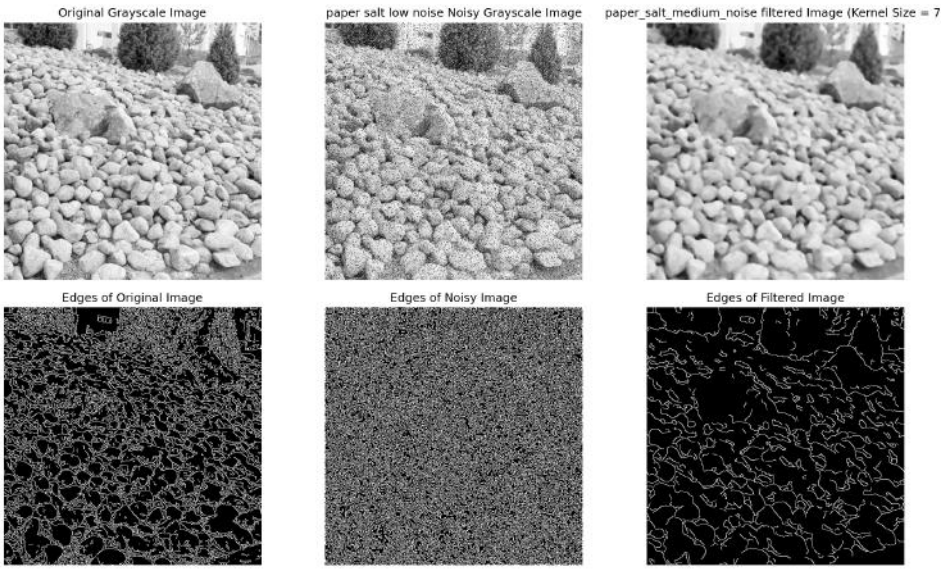
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	45.9302	31.5098	0.0000	0.1571	0.1175	0.4653
5	60.3895	30.3212	0.0040	0.1027	0.1175	0.3226
7	68.8456	29.7520	0.0080	0.0727	0.1175	0.2462



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 26: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

3.3.7 Summary of Results

Table 23: Summary of Results for Different Noise Types and Levels

Noise Type & Level	Kernel Size	Average MSE	Average PSNR (dB)	Average Filtering Time (s)	Average Edge Density - Filtered	Average SSIM (Original vs Noisy)	Average SSIM (Original vs Filtered)
Gaussian Noise - Low Level	3	39.3593	32.1803	0.0000	0.0078	0.2522	0.9762
	5	23.6599	34.3907	0.0040	0.0060	0.2522	0.9784
	7	19.1871	35.3007	0.0042	0.0054	0.2522	0.9768
Gaussian Noise - Medium Level	3	76.7794	29.2784	0.0085	0.1727	0.1347	0.2791
	5	65.7728	29.9503	0.0032	0.0540	0.1347	0.5102
	7	60.9982	30.2776	0.0119	0.0187	0.1347	0.4872
Gaussian Noise - High Level	3	95.0296	28.3522	0.0000	0.3407	0.0373	0.0353
	5	84.5429	28.8600	0.0000	0.1567	0.0373	0.1643
	7	77.9243	29.2141	0.0035	0.0727	0.0373	0.4471
Salt-and-Pepper Noise - Low Level	3	24.6156	34.2187	0.0354	0.0480	0.2389	0.7081
	5	35.3798	32.6432	0.0010	0.0229	0.2389	0.6265
	7	41.0205	32.0008	0.0060	0.0133	0.2389	0.5995
Salt-and-Pepper Noise - Medium Level	3	8.3013	38.9394	0.0000	0.0295	0.0725	0.8789
	5	12.8869	37.0293	0.0020	0.0221	0.0725	0.8472
	7	16.4162	35.9781	0.0060	0.0177	0.0725	0.8241
Salt-and-Pepper Noise - High Level	3	45.9302	31.5098	0.0000	0.1571	0.1175	0.4653
	5	60.3895	30.3212	0.0040	0.1027	0.1175	0.3226
	7	68.8456	29.7520	0.0080	0.0727	0.1175	0.2462

3.4 Adaptive Mean Filter

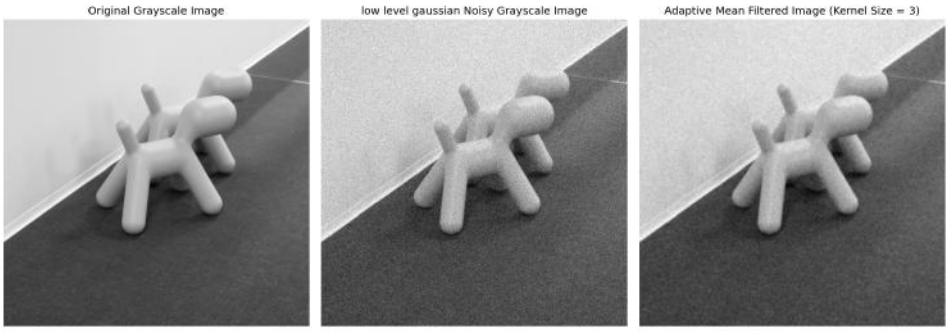
3.4.1 Gaussian Noise - Low Level

Edge Density:

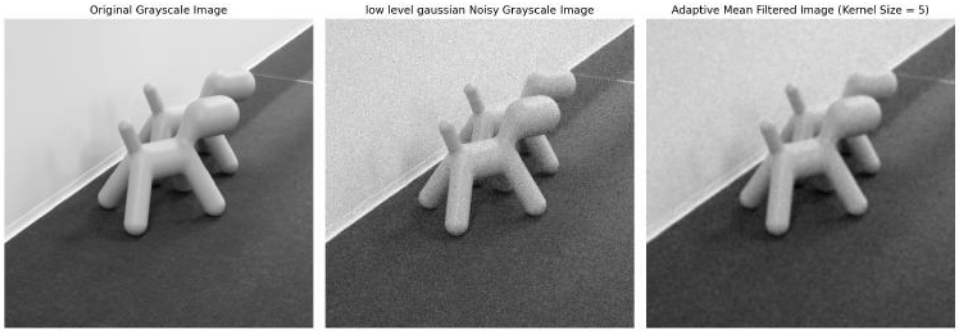
- **Original Image**: 0.0072
- **Noisy Image**: 0.1182

Table 24: Adaptive Mean Filter Performance Metrics - Gaussian Noise (Low Level)

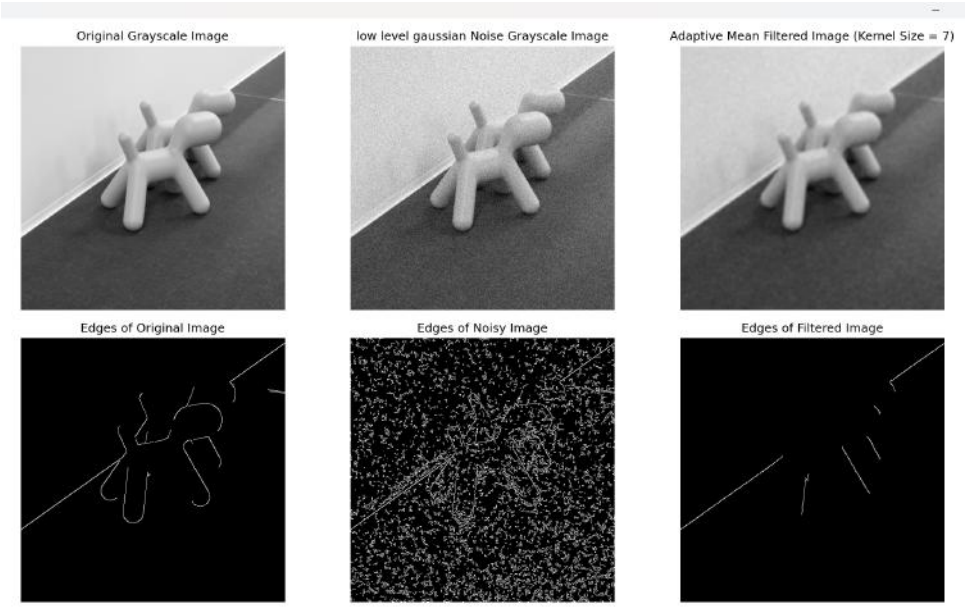
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	31.0670	33.2078	0.0140	0.0065	0.2522	0.9803
5	20.3246	35.0506	0.0070	0.0046	0.2522	0.9667
7	18.3635	35.4913	0.0060	0.0033	0.2522	0.9595



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 27: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

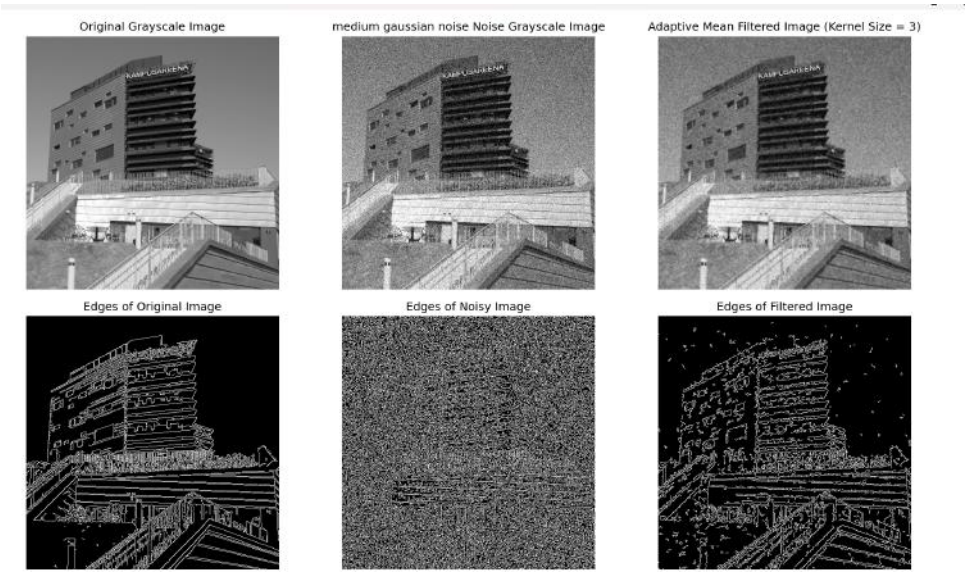
3.4.2 Gaussian Noise - Medium Level

Edge Density:

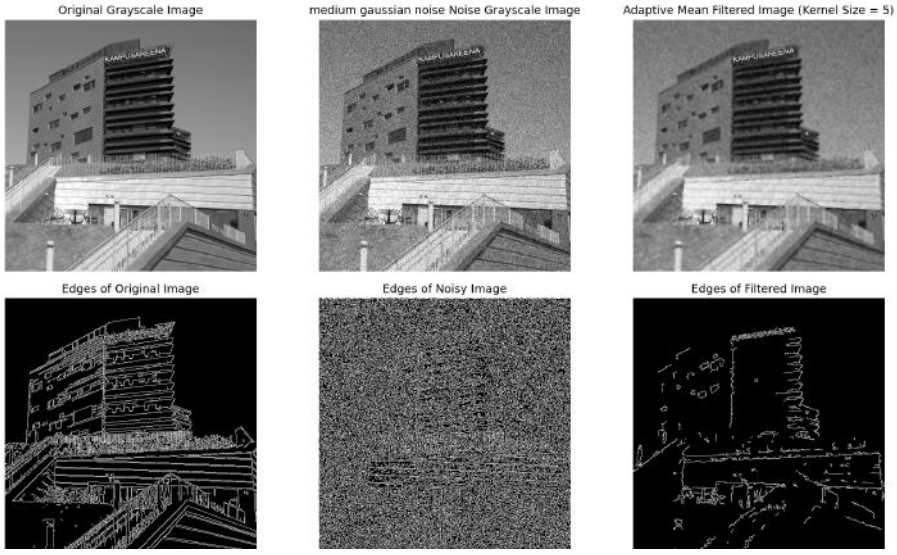
- **Original Image**: 0.1245
- **Noisy Image**: 0.3593

Table 25: Adaptive Mean Filter Performance Metrics - Gaussian Noise (Medium Level)

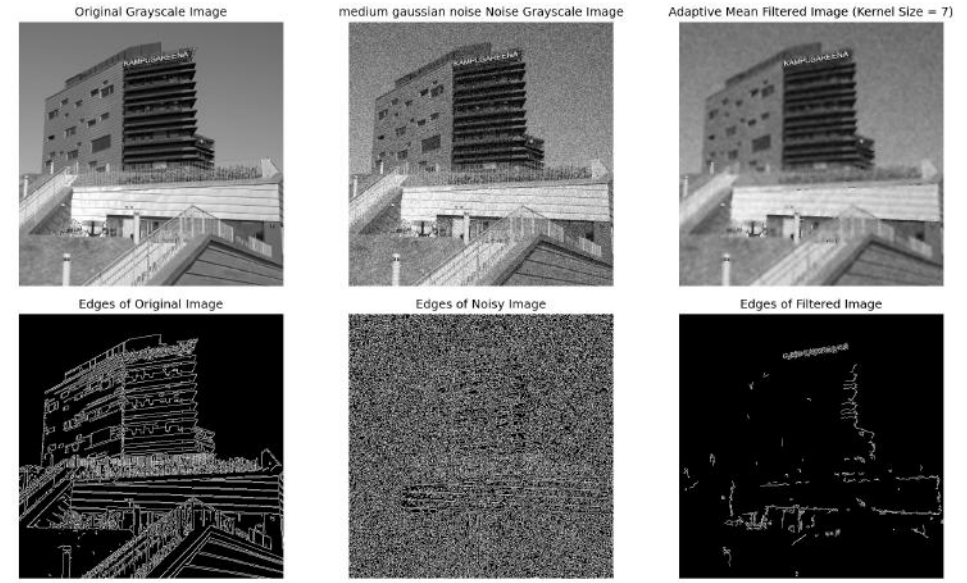
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	71.9701	29.5593	0.0075	0.1128	0.1347	0.5471
5	62.8834	30.1454	0.0060	0.0306	0.1347	0.4941
7	60.8896	30.2855	0.0060	0.0153	0.1347	0.4859



