IET Research Journals

IET Journals
The Institution of Engineering and Technology

Image Denoising using Patch Ordering and 3D Transformation of Patches

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Ozden Colak[1,2], *Ender M. Eksioglu*[2]

[1] *TÜBİTAK BİLGEM Information Technologies Institute, Kocaeli, Türkiye*
[2] *Istanbul Technical University, Electronics and Communication Engineering, İstanbul, Türkiye*
* *E-mail: ozden.colak@tubitak.gov.tr , eksioglue@itu.edu.tr*

Abstract: We propose a novel image denoising algorithm which is based on the ordering of noisy image patches into a 3D array and the application of 3D transformations on this image dependent patch cube. For a given noisy image, we extract all the patches with overlaps. Then, we order these patches according to a predefined similarity measure. After the reordering, a possibly separable 3D transformation is applied to the reordered 3D patch cube. The transform domain coefficients are thresholded using a suitably calculated thresholding parameter. Afterwards, the proper 3D inverse transformation is applied to these coefficients. The final denoised image is generated by repositioning the processed patches to their original locations on the image canvas. The developed algorithm presents a novel and efficient combination of patch ordering and 3D transformations. The forward analysis transform as defined by this complex procedure can get restated as the application of a single tight frame. This tight frame depends on the noisy image under consideration. This novel, image dependent forward operator which employs 3D transforms results in improved denoising performance. The experimental results indicate that the proposed algorithm achieves state-of-the-art denoising results with complexity comparable to competing methods.

Keywords: Image denoising, transforms, sparse representation

1 Introduction

The history of image denoising is almost as long as image processing itself [1]. Examples of early image denoising methods include spatial domain filters such as averaging filter, median filter or Gaussian filter [2, 3]. A significant advance in denoising performance was introduced using transform domain methods based on analytic transformations such as discrete cosine transform (DCT) [4] and wavelet transform [5, 6]. Isotropic and anisotropic diffusion filters [7], total variation based regularization [8], Tikhonov regularization [9], Yaroslavsky filter which uses pixel neighborhood [10], bilateral filter [11] are some further examples for denoising methods. Most of the image denoising methods listed above can be separated into two broad groups. The first group includes global methods, which process the whole image at once. The second group would encompass local methods, which process the handled image using localized sections called as patches. The transform based thresholding methods [4–6] and the more recently proposed algorithm of [12] are examples for the global image denoising methods. The Yaroslavsky filter [10] and the bilateral filter [11] on the other hand, would form examples for local image denoising methods.

In recent years, in various image processing applications much interest has been directed towards methods which utilize patches extracted from the image [13–16]. The extracted image patches are of a much smaller size compared to the overall image. This idea enables us to benefit from the interrelation between the individual patches of an image. On the other hand, recently proposed nonlocal methods provide a rather new approach to image denoising problem. The main idea behind the nonlocal methods is the utilization of the similarity between patches which are placed in rather spatially distant locations of the image. Nonlocal methods for image denoising such as nonlocal means filter [14, 17], nonlocal principal component analysis [18, 19], and the recently proposed methods given in [15, 20] can outperform more traditional local approaches. A well-known nonlocal method is the nonlocal means filter which obtains denoised estimate of a pixel by weighted averaging of pixels within similar patches placed at different locations of the image. The most important step in these methods is the search for patches similar to a reference patch. While some methods realize this step by considering the whole image as the possible search area, other methods rather search in a restricted area of the image.

After the development of effective algorithms which use transform domain sparsity as a prior for signal representation tasks, the sparsity based image denoising methods have been also quite popular [21, 22]. There are various sparsity based methods which achieved very high quality results for image denoising. These include methods which learn a sparsifying synthesis or analysis dictionary for image denoising [23–29]. Transform learning provides an efficient platform for sparsifying analysis dictionary learning [28]. In [28], transform learning has been successfully applied to image denoising. [24] proposed a combination of transform learning and K-SVD frameworks. In [29] on the other hand, an improved transform learning algorithm which utilizes structured sparsity is developed. The resulting OCTOBOS (Overcomplete sparsifying transform with block cosparsity) algorithm exhibits very competitive image denoising results [29]. In [30], the multi-scale expected patch log-likelihood (MS-EPLL) method is developed by considering a multi-scale prior. The resulting MS-EPLL algorithm has been applied to image denoising among other image restoration applications. In [31], a joint sparse and group low-rank model is used to develop the STROLLR algorithm. STROLLR has been applied to image denoising and inpainting.

In [32], an enhanced sparse representation based denoising approach is obtained by grouping similar noisy image patches together. [32] utilizes the application of sparsifying transforms on patch groups formed from similar patches. In another recent method, [33] introduces an image denoising framework where noisy image pixels are ordered according to a distance measure, and a 1D smoothing filter is applied to these reordered pixels. [33] presents state-of-the-art denoising results by reordering the noisy image pixels to a 1D array and the application of pre-learned linear smoothing filters to this pixel array. This method achieves superior denoising performance compared to some well-known methods from the literature, such as NLM [14].

In this paper, we propose a novel solution to image denoising problem by reordering noisy image patches into a 3D patch cube and transforming this 3D array to a transformation domain. The

denoised image is generated by a thresholding process in the transform domain and inverse transformation back to the image domain. The main steps of the proposed approach can be summarized as follows.
1) For a given noisy image, we extract all the overlapping patches of size $\sqrt{n} \times \sqrt{n}$,
2) Reorder these patches into a 3D patch cube according to a similarity based distance measure,
3) Apply a suitable transform to the 3D patch cube,
4) Realize thresholding in the transform domain to keep only the significant transform coefficients which include most of the original signal energy
5) Apply the inverse transform and relocate the obtained patches to their original locations.

There are some important advantages to the proposed method when compared to similar approaches from the literature. The first advantage is the usage of 3D sparsifying transforms instead of the 1D learned filters which were proposed in the method of [33]. The utilization of 3D transformation framework provides better sparsification power and improved denoising performance. In addition, unlike the approach of [33], the proposed algorithm does not necessitate any filter learning step, which presents another advantage of the proposed approach compared to the method presented in [33]. Another advantage of the proposed algorithm is with respect to the BM3D method from [32]. The proposed algorithm forms a single reordered patch sequence by realizing a single ordering step at the start of the algorithm. On the other hand, the BM3D algorithm proposes the formation of patch groups for many patches extracted from the image, by using that particular patch as a reference patch. The utilization of a single ordering step over the whole corpus of patches provides algorithmic simplicity when compared to the repeated "similar patch search" process inherent to the BM3D algorithm.

The application of the triplet steps of "transform, threshold, inverse transform" provides a well-established method that has been used for image denoising [34]. However, this "heuristic shrinkage" approach [35] does not in general represent the solution to a strictly defined optimization problem. Recently, it has been shown that the shrinkage based denoising paradigm can be modeled as the solution to a rigorously defined optimization problem, when the utilized forward and inverse transforms satisfy the tight frame conditions [36]. In this work, we will also prove that the forward and inverse transformation matrices defined for our patch ordering based denoising method are indeed tight frames. Hence, the denoising method that we propose will form the solution to a rigorously defined optimization problem defined over the noisy image.

We present an original combination of patch ordering with patch based 3D transformations. We apply this novel procedure to image denoising. We also present a novel tight frame formulation for the forward analysis and backward synthesis transformations, which are defined using 3D transformations over the ordered patch cube. The developed image denoising algorithm is described comprehensively in the following sections. A comparison of this algorithm with some state-of-the-art image denoising methods is also given. The obtained results demonstrate that the proposed algorithm achieves competitive image denoising performance. The paper is organized as follows. Firstly, we describe the proposed image denoising method in detail in Section II. The subsections include the extraction of the noisy image patches, reordering of the extracted patches, forward transformation, threshold selection and thresholding, inverse transformation, obtaining the preliminary denoised image and the application of a Wiener filter to enhance the denoising performance. Some important implementation aspects of the algorithm are clarified in Section III. Experimental results for the proposed method and other competing methods are presented in Section IV. The conclusions are given in Section V.

2 A novel patch ordering based image denoising method: PO3D

We will develop a 3D shrinkage based image denoising algorithm by building upon the recently proposed patch ordering paradigm of [33]. In this algorithm we apply 3D transformations to the image patches after a specific ordering step. We will describe the corresponding forward and backward transformation operations as tight frames. We will be able to define the denoising problem as an optimization problem using this tight frame formulation. The specific steps which constitute the proposed image denoising algorithm are explained in the forthcoming sections.

2.1 The extraction and ordering of the image patches

In the image denoising problem, the image acquisition model can be formulated by the following linear equation:

$$\mathbf{Y} = \mathbf{X} + \mathbf{N} \tag{1}$$

Here, $\mathbf{Y} \in \mathbb{R}^{\sqrt{N} \times \sqrt{N}}$ indicates the noisy image, and $\mathbf{X} \in \mathbb{R}^{\sqrt{N} \times \sqrt{N}}$ is the original image. $\mathbf{N}$ is zero mean additive independent and identically distributed white Gaussian noise with variance σ^2. We can further define the vectorized image vector as $\boldsymbol{y} = vec(\mathbf{Y}) \in \mathbb{R}^N$, where $vec(\cdot)$ operator converts a matrix to its column stacked vector.

