Image Denoising using Wavelet Transform

Megha S Lekshmi, M241085EC
M.Tech in Signal Processing
Electronics and Communication Engineering
National Institute of Technology Calicut
Calicut, India
megha_m241085ec@nitc.ac.in

Haneena Sajan, M241294EC
M.Tech in Signal Processing
Electronics and Communication Engineering
National Institute of Technology Calicut
Calicut, India
haneena_m241294ec@nitc.ac.in

Introduction

Image denoising is a fundamental problem in image processing, aimed at removing noise from images while preserving important details and structures. Images captured in real world are subjected to noise due to environment, signal instability, camera sensor issues, poor lighting conditions, electrical loss etc. For further processing these images and to interpret results it is essential to have images with noise as low as possible. Noise can degrade image quality, making it difficult to analyze or interpret visual data. In applications such as medical imaging, satellite imaging, and computer vision, effective denoising techniques are crucial for accurate decision-making. Traditional denoising methods often struggle to balance noise removal and detail preservation, leading to residual noise or excessive blurring.

Motivation

The increasing demand for high-quality image processing has driven advancements in various fields, including medical imaging, satellite imaging, and computer vision. In medical imaging, noise can obscure critical details, potentially leading to misdiagnosis. Similarly, in satellite imaging, noise can degrade image quality, affecting the accurate analysis of environmental changes. Effective denoising techniques are crucial for preserving important features while reducing unwanted distortions.

Traditional denoising approaches include various filtering techniques. Smoothing filters, such as the Mean and Gaussian filters, reduce noise but often blur fine details. Sharpening filters, like the Laplacian and High-pass filters, enhance edges but may amplify noise. Edge detection filters, such as the Sobel, Prewitt, and Canny Edge Detector, emphasize boundaries but can be highly sensitive to noise. While these methods offer certain advantages, they often involve trade-offs between noise reduction and detail preservation.

To address these limitations, wavelet transform-based denoising has emerged as a powerful technique. Wavelet transforms provide a multi-resolution representation of images, enabling effective noise suppression while preserving essential features. By decomposing an image into different frequency components, wavelet-based methods can selectively attenuate noise without compromising structural details.

Problem Statement

The primary problem addressed is the removal of noise from images while preserving important details and structures. Noise can arise from various sources, such as sensor imperfections, transmission errors, or environmental factors, and it can significantly degrade the quality of images. Traditional denoising methods often face a trade-off between effectively removing noise and maintaining image details. For instance, excessive smoothing can blur edges and textures, while insufficient denoising may leave residual noise, resulting in poor image quality. The challenge lies in developing a denoising pipeline that can effectively eliminate noise without compromising the integrity of critical image features.

Wavelet-based thresholding methods, namely BayesShrink and VisuShrink, are explored for denoising noisy images. Gaussian noise is introduced to an input image, and the effectiveness of the denoising techniques is evaluated in terms of their ability to recover the original image. Performance comparison is analyzed using Peak Signal-to-Noise Ratio (PSNR), which serves as a key metric for quantifying image quality. The goal is to assess and compare the performance of these methods in balancing noise reduction with the preservation of important image details.

Literature Survey

Ruikar and Doye (2010) in [1] proposed a novel wavelet-based thresholding approach for image denoising, comparing its performance against established techniques such as VisuShrink, SureShrink, BayesShrink, Normal Shrink, and Universal threshold. The method applies discrete wavelet transform (DWT) and introduces an adaptive thresholding function to suppress noise while preserving key image features. Using SNR as the evaluation metric on the Lena image corrupted with Gaussian, Speckle, Salt Pepper, and Poisson noise, the results showed SureShrink and Normal Shrink performed best for Gaussian noise, while BayesShrink, Normal Shrink, and SureShrink were most effective for Poisson noise. For Salt Pepper and Speckle noise, BayesShrink and Normal Shrink delivered superior results. VisuShrink and Universal thresholding underperformed, especially with Salt Pepper and multiplicative noise. The proposed method consistently enhanced SNR, particularly in block-based denoising, showing robustness across noise types. The study also suggests future work should assess computational complexity alongside denoising quality.

