Faculty of Engineering & Technology

Electrical & Computer Engineering Department

COMPUTER VISION– ENCS5343

Assignment1 Report

Prepared by:

Name: Mohammad Abu Shams **ID:** 1200549

Instructor: Dr. Aziz Qaroush

Section: 2

Date: 21-11-2024

BIRZEIT

List of Figures

Table of Contents

1. Introduction

Image denoising is an important task in computer vision that helps improve image quality by removing unwanted noise while keeping important details. This is very important for tasks like object detection, image classification, and image analysis, where noise can affect the performance. In this assignment, I test different denoising methods using black-and-white images of animals like cat, dog, and bird. These images have many different textures and fine details, which makes them good for testing how well different denoising methods work.

Noise in images can hide important information and make the image look bad. In this study, I focus on two common types of noise:

- **Gaussian Noise**: This noise changes pixel values randomly, making the image look grainy.
- **Salt-and-Pepper Noise**: This noise adds random black and white pixels to the image, making it hard to see fine details.

To reduce this noise, I test four different filters, each with its own way of removing noise:

- **Box Filter**: This filter smooths the image by averaging the pixel values around each pixel. It works for noise removal but can make edges blurry.
- **Gaussian Filter**: This filter gives more weight to the center pixels in a neighborhood, which helps keep edges clearer than the box filter.
- **Median Filter**: This filter replaces each pixel with the median value of its neighbors, which is really good for removing salt-and-pepper noise while keeping edges intact.
- **Adaptive Filters**: These filters change based on the local image details, giving better noise removal and edge preservation, but they usually take more time to process.

The goal of this assignment is to test how well these filters work under different types of noise. I also want to see how the kernel size affects each filter's ability to remove noise, keep edges, and process the image. I will evaluate the filters using some performance measures like Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), edge preservation, and how much time it takes to process the image. The results from this study will help choose the best denoising methods for different types of images and noise.

2. Experiments

The goal of this experiment was to test and compare how different image denoising filters work under different noise levels and kernel sizes. The steps below show how I created noisy images, applied the filters, and checked their performance.

2.1 Generate or Load Noisy Images

- **Select Images**: I used a set of clean animal images, like cat, dog, and bird, for testing. These images were chosen because they have different textures and details, which helps test the filters.

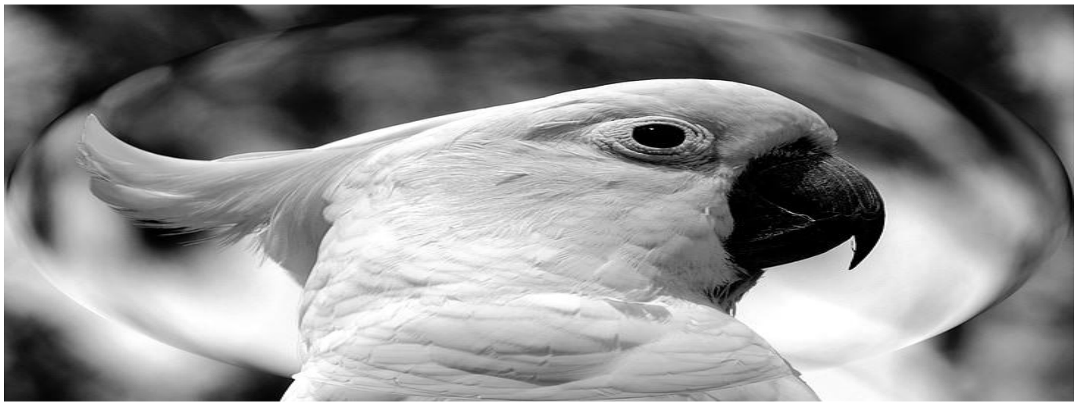


Figure 1: Original Bird Image

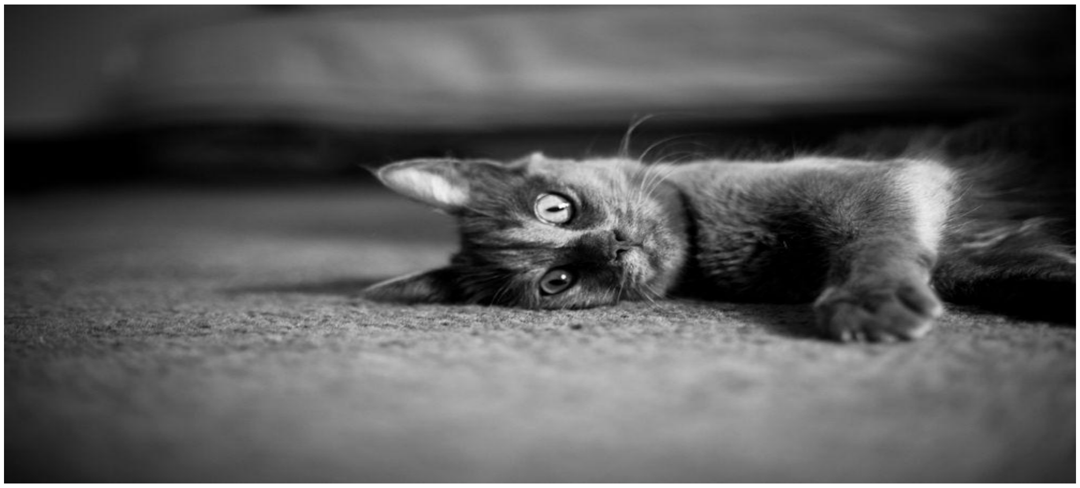


Figure 2: Original Cat Image

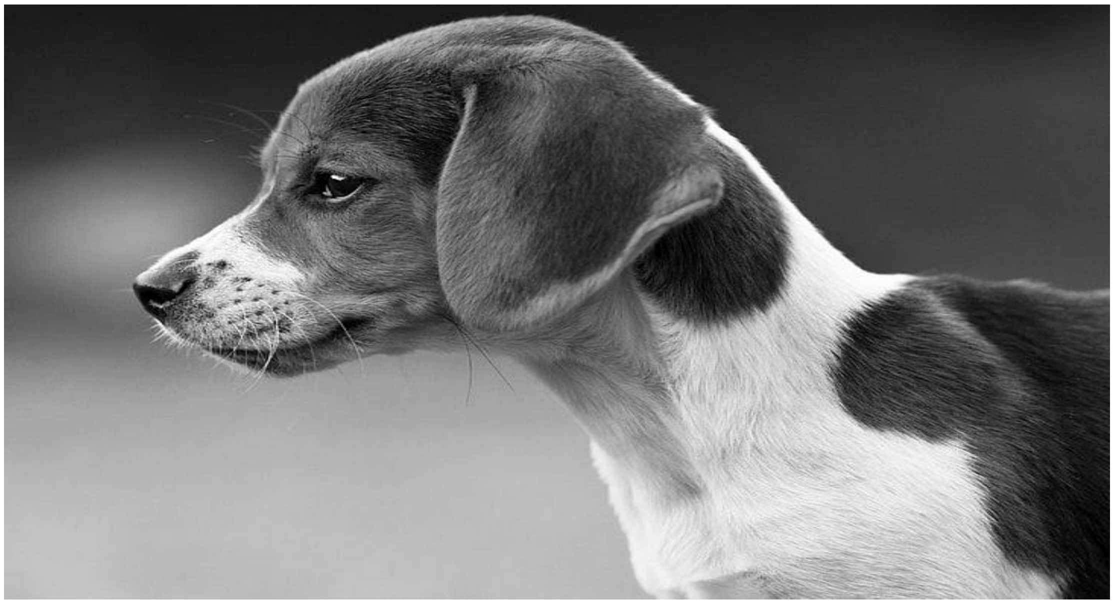


Figure 3: Original Dog Image

- **Add Noise**: I added different types of noise to the clean images to simulate real-world noise:
    - **Gaussian Noise**: This noise randomly changes pixel values, making the image look grainy.
    - **Salt-and-Pepper Noise**: This noise adds random black and white pixels, creating small "pepper" and "salt" spots that hide image details.
      I added the noise at three different levels:
    - Low Intensity (0.05).
    - Medium Intensity (0.1).
    - High Intensity (0.2).

2.2 Apply Filters

After creating the noisy images, I applied different filters to remove the noise. The filters tested were simple and advanced filters, each designed to reduce noise and keep image quality.

- **Simple Filters**: These are basic filters used for smoothing and noise reduction.
  - **Box Filter**: This filter averages pixel values in a small square around each pixel. It smooths the image but can blur the edges.
  - **Gaussian Filter**: Similar to the box filter, but it gives more weight to the center of the area, which helps keep edges better than the box filter.
  - **Median Filter**: This filter replaces each pixel with the middle value of its neighbors. It's good for removing salt-and-pepper noise while keeping edges sharp.

- **Advanced Filters**: These filters change based on the image details and work well, but they need more time to process.
  - **Adaptive Mean Filter**: This filter changes the kernel size based on the image's local statistics. It removes noise while keeping edges.
  - **Adaptive Median Filter**: Similar to the adaptive mean filter, but it uses the median value to filter the noise.
  - **Bilateral Filter**: This filter smooths the image by looking at both the distance and the difference in pixel values. It helps keep edges better than the Gaussian filter.

I applied each filter to the noisy images using different kernel sizes:

- 3, 7, 11, and 15 pixels.

2.3 Measure Performance

I measured how well each filter worked using a few methods:

- **MSE and PSNR**:
  - **Mean Squared Error (MSE)**: This measures the difference between the clean image and the filtered image. A lower MSE means the noise was removed better.
  - **Peak Signal-to-Noise Ratio (PSNR)**: PSNR compares the maximum power of the image to the noise power. A higher PSNR means better image quality after filtering.
- **Edge Preservation**:
  To check how well the filters kept the edges, I used Canny edge detection on both the clean and filtered images. I compared the edges in the noisy image and the filtered image. The difference between these edges was used to measure how well the filter kept the image's edges.
- **Computational Time**:
  I measured the time each filter took to process the images. This helps show how efficient the filters are, especially with larger kernels and advanced filters.
- **Kernel Size Effect**:
  I looked at how changing the kernel size affected the filter's performance. Larger kernels usually remove noise better but can blur edges. Smaller kernels keep more details but may not remove noise as well. I compared the results to understand the trade-offs between noise removal, edge preservation, and processing time.

