Image Denoising using the Higher Order Singular Value Decomposition

Ajit Rajwade, *Member, IEEE,* Anand Rangarajan, *Member, IEEE,* and Arunava Banerjee, *Member, IEEE*

Abstract—In this paper, we propose a very simple and elegant, patch-based, machine learning technique for image denoising using the higher order singular value decomposition (HOSVD). The technique simply groups together similar patches from a noisy image (with similarity defined by a statistically motivated criterion) into a 3D stack, computes the HOSVD coefficients of this stack, manipulates these coefficients by hard thresholding, inverts the HOSVD transform and performs hypotheses averaging at each pixel to produce the final filtered image. Our technique chooses all required parameters in a principled way, relating them to the noise model. We also discuss our motivation for adopting the HOSVD as an appropriate transform for image denoising. We experimentally demonstrate the excellent performance of the technique on grayscale as well as color images with our method producing state of the art results on the latter, outperforming other color image denoising algorithms at moderately high noise levels. A criterion for optimal patch-size selection and noise variance estimation from the residual images (after denoising), is also presented.

Index Terms—image denoising, singular value decomposition (SVD), higher order singular value decomposition (HOSVD), coefficient thresholding, learning orthonormal bases, patch similarity.

1 Introduction and Overall Description

Image denoising has a very rich history beginning in the mid-1970s. A plethora of different techniques have been proposed, some of which we will survey later. In recent times, transform-based techniques, especially in conjunction with machine learning, have gained popularity and success in terms of performance. In this paper, we propose a very simple, elegant and effective algorithm that contributes to the paradigm of learning a pointwise varying transform basis from the noisy image pixels by exploiting the non-local self-similarity of the image. A high level description follows. Given an image I_n, which is the degraded version of an underlying clean image I, our aim is to recover an estimate of I from I_n. We assume a zero mean i.i.d. Gaussian distribution of fixed, known standard deviation σ [i.e. $\mathcal{N}(0,\sigma)$] as the noise model. The steps involved in the denoising algorithm are as follows. (1) At each pixel and for a fixed patch size, a stack comprising similar image patches is constructed (using a similarity measure derived from the noise model, as described in Section 3.4). (2) Higher order singular value decomposition (HOSVD) bases (3D for grayscale and 4D for color) are extracted for each stack. (3) Each stack is projected onto the bases and coefficients with values below a hard threshold (determined using well known signal processing principles) are truncated to obtain a set of hypotheses (which are possible denoised values for each pixel). (4) The patches are reassembled in image space and the set of hypotheses at each pixel averaged to obtain a denoised image. The only free parameter is the patch size. This HOSVD-based image denoising algorithm achieves close to state of the art performance. In Figure 1, we demonstrate sample results of our method on a grayscale image corrupted by noise [drawn from $\mathcal{N}(0,20)$], and a color version of the same image under $\mathcal{N}(0,20)$ noise on the R,G,B channels.

• *The authors are with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611-6120, USA.*
E-mail: {avr,anand,arunava}@cise.ufl.edu

The HOSVD is a generalization of the matrix SVD to higher order matrices [11]. Some pioneering and successful applications of the HOSVD in computer vision have been proposed in [35]. In this paper, we demonstrate the aptness of the HOSVD as a transform basis for efficient and effective patch-based denoising. The main point we wish to emphasize is: *this simple approach yields performance comparable to techniques that are far more complex conceptually and in terms of implementation.* Note that our approach should not be confused with HOSVD-based denoising approaches such as [17], which are solely designed for hyperspectral images, and which treat the *entire* image as a single tensor thereby ignoring non-local patch similarity.

The rest of the paper is organized as follows. We first briefly survey the existing literature on denoising in Section 2. In Section 3, we describe the genesis of our main idea, propose an intermediate algorithm called the non-local SVD (NL-SVD), and explain why the HOSVD is an appropriate choice of transform for

Fig. 1. First three images: original grayscale image, noisy images from $\mathcal{N}(0,20)$ (PSNR 22.0), and HOSVD output (PSNR 27.47). Last three images: original grayscale image, noisy images from $\mathcal{N}(0,20)$ (PSNR 22.0), and HOSVD output (PSNR 30.464).

denoising. This is followed by extensive experimental results and comparisons with the generally more complex state of the art algorithms in Section 4. We also present a criterion for automated selection of patch-size and for estimating the noise standard deviation (in case it is unknown), by leveraging the statistical properties of the residual image (the difference between the noisy and denoised image). We conclude in Section 8.

2 Overview of Literature and Relation to Proposed Technique

The following is a rough categorization of the variety of image denoising techniques that have been developed so far: partial differential equations (PDEs), spatially varying convolution and regression, non-local techniques, transform-based techniques and methods based on machine learning. We begin this paper with a brief review of these categories.

PDE-based methods diffuse a noisy image in an anisotropic manner that extracts and respects the edge geometry, allowing diffusion along but not across the image edges [23], [37]. Some PDEs are obtained from the Euler-Lagrange equations corresponding to functionals that are based on a piecewise constant [28] or piecewise linear [40] model for natural images. In practice, the energy functionals are augmented with prior terms that penalize the error between the noisy and filtered image, as per the assumed noise model [28]. A rich class of techniques for image filtering involve the so-called **spatially varying convolutions**. In these methods, an image is convolved with a pointwise-varying, local geometry-driven mask [34]. A closely related idea is the local modeling of an image with a low-order polynomial function whose coefficients are computed by a weighted least-squares regression, and these are then used to compute the value of the (filtered) image at a central point. These weights are chosen based on estimates of local geometry [32] or the difference in the intensity/spatial coordinate values between neighboring pixels and the one to be filtered [5], [33], [8]. The most recent advancement in the area of local convolutions is the work in [31] which preserves corners and junctions in addition to edges, using Gabor filter responses at several orientations in conjunction with an innovative mixture model.

