IMA01 - Introduction to Image Processing
Comparison of two non-local image denoising methods

Lorenza Martins Guimaraes Tarallo, Mariana Dutra Diniz Costa

1 Introduction

Image denoising is a critical challenge in image processing and computer vision, aiming to reconstruct the original image by minimizing the impact of noise present in a degraded version. Noise in images can arise due to intrinsic factors, such as sensor limitations, or extrinsic factors, such as environmental conditions, which are often unavoidable in real-world scenarios. As a result, denoising techniques are essential in numerous applications, including image restoration, visual tracking, image registration, segmentation, and classification, where preserving the original image's content is vital for achieving high performance. Despite the development of various algorithms for image denoising, effectively reducing noise remains a complex task, particularly in cases where images are captured under adverse conditions with significant noise levels. This project focuses on implementing and analyzing two denoising methods: Block-Matching and 3D Filtering (BM3D) and Non-Local Means (NLM).

The BM3D (Block-Matching and 3D Filtering) is a method based on grouping similar two-dimensional patches into three-dimensional blocks. These blocks are processed using transformations, followed by coefficient thresholding techniques to remove noise while preserving the detailed features of the image. BM3D operates in two main steps: an initial filtering using hard thresholding and a more refined subsequent filtering using Wiener filtering, further improving the details and overall quality of the final result.

The NLM (Non-Local Means), on the other hand, is a method that uses the principle of replacing the value of a pixel with a weighted average of the values of similar pixels found in large regions of the image. The similarity between pixels is assessed by comparing windows around the pixels, allowing consistent patterns to be identified even in different parts of the image. This method uses exponential functions to calculate the weights of each pixel in the weighted average in order to prioritize more similar patches.

The aim of this project was to compare two image denoising methods, NLM (Non-Local Means) and BM3D (Block-Matching and 3D Filtering). Both methods were implemented and tested, and analyses were conducted based on the obtained results. The implementation covered the algorithms described in the reference papers, and the results were evaluated in terms of their efficiency in noise reduction and preservation of image details.

2 Description of Methods

2.1 BM3D

The BM3D (Block-Matching and 3D Filtering) method is an image denoising algorithm based on the principle of exploiting the sparsity of image representations in the transform domain. It involves grouping similar patches from a noisy image into 3D blocks, which are then collaboratively filtered through a series of transformations and thresholding steps. This process is divided into two main stages: the first denoising step applies hard thresholding with the transforms Walsh-Hadamard or Bior 1.5, while the second denoising step uses Wiener filtering to refine the noise reduction and enhance

detail preservation. In this case the Walsh-Hadamard and Discrete Cosine Transform (DCT) are used. Each stage concludes with an aggregation phase, where overlapping patches are averaged to produce the final estimate. These detailed steps will be further elaborated upon in subsequent sections to highlight the BM3D method's effectiveness and computational structure.

2.1.1 First Denoising Step

- **Grouping**
  In the first step of the BM3D algorithm, the grouping is a fundamental stage that aims to identify and cluster similar patches. This step operates by searching for patches within a defined neighborhood of a reference patch that exhibit high similarity based on a distance metric (Euclidean distance). The identified patches are then stacked together to form a three-dimensional block, which serves as the input for the subsequent collaborative filtering.

- **Collaborative Filtering**
  Collaborative Filtering in the first step of the BM3D method is a pivotal operation. In this stage, the 3D block formed during the grouping step undergoes a sequence of transformations, starting with a 2D transform applied to each patch, followed by a 1D transform along the third dimension of the block. After these, a hard thresholding is applied to attenuate noise by suppressing coefficients below a threshold. The filtered coefficients are then subjected to an inverse transform sequence to reconstruct the denoised 3D block. By collaboratively processing all patches in the block, this step enhances shared features while diminishing uncorrelated noise, laying the groundwork for accurate aggregation and image reconstruction.

  - **Bior 1.5 Transform**
    The Bior 1.5 transform is a biorthogonal wavelet transform that operates on the two-dimensional patches within the 3D block. It enables a detailed representation of the spatial and frequency characteristics of the image. In the project, the Bior 1.5 transform was used to process individual 2D patches before further collaborative filtering steps. Its role is crucial for enhancing the sparsity of the patch representation, which improves the effectiveness of the subsequent thresholding process by isolating noise from meaningful image structures.
  - **Walsh-Hadamard transform**
    The Walsh-Hadamard transform is a non-sinusoidal orthogonal transformation that decomposes data into a set of basis functions characterized by square waves. In the first step of BM3D, it is applied along the third dimension of the 3D block, facilitating the separation of noise and signal by concentrating significant features into fewer coefficients. This transform is integral to the collaborative filtering process, as it prepares the 3D block for effective thresholding, ensuring that shared patterns among similar patches are preserved while uncorrelated noise is suppressed.
  - **Hard Thresholding**
    Hard thresholding is a denoising technique that operates by suppressing transform coefficients below a predefined threshold. Within the BM3D framework, it is applied to the coefficients obtained after the Walsh-Hadamard and Bior 1.5 transforms. This operation plays a central role in collaborative filtering, by effectively removing noise components, hard thresholding ensures that only the most relevant components of the image contribute to the denoised output.
  - **Inverse Transform**
    The inverse transform is the final step in the collaborative filtering process, responsible for reconstructing the denoised 3D block from the filtered coefficients obtained after thresholding. In the BM3D algorithm, both the 1D Walsh-Hadamard transform and the 2D Bior 1.5 transform are reversed to bring the processed data back into the spatial domain. This ensures that the contributions of all filtered patches are accurately restored while retaining the structural integrity of the original image.