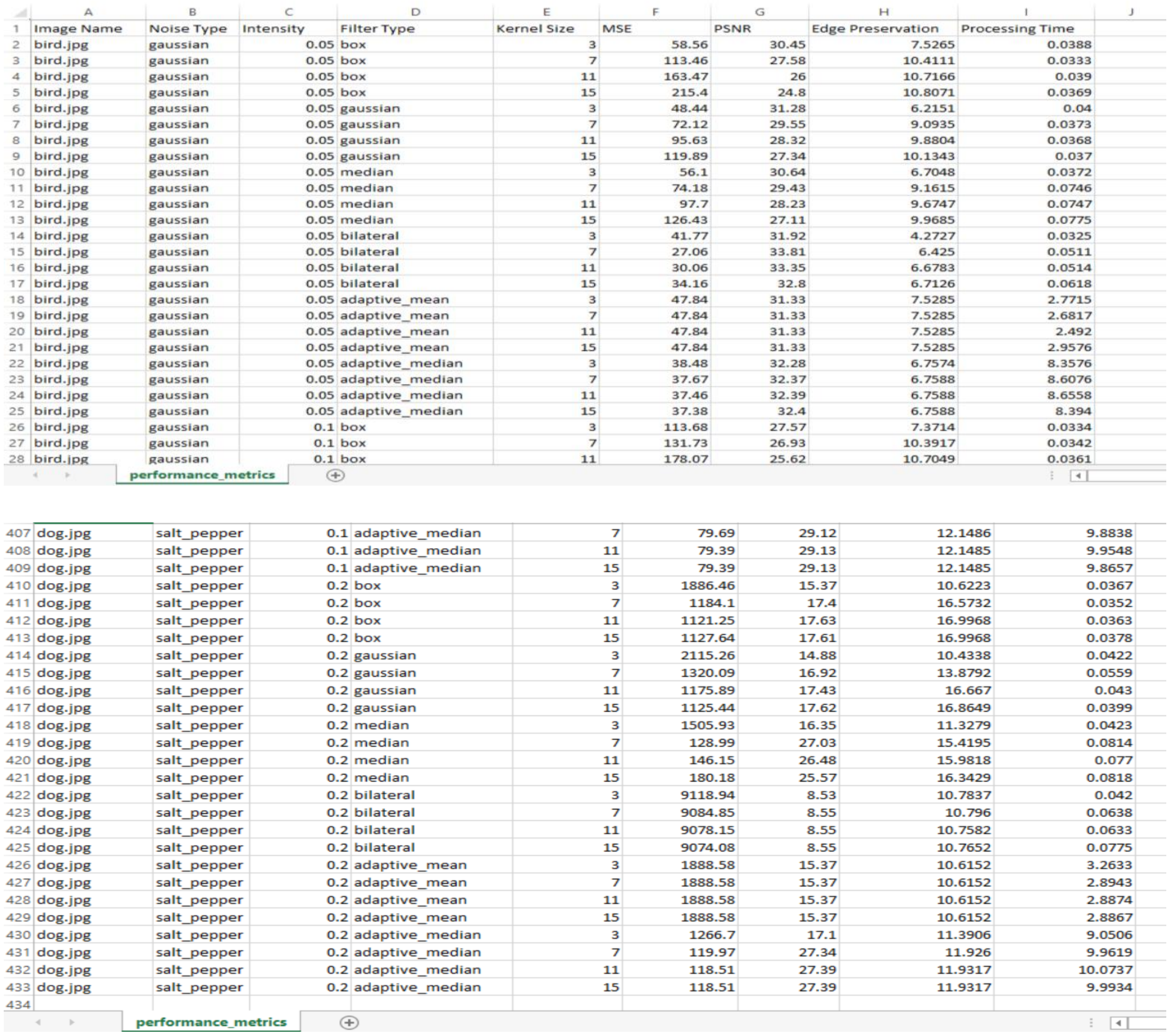
3. Results

3.1 Filters Results

First, the output consists of 450 images (432 filtered images and 18 noisy images), along with a CSV file containing all the results for these images. They are attached in the Google Drive link below:

https://drive.google.com/drive/folders/1CtCjvtdBmUTygRp4p0Pj1C4UDs7_vlP3?usp=drive_link

Below is the some performance metrics:



	A	B	C	D	E	F	G	H	I	J
1	Image Name	Noise Type	Intensity	Filter Type	Kernel Size	MSE	PSNR	Edge Preservation	Processing Time	
2	bird.jpg	gaussian	0.05	box	3	58.56	30.45	7.5265	0.0388	
3	bird.jpg	gaussian	0.05	box	7	113.46	27.58	10.4111	0.0333	
4	bird.jpg	gaussian	0.05	box	11	163.47	26	10.7166	0.039	
5	bird.jpg	gaussian	0.05	box	15	215.4	24.8	10.8071	0.0369	
6	bird.jpg	gaussian	0.05	gaussian	3	48.44	31.28	6.2151	0.04	
7	bird.jpg	gaussian	0.05	gaussian	7	72.12	29.55	9.0935	0.0373	
8	bird.jpg	gaussian	0.05	gaussian	11	95.63	28.32	9.8804	0.0368	
9	bird.jpg	gaussian	0.05	gaussian	15	119.89	27.34	10.1343	0.037	
10	bird.jpg	gaussian	0.05	median	3	56.1	30.64	6.7048	0.0372	
11	bird.jpg	gaussian	0.05	median	7	74.18	29.43	9.1615	0.0746	
12	bird.jpg	gaussian	0.05	median	11	97.7	28.23	9.6747	0.0747	
13	bird.jpg	gaussian	0.05	median	15	126.43	27.11	9.9685	0.0775	
14	bird.jpg	gaussian	0.05	bilateral	3	41.77	31.92	4.2727	0.0325	
15	bird.jpg	gaussian	0.05	bilateral	7	27.06	33.81	6.425	0.0511	
16	bird.jpg	gaussian	0.05	bilateral	11	30.06	33.35	6.6783	0.0514	
17	bird.jpg	gaussian	0.05	bilateral	15	34.16	32.8	6.7126	0.0618	
18	bird.jpg	gaussian	0.05	adaptive_mean	3	47.84	31.33	7.5285	2.7715	
19	bird.jpg	gaussian	0.05	adaptive_mean	7	47.84	31.33	7.5285	2.6817	
20	bird.jpg	gaussian	0.05	adaptive_mean	11	47.84	31.33	7.5285	2.492	
21	bird.jpg	gaussian	0.05	adaptive_mean	15	47.84	31.33	7.5285	2.9576	
22	bird.jpg	gaussian	0.05	adaptive_median	3	38.48	32.28	6.7574	8.3576	
23	bird.jpg	gaussian	0.05	adaptive_median	7	37.67	32.37	6.7588	8.6076	
24	bird.jpg	gaussian	0.05	adaptive_median	11	37.46	32.39	6.7588	8.6558	
25	bird.jpg	gaussian	0.05	adaptive_median	15	37.38	32.4	6.7588	8.394	
26	bird.jpg	gaussian	0.1	box	3	113.68	27.57	7.3714	0.0334	
27	bird.jpg	gaussian	0.1	box	7	131.73	26.93	10.3917	0.0342	
28	bird.jpg	gaussian	0.1	box	11	178.07	25.62	10.7049	0.0361	

performance_metrics

407	dog.jpg	salt_pepper	0.1	adaptive_median	7	79.69	29.12	12.1486	9.8838
408	dog.jpg	salt_pepper	0.1	adaptive_median	11	79.39	29.13	12.1485	9.9548
409	dog.jpg	salt_pepper	0.1	adaptive_median	15	79.39	29.13	12.1485	9.8657
410	dog.jpg	salt_pepper	0.2	box	3	1886.46	15.37	10.6223	0.0367
411	dog.jpg	salt_pepper	0.2	box	7	1184.1	17.4	16.5732	0.0352
412	dog.jpg	salt_pepper	0.2	box	11	1121.25	17.63	16.9968	0.0363
413	dog.jpg	salt_pepper	0.2	box	15	1127.64	17.61	16.9968	0.0378
414	dog.jpg	salt_pepper	0.2	gaussian	3	2115.26	14.88	10.4338	0.0422
415	dog.jpg	salt_pepper	0.2	gaussian	7	1320.09	16.92	13.8792	0.0559
416	dog.jpg	salt_pepper	0.2	gaussian	11	1175.89	17.43	16.667	0.043
417	dog.jpg	salt_pepper	0.2	gaussian	15	1125.44	17.62	16.8649	0.0399
418	dog.jpg	salt_pepper	0.2	median	3	1505.93	16.35	11.3279	0.0423
419	dog.jpg	salt_pepper	0.2	median	7	128.99	27.03	15.4195	0.0814
420	dog.jpg	salt_pepper	0.2	median	11	146.15	26.48	15.9818	0.077
421	dog.jpg	salt_pepper	0.2	median	15	180.18	25.57	16.3429	0.0818
422	dog.jpg	salt_pepper	0.2	bilateral	3	9118.94	8.53	10.7837	0.042
423	dog.jpg	salt_pepper	0.2	bilateral	7	9084.85	8.55	10.796	0.0638
424	dog.jpg	salt_pepper	0.2	bilateral	11	9078.15	8.55	10.7582	0.0633
425	dog.jpg	salt_pepper	0.2	bilateral	15	9074.08	8.55	10.7652	0.0775
426	dog.jpg	salt_pepper	0.2	adaptive_mean	3	1888.58	15.37	10.6152	3.2633
427	dog.jpg	salt_pepper	0.2	adaptive_mean	7	1888.58	15.37	10.6152	2.8943
428	dog.jpg	salt_pepper	0.2	adaptive_mean	11	1888.58	15.37	10.6152	2.8874
429	dog.jpg	salt_pepper	0.2	adaptive_mean	15	1888.58	15.37	10.6152	2.8867
430	dog.jpg	salt_pepper	0.2	adaptive_median	3	1266.7	17.1	11.3906	9.0506
431	dog.jpg	salt_pepper	0.2	adaptive_median	7	119.97	27.34	11.926	9.9619
432	dog.jpg	salt_pepper	0.2	adaptive_median	11	118.51	27.39	11.9317	10.0737
433	dog.jpg	salt_pepper	0.2	adaptive_median	15	118.51	27.39	11.9317	9.9934
434									

performance_metrics

Figure 4: Some performance metrics in the csv file

Below is the some filtered images:

The image below shows the "Cat" image with Salt-and-Pepper noise (intensity = 0.1) applied. The Gaussian filter (kernel size = 3) is used to reduce the noise.