Transform-domain denoising approaches typically work at the level of small image patches. In these approaches, the image patch is projected onto an orthonormal basis, such as a wavelet [7] or DCT [38] to yield a set of coefficients, which for natural images, are known to be sparse and decorrelated [15]. The smaller coefficients usually correspond to the higher frequency components of the signal which are often dominated by noise. To perform denoising, the smaller coefficients are modified (typically, by 'hard thresholding' [16]), and the patch is reconstructed by inversion of the transform. This procedure is repeated for every patch. If the patches are chosen to be non-overlapping, one can observe seam artifacts at the patch boundaries and ringing artifacts around image edges or salient features, which can be attenuated by performing the aforementioned three steps in a sliding window fashion and averaging the multiple hypotheses, yielding superior results [7], [38]. Hard thresholding also corresponds to the MAP estimate of the wavelet coefficients under the assumption that the original wavelet coefficients of clean natural images have very sparse distributions [30], [15]. There exist several more sophisticated methods to manipulate wavelet coefficients, such as those that exploit dependencies in transform coefficients at the same spatial location but at different scales (e.g. in [29], or the BLS-GSM method in [25]), or at adjacent spatial locations [30].

Non-local techniques [4], [42], exploit the fact that natural images often contain patches in distant regions that are very similar to each other. NL-Means obtains a denoised image by minimizing a penalty term on the average weighted distance between an image patch and all other patches in the image, where the weights are decreasing functions of the squared difference between the intensity values in the patches. This yields an update rule that can be interpreted as a spatially varying convolution with non-locally derived masks. NL-Means can also be interpreted as a minimizer of the conditional entropy of a central pixel value given the intensity values in its neighborhood [3], [24].

A **combination of non-local and transform-domain approaches** has led to the development of the BM3D (block matching in three dimensions) method [9] which is considered the current state of the art in image denoising. This method operates at the patch level and for each reference patch in the image, it collects a group of similar patches (after a DCT-based pre-filtering step), which are then stacked together to form a 3D array. The entire 3D array is projected onto a 3D transform basis (product of DCT/biorthogonal and Haar bases) to yield a set of coefficients which are hard-thresholded. The filtered patches are then reconstructed by inversion of the transform. This process is repeated over the entire image in a sliding window fashion with averaging of hypotheses to yield an intermediate image. This image is then smoothed (heuristically) with a non-local empirical Wiener filter to produce a final filtered image. In this paper, we refer to these two output stages as 'BM3D1' and 'BM3D2' respectively. The authors claim that the group of patches exhibit greater sparsity collectively than each individual patch in the group, citing that as the reason for the state of the art performance of the BM3D method. The results using the BM3D method are outstanding. However the method is complex with several tunable parameters such as choice of bases, patch-size, transform thresholds, similarity measures, etc.

In the transform domain methods such as [9], [38], a fixed transform basis is chosen for signal representation. There exist methods e.g. [15], based on **learning the transform basis** from the statistics of image features or patches. There has been recent interest in learning overcomplete bases (also called dictionaries) [22], [18], whose inherent redundancy leads to sparser representation of natural signals. In the popular KSVD algorithm [1], an overcomplete dictionary as well as sparse representations of the patches in that dictionary are learned in an alternating minimization framework, starting from the overlapping patches from a noisy image, using column-wise SVD updates. A multi-scale variant of this algorithm (known as MS-KSVD) learns dictionaries to represent patches at two or more scales leading to further redundancy [20]. This algorithm has yielded state of the art performance, on par with the BM3D algorithm [20]. As against learning a single overcomplete dictionary for the entire image, the authors of the KLLD (K locally learned dictionaries) approach [6] perform a clustering step using K-Means on (coarsely pre-filtered) patches from the noisy image and then filter the patches from each cluster separately by projecting them onto lower-rank bases (learned by PCA) coupled with a kernel regression framework from [32]. The entire procedure is iterated for better performance. There also exist works such as [21] and [43] which infer spatially-varying orthonormal bases at each pixel using PCA.

3 Genesis of the Idea

In this section, we visit several variants of the patch SVD for image denoising, propose an intermediate ensemble SVD algorithm, motivate the idea of HOSVD as an appropriate transform for image denoising, and then describe the HOSVD algorithm in detail.

