
Amirkabir University of Technology

(Tehran Polytechnic)

# Image Denoising using the Higher Order Singular Value Decomposition

Zohreh Sarmeli Saeedi
Supervisor: Dr. Shakeri
Faculty of Mathematics and Computer Science

June 2025

## Contents

**Abstract**

In this project, a simple, elegant, and effective method for image denoising is introduced, based on machine learning and higher-order singular value decomposition (HOSVD). This method operates on a patch-based approach to the image. The main steps of the method are as follows: First, for each point in the noisy image, a set of similar patches to that point is selected. The similarity between patches is measured using a statistical criterion derived from the noise model. These similar patches are organized into a three-dimensional stack (for grayscale images) or four-dimensional (for color images). Then, the HOSVD coefficients of this stack are calculated. These coefficients are filtered using hard thresholding and reconstructed by applying the inverse HOSVD transform. To obtain the final image, hypotheses averaging at the pixel level is used. One of the important features of this method is that all its required parameters are selected principledly based on the noise model. Additionally, the algorithm utilizes non-local similarities between image patches, which improves its efficiency. The experimental results of this method show very good performance in both grayscale and color images. Especially at relatively high noise levels, the proposed method has been able to provide better performance compared to other color denoising algorithms. Furthermore, in this research, a criterion for optimal selection of patch size and estimation of noise variance from the residual image is also presented. These criteria are designed based on the statistical properties of noise in reconstructed images.

# 1 Introduction

Image denoising is one of the classic and fundamental problems in image processing that has received widespread attention from researchers since the 1970s. Over these years, a wide range of techniques have been proposed to solve this problem. These methods include partial differential equation (PDE)-based methods, variable spatial smoothing, non-local methods, transform-domain methods, and machine learning methods. In recent years, transform-based techniques combined with machine learning have gained significant popularity due to their strong performance and ability to extract structural features of the image. One of the novel and effective approaches in this field is learning local and location-variable transform bases from the noisy image data itself; this approach, by utilizing non-local similarities between image patches, is capable of more accurate reconstruction of the original image structures.

In this project, a very simple yet effective algorithm for denoising is introduced, based on Higher Order Singular Value Decomposition (HOSVD). This algorithm operates locally but uses non-local information within the image. The main idea is that for each point in the image, a set of similar patches to that point across the image is selected and organized into a three-dimensional stack (for grayscale images) or four-dimensional (for color images). Then, using HOSVD, the transform coefficients are obtained, and noise is removed from them by applying hard thresholding. Finally, by performing the inverse transform and combining hypotheses from different points in the image, a smooth and reconstructed version of the original image is obtained. One of the important advantages of this algorithm is its conceptual and implementation simplicity. Unlike many existing complex methods that require setting multiple parameters and iterative optimizations, the proposed algorithm selects all its parameters in a principled manner based on the statistical noise model.

Additionally, this project shows that this method achieves noteworthy and competitive results compared to many more advanced algorithms such as BM3D and K-SVD. In the following, first a review of previous methods is conducted, then the main idea is introduced, followed by the precise implementation of the algorithm, analysis of numerical results, and finally a summary.

In recent years, very diverse methods for image denoising have been presented, which can be placed into several general categories: PDE-based methods, local convolution and regression methods with variable coefficients, non-local methods, transform-based methods, and machine learning methods. In the following, a brief overview of these categories is given, and the position of the proposed method relative to them is clarified.

PDE-based methods diffuse the image anisotropically, such that edge structures are preserved and diffusion occurs only along them. Some of these methods are derived from Euler-Lagrange equations whose objective function is designed based on piecewise constant or piecewise linear models of natural images. In practice, these energy functions are enhanced with penalty terms that control the deviation of the noisy image from the filtered image.

Variable spatial convolution-based methods convolve the image with masks that are adjusted point-by-point according to local geometric features. An idea close to this approach is modeling the image in small regions with low-degree polynomials whose coefficients are calculated using weighted least squares regression. Weighting is usually based on spatial distance or intensity difference between pixels. The latest works in this field even provide the ability to preserve corners and junctions in addition to edges, for

example, by using Gabor filter responses in different directions and combining them with innovative statistical models.

Transform-domain methods are usually applied to small patches of the image. In these methods, a patch of the image is mapped onto an orthogonal basis such as wavelet or DCT. The smaller coefficients, which are usually associated with high-frequency signal components and thus noise, are removed or adjusted using methods like hard thresholding. Then the patch is reconstructed by applying the inverse transform. To prevent boundary discontinuities and distortions, this process is usually performed in a sliding window manner, and the results are averaged.

More advanced methods in this category utilize correlations between coefficients at different locations or scales, such as the BLS-GSM method. Non-local methods are based on the principle that many image patches, even if spatially distant, may have very similar structures. The famous NL-Means algorithm smooths the image by computing a weighted average of all similar patches for each reference patch. Weights are usually determined based on statistical similarity between patches. This approach has also been interpreted theoretically as equivalent to minimizing the conditional entropy of a central pixel's intensity given its neighbors. Combining non-local and transform-based methods led to the design of the BM3D algorithm, which is one of the most powerful denoising methods. In this method, for each reference patch, similar patches are selected and arranged in a three-dimensional stack. Then the stack is transformed using a three-dimensional basis (e.g., a combination of DCT and Haar), noisy coefficients are removed, and the stack is reconstructed. This process is repeated for the entire image and averaged. The second version of this algorithm (BM3D2) uses a non-local Wiener filter for final refinement. Although BM3D has very high performance, setting its parameters is complex and depends on precise selection of bases, patch sizes, thresholds, and similarity criteria. In some transform-domain methods, instead of using a fixed basis, bases are learned from the statistical features of image patches. The famous K-SVD algorithm is an example of this category, which learns both an overcomplete dictionary and sparse representation coefficients from noisy data alternately using column SVD. Its multi-scale version (MS-KSVD) has performance on par with BM3D. Also, approaches like KLLD with initial clustering of noisy patches and separate filtering of each cluster based on PCA have been introduced.

The proposed method in this project, aiming to utilize the strengths of algorithms like BM3D and K-SVD, employs a simpler approach. In it, instead of applying transforms to single patches, a group of similar patches is represented as a tensor stack, and using HOSVD, a suitable representation of the image's structural information is obtained. Denoising is performed through statistical thresholding of coefficients and final reconstruction through averaging, without the need for complex learning or multiple settings. To enhance the discussion with recent advancements, recent works have explored hybrid singular value thresholding operators for improved denoising efficiency. Additionally, for hyperspectral images, multi-modal and double-learning approaches using tensor nuclear norms have been proposed, and stable rank methods for tensor denoising have shown promise in amplification techniques.

# 2 Theoretical Foundations

In this section, the basic concepts and methods used in the HOSVD-based denoising algorithm are introduced. Understanding these concepts is essential for a precise comprehension of the algorithm's structure and functionality.

## 2.1 Singular Value Decomposition (SVD) and Its Application in Image Denoising

In singular value decomposition (SVD), any matrix $A$ of dimensions $m_1 \times m_2$ can be decomposed as:

$$A = USV^T \tag{1}$$

where $U$ is an $m_1 \times m_1$ orthogonal matrix, $V$ is an $m_2 \times m_2$ orthogonal matrix, and $S$ is an $m_1 \times m_2$ diagonal matrix with positive singular values. The columns of matrix $V$ and the columns of matrix $U$—called right and left singular vectors, respectively—are the eigenvectors of the column covariance matrix $A^TA$ and the row covariance matrix $AA^T$, respectively. The singular values in matrix $S$ are equal to the square root of the eigenvalues of matrix $A^TA$, or equivalently $AA^T$. SVD also provides an optimal tool for low-rank compression; such that the best approximation of matrix $A$ with rank $k$ (where $k < m_1$, $k < m_2$) is the optimal solution to the problem:

$$E(\tilde{A}) = \|A - \tilde{A}\|_2^2 \tag{2}$$

with the constraint $(\tilde{A}) = k$, given by:

$$\tilde{A} = U_k\tilde{S}V_k^T \tag{3}$$

where $U_k$ and $V_k$ include the first $k$ columns of $U$ and $V$, and $\tilde{S}$ includes the $k$ largest singular values. In natural images, singular values decrease exponentially, and SVD vectors often have frequency interpretations. For image denoising, the following process is usually performed: 1. Divide the image into square patches $p \times p$ (in a sliding window format) and apply SVD to each patch:

$$A_i = U_iS_iV_i^T \tag{4}$$

2. Modify or threshold the singular values $\{S_i\}$ to remove noise, especially small values. 3. Reconstruct each patch and average the results to determine the final value of each pixel. Methods such as: - Truncation to rank 1 or 2 - Hard thresholding with value $\sigma\sqrt{2\log p^2}$, based on minimum statistical risk theory for Gaussian noise. - Adjusting coefficients so that the standard deviation of the residual reaches $\sigma$. All are used in this framework.

