Project: Image Processing

Marta Casado Carrasquer

June 5, 2025

1 Introduction

Image denoising is one of the tasks found in image processing. In this project, we study image denoising through filtering using the Fast Fourier Transform (FFT) in the frequency-domain. We apply three types of noise: Gaussian, salt-and-pepper, and multiplicative to greyscale images, and compare its metrics against some spatial filters: mean, Gaussian, and median.

2 Theory and Methods

2.1 Image Denoising in Image Processing

Image denoising is a task in image processing where we restore an image that has been corrupted by noise. The objective is to remove or reduce noise while maintaining as much detail as possible. Denoising techniques are applied in different fields, such as medicine or astrophysics.

Noise in images can come from different sources, such as transmission errors, or environmental conditions. Different noise models reflect different statistical characteristics, and the effectiveness of a denoising method often depends on the type of noise present in the image.

2.2 Different Types of Noise

In this project, we simulate three common types of noise: Gaussian noise, salt-and-pepper noise, and multiplicative noise.

2.2.1 Gaussian Noise

Gaussian noise is distributed according to a normal distribution:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1}$$

where μ is the mean (usually 0) and σ^2 is the variance of the distribution. This type of noise affects all pixels in the image independently.

2.2.2 Salt-and-Pepper Noise

Salt-and-pepper noise is a type of impulsive noise where random pixels are set to either the minimum or maximum intensity value. The parameter **amount** determines the amount of pixels affected.

2.2.3 Multiplicative Noise

Multiplicative noise can be found in imaging systems like untrasound. It follows the model:

$$I_{\text{noisy}}(x, y) = I(x, y) \cdot n(x, y) \tag{2}$$

where $n(x, y)$ is a random variable with a Gaussian distribution centred around 1.

2.3 Spatial Filters

Spatial filtering manipulates pixels in the spatial domain directly using a kernel. Here, three types of spatial filters were used:

2.3.1 Mean Filter

The mean filter is a linear filter that replaces each pixel value with the average of the neighbouring pixel values defined by the kernel size:

$$I_{\text{filtered}}(x, y) = \frac{1}{N^2} \sum_{i=-k}^{k} \sum_{j=-k}^{k} I(x+i, y+j) \tag{3}$$

where $N = 2k + 1$ is the kernel size.

2.3.2 Gaussian Filter

The Gaussian filter is a linear smoothing filter where weights are assigned based on the Gaussian function:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{4}$$

It reduces noise while preserving edge characteristics better than a mean filter.

2.3.3 Median Filter

The median filter is a non-linear filter that replaces each pixel with the median of its neighbouring pixels.

2.4 Frequency-Domain Filtering and FFT

2.4.1 Fast Fourier Transform (FFT)

The Fourier Transform (FT) converts a spatial domain image into the frequency domain. The 2D Discrete Fourier Transform is defined as:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x, y) e^{-2\pi i \left(\frac{ux}{M} + \frac{vy}{N}\right)} \tag{5}$$

Its inverse is given by:

$$I(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{2\pi i \left(\frac{ux}{M} + \frac{vy}{N}\right)} \tag{6}$$

FFT is an efficient algorithm to compute the DFT and is used here to convert images to the frequency domain, apply filters, and then return to the spatial domain using the inverse FFT.

FFT is applied in the code using the built-in functions from NumPy.

2.4.2 FFT-Based Filters

For this project, different frequency-domain filters were implemented:

- **Ideal Lowpass Filter (ILPF)**: Passes all frequencies below a cutoff D_0 and rejects the rest.

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases} \tag{7}$$

- **Ideal Highpass Filter (IHPF)**: Rejects all frequencies below D_0 and passes the rest.

$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \leq D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases} \tag{8}$$

- **Gaussian Lowpass Filter (GLPF)**: Expressed as:

$$H(u,v) = \exp\left(-\frac{D(u,v)^2}{2D_0^2}\right) \tag{9}$$

- **Gaussian Highpass Filter (GHPF)**: Expressed as:

$$H(u,v) = 1 - \exp\left(-\frac{D(u,v)^2}{2D_0^2}\right) \tag{10}$$

where $D(u,v)$ is the distance from the centre of the frequency domain:

$$D(u,v) = \sqrt{(u - M/2)^2 + (v - N/2)^2} \tag{11}$$

2.5 Evaluation Metrics

To evaluate the quality of the denoised images, two standard metrics were used:

2.5.1 Peak Signal-to-Noise Ratio (PSNR)

PSNR is defined as:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{12}$$

where MAX_I is the maximum possible pixel value and MSE is the mean squared error between the original and denoised images.