As the initial step of our proposed algorithm, we firstly extract possibly overlapped patches of size $\sqrt{n} \times \sqrt{n}$ from the noisy image. Here, $\sqrt{n}$ indicates edge size of a square patch. The overlap size can be selected arbitrarily. However, maximum overlaps in general lead to best denoising results. Let $\tilde{\mathbf{Z}}_i \in \mathbb{R}^{\sqrt{n} \times \sqrt{n}}$ define an image patch corresponding to ith pixel. The vectorized image patch can be written as $\tilde{\boldsymbol{z}}_i = vec(\tilde{\mathbf{Z}}_i)$ where $\tilde{\boldsymbol{z}}_i \in \mathbb{R}^n$ represents the vectorized form for $\tilde{\mathbf{Z}}_i$. The total patch vector $\tilde{\boldsymbol{z}}$ which is obtained from the concatenation of all vectorized patches $\tilde{\boldsymbol{z}}_i$ can be written as follows.

$$\tilde{\boldsymbol{z}} = \mathbf{K}\boldsymbol{y} \tag{2}$$

Here, $\mathbf{K} \in \mathbb{R}^{M \times N}$ indicates the patch vector generating matrix which generates the patch vector $\tilde{\boldsymbol{z}} \in \mathbb{R}^M$ from the vectorized image $\boldsymbol{y} \in \mathbb{R}^N$. $\mathbf{K}$ depends on the overlap used in generating the patches. The patch vector $\tilde{\boldsymbol{z}}$ can be described by the extraction of all patches of a certain overlap and sorting their vectorized versions serially. By assigning a patch vector to each pixel of the image (by allowing an extension of the image on the edges and by setting the overlap equal to unity), the number of patches will be equal to the number of pixels. In this case the size of the patch vector $\tilde{\boldsymbol{z}}$ is computed as $M = nN$ when the size of the rectangular patches is $\sqrt{n} \times \sqrt{n}$. Hence, $\mathbf{K}$ becomes a redundant permutation matrix with $\mathbf{K} \in \mathbb{R}^{M \times N}$.

At this stage we also want to describe our preferred method for the back conversion from the patch vector to the image vector. After the processing of the patch vector, a denoised patch vector $\hat{\boldsymbol{z}}$ will be obtained. The conversion from the patch vector to the image vector will be realized by the least square solution $\hat{\boldsymbol{y}} = \mathbf{K}^{\dagger}\hat{\boldsymbol{z}}$. Here $(\cdot)^{\dagger}$ indicates the pseudoinverse with $\mathbf{K}^{\dagger}\mathbf{K} = \mathbf{I}_N$. For the particular $\mathbf{K}$ matrix as described above $\mathbf{K}^{\dagger} = \mathbf{K}^T/n$, and therefore $\mathbf{K}^T\mathbf{K} = n\mathbf{I}_N$.

After the patch extraction process, the next step will be the reordering of these patches to form a 3D patch cube. Reordering process is realized by using a distance measure based on similarity. There are different methods to measure the similarity, and in this work the squared Euclidean distance is preferred as in [33]. Considering two patch vectors $\tilde{\boldsymbol{z}}_i$ and $\tilde{\boldsymbol{z}}_j$ which correspond to the vectorized versions of $\tilde{\mathbf{Z}}_i$ and $\tilde{\mathbf{Z}}_j$ patch matrices respectively, the squared Euclidean distance will be given as follows.

$$w(\tilde{\boldsymbol{z}}_i, \tilde{\boldsymbol{z}}_j) = \|\tilde{\boldsymbol{z}}_i - \tilde{\boldsymbol{z}}_j\|^2 \tag{3}$$

By using this distance measure of patches as the basis of a new ordering, the patches will be rearranged with respect to their similarity. This new rearrangement can be defined by a special permutation matrix $\mathbf{P}_{\boldsymbol{y}} \in \mathbb{R}^{M \times M}$, where $\mathbf{P}_{\boldsymbol{y}}$ has unit values occurring in diagonal blocks as described in [33]. $\mathbf{P}_{\boldsymbol{y}}$ matrix will include the patch ordering information pertaining to the particular $\boldsymbol{y}$, hence the

$(\cdot)_{\boldsymbol{y}}$ subscript. The reordered patch vector can now be denoted as $\boldsymbol{z} = \mathbf{P}_{\boldsymbol{y}}\tilde{\boldsymbol{z}} = \mathbf{P}_{\boldsymbol{y}}\mathbf{K}\boldsymbol{y}$. The patch extraction and reordering processes together are described in Fig. 1.

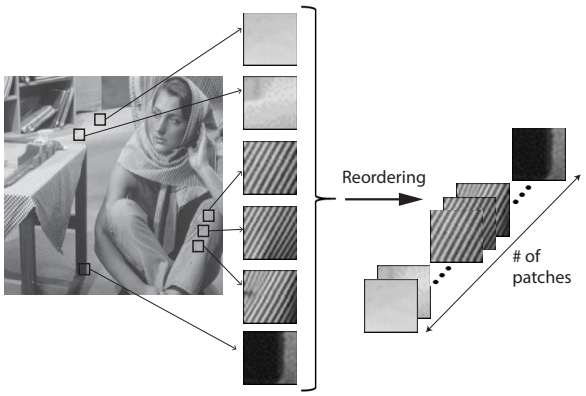

Fig. 1: Patch extraction and reordering of the extracted patches.

2.2 3D grouping of the extracted patches

The cube that is formed by stacking the ordered image patches serially will be called as the 3D patch cube. The transformations will be applied to the reordered 3D patch cube followed by a thresholding of the spectral coefficients of the 3D patch cube in the transform domain. As the next step, the thresholded spectral coefficients will be inverse transformed back to the patch domain. We will consider 3D transformations separable onto individual 2D and 1D transforms as is the case in [32]. The 2D transformation will utilize the intrapatch, local correlation of pixels in a single patch, whereas the 1D transform will utilize the nonlocal interpatch correlation between the spectral coefficients of similar patches. The 2D transformation will be applied on each patch, while the 1D transformation is implemented on the 1D fibers along the third dimension of the 2D patch spectrum stacks. The inverse transformation will also be handled in the same separable way as a succession of two transformations.

The method as devised here is different from the previous approaches given in the literature. In [32], similar image patch groups are found for most (if not all) patches of the image by considering them as the reference patch. In this work on the other hand, the extracted patches are arranged and reordered according to a similarity measure in a single ordering step, and a single ordered patch cube is created for further transformations. This is in contrast to the procedure in [32], where the similar patch group formation step is repeated for all the reference patches as described above. As a further note, in [33] the reordered patches are only utilized to form a smooth sequence of the actual pixel values. All pixels are ordered to a single 1D fiber according to the ordering learned from the patch similarities [33]. The further filtering process is realized in the spatial domain by filtering this 1D pixel fiber. However, in this work the "filtering" is applied on 3D patch structures in the transform domain. The utilization of both interpatch and intrapatch correlations by 3D transforms as described above leads to much better sparsification in the transform domain.

After the ordering is realized, the 3D patch cube is separated into groups with the same number of patches. The number of patches in all groups is fixed to a constant number K. The 3D patch cube is separated into R groups with K patches in each. Here, we consider the case where the obtained groups are disjoint (no patch overlaps) for the sake of simplicity . However, a sliding overlap might also considered for the formation of the patch groups. For this principle case of disjoint groups, the number of patches in a group multiplied by the number of groups gives the number of pixels of the image as $KR = N$. We will adopt the notation where $\boldsymbol{z}_{(i,j)}$ represents the jth patch of the ith patch group. Using this notation, we can define the total patch vector of the rth patch group as $\boldsymbol{z}_r = [\boldsymbol{z}_{(r,1)}^T, \boldsymbol{z}_{(r,2)}^T, \dots \boldsymbol{z}_{(r,K)}^T]^T$. The $\mathbf{Z}_r$ matrix containing the vectorized patches of a group in its columns can be given as $\mathbf{Z}_r = [\boldsymbol{z}_{(r,1)}\boldsymbol{z}_{(r,2)} \dots \boldsymbol{z}_{(r,K)}]$ with $\boldsymbol{z}_r = vec(\mathbf{Z}_r)$. The reordered total patch vector $\boldsymbol{z}$ can be restated as follows.

$$\boldsymbol{z} = [\boldsymbol{z}_1^T, \boldsymbol{z}_2^T, ..., \boldsymbol{z}_R^T]^T \tag{4}$$

The grouping of the image patches into R unique disjoint groups after reordering provides a data dependent new representation for the noisy image. This adaptivity to underlying observation, in this case the noisy image, will provide improved performance when compared to straightforward application of non-adaptive analytic transforms to data.