- **Aggregation**
  Aggregation is the final stage of the step, where the results from collaboratively filtered patches are merged to form a preliminary estimate of the denoised image. This process addresses the

overlapping nature of the grouped patches by averaging the multiple estimates available for each pixel, thereby reducing noise while preserving image details. Weighted averaging is employed, with the weights reflecting the reliability of each patch based on the sparsity of its coefficients after thresholding. In the project, aggregation was implemented by combining the inverse-transformed patches back into their respective positions in the image, ensuring a smooth reconstruction.

In conclusion, the first step of the BM3D algorithm, consisting of grouping, collaborative filtering, and aggregation, establishes the framework for initial noise reduction. By identifying similar patches, transforming them, applying hard thresholding, and reconstructing them through aggregation, this step efficiently separates noise from significant image features. The implementation in this project followed the theoretical guidelines provided in the reference article. The resulting intermediate image serves as a strong foundation for the second step, with significant noise suppression and detail preservation achieved. This ensures that the Wiener filtering in the second step can focus on refining the results further, enhancing overall denoising performance.

2.1.2 Second Denoising Step

- **Grouping**
  The grouping in the second step serves a similar purpose as in the first step: identifying and clustering similar patches into 3D blocks for collaborative filtering. The key difference lies in the input data and the refinement focus. Now, the grouping is performed on the denoised image produced by the first step. This refinement allows more accurate identification of similar patches.

- **Collaborative Filtering**
  The collaborative filtering in the second step differs from the first step in both its approach and purpose. While the first step focuses on initial noise suppression through hard thresholding, the second step employs Wiener filtering, which leverages the denoised image from the first step as a reference to refine the filtering process. This step enhances the precision of noise removal by weighting the coefficients based on the reliability of the patches, as estimated from the first-step output. The goal is to further reduce residual noise while preserving even finer details and textures, leading to an image with higher fidelity and visual quality.

  - **Wiener Filtering**
    Wiener filtering is the core operation of the second step collaborative filtering process. It estimates the noise-free coefficients of the image patches using a reference provided by the output of the first step. The Wiener filter applies a weighting scheme that minimizes the mean squared error between the estimated and original coefficients, effectively suppressing noise while retaining important image details.
  - **Discrete Cosine Transform (DCT)**
    The DCT is a frequency-domain transformation that represents data in terms of cosine functions, which are efficient for capturing energy compaction in natural images. The DCT is applied to each 2D patch in the 3D block, enabling the separation of signal and noise components in the frequency domain. This choice ensures effective noise suppression while preserving the visual quality of the image, especially for smooth regions and fine details.
  - **Walsh-Hadamard Transform**
    The Walsh-Hadamard Transform is a non-sinusoidal orthogonal transformation that operates on binary waveforms. In the second step of BM3D, the Walsh-Hadamard Transform is applied along the third dimension of the 3D block, similar to its role in the first step. However, its use in conjunction with Wiener filtering ensures that redundancies across similar patches are efficiently used to suppress noise and refine structural details.

- **Aggregation**
  The aggregation in the second step serves the same purpose as in the first one: merging the contributions of overlapping patches to reconstruct the final denoised image. However, the key difference lies in the level of refinement and accuracy. In the second step, aggregation incorporates the results of Wiener filtering, which provides more reliable patch estimates by leveraging the reference image from the first step. The weights used during aggregation are adjusted to reflect

the reliability of each patch's coefficients, ensuring that the most accurate patches contribute more significantly to the final reconstruction.

The second step of the BM3D algorithm culminates in the reconstruction of the final denoised image by combining the refined outputs of collaborative filtering and aggregation. It begins with grouping patches from the first-step output, followed by collaborative filtering using Wiener filtering, DCT, and Walsh-Hadamard transforms to refine the 3D blocks. The refined patches are then aggregated into the image domain, leveraging their overlapping nature to minimize residual noise and enhance structural consistency. Through this approach, it is produced a final image that balances noise suppression with the preservation of image details, delivering an output that resembles the original image.

2.2 Non-Local Means

The Non-Local Means (NLM) method is the other image denoising algorithm within the scope of this project. It leverages the similarity between image patches across the entire image. Unlike traditional local filtering techniques, NLM replaces the intensity value of a pixel with a weighted average of intensities from pixels in a larger search window, where the weights are determined by the similarity between the patches surrounding the pixels.

The NLM algorithm combines patchwise similarity measurements, weighted averaging, and iterative processing to produce a denoised image that balances noise reduction with detail preservation. To compute weights, patches are extracted from the image using sliding window views. The squared differences between a reference patch and other patches in the search window are aggregated to compute a normalized sum of squared distances. This similarity measure is then transformed into weights using an exponential decay function. This weighting ensures that patches similar to the reference patch contribute more to the denoised pixel value.

2.2.1 Patchwise Implementation

The patchwise approach allows the identification of similar structures across the entire image. It begins by dividing the image into patches and comparing each patch to its neighbors within a defined search window. For each pixel, a weighted average of these patches is calculated. This approach allows the algorithm to effectively reduce noise by incorporating information from similar patches found throughout the search window, rather than relying solely on nearby pixels.