In [2], various types of thresholding methods are experimented. VisuShrink, SureShrink, BayesShrink, and NeighShrink are some of the well-known threshold selection techniques that further refine performance. VisuShrink uses a universal threshold and often leads to over-smoothing. SureShrink minimizes Stein's Unbiased Risk Estimator to balance noise reduction and detail preservation. BayesShrink utilizes Bayesian estimation, assuming a generalized Gaussian distribution for wavelet coefficients, while NeighShrink considers neighboring coefficients for better contextual denoising.

Methodology

The proposed denoising pipeline consists of several steps. The steps are outlined follows:

1) **Image Loading and Preprocessing**: A clean image is loaded into memory using io.imread() from the skimage library. The image is converted into a floating-point format using img_as_float() to standardise the image's pixel values in the range [0,1].

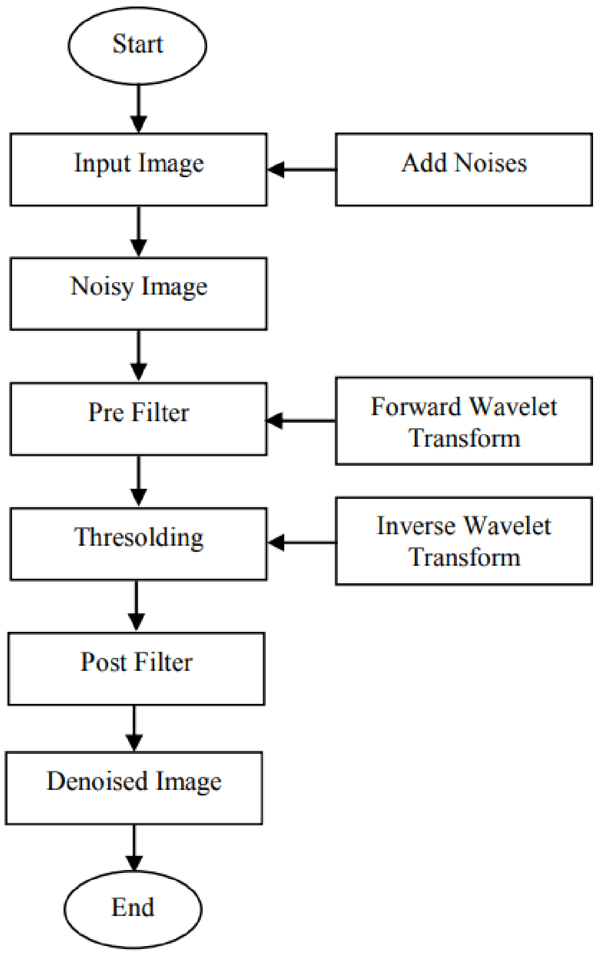

Fig. 1. Implementation flow diagram

2) **Noise Addition**: The image is added with Gaussian noise with a specified noise level. The noise is applied in such a way that it follow a Gaussian distribution.
3) **Wavelet Decomposition**: In wavelet decomposition, we break down an image into different frequency components at multiple resolutions. Unlike traditional Fourier transforms which only provide frequency information, wavelet decomposition gives both spatial and frequency information. This makes it ideal for analyzing images that contain localized features such as edges, textures, and noise. Wavelet decomposition works by applying two types of filters to the image: a low-pass filter and a high-pass filter. These filters are applied separately along the rows and columns of the image. After filtering, the results are downsampled (every second pixel is kept), and this creates four sub-band images:
   - **LL(Low-Low)**: This sub-band contains the approximation of the original image(smooth areas). It is of smaller resolution compared to the original image.
   - **LH(Low-High)**: This sub-band captures the horizontal edges details of the image.
   - **HL(High-Low)**: This sub-band captures the vertical edges details of the image.
   - **HL(High-Low)**: This sub-band captures the diagonal details and the high-frequency noise.

   This first step is known as level-1 decomposition. For multi-level decomposition, the LL sub-band is decomposed further to analyze the image at coarser scales. At each level, new LH, HL, and HH sub-bands are generated, allowing for a hierarchical analysis of the image. In this implementation, we have experimented with level 1 and level 2 decompositions. This multi-resolution representation is a key strength of wavelets.