2.3 3D transformation of the grouped patches and thresholding process

After the grouping process is done, the 3D transformation process is handled by using a separable combination of 2D and 1D transforms as in [37]. Our presentation here is based on the notation as introduced in [37]. The vectorized form for the 2D transform of the patch vector $\boldsymbol{z}_{(r,j)} = vec(\mathbf{Z}_{(r,j)})$ can be realized using a transformation matrix $\mathcal{T}_2 \in \mathbb{C}^{n\times n}$.

$$\boldsymbol{\theta}_{(r,j)} = \mathcal{T}_2\boldsymbol{z}_{(r,j)} \tag{5}$$

Here, $\boldsymbol{\theta}_{(r,j)} \in \mathbb{C}^n$ indicates the vectorized 2D spectrum of the jth patch of the rth group. As a further note, if the 2D transformation is separable over 1D transforms (as is the case for 2D DFT), the 2D spectrum in matrix form can be obtained as $\mathbf{D}_1\mathbf{Z}_{(r,j)}\mathbf{D}_2^T$. Here $\mathbf{D}_1$ and $\mathbf{D}_2$ denote the constituent 1D transforms. As an example, for the case of 2D-DFT, $\mathbf{D}_1 = \mathbf{D}_2$ where $\mathbf{D}_1$ is the 1D-DFT matrix. If $\mathcal{T}_2$ is indeed separable, then $\mathcal{T}_2$ can be written as $\mathcal{T}_2 = \mathbf{D}_2 \otimes \mathbf{D}_1$, where "$\otimes$" denotes the Kronecker product. If both $\mathbf{D}_1$ and $\mathbf{D}_2$ are unitary transforms, then $\mathcal{T}_2$ will also be a unitary transform.

After obtaining the spectra of all patches in the patch group, a 1D transform $\mathcal{T}_1 \in \mathbb{C}^{K\times K}$ is applied to each fiber across the 3rd dimension of the group spectrum.

$$\boldsymbol{\Omega}_r = [\boldsymbol{\theta}_{(r,1)}\boldsymbol{\theta}_{(r,2)}...\boldsymbol{\theta}_{(r,K)}]\mathcal{T}_1^T = \boldsymbol{\Theta}_r\mathcal{T}_1^T \tag{6}$$

The 1D transform is applied to each row of the $\boldsymbol{\Theta}_r \in \mathbb{C}^{n\times K}$ matrix which contains in its columns the vectorized 2D spectrums of patches in the rth group. The application of the 1D transform to the fibers across this group spectrum benefits from the correlation between the 2D spectral coefficients of patches in the same group. If the columns of $\boldsymbol{\Omega}_r$ are stacked in a vector as in $\boldsymbol{\omega}_r = vec(\boldsymbol{\Omega}_r) \in \mathbb{C}^{nK}$, the 3D spectrum of a patch group can now be fully formalized as follows. Here, $\mathbf{Z}_r$ is the 2D spectrum matrix which contains in its columns all $\boldsymbol{z}_{(r,j)}$ spectral vectors for the rth group.

$$\begin{aligned} \boldsymbol{\Omega}_r &= \mathcal{T}_2\mathbf{Z}_r\mathcal{T}_1^T \\ vec(\boldsymbol{\Omega}_r) &= (\mathcal{T}_1 \otimes \mathcal{T}_2)vec(\mathbf{Z}_r) \\ \boldsymbol{\omega}_r &= \boldsymbol{\Upsilon}\boldsymbol{z}_r \end{aligned} \tag{7}$$

The transform matrix $\boldsymbol{\Upsilon} \in \mathbb{C}^{nK\times nK}$ transforms the patch vector of a group. Here we should note that, if the transforms $\mathcal{T}_1$ and $\mathcal{T}_2$ are unitary, $\boldsymbol{\Upsilon} = \mathcal{T}_1 \otimes \mathcal{T}_2$ is also unitary.

We can denote the overall 3D spectrum of the total patch vector $\tilde{\boldsymbol{z}}$ in a compact form as follows.

$$\boldsymbol{\omega} = (\boldsymbol{\Upsilon} \otimes \mathbf{I}_R)\mathbf{P}_{\boldsymbol{y}}\tilde{\boldsymbol{z}} \tag{8}$$

Here, $\boldsymbol{\omega} = [\boldsymbol{\omega}_1^T, \boldsymbol{\omega}_2^T, ..., \boldsymbol{\omega}_R^T]^T \in \mathbb{C}^M$ and $\mathbf{I}_R \in \mathbb{R}^{R\times R}$ indicates the identity matrix. By using this definition, the complete forward

analysis transform process starting from image vector and ending with the full spectral vector can be described as follows.

$$\boldsymbol{\omega} = (\boldsymbol{\Upsilon} \otimes \mathbf{I}_R)\mathbf{P}_{\boldsymbol{y}}\mathbf{K}\boldsymbol{y} = \boldsymbol{\Phi}_{\boldsymbol{y}}\boldsymbol{y} \tag{9}$$

Here, $\boldsymbol{\Phi}_{\boldsymbol{y}} \in \mathbb{C}^{M\times N}$ summarizes the full forward transform matrix specific to the initial noisy image $\boldsymbol{y}$. The forward transformation procedures as described above are visualized in Fig. 2. Denoising can be realized by thresholding the obtained 3D spectral coefficients $\boldsymbol{\omega}$. If we denote the thresholding operator with threshold λ as Γ_λ, the obtained coefficients after the thresholding operation can be written as $\hat{\boldsymbol{\omega}} = \Gamma_\lambda(\boldsymbol{\omega})$. The issue of threshold selection and the type of thresholding utilized are explained in detail in Section III. The results while using various different transforms for the constituent 1D and 2D transformations are also analyzed in detail in Section III.

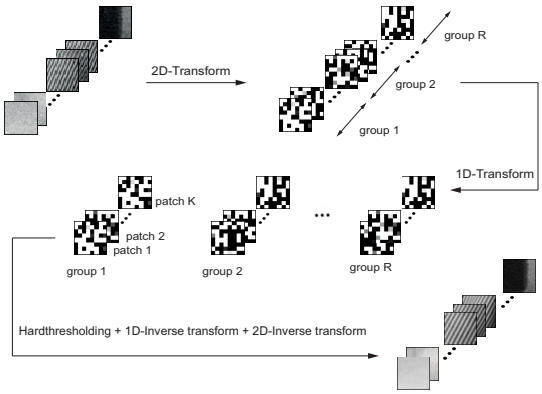

Fig. 2: Transformation and inverse transformation processes.

2.4 3D inverse transformation of thresholded coefficients

The inverse transformation matrix from the 3D spectral coefficients to the image domain can be obtained using procedures similar to the forward transform. The thresholded spectrum $\hat{\boldsymbol{\omega}}$ should be inverse transformed back to the image domain, where the denoised patch estimates will be relocated to their original locations. The inverse transformation steps start with the 1D inverse transform followed by the 2D inverse transform of the spectral coefficients. We firstly consider the 1D inverse transform starting from the thresholded spectral coefficients of group r, $\hat{\boldsymbol{\Omega}}_r = \Gamma_\lambda(\boldsymbol{\Omega}_r)$. Assuming unitary $\mathcal{T}_1$, the 1D inverse transformation can be written as follows.

$$\hat{\boldsymbol{\Theta}}_r = \hat{\boldsymbol{\Omega}}_r\mathcal{T}_1^* \tag{10}$$

The denoised image patches $\hat{\mathbf{Z}}_r$ are obtained after the inverse 2D transform is realized.

$$\hat{\mathbf{Z}}_r = \mathcal{T}_2^H\hat{\boldsymbol{\Theta}}_r = \mathcal{T}_2^H\hat{\boldsymbol{\Omega}}_r\mathcal{T}_1^* \tag{11}$$

In the above equation, we again assume unitary transformation $\mathcal{T}_2$, and this equation can be rewritten using the Kronecker product notation.

$$\begin{aligned} vec(\hat{\mathbf{Z}}_r) &= (\mathcal{T}_1^H \otimes \mathcal{T}_2^H)vec(\hat{\boldsymbol{\Omega}}_r) \\ \hat{\boldsymbol{z}}_r &= (\mathcal{T}_1 \otimes \mathcal{T}_2)^H\hat{\boldsymbol{\omega}}_r = \boldsymbol{\Upsilon}^H\hat{\boldsymbol{\omega}}_r \end{aligned} \tag{12}$$

Using the above results, the estimate for the denoised total patch vector $\hat{\boldsymbol{z}}$ can be obtained in a form similar to the forward transform case.

$$\hat{\boldsymbol{z}} = \mathbf{P}_{\boldsymbol{y}}^T(\boldsymbol{\Upsilon}^H \otimes \mathbf{I}_R)\hat{\boldsymbol{\omega}} \tag{13}$$

The denoised image estimate can be formed as follows.

$$\hat{\boldsymbol{y}} = \mathbf{K}^\dagger\mathbf{P}_{\boldsymbol{y}}^T(\boldsymbol{\Upsilon}^H \otimes \mathbf{I}_R)\hat{\boldsymbol{\omega}} = \boldsymbol{\Psi}_{\boldsymbol{y}}\hat{\boldsymbol{\omega}} \tag{14}$$

Here, $\boldsymbol{\Psi}_{\boldsymbol{y}} \in \mathbb{C}^{N\times M}$ indicates the overall inverse transformation matrix from the 3D spectral coefficients to the image domain. Here, we want to clarify an important relation between the analysis and synthesis transforms.

$$\boldsymbol{\Psi}_{\boldsymbol{y}}\boldsymbol{\Phi}_{\boldsymbol{y}} = \mathbf{K}^\dagger\mathbf{P}_{\boldsymbol{y}}^T(\boldsymbol{\Upsilon}^H \otimes \mathbf{I}_R)(\boldsymbol{\Upsilon} \otimes \mathbf{I}_R)\mathbf{P}_{\boldsymbol{y}}\mathbf{K} \tag{15}$$

The above multiplication produces an identity matrix. By evaluating (15) and by using (9) and (14), one can write the following equalities.

$$\begin{aligned} \boldsymbol{\Psi}_{\boldsymbol{y}}\boldsymbol{\Phi}_{\boldsymbol{y}} &= \mathbf{I}_N \\ \boldsymbol{\Psi}_{\boldsymbol{y}} &= \boldsymbol{\Phi}_{\boldsymbol{y}}^H/n \end{aligned} \tag{16}$$

As mentioned in the previous subsection, the obtained transforms $\boldsymbol{\Psi}_{\boldsymbol{y}}$ and $\boldsymbol{\Phi}_{\boldsymbol{y}}$ are data adaptive since they are formed using a particular patch ordering which depends on the given data (image).