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 28: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

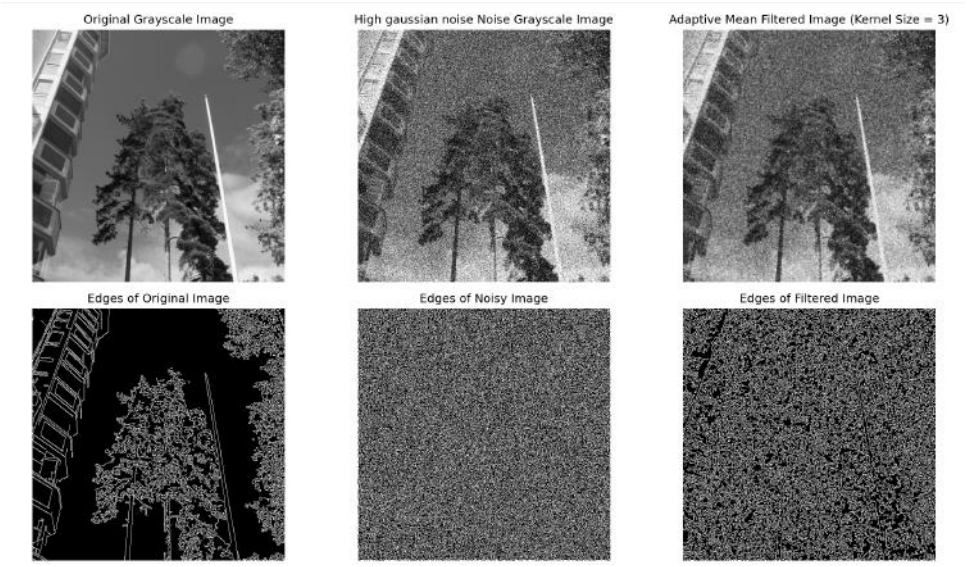
3.4.3 Gaussian Noise - High Level

Edge Density:

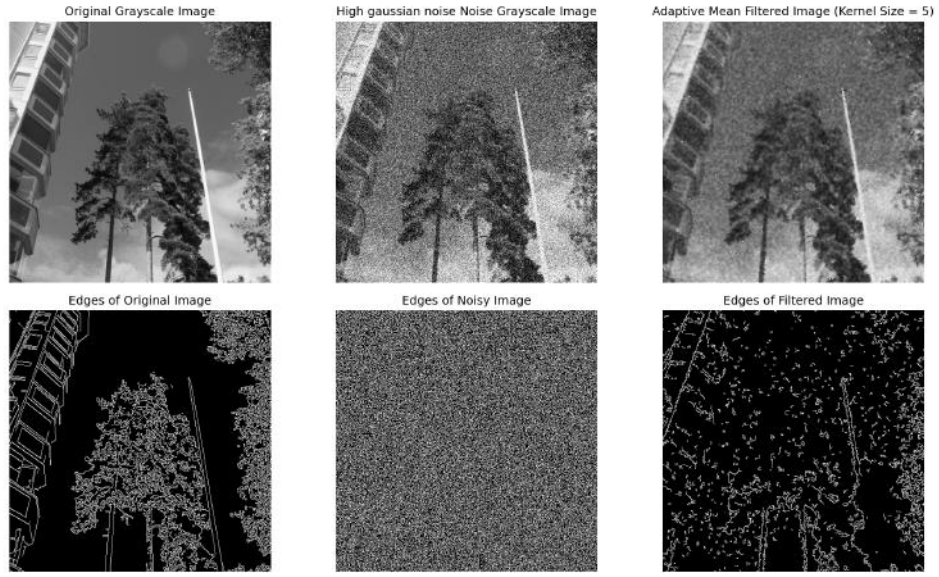
- **Original Image**: 0.1320
- **Noisy Image**: 0.3726

Table 26: Adaptive Mean Filter Performance Metrics - Gaussian Noise (High Level)

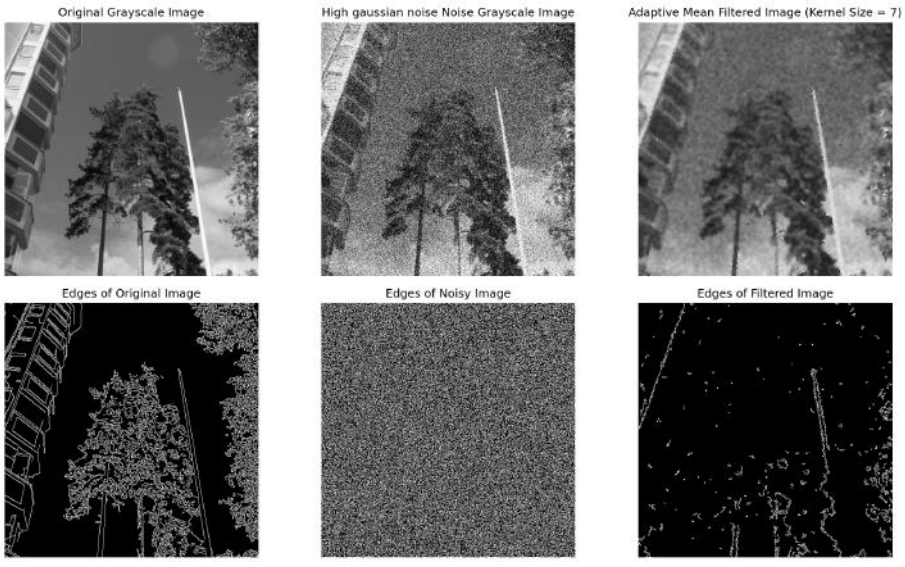
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	90.3340	28.5723	0.0040	0.3012	0.0373	0.0529
5	80.3724	29.0797	0.0039	0.0733	0.0373	0.3627
7	74.7422	29.3951	0.0079	0.1283	0.0373	0.0887



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 29: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

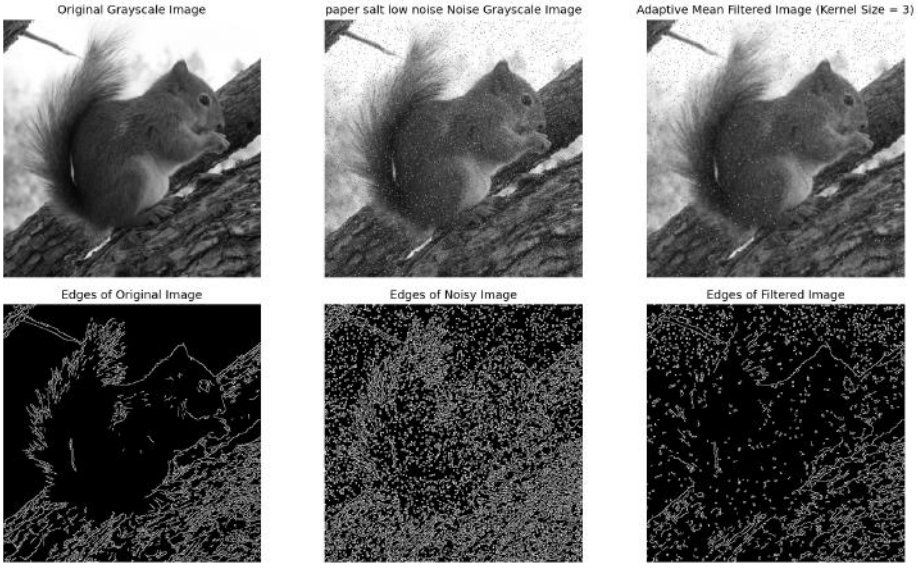
3.4.4 Salt-and-Pepper Noise - Low Level

Edge Density:

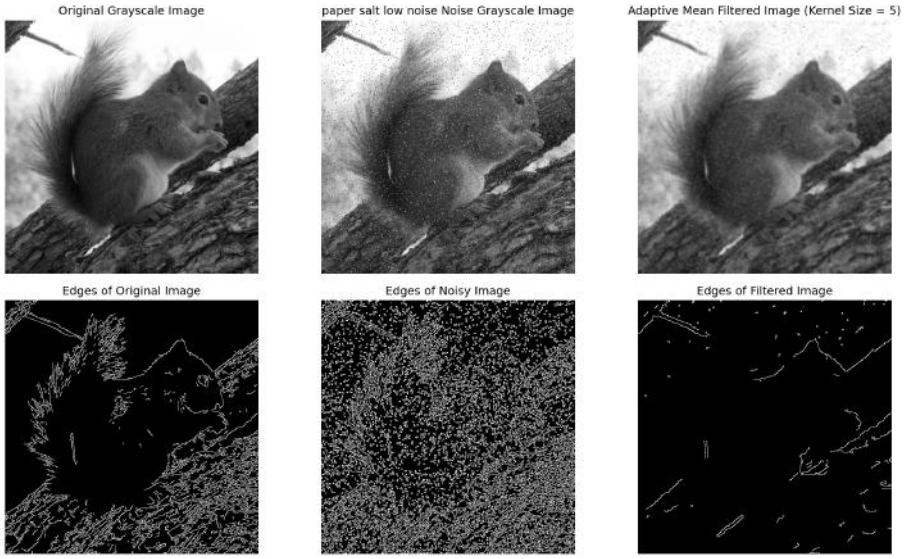
- **Original Image**: 0.0907
- **Noisy Image**: 0.2342

Table 27: Adaptive Mean Filter Performance Metrics - Salt-and-Pepper Noise (Low Level)

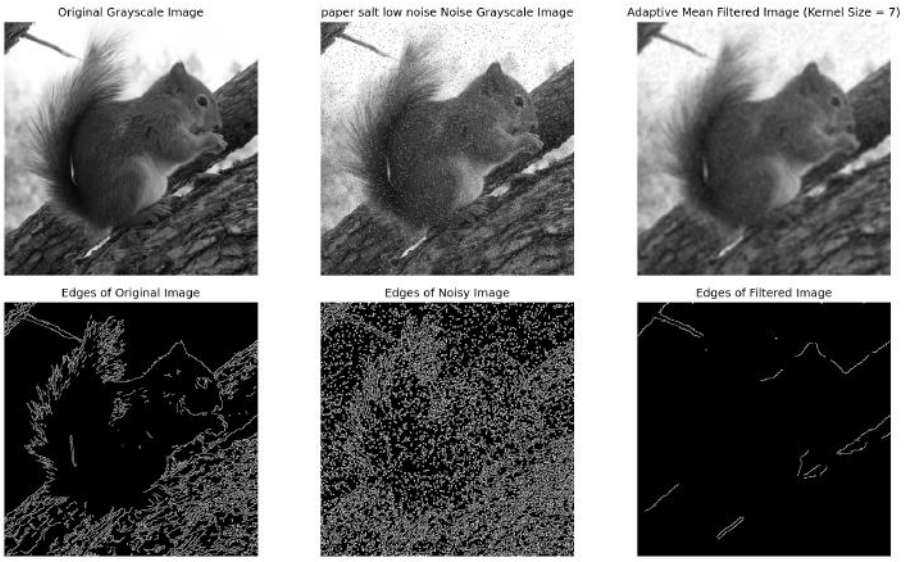
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	43.6089	31.7350	0.0089	0.0910	0.2389	0.4364
5	58.9730	30.4243	0.0120	0.0114	0.2389	0.5653
7	60.1052	30.3417	0.0090	0.0060	0.2389	0.5699



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 30: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

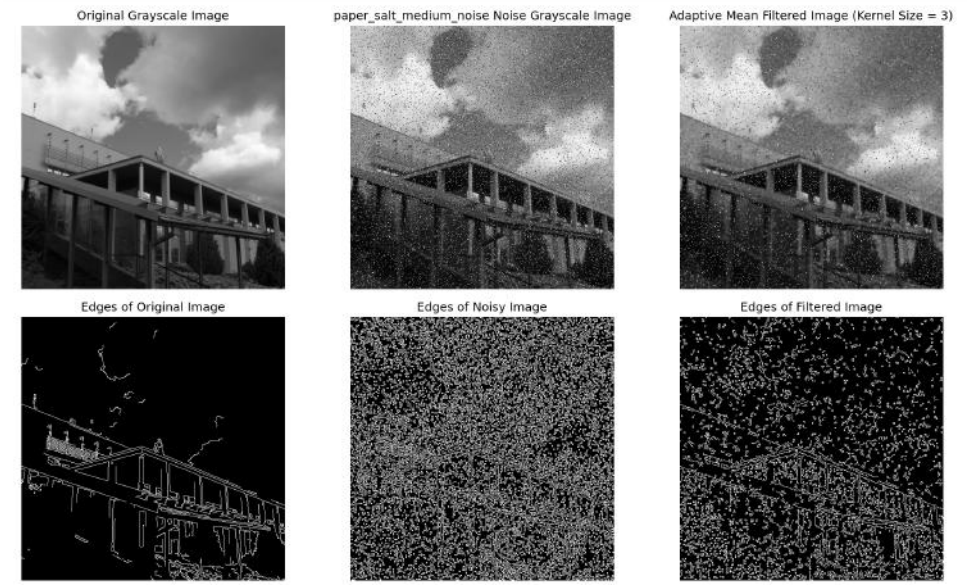
3.4.5 Salt-and-Pepper Noise - Medium Level

Edge Density:

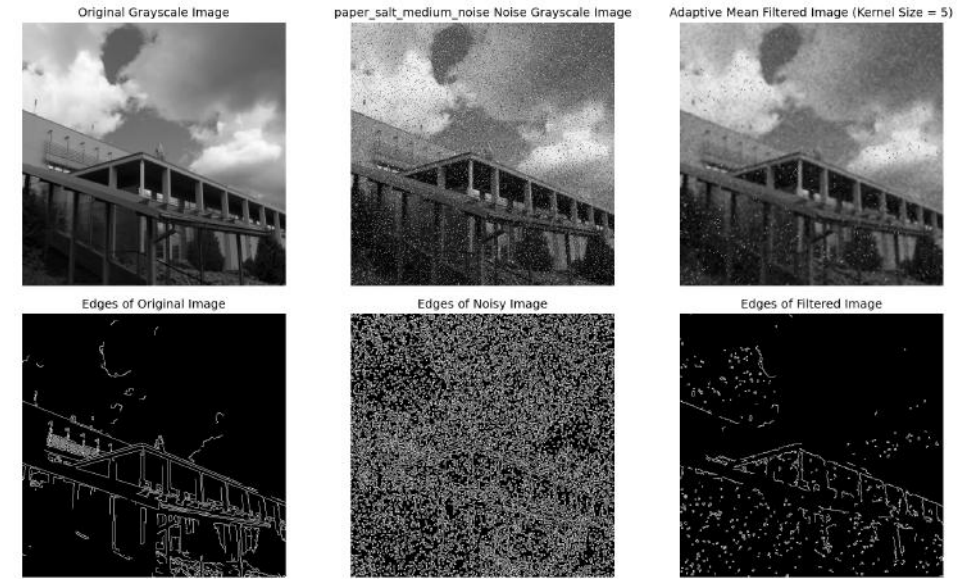
- **Original Image**: 0.04199
- **Noisy Image**: 0.29196

Table 28: Adaptive Mean Filter Performance Metrics - Salt-and-Pepper Noise (Medium Level)

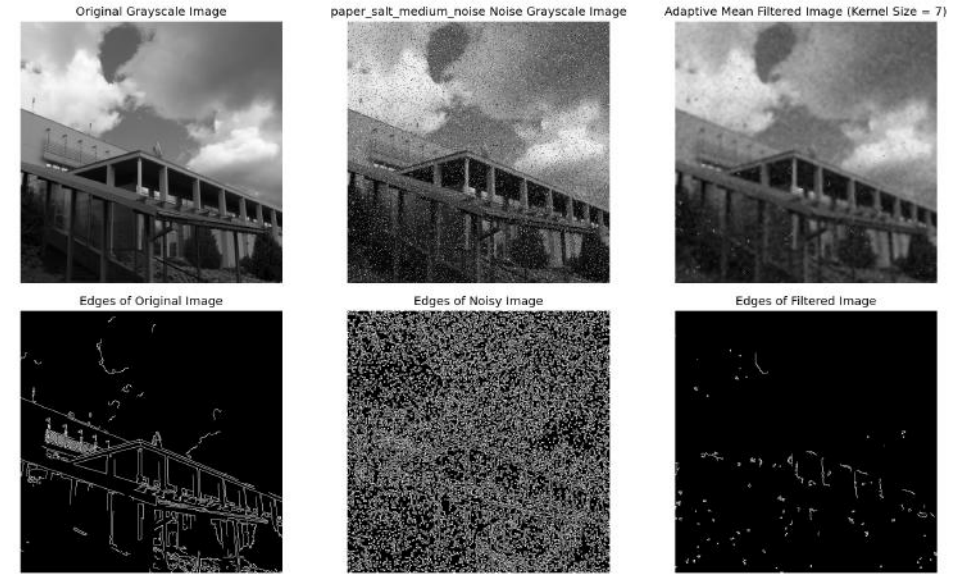
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	47.2935	31.3828	0.0080	0.1356	0.0725	0.3098
5	54.8966	30.7353	0.0070	0.0299	0.0725	0.6639
7	55.7865	30.6651	0.0070	0.0177	0.0725	0.7370



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 31: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

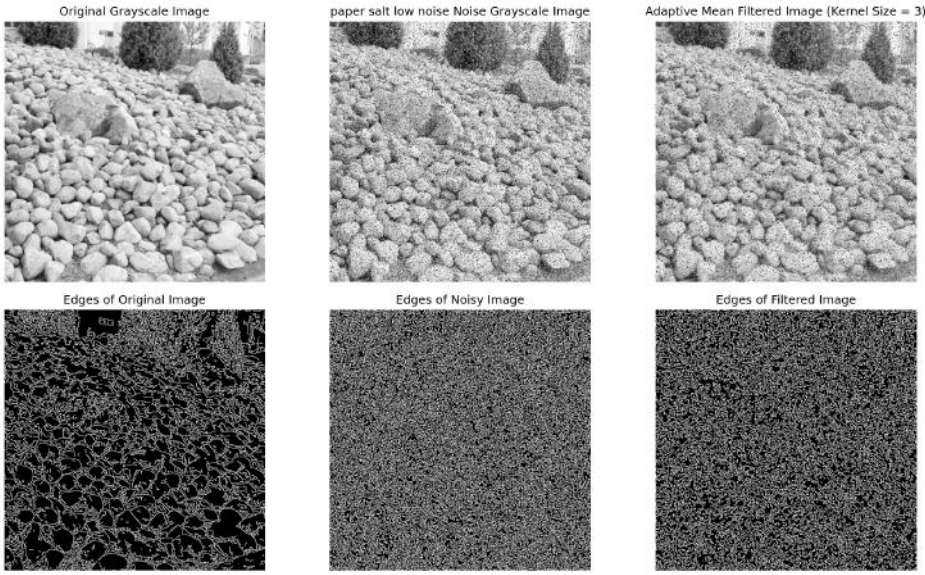
3.4.6 Salt-and-Pepper Noise - High Level

Edge Density:

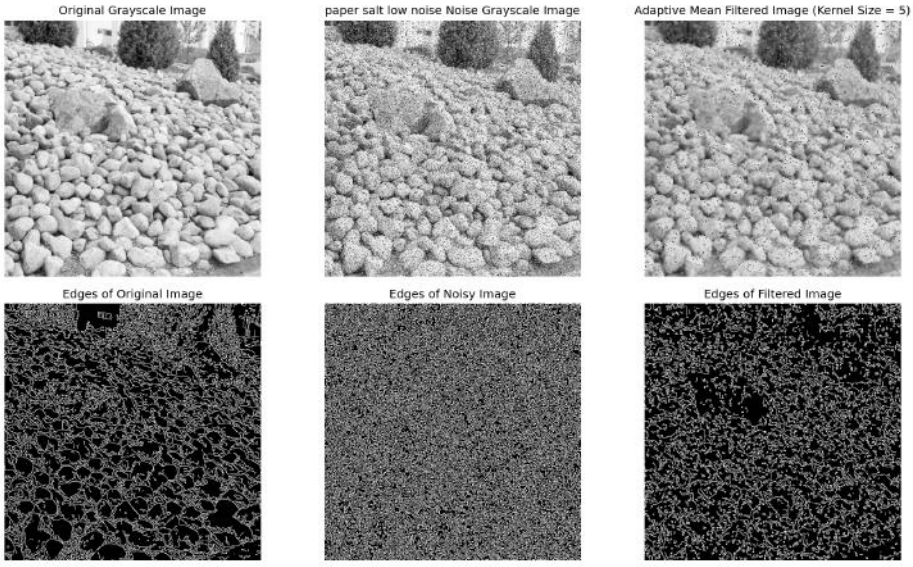
- **Original Image**: 0.2108
- **Noisy Image**: 0.3739

Table 29: Adaptive Mean Filter Performance Metrics - Salt-and-Pepper Noise (High Level)

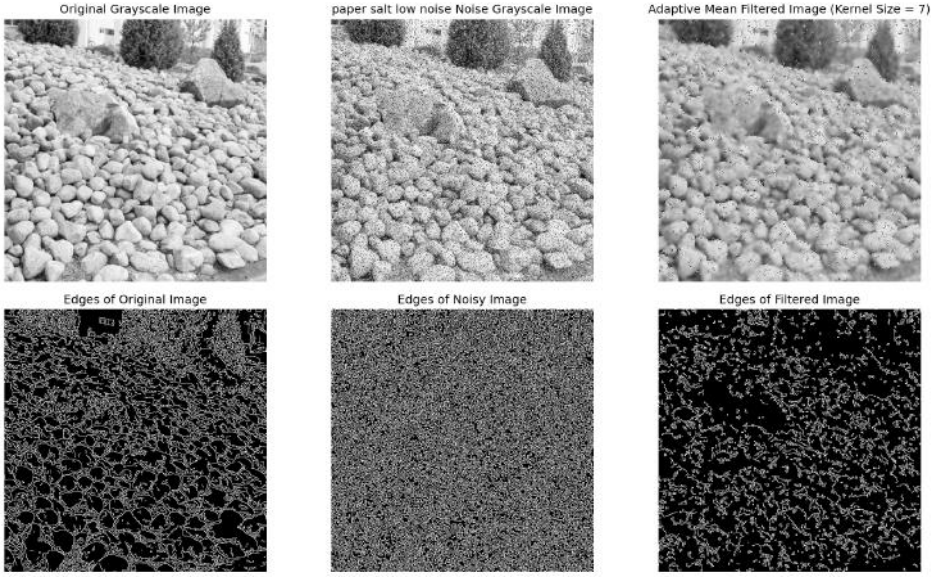
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	66.7082	29.8890	0.0080	0.3022	0.1175	0.4653
5	76.1978	29.3114	0.0040	0.1907	0.1175	0.3226
7	82.1229	28.9862	0.0080	0.1283	0.1175	0.0887



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 32: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

3.4.7 Summary of Results

Table 30: Summary of Results for Different Noise Types and Levels

Noise Type & Level	Kernel Size	Average MSE	Average PSNR (dB)	Average Filtering Time (s)	Average Edge Density - Filtered	Average SSIM (Original vs Noisy)	Average SSIM (Original vs Filtered)
Gaussian Noise - Low Level	3	31.0670	33.2078	0.0140	0.0065	0.2522	0.9803
	5	20.3246	35.0506	0.0070	0.0046	0.2522	0.9667
	7	18.3635	35.4913	0.0060	0.0033	0.2522	0.9595
Gaussian Noise - Medium Level	3	71.9701	29.5593	0.0075	0.1128	0.1347	0.5471
	5	62.8834	30.1454	0.0060	0.0306	0.1347	0.4941
	7	60.8896	30.2855	0.0060	0.0153	0.1347	0.4859
Gaussian Noise - High Level	3	90.3340	28.5723	0.0040	0.3012	0.0373	0.0529
	5	80.3724	29.0797	0.0039	0.0733	0.0373	0.3627
	7	74.7422	29.3951	0.0079	0.1283	0.0373	0.0887
Salt-and-Pepper Noise - Low Level	3	43.6089	31.7350	0.0089	0.0910	0.2389	0.4364
	5	58.9730	30.4243	0.0120	0.0114	0.2389	0.5653
	7	60.1052	30.3417	0.0090	0.0060	0.2389	0.5699
Salt-and-Pepper Noise - Medium Level	3	47.2935	31.3828	0.0080	0.1356	0.0725	0.3098
	5	54.8966	30.7353	0.0070	0.0299	0.0725	0.6639
	7	55.7865	30.6651	0.0070	0.0177	0.0725	0.7370
Salt-and-Pepper Noise - High Level	3	66.7082	29.8890	0.0080	0.3022	0.1175	0.4653
	5	76.1978	29.3114	0.0040	0.1907	0.1175	0.3226
	7	82.1229	28.9862	0.0080	0.1283	0.1175	0.0887

3.5 Adaptive Median filter

3.5.1 Gaussian Noise - Low Level

Edge Density:

- **Original Image**: 0.0072
- **Noisy Image**: 0.1182

Table 31: Adaptive Median Filter Performance Metrics - Gaussian Noise (Low Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	61.9972	30.2071	2.6053	0.0186	0.2522	0.8689

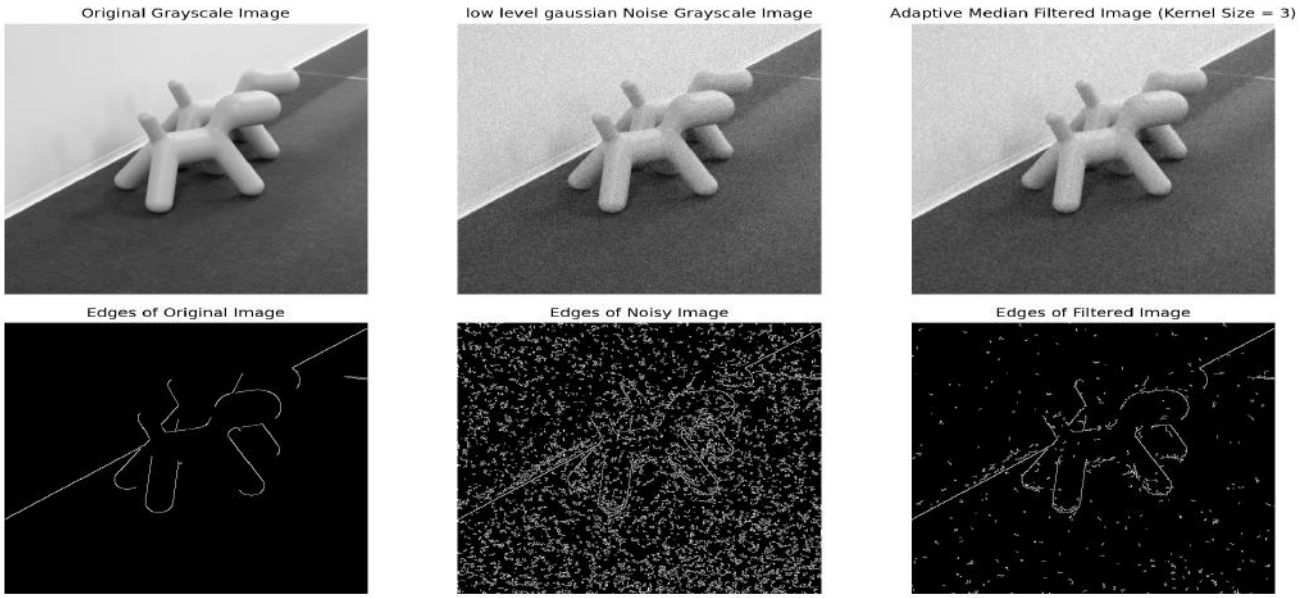


Figure 33: canny edge detector

3.5.2 Gaussian Noise - Medium Level

Edge Density:

- **Original Image**: 0.1245
- **Noisy Image**: 0.3593

Table 32: Adaptive Median Filter Performance Metrics - Gaussian Noise (Medium Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	88.3713	28.6677	2.9092	0.3208	0.1347	0.1268

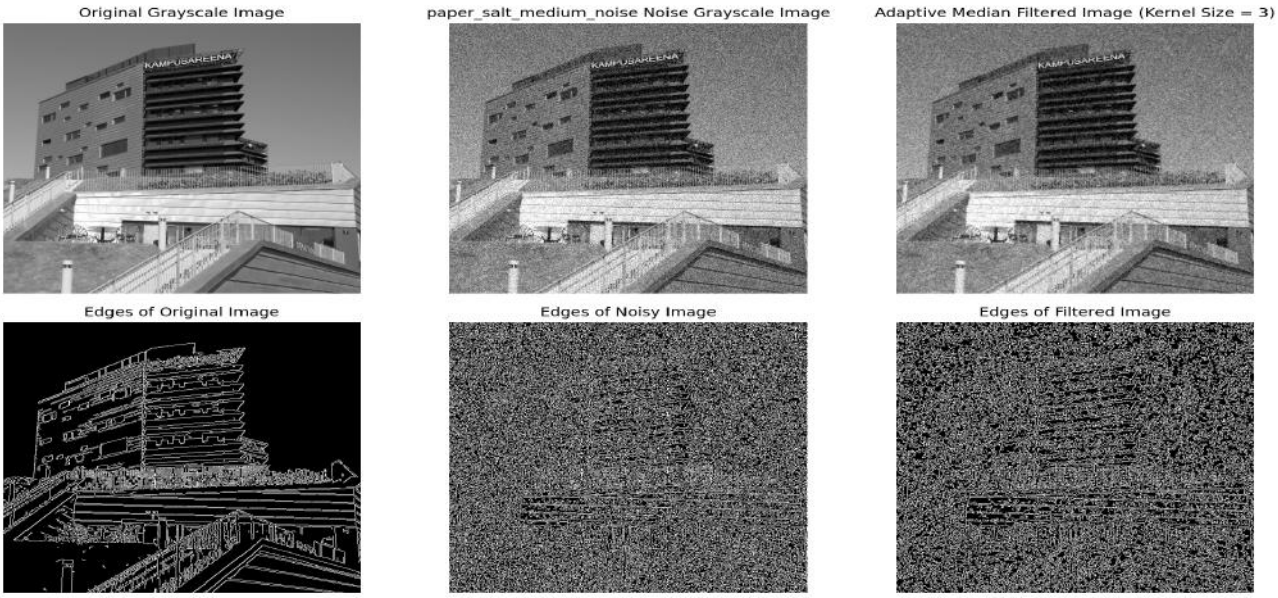


Figure 34: canny edge detector

3.5.3 Gaussian Noise - High Level

Edge Density:

- **Original Image**: 0.1320
- **Noisy Image**: 0.3726

Table 33: Adaptive Median Filter Performance Metrics - Gaussian Noise (High Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
7	74.7422	29.3951	0.0061	0.0249	0.0373	0.4549

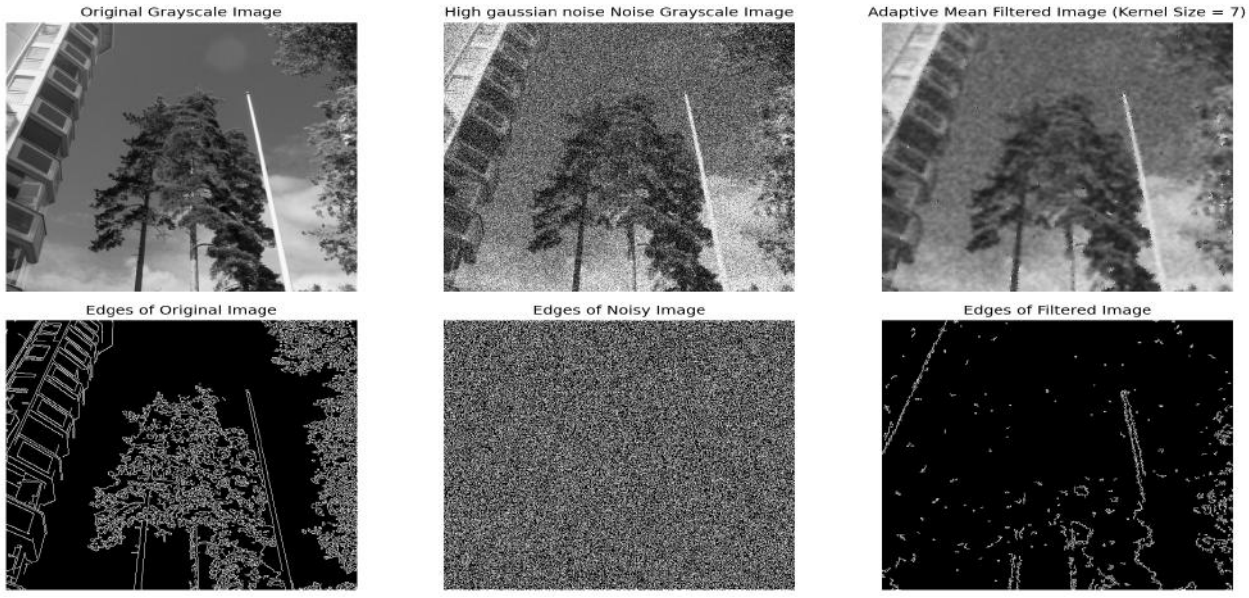


Figure 35: canny edge detector

3.5.4 Salt-and-Pepper Noise - Low Level

Edge Density:

- **Original Image**: 0.0907
- **Noisy Image**: 0.2342

Table 34: Adaptive Median Filter Performance Metrics - Salt-and-Pepper Noise (Low Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	10.7564	37.8141	3.3942	0.0816	0.2389	0.7599

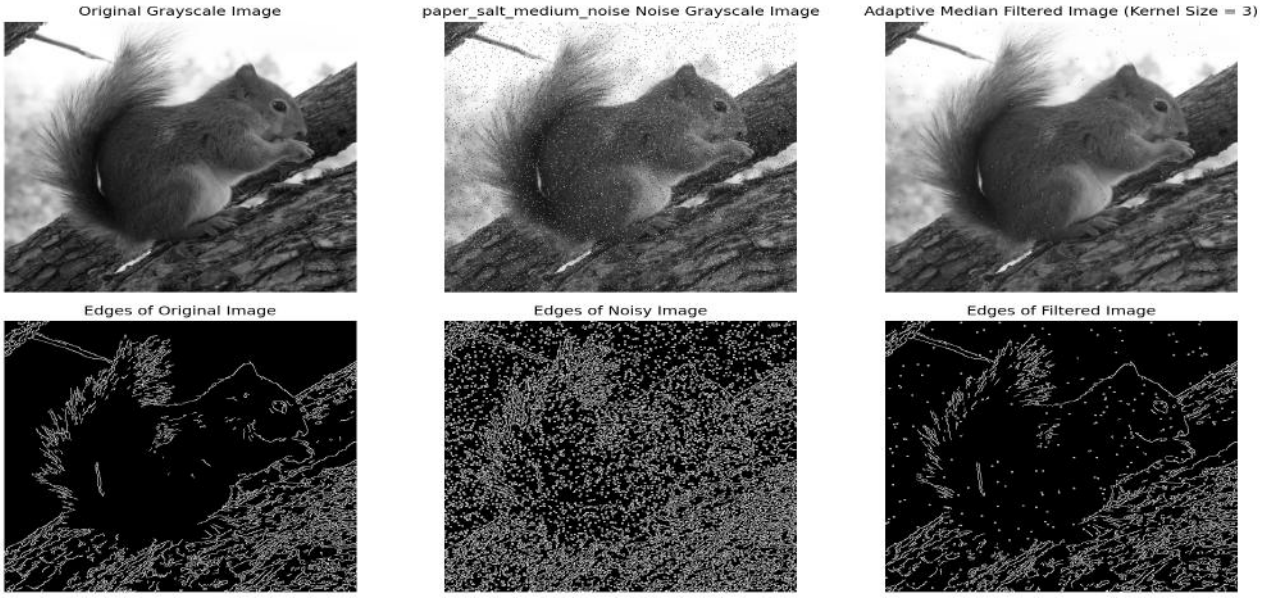


Figure 36: canny edge detector

3.5.5 Salt-and-Pepper Noise - Medium Level

Edge Density:

- **Original Image**: 0.04199
- **Noisy Image**: 0.29196

Table 35: Adaptive Median Filter Performance Metrics - Salt-and-Pepper Noise (Medium Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	5.6679	40.5966	2.5533	0.0687	0.0725	0.6851

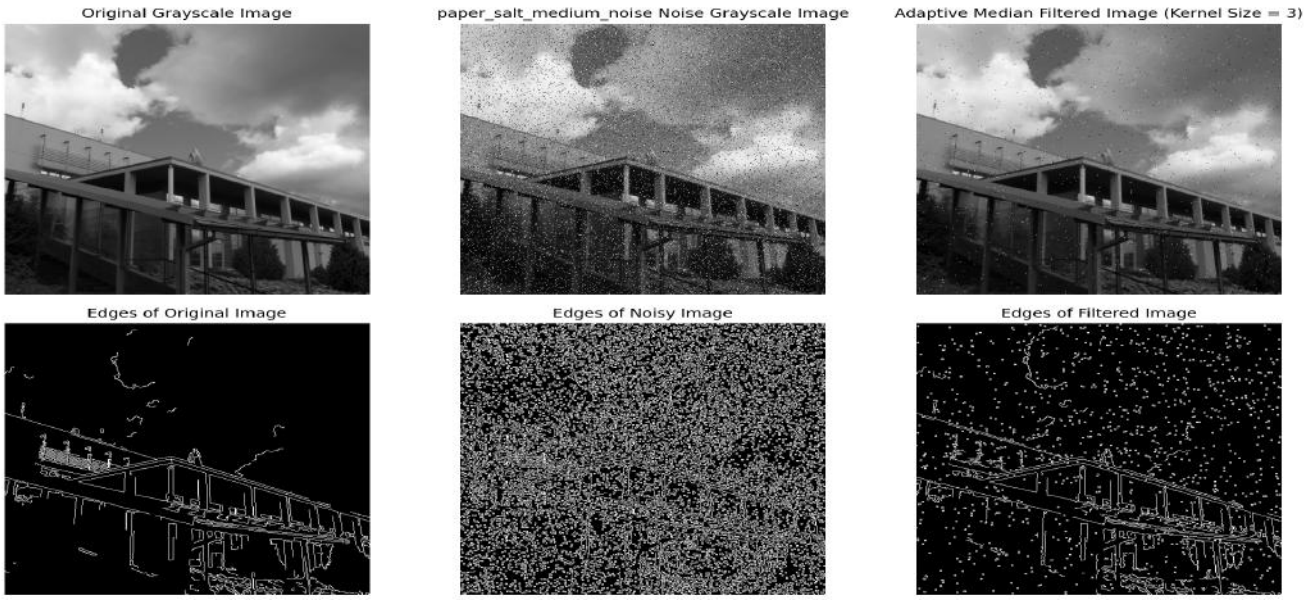


Figure 37: canny edge detector

3.5.6 Salt-and-Pepper Noise - High Level

Edge Density:

- **Original Image**: 0.2108
- **Noisy Image**: 0.3739

Table 36: Adaptive Median Filter Performance Metrics - Salt-and-Pepper Noise (High Level)

Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	19.0671	35.3279	2.6480	0.2455	0.1175	0.4561