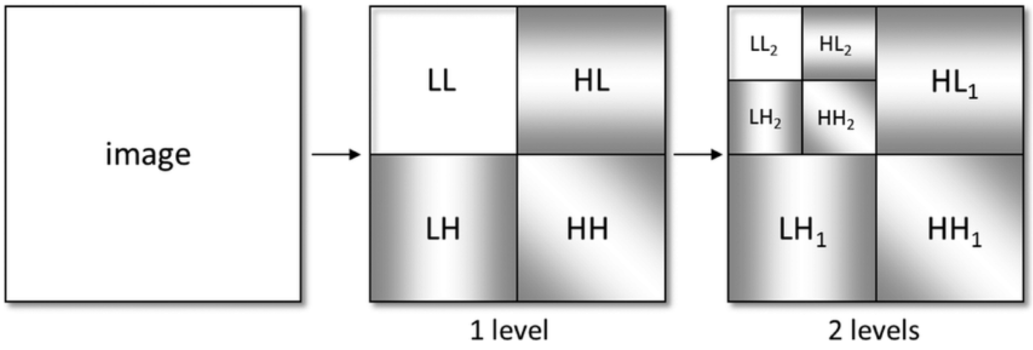

Fig. 2. Level-2 wavelet decomposition

Wavelet transforms are used to decompose signals into components at various scales and resolutions. The primary types of wavelet transforms include the following:

- **Continuous Wavelet Transform (CWT)**: Analyzes the signal on every possible scale and translation, offering a detailed time-frequency localization. CWT is used primarily for theoretical analysis.
- **Discrete Wavelet Transform (DWT)**: Decomposes signals at dyadic (power-of-two) scales. It is efficient and widely used for signal denoising, compression, and feature extraction.
- **Stationary Wavelet Transform (SWT)**: A redundant, translation-invariant version of the DWT. It does not downsample the signal and is used in applications like denoising and edge detection.
- **Wavelet Packet Transform (WPT)**: An extension of DWT that decomposes both approximation and detail coefficients. It offers a more detailed analysis of signals.

The Haar Wavelet Transform: It is the simplest and most intuitive form of wavelet transform. Introduced by Alfréd Haar in 1909, it serves as the foundation for modern wavelet theory. Despite its simplicity, the Haar wavelet remains relevant due to its computational efficiency and effectiveness in applications that involve abrupt changes or edges.
The Haar wavelet transform operates by computing averages and differences of adjacent data points, capturing both the low-frequency (approximation) and high-frequency (detail) components of a signal. This allows for multi-resolution analysis, where a signal is examined at different levels of detail.
The scaling function $\phi(t)$ and the wavelet function $\psi(t)$ for the Haar transform are defined as:

$$\phi(t) = \begin{cases} 1, & 0 \leq t < 1 \\ 0, & \text{otherwise} \end{cases} \qquad \psi(t) = \begin{cases} 1, & 0 \leq t < \frac{1}{2} \\ -1, & \frac{1}{2} \leq t < 1 \\ 0, & \text{otherwise} \end{cases}$$

This function is used in the filtering process where the signal is passed through a low-pass filter to obtain approximation coefficients and a high-pass filter to obtain detail coefficients. The resulting signal is then downsampled by a factor of

two at each decomposition level.
The key strength of the Haar wavelet is its ability to detect and represent sudden changes in signals. It is highly effective in edge detection and simple image compression.

4) **Estimate noise standard deviation**: The standard deviation of the noisy image is found out from the HH1 sub-band(high-high detail coefficients at the first level of the wavelet decomposition) because this part of the image that mostly contains fine details and noise. This is done using the Median Absolute Deviation(MAD) method. The median is more robust to outliers and the factor 0.6745 come from the properties of the normal distribution. It rescales the median to match the standard deviation for Gaussian noise.

$$\sigma_n = \frac{\text{median}(|HH1|)}{0.6745}$$

So,

$$\text{Noise Variance} = \sigma_n^2$$

5) **Wavelet Denoising**: The wavelet coefficients suggest that small coefficients are dominated by noise; coefficients with a maximum absolute value carry more signal information than noise.
In order to de-noise any signal, we need to put the noisy signal into the decomposition process by applying wavelet transform. Understanding how signal behaves in different frequency segments will allow us to select the most suitable threshold in the next stage. The next step is determining the best threshold values and applying threshold values to these set of coefficients so it is possible to eliminate unwanted data. Final step of the algorithm is where we use these filtered coefficient sets to recompose our signal. For this to be possible an inverse discrete Wavelet transform (IDWT) is done.

Here we use two wavelet denoising methods:

- **BayesShrink**: A method that adapts the shrinkage function based on a Bayesian criterion to reduce noise while preserving important image details.

$$T_{\text{BayesShrink}} = \frac{\sigma^2}{\hat{\sigma}_X} \tag{1}$$

where σ^2 is the estimated noise variance and $\hat{\sigma}_X$ is the standard deviation of the wavelet coefficients of the noisy image.

