Denoising Image with Fast Fourier Transform

Introduction

Images are commonly used in people's daily life and it is common to find an image with noise. This paper studies denoising method on 25 images with three different level of noises, noise 10, noise 25 and noise 50 – the image dataset is provided from The Berkeley Segmentation Dataset by Mark *et al.* [1]. The given images have Gaussian noise which is well known as additive noise. According to Suryanarayana *et al.* [2], Gaussian noise can be effectively removed by Gaussian filter but it can make the edges blur since it is a linear filter.

For solving this problem, Fast Fourier Transformation [3] is executed to convert image into frequency so that Gaussian filter can be operated on the frequency domain instead of image. Furthermore, deconvolution technique is performed to clear the blurred edges using Wiener filter [4].

Architecture

Since the grayscale image is advantageous in noise reduction, the image dataset has been converted from 3 colour channels, RGB into 1 colour channel, grayscale as a pre-processing step.

After that, the image has transformed into frequency by implementing Fourier Transformation, especially Fast Fourier Transformation (FFT). According to Liu *et al.* [3], unlike light or sound wave, digital image is discrete due to the pixel's discontinuity. In other words, Discrete Fourier Transformation (DFT) is a proper method for image processing. However, as the pace of DFT is slow, FFT is performed in the algorithm as displayed in Fig. 1.

Once FFT is successfully applied, it is possible to visualise frequencies on the spectrum. Mostly, low frequencies locate at the corner of spectrum so centralising them makes it easy to apply the mask.

Since the purpose of the algorithm is reducing the noise, low pass filter is created as a mask. Especially, Gaussian low pass filter is implemented as it has a smoother edge than Ideal filter and it is well known tool for removing Gaussian noise [2].

$$G(k_x, k_y) = e^{-({k_x}^2+{k_y}^2)/2\sigma^2} \tag{1}$$

After putting a mask on the frequency domain, the spectrum is decentralised and Inverse Fast Fourier Transformation is performed so that the frequencies with the mask can be converted into the filtered image.

More importantly, the filtering result could be improved depending on the diameter σ of Gaussian Low Pass Filter. Thus, FFT process is iterated with different size of mask to find the best parameter.

For further post-processing in Fig. 1., Wiener filter is implemented as a deconvolution tool so that the blurred edge can be clear.

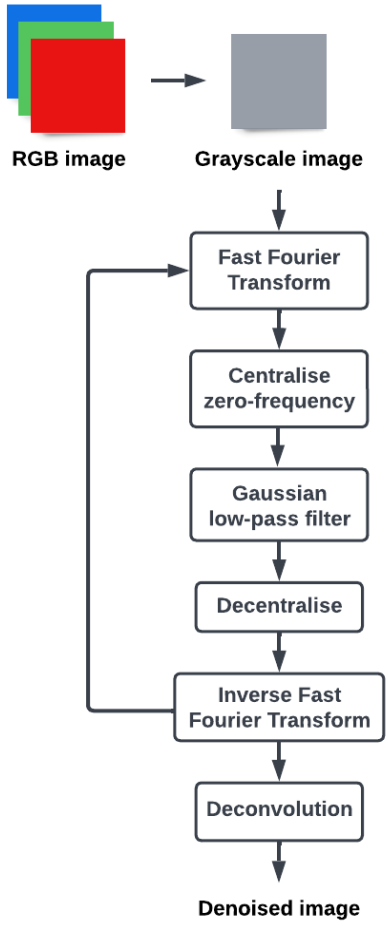


Fig. 1. Flowchart of Denoising Algorithm

Tools

- **OpenCV**
  OpenCV is an open source library for computer vision and machine learning software [5]. `cvtColor()` function from OpenCV is used in pre-processing step for converting RGB image into grayscale image. Also, it has a couple of existing denoising filters such as Mean Filter and Median Filter. To be specific, Mean Filter is created using `filter2D()` with the mean kernel and Median Filter is used by `medianBlur()` function for denoising.

- **Scikit-image**
  Scikit-image is an open source library for image processing [6]. `imread_collection()` function is used in dataset preparation stage. It helps to save time by loading multiple images at once instead of reading 75 images manually. Additionally, scikit-image has several metrics function such as Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM), which helps to evaluate the

filtered results. Also, VisuShrink approach is employed by `denoise_wavelet()` function.

- **NumPy**
  NumPy enables numerical computing with Python [7]. `fft2()`, `fftshift()`, `ifftshift()` and `ifft2()` functions are computed for implementing FFT. Not only that, NumPy creates a new array to compute the mask – Gaussian Low Pass Filter. Also, it can compute array to process image from 0-1 to 0-255.

- **Matplotlib**
  Matplotlib provides a comprehensive tool for creating static and interactive visualisations [8]. It is used for showing the images, spectrum and result of filtered images in this paper. Also, it helps to plot graphs for comparing filters' performance.

Results & Evaluation

Pre-processing

As mentioned earlier, the entire image dataset – original, noisy 10, noisy 25, noisy 50 – have been converted from colour images to grayscale images. Fig.2. shows the changes of one image among 25 samples. Also, the whole dataset in grayscale can be found in Appendix A.