2.5 Wiener filtering using the intermediary denoised image

As a final step, we enhance our denoised estimate $\hat{\boldsymbol{y}}$ by using a further Wiener filtering step [32]. We firstly calculate Wiener shrinkage coefficients by recalculating the 3D spectral coefficients for the intermediary denoised image $\hat{\boldsymbol{y}}$.

$$\mathbf{W}^{wie} = \frac{|\hat{\boldsymbol{\Phi}}\hat{\boldsymbol{y}}|^2}{|\hat{\boldsymbol{\Phi}}\hat{\boldsymbol{y}}|^2 + \sigma^2} \tag{17}$$

Here, $\hat{\boldsymbol{\Phi}} = \boldsymbol{\Phi}_{\hat{\boldsymbol{y}}}$ denotes the analysis operator corresponding to the intermediary denoised image $\hat{\boldsymbol{y}}$. The Wiener filter is realized by pointwise multiplication of the 3D spectral coefficients of the original noisy image $\boldsymbol{y}$ with the Wiener coefficients $\mathbf{W}^{wie}$. Afterwards, the Wiener filtered coefficients are converted back to the image domain. The Wiener filtering step is fully described by the following equation.

$$\hat{\boldsymbol{y}}^{wie} = \hat{\boldsymbol{\Psi}}(\mathbf{W}^{wie} \cdot (\hat{\boldsymbol{\Phi}}\boldsymbol{y})) \tag{18}$$

Here, $\hat{\boldsymbol{\Psi}} = \boldsymbol{\Psi}_{\hat{\boldsymbol{y}}}$ denotes the synthesis operator corresponding to the intermediary denoised image $\hat{\boldsymbol{y}}$. By the further addition of the Wiener filter, the denoising performance of the overall algorithm gets enhanced.

All the explained steps of the proposed denoising algorithm are summarized in Algorithm 1. We will be calling the proposed image denoising algorithm which uses patch ordering and 3D transformations as the PO3D method. In Fig. 3, the main steps of the proposed algorithm are depicted sequentially with block diagrams. The obtained intermediary denoised image, which corresponds to the output of the last block in Fig. 3, is used to calculate the Wiener shrinkage coefficients in the Wiener filtering step.

3 Implementation aspects for the proposed algorithm

3.1 Tight frame formulation for the forward transform

In the literature, the method that is composed of transforming a given signal to a transform domain, thresholding the obtained spectrum and then inverse transforming the thresholded coefficients is a well-established solution for denoising problems. This approach is known as "heuristic shrinkage" [35]. In a recent work in literature [36], it has been shown that if the used transforms are tight frames then the heuristic shrinkage can be formulated as the exact solution to a certain optimization problem. Let us consider the frame $\{\phi_i\} \in \mathbb{C}^N$,

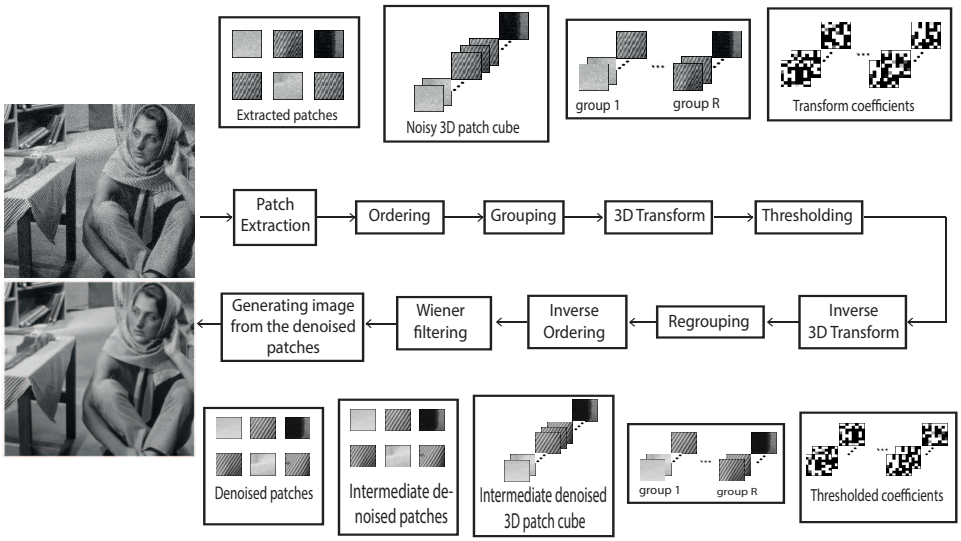

Fig. 3: Block diagram for the proposed PO3D image denoising method.

Algorithm 1 Image Denoising using Patch Ordering and 3D Transformations - PO3D

Input: The noisy image $\mathbf{Y}$.

1: Obtain the image and total patch vectors: $\boldsymbol{y} = vec(\mathbf{Y})$, $\tilde{\boldsymbol{z}} = \mathbf{K}\boldsymbol{y}$.
2: Reorder the patch vector: $\boldsymbol{z} = \mathbf{P}_{\boldsymbol{y}}\tilde{\boldsymbol{z}}$.
3: **for** $r := 1, 2, \ldots, R$ **do** ▷ for every group in the 3D patch cube.
4: **for** $j := 1, 2, \ldots, K$ **do** ▷ for every patch in a group.
5: Apply 2D transform to rth group: $\boldsymbol{\theta}_{(r,j)} = \mathcal{T}_2\boldsymbol{z}_{(r,j)}$.
6: **end for**
7: Obtain the 2D spectra: $\boldsymbol{\Theta}_r = [\boldsymbol{\theta}_{(r,1)}, \boldsymbol{\theta}_{(r,2)}, ..., \boldsymbol{\theta}_{(r,K)}]$.
8: **for** $i := 1, 2, \ldots, n$ **do** ▷ for every fiber in a group.
9: Apply 1D transform: $\boldsymbol{\Omega}_r = \boldsymbol{\Theta}_r\mathcal{T}_1^T$
10: **end for**
11: Form the vector: $vec(\boldsymbol{\Omega}_r) = \boldsymbol{\omega}_r = (\mathcal{T}_1 \otimes \mathcal{T}_2)\boldsymbol{z}_r$.
12: **end for**
13: Obtain overall 3D spectrum: $\boldsymbol{\omega} = [\boldsymbol{\omega}_1^T, \boldsymbol{\omega}_2^T, ..., \boldsymbol{\omega}_R^T]^T$.
14: Apply thresholding to the 3D spectrum: $\hat{\boldsymbol{\omega}} = \Gamma_\lambda(\boldsymbol{\omega})$.
15: **for** $r := 1, 2, \ldots, R$ **do** ▷ for every group.
16: **for** $i := 1, 2, \ldots, n$ **do** ▷ for every fiber in a group.
17: Considering $\hat{\boldsymbol{\omega}}_r = vec(\hat{\boldsymbol{\Omega}}_r)$, apply inverse 1D transform: $\hat{\boldsymbol{\Theta}}_r = \hat{\boldsymbol{\Omega}}_r\mathcal{T}_1^*$
18: **end for**
19: **for** $j := 1, 2, \ldots, K$ **do** ▷ for every patch in a group.
20: Apply inverse 2D transform: $\hat{\mathbf{Z}}_r = \mathcal{T}_2^H\hat{\boldsymbol{\Theta}}_r$.
21: **end for**
22: Form: $\hat{\boldsymbol{z}}_r = vec(\hat{\mathbf{Z}}_r) = (\mathcal{T}_1 \otimes \mathcal{T}_2)^H\hat{\boldsymbol{\omega}}_r$
23: **end for**
24: Obtain the intermediate denoised image vector: $\hat{\boldsymbol{y}} = \mathbf{P}_{\boldsymbol{y}}\mathbf{K}^\dagger\hat{\boldsymbol{z}}$.
25: Obtain Wiener shrinkage coefficients: $\mathbf{W}^{wie} = \frac{|\hat{\boldsymbol{\Phi}}\hat{\boldsymbol{y}}|^2}{|\hat{\boldsymbol{\Phi}}\hat{\boldsymbol{y}}|^2 + \hat{\sigma}^2}$.
26: Obtain final image vector estimate by Wiener filtering: $\boldsymbol{y}^{wie} = \hat{\boldsymbol{\Psi}}(\mathbf{W}^{wie} \cdot (\hat{\boldsymbol{\Phi}}\boldsymbol{y}))$

Output: The denoised image $\hat{\mathbf{Y}}^{wie}$.

where ϕ_i are the rows of $\boldsymbol{\Phi}$, where $\boldsymbol{\Phi}$ is the forward (analysis) transform matrix calculated by the PO3D method for a given image. We want to calculate the frame bounds for this frame. The following equation can be written by using (16).

$$\sum_i |\langle \phi_i, \boldsymbol{y}\rangle|^2 = \boldsymbol{y}^T\boldsymbol{\Phi}^H\boldsymbol{\Phi}\boldsymbol{y} = n\|\boldsymbol{y}\|_2 \tag{19}$$

This equation shows that rows of $\boldsymbol{\Phi}$ form a tight frame with frame constant n, where n is the patch size. Hence, the frame $\hat{\boldsymbol{\Phi}} = \boldsymbol{\Phi}/\sqrt{n}$ will be a Parseval frame, that is it is a tight frame with unit frame constant. Using the results from [36] and the fact that $\boldsymbol{\Phi}$ is a tight frame, we can come up with the following proposition.