2.2.2 Weighting

The weighting in the NLM algorithm is a pivotal component that ensures the denoising process effectively distinguishes between similar and dissimilar patches. Weights are computed based on the similarity between a reference patch and its neighboring patches within a search window. This similarity is measured using the normalized squared Euclidean distance between corresponding pixel intensities of the patches. To transform these distances into weights, an exponential kernel is applied, modulated by a filtering parameter h, which controls the degree of smoothing. The kernel ensures that patches with smaller distances (greater similarity) contribute more heavily to the weighted average, while those with larger distances (less similarity) are exponentially down-weighted. Additionally, the scheme includes a normalization step to ensure the weights sum to unity. This approach ensures that the final denoised pixel values are influenced predominantly by the most relevant and similar patches, effectively reducing noise while preserving important image structures and textures.

2.2.3 Aggregation

The aggregation step in the NLM algorithm consolidates the overlapping contributions of all processed patches to reconstruct the final denoised image. Each pixel value is computed as a weighted sum of intensities from similar patches identified within a search window. As patches are extracted with overlap, multiple estimates are generated for each pixel, contributing redundant information. During aggregation, these overlapping estimates are averaged, ensuring a smooth transition between regions and mitigating inconsistencies caused by noise. The weights used for averaging, derived from the similarity calculations, ensure that the most relevant patches have a greater influence on the final pixel

values. This step effectively combines local and non-local information to achieve a balance between noise suppression and structural detail preservation, resulting in the reconstructed denoised image.

3 Implementation

3.1 BM3D

To initiate the process, the original image undergoes normalization, ensuring its pixel intensity values are in the range [0,255]. Subsequently, Gaussian noise is added to the normalized image.

3.1.1 Step 1

- **Grouping**

The implementation of the first step starts with the grouping. The grouping process begins by extracting the reference patch from the input image based on the provided top-left corner coordinates. The patch is reshaped into a vectorized form to facilitate comparison with other patches. To identify candidate patches for grouping, a search window centered around the reference patch is generated. Within this search window, patches of the same size as the reference patch are extracted and vectorized using a sliding window approach.
If the noise standard deviation σ exceeds 40, hard thresholding is applied to the patches. This process suppresses coefficients below a certain threshold, reducing the influence of noise in the comparison.
The normalized quadratic distance between patches is calculated by the equation 1:

$$d(P,Q) = \frac{\|\gamma'(P) - \gamma'(Q)\|_2^2}{(k_{\text{hard}})^2} \tag{1}$$

where $\|\gamma'(P)$ and $\gamma'(Q)$ represent vectorized forms of the reference patch P and a candidate patch Q from the search window. This distance metric ensures the similarity between patches is accurately evaluated, taking into account the normalized variance of pixel intensities.
To identify similar patches, the algorithm sorts the distances in ascending order and selects the top N patches, where N is restricted to be a power of two for computational efficiency during the subsequent collaborative filtering step. A similarity threshold, τ^{hard}, is applied to further refine the selection, excluding patches with a distance greater than the threshold. If the number of selected patches does not conform to the power-of-two requirement, it is reduced to the nearest lower power of two.
Once the grouping is complete, the algorithm retrieves the top-left coordinates of the selected patches within the search window and organizes the corresponding vectorized patches into a 3D group by reshaping them back into their original dimensions. This 3D group, along with the coordinates of the patches, is returned as the output of the grouping step, serving as input for the collaborative filtering process.

- **Collaborative Filtering**

The process of collaborative filtering begins with the application of a 3D transform to the grouped patches, which involves a combination of 2D and 1D transforms. Each patch in the 3D group is transformed using a 2D Biorthogonal Wavelet Transform applied along the spatial dimensions, followed by a 1D Walsh-Hadamard Transform applied along the grouping dimension. This operation enhances sparsity by projecting the patches onto a transform domain where noise and image content are better separated

- **2D Biorthogonal Wavelet Transform**

The 2D Biorthogonal Wavelet Transform is applied along the spatial dimensions of each patch within the group. It decomposes each patch into low-frequency and high-frequency components using a set of basis functions designed for sparse representation. It was implemented computing 8x8 matrices that, when applied to a vector result in the 1d Bior1.5 transform. The low-frequency coefficients capture the overall structure of the patch, while the high-frequency coefficients correspond to finer details and noise.

- **1D Walsh-Hadamard Transform**

The 1D Walsh-Hadamard Transform is applied along the third dimension of the group. It operates on data arrays of size 2^n, where n is a non-negative integer, and decomposes the data into a set of orthogonal Walsh basis functions. The function employs a divide-and-conquer approach, splitting the input array into even-indexed elements *x[0::2]* and odd-indexed elements *x[1::2]*. The transform is recursively applied to the even and odd parts. After recursion, the results from the even and odd parts are combined using addition and subtraction, where the first half of the transformed array is the sum of the even and odd parts and the second half is the difference, as showed in equation 2. The output is the transformed array, which has the same size as the input.
Let x be the input array of length $N = 2^n$. The WHT is computed as:

$$\mathrm{WHT}(x) = \begin{cases} x, & \text{if } N = 1 \\ \begin{bmatrix} \mathrm{WHT}(\mathrm{even}(x)) + \mathrm{WHT}(\mathrm{odd}(x)) \\ \mathrm{WHT}(\mathrm{even}(x)) - \mathrm{WHT}(\mathrm{odd}(x)) \end{bmatrix}, & \text{if } N > 1 \end{cases} \tag{2}$$

Where:

- $\mathrm{even}(x)$ = elements at even indices of x,
- $\mathrm{odd}(x)$ = elements at odd indices of x.