2.5.2 Structural Similarity Index (SSIM)

SSIM evaluates image quality by comparing structural information, luminance, and contrast:

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{13}$$

where μ_x, μ_y are local means, σ_x^2, σ_y^2 are variances, σ_{xy} is the covariance, and C_1, C_2 are constants for stability.

3 Results and Discussion

3.1 Overview

The goal of this project was to study different image denoising techniques, comparing spatial domain filters with frequency domain (FFT-based) filters. We first applied different types of noise to standard test images from the `skimage` library. For simplicity, the images used were in greyscale or converted to it since it reduced the complexity of filtering and visualization.

We then applied some spatial filters to analyse their performance. Following the spatial filter implementation, we proceeded to implement and study frequency-domain filters using the Fast Fourier Transform (FFT).

3.2 Noise Analysis

Figure 1 shows the different types of noises applied where the variance, σ^2 and **amount** parameters were varied.

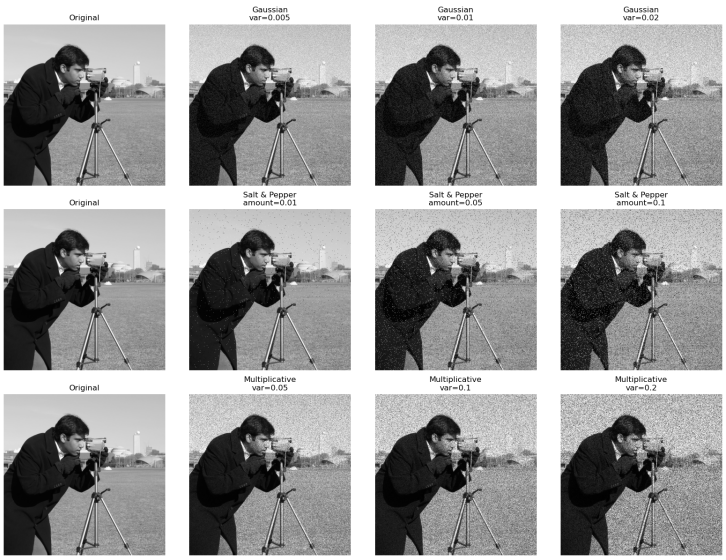

Figure 1: Different types of noises applied for different variances and amount parameters.

- **Gaussian Noise:** Controlled by its variance. As the variance increased, the image became increasingly blurry and grainy.
- **Salt-and-Pepper Noise:** Controlled by the **amount** parameter. This noise manifests as white and black pixels. Higher amounts lead to more corrupted pixels.

- **Multiplicative Noise:** It simulates granular distortions. Like Gaussian noise, it increases in intensity with variance.

3.3 Spatial Filter Analysis

For each implemented noise, the three spatial filters were applied to denoise each image.

Figure 2 below shows the results obtained:

Figure 2: Application of different spatial filters for three different noises.

Table 1 below shows the spatial filter performance metrics:

Noise Type	**Filter**	**PSNR (dB)**	**SSIM**
Gaussian	Mean	26.516713	0.572430
Gaussian	Gaussian	27.174458	0.638544
Gaussian	Median	25.752296	0.496391
Salt & Pepper	Mean	24.907437	0.506737
Salt & Pepper	Gaussian	25.692408	0.569048
Salt & Pepper	Median	30.113861	0.864012
Multiplicative	Mean	25.358470	0.575754
Multiplicative	Gaussian	26.147739	0.626769
Multiplicative	Median	24.110115	0.523582

Table 1: Comparison of spatial filters performance for different noise types using PSNR and SSIM metrics.

- **Gaussian Noise:** Gaussian filtering performed best with PSNR $\approx$ 27.17 dB and SSIM $\approx$ 0.64.
- **Salt-and-Pepper Noise:** Median filtering outperformed the others with PSNR $\approx$ 30.11 dB and SSIM $\approx$ 0.86.
- **Multiplicative Noise:** Gaussian filtering gave slightly better results here again in comparison with the other two filters.

3.4 Frequency Filter Analysis

The FFT transforms the spatial image into its frequency representation, where low frequencies represent smooth variations and high frequencies represent rapid ones.

Each FFT-based filter consists of the following steps:

1. Compute the FFT of the noisy image and shift the zero frequency to the center.
2. Design a filter mask (ideal or Gaussian, low-pass or high-pass).
3. Multiply the FFT image by the mask.
4. Apply the inverse FFT to obtain the filtered image.

Figure 3 shows for each FFT-based filter applied: the original FFT spectrum of the image, the designed mask (white = passed frequencies, black = blocked) and the reconstructed image after filtering.