Proposition: Assume $\boldsymbol{\Phi}$ is the forward analysis operator for an image vector $\boldsymbol{y}$ as defined by the PO3D algorithm. Consider the $\hat{\boldsymbol{\omega}} = \Gamma_\lambda(\hat{\boldsymbol{\Phi}}\boldsymbol{y})$ heuristic shrinkage operation as defined by the PO3D algorithm, where $\Gamma_\lambda(\cdot)$ denotes hard-thresholding by λ and $\hat{\boldsymbol{\Phi}} = \boldsymbol{\Phi}/\sqrt{n}$. The calculated spectral coefficients $\hat{\boldsymbol{\omega}}$ for the above operation are the actual solution to the following optimization problem.

$$\hat{\boldsymbol{\omega}} = \min_{\boldsymbol{\omega}} \|\boldsymbol{y} - \hat{\boldsymbol{\Phi}}^H\boldsymbol{\omega}\|_2^2 + \|(\mathbf{I} - \hat{\boldsymbol{\Phi}}\hat{\boldsymbol{\Phi}}^H)\boldsymbol{\omega}\|_2^2 + \lambda^2\|\boldsymbol{\omega}\|_0 \tag{20}$$

This result follows from the theoretical discussion as detailed in [36]. The optimization problem defined in (20) is known as the balanced sparse representation approach in the literature [38]. The denoised image corresponding to the solution $\hat{\boldsymbol{\omega}}$ for this optimization problem will be calculated as $\hat{\boldsymbol{y}} = \hat{\boldsymbol{\Phi}}^H\hat{\boldsymbol{\omega}}$.

3.2 Determining the threshold value

The determination of a good threshold value for thresholding based denoising methods is an important problem. In the literature there are different approaches for threshold selection. Some of the well-known methods include the VisuShrink method which applies the Universal threshold proposed by [39], the SureShrink method which attempts to minimize the Unbiased Risk Estimate as proposed in [40], and the BayesShrink method which uses the Bayesian framework as proposed by [41]. We employ the threshold selection method as proposed in [42], where the threshold is chosen as a value proportional to an estimate of the noise standard deviation. The utilized threshold selection method can be summarized with the following formula.

$$T = \beta\frac{\hat{\mu}}{\sqrt{2}\text{erf}^{-1}(\frac{1}{2})} \tag{21}$$

Here, T is the calculated threshold level. β is a suitably chosen constant, and erf(z) is the "error function". The parameter $\hat{\mu}$ defines the median absolute value of the spectral coefficients which are obtained from the 3D transformation step. As denoted in [42], β depends on the input SNR. Hence, the value of β is to be decided after several experiments for various input SNR values. These obtained β parameters are given in the Simulations section. By using these β values

different threshold levels are obtained from (21) for different input SNR values.

Similar to the threshold selection, there are various types of thresholding methods. Soft-thresholding and hard-thresholding are the most utilized methods. We employ the hard-thresholding approach in our work.

3.3 Selecting the utilized 1D and 2D subtransforms

One of the most important aspects of the proposed work is the choice for the utilized transforms. As previously mentioned, the 3D transformation task is handled in a separable manner via 2D and 1D transforms. The transformations we utilized in the proposed approach include, DCT, DST, Wavelet (Haar, Bior 1.5) transform and Fourier transform (DFT). Hence there are several combinations possible for the 2D and 1D transform pairs. We observe the results for these different combinations of transforms to decide which transform combinations to use for the initial denoising and Wiener filtering steps. Final choices for the transforms are obtained by optimizing the performance for the initial denoising and Wiener filtering steps separately. The experiments for this transform selection setup are realized for the "House" image and a noise standard deviation of $\sigma = 10$. We use the peak-signal-to-noise ratio (PSNR [dB]) as an image quality metric to measure the performance of the different transforms. This metric is defined by the following equation for a reference image $\mathbf{X}$ and a test image $\mathbf{Y}$, both of size $M \times N$ [43].

$$\mathrm{PSNR}(\mathbf{X},\mathbf{Y}) = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}(\mathbf{X},\mathbf{Y})}\right) \qquad (22)$$

Here, $\mathrm{MSE}(\mathbf{X},\mathbf{Y}) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(\mathbf{X}_{ij}-\mathbf{Y}_{ij})^2$. The maximum intensity for the images is assumed to be 255.

The output PSNR values for the use of different 2D and 1D transforms pairs in the initial thresholding-based denoising step are given in Table 1. These results are obtained using fixed 2D DCT and 1D Haar transforms in the Wiener filtering step. In this table, the first row shows the result of the for the initial thresholding-based denoising step, and the second row shows the result with the further application of the Wiener filtering step.

It can be seen from Table 1 that the best results are obtained when the 2D and 1D transforms are set to DST and DCT respectively. The transforms in the Wiener filtering step are updated by repeating the experiments using these transforms in the initial thresholding-based step. The results for these experiments are shown in Table 2. The results given in Table 2 indicate that the best results are obtained when the 2D and 1D transforms are set to the Bior 1.5 wavelet and Haar transforms, respectively. Thus, the transforms in Wiener filtering step are fixed as the 2D Bior 1.5 and 1D Haar transforms. Afterwards we have checked whether better performance can be achieved by using different transform combinations in the thresholding-based step. As can be inferred from Table 3, we get the best results when we reuse the exact same transform combination.

Table 1 PSNR (dB) results for the use of different transforms pairs in thresholding-based denoising step.

2D \ 1D	DCT	DST	DFT	Haar
DCT	36.26 36.58	36.22 36.51	35.24 36.12	36.28 36.56
DST	36.28 **36.60**	36.09 36.40	35.37 36.23	36.29 36.57
DFT	35.24 36.19	35.25 36.14	35.16 36.18	35.18 36.13
Bior1.5	36.24 36.56	36.19 36.48	35.19 36.00	36.29 36.55

Table 2 PSNR (dB) results for the use of different transforms pairs in Wiener filtering step.

2D \ 1D	DCT	DST	DFT	Haar
DCT	36.54	32.68	31.78	36.59
DST	33.37	29.22	28.75	33.35
DFT	31.51	28.54	28.38	31.51
Bior1.5	36.54	32.38	31.36	**36.63**

Table 3 PSNR (dB) values for the use of different transforms in thresholding step when Wiener step transforms are set to Haar and Bior1.5 wavelet.

2D \ 1D	DCT	DST	DFT	Haar
DCT	36.26 36.63	36.22 36.56	35.24 36.27	36.28 36.59
DST	36.28 **36.64**	36.09 36.42	35.37 36.35	36.29 36.59
DFT	35.24 36.31	35.25 36.25	35.16 36.28	35.18 36.21
Bior1.5	36.24 36.57	36.19 36.48	35.19 36.10	36.29 36.50

After several experiments, the 2D and 1D transforms for the thresholding-based denoising step are selected as 2D-DST and 1D-DCT respectively. The 2D and 1D transforms for the Wiener filtering step are selected as Bior 1.5 2D-wavelet transform and 1D-Haar transform, respectively.

4 Simulations

In this section the natural image denoising performance of the proposed algorithm is studied. We give the image denoising results of the proposed PO3D algorithm together with a comprehensive comparison to state-of-the-art methods including the nonlocal means (NLM) [14], K-SVD [23], OCTOBOS [29], the patch ordering based method (PO) of [33], and the recent denoising methods MS-EPLL [30] and STROLLR [31]. The implementations were performed in Matlab on a system with an Intel Core i7 CPU at 2.4GHz, 8 GB memory and 64-bit Windows 7 operating system. The denoising results are obtained by realizing all the algorithms for various different noise levels and different test images, including the Set12 dataset [44]. The Set12 dataset includes frequently used natural images as pictured in Fig. 4.

We utilize the PSNR and the structural similarity index measure (SSIM) as image quality metrics for performance comparison of the

Fig. 4: Images in the Set12 dataset.

Table 4 PSNR / SSIM results for different methods and images in the high SNR region.