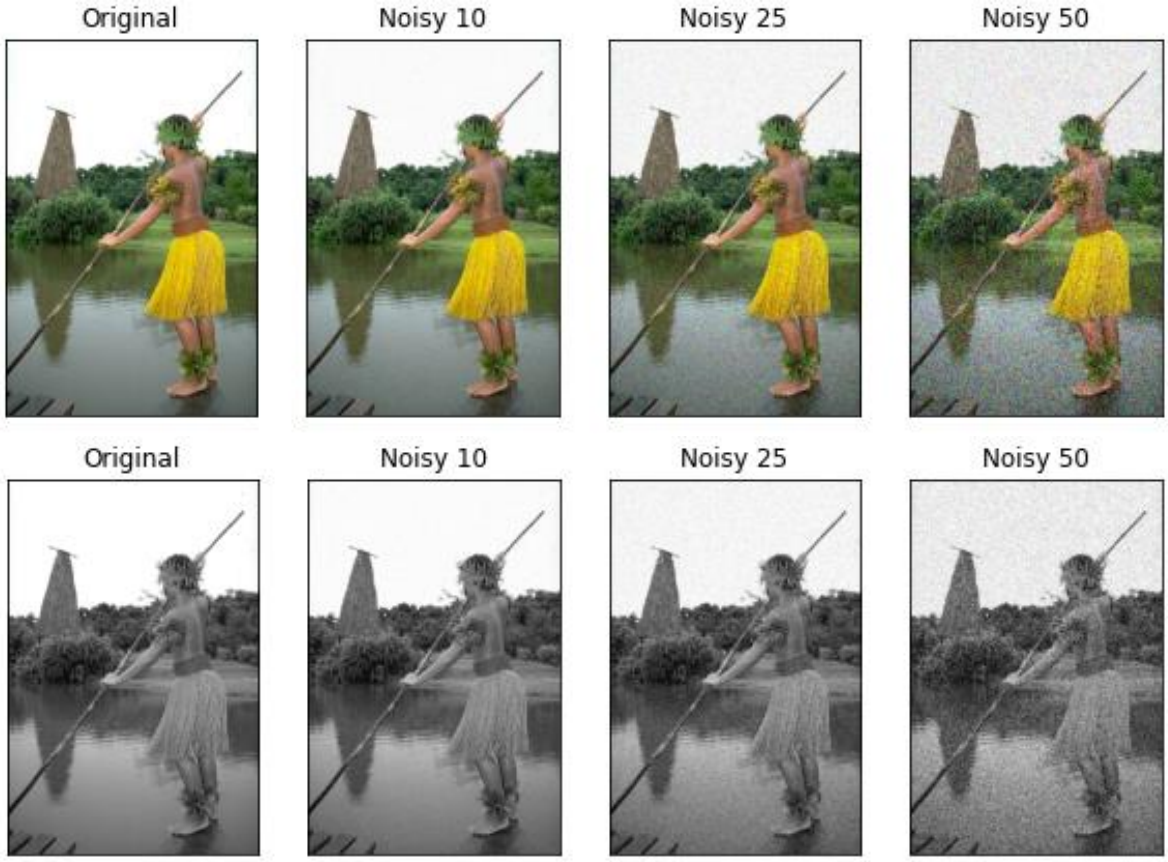


Fig. 2. RGB image (Top) and Grayscale image (Bottom)

Fast Fourier Transformation

To find the best performance of FFT, parameter tuning has been conducted on Gaussian Low Pass Filter. Noisy 10 has been tried with the different value of σ in the range of $\sigma = 65$ to $\sigma = 135$, Noisy 25 from $\sigma = 30$ to $\sigma = 100$ and Noisy 50 from $\sigma = 5$ to $\sigma = 75$. As Fig.3. indicated, even though MSE is decreasing continuously, SSIM starts to decrease at some point. Therefore, the best parameter would be the peak point of SSIM before it decreases.

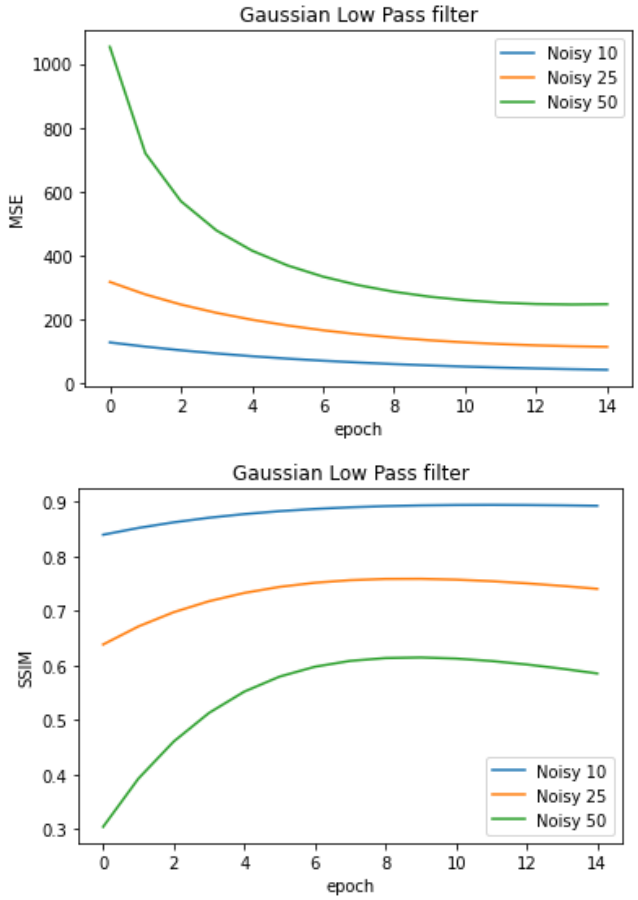


Fig. 3. Parameter tuning of MSE (Top) and SSIM (Bottom)

After iteration of FFT, the best parameter of Gaussian Low Pass Filter is figured as follows:

- Noisy 10: $\sigma = 135$
- Noisy 25: $\sigma = 80$
- Noisy 50: $\sigma = 60$

Since each image dataset contains 25 images, the performance is determined by the average score of MSE and SSIM.

Fig.4. is one of the sample images with noise 10. Since it has a little noise, FFT denoises well without damaging the image. Furthermore, according to Fig.5., FFT still reduces the noise well with Noisy 25 image even though the edges become blurred a little. However, unlike Noisy 10 and 25 datasets, FFT performs the least with Noisy 50 dataset due to the high level of noise (Fig. 6.). Also, although FFT successfully reduce the noise, the image becomes blurred.