3.1 Matrix SVD for Image Denoising

Given a matrix A of size $m_1 \times m_2$, there exists a factorization of the form $A = USV^T$, where U is a $m_1 \times m_1$ orthonormal matrix, S is a $m_1 \times m_2$ diagonal matrix of positive 'singular' values and V is a $m_2 \times m_2$ orthonormal matrix. The columns of V and the columns of U (respectively called the right and left singular vectors) are respectively the eigenvectors of the column-column correlation matrix A^TA and the row-row correlation matrix AA^T. The singular values in S are the square roots of the eigenvalues of A^TA (or AA^T). The SVD also gives us the optimal low-rank decomposition of A, i.e. the optimal solution to $E(\tilde{A}) = \|A - \tilde{A}\|^2$ subject to the constraint rank$(\tilde{A}) = k,\ k < m_1, k < m_2$ is given by $\tilde{A} = \|U_k \tilde{S} V_k^T\|^2$ where U_k and V_k are the first k columns of U and V respectively and $\tilde{S}$ contains the k largest singular values of S. The singular values of natural images tend to decay exponentially and the SVD bases have a frequency interpretation [26], [2]. Given a noisy image A (a degraded version of an underlying clean image A_c) affected by noise from $\mathcal{N}(0, \sigma)$, filtering is accomplished in three steps: (1) computing the decomposition of small patches $A_i = U_i S_i V_i^T$ of size $p \times p$ in sliding window fashion, (2) manipulating the singular values $\{S_i\}$, and (3) averaging the hypotheses appearing at each pixel to produce a final filtered image. An example of this procedure is illustrated in Figure 2 for different methods of manipulating the singular values: (1) rank 1 or rank 2 truncation of each patch, (2) hard thresholding, i.e. nullification of patch singular values below a fixed threshold, in this case chosen to be $\sigma\sqrt{2 \log p^2}$ [which, as shown in [13]), is the optimal threshold from a statistical risk viewpoint, for any orthonormal basis, and for $\mathcal{N}(0, \sigma)$], and (3) truncation of singular values in such a way that the residual at each patch has a standard deviation of σ.

3.2 Oracle Denoiser with the SVD

Figure 2 shows the poor performance of the aforementioned approach. Basically, the singular vectors of a noisy patch are unable to adequately separate signal from noise. There are two key observations we make here. Firstly, let Q and Q_n be corresponding patches from A_c and A_n respectively. Given the decomposition $Q_n = U_n S_n V_n^T$, the projection of Q (the true patch) onto the bases (U_n, V_n) is given as $S_Q = U_n^T Q V_n$. This matrix S_Q is non-diagonal and hence contains more non-zero elements than S_n. Despite this, if we could somehow change the entries in S_n to match those in S_Q, we would have a superior denoising technique. Secondly, the additive noise doesn't affect just the singular values of the patch but the singular vectors as well. Keeping this in mind, it is strange that SVD-based denoising techniques do not seek to manipulate the orthonormal bases and instead focus only on further sparsifying the singular values. We now perform the following experiment which starts with a noisy image and assumes that the true singular vectors of the clean patch underlying every noisy patch in the image are provided to us by an oracle. Let the SVD for patch Q (from A_c) be $Q = USV^T$. We project the noisy patch Q_n onto (U, V) to produce a matrix $S_{Q_n} = U^T Q_n V$, following a sliding window

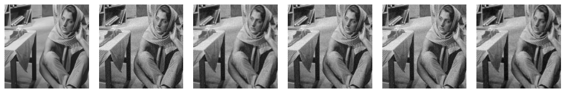


Fig. 2. Patch-based SVD filtering on the Barbara image: (left to right) clean Barbara image, noisy image with Gaussian noise of $\sigma = 20$ (PSNR = 22.11), filtered image with rank 1 truncation in each patch (PSNR = 23.9), filtered image with rank 2 truncation in each patch (PSNR = 25.05), filtered image with nullification of singular values below 3σ in each patch (PSNR = 23.42), filtered image with truncation of singular values in each patch so as to match noise variance (PSNR = 25.8). Zoom into pdf file for a detailed view.

Fig. 3. Oracle filter with SVD: (left to right) clean Barbara image, noisy image with Gaussian noise of $\sigma = 20$ (PSNR = 22.11), filtered image with hard threshold 3σ in each patch (PSNR = 36.9), noisy image with Gaussian noise of $\sigma = 40$ (PSNR = 22.11), filtered image with hard threshold 3σ in each patch (PSNR = 31.34). Zoom into pdf file for a detailed view.

approach with a hard threshold $\sigma\sqrt{2\log p^2}$ (as per [13]) on S_{Q_n}, and averaging of multiple hypotheses. This method is called the 'oracle denoiser'. Sample experimental results with the above technique are shown in Figure 3 for two noise levels: 20 and 40. The resulting PSNR values of this ideal denoiser far exceed the state of the art methods (see Tables 1 and 2). Clearly, this experiment is not possible in practice, however, it serves as a benchmark and chalks out a path for us to explore: manipulation of the SVD bases of a noisy patch, or somehow using bases that are 'better' than the SVD bases of the noisy patch may be the key to improving denoising performance.

3.3 Nonlocal SVD with Ensembles of Similar Patches

We now explore a non-local generalization of the SVD. Given a patch P from the noisy image, we look for other patches in the image that are 'similar' to P, where the notion of similarity is defined in Section 3.4. Assume that there are K such similar patches (including P) which we label as $\{P_i\}$ where $1 \leq i \leq K$. Next, we ask what *single* pair of orthonormal matrices U_k and V_k provide the best rank-k approximation to all the patches $\{P_i\}$. In other words, what orthonormal pair (U_k, V_k) minimizes the following energy?