σ / PSNR of noisy image	method	Lena	House	Hill	Lake	Boats	Clock	F.print	Avgs.
10/28.14	OCTOBOS	35.59 / 0.58	36.10 / 0.46	33.39 / 0.69	32.75 / 0.63	33.64 / 0.65	36.09 / 0.51	32.30 / 0.96	34.26 / 0.64
	PO	35.40 / 0.65	35.82 / 0.56	33.47 / 0.73	32.74 / 0.69	33.66 / 0.71	35.27 / 0.52	31.98 / 0.96	34.01 / 0.69
	KSVD	35.49 / 0.62	35.93 / 0.51	33.37 / 0.71	32.84 / 0.66	33.62 / 0.68	35.71 / 0.54	32.32 / 0.96	34.18 / 0.67
	NLM	35.01 / 0.58	35.48 / 0.46	32.19 / 0.62	31.37 / 0.59	32.50 / 0.61	34.59 / 0.52	31.03 / 0.95	33.17 / 0.62
	MS-EPLL	35.63 / 0.62	35.76 / 0.48	33.50 / 0.73	32.81 / 0.66	33.67 / 0.67	36.12 / 0.53	32.11 / 0.96	34.23 / 0.66
	STROLLR	35.93 / 0.64	36.61 / 0.57	33.59 / 0.74	32.89 / 0.70	33.81 / 0.70	35.81 / 0.53	32.47 / 0.96	34.44 / 0.69
	PO3D	35.86 / 0.64	36.64 / 0.56	33.64 / 0.73	32.77 / 0.65	33.84 / 0.68	36.15 / 0.55	32.47 / 0.96	**34.48** / 0.68
15/24.61	OCTOBOS	33.89 / 0.54	34.36 / 0.40	31.59 / 0.61	30.93 / 0.57	31.83 / 0.59	33.91 / 0.47	30.05 / 0.94	32.37 / 0.59
	PO	33.84 / 0.58	34.37 / 0.49	31.75 / 0.65	30.96 / 0.61	31.90 / 0.63	33.47 / 0.48	29.82 / 0.93	32.30 / 0.63
	KSVD	33.68 / 0.56	34.24 / 0.43	31.50 / 0.61	30.98 / 0.59	31.76 / 0.59	33.68 / 0.49	30.09 / 0.94	32.29 / 0.60
	NLM	33.16 / 0.53	34.03 / 0.41	30.29 / 0.52	29.89 / 0.53	30.55 / 0.52	32.95 / 0.47	28.91 / 0.91	31.40 / 0.56
	MS-EPLL	33.99 / 0.57	34.22 / 0.42	31.72 / 0.64	31.09 / 0.60	31.94 / 0.61	34.05 / 0.50	29.84 / 0.94	32.41 / 0.61
	STROLLR	34.25 / 0.58	34.77 / 0.50	31.79 / 0.66	31.03 / 0.61	31.97 / 0.63	33.67 / 0.50	30.27 / 0.94	32.54 / 0.63
	PO3D	34.18 / 0.58	34.94 / 0.46	31.85 / 0.65	31.02 / 0.59	32.06 / 0.62	34.07 / 0.51	30.24 / 0.94	**32.63** / 0.62
20/22.11	OCTOBOS	32.59 / 0.50	33.14 / 0.37	30.34 / 0.53	29.72 / 0.52	30.49 / 0.45	32.28 / 0.45	28.46 / 0.91	31.00 / 0.54
	PO	32.72 / 0.54	33.32 / 0.44	30.59 / 0.58	29.76 / 0.56	30.67 / 0.58	32.18 / 0.45	28.35 / 0.91	31.08 / 0.58
	KSVD	32.44 / 0.51	33.10 / 0.40	30.16 / 0.53	29.75 / 0.54	30.37 / 0.53	31.97 / 0.46	28.47 / 0.91	30.89 / 0.55
	NLM	31.64 / 0.48	32.59 / 0.38	28.92 / 0.45	28.54 / 0.48	29.07 / 0.46	31.32 / 0.43	27.12 / 0.86	29.88 / 0.51
	MS-EPLL	32.86 / 0.52	33.22 / 0.37	30.54 / 0.56	29.92 / 0.55	30.73 / 0.55	32.68 / 0.47	28.39 / 0.91	31.19 / 0.56
	STROLLR	33.06 / 0.55	33.49 / 0.44	30.61 / 0.59	29.79 / 0.56	30.66 / 0.57	32.16 / 0.47	28.76 / 0.93	31.22 / 0.59
	PO3D	32.94 / 0.54	33.80 / 0.41	30.69 / 0.59	29.83 / 0.55	30.81 / 0.57	32.59 / 0.48	28.78 / 0.92	**31.35** / 0.58
25/20.18	OCTOBOS	31.58 / 0.47	32.13 / 0.34	29.37 / 0.47	28.75 / 0.48	29.45 / 0.47	31.09 / 0.43	27.26 / 0.88	29.95 / 0.51
	PO	31.79 / 0.50	32.58 / 0.39	29.74 / 0.53	28.80 / 0.52	29.71 / 0.53	31.16 / 0.43	27.32 / 0.89	30.15 / 0.54
	KSVD	31.33 / 0.48	32.11 / 0.37	29.16 / 0.47	28.75 / 0.50	29.28 / 0.48	30.83 / 0.43	27.23 / 0.88	29.81 / 0.52
	NLM	30.44 / 0.44	31.18 / 0.35	27.91 / 0.40	27.38 / 0.44	27.91 / 0.41	29.65 / 0.39	25.71 / 0.81	28.60 / 0.46
	MS-EPLL	31.95 / 0.49	32.39 / 0.36	29.68 / 0.50	29.02 / 0.51	29.79 / 0.51	31.58 / 0.45	27.26 / 0.89	30.24 / 0.53
	STROLLR	31.99 / 0.51	32.46 / 0.40	29.71 / 0.54	28.90 / 0.53	29.61 / 0.53	31.05 / 0.44	27.63 / 0.90	30.19 / 0.55
	PO3D	31.96 / 0.51	32.94 / 0.37	29.78 / 0.54	28.91 / 0.51	29.81 / 0.53	31.41 / 0.46	27.66 / 0.89	**30.35** / 0.54

different methods. The definition of SSIM is given as follows, while the definition of PSNR was given in Section 3.3.

$$\text{SSIM}(\mathbf{X},\mathbf{Y}) = l(\mathbf{X},\mathbf{Y})c(\mathbf{X},\mathbf{Y})s(\mathbf{X},\mathbf{Y}) \tag{23}$$

Here, $l(\mathbf{X},\mathbf{Y})$, $c(\mathbf{X},\mathbf{Y})$ and $s(\mathbf{X},\mathbf{Y})$ indicate the luminance, contrast and structure comparisons, respectively. The detailed expressions for these two image quality metrics are available in [43].

The single input of the proposed algorithm is the noisy image. 512×512 sized noisy images are used to produce $N = 262{,}144$ patches, and 256×256 noisy images result in a total of $N = 65{,}536$ patches by using maximal overlap and periodic extension of the image. As explained with details in the previous sections, these patches are reordered according to a similarity measure, then the obtained 3D patch cube is transformed using a 3D transformation. The transforms which are used in the thresholding step and the Wiener filtering step are chosen as the 2D DST-1D DCT and 2D Bior 1.5 wavelet-1D Haar transform pairs, respectively. The β constant is chosen between the values 3 and 5.5 depending on the noise levels. It should be noted that the filters used in 1D filtering process of the patch ordering based method of [33] are pre-learned for different noise levels. Additionally, this method necessitates several parameters for the filter learning process. The parameter selection and filter learning processes for this method are computationally expensive. There is no pre-learned filter for $\sigma = 35$ level, hence the simulations of this method for $\sigma = 35$ have been realized by using the same parameters as listed for $\sigma = 25$.

Table 4 and Table 5 detail the image denoising results for the proposed PO3D algorithm and for the competing denoising algorithms from the literature. As seen from Table 4 and Table 5, the proposed PO3D algorithm achieves PSNR and SSIM performances competitive with the state-of-the-art algorithms. The proposed algorithm performs better than the NLM [14], K-SVD [23], OCTOBOS [29], PO [33], MS-EPLL [30] and STROLLR [31] algorithms on average. The proposed PO3D denoising scheme performs 1.8 dB better on average (over all images and noise levels) than the image denoising framework of NLM [14]. The results of PO3D are also 0.65 dB better on average than the dictionary learning based denoising with KSVD. When the results of OCTOBOS [29] framework are compared with the PO3D framework, it is seen that the PO3D algorithm provides 0.42 dB improvement on average. Finally, although the PO can achieve quite well performances especially for the high noise levels, the proposed PO3D algorithm performs on average 0.21 dB better than PO [33]. The proposed PO3D algorithm is about 0.1 dB better than the recently proposed MS-EPLL and STROLLR methods [30, 31]. Fig. 5 and Fig. 6 depict the noisy "house" and "boats" images together with the denoised images for the different methods at $\sigma = 10$. The results indicate that the proposed PO3D algorithm provides better denoising performance than the competing algorithms in most of the simulation setups.

We give in Table 6 the required computation times for all the compared methods. These are the runtimes needed to denoise a 256×256 image. For all the algorithms, we utilized their publicly available codes with default settings. As seen from the Table 6, the proposed method necessitates the least computational time amongst the included algorithms. Table 7 lists the denoising results for the Set12 dataset for $\sigma = 10, 25, 50$. These results are the averages over all the images in the dataset. Table 7 indicates that for the Set12 dataset, the PO3D algorithm performs on average better than all the other competing algorithms.

5 Conclusions

We have presented a new image denoising algorithm which proposes an original combination of the patch ordering scheme with potent 3D sparsifying transforms. The proposed algorithm simplifies the search process for similar patches by employing a single and efficient patch ordering step at the start of the algorithm. The performance of the proposed algorithm is improved by using effective 3D transformations on groups of patches, instead of simply filtering 1D fibers of pixels as suggested in the literature. Simulations with various natural test images show that the proposed algorithm perform better than the well-known denoising methods such as NLM and KSVD, while its performance is on a par with other state-of-the-art methods.