- **VisuShrink**: This method applies a universal threshold based on the noise level (sigma) to shrink the wavelet coefficients. VisuShrink is thresholding by applying the Universal threshold proposed by Donoho and Johnstone

$$T_{\text{VisuShrink}} = \sigma\sqrt{2\log N} \tag{2}$$

where σ is the estimated noise standard deviation and N is the total number of pixels in the image.

6) **Thresholding**: The most important thing in applications of thresholding is the determination of threshold value.

Thresholding can be applied in two different methods: hard and soft thresholding. Hard threshold can be explained as setting elements to zero where their absolute values are lower than the threshold. Hard threshold gives sharper results. On the other hand, soft thresholding softens the coefficients exceeding the threshold by lowering them as much as threshold value. Soft thresholding requires more computations but gives better denoising performance.

7) **Image Quality Evaluation**: We evaluate the obtained outputs using Peak Signal-to-Noise Ratio(PSNR). It is computed for the noisy image and each denoised images. PSNR is a common metric for image quality that compares the denoised image to the original clean image. Higher PSNR values indicate better quality

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{3}$$

where MAX is the maximum pixel value and MSE is the Mean Squared Error between the original and the processed image.

The pipeline is implemented in Python, using libraries such as OpenCV, PyWavelets, and SciPy. The effectiveness of the method is evaluated using both synthetic and real-world images.