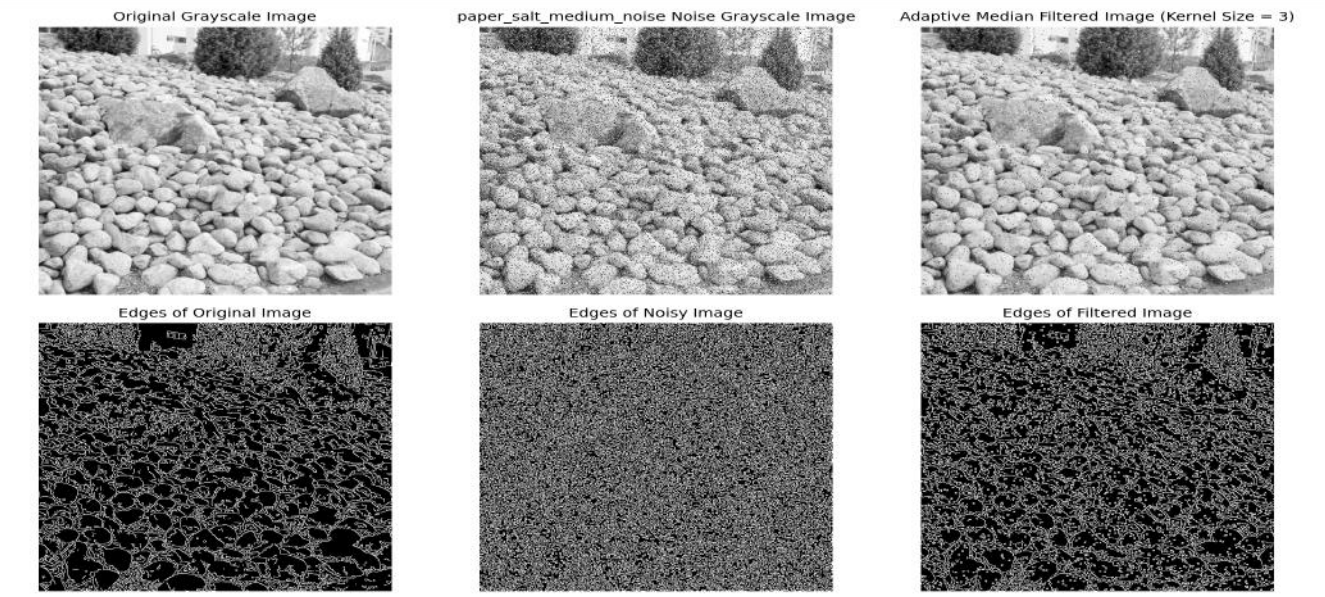


Figure 38: canny edge detector

3.5.7 Summary of Results

Table 37: Summary of Adaptive Median Filter Performance Metrics for Different Noise Types and Levels

Noise Type & Level	Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
Gaussian Noise - Low Level	3	61.9972	30.2071	2.6053	0.0186	0.2522	0.8689
Gaussian Noise - Medium Level	3	88.3713	28.6677	2.9092	0.3208	0.1347	0.1268
Gaussian Noise - High Level	7	74.7422	29.3951	0.0061	0.0249	0.0373	0.4549
Salt-and-Pepper Noise - Low Level	3	10.7564	37.8141	3.3942	0.0816	0.2389	0.7599
Salt-and-Pepper Noise - Medium Level	3	5.6679	40.5966	2.5533	0.0687	0.0725	0.6851
Salt-and-Pepper Noise - High Level	3	19.0671	35.3279	2.6480	0.2455	0.1175	0.4561

3.6 Bilateral filter

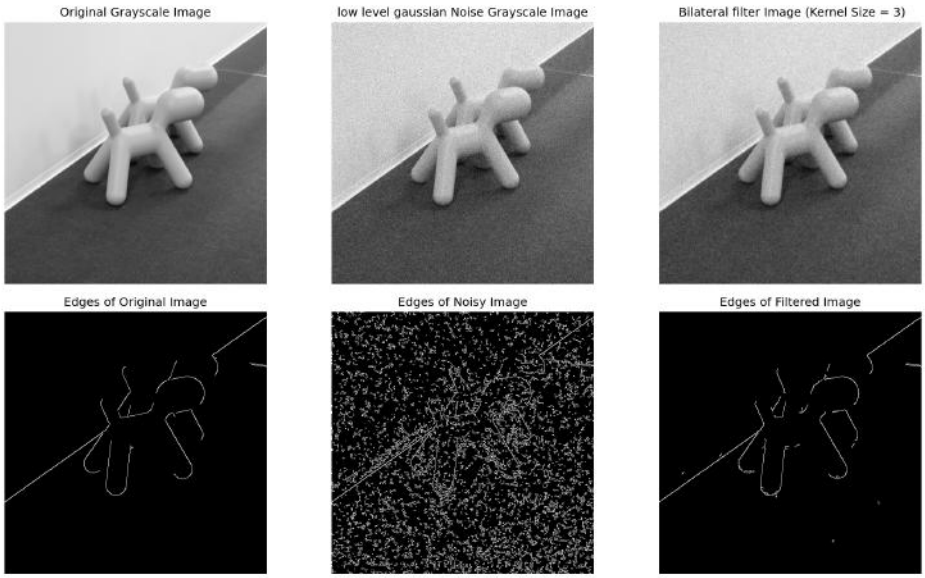
3.6.1 Gaussian Noise - Low Level

Edge Density:

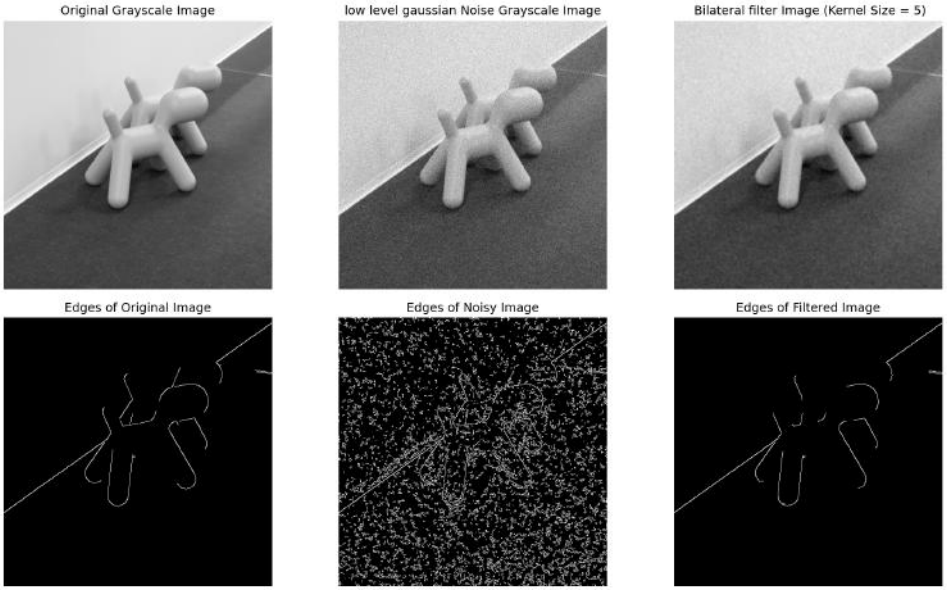
- **Original Image**: 0.0072
- **Noisy Image**: 0.1182

Table 38: Bilateral Filter Performance Metrics - Gaussian Noise (Low Level)

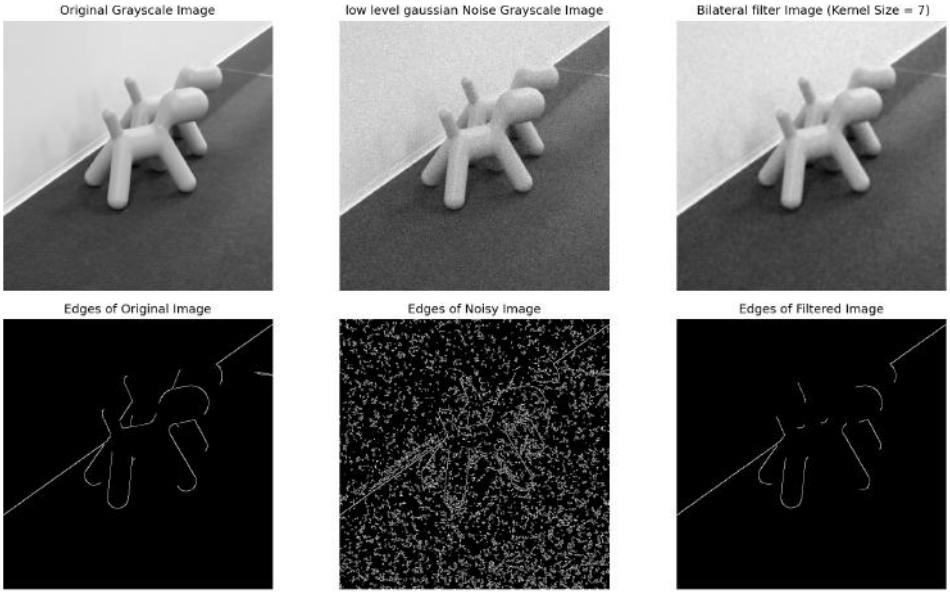
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	43.2225	31.7737	0.0111	0.0076	0.2522	0.9782
5	25.7917	34.0160	0.0039	0.0061	0.2522	0.9813
7	18.6321	35.4282	0.0030	0.0055	0.2522	0.9777



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 39: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

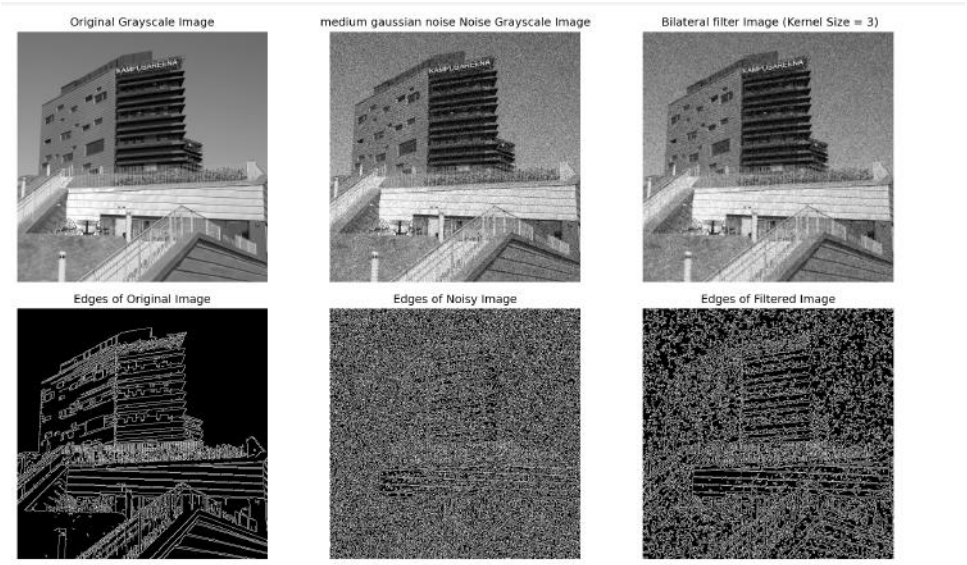
3.6.2 Gaussian Noise - Medium Level

Edge Density:

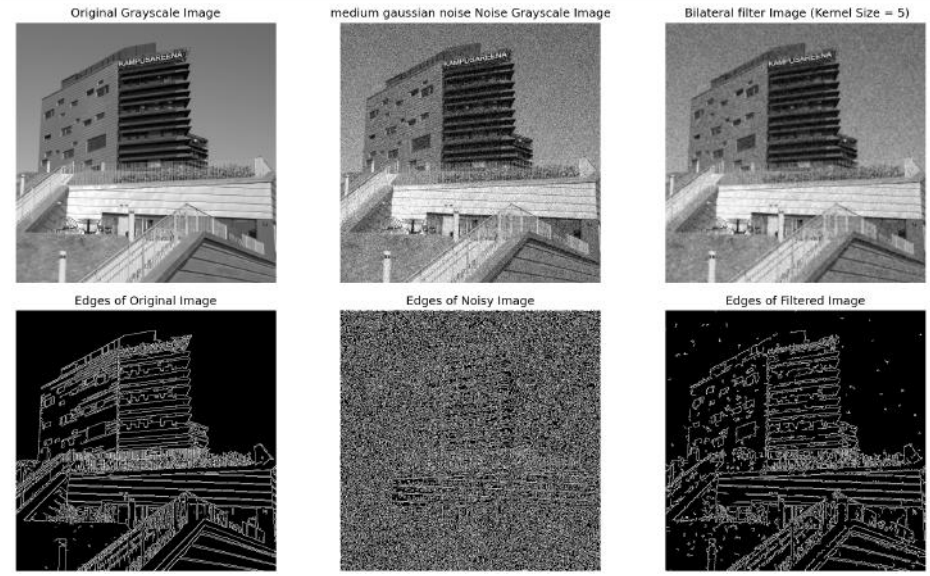
- **Original Image**: 0.1245
- **Noisy Image**: 0.3593

Table 39: Bilateral Filter Performance Metrics - Gaussian Noise (Medium Level)

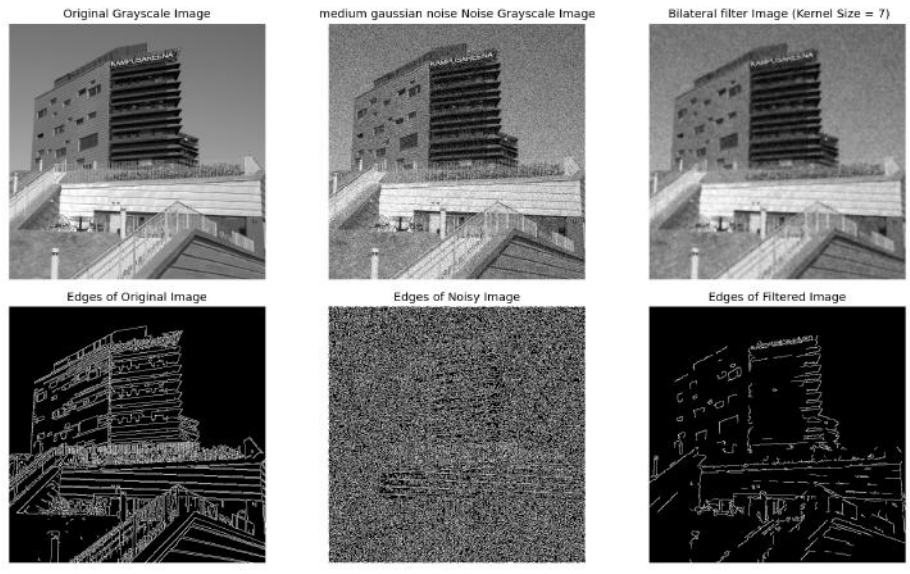
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	78.4473	29.1850	0.0185	0.2645	0.1347	0.2105
5	68.8467	29.7520	0.0040	0.1057	0.1347	0.6296
7	61.5104	30.2413	0.0030	0.0364	0.1347	0.5561