## 2.2 Theoretical Evaluation with "Oracle Denoiser"

Although the above method is theoretically simple and efficient, its practical efficiency is limited because the eigenvectors obtained from the noisy patch are not necessarily able to completely separate the signal and noise. Here, two key observations are raised.

First, suppose $Q$ and $Q_n$ are corresponding patches from the clean image $A_C$ and the noisy image $A_n$. If the SVD decomposition of the noisy patch is $Q_n = U_nS_nV_n^T$, the

clean patch $Q$ projected onto the bases $(U_n, V_n)$ is represented as matrix $S_Q = U_n^T Q V_n$. This matrix is not diagonal and therefore contains more non-zero components than $S_n$. Although such a case occurs, if we can somehow change the entries in $S_n$ to match the values of $S_Q$, then we can achieve a superior denoising technique.

The second observation is that the added noise not only affects the singular values of a patch but also influences the singular vectors (orthonormal bases). Considering this, it is strange that many SVD-based denoising methods, instead of manipulating the orthonormal bases, focus only on further reducing (sparsifying) the singular values. We now perform the following experiment:

We start with a noisy image and assume that the actual singular vectors of the clean patch underlying each noisy patch in the image are provided by an "oracle." Suppose the SVD decomposition of the patch $Q$ from the clean image $A_C$ is $Q = USV^T$. Then, we project the noisy patch $Q_n$ onto the bases $(U, V)$ to obtain matrix $S_{Q_n} = U^T Q_n V$. Then, using a sliding window, we apply a hard threshold of $\sigma\sqrt{2\log p^2}$ on $S_{Q_n}$ and build the final result by averaging multiple hypotheses.

Although this method is practically impossible, two important points are extracted from it:

1. Modifying singular values alone is not sufficient; because the eigenvectors are also affected by noise.

2. If better bases than those directly extracted from the noisy patch can be obtained, for example by using information from other similar patches, the denoising performance can be improved.

These considerations are the foundation for moving from local SVD to HOSVD, which utilizes a multi-patch structure and produces more stable and noise-resistant bases by combining shared information between similar patches.

## 2.3 Patch Similarity in NL-SVD

Non-local Singular Value Decomposition (NL-SVD) and Patch Similarity Criterion: NL-SVD (non-local extension of SVD) using groups of similar patches: Continuing the previous ideas, in this section, a non-local method for denoising based on singular value decomposition is presented that uses a set of similar patches. In this method, for a reference patch from the noisy image, several similar patches across the image are selected. Suppose $K$ similar patches including the reference patch are found, denoted by $\{P_i\}$, $1 \leq i \leq K$. The goal is to find a pair of orthogonal matrices $U_k$ and $V_k$ that provide the best joint low-rank approximation for all these patches. This goal is expressed by minimizing the following energy function:

$$E(U_k, \{S_i^{(k)}\}, V_k) = \sum_{i=1}^{k} \|P_i - U_k S_i^{(k)} V_k^T\|_2^2 \tag{5}$$

where $\forall i, S_i^{(k)} \in \mathbb{R}^{k\times k}$. The solution is obtained using an iterative minimization process (starting from random initial conditions). Note that the matrices $\{S_i^{(k)}\}$ in this case are not diagonal, and the bases $(U_k, V_k)$ are not equivalent to the SVD bases of individual patches, but form a joint basis for all selected patches. An approximate solution for this

optimization is obtained by computing the eigenvectors of the row and column covariance matrices of the set of similar patches:

Row covariance matrix:

$$C_r = \sum_{i=1}^{k} P_i P_i^T \tag{6}$$

Column covariance matrix:

$$C_c = \sum_{i=1}^{k} P_i^T P_i \tag{7}$$

(corresponding to the $k$ largest eigenvalues) The bases $U$ and $V$ are determined by the eigenvectors corresponding to the largest eigenvalues of these matrices. These bases, called NL-SVD bases, have a structure close to DCT bases. We use this framework in a denoising algorithm and will show later that this approach yields very good denoising results. First, the given noisy image is divided into patches. For each reference patch $P$, similar patches are collected, and full-rank NL-SVD bases $U$ and $V$ are obtained (to eliminate the need to select the optimal rank $k$, as this value may differ for each patch). Then, each patch $P$ is projected onto the bases $(U, V)$ and the coefficient matrix

$$S(P) = U^T P V \tag{8}$$

is produced. Coefficients whose value is less than the threshold $\sigma\sqrt{2\log p^2}$ are removed (set to zero), and after that, by inverting the transform and averaging multiple hypotheses, the filtered (denoised) image is reconstructed. Patch Similarity Criterion in NL-SVD Selecting the patch similarity criterion: Given a reference patch $P_{ref}$ in a noisy image, the $K$ nearest neighbors can be found among the image patches. However, this requires selecting the value $K$, which may not be the same for each patch. Therefore, we use a distance threshold $\tau_d$ and select all patches $P_i$ such that:

$$\|P_{ref} - P_i\| < \tau_d \tag{9}$$

Assuming a fixed and known noise model as $\mathcal{N}(0, \sigma)$, if $P_{ref}$ and $P_i$ are different noisy versions of the same clean patch, the random variable below has a $\chi^2(n^2)$ distribution:

$$x = \sum_{k=1}^{n^2} \frac{(P_{ref,k} - P_{i,k})^2}{2\sigma^2} \tag{10}$$

The cumulative distribution function (CDF) of a $\chi^2(n^2)$ random variable is defined as:

$$F(x, z) = \gamma\left(\frac{x}{2}, \frac{z}{2}\right) \tag{11}$$

where $\gamma(x, a)$ is the incomplete gamma function and is defined as:

$$\gamma(x, a) = \frac{1}{\Gamma(a)} \int_0^x e^{-t} t^{(a-1)} dt \tag{12}$$

and $\Gamma(a) = \int_0^\infty e^{-t} t^{(a-1)} dt$ is the gamma function.

$$\Gamma(a) = \int_0^\infty e^{-t} t^{(a-1)} dt \tag{13}$$

It is observed that if $z \geq 3$ and $x \geq 3z$, then:

$$F(x; z) \geq 0.99$$

Therefore, for patches of size $n \times n$ and noise with variance $\sigma$, we set the similarity threshold as $\tau_d = 6\sigma^2 n^2$.

$$\tau_d = 6\sigma^2 n^2 \tag{14}$$

In this way, if two patches are noisy versions of a common clean patch, this threshold considers them similar with very high probability. However, the converse is not necessarily true, so we may select patches that satisfy the threshold but are structurally completely different. This issue led us to use a statistical hypothesis test—namely the one-sided Kolmogorov-Smirnov (K-S) test—in the NL-SVD algorithm. To avoid the need to select a fixed significance level, we use the p-values output from the K-S test as weights in calculating the covariance matrices and rewrite them as follows:

$$C_r = \sum_{i=1}^{k} P_{ks}(P_{ref}, P_i) P_i P_i^T \tag{15}$$

$$C_c = \sum_{i=1}^{k} P_{ks}(P_{ref}, P_i) P_i^T P_i \tag{16}$$

where $P_{ks}(P_{ref}, P_i)$ is the p-value from the K-S test that checks how consistent the data $P_{ref} - P_i$ is with the distribution $\mathcal{N}(0, 2\sigma)$. In this way, a robust version of the 2D-SVD algorithm is obtained. In practice, we observed that if $\|P_{ref} - P_i\| > 3\sigma^2 n^2$, the value $P_{ks}(P_{ref}, P_i)$ is usually very close to zero. Therefore, in our experiments, we used the less conservative bound:

$$\tau_d = 3\sigma^2 n^2 \tag{17}$$

which led to increased computational speed. We also implemented a version of the algorithm where the statistical hypothesis test is completely omitted and equal weights are assigned to all patches. Surprisingly, this change did not cause a significant decrease in denoising performance. However, we still used the K-S test, as it is a principled method for reducing the effect of false positives. It is worth mentioning that the K-S test criterion was only used in the NL-SVD algorithm and not in the HOSVD algorithm.