- **Inverse Transforms**

The goal of the inverse transform is to project the data back from the transform domain to the spatial domain while preserving the denoised structural information. The Inverse Bior 1.5 is implemented using two steps of matrix multiplication along two dimensions of the transformed data. The inverse Walsh-Hadamard Transform is implemented using the same recursive structure as the forward transform, but since the Walsh-Hadamard matrix is orthogonal, the inverse transform is identical to the forward transform except for a normalization factor. These inverse operations are essential to recover image blocks or patches for further aggregation and reconstruction of the final denoised image.

- **Aggregation**

The aggregation process involves combining the filtered patches into a preliminary estimate of the denoised image. The function implemented manages two buffers, the numerator (ν) and the denominator (δ), which are used to combine patches. The filtered patches are multiplied by their respective weights and added to the numerator buffer at their corresponding positions in the original image. Similarly, the weights are aggregated in the denominator buffer, as it follows in equation 3.

$$\forall Q \in \mathcal{P}(P), \forall x \in Q, \begin{cases} \nu(x) = \nu(x) + w_P^{\mathrm{hard}} u_{Q,P}^{\mathrm{hard}}(x), \\ \delta(x) = \delta(x) + w_P^{\mathrm{hard}} \end{cases} \tag{3}$$

Where ν (resp. δ) designates the numerator (resp. denominator) part of the basic estimate of the image obtained at the end of the grouping step; $u_{Q,P}^{\mathrm{hard}}(x)$ is the estimate of the value of the pixel x belonging to the patch Q obtained during collaborative filtering of the reference patch P.
The weight assigned to the patch is defined by the formula 4.

$$w_P^{\mathrm{hard}} = \begin{cases} \left(N_P^{\mathrm{hard}}\right)^{-1} & \text{if } N_P^{\mathrm{hard}} \geq 1, \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

where N_P^{hard} is the number of retained (non-zero) coefficients in the 3D block after hard-thresholding: $\gamma\left(\tau_{3D}^{\mathrm{hard}}(\mathcal{P}(P))\right)$.

The purpose of this weighting scheme is to prioritize homogeneous patches, which are characterized by a higher number of canceled coefficients. Patches containing edges are given less significance compared to uniform patches, especially those located along the boundaries of edges. This approach ensures that smooth, homogeneous areas contribute more during aggregation

3.1.2 Second Step Denoising

- **Grouping**

The grouping procedure in the second step identifies and clusters similar patches from the basic estimate (output of Step 1) rather than the noisy image. Patches are extracted from the basic estimate using the same distance metric as in the first step, ensuring high similarity among grouped patches. The coordinates of the grouped patches are used to extract corresponding patches from the original noisy image. These two groups—basic and original—serve as inputs for the collaborative filtering process.

- **Collaborative Filtering**

The collaborative filtering in the second step employs Wiener filtering to enhance precision in noise suppression. The filtering process involves the 3D transformation (where each patch is transformed using a 2D Discrete Cosine Transform along the spatial dimensions and a 1D Walsh-Hadamard Transform along the grouping dimension), Wiener filtering and the inverse transforms.

- **2D Discrete Cosine Transform**

The 2D Discrete Cosine Transform (DCT) was implemented using the 'dct' function from the *SciPy* library. The transform is applied along the rows and then along the columns of each patch, ensuring orthonormal normalization. This implementation preserves the energy of the data while converting it into the frequency domain, where most of the important information is stored in the low-frequency components.

- **1D Walsh-Hadamard Transform**

The 1D Walsh-Hadamard Transform in the second step is implemented using the same recursive structure as in the first step, where the input array is split into even and odd components and combined using addition and subtraction. However, in the second step, it is applied to both the basic estimate and the original patches, unlike the first step, which only processes grouped patches from the noisy image.

- **Wiener Filtering**

The Wiener filter was implemented by calculating Wiener coefficients based on the transformed coefficients of the basic estimate. For each coefficient, the Wiener weight is determined as follows in equation 5:

$$\omega_P(\xi) = \frac{\left|\tau_{3D}^{\text{wien}}\left(\mathcal{P}_{\text{basic}}(P)\right)(\xi)\right|^2}{\left|\tau_{3D}^{\text{wien}}\left(\mathcal{P}_{\text{basic}}(P)\right)(\xi)\right|^2 + \sigma^2}. \tag{5}$$

These weights balance the influence of noise (σ^2) and the signal strength in the basic estimate. The Wiener filter then multiplies these coefficients with the transformed coefficients of the original patches to suppress noise while retaining important signal features.

- **Inverse Transforms**

The inverse transforms are applied to reconstruct the filtered patches back into the spatial domain after collaborative filtering. The inverse WHT is identical to the forward WHT, as the Walsh-Hadamard matrix is orthogonal. The implementation recursively combines the even and odd components of the transformed data using addition and subtraction. The inverse 2D DCT is implemented using a function from *SciPy*. The transform is applied first along the rows and then along the columns of each patch. These reconstructed patches are then aggregated to form the final denoised image.