Table 5 PSNR / SSIM results for different methods and images in the low SNR region.

σ / PSNR of noisy image	method	Lena	House	Hill	Lake	Boat	Clock	F.print	Avgs.
35/17.25	OCTOBOS	29.99 / 0.42	30.46 / 0.31	28.01 / 0.38	27.29 / 0.43	27.90 / 0.40	29.27 / 0.40	25.50 / 0.83	28.35 / 0.45
	PO	30.14 / 0.44	30.93 / 0.35	28.32 / 0.46	27.47 / 0.47	28.21 / 0.47	29.60 / 0.38	25.51 / 0.84	28.60 / 0.49
	KSVD	29.70 / 0.43	30.37 / 0.32	27.72 / 0.39	27.24 / 0.44	27.64 / 0.41	29.10 / 0.38	25.44 / 0.82	28.17 / 0.46
	NLM	28.68 / 0.38	28.83 / 0.29	26.59 / 0.33	25.68 / 0.38	26.20 / 0.34	27.06 / 0.32	23.54 / 0.72	26.65 / 0.40
	MS-EPLL	30.53 / 0.45	31.06 / 0.33	28.44 / 0.43	27.64 / 0.46	28.35 / 0.44	29.91 / 0.42	25.59 / 0.84	28.79 / 0.48
	STROLLR	30.60 / 0.46	30.72 / 0.34	28.37 / 0.45	27.45 / 0.47	28.17 / 0.46	29.30 / 0.39	26.06 / 0.86	28.67 / 0.49
	PO3D	30.42 / 0.46	31.46 / 0.34	28.46 / 0.46	27.52 / 0.46	28.28 / 0.46	29.65 / 0.42	26.07 / 0.85	**28.84** / 0.49
50/14.16	OCTOBOS	28.34 / 0.36	28.61 / 0.27	26.65 / 0.30	25.69 / 0.36	26.30 / 0.32	27.03 / 0.34	23.67 / 0.74	26.61 / 0.39
	PO	28.96 / 0.40	29.44 / 0.31	26.99 / 0.36	25.89 / 0.40	26.67 / 0.38	27.90 / 0.36	24.13 / 0.79	27.14 / 0.43
	KSVD	27.89 / 0.37	28.10 / 0.27	26.25 / 0.32	25.44 / 0.37	25.96 / 0.33	26.81 / 0.31	23.27 / 0.71	26.24 / 0.38
	NLM	26.94 / 0.32	26.49 / 0.23	25.45 / 0.26	24.03 / 0.31	24.60 / 0.27	24.88 / 0.25	21.07 / 0.58	24.78 / 0.32
	MS-EPLL	28.99 / 0.39	29.45 / 0.30	27.15 / 0.35	26.13 / 0.40	26.80 / 0.37	28.15 / 0.38	23.84 / 0.77	**27.22** / 0.42
	STROLLR	28.86 / 0.40	29.01 / 0.31	26.98 / 0.37	25.95 / 0.40	26.58 / 0.38	27.20 / 0.33	24.47 / 0.81	27.01 / 0.43
	PO3D	28.78 / 0.40	29.67 / 0.30	27.08 / 0.38	25.98 / 0.40	26.64 / 0.37	27.69 / 0.37	24.44 / 0.80	27.18 / 0.43
75/10.63	OCTOBOS	26.38 / 0.30	26.46 / 0.22	25.23 / 0.24	23.87 / 0.29	24.51 / 0.25	24.56 / 0.26	21.37 / 0.61	24.62 / 0.31
	PO	27.22 / 0.33	27.34 / 0.26	25.44 / 0.28	24.13 / 0.32	25.02 / 0.30	25.84 / 0.30	22.45 / 0.71	25.35 / 0.36
	KSVD	25.74 / 0.30	25.34 / 0.20	24.87 / 0.26	23.46 / 0.30	23.98 / 0.25	24.26 / 0.23	19.89 / 0.52	23.93 / 0.29
	NLM	25.13 / 0.24	24.30 / 0.17	24.33 / 0.21	22.38 / 0.24	23.12 / 0.20	22.89 / 0.18	18.63 / 0.38	22.96 / 0.23
	MS-EPLL	27.24 / 0.33	27.48 / 0.25	25.73 / 0.27	24.39 / 0.32	25.11 / 0.28	25.99 / 0.31	21.77 / 0.65	**25.39** / 0.34
	STROLLR	26.96 / 0.32	26.88 / 0.24	25.44 / 0.28	24.25 / 0.33	24.81 / 0.29	25.29 / 0.27	22.70 / 0.72	25.19 / 0.35
	PO3D	26.92 / 0.33	27.32 / 0.25	25.60 / 0.29	24.22 / 0.32	24.79 / 0.27	25.22 / 0.29	22.66 / 0.71	25.25 / 0.35
Overall averages (for Tables 4 and 5)	OCTOBOS	31.19 / 0.45	31.61 / 0.34	29.22 / 0.46	28.43 / 0.47	29.16 / 0.45	30.60 / 0.41	26.94 / 0.84	29.59 / 0.49
	PO	31.44 / 0.49	31.97 / 0.40	29.47 / 0.51	28.53 / 0.51	29.40 / 0.51	30.77 / 0.42	27.08 / 0.86	29.80 / 0.53
	KSVD	30.89 / 0.47	31.31 / 0.36	29.01 / 0.47	28.35 / 0.49	28.94 / 0.47	30.34 / 0.41	26.67 / 0.82	29.36 / 0.50
	NLM	30.14 / 0.42	30.41 / 0.33	27.95 / 0.40	27.04 / 0.42	27.70 / 0.40	29.05 / 0.37	25.14 / 0.74	28.21 / 0.44
	MS-EPLL	31.59 / 0.48	31.94 / 0.36	29.54 / 0.50	28.71 / 0.50	29.48 / 0.49	31.21 / 0.44	26.97 / 0.85	29.92 / 0.52
	STROLLR	31.66 / 0.49	31.99 / 0.40	29.50 / 0.52	28.61 / 0.52	29.37 / 0.51	30.64 / 0.42	27.48 / 0.87	29.89 / 0.53
	PO3D	31.58 / 0.49	32.39 / 0.38	29.59 / 0.52	28.61 / 0.50	29.46 / 0.51	30.97 / 0.44	27.47 / 0.87	**30.01** / 0.53

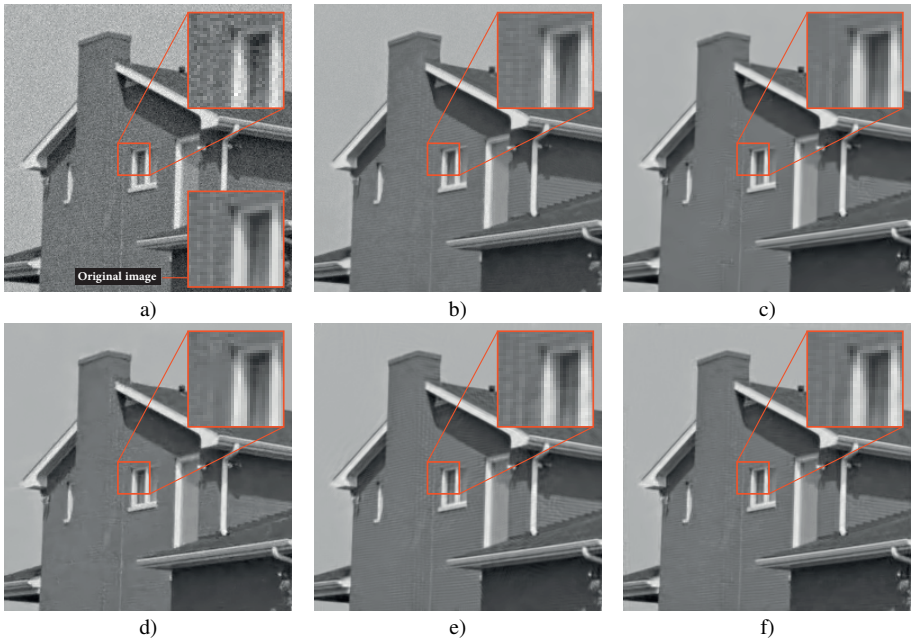

Fig. 5: Image denoising results for "house" image, $\sigma = 10$: a) Noisy image (28.14 dB) together with the zoomed section of the original image, b) Denoised image using PO (35.82 dB), c) Denoised image using OCTOBOS (36.10 dB), d) Denoised image using MS-EPLL (35.76 dB) e) Denoised image using STROLLR (36.61 dB) f) Denoised image using proposed algorithm (36.64 dB).

Acknowledgments

The authors thank TÜBİTAK (The Scientific and Technological Research Council of Turkey) for funding this research under project no. 217E076.