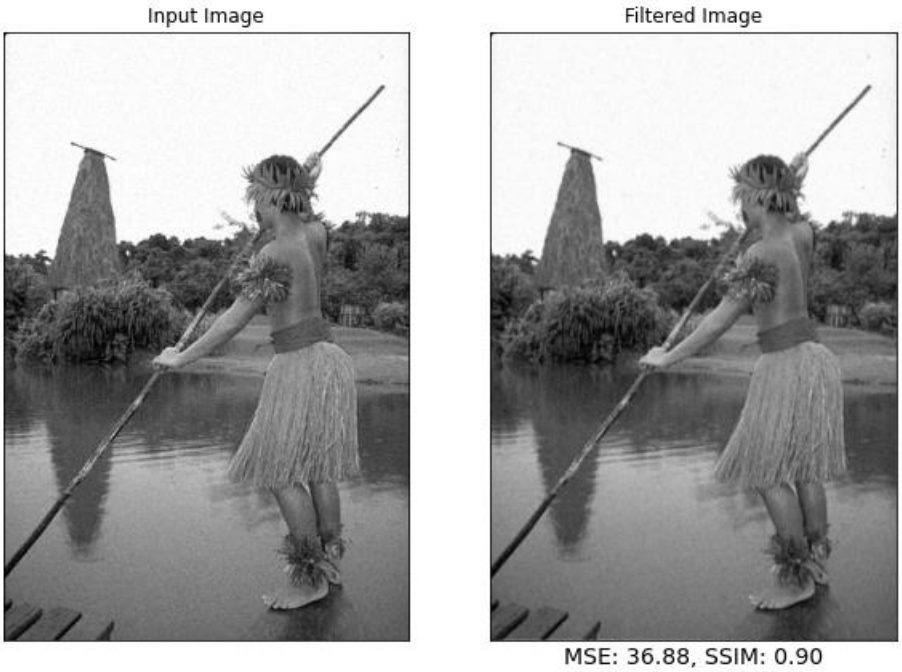


Fig. 4. Noisy 10 image (Left) and after FFT (Right)

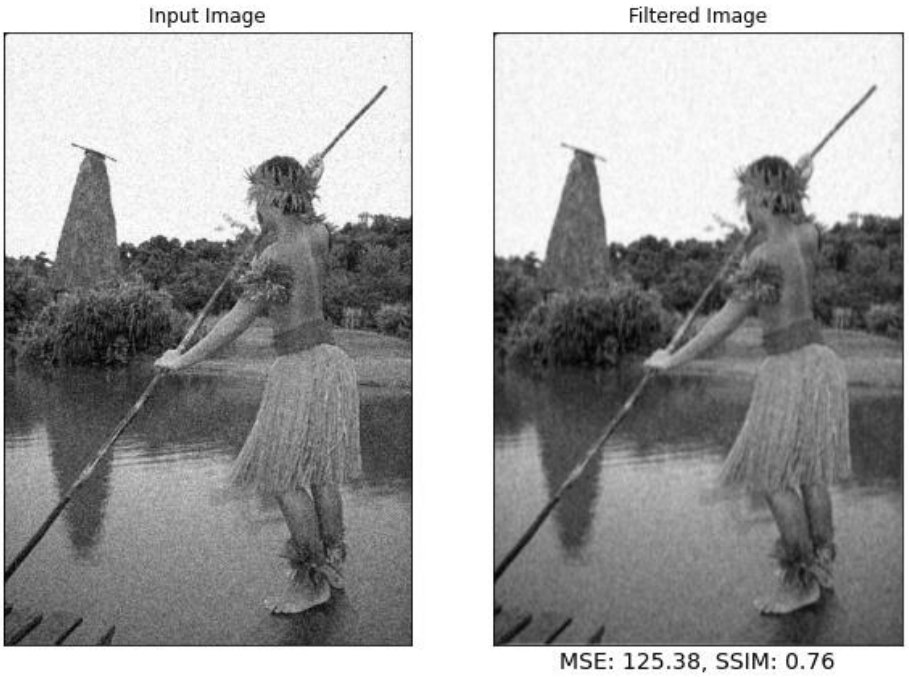


Fig. 5. Noisy 25 image (Left) and after FFT (Right)

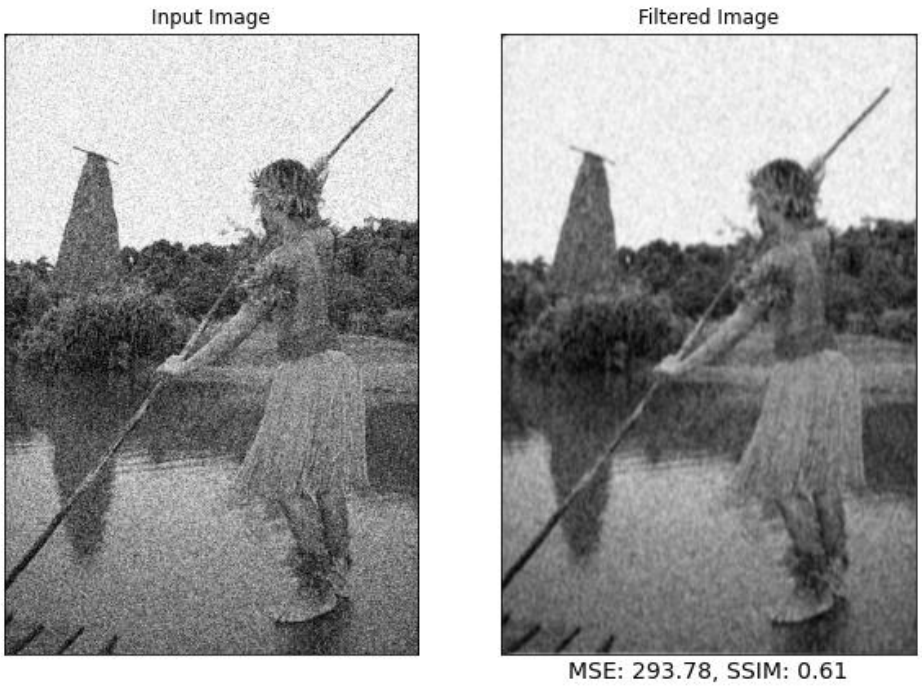


Fig. 6. Noisy 50 image (Left) and after FFT (Right)

Post-processing

Since the Gaussian Low Pass Filter blurred edges, Wiener Filter is applied for deconvolution – to deblur the images.

However, after applying Wiener Filter, the performance gets lower than before with all noisy level as Table 1 and Table 2 displayed. According to Appendix C, after applying Wiener Filter, although it deblurs the image, the brightness is changed for all. It is doubted that the changes of brightness might decrease the evaluation metrics score.

Table I: MSE with and without Deconvolution

	Noisy 10	**Noisy 25**	**Noisy 50**
FFT	41.15	126.93	251.53
FFT + Deconvolution	393.34	364.09	413.04

Table II: SSIM with and without Deconvolution

	Noisy 10	**Noisy 25**	**Noisy 50**
FFT	0.89	0.76	0.61
FFT + Deconvolution	0.63	0.59	0.52

Performance Comparison

To compare the performance of proposed algorithms, other filters has been conducted – Mean Filter, Median Filter and Wavelet Denoising. Especially, VisuShrink is implemented as Wavelet Denoising method because it is known as its high accuracy when denoising Gaussian Noise.