- **Aggregation**

In the second step of BM3D, aggregation combines the filtered patches into a final denoised image using two buffers, the numerator (ν) and denominator (δ), as shown in equation 6, to accumulate the contributions of all patches. They are updated iteratively during the processing of each patch.

$$\forall Q \in \mathcal{P}(P), \forall x \in Q, \begin{cases} \nu(x) = \nu(x) + w_P^{\text{wien}} u_{Q,P}^{\text{wien}}(x), \\ \delta(x) = \delta(x) + w_P^{\text{wien}} \end{cases} \tag{6}$$

Where ν (resp. δ) designates the numerator (resp. denominator) part of the final estimation of the image obtained at the end of the previous step, $u^{\text{wien}}_{Q,P}(x)$ is the estimation of the value of the pixel x belonging to the patch Q obtained during the collaborative filtering of the reference patch P. The weight w^{wien}_P is calculated as the inverse of the squared ℓ_2-norm of the Wiener coefficients ω_P, as follows in equation 9:

$$w^{\text{wien}}_P = \|\omega_P\|_2^{-2} \tag{7}$$

The aggregation process in the second step of the BM3D method builds on the principles established in the first step but refines the final estimation by leveraging Wiener-filtered patches. It combines these filtered patches into a coherent denoised image using two iterative buffers. The numerator accumulates the weighted contributions of the patches, while the denominator aggregates the weights, ensuring balanced reconstruction. The weights depend on the Wiener coefficients, which assess the reliability of the patches. This weighting strategy prioritizes accurate patches while minimizing the influence of noisy ones. The final image is reconstructed by dividing the accumulated numerator by the denominator for each pixel. This process ensures that the denoised image achieves good noise suppression while retaining intricate details and textures, culminating in a faithful representation of the original image.

3.2 Non-Local Means

The Non-Local Means (NLM) algorithm implemented in this work adheres closely to the principles described in the article. The implementation incorporates key elements such as the parameter selection and the patch-based weighting.

- **Parameter Definition**

The parameters for the algorithm are defined based on the standard deviation of noise (σ). These include the patch size, search window size, and filtering parameter h, all of which are adjusted dynamically according to predefined ranges for σ. For example, for small noise levels ($0 < \sigma \leq 15$), a 3×3 patch and a 21×21 search window are used, with $h = 0.4\sigma$. As σ increases, larger patches and windows are employed to ensure sufficient robustness in patch comparisons.

- **Denoising**

The first step of the implementation regards the patch extraction and the search window. For each pixel in the image, a reference patch centered on that pixel is extracted and all patches within a predefined search window around the reference patch are extracted. These patches are vectorized to facilitate efficient computations.
The similarity between the reference patch and other patches is calculated using the normalized squared Euclidean distance formula stated in equation 8.

$$d^2(p,q) = \frac{1}{(2f+1)^2} \sum_{j \in B(0,f)} \left(u(p+j) - u(q+j)\right)^2 \tag{8}$$

In this equation, p and q represent the central pixels of the patches being compared. u is the intensity value of the pixel and f is the half size of the patch. Smaller values of d^2 indicate higher similarity, these distances are then utilized in the subsequent weighting step to determine the contribution of each patch during the aggregation phase.
The weights for each patch are computed using an exponential kernel described in equation 9:

$$w(p,q) = \exp\left(-\frac{\max(d^2 - 2\sigma^2, 0.0)}{h^2}\right) \tag{9}$$

Where *w(p,q)* is the weight assigned to the pixel q based on its similarity to the reference pixel p, d^2 is the normalized squared Euclidean distance calculated before, σ^2 is the variance of the noise in the image, h is the filtering parameter that controls the decay of the weights.
$\max(d^2 - 2\sigma^2, 0.0)$ ensures that negative distances are truncated to 0, which prevents over-penalizing patches.And the exponential kernel ensures that patches more similar to the reference patch contribute more significantly, with the weight decaying exponentially for less similar patches.

- **Weighted aggregation**

In the weighted Aggregation step, the algorithm computes the final pixel intensities by averaging patches weighted by their similarity to a reference patch. First, the weights were determined using an exponential kernel. These weights are then normalized by dividing the weighted sum of all patches by the total weight. The aggregated values are reshaped back into the patch size and added to the denoised image buffer. After all patches are processed, the aggregated values are normalized by the patch area, and the resulting image is cropped to its original size. This step ensures that information from multiple similar patches contributes to the denoised image, effectively reducing noise while preserving image details.

4 Results and Discussion

In order to visualize and evaluate the performance of each method, a series of tests were carried out for different grayscale images and varying standard deviation (σ) values, specifically 15, 30, 40, 50 and 60. While in this section we present and discuss the outcomes of a selection of the conducted experiments to illustrate key findings, the complete set of tests and corresponding data is available in the GitHub repository.

The Root Mean Square Error (RMSE) and the Peak Signal-to-Noise Ratio (PSNR) will be employed as primary metrics. The RMSE, defined mathematically by equation 10, measures the average deviation between the denoised image u_d and the reference noiseless image u_r.

$$\text{RMSE} = \sqrt{\frac{\sum_{x \in X} \left(u_R(x) - u_D(x)\right)^2}{|X|}} \tag{10}$$