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

where $\forall i, S_i^{(k)} \in R^{k\times k}$. The solution is given by an iterative minimization (starting from random initial conditions) presented in [26], [39]. Note that the matrices $\{S_i^{(k)}\}$ in this case are *not* diagonal and the bases (U_k, V_k) do not correspond to the individual SVD bases but to a basis pair that is common to all the chosen patches. An approximate closed-form solution to minimize the energy function in Equation 1 with associated error bounds was presented in [12]. The solution is given by the first k eigenvectors of the ensemble row-row and column-column correlation matrices $C_r = \sum_{i=1}^{K} P_i P_i^T$ and $C_c = \sum_{i=1}^{K} P_i^T P_i$ respectively (corresponding to the k largest eigenvalues). We call the bases thus derived the 'NL-SVD' bases, and have observed that they are related to the discrete cosine transform (DCT) bases (please see the *supplemental material* accompanying this paper).

We use this framework in a denoising algorithm and we shall show later that this produces excellent denoising results. We divide the given noisy image into patches. For each 'reference' patch P, we collect patches similar to it and obtain the *full rank* NL-SVD bases U and V (to avoid having to select the 'optimal' rank k, which need not be the same for every patch). Next, we project each P onto (U, V) producing the coefficient matrix $S^{(P)} = U^T P V$, nullify the coefficients with values smaller than the threshold $\sigma\sqrt{2\log p^2}$ (from [13]), followed by transform inversion and hypotheses averaging to produce the filtered image.

3.4 Choice of Patch Similarity Measure

Given a reference patch P_{ref} in a noisy image, we can compute its K nearest neighbors from the image, but this requires a choice of K which may not be the same for every image patch. Hence, we use a distance threshold τ_d and select all patches P_i such that $\|P_{\text{ref}} - P_i\|^2 < \tau_d$. Assuming a fixed, known noise model - $\mathcal{N}(0, \sigma)$, if P_{ref} and P_i were different noisy versions of the same underlying patch, the following random variable would have a $\chi^2(n^2)$ distribution:

$$x = \sum_{k=1}^{n^2} \frac{(P_{\text{ref},k} - P_{ik})^2}{2\sigma^2}. \qquad (2)$$

The cumulative of a $\chi^2(z)$ random variable is given by $F(x; z) = \gamma(\frac{x}{2}, \frac{z}{2})$ where $\gamma(x, a)$ stands for the incomplete Gamma function defined as $\gamma(x, a) = \frac{1}{\Gamma(a)} \int_{t=0}^{x} e^{-t} t^{a-1} dt$, with $\Gamma(a) = \int_0^\infty e^{-t} t^{(a-1)} dt$ being the Gamma function. We observe that if $z \geq 3$, for any $x \geq 3z$, we have $F(x; z) \geq 0.99$. Therefore, for a patch-size of $n \times n$ and for the given σ, we choose $\tau_d = 6\sigma^2 n^2$. Thus, *if* two patches are noisy versions of the same clean patch, this threshold will consider them to be similar with a very high probability. But, the converse is not true, and therefore, we may collect patch pairs that satisfy the threshold but are quite different structurally (please see *supplemental material* for an example). This motivates us to use a hypothesis test (the one-sided Kolmogorov-Smirnov (K-S) test), in the NL-SVD algorithm. To avoid having to choose a fixed significance level, we use the p-values output by the K-S tests as a weighting factor in the computation of the correlation matrices, rewriting them as follows:

$$C_r = \sum_{i=1}^{K} p_{KS}(P_{\text{ref}}, P_i) P_i P_i^T \tag{3}$$

$$C_c = \sum_{i=1}^{K} p_{KS}(P_{\text{ref}}, P_i) P_i^T P_i \tag{4}$$

with $p_{KS}(P_{\text{ref}}, P_i)$ being the p-value for the K-S test to check how well the values in $P_{\text{ref}} - P_i$ conform to $\mathcal{N}(0, \sqrt{2}\sigma)$. Thus, this gives us a robust version of the 2D-SVD. In practice, we observed that $p_{KS}(P_{\text{ref}}, P_i)$ was usually very close to zero if $\|P_{\text{ref}} - P_i\|^2 \geq 3\sigma^2 n^2$. Hence, we used the less conservative bound $\tau_d = 3\sigma^2 n^2$ in our experiments, which led to some improvement in computational speed. We also implemented a variant of our method in which the hypothesis test was entirely ignored and equal weights were used for all patches. Surprisingly, this did not lead to decrease in denoising performance. Nonetheless, we still used the hypothesis test because it is a *principled* way of mitigating the effect of false positives. Note that the K-S test criterion was used only in the NL-SVD algorithm, and not in the HOSVD algorithm from Sections 1 and 3.5.