## 2.4 Motivation and Implementation of HOSVD for Image Denoising

In the NL-SVD algorithm presented in the previous section, a reference patch $P_{ref}$ from the noisy image is considered, and it is assumed that the corresponding clean patch is $Q_{ref}$. Now, consider a scenario where all $K$ selected patches $\{P_i\}$ are noised versions of the patch $Q_{ref}$. In such a case, we will have:

$$\lim_{k \to \infty} \sum_{i=1}^{k} P_i P_i^T = Q_{ref} Q_{ref}^T + \sigma^2 I \tag{18}$$

Therefore, when the number of patches is large, the SVD bases of the patch $Q_{ref}$ can be estimated with good accuracy, approaching the oracle estimator mentioned in the

previous section. However, in most natural images, such conditions are not realized, and patches detected as similar are usually not noised versions of a completely identical patch. Therefore, the following principle is adopted: If a set of patches in the noisy image are similar to each other, the denoising process should consider this similarity and not denoise the patches independently of each other. Considering this principle, similar patches are grouped into a three-dimensional stack (equation 19). The main idea is that filtering is not only performed along the length and width of each two-dimensional patch but also applied in the third dimension to enable the use of similarity in intensity values at corresponding locations from different patches. This idea of joint filtering for multiple patches has been previously implemented in the BM3D algorithm. However, in this article, this idea is used to learn adaptive spatial bases.

## 2.5 Implementation of HOSVD for Denoising

First, using a square reference patch $p \times p$ from the noisy image $I_n$, a stack of $K-1$ similar patches is formed. The similarity criterion is specified as mentioned, and as a result, the value $K$ varies for each location in the image. This stack is denoted by $Z \in \mathbb{R}^{p \times p \times k}$.

HOSVD Decomposition The HOSVD decomposition on the stack $Z$ is performed as follows:

$$Z = S \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)} \tag{19}$$

where $U^{(1)} \in \mathbb{R}^{p \times p}$, $U^{(2)} \in \mathbb{R}^{p \times p}$, and $U^{(3)} \in \mathbb{R}^{k \times k}$ are orthogonal matrices, and $S \in \mathbb{R}^{p \times p \times k}$ is the three-dimensional coefficient tensor. The symbol $\times_n$ represents tensor multiplication in dimension $n$. The matrices $U^{(1)}, U^{(2)}, U^{(3)}$ are obtained by applying SVD to the unfolded tensors $Z_{(1)}, Z_{(2)}, Z_{(3)}$. The exact relations are as follows:

$$Z_{(k)} = U^{(k)} \cdot S^{(k)} \cdot (U^{\mod(k+1,3)} \otimes U^{\mod(k+2,3)})^T \quad 1 \leq k \leq 3 \tag{20}$$

which are equivalent representations for HOSVD. Computational Limitation The time complexity of computing SVD for $k \times k$ matrices is $O(k^3)$. To reduce computational cost, the limitation $k \leq 30$ is applied. Thresholding and Reconstruction After transforming the patches to the HOSVD domain, coefficient thresholding is performed. The threshold value is selected as follows:

$$\tau = \sigma\sqrt{2 \log p^2 k} \tag{21}$$

Then, the tensor $Z$ is reconstructed using the inverse HOSVD, and as a result, all patches in the stack are filtered. Overall Image Processing This process is repeated for all pixels in the image in a sliding window manner, and finally, the results from overlapping regions are averaged. Unlike NL-SVD, which only filters the reference patch, in this method, all patches present in each stack are filtered. This feature leads to better smoothing, which is necessary due to the limitation $k \leq 30$. Second Stage: Wiener Filter (HOSVD2) For further improvement, a second stage of filtering with the Wiener filter is performed: Suppose $\hat{Z}$ is a stack of similar patches in the image filtered with HOSVD (using the same statistical similarity criterion used in the first stage), and $Z_n$ is the corresponding stack in the noisy image. The HOSVD coefficients related to $\hat{Z}$ and $Z_n$ are denoted by $\hat{c}$ and $c_n$, respectively, expressed in the HOSVD bases obtained from $\hat{Z}$. In the second filtering stage, called HOSVD2, the filtered coefficients of $Z_n$, namely $\hat{c}_n$, are calculated as follows:

$$\hat{c}_n = \frac{c_n \hat{c}^2}{\hat{c}^2 + \sigma^2} \tag{22}$$

Then, by applying the inverse HOSVD transform and averaging the overlapping patches, the final image is reconstructed. This formula is a version of the Wiener filter that corrects the noisy image coefficients based on the estimate of coefficients from the previously filtered image. The goal of this stage is to further improve image quality using a better statistical estimate for the principal components of the image.

## 2.6 Comparison of NL-SVD and HOSVD

At first glance, the HOSVD and HOSVD2 algorithms are similar to BM3D in terms of the group filtering idea. However, there are important differences: 1. The bases in HOSVD are adaptive and learned from the data itself, while BM3D uses fixed bases like Haar. 2. In BM3D, the third dimension of the stack is considered as a continuous signal and the order of patches matters, but in HOSVD, the order of patches in the third dimension has no effect on the final result (except for sorting changes). Conceptually, the HOSVD algorithm contrasts with approaches like K-SVD or KLLD that learn global or clustered dictionaries with iterative optimization. HOSVD and NL-SVD learn bases adaptively and locally, without the need for complex or time-consuming learning. Compared to PCA-based methods that extract feature vectors from vectorized patches, the proposed methods NL-SVD and HOSVD extract bases from the matrix representation of patches, which is computationally much more efficient. Experimental results have shown that HOSVD and HOSVD2 outperform many existing methods in both grayscale and color images. Moreover, all parameters of these algorithms are set in a principled manner based on the noise standard deviation. It is worth mentioning that all presented methods, including NL-SVD and HOSVD/HOSVD2, are designed for removing Gaussian noise with zero mean and specified variance. However, the authors of the article have also provided an indirect and statistical method for estimating the noise standard deviation based on the features of the difference image (residual image).

# 3 Method

The proposed algorithm for image denoising using higher-order singular value decomposition (HOSVD) includes several key steps, which are described in order below. These steps are applicable to both grayscale and color images (with different details in tensor representation).

1. Patch Extraction For each pixel in the noisy image, a square patch of fixed size $p \times p$ is extracted as the "reference patch." This size is usually considered 8x8.

2. Patch Grouping For each reference patch, similar patches from different regions of the image are selected.

Patch similarity is calculated using the Euclidean distance between them and then filtered with a statistical threshold based on the Gaussian noise model:

$$\|P_{ref} - P_i\| < \tau_d = 3\sigma^2 p^2 \tag{23}$$

If using the K-S statistical test, the weighting coefficient for each similar patch is set with the p-value of the test. 3. 3D Stack Construction The selected similar patches are organized into a three-dimensional tensor $Z \in \mathbb{R}^{p \times p \times k}$ (or four-dimensional for color images).

$$Z \in \mathbb{R}^{p \times p \times k} \tag{24}$$

Here, $k$ is the number of similar patches (usually between 20 and 30). 4. Applying HOSVD The tensor $Z$ is decomposed using HOSVD:

$$Z = S \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)} \tag{25}$$

where $S \in \mathbb{R}^{p \times p \times k}$ is the coefficient tensor, and $U^{(1)} \in \mathbb{R}^{p \times p}$, $U^{(2)} \in \mathbb{R}^{p \times p}$, $U^{(3)} \in \mathbb{R}^{k \times k}$ are the orthogonal matrices related to the three dimensions of the tensor. This step provides optimal data-driven bases for representing similar patches. 5. Thresholding of Coefficients The HOSVD coefficients in S are filtered using hard thresholding. The threshold is determined as follows:

$$\tau = \sigma \sqrt{2 \log p^2 k} \tag{26}$$

All coefficients smaller than this value are set to zero. This step causes the removal of dominant noise components. 6. Inverse Transform After thresholding, using the inverse HOSVD, the tensor $Z$ is reconstructed, and the denoised patches are obtained. 7. Aggregation / Averaging Each pixel may be present in several different patches. Therefore, to determine the final value of each pixel in the reconstructed image, simple averaging is used between all estimates obtained for that pixel. This process increases smoothness and reduces the final image noise. 8. Optional Wiener Filter (Second Stage) In the more advanced version of the algorithm called HOSVD2, after producing the initial smoothed image, a Wiener filter stage is also executed. In this stage, for each stack of similar patches (from the initial image), the HOSVD coefficients are recalculated and corrected:

$$\hat{c}_n = \frac{c_n \hat{c}^2}{\hat{c}^2 + \sigma^2} \tag{27}$$

where $c_n$ are the noisy image coefficients, and $\hat{c}$ are the coefficients of the reconstructed image in the first stage. This filter increases reconstruction accuracy, especially in high-noise conditions.