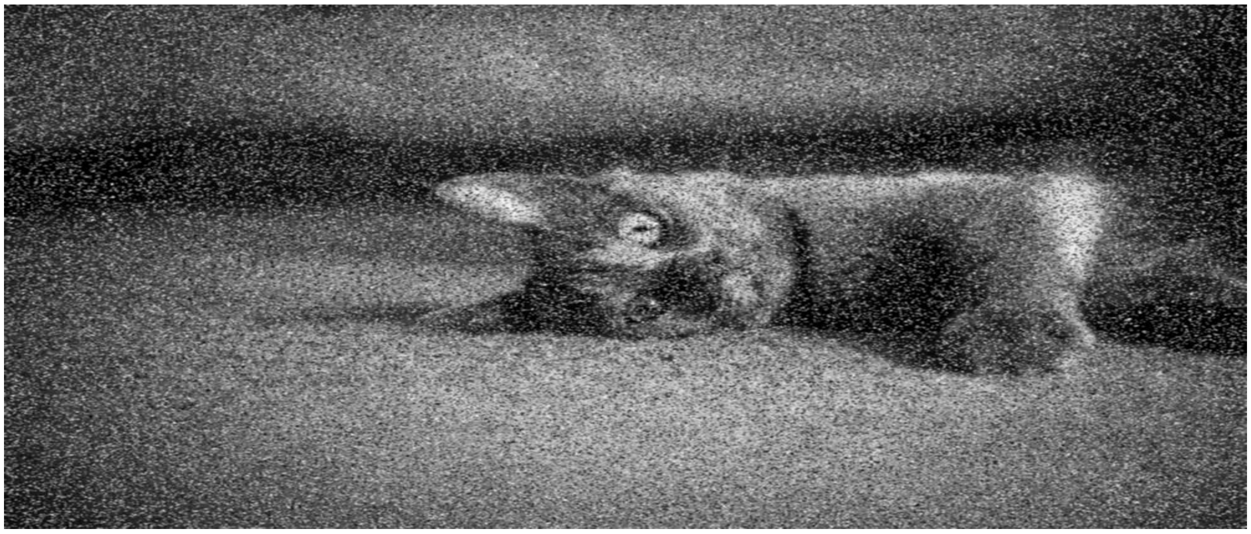


Figure 5: Noisy Image with Gaussian Filter (k=3, Intensity=0.1, Salt-and-Pepper Noise) Applied to 'Cat' Image

The image below shows the "Bird" image with Gaussian noise (intensity = 0.05) applied. The Median filter (kernel size = 7) is used to reduce the noise.

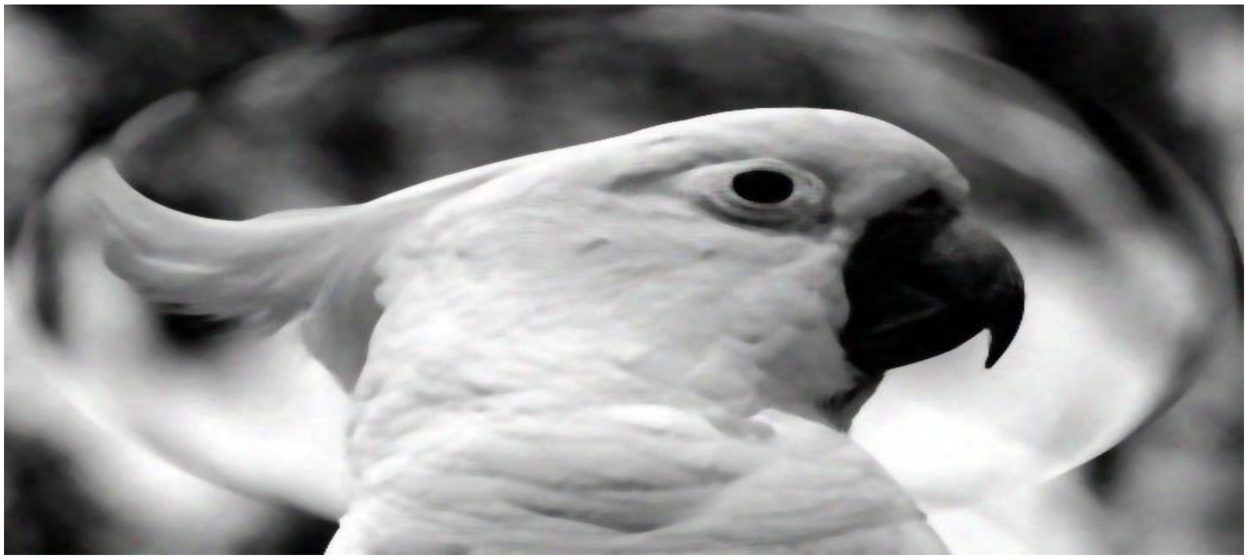


Figure 6: Noisy Image with Median Filter (k=7, Intensity=0.05, Gaussian Noise) Applied to 'Bird' Image

The image below shows the "Dog" image with Gaussian noise (intensity = 0.2) applied. The Box filter (kernel size = 11) is used to reduce the noise.

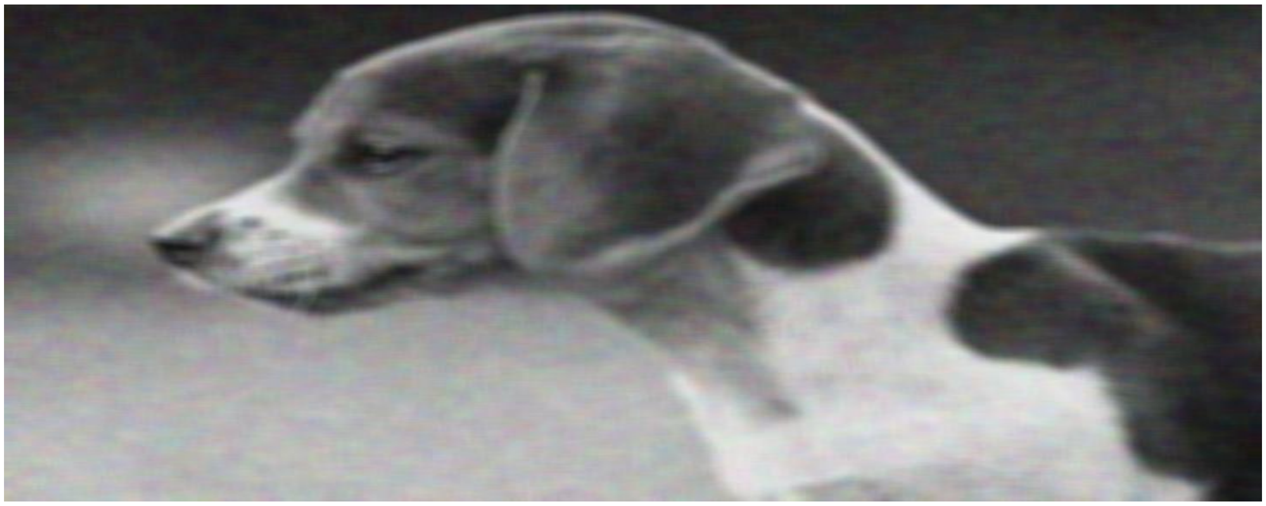


Figure 7: Noisy Image with Box Filter (k=11, Gaussian Noise Intensity=0.2) Applied to 'Dog' Image

The image below shows the "Dog" image with Salt-and-Pepper noise (intensity = 0.05). The Adaptive mean filter (kernel size = 15) is used to reduce the noise.

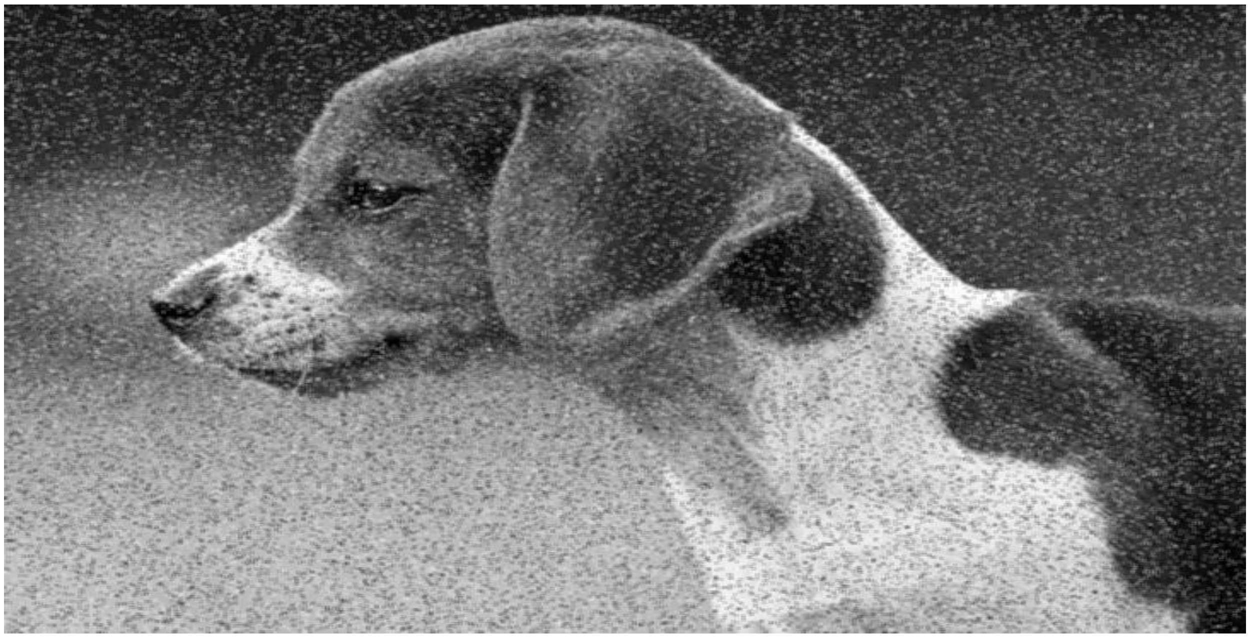


Figure 8: Noisy Image with Adaptive mean Filter (k=15, Salt-and-Pepper Noise Intensity=0.05) Applied to 'Dog' Image

The image below shows the "Bird" image with Gaussian noise (intensity = 0.2). The Adaptive median filter (kernel size = 15) is used to reduce the noise.