3.5 Motivation for the Higher Order Singular Value Decomposition

In the NL-SVD algorithm from Section 3.3, consider a reference patch P_{ref} in the noisy image and let its underlying clean patch be Q_{ref}. Now, assume a scenario where all the K patches $\{P_i\}$ (in Section 3.3) were noise-corrupted versions of Q_{ref}. In such a case, we observe that $\lim_{K\to\infty} \sum_{i=1}^{K} P_i P_i^T = Q_{\text{ref}} Q_{\text{ref}}^T + \sigma^2 I$. Consequently, for large K, we have a reasonable chance of estimating the SVD bases of Q_{ref} and thus approach the oracle estimator from Section 3.2. However, such a situation is not possible in most natural images, and the patches that qualify as similar will usually not be exact copies of Q_{ref} modulo noise. Hence, we adopt the following principle: if a set of patches are similar to one another in the noisy image, denoising should take this fact into account and not denoise them independently. With this is in mind, we group together similar patches and represent them in the form of a 3D stack (see Equation 5 below). The main idea is that filtering is performed not only across the length and breadth of each individual (2D) patch, but also in the third dimension so as to allow for similarity between intensity values at corresponding pixels of the different patches. The idea of joint filtering of multiple patches has been implemented earlier in the BM3D algorithm [9] (see also Section 2), but with *fixed* bases. However, in this paper, we use this idea to *learn* spatially adaptive bases. An example in Figure 4 illustrates the superiority of our HOSVD approach over NL-SVD, for denoising a portion of the Barbara image (the tablecloth texture) from Figure 5(a). The third and fourth row of Figure 4 show the application of coefficient thresholding for smoothing of the 11 structurally similar patches of size 64×64 using the NL-SVD and HOSVD transforms respectively, while the last two rows show the filtered patches after the averaging operations (employing the same criteria for patch similarity and coefficient thresholding). These figures reveal that HOSVD preserves the finer textures on the tablecloth surface much better than NL-SVD which almost erases these textures. We have also experimentally confirmed the importance of building the stack from *similar* patches: randomly created stacks produce transforms that yield blurry and grainy images (please see *supplemental material*).

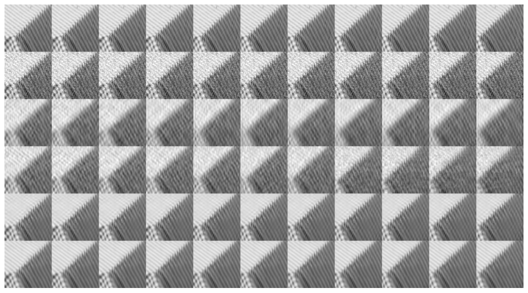


Fig. 4. Eleven patches of size 64×64 from the original Barbara image (row 1), its noisy version under $\mathcal{N}(0, 20)$ (row 2), from the NL-SVD output before averaging (row 3), from the HOSVD output before averaging (row 4), from the NL-SVD output after averaging (row 5), from the HOSVD output after averaging (row 6). Zoom into pdf file for a better view.

3.6 Implementation of the HOSVD for denoising

Given a $p \times p$ reference patch P_{ref} in the noisy image I_n, we create a stack of $K - 1$ similar patches. Here, similarity is defined as in Section 3.4, and hence K varies from pixel to pixel. Let us denote the stack as $\mathcal{Z} \in R^{p\times p\times K}$. The HOSVD of this stack is as follows [11]:

$$\mathcal{Z} = S \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)} \tag{5}$$

where $U^{(1)} \in R^{p\times p}$, $U^{(2)} \in R^{p\times p}$ and $U^{(3)} \in R^{K\times K}$ are orthonormal matrices, and S is a 3D coefficient array of size $p\times p\times K$. Here, the symbol $\times_n$ stands for the n^{th} mode tensor product defined in [11]. The orthonormal matrices $U^{(1)}$, $U^{(2)}$ and $U^{(3)}$ are in practice computed from the SVD of the unfoldings $\mathcal{Z}_{(1)}$, $\mathcal{Z}_{(2)}$ and $\mathcal{Z}_{(3)}$ respectively [11]. The exact equations are of the form $\mathcal{Z}_{(k)} = U^{(k)} \cdot S_{(k)} \cdot (U^{\text{mod}(k+1,3)} \otimes U^{\text{mod}(k+2,3)})^T$, where $1 \leq k \leq 3$ (which are equivalent representations for the HOSVD). However, the complexity of the

SVD computations for $K \times K$ matrices is $\mathcal{O}(K^3)$. For computational speed, we impose the constraint that $K \leq 30$[1]. The patches from $\mathcal{Z}$ are then projected onto the HOSVD transform domain. The parameter for thresholding the transform coefficients is picked to be $\sigma\sqrt{2\log p^2 K}$, again as per the rule from [13]. The stack $\mathcal{Z}$ is then reconstructed after inverting the transform, thereby filtering all the individual patches. The procedure is repeated over all pixels in sliding window fashion with averaging of hypotheses. Note that unlike NL-SVD (see Section 3.3), we filter *all* the individual patches in the ensemble and not just the reference patch. This affords additional smoothing on all the patches which was required due to the upper limit of $K \leq 30$ unlike the case with NL-SVD.

We also augment the HOSVD denoising with a Wiener filter step. Let $\hat{\mathcal{Z}}$ be a stack of similar patches from the HOSVD filtered image (using the same statistical criterion for similarity as before), and $\mathcal{Z}_n$ be the corresponding stack from the noisy image. Let the coefficients of $\hat{\mathcal{Z}}$ and $\mathcal{Z}_n$ in the HOSVD bases of $\hat{\mathcal{Z}}$ be denoted as $\hat{c}$ and c_n respectively. Then, the filtered coefficients of $\mathcal{Z}_n$, denoted as $\hat{c}_n$, are computed as follows, followed by the usual transform inversion and averaging:

$$\hat{c}_n = \frac{c_n \hat{c}^2}{\hat{c}^2 + \sigma^2}. \tag{6}$$

This second stage is termed 'HOSVD2'.