Table 6 Computation times (in seconds) for the competing algorithms for 256×256 image.

method	PO3D	OCTOBOS	PO	KSVD	NLM	MS-EPLL	STROLLR
Computation time (s)	**20.5**	143.6	359.5	198.6	72.1	139.4	76.3

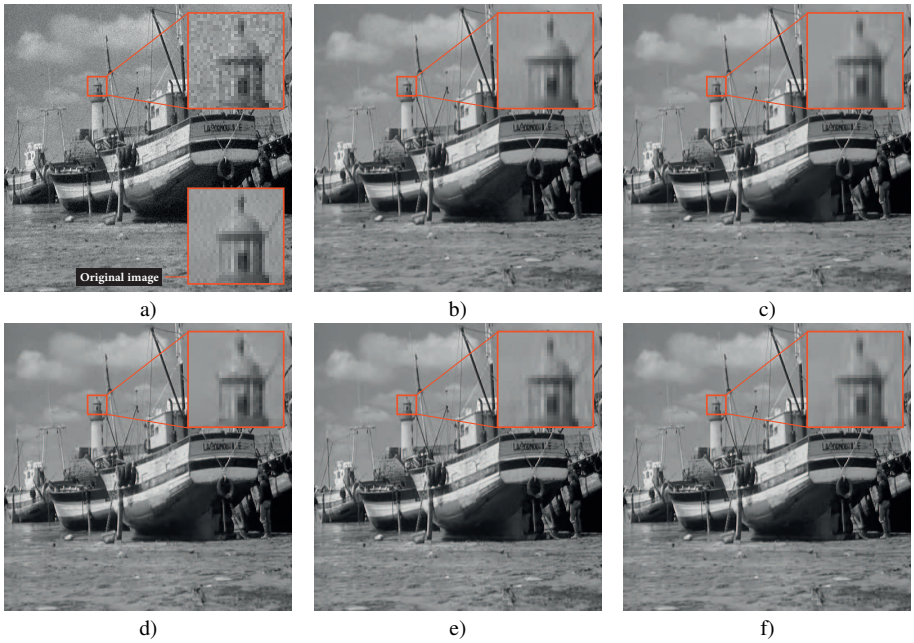

Fig. 6: Image denoising results for "boats" image, $\sigma = 10$: a) Noisy image (28.14 dB) together with the zoomed section of the original image, b) Denoised image using PO (33.66 dB), c) Denoised image using OCTOBOS (33.64 dB), d) Denoised image using MS-EPLL (33.67 dB) e) Denoised image using STROLLR (33.81 dB) f) Denoised image using proposed algorithm (33.84 dB).

Table 7 PSNR / SSIM results of different methods for the Set12 dataset.

method	$\sigma = 10$	$\sigma = 25$	$\sigma = 50$	Average
OCTOBOS	34.19 / 0.67	29.57 / 0.53	26.14 / 0.40	29.97 / 0.54
PO	33.82 / 0.71	29.67 / 0.56	26.61 / 0.44	30.03 / 0.57
KSVD	33.96 / 0.70	29.35 / 0.54	25.84 / 0.40	29.72 / 0.55
NLM	32.60 / 0.64	28.29 / 0.48	24.52 / 0.34	28.47 / 0.49
MS-EPLL	34.14 / 0.71	29.78 / 0.56	**26.59** / 0.43	30.17 / 0.57
STROLLR	34.18 / 0.72	29.66 / 0.57	26.46 / 0.44	30.10 / 0.57
PO3D	**34.29** / 0.71	**29.79** / 0.57	26.49 / 0.43	**30.19** / 0.57

6 References

1 X. L. Bahadir K. Gunturk, *Image Restoration Fundamentals and Advances*. CRC Press, 2017.
2 S. E. Umbaugh, *Computer Vision and Image Processing: A Practical Approach Using CVIPTools*. NJ, USA: Prentice Hall PTR, 1997.
3 I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters, Principles and Applications*. Boston: Kluwer Academic Publishers, 1990.
4 G. Yu and G. Sapiro, "DCT Image Denoising: a Simple and Effective Image Denoising Algorithm," *Image Processing On Line*, vol. 1, pp. 292–296, 2011.
5 S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1532–1546, Sep 2000.
6 M. A. T. F. Ivan W. Selesnick, "Signal restoration with overcomplete wavelet transforms: comparison of analysis and synthesis priors," *Proc.SPIE*, vol. 7446, pp. 7446 – 7446 – 15, 2009. [Online]. Available: http://dx.doi.org/10.1117/12.826663
7 P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, Jul 1990.
8 L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259 – 268, 1992.
9 A. N. Tikhonov and V. Y. Arsenin, "Solution of ill-posed problems," *Washington DC: VH. Winston*, 1977.
10 L. Yaroslavsky, *Digital Picture Processing : An Introduction*. Springer-Verlag, 1985.
11 C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Jan 1998, pp. 839–846.
12 G. Shikkenawis and S. K. Mitra, "2D orthogonal locality preserving projection for image denoising," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 262–273, Jan 2016.
13 C. Kervrann and J. Boulanger, "Optimal spatial adaptation for patch-based image denoising," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 2866–2878, Oct 2006.
14 A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 490–530, 2005. [Online]. Available: http://dx.doi.org/10.1137/040616024
15 J. Ji, F. Ren, H. Ji, Y. Yao, and G. Hou, "Generalised non-locally centralised image de-noising using sparse dictionary," *IET Image Processing*, vol. 12, no. 7, pp. 1072–1078, 2018.
16 M. Diwakar and M. Kumar, "CT image denoising using NLM and correlation-based wavelet packet thresholding," *IET Image Processing*, vol. 12, no. 5, pp. 708–715, 2018.
17 A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, June 2005, pp. 60–65 vol. 2.
18 D. D. Muresan and T. W. Parks, "Adaptive principal components and image denoising," in *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, vol. 1, Sept 2003, pp. I–101–4 vol.1.
19 L. Zhang, W. Dong, D. Zhang, and G. Shi, "Two-stage image denoising by principal component analysis with local pixel grouping," *Pattern Recogn.*, vol. 43, no. 4, pp. 1531–1549, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2009.09.023
20 M. P. Nguyen and S. Y. Chun, "Bounded self-weights estimation method for non-local means image denoising using minimax estimators," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1637–1649, April 2017.
21 M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010.
22 M. Aharon, "Overcomplete dictionaries for sparse representations of signals," Ph.D. dissertation, Israel Institute of Technology,Haifa, 2006.
23 M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, Dec 2006.
24 E. M. Eksioglu and O. Bayir, "K-SVD meets transform learning: Transform K-SVD," *IEEE Signal Processing Letters*, vol. 21, no. 3, pp. 347–351, March 2014.
25 R. Rubinstein, T. Peleg, and M. Elad, "Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model," *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 661–677, Feb 2013.

26 E. M. Eksioglu and O. Bayir, "Overcomplete sparsifying transform learning algorithm using a constrained least squares approach," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 7158–7162.

27 M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, "Constrained overcomplete analysis operator learning for cosparse signal modelling," *IEEE Transactions on Signal Processing*, vol. 61, no. 9, pp. 2341–2355, May 2013.

28 S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1072–1086, March 2013.

29 B. Wen, S. Ravishankar, and Y. Bresler, "Structured overcomplete sparsifying transform learning with convergence guarantees and applications," *International Journal of Computer Vision*, vol. 114, no. 2, pp. 137–167, Sep 2015. [Online]. Available: https://doi.org/10.1007/s11263-014-0761-1

30 V. Papyan and M. Elad, "Multi-scale patch-based image restoration," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 249–261, Jan 2016.

31 B. Wen, Y. Li, and Y. Bresler, "When sparsity meets low-rankness: Transform learning with non-local low-rank constraint for image restoration," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 2297–2301.

32 K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, Aug 2007.

33 I. Ram, M. Elad, and I. Cohen, "Image processing using smooth ordering of its patches," *IEEE Transactions on Image Processing*, vol. 22, no. 7, pp. 2764–2774, July 2013.

34 M. Jansen, *Noise Reduction by Wavelet Thresholding*. Springer-Verlag, 2001.

35 M. Elad, "Why simple shrinkage is still relevant for redundant representations?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5559–5569, Dec 2006.

36 J.-F. Cai, H. Ji, Z. Shen, and G.-B. Ye, "Data-driven tight frame construction and image denoising," *Applied and Computational Harmonic Analysis*, vol. 37, no. 1, pp. 89 – 105, 2014.

37 A. Danielyan, V. Katkovnik, and K. Egiazarian, "BM3D frames and variational image deblurring," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1715–1728, April 2012.

38 Y. Liu, J.-F. Cai, Z. Zhan, D. Guo, J. Ye, Z. Chen, and X. Qu, "Balanced sparse model for tight frames in compressed sensing magnetic resonance imaging," *PLOS ONE*, vol. 10, no. 4, pp. 1–19, 04 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0119584

39 D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994. [Online]. Available: http://dx.doi.org/10.1093/biomet/81.3.425

40 ——, "Adapting to unknown smoothness via wavelet shrinkage," *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1200–1224, 1995. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1995.10476626

41 S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1532–1546, Sep 2000.

42 K. R. Alyson K. Fletcher, Vivek K Goyal, "Iterative projective wavelet methods for denoising," *Proc.SPIE*, vol. 5207, 2003. [Online]. Available: https://doi.org/10.1117/12.507250

43 A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *20th International Conference on Pattern Recognition*, Aug 2010, pp. 2366–2369.

44 W. Zuo, K. Zhang, and L. Zhang, *Convolutional Neural Networks for Image Denoising and Restoration*. Cham: Springer International Publishing, 2018, pp. 93–123. [Online]. Available: https://doi.org/10.1007/978-3-319-96029-6_4