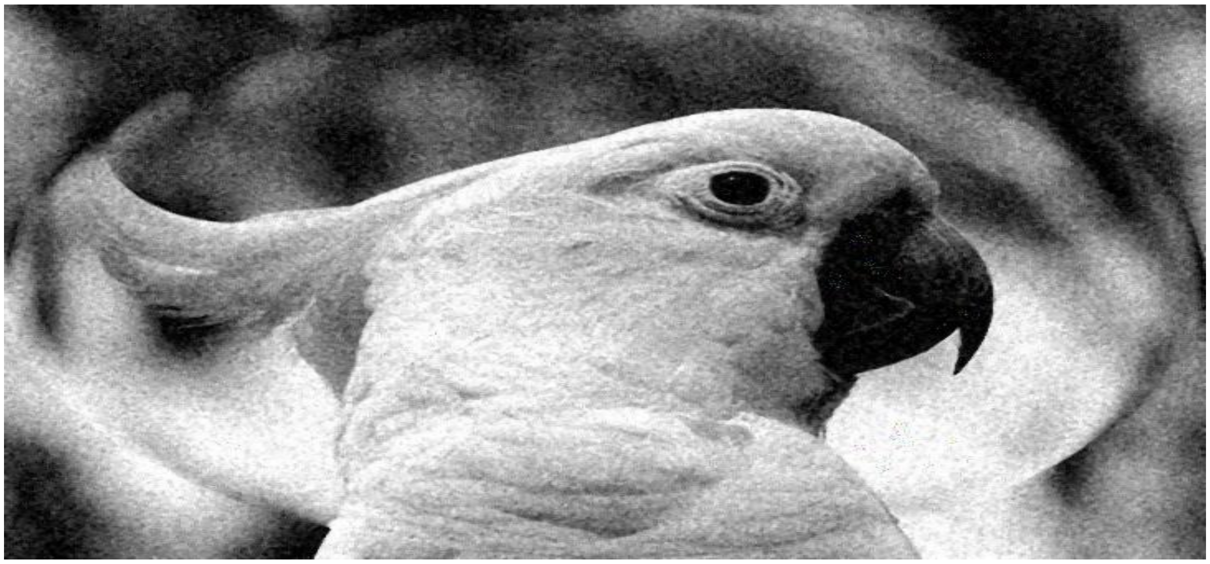


Figure 9: Noisy Image with Adaptive median Filter (k=15, Gaussian Noise Intensity=0.2) Applied to 'Bird' Image

The image below shows the "Cat" image with Gaussian noise (intensity = 0.1). The Bilateral filter (kernel size = 7) is used to reduce the noise.

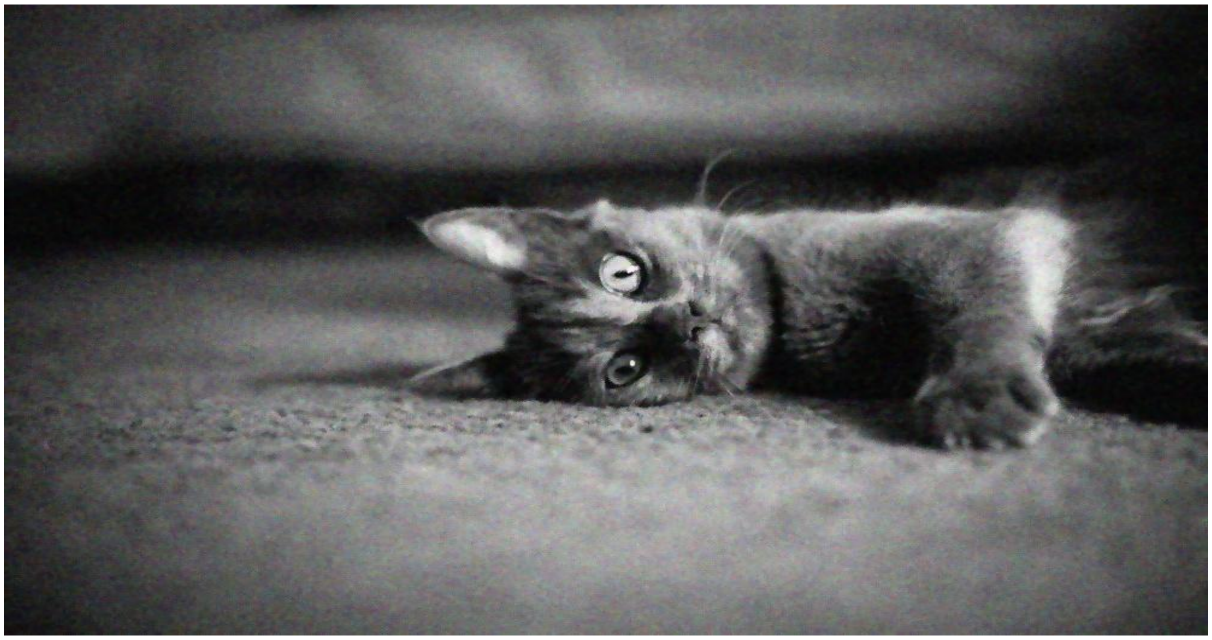


Figure 10: Noisy Image with Bilateral Filter (k=7, Gaussian Noise Intensity=0.1) Applied to 'Cat' Image

3.2 Noisy Results

There is Noisy Image Examples with Gaussian and Salt-Pepper Noise at Different Intensities:

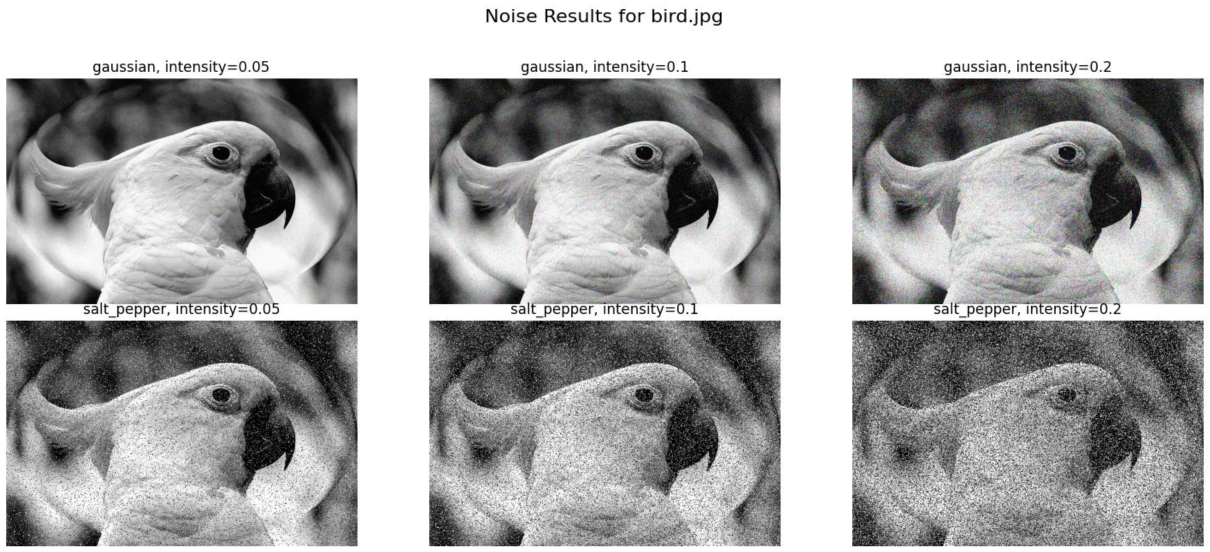

Noise Results for bird.jpg
gaussian, intensity=0.05
gaussian, intensity=0.1
gaussian, intensity=0.2
salt_pepper, intensity=0.05
salt_pepper, intensity=0.1
salt_pepper, intensity=0.2

Figure 11: Noise Results for bird

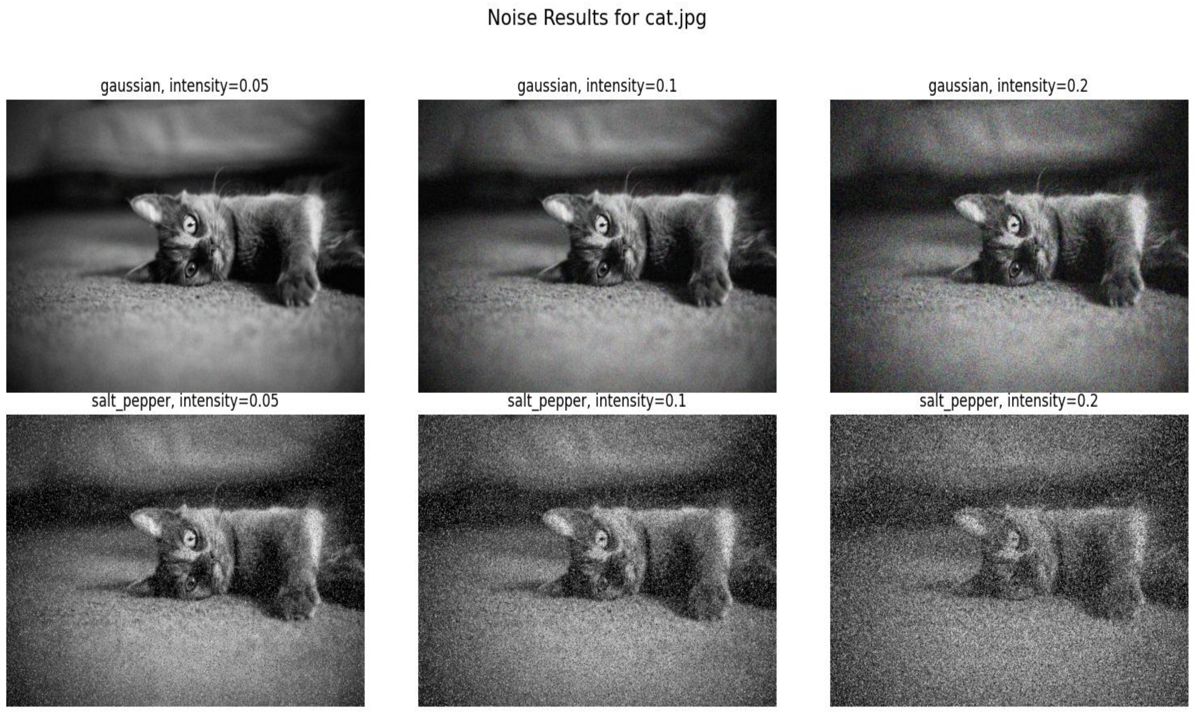

Noise Results for cat.jpg
gaussian, intensity=0.05
gaussian, intensity=0.1
gaussian, intensity=0.2
salt_pepper, intensity=0.05
salt_pepper, intensity=0.1
salt_pepper, intensity=0.2