Code

```python
import cv2
import numpy as np
import pywt
import matplotlib.pyplot as plt

# Read grayscale image
img = cv2.imread('lena.png', cv2.IMREAD_GRAYSCALE)

# 2D Haar wavelet decomposition (single-level)
cA, (cH, cV, cD) = pywt.dwt2(img, 'haar')

# Plot the results
titles = ["Input Image", "Approximation", "Horizontal Detail", "Vertical Detail", "Diagonal 
    Detail"]
components = [img, cA, cH, cV, cD]

fig, axs = plt.subplots(1, 5, figsize=(16, 4))
for ax, data, title in zip(axs, components, titles):
    ax.imshow(data, cmap='gray')
    ax.set_title(title, fontsize=10)
    ax.axis('off')

plt.tight_layout()
plt.show()

import cv2
import pywt
import matplotlib.pyplot as plt

# Load grayscale image
img = cv2.imread('lena.png', cv2.IMREAD_GRAYSCALE)

# 2-level wavelet decomposition using Haar wavelet
coeffs = pywt.wavedec2(img, 'haar', level=2)
lowpass, (lh2, hl2, hh2), _ = coeffs  # Extract second-level components

# Titles and images to show
titles = ['Input Image', 'Approx (Level 2)', 'Horiz Detail (L2)', 'Vert Detail (L2)', 'Diag 
    Detail (L2)']
parts = [img, lowpass, lh2, hl2, hh2]

# Plot
fig, axs = plt.subplots(1, 5, figsize=(16, 4))
for ax, data, label in zip(axs, parts, titles):
    ax.imshow(data, cmap='gray')
    ax.set_title(label, fontsize=10)
    ax.axis('off')

plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt
from skimage import io, img_as_float
from skimage.util import random_noise
from skimage.restoration import denoise_wavelet, estimate_sigma
from skimage.metrics import peak_signal_noise_ratio
import numpy as np

# Load grayscale image from path
img_path = input("Enter image path: ")
img = img_as_float(io.imread(img_path, as_gray=True))

# Add Gaussian noise
std_dev = 0.12
noisy_img = random_noise(img, var=std_dev**2)
```

```

# Estimate noise level
sigma_hat = estimate_sigma(noisy_img, average_sigmas=True)
print(f"Estimated   = {sigma_hat:.4f}")

# Denoising using wavelet methods
bayes_soft = denoise_wavelet(noisy_img, wavelet='haar', wavelet_levels=3,
                              method='BayesShrink', mode='soft', rescale_sigma=True)

bayes_hard = denoise_wavelet(noisy_img, wavelet='haar', wavelet_levels=3,
                              method='BayesShrink', mode='hard', rescale_sigma=True)

visu_soft = denoise_wavelet(noisy_img, wavelet='haar', wavelet_levels=3,
                            method='VisuShrink', mode='soft', sigma=sigma_hat, rescale_sigma=
                                True)

visu_hard = denoise_wavelet(noisy_img, wavelet='haar', wavelet_levels=3,
                            method='VisuShrink', mode='hard', sigma=sigma_hat, rescale_sigma=
                                True)

visu_half = denoise_wavelet(noisy_img, wavelet='haar', wavelet_levels=3,
                            method='VisuShrink', mode='soft', sigma=sigma_hat/2, rescale_sigma=
                                True)

visu_quarter = denoise_wavelet(noisy_img, wavelet='haar', wavelet_levels=3,
                               method='VisuShrink', mode='soft', sigma=sigma_hat/4,
                                   rescale_sigma=True)

# PSNR values
psnr_vals = {
    "Noisy": peak_signal_noise_ratio(img, noisy_img),
    "Bayes (Soft)": peak_signal_noise_ratio(img, bayes_soft),
    "Bayes (Hard)": peak_signal_noise_ratio(img, bayes_hard),
    "Visu (Soft)": peak_signal_noise_ratio(img, visu_soft),
    "Visu (Hard)": peak_signal_noise_ratio(img, visu_hard),
    "Visu (  /2)": peak_signal_noise_ratio(img, visu_half),
    "Visu (  /4)": peak_signal_noise_ratio(img, visu_quarter),
}

# Panel view 1
fig, axes = plt.subplots(2, 3, figsize=(10, 6))
plt.gray()

images1 = [noisy_img, bayes_soft, visu_soft, img, visu_half, visu_quarter]
titles1 = [
    f"Noisy\nPSNR={psnr_vals['Noisy']:.2f} dB",
    f"Bayes (Soft)\nPSNR={psnr_vals['Bayes (Soft)']:.2f} dB",
    f"Visu (Soft)\nPSNR={psnr_vals['Visu (Soft)']:.2f} dB",
    "Original Image",
    f"Visu (  /2)\nPSNR={psnr_vals['Visu (  /2)']:.2f} dB",
    f"Visu (  /4)\nPSNR={psnr_vals['Visu (  /4)']:.2f} dB",
]

for ax, img_show, title in zip(axes.flat, images1, titles1):
    ax.imshow(img_show)
    ax.set_title(title)
    ax.axis('off')

fig.tight_layout()
plt.show()

# Panel view 2
fig2, axes2 = plt.subplots(1, 5, figsize=(15, 4))
plt.gray()

images2 = [noisy_img, bayes_soft, bayes_hard, visu_soft, visu_hard]
titles2 = [
    "Noisy␣Image",
    f"Bayes␣(Soft)\nPSNR={psnr_vals['Bayes␣(Soft)']:.2f}",
    f"Bayes␣(Hard)\nPSNR={psnr_vals['Bayes␣(Hard)']:.2f}",
    f"Visu␣(Soft)\nPSNR={psnr_vals['Visu␣(Soft)']:.2f}",
    f"Visu␣(Hard)\nPSNR={psnr_vals['Visu␣(Hard)']:.2f}",
]

for ax, im, title in zip(axes2, images2, titles2):
    ax.imshow(im)
    ax.set_title(title)
    ax.axis('off')

fig2.tight_layout()
plt.show()

# Bar plot
labels = list(psnr_vals.keys())
values = list(psnr_vals.values())

plt.figure(figsize=(10, 6))
bars = plt.bar(labels, values, color='navy', edgecolor='black')
plt.ylabel("PSNR␣(dB)")
plt.title("Denoising␣PSNR␣Comparison")
plt.xticks(rotation=20)
plt.ylim(min(values) - 1, max(values) + 2)
plt.grid(axis='y', linestyle='--', alpha=0.5)

for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 0.2, f"{height:.2f}", ha='center')

plt.tight_layout()
plt.show()
```

Output

The image shows 1-level decomposition of the original image and it is decomposed into four sub-bands: LL (approximation), LH (horizontal details), HL (vertical details), and HH (diagonal details).This level of decomposition captures the image's key structural and textural features for denoising.