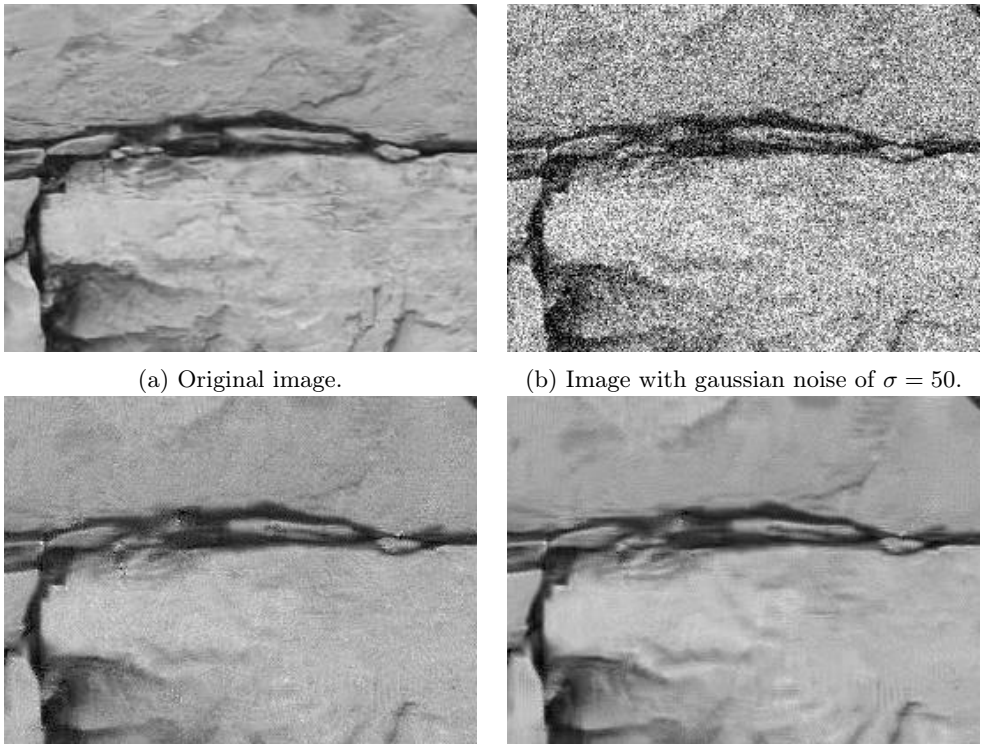
where X represents the set of pixel positions. A smaller RMSE indicates better denoising performance. The PSNR, expressed in decibels and defined by equation 11, evaluates the ratio between the maximum possible pixel intensity 255 and the RMSE. A higher PSNR reflects superior noise suppression while preserving image fidelity. These metrics will be computed for both algorithms across various images and noise levels to allow a detailed comparison.

$$\text{PSNR} = 20 \cdot \log_{10}\left(\frac{255}{\text{RMSE}}\right) \tag{11}$$

In addition to these quantitative metrics, qualitative visual inspections will be conducted to assess texture preservation, artifact reduction, and structural fidelity. This systematic evaluation aims to identify trade-offs between noise reduction, detail preservation, and computational efficiency, providing a comprehensive understanding of the algorithms' performance.

4.1 BM3D

The experiments' results demonstrate that BM3D is a highly effective denoising method, with most images showing satisfactory results where the algorithm performed well in preserving fine details and structures while effectively reducing noise. Figure 1 illustrates the important role that BM3D's second step plays in enhancing the denoising results. The collaborative Wiener filtering leads to improvements that can be stated both visually and through the increase in PSNR from 23.93 to 25.54.



(a) Original image.

(b) Image with gaussian noise of $\sigma = 50$.

(c) Basic estimate after BM3D's first step. (PSNR = 23.93)

(d) Final estimate after BM3D's second step. (PSNR = 25.54)

Figure 1: Results of the BM3D method for comparison between first and second steps.

The visual analysis of results, shown in Figure 2, reveals that BM3D maintains excellent performance for moderate noise levels (up until $\sigma = 30$). At these levels, the method effectively suppresses noise while preserving edge details and textures, ensuring good quality reconstructions. However, for higher noise levels, such as $\sigma = 40$, 50 and 60, while the algorithm still produces satisfactory results in terms of removing overall Gaussian noise artifacts and texture loss in the image come up and become way more noticeable. Despite this, BM3D still outperforms the original noisy images, achieving improved PSNR values and improving the visual appearance.

(a) Original

(b) $\sigma = 15$
PSNR = 32.82

(c) $\sigma = 30$
PSNR = 29.87

(d) $\sigma = 40$
PSNR = 26.43

(e) $\sigma = 50$
PSNR = 27.31

(f) $\sigma = 60$
PSNR = 23.75

Figure 2: BM3D denoising results for noises of different standard deviation.

4.2 Non-Local Means

The results of the held experiments demonstrate that NLM effectively reduces noise in general, as seen in the satisfactory recovery of structural features of the image, especially when it comes to homogeneous areas. In most of the tests, the algorithm was successful in effectively removing much of the noise and preserving the overall structure of objects. However, some fine details can be lost or smoothed out.

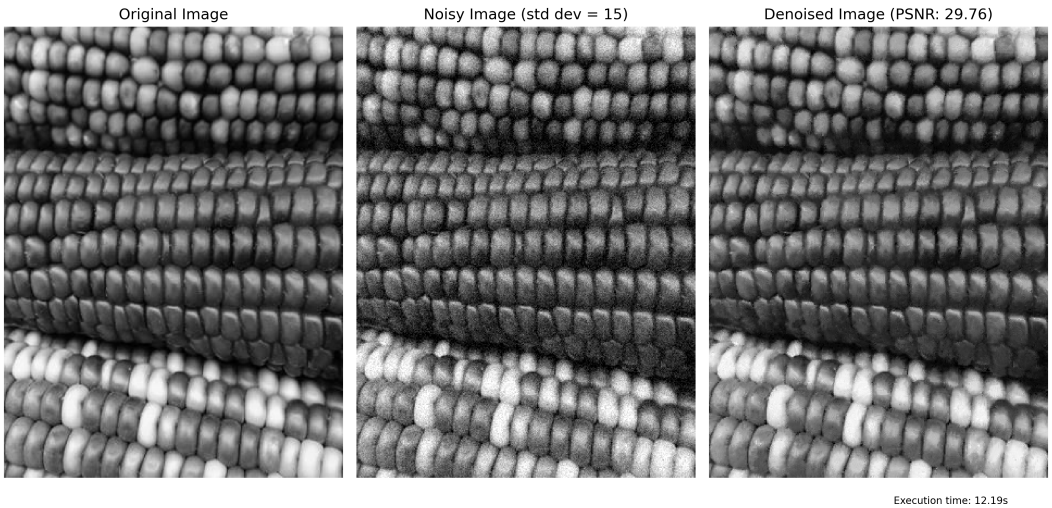


Figure 3: Results of NLM denoising for an image of $\sigma = 15$.