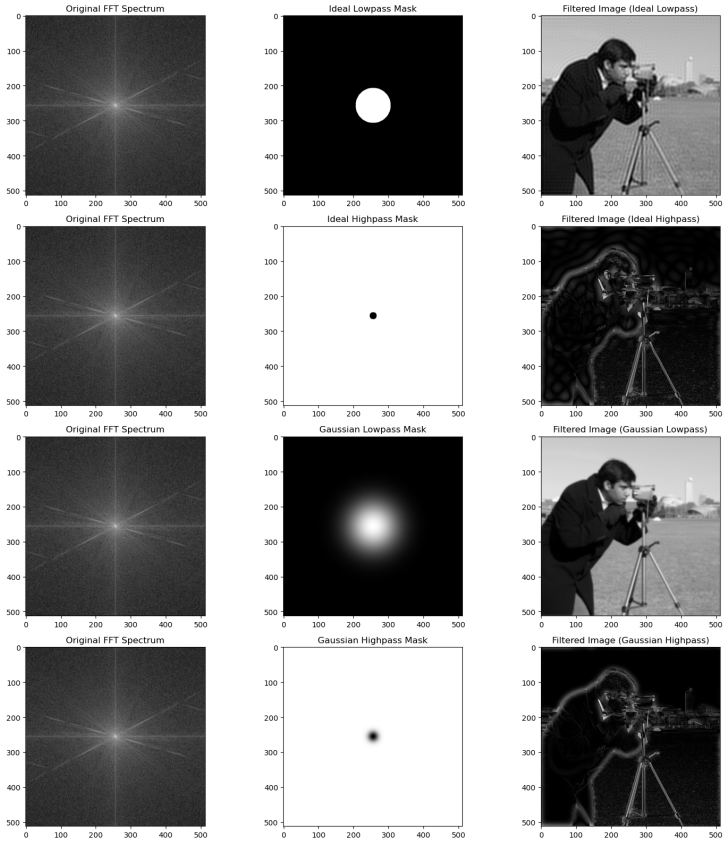

Figure 3: Application of different FFT-based filters. $D_0 = 50$ for low-pass filters and $D_0 = 10$ for hig-pass filters.

Low-Pass Filters

- **Ideal Low-Pass Filter (ILPF):** Cuts off frequencies beyond a fixed radius D_0. Produces a sharp cutoff, which leads to blurring.
- **Gaussian Low-Pass Filter (GLPF):** Applies a smoother attenuation of high frequencies, resulting in less artifacting and a more natural blurring.

Low-pass filters aim to retain the low-frequency components while removing high-frequency noise. Hence, they tend to preserve the overall appearance but blur fine details.

High-Pass Filters

- **Ideal High-Pass Filter (IHPF):** Retains frequencies above D_0, removing smooth content. Produces sharp edges and causes strong contrast and darkening.
- **Gaussian High-Pass Filter (GHPF):** Suppresses low frequencies. Enhances edges and details. Less abrupt transitions than with **IHPF**.

High-pass filters emphasize edges. This is because much of the image's energy resides in the low-frequency domain. By suppressing it, we darken the image and contours more visible.

3.5 Effects of Varying D_0

To tune the performance of the FFT-based filters, we varied the cutoff frequency D_0.

Figures 4, 5, 6 and 7 show how the output image varies as D_0 varies.

Figure 4: GLPF for different D_0 values.

Figure 5: ILPF for different D_0 values.

A small D_0 for the low-pass filter increases blurring but removes noise more aggressively when present, whereas larger D_0 values preserve more detail but retain some noise.

Figure 6: GHPF for different D_0 values.

Figure 7: IHPF for different D_0 values.

For high-pass filters, a small D_0 keeps only the highest frequencies which leads to strong edge enhancement but looses smoothness. A larger D_0 maintains more mid-frequencies which leads to softer enhancement and sharpness.

3.6 Different FFT-based Filters for Different Cases

We chose a noise and image combination that we thought would work best for each FFT-filter.

Figures 8, 9, 10 and 11 show each filter's application.

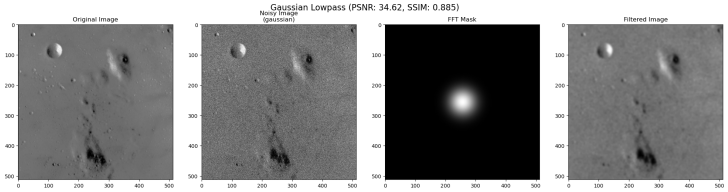

Figure 8: GLPF applied into a noisy image containing Gaussian noise for $D_0 = 30$.