Figure 12: Noise Results for cat

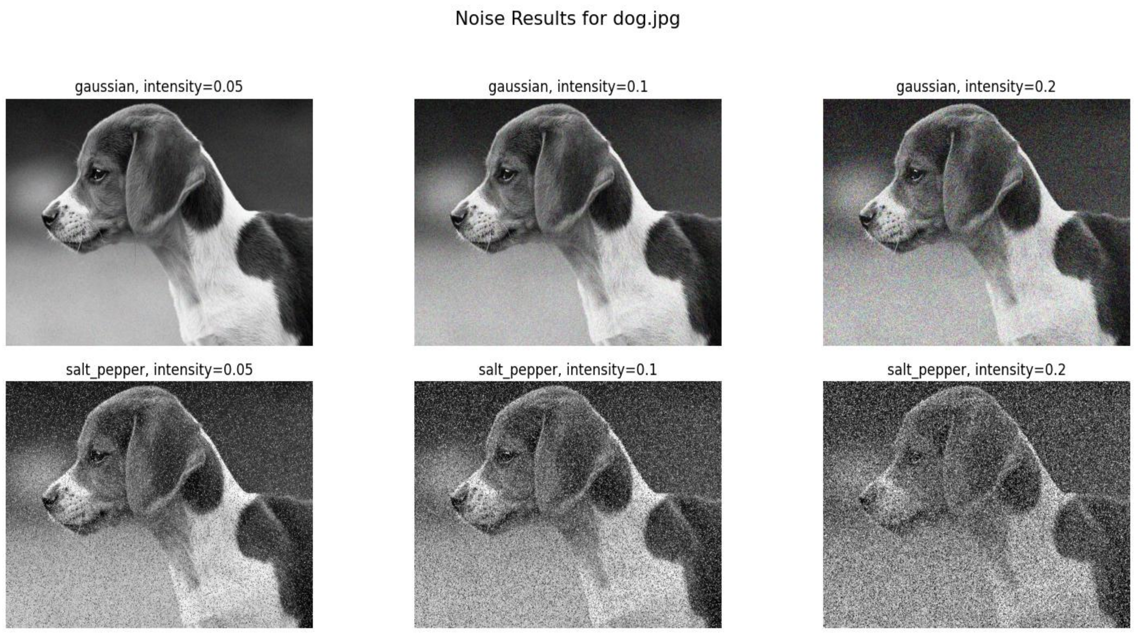

Noise Results for dog.jpg
gaussian, intensity=0.05
gaussian, intensity=0.1
gaussian, intensity=0.2
salt_pepper, intensity=0.05
salt_pepper, intensity=0.1
salt_pepper, intensity=0.2

Figure 13: Noise Results for dog

3.3 MSE and PSNR Comparison

The following results in the below, compare how well different filters work on noisy images in terms of noise removal (MSE), image quality (PSNR), and the effect of kernel size.

MSE Comparison: The plot shows the Mean Squared Error (MSE) for different filters with different kernel sizes and noise levels.

PSNR Comparison: The Peak Signal-to-Noise Ratio (PSNR) values are shown for each filter type and kernel size.

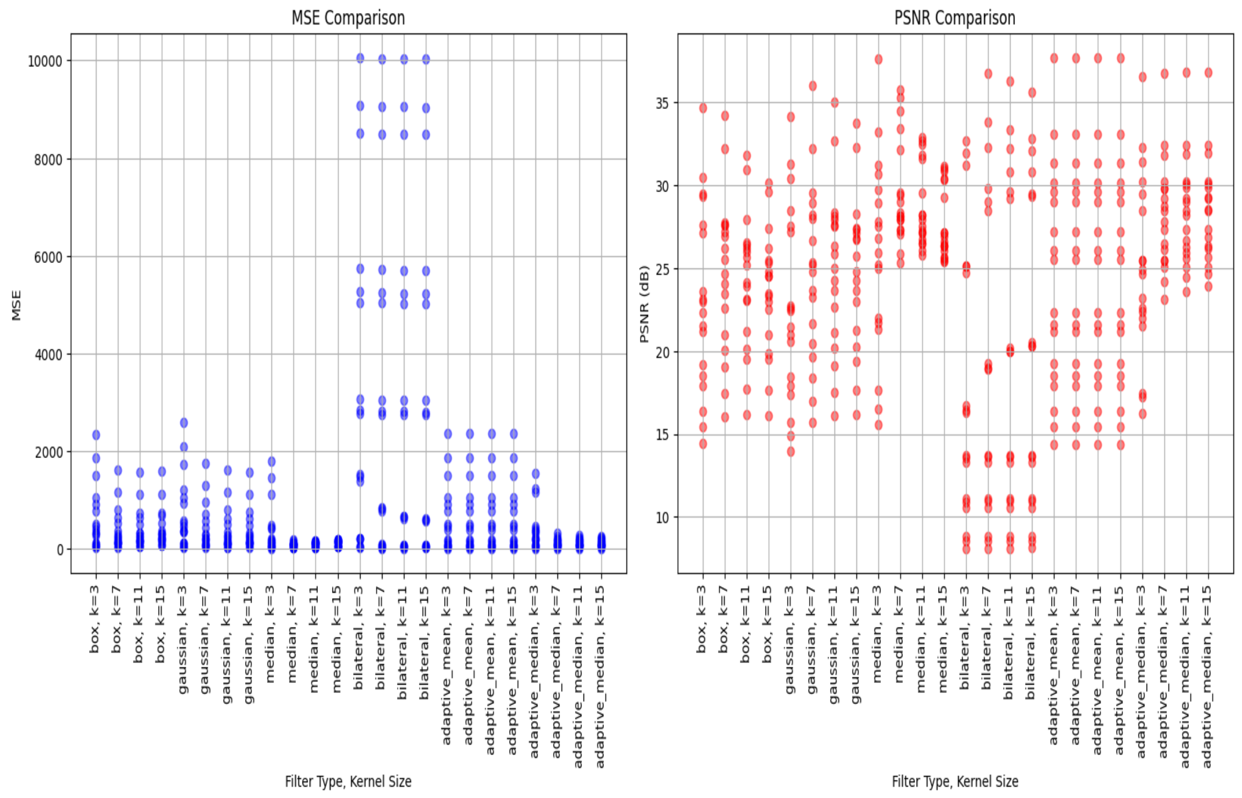


Figure 14: MSE and PSNR Comparison

3.4 Edge Preservation Comparison

The figures below show how different filters keep the edges of images while reducing noise. Each image shows edge maps made by using different filters (Box, Gaussian, Median, Bilateral, Adaptive Mean, and Adaptive Median) on noisy images. The noise was added as Gaussian and Salt-and-Pepper at different levels (0.05, 0.1, and 0.2). These edge maps help to see how well each filter keeps the image edges while getting rid of the noise and also show how changing the kernel size affects the results.

The figure below shows the edge preservation maps for the bird image, showing the impact of various filters and kernel sizes on edge detection, with different noise levels.