# 4 Experimental Results on Grayscale Images

In this section, we observe how denoising methods like NL-SVD, HOSVD, and HOSVD2 perform on grayscale images and compare them with other methods. 13 famous images of size 512x512 (like the Barbara image) were used, and Gaussian noise with various intensities (15, 20, 25, 30, and 35) was added to them. The methods were compared with NL-Means, K-SVD, BM3D (two types: BM3D1 and BM3D2), a special version of BM3D called D-DCT3, the Oracle method, and two stages of LPG-PCA. To measure quality, two criteria were used: PSNR (higher means cleaner image) and SSIM (shows how close the image is to the original version). The images were saved and reloaded to simulate real conditions, and noise was created by adding Gaussian noise. For NL-SVD and HOSVD, 8x8 patches with a search radius of 20 were used (larger radius only reduced speed), and for NL-Means and BM3D, specific settings were applied. Results for noise 20 and 30 are given in tables (1 and 2) and show that HOSVD is better than other methods like NL-Means and D-DCT3, even surpassing K-SVD and BM3D1 in high noise, but slightly weaker than BM3D2. Oracle always had the best result, and by adding the Wiener filter to HOSVD (called HOSVD2), performance approached BM3D2 and sometimes better in very high noise.

**Comparison with K-SVD:** NL-SVD is almost identical to K-SVD, but HOSVD and HOSVD2 perform better. K-SVD may yield weak results with complex dictionary computations, while NL-SVD and HOSVD are simpler and easier to parameterize.

**Comparison with BM3D and D-DCT3:** BM3D has many parameters that are hard to set, but NL-SVD and HOSVD select thresholds better using the noise model. D-DCT3 is weaker than both because it keeps bases fixed.

**Visual Comparison:** Figures 5, 6, and 7 show original images, noisy (with intensity 20), and filtered with various methods. BM3D2 sometimes creates strange artifacts (like on Barbara's face), but NL-SVD does not have these artifacts and is slightly blurry. HOSVD preserves edges better, and Oracle has the best quality. The residuals (difference between noisy and filtered) were also examined, and BM3D2 and HOSVD are closer to Oracle.

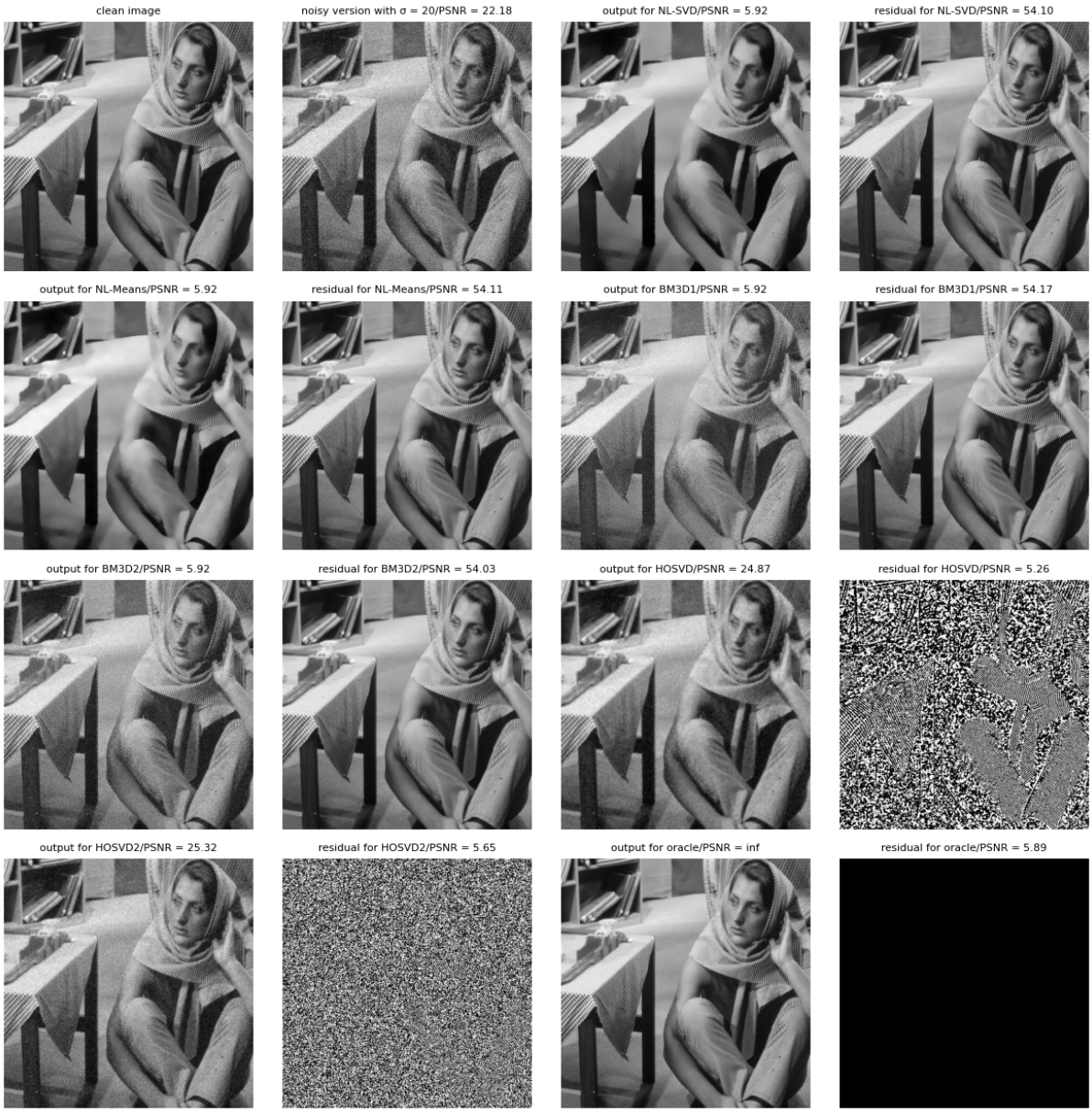



Figure 1: Denoising results on Barbara image with noise level 20.