According to Fig.7. and Fig.8., VisuShrink and FFT appears to have the highest accuracy with Noisy10 and FFT exhibits its highest performance with Noisy 25 and Noisy 50 as well. Especially, unlike the expectation, it is interesting that the Wiener Filter decreases the FFT's performance and it becomes the lowest in general. It seems like FFT is the best filter among all and VisuShrink is the second although it is the weakest filter with high level of noise.

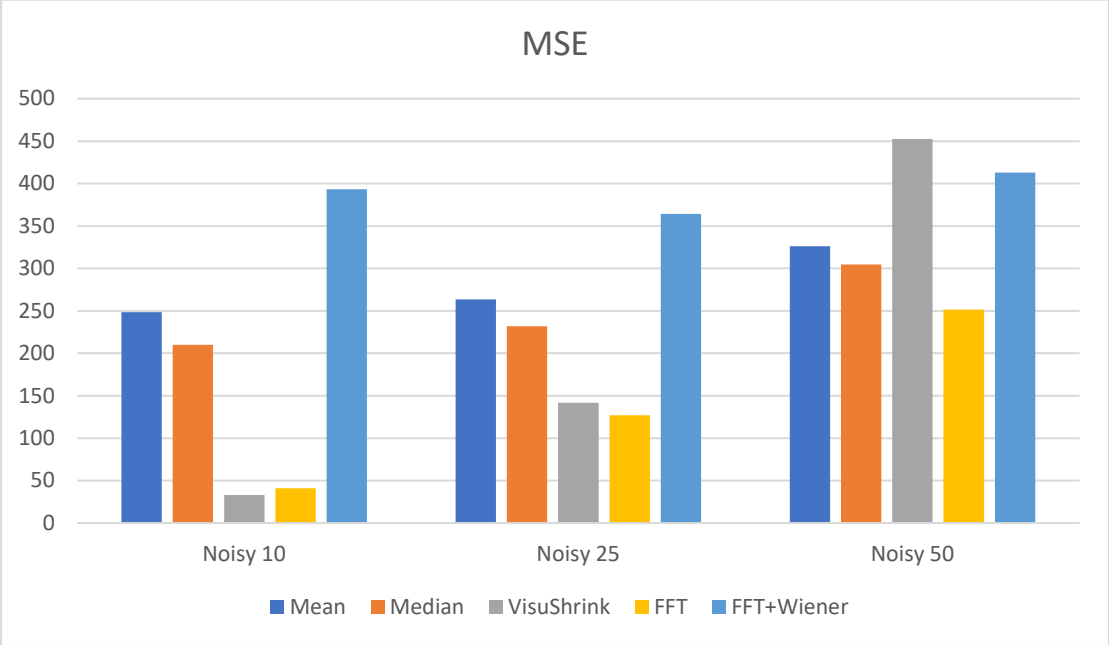


Fig. 7. Comparison of MSE metrics

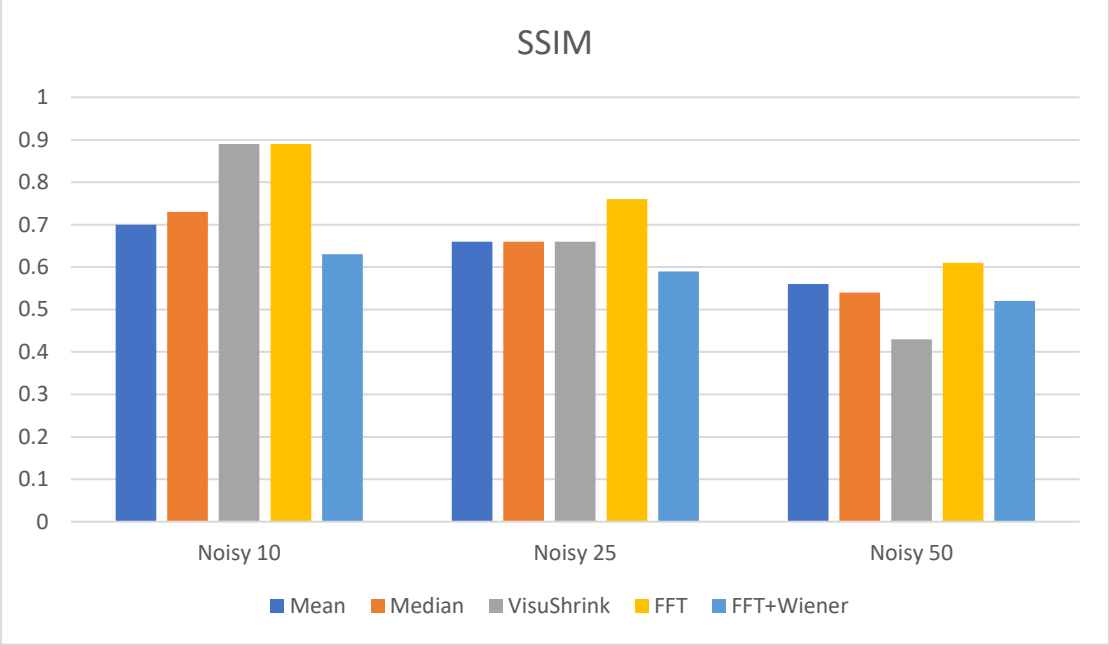


Fig. 8. Comparison of SSIM metrics

Conclusion

The proposed denoising algorithm successfully shows its highest performance compared to other methods such as Mean Filter, Median Filter and VisuShrink. However, it is recommended that the algorithm is better without deconvolution.

For the future work, it is suggested that trying denoising with colour image as well. Since the input image is RGB, it is recommended to split them in three individual channel and merge them after filtering to get the RGB as an output. In addition, as there might be no original image in real problem, it is suggested to use more sample images to compare filter's improvements.

References

[1] D. Martin, C. Fowlkes , D. Tal and J. Malik, "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics," *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference,* vol. 2, p. 416–423, 2001.

[2] S. Suryanarayana, B. Deekshatulu, K. Lal Kishore and Y. Rakesh Kumar, "Estimation and removal of Gaussian noise in digital images," *International Journal of Electronics and Communication Engineering,* vol. 5, no. 1, pp. 23-33, 2012.