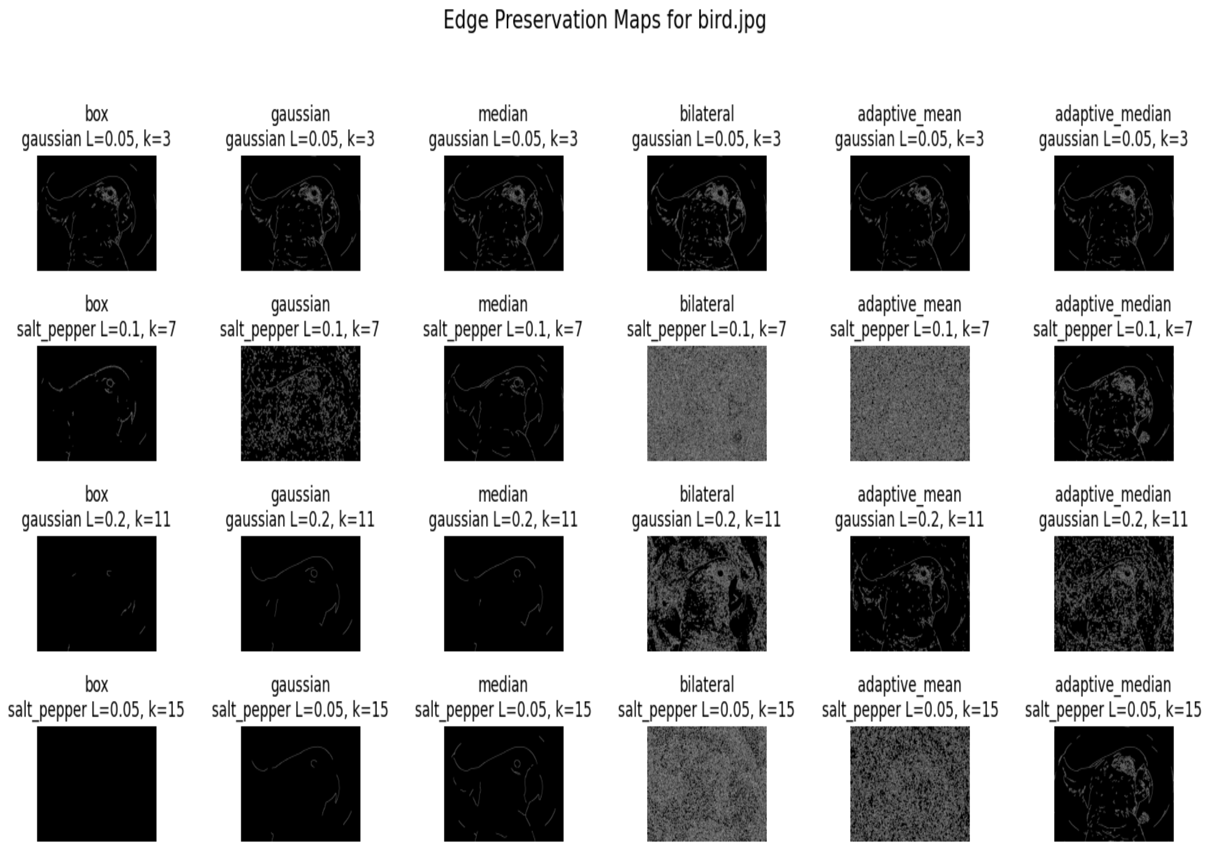

Edge Preservation Maps for bird.jpg
box gaussian L=0.05, k=3
gaussian gaussian L=0.05, k=3
median gaussian L=0.05, k=3
bilateral gaussian L=0.05, k=3
adaptive_mean gaussian L=0.05, k=3
adaptive_median gaussian L=0.05, k=3
box salt_pepper L=0.1, k=7
gaussian salt_pepper L=0.1, k=7
median salt_pepper L=0.1, k=7
bilateral salt_pepper L=0.1, k=7
adaptive_mean salt_pepper L=0.1, k=7
adaptive_median salt_pepper L=0.1, k=7
box gaussian L=0.2, k=11
gaussian gaussian L=0.2, k=11
median gaussian L=0.2, k=11
bilateral gaussian L=0.2, k=11
adaptive_mean gaussian L=0.2, k=11
adaptive_median gaussian L=0.2, k=11
box salt_pepper L=0.05, k=15
gaussian salt_pepper L=0.05, k=15
median salt_pepper L=0.05, k=15
bilateral salt_pepper L=0.05, k=15
adaptive_mean salt_pepper L=0.05, k=15
adaptive_median salt_pepper L=0.05, k=15

Figure 15: Edge Preservation Maps for bird image

The figure below shoes the edge preservation maps for the cat image, comparing how edge preservation varies with filter types and kernel sizes, with Gaussian and Salt-and-Pepper noise.

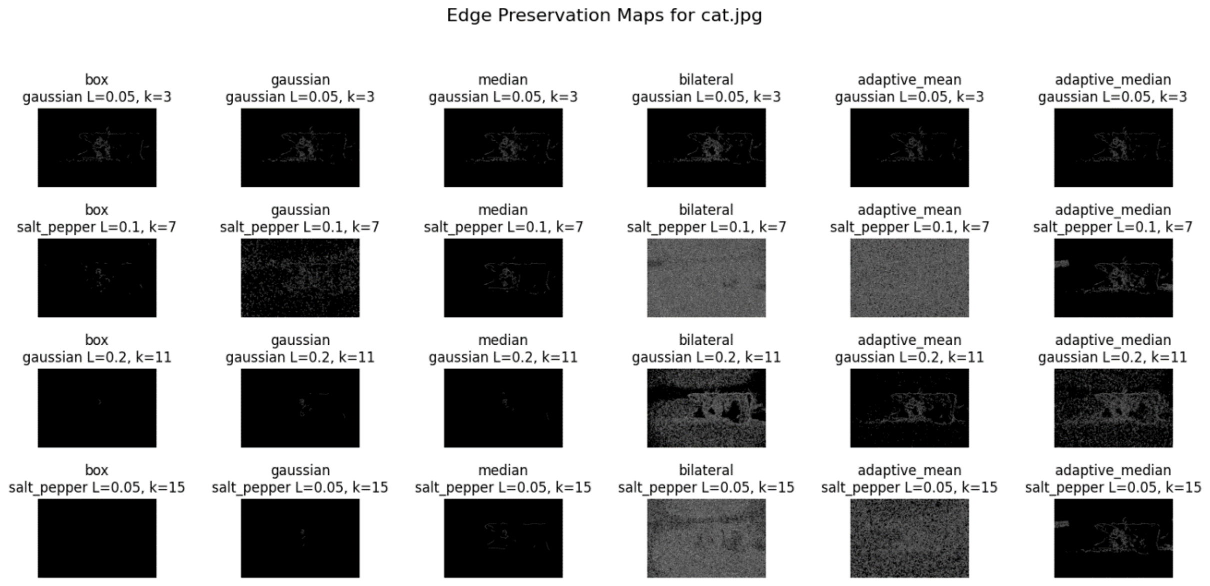

Edge Preservation Maps for cat.jpg
box gaussian L=0.05, k=3
gaussian gaussian L=0.05, k=3
median gaussian L=0.05, k=3
bilateral gaussian L=0.05, k=3
adaptive_mean gaussian L=0.05, k=3
adaptive_median gaussian L=0.05, k=3
box salt_pepper L=0.1, k=7
gaussian salt_pepper L=0.1, k=7
median salt_pepper L=0.1, k=7
bilateral salt_pepper L=0.1, k=7
adaptive_mean salt_pepper L=0.1, k=7
adaptive_median salt_pepper L=0.1, k=7
box gaussian L=0.2, k=11
gaussian gaussian L=0.2, k=11
median gaussian L=0.2, k=11
bilateral gaussian L=0.2, k=11
adaptive_mean gaussian L=0.2, k=11
adaptive_median gaussian L=0.2, k=11
box salt_pepper L=0.05, k=15
gaussian salt_pepper L=0.05, k=15
median salt_pepper L=0.05, k=15
bilateral salt_pepper L=0.05, k=15
adaptive_mean salt_pepper L=0.05, k=15
adaptive_median salt_pepper L=0.05, k=15

Figure 16: Edge Preservation Maps for cat image

The figure below shows the edge preservation maps for the dog image, demonstrating the effects of different filtering techniques on edge clarity and noise removal.

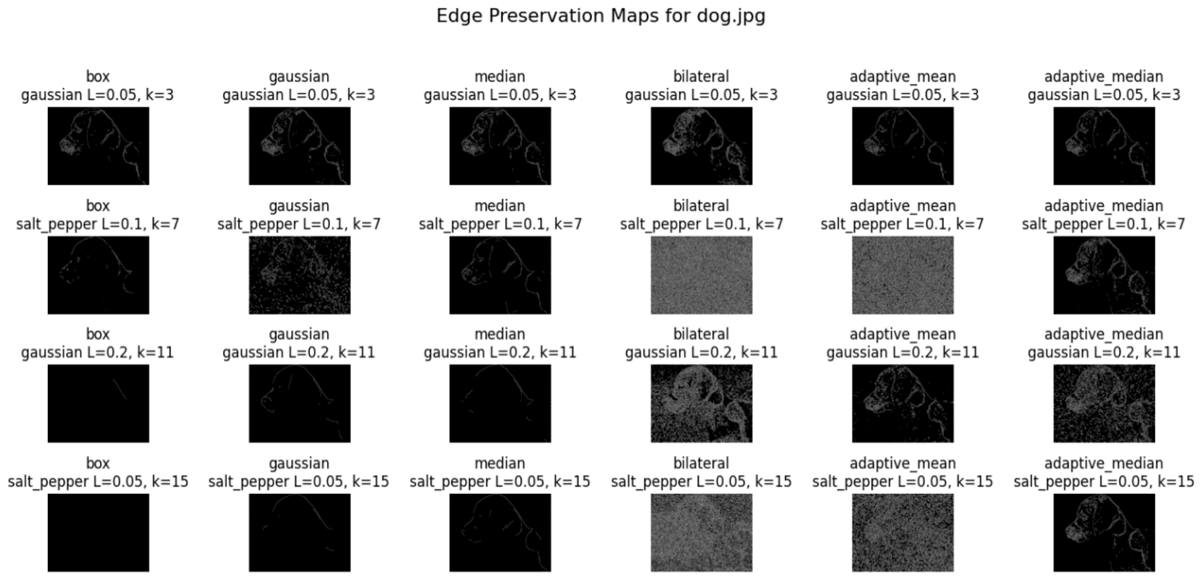

Edge Preservation Maps for dog.jpg
box gaussian L=0.05, k=3
gaussian gaussian L=0.05, k=3
median gaussian L=0.05, k=3
bilateral gaussian L=0.05, k=3
adaptive_mean gaussian L=0.05, k=3
adaptive_median gaussian L=0.05, k=3
box salt_pepper L=0.1, k=7
gaussian salt_pepper L=0.1, k=7
median salt_pepper L=0.1, k=7
bilateral salt_pepper L=0.1, k=7
adaptive_mean salt_pepper L=0.1, k=7
adaptive_median salt_pepper L=0.1, k=7
box gaussian L=0.2, k=11
gaussian gaussian L=0.2, k=11
median gaussian L=0.2, k=11
bilateral gaussian L=0.2, k=11
adaptive_mean gaussian L=0.2, k=11
adaptive_median gaussian L=0.2, k=11
box salt_pepper L=0.05, k=15
gaussian salt_pepper L=0.05, k=15
median salt_pepper L=0.05, k=15
bilateral salt_pepper L=0.05, k=15
adaptive_mean salt_pepper L=0.05, k=15
adaptive_median salt_pepper L=0.05, k=15

Figure 17: Edge Preservation Maps for dog image

The figure below shows a comparison of how well different filters keep edges with various kernel sizes. The plot show how each filter works to preserve edges when noise levels change. The points on the plot are spread out based on the kernel sizes (k=3, k=7, k=11, and k=15) for each type of filter, like Box, Gaussian, Median, Bilateral, Adaptive Mean, and Adaptive Median.