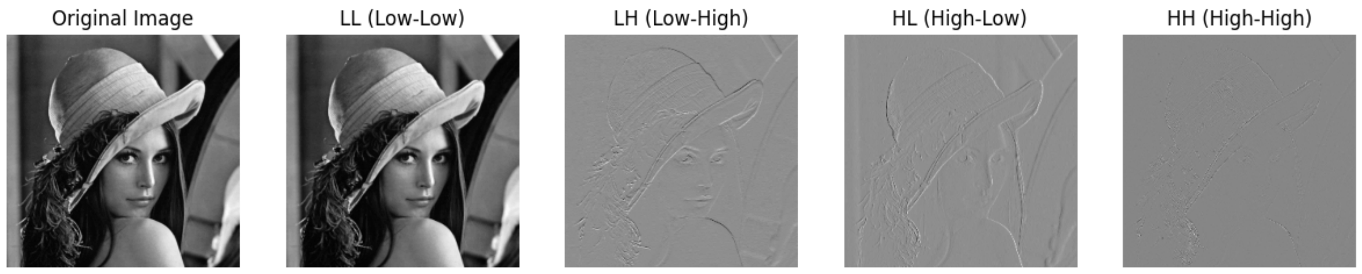

Fig. 3. Output of level-1 wavelet decomposition

The image undergoes a **two-level two-dimensional (2D) wavelet decomposition**. This process involves the following steps: The LL_1 sub-band is further decomposed using the same filtering process, resulting in:

- LL_2: Refined approximation coefficients
- LH_2: Horizontal detail coefficients at level 2
- HL_2: Vertical detail coefficients at level 2
- HH_2: Diagonal detail coefficients at level 2

  Final Output: After the two-level 2D wavelet decomposition, the image is represented by the following set of sub-bands:
  - LL_2
  - LH_2, HL_2, HH_2
  - LH_1, HL_1, HH_1

This hierarchical structure provides a multi-resolution representation of the image, which is highly effective for tasks such as image compression, denoising, and feature extraction.

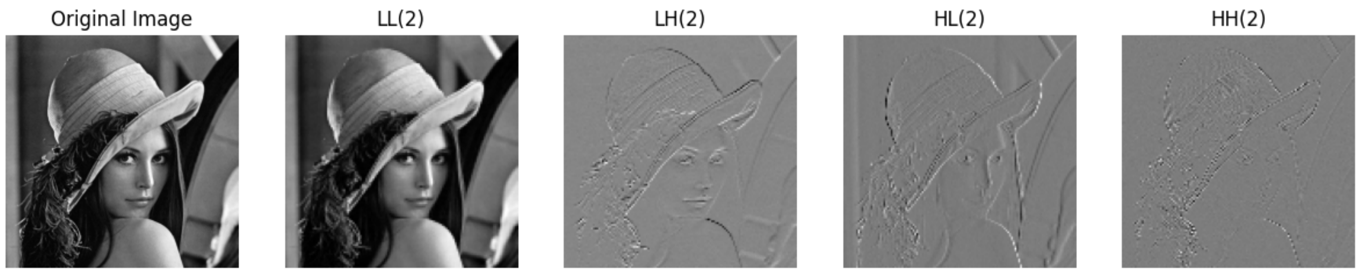

Fig. 4. Output of level-2 wavelet decomposition

The following output was obtained using the BayesShrink and VisuShrink method for coloured image and greyscale image respectively.