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 40: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

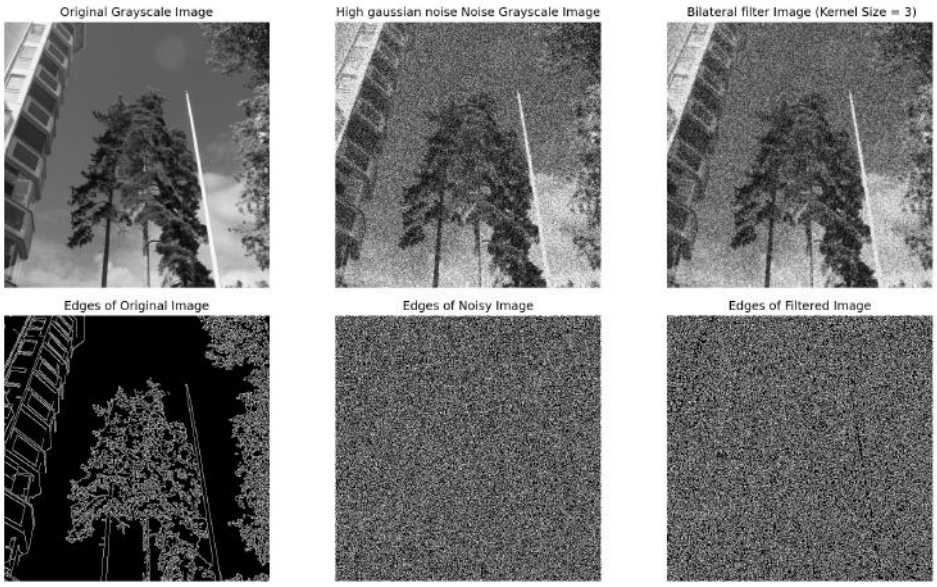
3.6.3 Gaussian Noise - High Level

Edge Density:

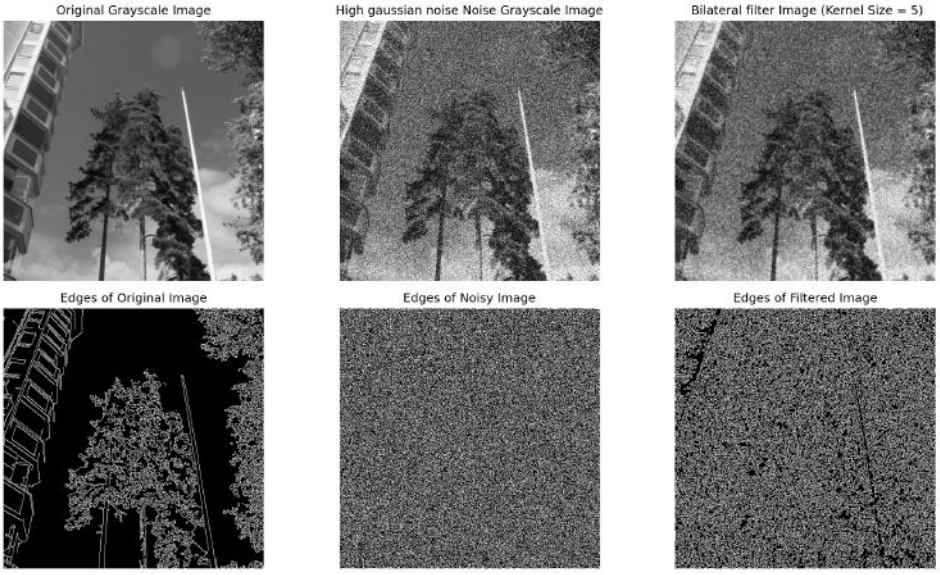
- **Original Image**: 0.1320
- **Noisy Image**: 0.3726

Table 40: Bilateral Filter Performance Metrics - Gaussian Noise (High Level)

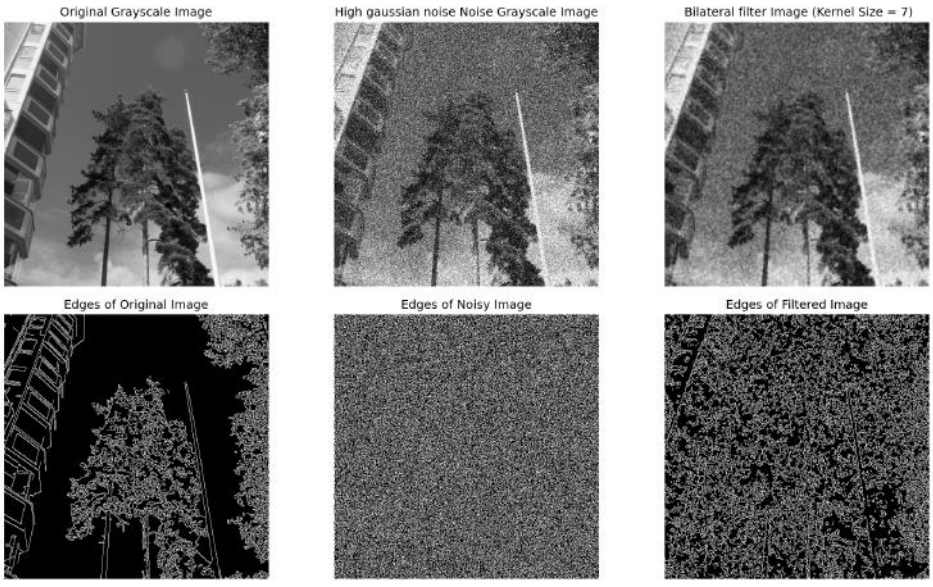
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	99.4321	28.1555	0.0055	0.3738	0.0373	0.0428
5	62.6232	28.3496	0.0040	0.3301	0.0373	0.0554
7	68.8940	29.7490	0.0041	0.3148	0.0373	0.0817



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 41: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

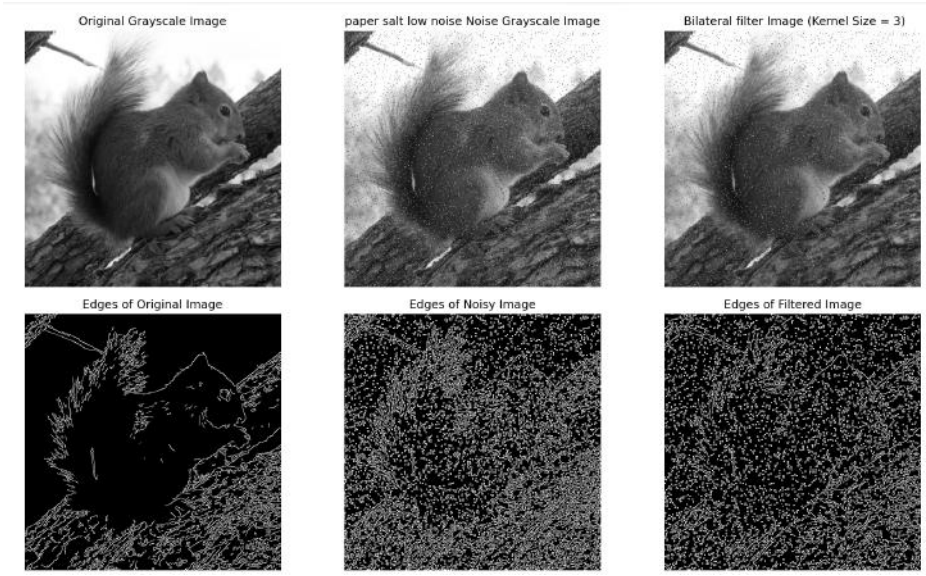
3.6.4 Salt-and-Pepper Noise - Low Level

Edge Density:

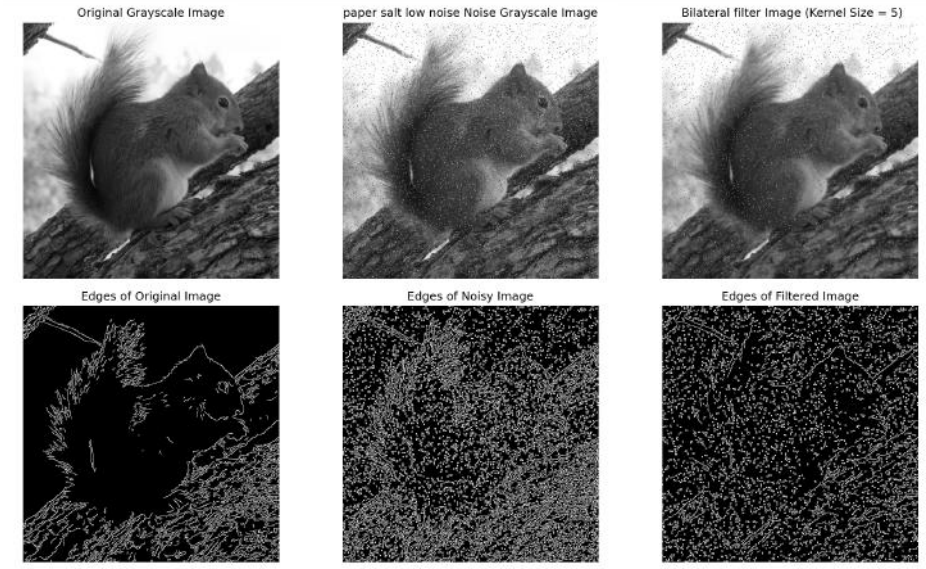
- **Original Image**: 0.0907
- **Noisy Image**: 0.2342

Table 41: Bilateral Filter Performance Metrics - Salt-and-Pepper Noise (Low Level)

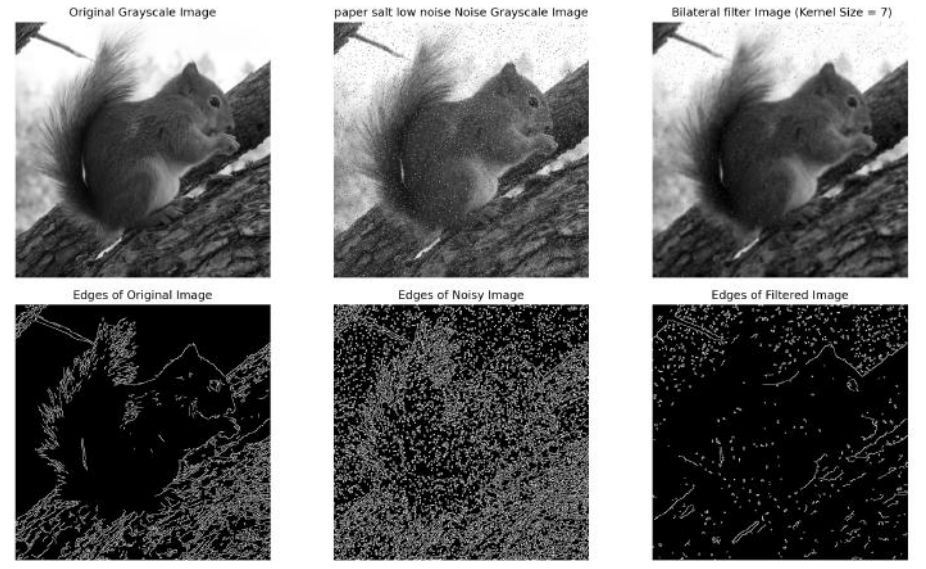
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	27.4256	33.7492	0.0025	0.1795	0.2389	0.2252
5	35.8845	32.5817	0.0047	0.1383	0.2389	0.2118
7	43.2004	31.7759	0.0035	0.0384	0.2389	0.4456



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 42: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

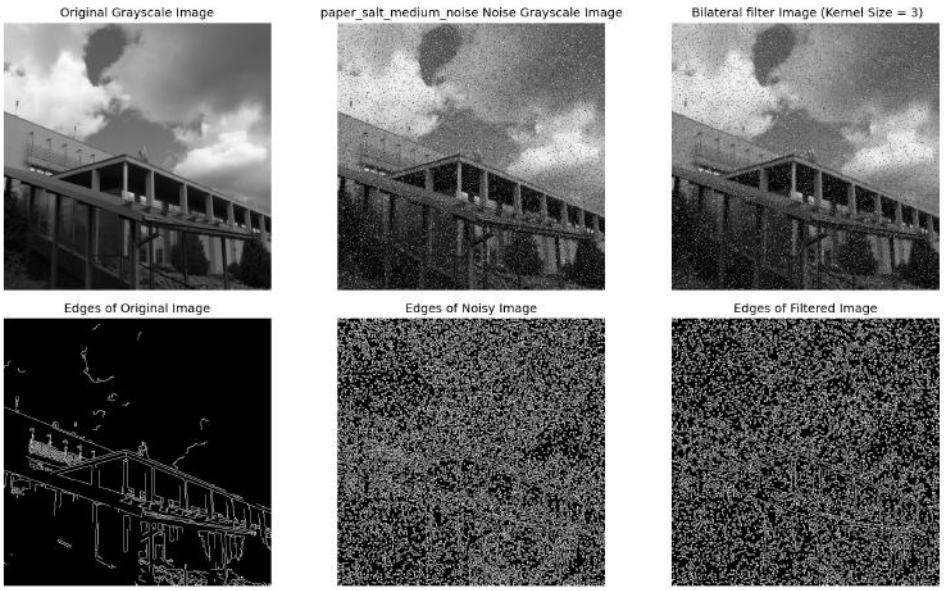
3.6.5 Salt-and-Pepper Noise - Medium Level