[3] Z. Liu, L. Li and H. Li, "Application Research on Sparse Fast Fourier Transform Algorithm in White Gaussian Noise," *Procedia Computer Science,* vol. 107, pp. 802-807, 2017.

[4] M. S. Lambert, T. T. Miriam and F. M. Susan, Wiener Filter: Norbert Wiener, Noise, Andrey Kolmogorov, Frequency Response, Stochastic Process, Cross-correlation, Deconvolution, Wiener Deconvolution, Expected Value, Quantization Error, Betascript Publishing, 2010.

[5] "About," OpenCV, 2023. [Online]. Available: https://opencv.org/about/. [Accessed 3 Jan 2023].

[6] "scikit-image 0.19.2 docs," scikit-image, 2023. [Online]. Available: https://scikit-image.org/docs/stable/index.html. [Accessed 3 Jan 2023].

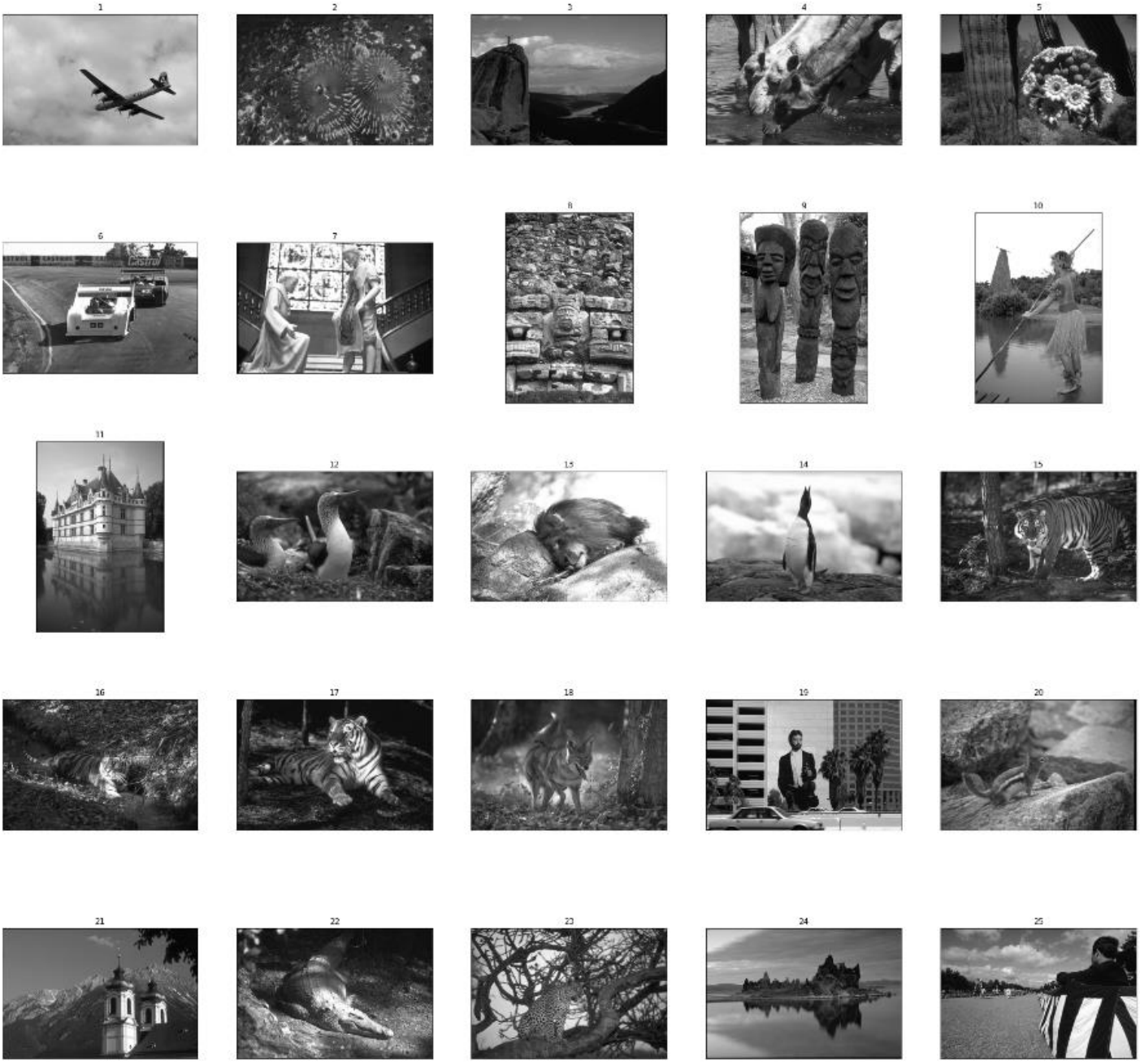
[7] "About Us," NumPy, 2023. [Online]. Available: https://numpy.org/about/. [Accessed 3 Jan 2023].

[8] "Matplotlib: Visualization with Python," Matplotlib, 2023. [Online]. Available: https://matplotlib.org/. [Accessed 3 Jan 2023].