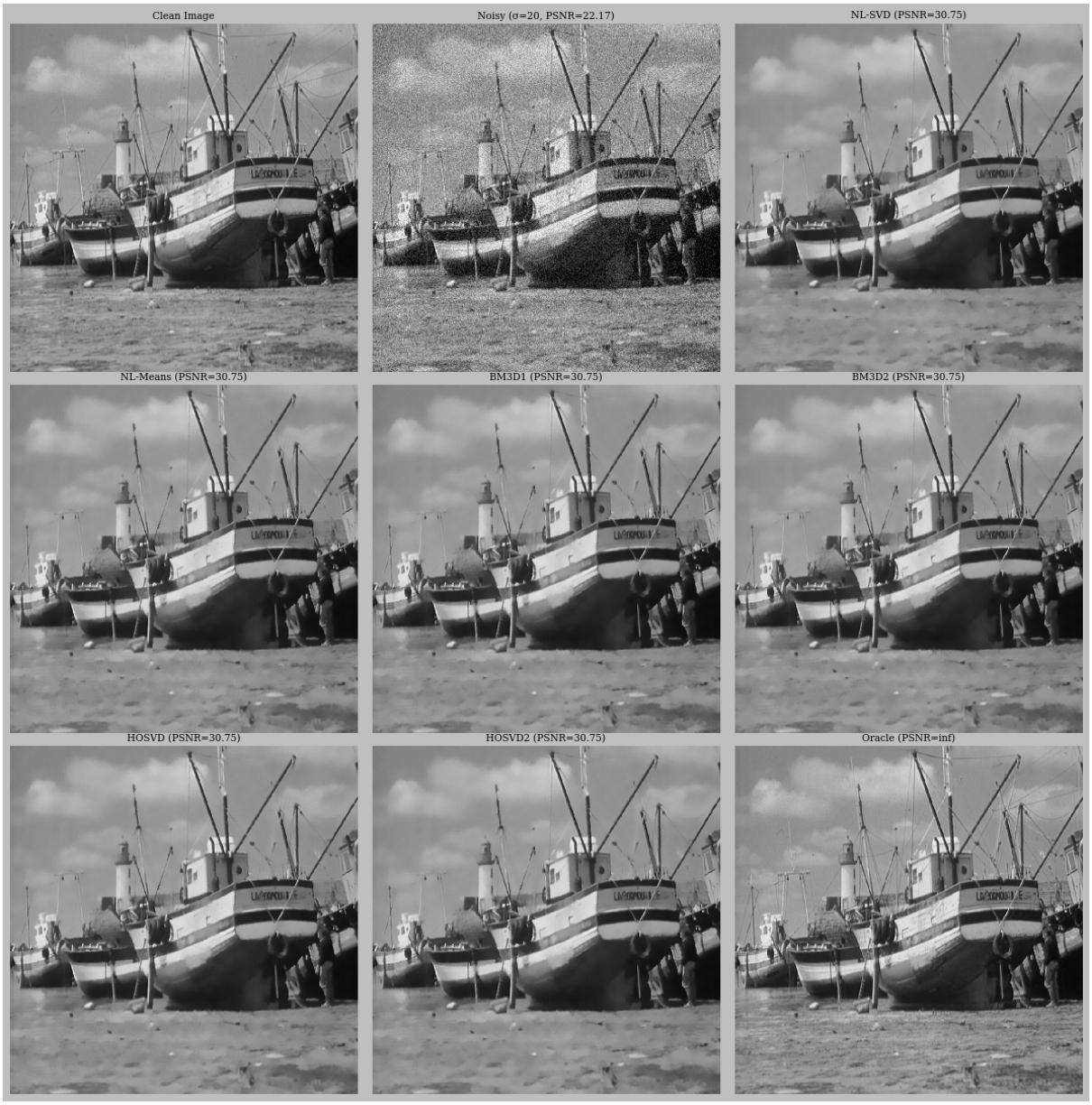



Figure 2: Denoising results on Boat image with noise level 20.

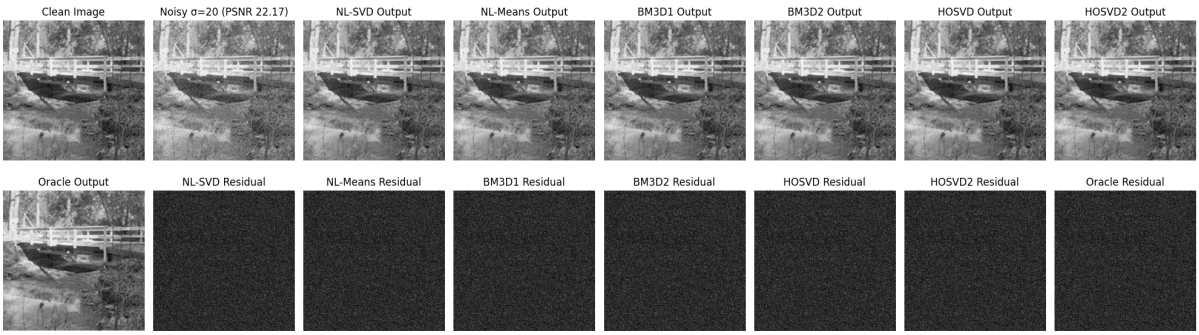



Figure 3: Residual images for different methods.

**Computational Complexity Comparison:** In this section, the time required to run various denoising methods was examined. Assuming the number of pixels in the image is $N$, the time to find similar patches for each reference patch is $T_s$, the number of similar patches is $K$, and the size of each patch is $p \times p$. The NL-SVD method has complexity $O((T_s + Kp^3)N)$, because both singular value decomposition and matrix multiplication depend on $O(p^3)$. BM3D requires $O(Kp^3)$ for two-dimensional transforms and $O(K^2p^2)$ for the third dimension, totaling $O((T_s+Kp^3+K^2p^2)N)$. If using fast Fourier transform (FFT), the complexity reduces to $O((T_s + Kp^2 \log p + p^2K \log K)N)$. If $p < K$ (which is good for non-local methods), NL-SVD is faster than the optimized version of BM3D. The complexity of HOSVD is calculated similarly: for the stack $p \times p \times K$, two unfoldings are $O(Kp^3)$ and the third unfolding is $O(\min(K^2p^2, Kp^4))$, totaling $O(T_s + Kp^3 + \min(K^2p^2, Kp^4)N)$. PCA methods with complexity $O((T_s+Kp^4+p^6)N)$ and K-SVD with $O(p^2KLNJ)$ (where $L$ is the sparsity factor and $J$ is iterations) are much heavier. NL-SVD and HOSVD are more efficient than these methods due to using matrix representation of patches.


Figure 4: Denoising results comparison on Lena image.

Optimal Local Patch Size Selection and Noise Standard Deviation Estimation Up to this stage, results with fixed 8x8 patches were reported, which is common in patch-based methods. With NL-SVD and threshold $\sqrt{2 \log p^2}$, residual images were divided into patches $q \times q$ (from 8 to 16). For each $q$, the absolute correlation coefficient between patch pairs was calculated (with formula

$$\rho_q(v_1, v_2) = \frac{1}{q^2} \frac{|(v_1 - \mu_1)^T (v_2 - \mu_2)|}{\sigma_{v1}\sigma_{v2}} \tag{28}$$

where $\mu$ is the mean and $\sigma$ is the standard deviation). The goal was for residuals to be low-correlated. The sum of these values $\mu$ for all $q$ and pairs was calculated, and the patch with the lowest $\mu$ was selected as the best (tested from 3 to 16). Table 1 showed that PSNR is close to the best value with this method, and its drop is minimal. Figure 9 shows filtered images and residuals with different patches (for noise 20). Recall that adaptive HOSVD with local structure-based patch is consistent, but due to dependence on adjacent patches, this is difficult. Also, the mean standard deviation for noise variance estimation from NL-SVD residuals (with 8x8 patch) was used.

Table 1: Optimal patch size by PSNR for different images.

| Image # | Best PSNR | Best patch-size (by PSNR) | $\mu$ | Best patch-size (by $\mu$) | Best PSNR (by $\mu$) |
|---|---|---|---|---|---|
| 13 | 32.000 | 8 | 9.648 | 14 | 31.690 |
| 12 | 30.990 | 10 | 9.650 | 11 | 30.958 |
| 11 | 30.260 | 8 | 9.640 | 14 | 29.910 |
| 10 | 30.030 | 9 | 9.650 | 11 | 29.988 |
| 9 | 31.210 | 8 | 9.713 | 16 | 30.968 |
| 8 | 28.120 | 8 | 9.680 | 14 | 28.110 |
| 7 | 30.190 | 10 | 9.644 | 14 | 30.024 |
| 6 | 29.150 | 5 | 9.644 | 9 | 28.890 |
| 5 | 27.190 | 8 | 9.736 | 6 | 26.804 |
| 4 | 26.166 | 4 | 9.759 | 12 | 31.806 |
| 3 | 32.020 | 8 | 9.639 | 12 | 27.029 |
| 2 | 27.020 | 5 | 9.673 | 11 | 35.499 |
| 1 | 33.510 | 10 | 9.635 | 11 | 35.499 |

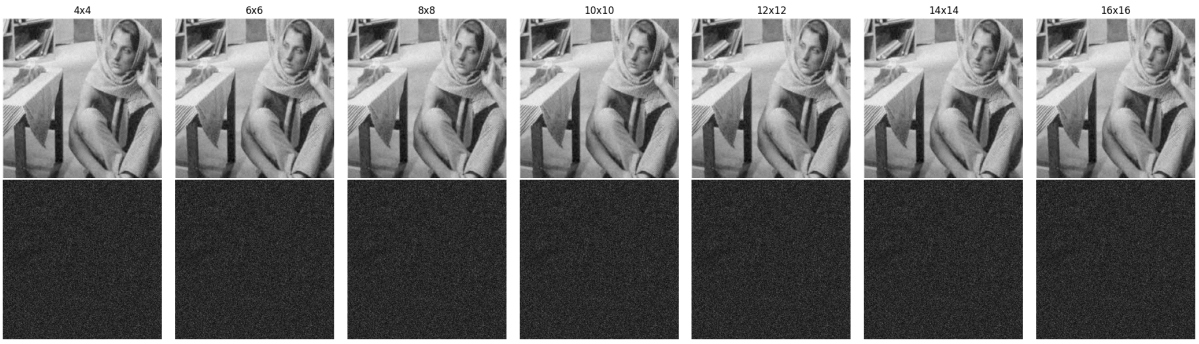



Figure 5: Denoised and residual images for different patch sizes on Boat image with noise 20.