3.7 Relationship of NL-SVD and HOSVD/HOSVD2 to Existing Literature

While our HOSVD/HOSVD2 approach and the BM3D algorithm (Section 2) both jointly filter sets of similar patches, there are two major differences. Firstly, in our approach, the entire 3D basis is learned from the noisy data, whereas BM3D uses fixed bases. Secondly, as BM3D performs a Haar transform in the third dimension, it implicitly treats the patches as a signal in the third dimension. On the other hand, our HOSVD/HOSVD2 method does not impose any such continuity properties or 'signalness' in the third dimension. In fact, scrambling the order of the patches in the third dimension will produce the same values of the projection coefficients, except for corresponding permutation operations. Hence, the ordering of patches in the third dimension may potentially alter the output of a denoising algorithm such as BM3D, whereas our method will still remain invariant to this change.

In contrast to learning a global dictionary (as in KSVD [1]), or dictionaries for each patch-cluster (as in KLLD [6]), our proposed HOSVD/HOSVD2 technique learns bases that vary from pixel to pixel, obviating the need for any iterative optimization (just like PCA-based techniques [43], [21]). Both [21] and [43] use PCA to derive bases from vectorized patches, whereas NL-SVD and HOSVD/HOSVD2 derive orthonormal basis-pairs from patches represented as matrices with a row-column separation, leading to much better computational efficiency (see Section 5). We present excellent empirical results, among the best reported in current literature, on both grayscale and color images. Other points of merit of our technique are its conceptual and implementation simplicity and a principled approach toward selection of all required parameters in terms of the noise standard deviation. This also holds true for the NL-SVD algorithm, but its empirical performance is not on par with that of HOSVD/HOSVD2. We would like to emphasize that just like most contemporary techniques [9], [1], our methods NL-SVD, HOSVD/HOSVD2 are specifically designed for zero mean, i.i.d. Gaussian noise of *known* noise standard deviation. However, we present a principled method to estimate the noise standard deviation in a blind scenario in an indirect manner, based upon the properties of the residual images (i.e. the differences between the true and denoised images).

4 Experiments on Grayscale Images

We now describe our experimental results. For our noise model [i.e. additive and i.i.d. $\mathcal{N}(0,\sigma)$], we pick $\sigma \in \{15, 20, 25, 30, 35\}$. We perform experiments on Lansel's benchmark dataset [16] consisting of 13 commonly used images, each of size 512×512. We pit NL-SVD, HOSVD/HOSVD2 against the following: NL-Means [4], KSVD [1], BM3D1 and BM3D2 (i.e. post Wiener filtering) [9], our implementation of BM3D using DCT bases for all three dimensions of the patch stack (as against the combination of biorthogonal/DCT bases and Haar bases as in [9]), which we term '3D-DCT' (see Section 4.2), the oracle denoiser from Section 3.2, and both stages of the PCA-based algorithm from [43] (denoted here as LPG-PCA1 and LPG-PCA2, where LPG stands for local pixel grouping). At each noise level, we compare results in terms of two metrics: (1) PSNR value (where PSNR $\overset{\text{def}}{=} 10\log_{10}\frac{255^2}{\text{MSE}}$), and (2) SSIM value (structured similarity index) computed at patch-size 11×11 (as per the implementation in [36]). All these metrics are measured by first writing the images into a file in a standard image format and then reading them back. Despite the minor quantization issues introduced, we follow this approach as it represents realistic digital storage of images. Similarly, for all σ values, the noisy images are generated by adding Gaussian noise to the original image and converting the result to an image file ([0-255] range). In the case of BM3D, NL-Means and LPG-PCA, we used the software provided by the

1. We experimentally confirmed that increasing the value of K (on all images, on several noise levels, for $K \in 10:10:120$) does not adversely affect performance.

authors online[2]. For KSVD, we used the results already reported by the authors on the denoising benchmark [16]. These results were available only for noise levels in the range $\sigma = 5$ to $\sigma = 25$. The noise-level σ is specified as input for all algorithms, which is required for optimal parameter selection in their provided implementations. For NL-SVD and HOSVD/HOSVD2, we used 8×8 patches in all experiments and a search window radius of 20 around each point. The search window radius is not a free parameter as it affects only computational efficiency and not accuracy. In fact, larger sizes of the search window did not improve the results in our experiments. There are no other free parameters in our technique, apart from the patch-size which is also true of all other patch-based algorithms. Later, in Section 6, we present a criterion for patch-size selection which uses the correlation coefficient between patches from the residual image (i.e. difference between noisy and denoised images). For NL-Means, we used 9×9 patches throughout, with a search window radius of 20. For the BM3D implementation, we used the default settings of all the various parameters as obtained from the authors' software (their selected patch-size is again 8×8). All our experimental results have been reported on all 13 images from the Lansel benchmark [16]. The PSNR and SSIM results for $\sigma \in \{20, 30\}$ are presented in Tables 1 and 2, where we have used numbers to refer to image names as follows: 13 - airplane, 12 - Barbara, 11 - boats, 10 - couple, 9 - elaine, 8 - fingerprint, 7 - goldhill, 6 - Lena, 5 - man, 4 - mandrill, 3 - peppers, 2 - stream, 1 - Zelda. (Please see the *supplemental material* for results on noise levels such as $\sigma \in \{15, 25, 60, 80, 100\}$.) From these tables, it can be observed that HOSVD is superior to NL-Means, 3D-DCT and NL-SVD, whereas NL-SVD is superior to 3D-DCT and NL-Means. Indeed, HOSVD is also superior to KSVD and BM3D1 at higher noise levels ($\sigma \geq 20$) on most images in terms of PSNR/SSIM values, though it lags slightly behind BM3D2. The average difference between the PSNR values produced by HOSVD and BM3D2 at noise levels 20 and 30 is 0.281 and 0.343 respectively (Tables 1 and 2). In all cases, the oracle is a clear winner in terms of PSNR. BM3D2 involves a Wiener filter step, which relies on the assumption that the transform coefficients of the underlying image are Gaussian distributed, which is not a valid assumption for coefficients of natural image patches, but which alone makes a Wiener filter the optimal least squares estimator [27]. We, however, have obtained a prominent gain in performance when we augmented HOSVD with a Wiener filter step (denoted as 'HOSVD2'), further reducing the gap between our method and BM3D2. At very high noise levels ($\sigma >= 60$), HOSVD2 in fact occasionally outperforms BM3D2 (please see *supplemental material*).