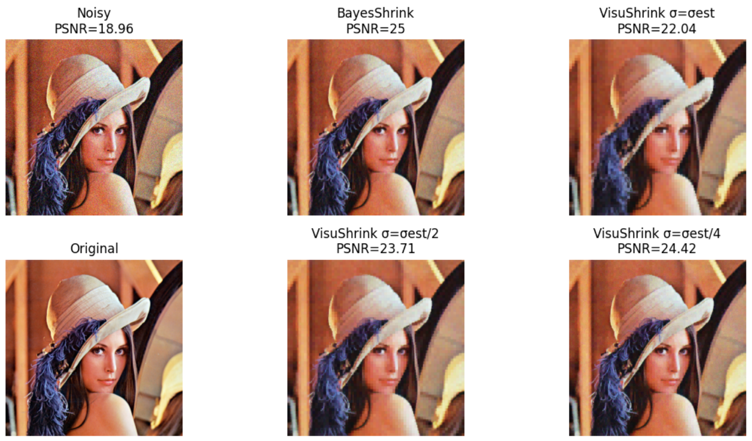

Fig. 5. Output of BayesShrink and VisuShrink methods for colour image

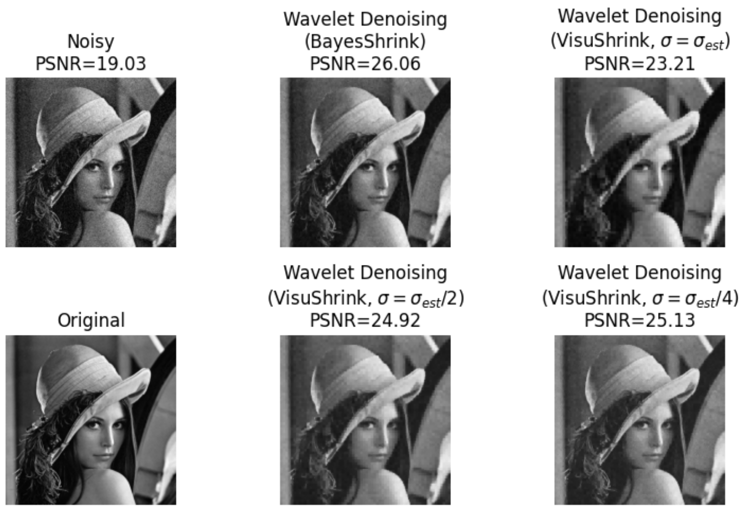

Fig. 6. Output of BayesShrink and VisuShrink methods for greyscale image

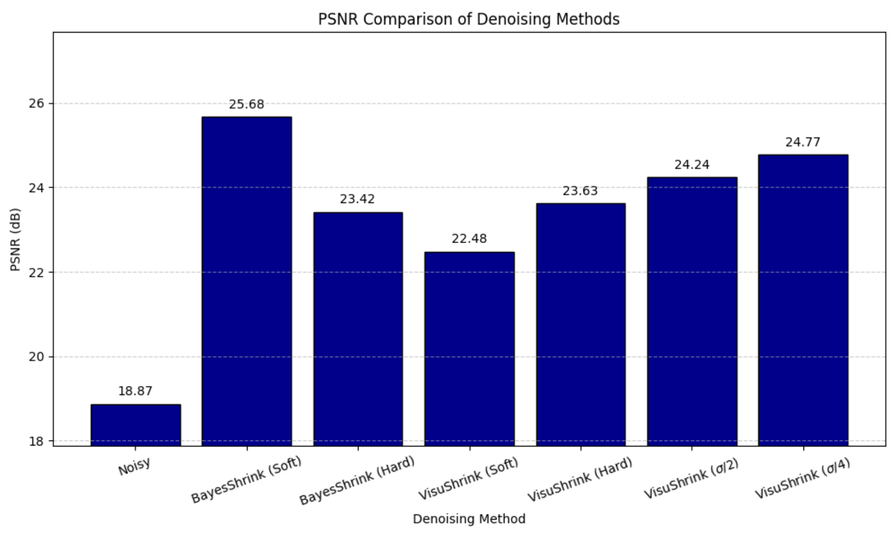

Fig. 7. SNR comparison of denoising methods

Inference

Fig. 3 shows the 1-level 2D Discrete Wavelet Transform (DWT) decomposition of the Lena image, resulting in four subbands: LL, LH, HL, and HH. The LL (Low-Low) subband contains low-frequency components in both directions and preserves most of the image's structure, appearing as a blurred version of the original. The LH (Low-High) and HL (High-Low) subbands highlight horizontal and vertical edges, respectively, making them useful for edge detection. The HH (High-High) subband captures diagonal details and fine textures, typically appearing darker due to lower energy content. This decomposition effectively separates image information by frequency, useful for denoising, compression, and feature extraction.

Fig. 4 illustrates the 2-level wavelet decomposition of the Lena image. The LL(2) subband, representing low-frequency components after two levels of decomposition, retains a smoother and more compressed version of the original image. The LH(2), HL(2), and HH(2) subbands capture finer horizontal, vertical, and diagonal details at this coarser scale. Compared to level 1, these detail subbands highlight broader edge structures with reduced resolution, demonstrating how higher-level decompositions extract coarser features while discarding finer textures.

The table compares several wavelet denoising methods based on key performance metrics, including Peak Signal-to-Noise Ratio (PSNR), noise reduction, detail preservation, and qualitative observations. **BayesShrink** emerges as the most optimal method among those tested, achieving the highest PSNR of 27.29. This suggests that BayesShrink offers the best balance between noise suppression and detail retention, making it highly effective for applications that require both high noise reduction and the preservation of fine details in the image.