The "Boat" image with noise from 5 to 50 was tested, and with $t$ from 2 to 82, $\hat{\sigma}$ was selected with the minimum $\mu$. The results were accurate, and Figure 10 shows the graph of $\mu$ versus $t$ for noise 20 and 40.

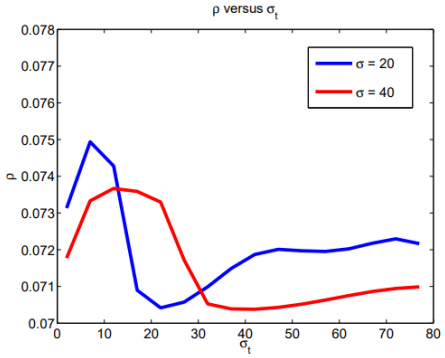



Figure 6: Graph of $\mu$ versus $t$ for noise levels 20 and 40.

# 5 Experimental Results on Color Images

Here, the performance of the methods on 24 RGB color images from the Kodak gallery with independent Gaussian noise ($\mathcal{N}(0, \sigma)$) in each channel (with $\sigma$ equal to 30, 40, and 50) was examined. Tables 2, 3, and 4 show PSNR and SSIM values. Like grayscale images, noise was created by adding Gaussian noise and converting to the range 0 to 255. The D-HOSVD4 method (HOSVD on a 4-dimensional stack with channels as the fourth dimension) was compared with NL-Means and the color version of BM3D in YCbCr space. Also, 4D-HOSVD2 (with Wiener filter) and 3D-IHOSVD (HOSVD after separating channels with PCA) were implemented. For 3D-IHOSVD, patch similarity with threshold $d = 3\sigma^2 n^2$ in three channels,

$$d = 3\sigma^2 n^2 \tag{29}$$

and for 4D-HOSVD/4D-HOSVD2 with $d = 3.3\sigma^2 n^2$.

$$d = 3.3\sigma^2 n^2 \tag{30}$$

All methods worked with 8x8 patches. 4D-HOSVD was better than NL-Means and 3D-IHOSVD and sometimes surpassed BM3D1, but its PSNR was slightly less than BM3D2. 4D-HOSVD2 was better than BM3D2 in 18 images (for $\sigma = 40$) and 15 images (for $\sigma = 50$). In high noise, 3D-IHOSVD and BM3D2 clipped colors, but 4D-HOSVD and 4D-HOSVD2 did not have this problem (Figures 11 and 12). So 4D-HOSVD2 is an advanced method for color images. The considered method was 1 to 1.5 dB better than LPG-PCA, because LPG-PCA processes channels independently.

Table 2: PSNR/SSIM for color images with noise level 30.

| Image # | BM3D1 | BM3D2 | NL-Means | 3D-IHOSVD | 4D-HOSVD | 4D-HOSVD2 |
|---|---|---|---|---|---|---|
| 1 | 27.314 / 0.766 | 29.478 / 0.742 | 26.155 / 0.791 | 27.027 / 0.771 | 27.341 / 0.798 | 27.905 / 0.807 |
| 2 | 30.821 / 0.842 | 31.254 / 0.816 | 28.958 / 0.732 | 30.532 / 0.788 | 30.827 / 0.796 | 31.388 / 0.823 |
| 3 | 28.971 / 0.819 | 29.497 / 0.815 | 26.262 / 0.723 | 28.402 / 0.816 | 28.906 / 0.839 | 29.142 / 0.822 |
| 4 | 29.678 / 0.813 | 29.987 / 0.799 | 26.842 / 0.723 | 29.081 / 0.789 | 29.306 / 0.809 | 31.423 / 0.807 |
| 5 | 27.840 / 0.838 | 28.385 / 0.859 | 26.110 / 0.766 | 27.797 / 0.826 | 27.846 / 0.834 | 28.348 / 0.861 |
| 6 | 26.640 / 0.818 | 27.013 / 0.840 | 26.262 / 0.799 | 28.031 / 0.826 | 28.259 / 0.860 | 28.890 / 0.825 |
| 7 | 28.214 / 0.794 | 28.844 / 0.855 | 25.792 / 0.724 | 28.442 / 0.837 | 28.801 / 0.850 | 28.881 / 0.833 |
| 8 | 30.588 / 0.871 | 31.400 / 0.881 | 25.789 / 0.764 | 31.025 / 0.870 | 31.440 / 0.896 | 32.912 / 0.892 |
| 9 | 32.158 / 0.871 | 33.060 / 0.886 | 27.170 / 0.675 | 32.735 / 0.826 | 32.826 / 0.834 | 33.012 / 0.892 |
| 10 | 30.444 / 0.890 | 30.995 / 0.823 | 27.356 / 0.734 | 30.919 / 0.827 | 31.260 / 0.870 | 32.791 / 0.812 |
| 11 | 31.719 / 0.838 | 32.380 / 0.884 | 27.974 / 0.737 | 31.952 / 0.832 | 31.980 / 0.832 | 32.276 / 0.844 |
| 12 | 30.449 / 0.820 | 30.995 / 0.822 | 27.356 / 0.734 | 30.919 / 0.827 | 31.260 / 0.870 | 32.791 / 0.812 |
| 13 | 25.857 / 0.744 | 26.350 / 0.763 | 24.832 / 0.739 | 25.761 / 0.745 | 25.427 / 0.713 | 26.078 / 0.749 |
| 14 | 24.810 / 0.739 | 25.872 / 0.799 | 26.972 / 0.741 | 25.367 / 0.793 | 25.270 / 0.784 | 25.854 / 0.835 |
| 15 | 28.518 / 0.804 | 28.499 / 0.815 | 26.714 / 0.720 | 28.175 / 0.801 | 28.688 / 0.819 | 29.713 / 0.829 |
| 16 | 30.616 / 0.809 | 31.335 / 0.840 | 28.192 / 0.716 | 30.420 / 0.817 | 30.421 / 0.807 | 31.202 / 0.828 |
| 17 | 30.755 / 0.846 | 31.210 / 0.861 | 28.426 / 0.771 | 30.607 / 0.839 | 30.686 / 0.850 | 31.051 / 0.864 |
| 18 | 28.399 / 0.800 | 28.984 / 0.834 | 26.399 / 0.641 | 28.377 / 0.824 | 28.542 / 0.710 | 29.647 / 0.747 |
| 19 | 28.256 / 0.770 | 29.143 / 0.789 | 26.330 / 0.666 | 28.986 / 0.767 | 29.070 / 0.784 | 29.337 / 0.794 |
| 20 | 28.407 / 0.886 | 28.189 / 0.892 | 27.253 / 0.840 | 28.002 / 0.881 | 28.350 / 0.787 | 28.456 / 0.884 |
| 21 | 29.370 / 0.796 | 29.871 / 0.807 | 27.448 / 0.714 | 29.320 / 0.793 | 29.424 / 0.818 | 29.710 / 0.800 |
| 22 | 28.046 / 0.726 | 28.342 / 0.794 | 26.081 / 0.632 | 27.456 / 0.704 | 28.053 / 0.772 | 28.520 / 0.785 |
| 23 | 32.414 / 0.837 | 32.248 / 0.911 | 27.597 / 0.779 | 32.032 / 0.836 | 32.036 / 0.903 | 33.379 / 0.907 |
| 24 | 28.056 / 0.828 | 28.503 / 0.849 | 26.552 / 0.759 | 27.930 / 0.831 | 27.834 / 0.831 | 28.505 / 0.840 |