2. http://www.ipol.im/pub/algo/bcm_non_local_means_denoising, http://www.cs.tut.fi/~foi/GCF-BM3D, http://www4.comp.polyu.edu.hk/~cslzhang/code/Program_lpgpca.zip

4.1 Comparison with KSVD

The PSNR and SSIM values of NL-SVD are comparable to those reported for KSVD, whereas HOSVD (and especially HOSVD2) often outperforms KSVD. Also, NL-SVD and HOSVD have other conceptual and implementation-related advantages as compared to KSVD. KSVD learns an overcomplete dictionary on the fly from the noisy image, requiring expensive, iterated optimizations prone to local minima (unlike NL-SVD/HOSVD/HOSVD2), which puts artificial limits on the size of the dictionary that can be learned [20]. The KSVD algorithm requires parameters that are not easy to tune: the number of dictionary vectors (K), parameter for the stopping criterion for the projection pursuit algorithm and the tradeoff between data fidelity and sparsity terms. Some of these parameters depend on the noise standard deviation which actually changes when the image is iteratively filtered.

4.2 Comparison with BM3D and 3D-DCT

We refer the reader to Section 2 for a brief description/comparison of the BM3D method. The overall BM3D algorithm contains a number of parameters: the choice of transform for 2D and 1D filtering (whether Haar/DCT/Biorthogonal wavelet), the distance threshold for patch similarity, the thresholds for truncation of transform domain coefficients, a parameter to restrict the maximum number of patches that are similar to any one reference patch, and the choice of pre-filter while computing the similarity between patches in the first stage (BM3D1). There is an analogous set of parameters for the second stage that uses empirical Wiener filtering (BM3D2). In fact, given the complex nature of this algorithm, it may be difficult to isolate the relative contribution of each of its components. Note that NL-SVD and HOSVD also require thresholds for patch similarity and truncation of transform domain coefficients, but these are obtained in a principled manner from the noise model as explained in Section 3.4. The BM3D implementation in [9] uses fixed thresholds with an imprecise relationship to the noise standard deviation. For instance, it uses a distance threshold of 2500 if the noise $\sigma \leq 40$ and a threshold of 5000 otherwise, a transform domain threshold of 2.7σ, and a patch-size of 32×32 and a distance threshold of 400 in the Wiener filtering step. Unlike BM3D, we do not resort to any pre-filtering methods to determine the distance between noisy patches even at high noise levels. Also, for HOSVD2 which uses Wiener filtering, we stick to the same patch-size (8×8) and distance threshold, as for HOSVD.

We emphasize that the principled selection of a fixed transform basis (whether DCT or wavelet) is a difficult

TABLE 1
PSNR and SSIM values for noise level $\sigma = 20$ on the benchmark dataset

Image #	NL-SVD	NL-Means	KSVD	HOSVD	HOSVD2	3DDCT	BM3D1	BM3D2	Oracle	LPG-PCA1	LPG-PCA2
13	31.936	30.540	32.266	32.015	32.382	31.433	32.028	32.552	37.695	31.446	32.241
12	30.878	29.421	30.762	31.533	31.845	30.543	31.026	31.660	36.603	30.798	31.385
11	30.187	28.911	30.360	30.491	30.646	29.596	30.395	30.802	35.510	30.020	30.251
10	29.961	28.387	29.929	30.299	30.472	29.422	30.252	30.698	36.066	29.839	30.054
9	31.135	29.924	31.341	31.354	31.207	30.887	31.284	31.433	34.178	31.09	31.363
8	28.053	27.424	28.454	28.563	28.724	27.389	28.403	28.794	35.318	28.367	28.642
7	30.098	28.931	30.166	30.536	30.567	29.532	30.397	30.726	35.241	30.052	30.227
6	32.240	30.473	32.371	32.411	32.653	31.903	32.375	32.950	36.975	31.853	32.633
5	28.939	27.995	28.853	29.291	29.448	28.250	29.200	29.464	34.989	29.234	29.360
4	25.976	25.933	26.372	25.720	26.360	25.543	26.260	26.582	33.434	26.509	26.420
3	32.009	30.357	32.005	32.166	32.245	31.740	32.138	32.498	36.260	31.663	32.331
2	26.800	26.375	27.062	26.722	27.026	25.892	26.918	27.192	33.485	27.022	26.947
1	33.401	30.902	33.494	33.525	33.667	33.169	33.430	34.075	37.971	32.784	33.770
13	0.885	0.802	0.893	0.869	0.888	0.885	0.875	0.899	0.959	0.817	0.895
12	0.882	0.821	0.877	0.897	0.902	0.884	0.884	0.903	0.956	0.862	0.899
11	0.801	0.753	0.803	0.814	0.821	0.789	0.809	0.824	0.922	0.789	0.809
10	0.816	0.755	0.812	0.831	0.840	0.806	0.828	0.845	0.945	0.807	0.825
9	0.747	0.723	0.755	0.761	0.750	0.740	0.755	0.754	0.859	0.753	0.761
8	0.914	0.903	0.922	0.926	0.930	0.899	0.922	0.930	0.984	0.922	0.923
7	0.778	0.736	0.776	0.800	0.804	0.761	0.793	0.807	0.924	0.775	0.790
6	0.858	0.782	0.861	0.852	0.867	0.861	0.855	0.875	0.938	0.817	0.872
5	0.775	0.729	0.768	0.792	0.798	0.753	0.784	0.796	0.930	0.789	0.808
4	0.765	0.760	0.780	0.764	0.789	0.722	0.776	0.792	0.936	0.787	0.778
3	0.835	0.769	0.835	0.830	0.835	0.836	0.831	0.843	0.918	0.800	0.846
2	0.764	0.746	0.773	0.767	0.783	0.700	0.771	0.786	0.942	0.778	0.773
1	0.867	0.777	0.869	0.859	0.866	0.871	0.862	0.880	0.944	0.823	0.879