In contrast, **VisuShrink**, when applied with the full noise standard deviation threshold (σ_{est}), uses universal hard thresholding, which results in significant noise reduction but also causes a loss of finer details due to over-smoothing. This leads to a relatively lower PSNR of 22.73. However, when the threshold is reduced to $\sigma_{\text{est}}/2$, VisuShrink improves in terms of detail preservation, yielding a moderate PSNR of 25.18. Further lowering the threshold to $\sigma_{\text{est}}/4$ results in an even better trade-off between noise removal and sharpness, with an improved PSNR of 25.59.

From these observations, it is clear that **BayesShrink** or **VisuShrink with a reduced threshold** ($\sigma_{\text{est}}/4$) are the most recommended methods for achieving an optimal balance between noise suppression and the preservation of image texture and details.

Conclusion

In conclusion, both BayesShrink and VisuShrink are effective tools for denoising images, with each offering distinct advantages depending on the nature of the noise and the image. By evaluating the performance using PSNR, we were able to quantify the improvements made by each technique, ultimately finding that wavelet-based methods can significantly improve image quality, particularly when noise is present. BayesShrink outperforms VisuShrink in terms of PSNR, making it a better choice for denoising in this scenario. However, tuning the threshold in VisuShrink improves performance, and using $\sigma_{est}/4$ provides a good balance between noise removal and detail preservation.

Method	Thresholding Approach	PSNR
Noisy Image (Gaussian Noise Added)	N/A (Original image with Gaussian noise, $\sigma = 0.12$)	18.53
BayesShrink	Adaptive Bayesian Thresholding	27.29
VisuShrink ($\sigma = \sigma_{\text{est}}$)	Universal Hard Thresholding	22.73
VisuShrink ($\sigma = \sigma_{\text{est}}/2$)	Reduced Thresholding	25.18
VisuShrink ($\sigma = \sigma_{\text{est}}/4$)	Lower Thresholding	25.59

Applications

Wavelet transform-based image denoising has a wide range of practical applications due to its effectiveness in separating noise from critical image details while preserving edges and textures. In medical imaging, it plays a crucial role in enhancing the quality of MRI, CT, and ultrasound images by reducing noise without affecting important diagnostic information. Satellite and aerial imaging also benefit significantly, as wavelet denoising helps mitigate noise caused by atmospheric conditions or sensor limitations, improving the clarity of images used in remote sensing, agriculture, and environmental studies. Additionally, it is widely employed in digital photography and video surveillance to enhance images captured in low-light or noisy environments. Wavelet denoising also improves the accuracy of optical character recognition (OCR) systems by enhancing the readability of scanned documents. In astronomical imaging, it aids in eliminating interference such as cosmic rays, thereby revealing finer details in space observations.

Future Scope

The future of wavelet-based image denoising holds immense potential for innovation through the integration of advanced and hybrid techniques. One promising direction involves combining traditional denoising methods such as BM3D (Block-Matching 3D Denoising) with wavelet transforms and deep learning models like convolutional neural networks (CNNs) and transformers. These hybrid approaches aim to achieve better adaptability to diverse noise patterns and image content. Additionally, optimizing the choice of wavelet basis functions—such as Haar, Daubechies, and Symlets—using machine learning or heuristic algorithms can lead to more efficient and noise-aware representations. Another critical research focus is on enhancing robustness by training and evaluating denoising algorithms using real-world noise rather than idealized Gaussian noise, with datasets like SIDD (Smartphone Image Denoising Dataset) and DND (Darmstadt Noise Dataset) serving as important benchmarks. Furthermore, real-time and hardware-accelerated implementations are gaining traction, especially for use in mobile devices, autonomous systems, and augmented reality applications. The role of wavelet denoising is also expanding in areas like compressive sensing, multi-modal sensor fusion, and generative models, where clean and high-fidelity image data is essential.

References

[1] *Gupta, Vikas, Rajesh Mahle, and Raviprakash S. Shriwas. "Image denoising using wavelet transform method." 2013 Tenth International Conference on Wireless and Optical Communications Networks (WOCN). IEEE, 2013.*

[2] *Mupparaju, Shivani, and B. Naga Venkata Satya Durga Jahnavi. "Comparison of various thresholding techniques of image denoising." International Journal of Engineering Research Technology (IJERT) 2.9 (2013): 3294-3301.*