Table 3: PSNR/SSIM for color images with noise level 40.

| Image # | BM3D1 | BM3D2 | NL-Means | 3D-IHOSVD | 4D-HOSVD | 4D-HOSVD2 |
|---|---|---|---|---|---|---|
| 1 | 26.314 / 0.790 | 26.478 / 0.747 | 25.155 / 0.676 | 26.143 / 0.742 | 26.045 / 0.712 | 26.472 / 0.740 |
| 2 | 29.197 / 0.746 | 29.497 / 0.777 | 27.202 / 0.670 | 29.043 / 0.761 | 29.236 / 0.748 | 29.574 / 0.772 |
| 3 | 27.905 / 0.786 | 28.200 / 0.829 | 25.965 / 0.801 | 27.796 / 0.824 | 27.906 / 0.834 | 28.514 / 0.832 |
| 4 | 29.567 / 0.739 | 29.818 / 0.791 | 26.262 / 0.723 | 29.227 / 0.774 | 29.795 / 0.777 | 30.085 / 0.790 |
| 5 | 25.466 / 0.724 | 25.758 / 0.798 | 24.181 / 0.697 | 25.099 / 0.762 | 25.535 / 0.776 | 26.430 / 0.812 |
| 6 | 25.467 / 0.840 | 26.024 / 0.881 | 26.002 / 0.696 | 26.246 / 0.899 | 26.789 / 0.882 | 27.361 / 0.890 |
| 7 | 26.508 / 0.758 | 27.013 / 0.840 | 23.862 / 0.723 | 26.031 / 0.826 | 26.259 / 0.860 | 26.890 / 0.825 |
| 8 | 26.258 / 0.828 | 26.339 / 0.832 | 24.064 / 0.753 | 26.248 / 0.838 | 26.192 / 0.829 | 26.781 / 0.844 |
| 9 | 30.644 / 0.834 | 31.160 / 0.846 | 27.170 / 0.675 | 30.735 / 0.826 | 30.826 / 0.834 | 31.012 / 0.892 |
| 10 | 30.034 / 0.790 | 30.595 / 0.823 | 27.356 / 0.734 | 30.919 / 0.827 | 31.260 / 0.870 | 32.791 / 0.812 |
| 11 | 30.219 / 0.739 | 30.051 / 0.816 | 27.340 / 0.666 | 29.442 / 0.726 | 29.054 / 0.807 | 30.085 / 0.790 |
| 12 | 27.272 / 0.745 | 28.060 / 0.759 | 25.792 / 0.662 | 27.742 / 0.754 | 27.706 / 0.750 | 28.120 / 0.771 |
| 13 | 26.857 / 0.744 | 27.135 / 0.736 | 25.184 / 0.664 | 26.807 / 0.734 | 26.527 / 0.725 | 27.024 / 0.741 |
| 14 | 28.518 / 0.804 | 28.499 / 0.815 | 26.714 / 0.720 | 28.175 / 0.801 | 28.688 / 0.819 | 29.713 / 0.829 |
| 15 | 25.158 / 0.802 | 25.133 / 0.875 | 25.181 / 0.864 | 25.308 / 0.874 | 25.260 / 0.884 | 25.718 / 0.815 |
| 16 | 30.616 / 0.809 | 31.335 / 0.840 | 28.192 / 0.716 | 30.420 / 0.817 | 30.421 / 0.807 | 31.202 / 0.828 |
| 17 | 29.735 / 0.766 | 29.761 / 0.719 | 28.275 / 0.711 | 28.875 / 0.766 | 29.011 / 0.718 | 29.842 / 0.718 |
| 18 | 28.396 / 0.800 | 28.984 / 0.834 | 26.399 / 0.641 | 28.377 / 0.824 | 28.542 / 0.710 | 29.647 / 0.747 |
| 19 | 26.273 / 0.716 | 26.607 / 0.733 | 25.399 / 0.641 | 25.377 / 0.724 | 25.542 / 0.710 | 26.647 / 0.747 |
| 20 | 28.412 / 0.786 | 28.871 / 0.807 | 27.253 / 0.740 | 28.002 / 0.781 | 28.350 / 0.787 | 28.456 / 0.884 |
| 21 | 24.321 / 0.813 | 24.304 / 0.821 | 24.308 / 0.811 | 24.508 / 0.831 | 24.423 / 0.819 | 24.723 / 0.819 |
| 22 | 26.959 / 0.657 | 27.779 / 0.693 | 27.114 / 0.675 | 27.976 / 0.849 | 27.412 / 0.861 | 27.882 / 0.864 |
| 23 | 29.255 / 0.820 | 29.749 / 0.862 | 26.976 / 0.779 | 29.113 / 0.836 | 29.136 / 0.903 | 30.379 / 0.907 |
| 24 | 26.058 / 0.769 | 26.639 / 0.794 | 24.858 / 0.676 | 25.178 / 0.730 | 25.122 / 0.786 | 25.505 / 0.780 |

Table 4: PSNR/SSIM for color images with noise level 50.

| Image # | BM3D1 | BM3D2 | NL-Means | 3D-IHOSVD | 4D-HOSVD | 4D-HOSVD2 |
|---|---|---|---|---|---|---|
| 1 | 24.976 / 0.658 | 25.330 / 0.683 | 24.995 / 0.646 | 25.332 / 0.671 | 25.232 / 0.671 | 25.732 / 0.671 |
| 2 | 27.575 / 0.706 | 27.761 / 0.719 | 28.275 / 0.711 | 28.242 / 0.718 | 28.842 / 0.718 | 29.842 / 0.718 |
| 3 | 29.105 / 0.780 | 29.000 / 0.829 | 29.365 / 0.801 | 29.551 / 0.836 | 29.900 / 0.745 | 29.900 / 0.745 |
| 4 | 28.284 / 0.707 | 28.719 / 0.748 | 28.654 / 0.736 | 28.846 / 0.746 | 29.000 / 0.745 | 29.000 / 0.745 |
| 5 | 24.226 / 0.689 | 24.663 / 0.718 | 24.212 / 0.689 | 25.577 / 0.698 | 25.588 / 0.724 | 25.588 / 0.724 |
| 6 | 23.703 / 0.701 | 24.005 / 0.732 | 25.577 / 0.689 | 25.773 / 0.688 | 25.890 / 0.714 | 25.890 / 0.714 |
| 7 | 28.124 / 0.794 | 28.844 / 0.855 | 28.342 / 0.837 | 28.801 / 0.850 | 28.881 / 0.833 | 28.881 / 0.833 |
| 8 | 24.478 / 0.771 | 24.931 / 0.794 | 24.661 / 0.769 | 25.222 / 0.793 | 25.222 / 0.793 | 25.222 / 0.793 |
| 9 | 28.421 / 0.780 | 29.028 / 0.816 | 29.213 / 0.824 | 28.884 / 0.830 | 28.804 / 0.850 | 28.804 / 0.850 |
| 10 | 28.844 / 0.751 | 29.564 / 0.815 | 29.120 / 0.791 | 29.546 / 0.805 | 29.546 / 0.805 | 29.546 / 0.805 |
| 11 | 28.366 / 0.743 | 29.467 / 0.789 | 29.435 / 0.775 | 29.635 / 0.784 | 29.946 / 0.783 | 29.946 / 0.783 |
| 12 | 25.036 / 0.653 | 25.247 / 0.700 | 25.434 / 0.694 | 25.620 / 0.736 | 26.031 / 0.701 | 26.031 / 0.701 |
| 13 | 25.365 / 0.643 | 25.371 / 0.654 | 26.516 / 0.648 | 26.003 / 0.674 | 26.003 / 0.674 | 26.003 / 0.674 |
| 14 | 25.582 / 0.567 | 25.791 / 0.584 | 26.622 / 0.544 | 26.168 / 0.581 | 26.003 / 0.674 | 26.003 / 0.674 |
| 15 | 26.727 / 0.765 | 26.754 / 0.779 | 26.599 / 0.768 | 26.865 / 0.714 | 26.785 / 0.772 | 26.785 / 0.772 |
| 16 | 26.109 / 0.686 | 28.574 / 0.737 | 28.265 / 0.714 | 28.514 / 0.744 | 28.489 / 0.732 | 28.489 / 0.732 |
| 17 | 29.390 / 0.732 | 29.277 / 0.654 | 27.132 / 0.745 | 27.532 / 0.753 | 27.535 / 0.753 | 27.535 / 0.753 |
| 18 | 28.742 / 0.643 | 28.597 / 0.727 | 27.624 / 0.763 | 27.174 / 0.653 | 27.147 / 0.653 | 27.147 / 0.653 |
| 19 | 27.590 / 0.712 | 28.050 / 0.725 | 27.569 / 0.719 | 27.659 / 0.745 | 27.624 / 0.739 | 27.624 / 0.739 |
| 20 | 28.421 / 0.786 | 28.871 / 0.807 | 27.253 / 0.740 | 28.002 / 0.781 | 28.350 / 0.787 | 28.456 / 0.884 |
| 21 | 26.339 / 0.727 | 26.909 / 0.782 | 26.344 / 0.754 | 26.881 / 0.781 | 26.773 / 0.773 | 26.773 / 0.773 |
| 22 | 29.255 / 0.820 | 29.749 / 0.862 | 26.976 / 0.779 | 29.113 / 0.836 | 29.136 / 0.903 | 30.379 / 0.907 |
| 23 | 24.488 / 0.654 | 24.851 / 0.698 | 24.886 / 0.681 | 24.851 / 0.698 | 24.712 / 0.691 | 24.712 / 0.691 |
| 24 | 24.920 / 0.693 | 25.296 / 0.731 | 24.901 / 0.697 | 25.730 / 0.737 | 25.422 / 0.734 | 25.422 / 0.734 |