Appendices

Appendix A

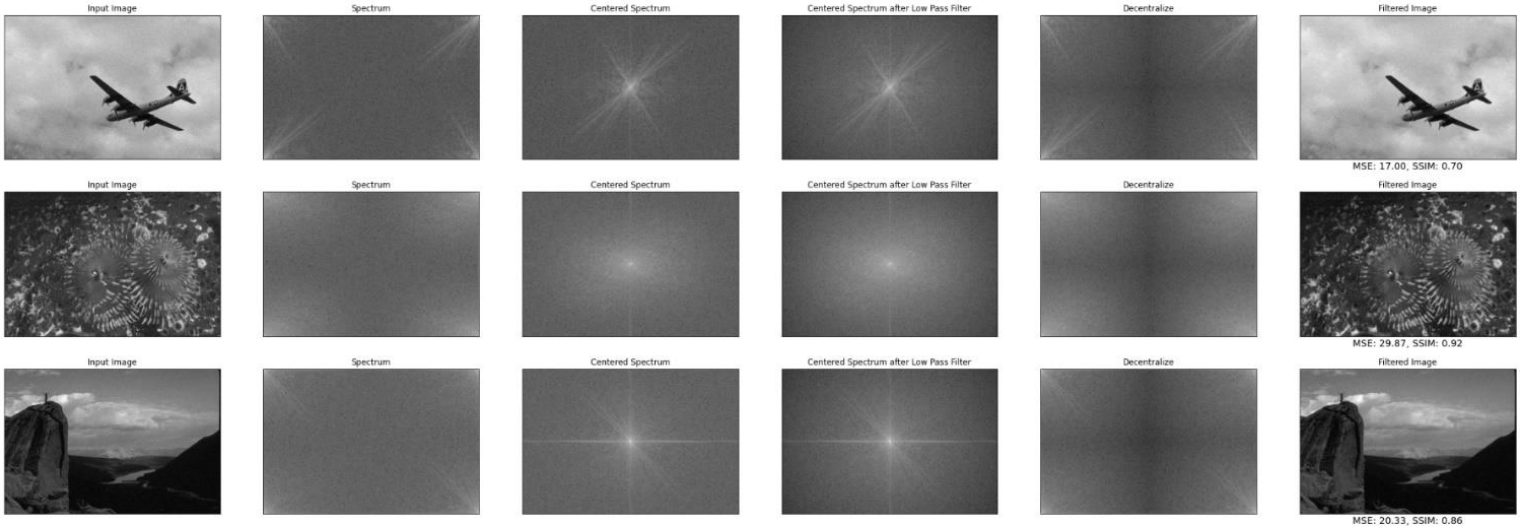
Grayscale images of 25 original samples



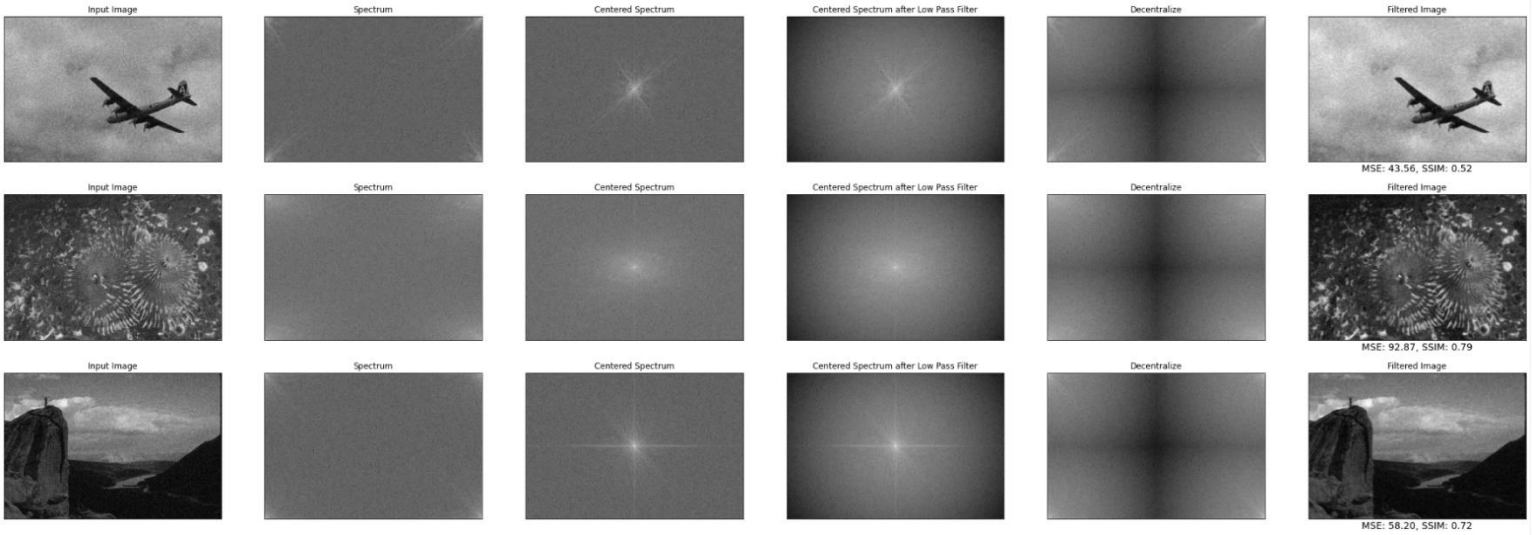
Appendix B

The process of Fast Fourier Transformation

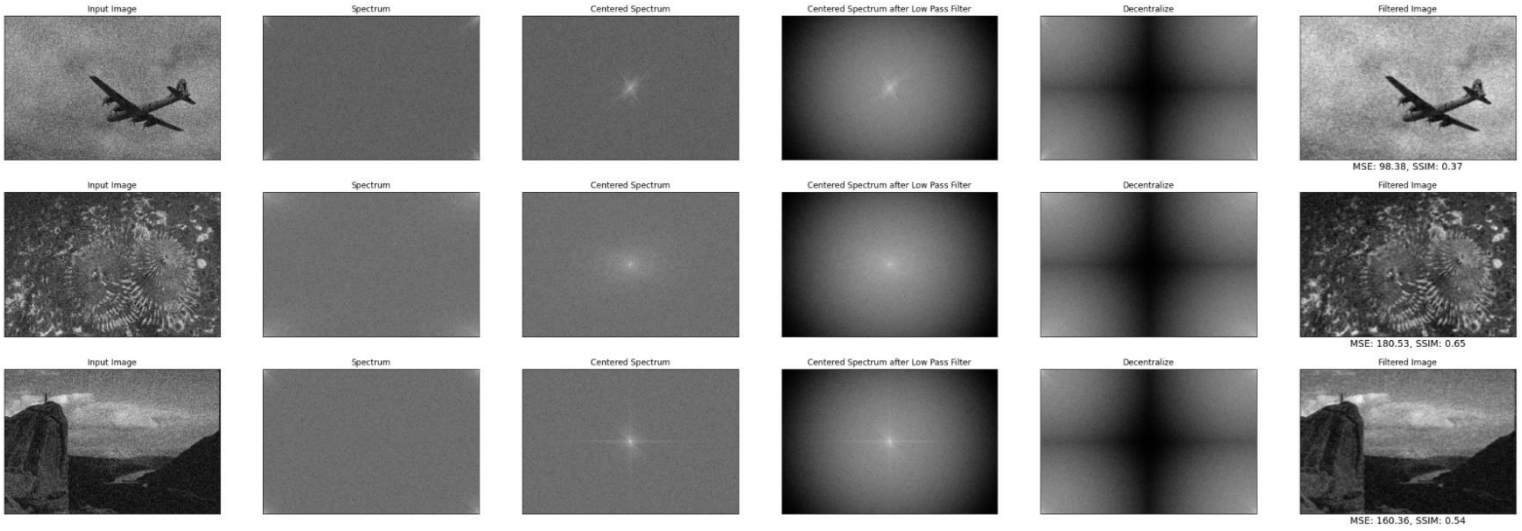
Noisy 10, $\sigma = 135$



Noisy 25, $\sigma = 80$



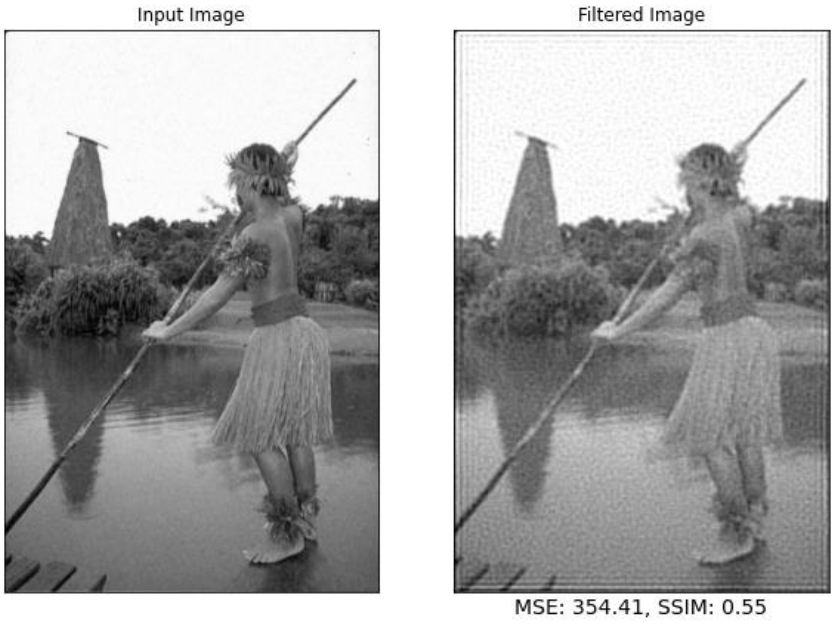
