Lab Report 8 : Methods of Corrupting and Restoring Images: A Study of Gaussian, Rayleigh, Erlang, and Salt and Pepper Noise Removal and Periodic Noise Removal

Course Title: Digital Image Processing Lab

Course Code: CSE-406

Date of Performance: October 20, 2024
Date of Submission: October 27, 2024

Submitted to:

Dr. Morium Akter
Professor

Dr. Md. Golam Moazzam
Professor

Department of Computer Science and Engineering
Jahangirnagar University

Submitted by:

Class Roll	Exam Roll	Name
370	202182	Rubayed All Islam

Department of Computer Science and Engineering
Jahangirnagar University
Savar, Dhaka, Bangladesh

Experiment Name

a **Corrupting Image by Gaussian, Rayleigh, and Erlang Noise**

b **Noise Removal by Arithmetic Mean Filter and Geometric Mean Filter**

c **Removal of Salt and Pepper Noise**

d **Periodic Noise Removal**

Objectives

The objectives of the experiments performed in this lab are as follows:

1. Introduce Various Noise Types: Understand how Gaussian, Rayleigh, and Erlang noise affect image quality by corrupting an original image.
2. Evaluate Filtering Techniques: Apply arithmetic and geometric mean filters to assess their effectiveness in removing noise from corrupted images.
3. Analyze Salt and Pepper Noise: Introduce salt and pepper noise to an image and implement a median filter to restore image quality.
4. Remove Periodic Noise: Employ Fourier Transform techniques, specifically band reject filtering, to remove periodic noise from images.

Experiment 1: Corrupting Image by Gaussian, Rayleigh, and Erlang Noise

MATLAB Code:

Listing 1: Gaussian_Reyligh_Erlang.m

```
% Read the image
img = imread('image.jpg'); % Replace with your image file
img = im2double(img); % Convert to double for noise addition

% Gaussian Noise
gaussian_noise = imnoise(img, 'gaussian', 0, 0.01); % mean=0,
    variance=0.01

% Rayleigh Noise
rayleigh_noise = img + raylrnd(0.2, size(img)); % scale
    parameter=0.2

% Erlang Noise (Using Gamma distribution since Erlang is a
    special case of Gamma)
k = 0.25; % Shape parameter (integer for Erlang)
lambda = 1; % Rate parameter
erlang_noise = img + gamrnd(k, 1/lambda, size(img));
```

```
% Plotting the original and noisy images
figure;

subplot(2,2,1);
imshow(img);
title('Original Image');

subplot(2,2,2);
imshow(gaussian_noise);
title('Image with Gaussian Noise');

subplot(2,2,3);
imshow(rayleigh_noise);
title('Image with Rayleigh Noise');

subplot(2,2,4);
imshow(erlang_noise);
title('Image with Erlang Noise');
```

Output:

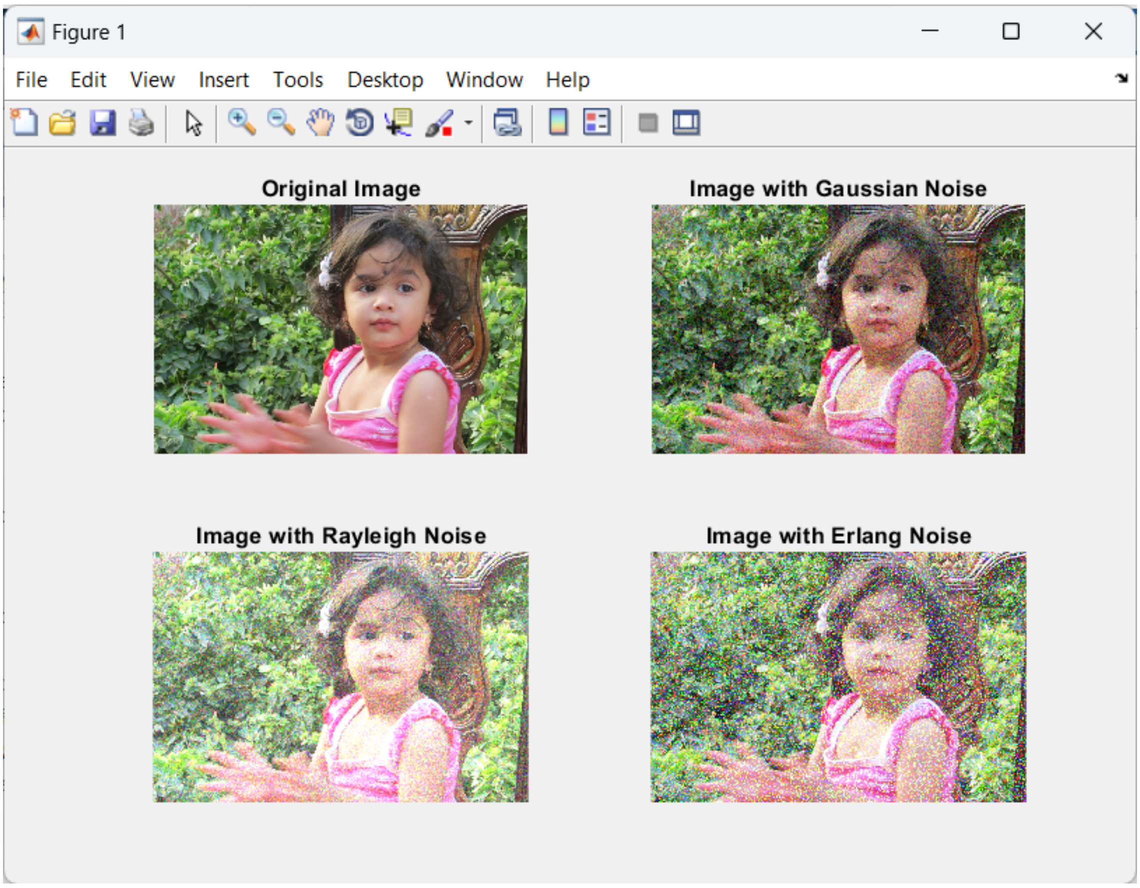

Figure 1: Corrupting Image by Gaussian, Rayleigh, and Erlang Noise

Experiment 2: Noise Removal by Arithmetic Mean Filter and Geometric Mean Filter

MATLAB Code:

Listing 2: GaussianNoise_Arithmetic_and_Geometric_Mean_Filter.m

```
% Read the image
img = imread('motherboard.png'); % Replace with your image file
img = im2double(img); % Convert to double for better precision

% Add Gaussian Noise to the Image
noisy_img = imnoise(img, 'gaussian', 0, 0.01); % mean=0, variance=0.01

% 3x3 Arithmetic Mean Filter
arithmetic_mean_filter = fspecial('average', [3 3]); % Create a 3x3 averaging filter
arithmetic_filtered_img = imfilter(noisy_img, arithmetic_mean_filter, 'replicate');

% 3x3 Geometric Mean Filter (Custom implementation)
geometric_filtered_img = exp(imfilter(log(noisy_img + eps), ones(3,3), 'replicate') / 9);

% Plotting the original, noisy, and filtered images
figure;

subplot(2,2,1);
imshow(img);
title('Original Image');

subplot(2,2,2);
imshow(noisy_img);
title('Image with Gaussian Noise');

subplot(2,2,3);
imshow(arithmetic_filtered_img);
title('Arithmetic Mean Filtered Image');

subplot(2,2,4);
imshow(geometric_filtered_img);
title('Geometric Mean Filtered Image');
```

Output:

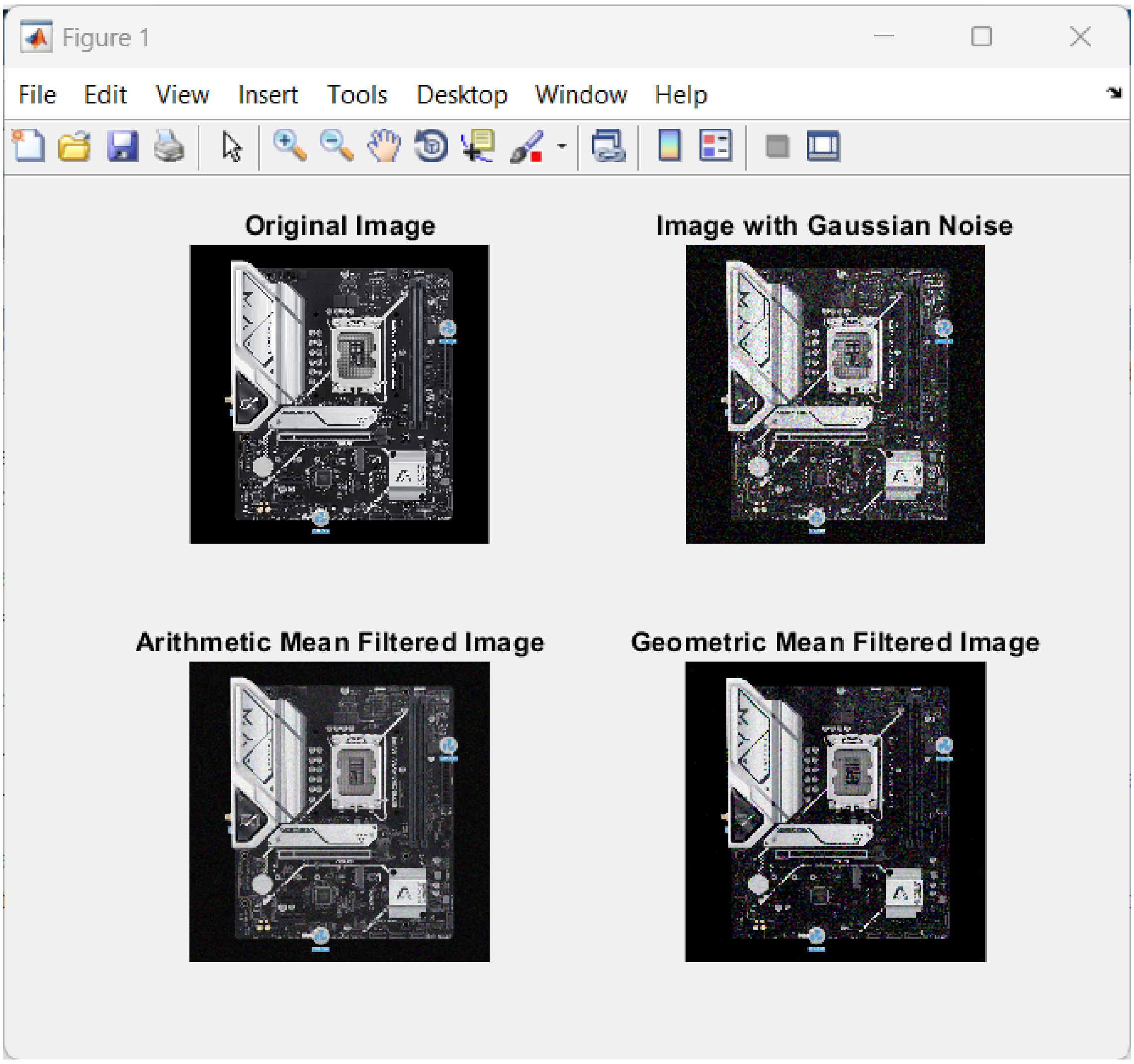

Figure 2: Noise Removal by Arithmetic Mean Filter and Geometric Mean Filter

Experiment 3: Removal of Salt and Pepper Noise

MATLAB Code:

Listing 3: Salt_Paper_Noise_Removal.m

```
% Read the image
img = imread('image.jpg'); % Replace with your image file

% Add Salt and Pepper Noise
noisy_img = imnoise(img, 'salt & pepper', 0.10); % Add salt &
    pepper noise with noise density 0.05

% Check if the image is grayscale or color
if size(noisy_img, 3) == 3
    % If color image, apply median filter to each channel
        separately
    denoised_img = noisy_img; % Initialize denoised_img with the
        same size
    for c = 1:3
        denoised_img(:,:,c) = medfilt2(noisy_img(:,:,c), [3 3]);
            % Apply median filter to each color channel
    end
else
    % If grayscale image, apply median filter directly
    denoised_img = medfilt2(noisy_img, [3 3]); % Apply median
        filter to grayscale image
end

% Plotting the original, noisy, and denoised images
figure;

subplot(2,2,1);
imshow(img);
title('Original Image');

subplot(2,2,2);
imshow(noisy_img);
title('Image with Salt & Pepper Noise');

subplot(2,2,3);
imshow(denoised_img);
title('Denoised Image (Median Filter)');
```

Output:

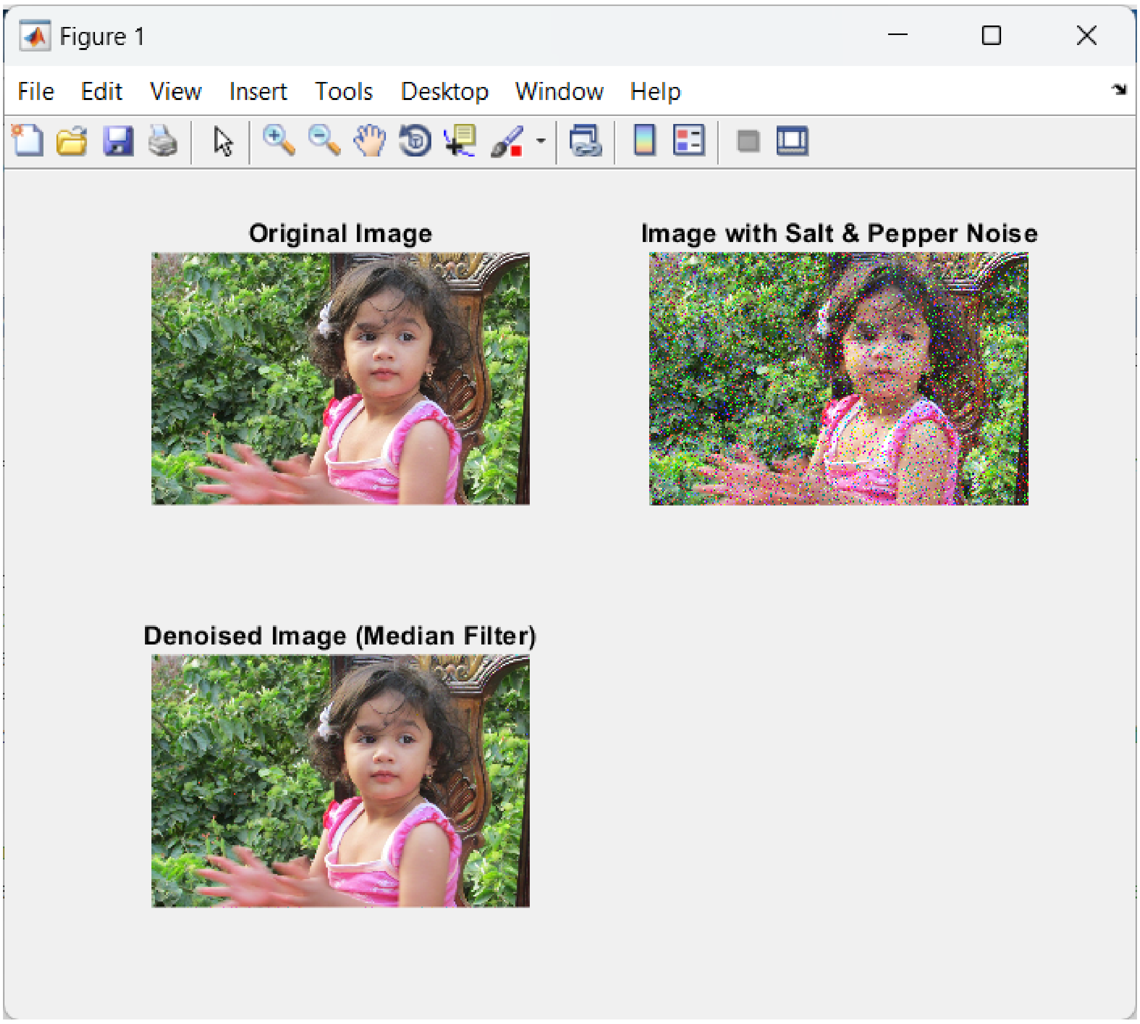

Figure 3: Removal of Salt and Pepper Noise

Experiment 4: Periodic Noise Removal

MATLAB Code:

Listing 4: Periodic_Noise_Removal.m

```
% Read the image
img = imread('image.jpg'); % Replace with your image file

% Check if the image is already grayscale, if not, convert it to
    grayscale
if size(img, 3) == 3
    img = rgb2gray(img); % Convert to grayscale if the image is
        in color
end

% Add periodic noise (for demonstration purposes)