Figure 7: Color denoising results with noise level 40.

Figure 8: Color denoising results with noise level 50.

# 6 Combining the Algorithm with Learning Neural Networks

Automatic Denoising Implementation for Images by Integrating HOSVD and ResNet Neural Network as a Hybrid Approach HOSVD is a tensor decomposition method that models the image as a higher-order tensor and reduces noise by removing low-importance components (based on threshold). In this method, the image is decomposed as a three-dimensional tensor (height, width, color channels) and the Core and Factors are obtained. Neural networks, especially deep architectures like ResNet, with the ability to learn complex features, can predict optimal thresholds.

## 6.1 Model Design

- **ResNet Neural Network:** A lightweight ResNet architecture with three residual blocks has been designed. Each block includes two convolutional layers with 64 and 128 filters, optimized with Batch Normalization and ReLU activation. Residual connections prevent vanishing gradients, and the output is an adaptive threshold between 0 and 1 produced by a Dense layer and Sigmoid activation.
- **Adaptive HOSVD:** The threshold predicted by ResNet is limited to the range 0.05 to 0.15 to prevent excessive detail removal. This threshold is multiplied by the maximum singular value in the HOSVD core.
- **Fixed HOSVD:** With a fixed threshold of 0.1 implemented, manually adjusted and used for comparison with the adaptive method.

## 6.2 Preprocessing and Postprocessing

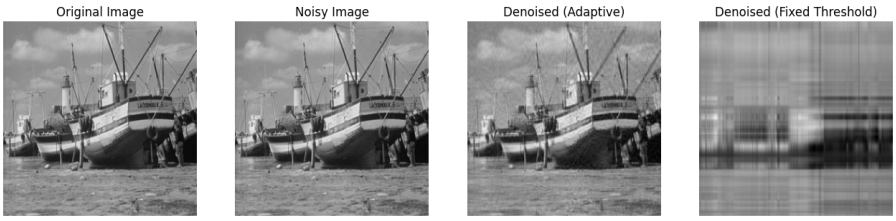
Images are resized to 256 x 256 and preprocessed with a Gaussian filter with sigma 0.3. Gaussian noise with variance 0.01 is added to simulate noisy conditions. **Implementation**

**Image Loading:** The input image is uploaded through the Colab user interface.

**Pre-training:** A quick training with 5 epochs on synthetic noisy data and original images is performed.

**Prediction and Processing:** ResNet predicts the threshold, and adaptive HOSVD is applied. Then, HOSVD with fixed threshold is executed separately.

**Result Display:** Four images (original, noisy, adaptively reconstructed, and reconstructed with fixed threshold) are displayed side by side.




## 6.3 Results

- **Original Image:** Details of the boat and background are clear.

- **Noisy Image:** Added Gaussian noise has reduced the clarity of details.
- **Adaptively Reconstructed Image (Adaptive):** Details are relatively preserved, but some information is lost and the image is slightly blurred.
- **Reconstructed Image with Fixed Threshold:** The quality is much lower, and the image is severely blurred and indistinguishable.
- The results show that the adaptive method has relative superiority over the fixed threshold method, but the reconstruction quality is still not desirable.

Observations indicate that simple pre-training with 5 epochs and random initial weights does not provide sufficient accuracy for threshold prediction. This may be due to the lack of training data and complex noise. The fixed threshold HOSVD, due to lack of adaptation to image features, has provided weaker results. For improvement, it is suggested:

1. Use larger datasets (like BSDS500) for model training.
2. Increase the number of epochs or use transfer learning techniques.
3. More precise adjustment of HOSVD parameters (e.g., higher ranks or dynamic thresholds).

# 7 Conclusions

In this project, a simple but effective algorithm for image denoising was introduced, based on higher-order singular value decomposition (HOSVD) and learning local and data-adaptive transform bases. The main idea of the proposed method is to consider similar patches in the image as a three-dimensional stack (or four-dimensional for color images) and remove noise from their transform coefficients by applying HOSVD. Then, by reconstructing the patches and averaging the results, the final smoothed image is produced. The important advantages of this algorithm include:

1. Full use of non-local similarities between image patches;
2. Data-driven learning of transform bases, instead of using fixed transforms like DCT or wavelet.
3. Principled and statistical selection of all parameters, including thresholding and patch similarity criterion.
4. Simplicity of implementation compared to advanced algorithms like BM3D or K-SVD.
5. Direct extension to color images using four-dimensional HOSVD.

Experimental results showed that the proposed method, especially its improved version (HOSVD2), competes with advanced algorithms like BM3D in many cases and even provides better performance at high noise levels. Also, numerical and visual analyses showed that the HOSVD method has an advantage in reconstructing fine image structures (such as fabric texture or thin edges) compared to other methods and prevents the creation of artificial noise and unrealistic structures.

# 8 Future Suggestions

Given the obtained results, the following paths are suggested for continuing the development and improvement of the method:

- Adaptation to more complex types of noise such as non-Gaussian noise or spatially heterogeneous noise.
- Adaptive selection of patch size and number of similar patches in different regions of the image, instead of using fixed size and number.
- Development of a faster version using fast SVD/HOSVD algorithms or intelligent dimension reduction before applying the transform.
- Examining the applications of the algorithm in fields like compression, image inpainting, and super-resolution.

# 9 References


1. A. Rajwade, A. Rangarajan, and A. Banerjee, “Image denoising using the higher order singular value decomposition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 4, pp. 849–862, 2013.

2. M. Aharon, M. Elad, and A. Bruckstein, “The K-SVD: an algorithm for designing of overcomplete dictionaries for sparse representation,” IEEE Trans. Signal Process., vol. 54, no. 11, pp. 4311–4322, 2006.

3. K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3-D transform-domain collaborative filtering,” IEEE Trans. Image Process., vol. 16, no. 8, pp. 2080–2095, 2007.

4. H. C. Andrews, ”Image Denoising Using Hybrid Singular Value Thresholding Operators,” in IEEE Access, vol. 8, pp. 1802-1810, 2020, doi: 10.1109/ACCESS.2020.2964683.

5. Y. Zhang, ”Hyperspectral Image Denoising via Multi-modal and Double-learning Tensor Nuclear Norm Minimization,” arXiv preprint arXiv:2101.07681, 2021.

6. A. Banerjee, ”Tensor Denoising via Amplification and Stable Rank Methods,” arXiv preprint arXiv:2301.03761, 2023.


# 10 Appendix

The complete project implementation, along with related documentation, is available in the GitHub repository below:

`https://github.com/Zohreh004/Implementation-Final-Project`