Noisy 50, $\sigma = 60$



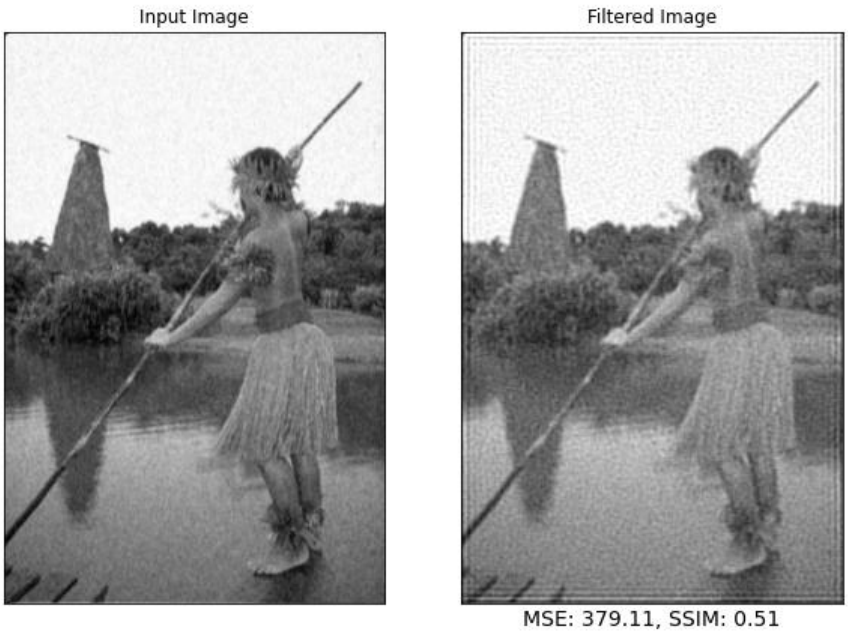
Appendix C

Filtered images after deconvolution

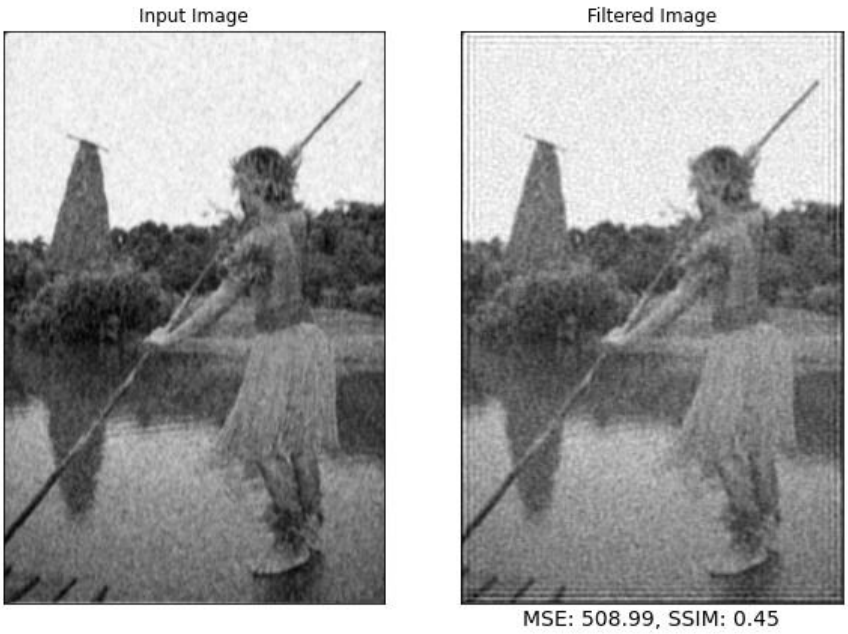
Noisy 10



Noisy 25



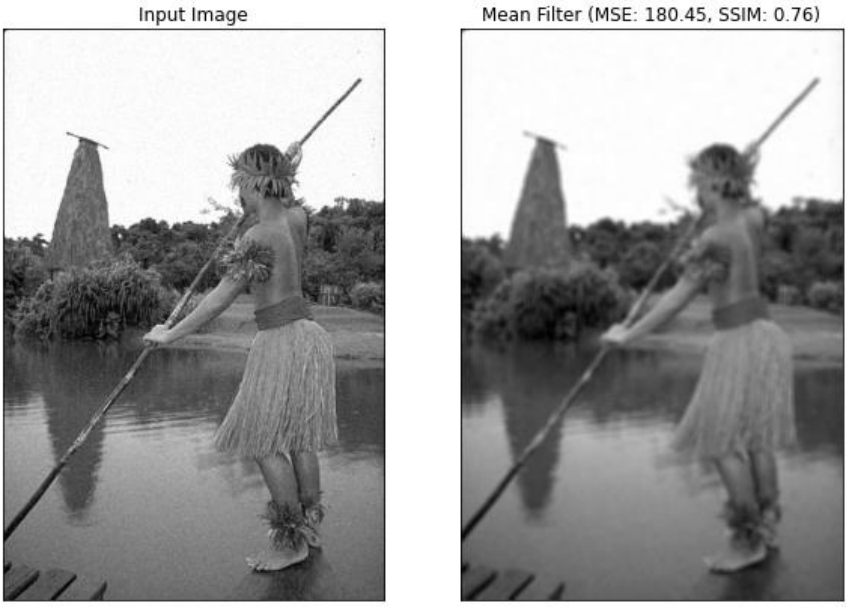
Noisy 50



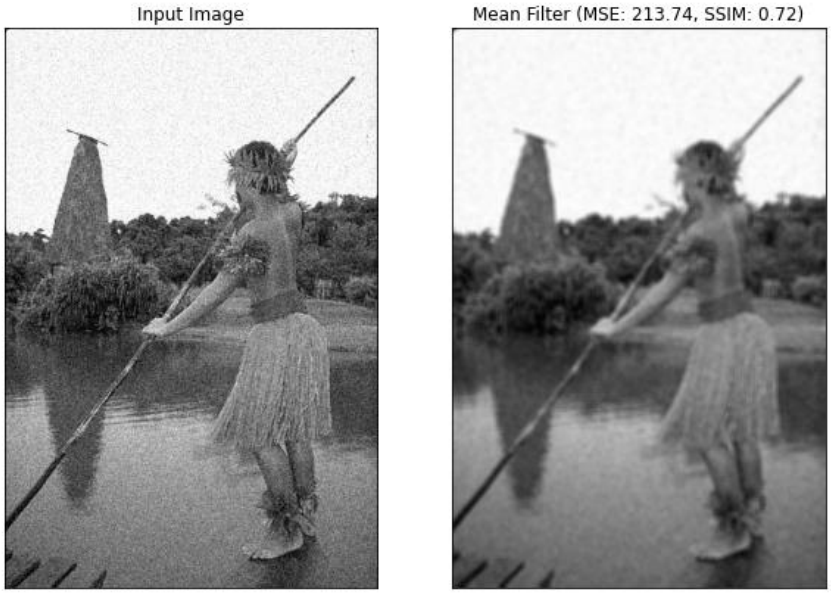
Appendix D

Mean Filter

Noisy 10