An important observation regarding the NLM method is that it can easily lose track of fine details as the noise level increases. This occurs due to the fact that the algorithm relies simply on a weighted average of similar patches, which, while effective for noise reduction, can lead to excessive smoothing of textures and edges. In Figure 4 we see that as the noise standard deviation increases, the NLM method, although still capable of reducing noise, progressively smooths out contours and details in areas such as the hat, skin, and background. Besides the visual perception, this is also reflected in the corresponding decrease in PSNR values.

(a) Original

(b) $\sigma = 15$
PSNR = 31.27

(c) $\sigma = 30$
PSNR = 28.65

(d) $\sigma = 40$
PSNR = 27.59

(e) $\sigma = 50$
PSNR = 26.67

(f) $\sigma = 60$
PSNR = 25.94

Figure 4: Non-local means results for different standard deviation of noise.

4.3 Comparison of both methods

Now, we compare the results of both methods side by side. One key aspect to consider is the computational complexity of each approach. NLM stands out for its simplicity and ease of implementation, making it a more straightforward method overall. While NLM tends to introduce a trade-off between noise reduction and smoothing—particularly noticeable at higher noise levels—this compromise may be acceptable in scenarios where a simpler, less computationally demanding approach is needed for the task.

After conducting comparison tests, BM3D's two-step process, incorporating Wiener filtering, outperformed NLM in terms of better noise reduction and maintenance of sharp edges. As illustrated in Figure 5, thin lines and intricate details are often lost in the NLM method due to its averaging process, whereas BM3D retains better the outlines of these features, ensuring greater structural fidelity in the denoised images.

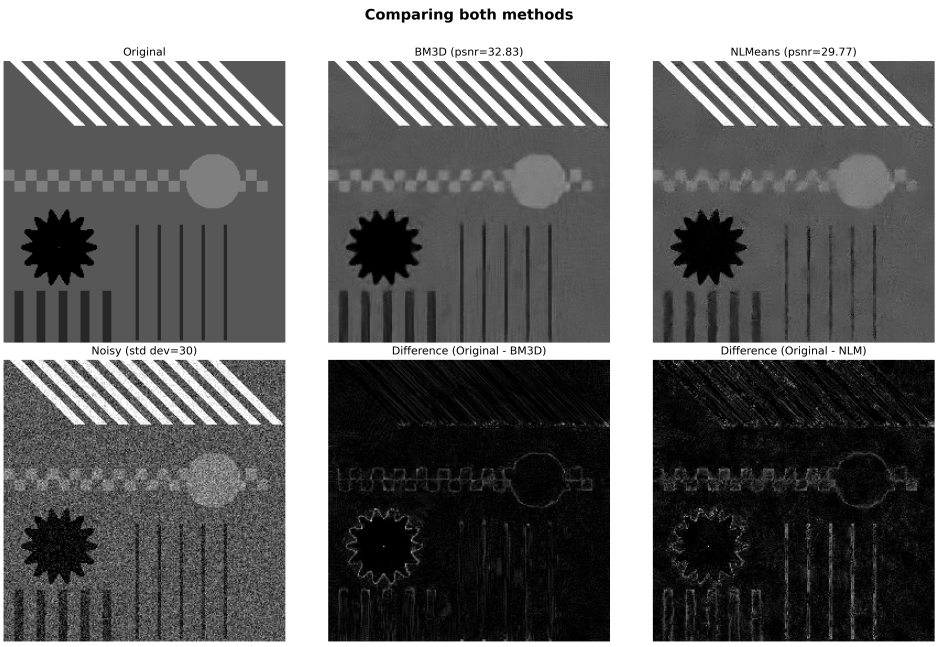


Figure 5: Comparing both methods for an input of $\sigma = 30$

For high noise levels, such as a standard deviation of 50, both BM3D and NLM exhibit limitations, but BM3D consistently outperforms NLM in preserving structural details. The BM3D result retains sharper edges and better reconstructs the fine details of the original image, as evidenced by the smaller differences in the "Original - BM3D" comparison. In contrast, NLM introduces noticeable smoothing effects, as its reliance on weighted averaging causes it to emphasize broader textures while losing finer details. This smoothing effect is particularly evident in the "Original - NLM" difference image (Figure 6), where only edge information is preserved, highlighting the algorithm's difficulty in maintaining texture and sharpness under high noise conditions. Despite this, NLM still achieves a reasonable level of denoising, reducing the overall noise intensity in the image. Additionally, the PSNR values confirm BM3D's superiority, indicating its better performance in balancing noise reduction and detail preservation.