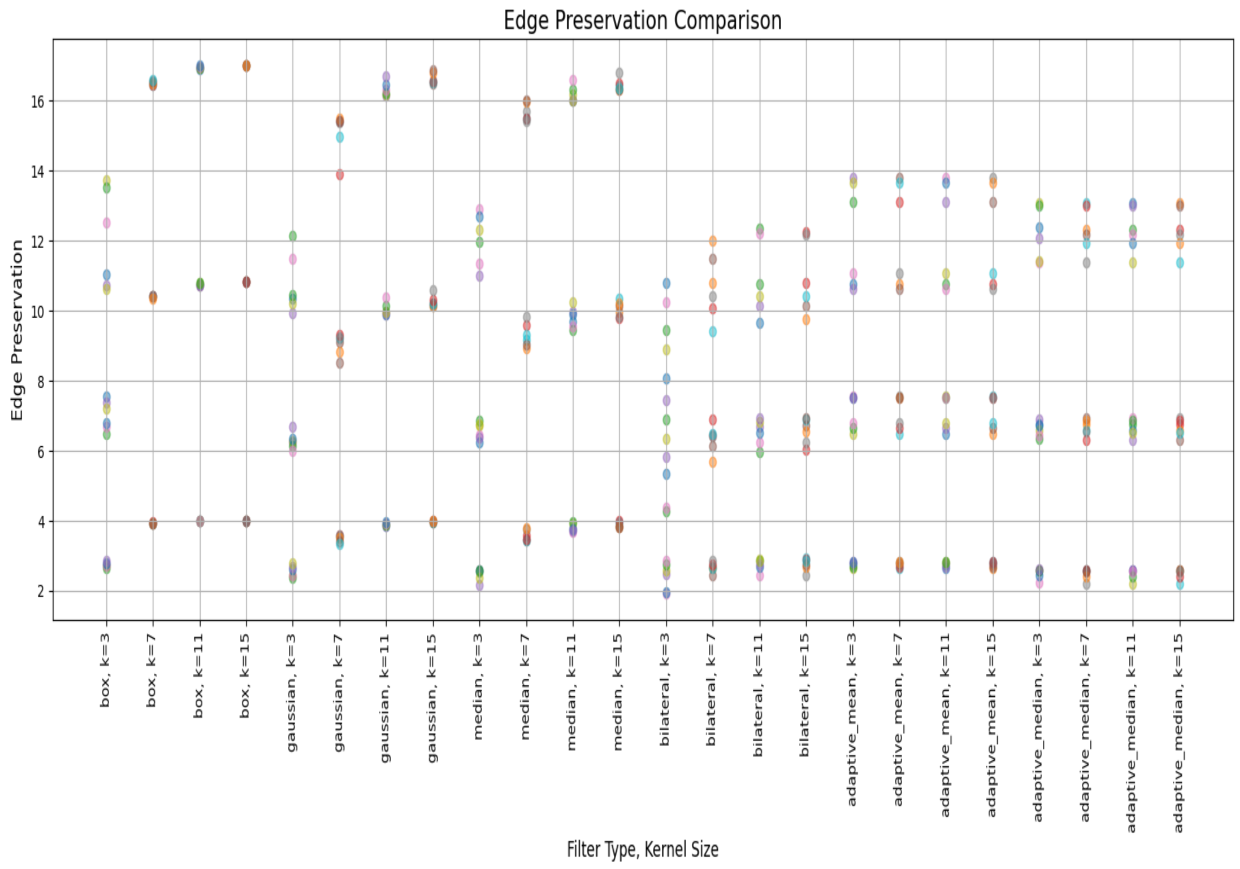


Figure 18: Edge Preservation Comparison

3.5 Computational time comparison

The figure below shows how much time different filters take to process images with different kernel sizes. The x-axis is for the kernel size (from 3 to 15), and the y-axis shows the time in seconds. Each filter, like Box, Gaussian, Median, Bilateral, Adaptive Mean, and Adaptive Median, has a marker on the plot.

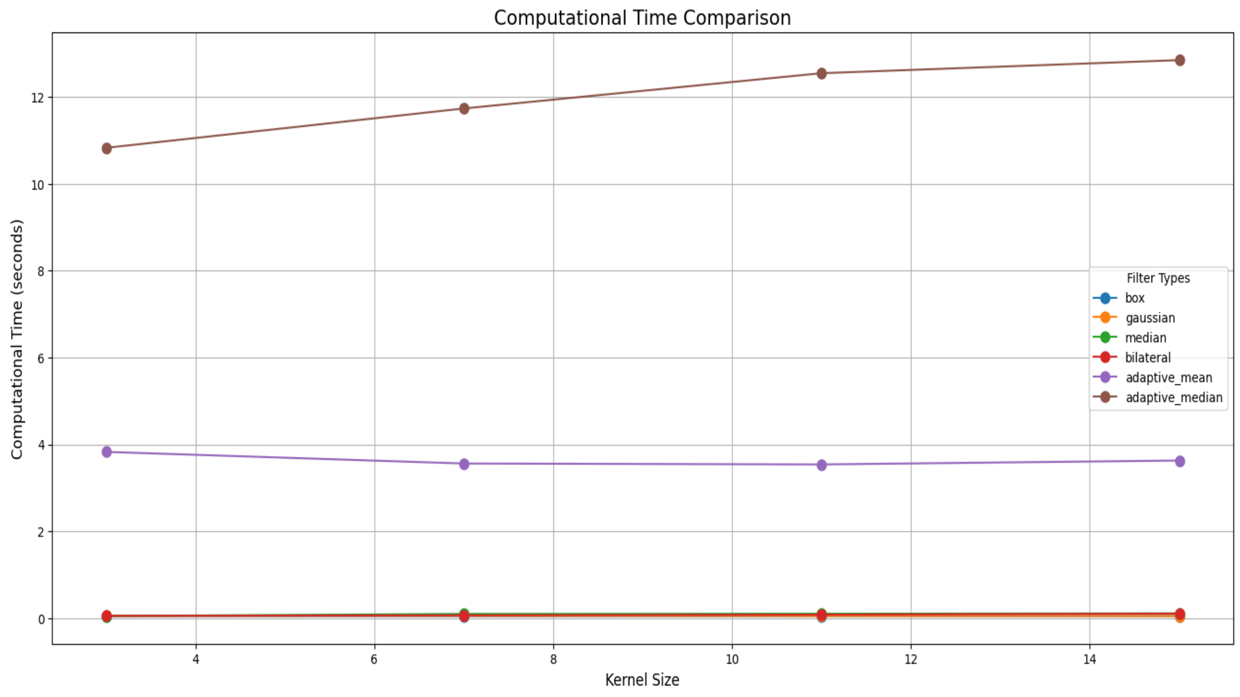


Figure 19: Computational Time Comparison

3.6 Effect of Kernel Size

The effect of kernel size on filter performance shows a balance between removing noise and keeping edges clear. When kernel size gets bigger, filters remove noise better, but they also lose more fine details and edges. For example, Box and Gaussian filters make images smoother with large kernels, but this also makes edges blurry and less sharp. The Median filter, which works well with salt-and-pepper noise, keeps edges better even with big kernels, but very large kernels still lose some details. The Bilateral filter, which tries to keep edges while removing noise, keeps edges sharper even with bigger kernels, but it needs more processing power. In general, smaller kernels keep edges clear but are not as good at removing noise. Bigger kernels remove noise well but make edges less sharp. So, choosing the right kernel size is important to balance noise removal and keeping edges clear.

4. Discussion

4.1 Filters Result

• **Gaussian Filter**: I used the Gaussian filter on the 'Cat' image with Salt-and-Pepper noise. With a small kernel size of 3, it smoothed noise a bit but made edges blurry. It works better with soft, gradual noise like Gaussian noise, not with sharp noise like Salt-and-Pepper.

• **Median Filter**: The Median filter did great on the 'Bird' image with low-intensity Gaussian noise. It kept edges sharp and reduced noise without blurring. It’s very good at removing Salt-and-Pepper noise because it ignores extreme pixel values.

• **Box Filter**: On the 'Dog' image with strong Gaussian noise, the Box filter made the image very blurry. It averages pixels evenly, removing details along with noise. It’s not the best choice when keeping fine details is important.

• **Adaptive Mean Filter**: I applied the Adaptive Mean filter to a 'Dog' image with light Salt-and-Pepper noise. It did well by adjusting to different parts of the image, keeping more details than the Box filter. It’s helpful for images with uneven noise.

• **Adaptive Median Filter**: The Adaptive Median filter was used on the 'Bird' image with high-intensity Gaussian noise. It adapted to the local area, reducing noise while keeping edges sharp. It did better than the regular Median filter in tough noise conditions.

• **Bilateral Filter**: On the 'Cat' image with medium Gaussian noise, the Bilateral filter did a great job. It reduced noise while keeping edges clear, much better than simple filters. It’s useful for images that need to keep detailed textures.

4.2 Noisy Results

The images of noise show how different types of noise, like Gaussian and Salt-and-Pepper, affect images of a bird, cat, and dog. These examples highlight the challenges of dealing with noise in image processing.

Gaussian Noise: Gaussian noise spreads across all the pixels in the image and makes it look grainy. It can hide fine details, and as the noise intensity goes from 0.05 to 0.2, the images become less clear and lose sharpness. This type of noise happens often in real life, like in low-light photos or from electronic sensor problems.

Salt-and-Pepper Noise: This noise looks like random white and black dots scattered over the image. It creates strong contrasts and can remove a lot of image details, especially at high intensity. This noise is common when there are errors in digital transmission or sensor issues.

An analysis of the bird, cat, and dog images reveals a pattern: lower noise intensity has a minimal impact on the image, allowing more details to remain visible. However, as the noise intensity increases, it becomes more challenging to discern important details, which complicates tasks such as edge detection and object recognition.

Knowing how different types of noise behave at various levels helps to choose the right filters to reduce noise. The goal is to keep as many important details as possible, which is very important for fields like computer vision, medical imaging, and remote sensing.