Edge Density:

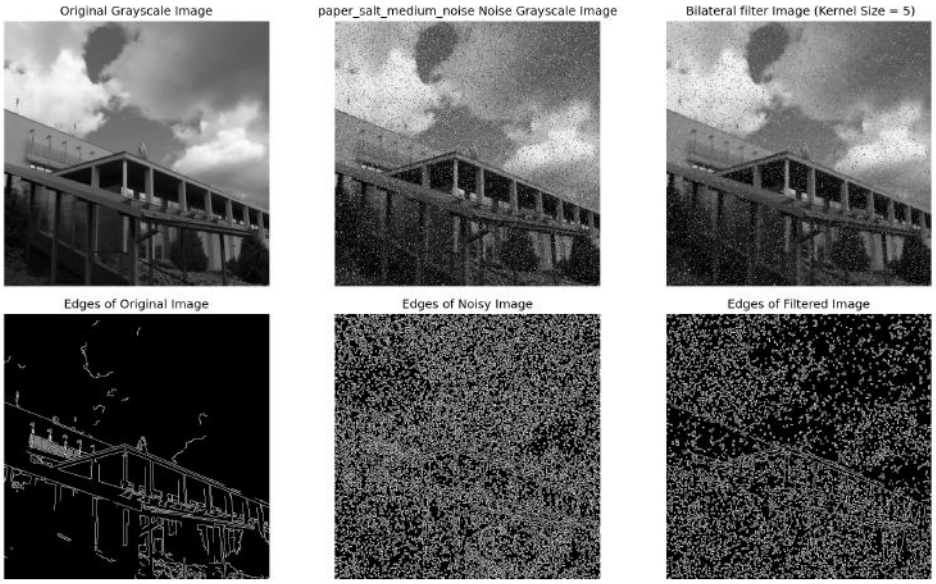
- **Original Image**: 0.04199
- **Noisy Image**: 0.29196

Table 42: Bilateral Filter Performance Metrics - Salt-and-Pepper Noise (Medium Level)

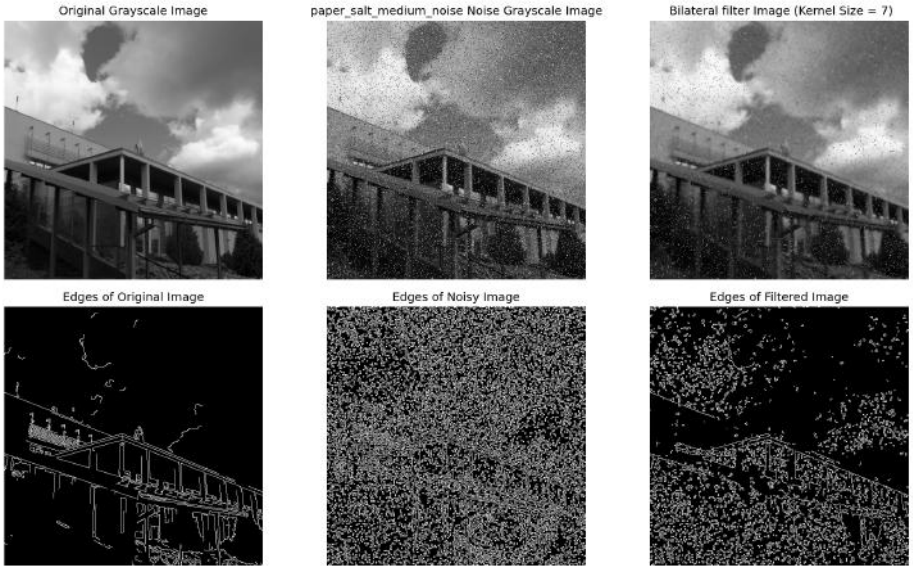
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	24.4859	34.2416	0.0030	0.2492	0.0725	0.0841
5	24.3354	34.2684	0.0040	0.1827	0.0725	0.17996
7	28.0820	33.6465	0.0036	0.1156	0.0725	0.3983



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 43: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

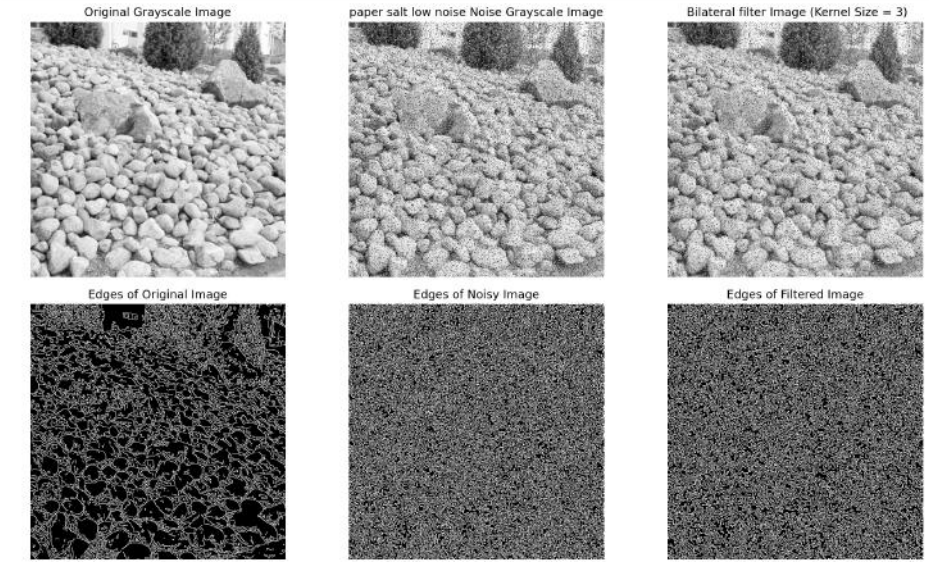
3.6.6 Salt-and-Pepper Noise - High Level

Edge Density:

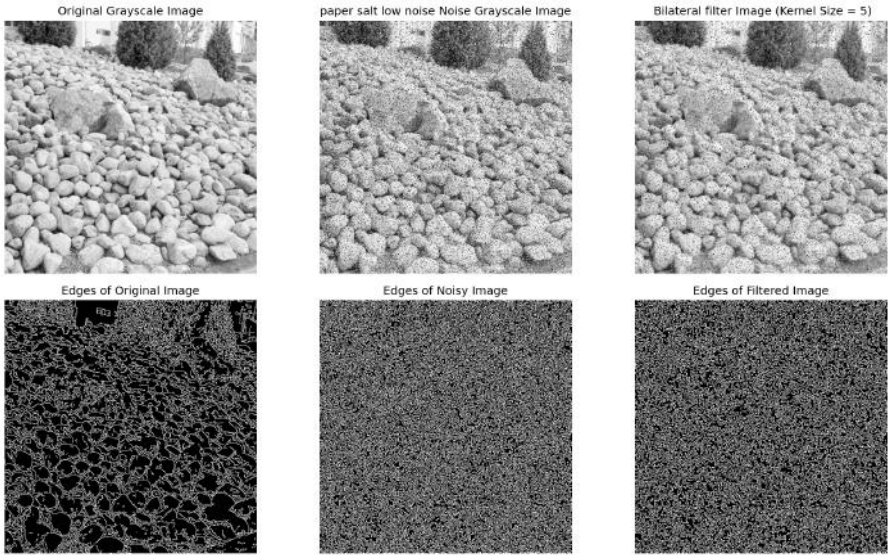
- **Original Image**: 0.2108
- **Noisy Image**: 0.3739

Table 43: Bilateral Filter Performance Metrics - Salt-and-Pepper Noise (High Level)

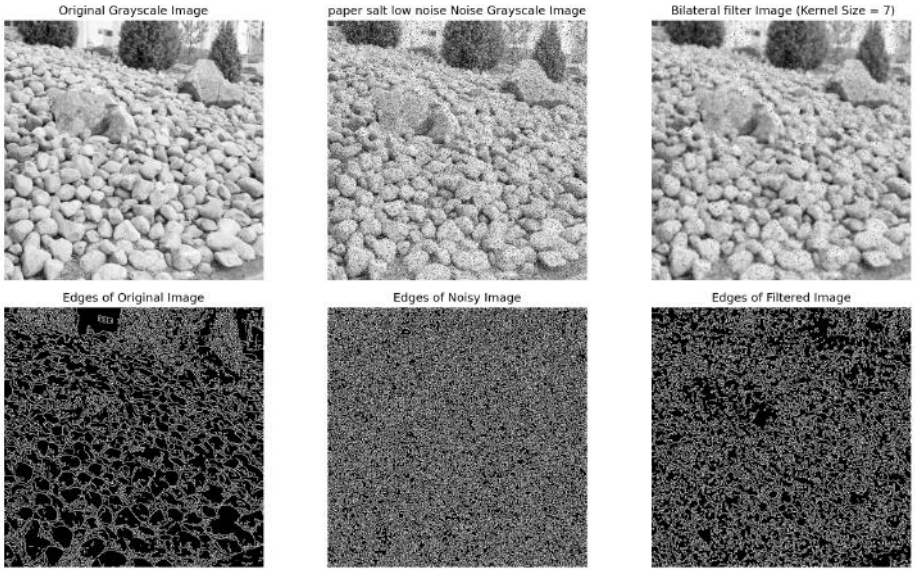
Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
3	57.2645	30.5519	0.0000	0.3510	0.1175	0.1155
5	62.6232	30.1634	0.0041	0.3301	0.1175	0.1018
7	68.8940	29.74899	0.0041	0.3148	0.1175	0.0817



((a)) Kernel 3



((b)) Kernel 5



((c)) Kernel 7

Figure 44: Application of Canny edge detector with varying kernel sizes: 3, 5, and 7.

3.6.7 Summary of Results

Table 44: Summary of Performance Metrics for Noise Types and Levels

Noise Type & Level	Kernel Size	MSE	PSNR (dB)	Filtering Time (s)	Edge Density - Filtered	SSIM (Original vs Noisy)	SSIM (Original vs Filtered)
Gaussian Noise - Low Level	3	43.2225	31.7737	0.0111	0.0076	0.2522	0.9782
	5	25.7917	34.0160	0.0039	0.0061	0.2522	0.9813
	7	18.6321	35.4282	0.0030	0.0055	0.2522	0.9777
Gaussian Noise - Medium Level	3	78.4473	29.1850	0.0185	0.2645	0.1347	0.2105
	5	68.8467	29.7520	0.0040	0.1057	0.1347	0.6296
	7	61.5104	30.2413	0.0030	0.0364	0.1347	0.5561
Gaussian Noise - High Level	3	99.4321	28.1555	0.0055	0.3738	0.0373	0.0428
	5	62.6232	28.3496	0.0040	0.3301	0.0373	0.0554
	7	68.8940	29.74899	0.0041	0.3148	0.0373	0.0817
Salt-and-Pepper Noise - Low Level	3	27.4256	33.7492	0.0025	0.1795	0.2389	0.2252
	5	35.8845	32.5817	0.0047	0.1383	0.2389	0.2118
	7	43.2004	31.7759	0.0035	0.0384	0.2389	0.4456
Salt-and-Pepper Noise - Medium Level	3	24.4859	34.2416	0.0030	0.2492	0.0725	0.0841
	5	24.3354	34.2684	0.0040	0.1827	0.0725	0.17996
	7	28.0820	33.6465	0.0036	0.1156	0.0725	0.3983
Salt-and-Pepper Noise - High Level	3	57.2645	30.5519	0.0000	0.3510	0.1175	0.1155
	5	62.6232	30.1634	0.0041	0.3301	0.1175	0.1018
	7	68.8940	29.74899	0.0041	0.3148	0.1175	0.0817

4 Discussion

The assignment aimed to compare image filters, from basic to advanced, in reducing noise. Filters tested include the Box, Gaussian, Median, Adaptive Mean, Adaptive Median, and Bilateral filters. Evaluation focused on Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) for Gaussian and Salt-and-Pepper noise at varying intensities (Low, Medium, High).

4.1 Noise Removal Effectiveness

4.1.1 Box filter

1. **Gaussian Noise**:
Low Level: Increasing the kernel size caused more blurring for the image which caused loss in details and edges. Despite Increasing the kernel size which reduced MSE, and increased the PSNR from 33 to 35, small kernel size (k = 3) is the best one since it performed good in removing noise and preserving edges. For computational time is remain zero for all kernels, which proves the simplicity of this filter.
Medium Level: While still showing some noise reduction, the blurring effect becomes more huge, resulting in a noticeable decrease in SSIM and edge density which dropped till 0.0027 compared to 0.0072 for the original image, and high loss in image details and edges. For computational time it remains zero.
High Level: The Box Filter struggles significantly to remove high-level Gaussian noise. Larger kernels do not significantly improve performance, as the blurring effect dominates, leading to a drastic decrease in image quality. We can see that in edge density which dropped to 0.009 compared to 0.132 for the original image, with poor similarity reached in it's best value to 0.46.
2. **Salt-and-Pepper Noise**:

Low Level: The Box Filter shows mixed results. While some noise removal occurs, the blurring effect significantly affects image clarity and edge detail. The filter struggles to maintain original edge density, also didn't remove all the noise, and that can be noticed from SSIM measure which equals 0.45.

Medium Level: The Box Filter demonstrates slightly better performance compared to low-level noise. Removed more noise but also caused huge blurring which led to huge loss in details and edges.