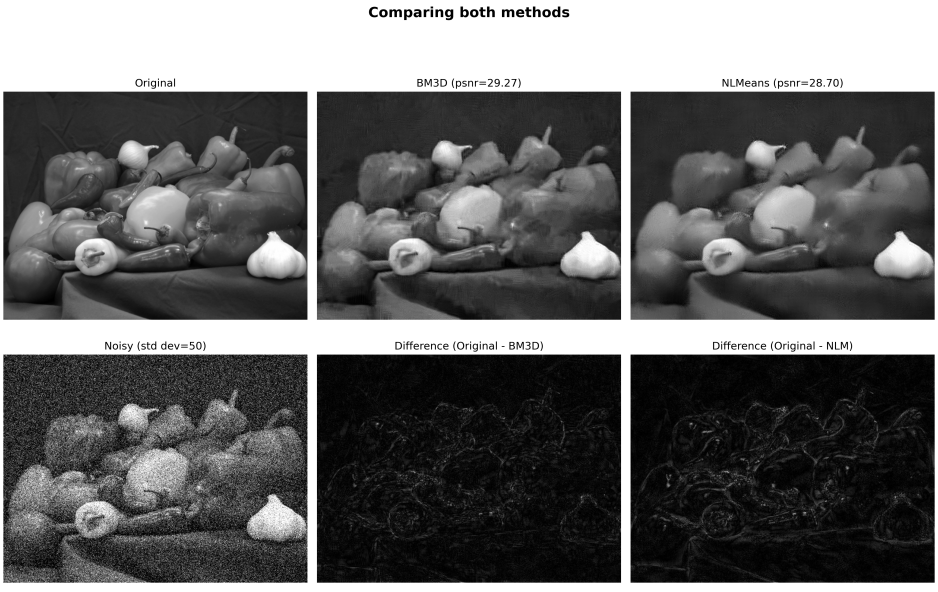


Figure 6: Comparing both methods for an input of $\sigma = 50$.

Figure 7 provides another comparison between the two methods, this time with a noise standard deviation of $\sigma = 30$. In this case, both methods face challenges in reconstructing fine details on the building's walls, although they remain relatively successful given the moderate noise level. The plants, in particular, are heavily smoothed, especially by the NLM method, which struggles to preserve their texture. However, the boats are well-reconstructed by both methods, delivering visually satisfactory results. Once again, the PSNR metric highlights the superior performance of BM3D, confirming its effectiveness in balancing trade-offs between noise reduction and detail preservation.

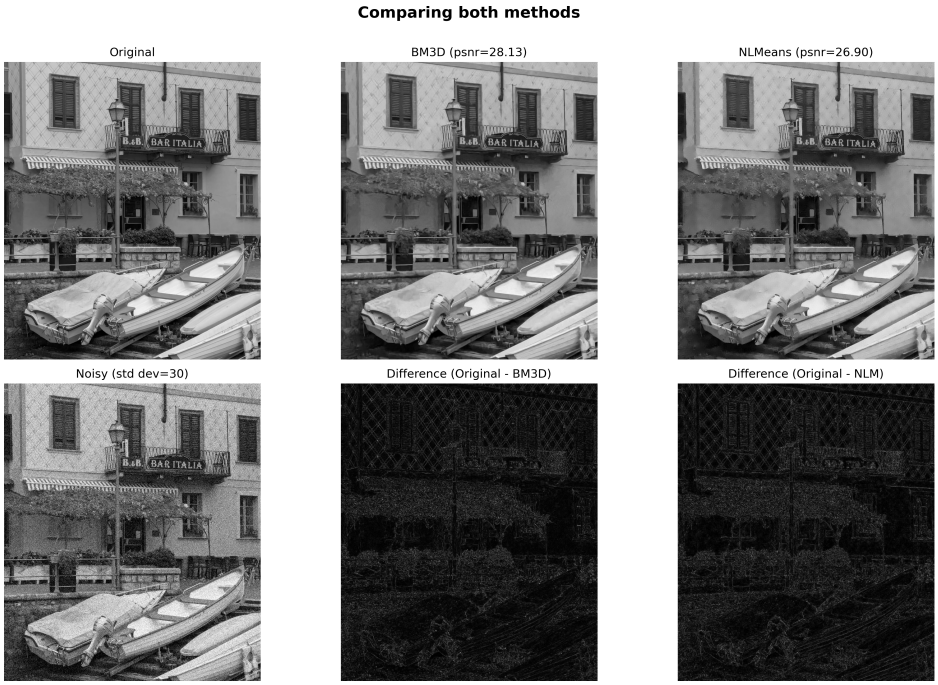


Figure 7: Comparing both methods for an input of $\sigma = 30$.

5 Conclusion

In general, the BM3D algorithm demonstrates superior performance compared to NLM in terms of denoising efficiency and detail preservation. However, both methods exhibit limitations regarding the amount of noise they can effectively remove and the extent to which image details are maintained or reconstructed.

BM3D excels in preserving edges and structural details while providing a more uniform noise reduction across the image. On the other hand, NLM relies on a weighted averaging approach that tends to preserve textures but may introduce smoothing artifacts, particularly in high-noise scenarios.

Regarding quantitative results, BM3D generally achieves higher PSNR values, indicating better overall noise suppression and image fidelity. Interestingly, as noise levels increase, NLM occasionally surpasses BM3D in terms of PSNR. Despite this, NLM struggles to maintain clear edges and shapes under such conditions, resulting in visually less satisfactory outputs. These findings highlight the trade-offs between the two methods.

In conclusion, the comparison between these two methods highlights the strengths and limitations of both of them, with BM3D having a superior denoising performance overall and NLM offering a simpler alternative. The choice between them depends on the specific requirements of each application, where it will be needed to consider balancing noise reduction, edges and texture fidelity, and also computational complexity.