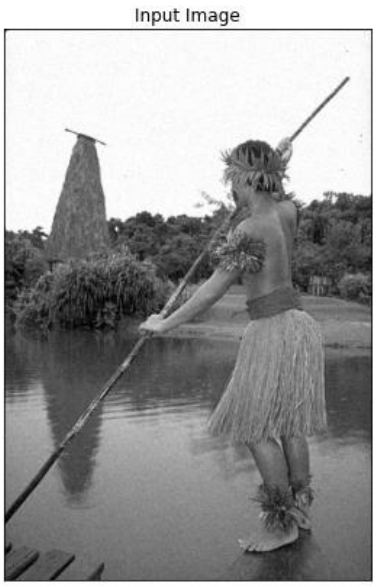
Noisy 25



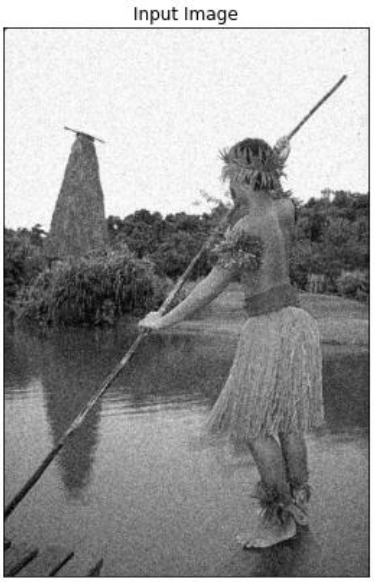
Noisy 50

Median Filter

Noisy 10



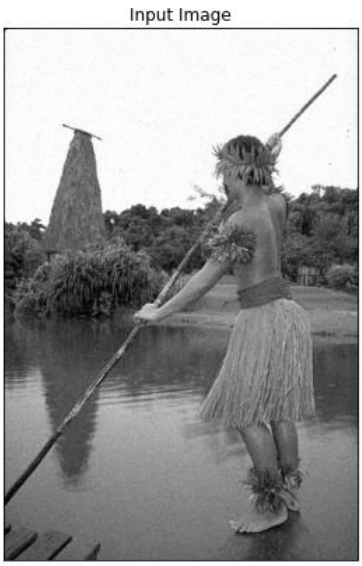
Noisy 25



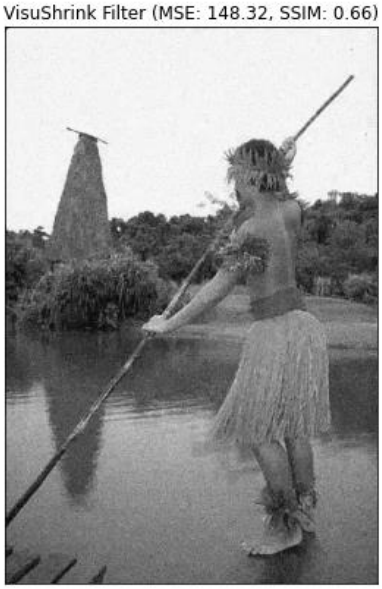
Noisy 50

VisuShrink

Noisy 10



Noisy 25



Noisy 50

Input Image

VisuShrink Filter (MSE: 507.44, SSIM: 0.39)