[M, N] = size(img);
[x, y] = meshgrid(1:N, 1:M);
periodic_noise = 20 * sin(2 * pi * x / 10) + 20 * cos(2 * pi * y
    / 15);
noisy_img = double(img) + periodic_noise;

% Fourier Transform of the Noisy Image
F = fftshift(fft2(noisy_img));

% Display the Fourier Spectrum
figure;
subplot(2,3,1);
imshow(img, []);
title('Original Image');

subplot(2,3,2);
imshow(noisy_img, []);
title('Image with Periodic Noise');

subplot(2,3,3);
imshow(log(1 + abs(F)), []);
title('Fourier Spectrum of Noisy Image');

% Band Reject Filter Design
band_reject_filter = ones(M, N);
center_freq_x = N/2; % Center frequency in the x direction
center_freq_y = M/2; % Center frequency in the y direction
band_width = 10; % Width of the band to reject

% Create the band reject filter
for u = 1:M
    for v = 1:N
        D = sqrt((u - center_freq_y)^2 + (v - center_freq_x)^2);
            % Distance from center
```

```
        if (D >= 15) && (D <= (15 + band_width)) % Reject the frequencies in the specified band
            band_reject_filter(u, v) = 0;
        end
    end
end

% Apply the Band Reject Filter to the Fourier Transform
filtered_F = F .* band_reject_filter;

% Inverse Fourier Transform to get the denoised image
filtered_img = real(ifft2(ifftshift(filtered_F)));

% Normalize the filtered image for better visualization
filtered_img = mat2gray(filtered_img);

% Plotting the results
subplot(2,3,4);
imshow(log(1 + abs(filtered_F)), []);
title('Filtered Fourier Spectrum');

subplot(2,3,5);
imshow(filtered_img, []);
title('Denoised Image (After Band Reject Filtering)');
```

Output:

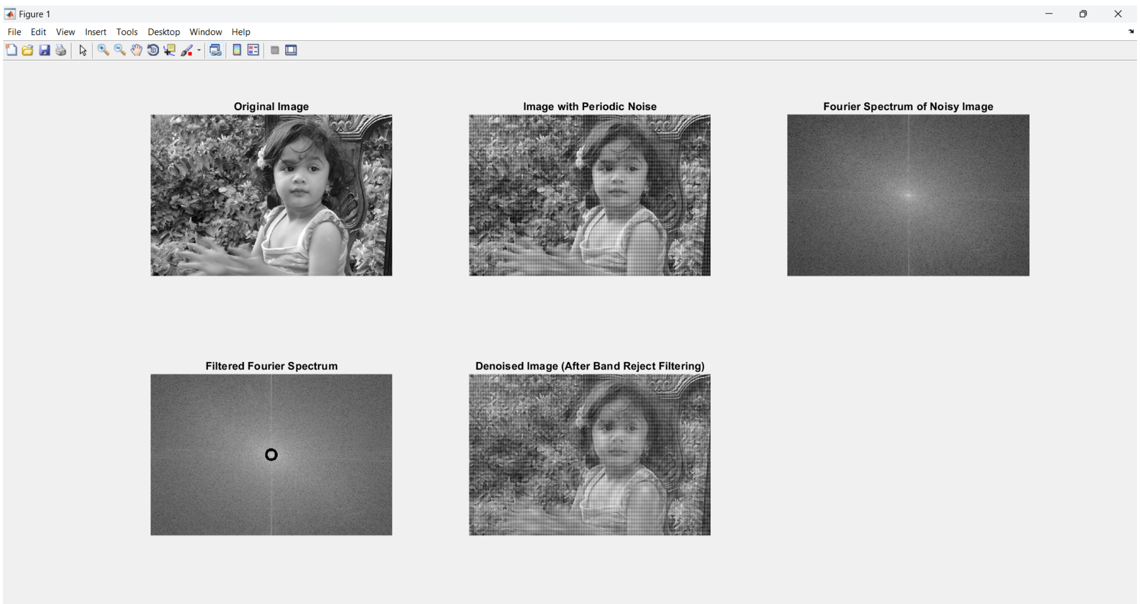

Figure 4: Periodic Noise Removal

Results

1. Corrupting Image by Gaussian, Rayleigh, and Erlang Noise

The original grayscale image was successfully corrupted by Gaussian, Rayleigh, and Erlang noise.

- **Gaussian Noise**: Introduced a random variation in pixel values based on a defined mean and variance, resulting in a grainy appearance.
- **Rayleigh Noise**: Showed a distinct pattern of noise, particularly affecting darker regions of the image.
- **Erlang Noise**: Similar to Rayleigh but exhibited a different statistical distribution, producing a noticeable distortion in pixel intensity.

2. Noise Removal by Arithmetic Mean Filter and Geometric Mean Filter

The filtering techniques were applied to the noisy images:

- **Arithmetic Mean Filter**: This filter effectively smoothed out the Gaussian noise but resulted in a slight blurring of image edges, leading to a loss of detail.
- **Geometric Mean Filter**: This filter demonstrated improved edge preservation compared to the arithmetic mean filter while still effectively reducing noise, making it particularly useful for images where detail retention is important.

3. Removal of Salt and Pepper Noise

After introducing salt and pepper noise, the median filter was applied:

- **Median Filter (3x3)**: Successfully removed the salt and pepper noise, restoring the image quality significantly. It preserved edges and fine details better than the arithmetic mean filter, highlighting the median filter's effectiveness in dealing with impulsive noise.

4. Periodic Noise Removal

Periodic noise was artificially added to an image, followed by the application of Fourier Transform techniques:

- **Band Reject Filtering**: The periodic noise was effectively targeted using a band reject filter in the frequency domain. The filter was designed to eliminate specific frequency bands where the periodic noise was concentrated. After applying the inverse Fourier Transform, the image quality improved markedly, demonstrating the efficacy of frequency domain methods for periodic noise removal. Adjustments to the band reject filter parameters enhanced the filtering results.

Conclusion

The experiments conducted demonstrated various aspects of image processing related to noise corruption and removal. The introduction of different noise types illustrated their impact on image quality. The application of filtering techniques, particularly the median and geometric mean filters, proved effective in restoring images corrupted by salt and pepper and Gaussian noise. Additionally, the use of Fourier Transform techniques highlighted a robust method for addressing periodic noise.

In conclusion, the lab exercises reinforced the understanding of noise characteristics, the effectiveness of different filtering techniques, and the significance of both spatial and frequency domain methods in image processing.