High Level: The Box Filter is ineffective in removing high-level salt-and-pepper noise. It fails to reduce noise significantly, leading to a dramatic decrease in image quality and edge clarity. Larger kernels worsen the situation due to excessive blurring. We can see that from edge preservation which reached 0.005 compared 0.09 for the original image.
In summary: The Box Filter shows better performance with Gaussian noise compared to salt-and-pepper noise. The filter generally struggles with high-level noise, regardless of type, resulting in a significant loss of image detail and edge information. While larger kernels improve noise reduction in some cases, they also increased the blurring effect, causing loss in edge preservation.

4.1.2 Gaussian filter

1. Gaussian Noise:

Low Level: The Gaussian filter performs exceptionally well, significantly reducing noise while preserving edge detail. Larger kernel sizes generally result in better PSNR and lower MSE, indicating superior noise reduction. The filter maintains a high SSIM value, confirming its effectiveness in preserving the original image features.

Medium Level: The Gaussian filter continues to show strong performance, effectively reducing noise with minimal blurring. Larger kernel sizes still contribute to improved noise removal, as reflected by the lower MSE and higher PSNR values. The filter also demonstrates excellent edge preservation, indicated by the high SSIM values and consistent edge density.

High Level: Even with high-level Gaussian noise, the filter demonstrates impressive results, effectively reducing noise but caused loss in edge detail. Larger kernel sizes continue to improve noise reduction and maintain a high SSIM despite it reduced a little (0.97) compared to smaller kernel size. But the negative impact is related to edge preservation where clearly we can see that the filter smoothed the image too much which reults in edge loss.

For computational time, it's still low, it reache 0.004 in it's maximum case, which still reasonable.

2. Salt-and-Pepper Noise:

Low Level: The Gaussian filter proves moderately effective in removing low-level salt-and-pepper noise. Larger kernel sizes generally lead to better PSNR and slightly higher SSIM values, indicating improved noise removal and a more significant resemblance to the original image. However, the filter exhibits a moderate decrease in edge density, suggesting some blurring and loosing huge details.

Medium Level: The Gaussian filter demonstrates better performance for medium-level salt-and-pepper noise compared to low-level noise. Larger kernel sizes generally lead to better results, achieving a higher SSIM and showing more significant noise reduction. Edge preservation remains a challenge, with a moderate decrease in edge density.

High Level: The Gaussian filter struggles to effectively remove high-level salt-and-pepper noise. While some noise reduction occurs, the filter exhibits significant blurring, as indicated by the decrease in edge density and SSIM. Larger kernel sizes generally worsen the situation, leading to excessive blurring and further degradation in image quality.

Also here for the computational time, it's higher than box filter but it's still low, where it reached 0.005 as maximum.

In summary:

The Gaussian filter excels in removing Gaussian noise, performing remarkably well across different levels. For salt-and-pepper noise, the filter exhibits moderate performance

at low levels, improving with increasing noise levels. However, it struggles to handle high-level salt-and-pepper noise effectively. The filter generally demonstrates a good balance between noise reduction and edge preservation, particularly for Gaussian noise. However, edge preservation remains a challenge for salt-and-pepper noise, especially at high levels.

4.1.3 Median filter

1. Gaussian Noise: Low Level: The Median filter shows really good performance, effectively reducing noise with minimal blurring. Larger kernel sizes (5 and 7) lead to improved PSNR and lower MSE, indicating superior noise reduction and better image quality. The filter maintains a high SSIM value, confirming its ability to preserve the original image features. Also with a reasonable computational time.

Medium Level: The Median filter continues to perform well, effectively reducing noise while preserving edge detail. Larger kernel sizes generally result in better PSNR and lower MSE, indicating improved noise removal. The filter also demonstrates excellent edge preservation, indicated by the high SSIM values and consistent edge density.

High Level: The Median filter proves effective in reducing high-level Gaussian noise, demonstrating better performance compared to the Box filter. Larger kernel sizes generally lead to better results, with a higher PSNR and SSIM. While some blurring is observed, the filter maintains a relatively high edge density, suggesting its ability to retain important image features.

2. Salt-and-Pepper Noise:

Low Level: The Median filter excels in removing low-level salt-and-pepper noise, demonstrating significantly better performance compared to other filters. It exhibits high PSNR and SSIM values while maintaining a high edge density, indicating effective noise removal with minimal blurring.

Medium Level: Actually the filter here showed a little bit weaker performance compared to lower level of noise, but still good. increasing kernel size caused more blurring which reduced noise, but caused a high loss in edges and details, where edge density reached 0.05 compared to 0.12 of the original image. For computational time still not bad, higher than box filter, but still reasonable since it showed better performance in edge preservation than box, and Gaussian filter.

High Level: The Median filter still exhibits a strong performance against high-level salt-and-pepper noise, significantly outperforming other filters. While some blurring is observed, the filter achieves a remarkably high PSNR and SSIM while maintaining a relatively high edge density.

In summary: The Median filter outperforms both the Box and Gaussian filters in removing salt-and-pepper noise, especially at higher levels. It demonstrates good performance with Gaussian noise, particularly for higher noise levels, showing a better balance between noise reduction and edge preservation compared to the Box filter.

The filter maintains a high edge density across various noise types and levels, proving its ability to preserve image features effectively.

4.1.4 Adaptive mean filter

1. Gaussian Noise:

Low Level: The Adaptive Mean Filter performs similarly to the Box Filter, showing good noise reduction. It achieves slightly higher PSNR values and lower MSE compared

to the Box filter, indicating better overall performance for low-level Gaussian noise. The filter maintains a high SSIM, suggesting good preservation of original image features. For computation time it showed a higher values than box filter but still a reasonable value (0.004).

Medium Level:Increasing kernel size caused huge blurring which results in a huge loss in edges and details, also struggle to handle such type of noise compared to median filter which showed a really good performance in saving details and removing noise. For computational time still good, where it reached 0.007 in it maximum case.

High Level: The filter struggles with high-level Gaussian noise, exhibiting comparable performance to the Box filter. While some noise reduction occurs, the blurring effect becomes noticeable, leading to a noticeable decrease in SSIM and a slight decrease in edge density. Gaussian filter showed much better performance in such case.

2. Salt-and-Pepper Noise:

Low Level: The Adaptive Mean Filter shows similar performance to the Box Filter, demonstrating moderate success in removing low-level salt-and-pepper noise. While some noise reduction occurs, the filter exhibits blurring and a moderate decrease in edge density. But not suitable for such noise type since the noise is still clear.

Medium Level: The Adaptive Mean Filter demonstrates better performance compared to the Box Filter, especially for larger kernel sizes. It achieves a higher SSIM and shows a more significant reduction in salt-and-pepper noise. While some blurring is present, the filter struggles save a high edge density which in higer kernel size, edge density reached to 0.006 where the original image has 0.01. For computation time still good where it reached 0.007 for kernel size = 7.

High Level: The Adaptive Mean Filter struggles to handle high-level salt-and-pepper noise effectively. Like the Box Filter, it exhibits blurring and a notable decrease in SSIM and edge density, indicating limited success in noise removal and edge preservation.

In summary:

The Adaptive Mean Filter generally performs slightly better than the Box Filter for removing Gaussian noise, especially at medium levels, demonstrating more effective noise reduction with less blurring.
For salt-and-pepper noise, the Adaptive Mean Filter shows improvements compared to the Box Filter, particularly for medium-level noise, achieving better noise removal and edge preservation.
Both filters struggle with high-level salt-and-pepper noise, resulting in significant blurring and decreased image quality.
And for computational time reached 0.009 in worst case, which is till reasonable.

4.1.5 Adaptive median filter

1. Gaussian Noise:

Low Level: The Adaptive Median Filter shows mixed results for low-level Gaussian noise. While it achieves a reasonable PSNR value, the SSIM is significantly lower com-

pared to other filters. It showed a good performance in preserving details but a weak performance in removing noise.

Medium Level: The filter exhibits poor performance for medium-level Gaussian noise. The PSNR is lower, the SSIM is significantly low, and the MSE is too high (88) showing weak noise reduction and some blurring, with a substantial loss of image quality.

High Level: The filter performs poorly with high-level Gaussian noise, achieving the lowest PSNR value across all filters and showing a significant decrease in SSIM. This indicates limited success in noise removal and a presence of significant blurring. Being the worst kernel for such noise type and level till now. For computational type, Huge Difference appeared where the time ranged between 2.9 to 3.1 second.

2. Salt-and-Pepper Noise:

Low Level: The Adaptive Median Filter shows strong performance for low-level salt-and-pepper noise. with moderate noise removal,since the filter didn't remove all the noises with minimal blurring and good preservation of image details. But when compared to Median filter, media filter is so much better in noise removal. For computational time, it's much worse than median filter, it reached 3.1 seconds compared to 0.03 for median filter.

Medium Level: The filter continues to perform well with medium-level salt-and-pepper noise, keeping a good image quality and reversing edges with no blurring, achieving a high PSNR and a decent SSIM. But has a small issue with noise removal, from the image we can see clearly some of the noise still exists after applying the filter.
High Level: The same as medium level, keeping image details and edges, but struggles to remove noise, Median filter showed petter performance in removing noise and preserving edges. For computational time, still high, it took the filter 3.3 seconds to process the image.
In summary moderate noise reduction while maintaining a good level of image detail. When comparing it to Median filter, median filter blurred the image more causing loss in details but removed noise successfully. For computational time, it needed 2.5 second to process the image.

4.1.6 Bilateral filter

1. Gaussian Noise:

Low Level: The Bilateral filter performs well for low-level Gaussian noise, showing good noise reduction while maintaining edge sharpness. The filter achieves a good balance between noise reduction and edge preservation, evident in its high PSNR and SSIM values. Compared to simpler filters like Box, or Gaussian filter, they perform slightly better in noise removal, and need less time compared to Bilateral which need 0.01 second to process the image.

Medium Level: The filter's effectiveness decreases with medium-level Gaussian noise. It still demonstrates some noise reduction but experiences blurring, leading to a moderate decrease in SSIM. The filter struggles to effectively remove noise at this level while maintaining edge sharpness. comparing to Gaussian filter, Gaussian performs much better in

noise removal. But both of them struggles when increasing the kernel size which lead to blurring and loss of details.

High Level: The Bilateral filter performs poorly with high-level Gaussian noise. The filter struggles to remove the noise effectively, resulting in a significant drop in PSNR and SSIM values and substantial blurring. It shows a dramatic decrease in edge density, indicating a significant loss of image detail. For computational time, it range between 0.001 to 0.004.

2. Salt-and-Pepper Noise:
Low Level: The Bilateral filter exhibits mixed performance for low-level salt-and-pepper noise. While it demonstrates some noise reduction, the filter experiences significant blurring, leading to a moderate decrease in SSIM and edge density. Compared to other complex filters like Median or Adaptive median, other complex performs better in noise removal and edge preservation.

Medium Level: The Bilateral filter demonstrates limited success with medium-level salt-and-pepper noise. The filter struggles to effectively remove noise, exhibiting significant blurring and a notable decrease in SSIM, indicating a loss of image detail and compromised edge preservation. Still much worse than Median and Adaptive median.

High Level: The Bilateral filter shows weak performance with high-level salt-and-pepper noise. It struggles to remove the noise effectively, leading to a noticeable decrease in PSNR and SSIM values and significant blurring. The filter also experiences a considerable reduction in edge density, highlighting its inability to preserve image details and sharp edges. For computational time, kinda normal where it ranges between 0.003 and 0.004 second.

In summary:
The Bilateral filter demonstrates mixed performance for low-level Gaussian noise but struggles with increasing noise levels. It exhibits limited effectiveness against salt-and-pepper noise, particularly at higher levels, failing to effectively remove noise and often leading to significant blurring. While it maintains edge sharpness with low-level Gaussian noise, it significantly struggles with preserving edges in the presence of higher noise levels.

5 Conclusion

This assignment explores the efficiency of Box, Gaussian, Median, Adaptive Mean, Adaptive Median, and Bilateral filters for noise removal. These filters are applied to test their efficiencies against Gaussian and salt-and-pepper noise at different intensities. The performance evaluation will be based on PSNR, SSIM, edge density, and computation time.

Overall, the Median filter turned out to be the best one for salt-and-pepper noise, especially at higher noise levels. It performed much better than the other filters, giving high PSNR and SSIM values and keeping the edges clear. On the other hand, for Gaussian noise, the Gaussian filter worked the best. It had a very great balance between

noise removal and saving of edge details. In this respect, this will be very suitable for the Gaussian noise removal.

Whereas the other filters had mixed results. In general, the Box and Adaptive Mean-filters did reasonably well with Gaussian, but they suffered with salt-and-pepper-noise-especially for high noise. The Adaptive Median filter had been less effective at smoothing out either type of noise, with a noticeably lower quality image showing strong blurring. The Bilateral filter worked moderately with low levels of Gaussian noise added but failed to do well as the noise levels went higher.
For computational time, both Box and Gaussian filters were among the fastest to compute. In the case of the Median filter, it showed a medium level of computational time while performing very strongly in most cases. The Adaptive Median and the Bilateral filters could reduce noise in some instances, but with lots of increased time for processing, thus perhaps an indication of a probable trade-off between performance and speed.
Recommendation
The following filter types should be recommended the most for different types of noises, according to the results of testing: **Salt-and-pepper noise**: The best is the Median filter. It is very good in noise removal, even for higher levels of noise; with only small blurring, it preserves edges very well. **Gaussian noise**: Gaussian filter. It reduces noise excellently and keeps all the image details, even when the level of noise rises to higher values. Whenever the computational time is very important, one could use the Box and Gaussian filters for Gaussian noise, too. Similarly, instead of the Box filter, one might use the Adaptive Mean filter since the improvement is decent and it takes around the same amount of processing time. Generally speaking, the conclusion of this assignment will be that the best filter choice strongly depends on the kind and amount of noise in an image.