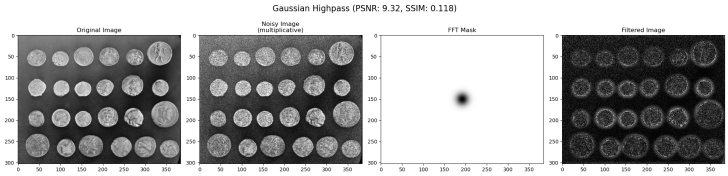

Figure 9: GHPF applied into a noisy image containing multiplicative noise for $D_0 = 10$.

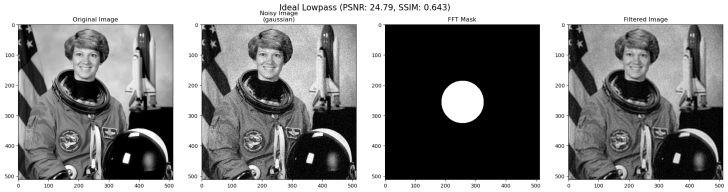

Figure 10: ILPF applied into a noisy image containing Gaussian noise for $D_0 = 70$.

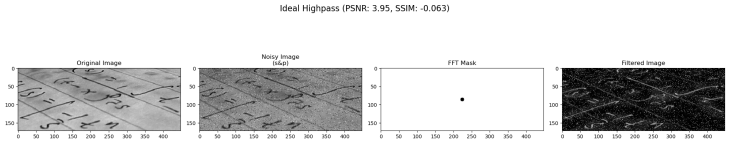

Figure 11: IHPF applied into a noisy image containing salt-and-pepper noise for $D_0 = 5$.

From the figures obtained we can observed how GLPF worked reasonably well and in line with theory since for a low D_0 (regarding the image's size) we were able to remove some noise and obtain good metrics. The trade-off paid is a blurred image compared to the original due to the size of the cutoff.

IDLF did not performed that well, a much bigger cut-off was needed and the output obtained did not differ much from the noisy image. A lower cutoff, although it removed noise, introduced a lot of ringing.

GHPF and IHPF performed very poorly when looking at the metrics. For the GHPF, we were able to enhance the edges of the coins, but the figures in their interiors were completely lost. On the other hand, we managed to sharpen the text in the image for the IHPF, which confirmed its reliability.

3.7 Comparison of Spatial and FFT-based Filters

Finally, we thought of comparing the best-performing spatial and FFT-based filters for each type of noise in order to draw some conclusions between the methods.

Table 2: Comparison of Best Spatial and FFT Filter Metrics per Noise Type

Noise Type	**Filter Type**	**Method**	**PSNR (dB)**	**SSIM**
Gaussian	FFT	Gaussian LPF	34.62	0.885
	Spatial	Gaussian Filter	27.17	0.638
Salt & Pepper	FFT	Ideal HPF	3.95	-0.063
	Spatial	Median Filter	30.11	0.864
Multiplicative	FFT	Gaussian HPF	9.32	0.119
	Spatial	Gaussian Filter	26.15	0.524

From the results obtained in the table above, we can see how the GLPF outperforms the spatial filter for the Gaussian noise, whereas spatial filters produced better metrics for the rest of the noises.

Note that **PSNR** measures how similar the output image is to the original. High-pass filters clearly alter the image much more than low-pass filters. That could explain such low metric values.

4 Analysis and Conclusions

While this project provided some insight into the application of image processing techniques, I wanted to mention how the direction of the project changed over time. Initially, with the big number of filters to test, I became overwhelmed and lost track of an objective.

While testing, other filters were implemented as well, like band-pass filters.

As my understanding of how different filters worked improved, I wanted to try and apply the techniques to a real-world context like astrophysics. However, I encountered difficulties using new libraries such as `astropy`, and was unsure how to properly load FITS files or find appropiate images.

Thus, I decided to continue working with images from the `skimage` library since they were much easier to access and manipulate.

Overall, this project shows the strengths and limitations of the FFT-based filters studied alongside with a comparison with some spatial filters. This project builds an initial foundation into denoising images in image processing.

5 Literature

References

[1] Gonzalez, R. C., & Woods, R. E. (2002). *Digital Image Processing* (2nd ed.). Prentice Hall.

[2] Tania, Sheikh & Rowaida, Raghad. (2016). A comparative study of various image filtering techniques for removing various noisy pixels in aerial image. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, *9*(3), 113–124. https://doi.org/10.14257/ijsip.2016.9.3.10