4.3 Noise Removal

This part evaluates the performance of different filters (Box, Gaussian, Median, Bilateral, and Adaptive) in terms of noise removal and image quality preservation. MSE (Mean Squared Error) measures the effectiveness of noise removal, while PSNR (Peak Signal-to-Noise Ratio) indicates the overall image quality.

Box Filter: The Box filter works well to remove noise when using small kernel sizes. When the kernel size gets bigger (from k=3 to k=15), the MSE gets lower, which means better noise removal. But, it makes edges blurry, and the PSNR goes down, which shows a loss in image detail. So, with bigger kernels, it removes more noise but loses important edge details.

Gaussian Filter: The Gaussian filter also gets better at noise removal with bigger kernel sizes, having lower MSE and higher PSNR. It blurs edges less than the Box filter, keeping a good balance between reducing noise and keeping details. The best results for the Gaussian filter are at kernel sizes k=7 and k=11, where it keeps image details better while still reducing noise.

Median Filter: The Median filter is really good at handling Salt-and-Pepper noise. It has low MSE and high PSNR, showing it works well. The filter keeps performing well across different kernel sizes (k=3, k=7, k=15), and it does not make edges blurry. It is better than Box and Gaussian filters for Salt-and-Pepper noise, making it the best pick for this type of noise.

Bilateral Filter: The Bilateral filter is great at keeping edges while removing noise. It uses both spatial and intensity information to smooth areas without losing edge sharpness. It gives low MSE and high PSNR, especially with bigger kernels (k=11 and k=15). But, this filter needs a lot of processing power, which can be a problem for real-time tasks.

Adaptive Filters: Adaptive filters, like Adaptive Mean and Adaptive Median, do the best at both removing noise and keeping edges. They adjust based on the image content, giving high PSNR values. These filters keep noise low even with bigger kernels, staying good at both noise reduction and edge clarity. But, they are more complex and need more time to run.

In conclusion, Bilateral and Adaptive filters are the best for balancing noise removal and keeping edges. Median filters are very good for Salt-and-Pepper noise and keep edges clear. Box filters are fast at noise removal but make images blurry, especially with big kernels. Gaussian filters are okay, keeping some details but still making edges a bit blurry. Adaptive filters are the best overall but need more processing power.

4.4 Edge Preservation

This section shows how effectively different filters (Box, Gaussian, Median, Bilateral, and Adaptive) preserve the edges and details of images when applied with varying kernel sizes.

Box Filter: The Box filter is not good at keeping edges, especially when the kernel size is big. With small kernel sizes, it removes noise okay but makes the edges blurry. As the kernel size gets bigger (from k=3 to k=15), the edges get more blurry and lose small details. The MSE gets lower, showing better noise removal, but the edges are lost.

Gaussian Filter: The Gaussian filter works better than the Box filter at keeping edges. When the kernel size gets bigger, the noise removal gets better (lower MSE), and the image quality improves (higher PSNR). The Gaussian filter keeps more edge details, especially at medium kernel sizes (k=7 and k=11). Bigger kernel sizes (k=15) still reduce noise well, but they blur edges a little.

Median Filter: The Median filter is very good at keeping edges, especially with Salt-and-Pepper noise. No matter the kernel size (k=3, k=7, k=15), the Median filter keeps edges well. It works better than Box and Gaussian filters for Salt-and-Pepper noise. Larger kernel sizes don't lose much edge detail, so it's the best for keeping edges.

Bilateral Filter: The Bilateral filter is great at keeping edges sharp while removing noise. It uses both spatial and intensity information to keep edges sharp. As the kernel size increases (k=7 to k=15), it still does well at keeping edges and removing noise. It is the best filter for images that need sharp edges and less noise, but it needs more processing power than other filters.

Adaptive Filters: Adaptive filters, like Adaptive Mean and Adaptive Median, are very good at keeping edges. These filters change based on the image, so they balance noise removal and edge preservation well. Even with bigger kernel sizes, they do better at keeping edges than other filters. They are the best for keeping details in the image.

Overall, Adaptive and Bilateral filters are the best at keeping edges while removing noise. The Median filter is also very good for Salt-and-Pepper noise. Box and Gaussian filters remove noise well, but they blur edges more as the kernel size gets bigger. So, kernel size is important because bigger kernels improve noise removal but sometimes make the edges less clear.

4.5 Computational Efficiency

This part shows how the computational time for each filter changes as the kernel size increases and the trade-off between processing speed and filter performance.

Box Filter: The Box filter generally requires more time as the kernel size increases, since more operations are needed to calculate the average over a larger area. However, if the filter is optimized (for example, using integral images), the processing time might remain constant or even decrease for larger kernels due to more efficient pixel summing methods.

Gaussian Filter: Similar to the Box filter, the Gaussian filter typically takes more time as the kernel size increases. However, optimizations such as Fast Fourier Transform (FFT) or separable convolutions may reduce the computational overhead, preventing the time from increasing as rapidly with larger kernels.

Median Filter: The Median filter usually experiences an increase in processing time with larger kernel sizes. But by using faster algorithms, such as the median-of-median or optimized sorting techniques, the computational time could either stay the same or decrease slightly, especially for larger kernels.

Bilateral Filter: The Bilateral filter tends to be slower due to its complex operations that involve both spatial and intensity information. Nevertheless, with optimizations (such as fast bilateral filtering methods or approximations), the processing time for larger kernels could be reduced, making it more efficient than the naive implementation.

Adaptive Filters: Adaptive filters (such as Adaptive Mean and Adaptive Median) are generally more computationally expensive as the kernel size increases due to their adaptive nature. However, optimizations or specific designs can allow these filters to maintain efficiency, and in some cases, the processing time might decrease with larger kernels if fewer operations are needed based on the local image content.

In general, processing time increases with kernel size, but optimizations or filter implementation may cause it to stabilize or decrease. If time decreases, it's likely due to these optimizations.

4.6 Kernel Size Sensitivity

This part shows how each filter reacts to changes in kernel size and discuss how choosing the right kernel size affects real-world applications.

Box Filter: The Box filter becomes more effective with larger kernel sizes, improving noise removal. However, larger kernels can blur edges, making it suitable for cases where noise reduction is prioritized over preserving fine details.

Gaussian Filter: Similar to the Box filter, the Gaussian filter improves with larger kernels, but excessive kernel sizes can lead to the loss of important image details. Medium-sized kernels (e.g., k=7 or k=11) generally provide a good balance between noise removal and edge preservation.

Median Filter: The Median filter is particularly effective for salt-and-pepper noise and excels in preserving edges even with larger kernel sizes. However, its computational cost increases with kernel size, making smaller kernels preferable for applications requiring faster processing times.

Bilateral Filter: The Bilateral filter is excellent at maintaining sharp edges while reducing noise, but it becomes slower with larger kernels. This makes it ideal for applications where edge preservation is critical, but care should be taken regarding the computational load, especially in real-time tasks.

Adaptive Filters: Adaptive filters, such as Adaptive Mean and Adaptive Median, offer great performance in both noise removal and edge preservation. They adapt to the image content, making them versatile for varying noise conditions. However, they are computationally expensive, especially with larger kernels, which may pose a challenge in resource-limited environments.

In general, larger kernels generally enhance noise removal but can cause blurring of image details and increase processing time. In real-world applications, the choice of kernel size depends on the specific needs for noise reduction, edge preservation, and the computational constraints of the system.

4.7 Exploring Trade-offs

This part discusses the trade-offs between noise removal, edge preservation, and the computational cost of each filter, particularly as the kernel size changes.

Box Filter: The Box filter is simple and fast, so it works well for removing noise, especially with small kernels. But, as the kernel size gets bigger, it removes noise better, but it blurs the edges more, which loses small details. The main trade-off is between quick noise removal and losing details, and bigger kernels take more time to process.

Gaussian Filter: The Gaussian filter is good at balancing noise removal and keeping edges. Bigger kernels remove more noise but blur edges more, especially with very large kernels. It doesn't blur edges as much as the Box filter. It costs more to run than the Box filter but works better when needed to keep some edges. The processing time goes up with larger kernels, but with tricks like FFT, it can be faster.

Median Filter: The Median filter works great for salt-and-pepper noise and keeps the edges well. It’s the best at keeping edges, even with bigger kernels. But, it costs more to compute than the Box and Gaussian filters, especially with large kernels. It’s a good choice when edge preservation is important, but it takes longer to process.

Bilateral Filter: The Bilateral filter keeps edges sharp while reducing noise. It’s perfect for images where keeping edges is important. It works well with bigger kernels, but it’s slow because it uses a lot of calculations. It keeps edges better than the Box and Gaussian filters, but it’s slow and expensive to compute, especially with big kernels.

Adaptive Filters: Adaptive filters, like Adaptive Mean and Adaptive Median, adjust to the image and are great for both noise removal and edge preservation. They work well in different noise conditions. But, they take more time to compute, especially with larger kernels. The trade-off is high-quality results, but they take longer to process.

In general, each filter has its pros and cons depending on how much noise removal, edge preservation, and processing time. Box and Gaussian filters are faster but blur edges more with bigger kernels. Median and Adaptive filters keep edges better but are slower, especially with big kernels. The Bilateral filter keeps edges the best but is the slowest. Choosing the right filter and kernel size depends on the quality needed and how fast the process should be.

5. Conclusion

In this study, I looked at different filters for removing noise in images: Box, Gaussian, Median, Bilateral, and Adaptive filters. The Box filter was the fastest but made edges blurry with big kernels. The Gaussian filter gave a good balance between removing noise and keeping edges, especially with medium-sized kernels. The Median filter was very good for cleaning Salt-and-Pepper noise and keeping edges clear but was a bit slow. The Bilateral filter kept edges very well but was slow and needed more processing power. Adaptive filters worked the best for both noise cleaning and edge keeping, but they were the slowest.

Recommendations:

- Use the Box filter if needed fast noise cleaning with small kernels.
- Median filter is best for Salt-and-Pepper noise and keeping edges.
- Gaussian filter is good for normal noise removal and edge balance.
- For keeping edges sharp, pick Bilateral or Adaptive filters, but know they are slower.
- Medium kernels (like k=7 or k=11) work well for most tasks.