TABLE 2
PSNR values for noise level $\sigma = 30$ on the benchmark dataset

Image #	NL-SVD	NL-Means	HOSVD	HOSVD2	3DDCT	BM3D1	BM3D2	Oracle	LPG-PCA1	LPG-PCA2
13	29.875	27.680	30.079	30.356	29.437	30.101	30.711	35.098	28.885	30.031
12	28.639	26.853	29.462	29.795	28.498	28.952	29.793	33.696	28.378	29.147
11	28.305	26.368	28.650	28.862	27.656	28.466	29.017	33.380	27.758	28.243
10	27.740	25.665	28.290	28.462	27.268	28.154	28.759	33.635	27.438	27.853
9	29.997	27.865	29.976	30.016	29.770	29.947	30.420	32.770	29.129	30.006
8	25.863	24.552	26.676	26.825	25.434	26.382	26.874	32.251	26.010	26.268
7	28.357	26.576	28.798	28.918	27.993	28.654	29.145	33.160	27.990	28.464
6	30.233	28.080	30.411	30.757	30.000	30.417	31.194	34.597	29.422	30.659
5	26.778	25.176	27.278	27.331	26.248	27.109	27.353	32.041	27.177	27.471
4	24.139	23.396	24.293	24.502	23.094	24.208	24.551	30.233	24.265	24.207
3	29.996	27.438	30.150	30.306	29.824	30.164	30.673	34.259	29.420	30.638
2	24.884	23.858	25.278	25.340	24.041	25.115	25.336	30.289	25.055	25.008
1	31.549	28.388	31.385	31.717	31.166	31.267	32.130	35.861	30.202	31.828
13	0.853	0.717	0.815	0.855	0.856	0.832	0.873	0.944	0.704	0.860
12	0.824	0.730	0.836	0.861	0.840	0.833	0.868	0.930	0.775	0.855
11	0.746	0.650	0.752	0.769	0.736	0.750	0.779	0.901	0.689	0.752
10	0.742	0.639	0.760	0.777	0.737	0.757	0.790	0.925	0.703	0.757
9	0.715	0.644	0.710	0.713	0.716	0.712	0.727	0.829	0.664	0.723
8	0.860	0.826	0.890	0.894	0.851	0.881	0.895	0.968	0.873	0.877
7	0.713	0.631	0.733	0.746	0.707	0.728	0.753	0.898	0.675	0.726
6	0.815	0.688	0.791	0.825	0.825	0.803	0.843	0.918	0.712	0.836
5	0.690	0.610	0.710	0.721	0.677	0.704	0.722	0.891	0.694	0.745
4	0.664	0.624	0.686	0.703	0.587	0.677	0.701	0.896	0.677	0.678
3	0.796	0.676	0.771	0.794	0.804	0.782	0.811	0.900	0.705	0.818
2	0.649	0.609	0.687	0.696	0.592	0.675	0.690	0.897	0.673	0.671
1	0.829	0.683	0.797	0.824	0.837	0.806	0.845	0.929	0.714	0.847

task and significantly affects the denoising performance. We seek to illustrate this point by comparing NL-SVD and HOSVD with our '3D-DCT' implementation. We again place an upper limit of $K = 30$ on the number of similar patches in an ensemble, and use the hard threshold of $\sigma\sqrt{\log n^2 K}$, exactly as in our HOSVD implementation. As can be seen in Tables 1 and 2 (and results in the *supplemental material*), NL-SVD and HOSVD consistently outperform 3D-DCT. We believe this sufficiently illustrates the advantages of our method for non-local basis learning.

4.3 Visual Comparison of the Denoised Images and their Residuals

The original and noisy images [from $\mathcal{N}(0, 20)$], and the denoised images produced by NL-SVD, NL-Means, BM3D1, BM3D2, HOSVD and HOSVD2 can be viewed in Figures 5, 6 and 7. The reader is urged to zoom into the pdf file to see finer differences between the images. Upon zooming, one can observe shock-like artifacts in certain portions of the denoised images produced by BM3D, especially by BM3D2. One example is Barbara's face from Figure 5 - see Figure 8 for a zoomed-in view. These artifacts are absent in NL-SVD even though it produces a blurrier image. HOSVD as well as BM3D outperform NL-SVD

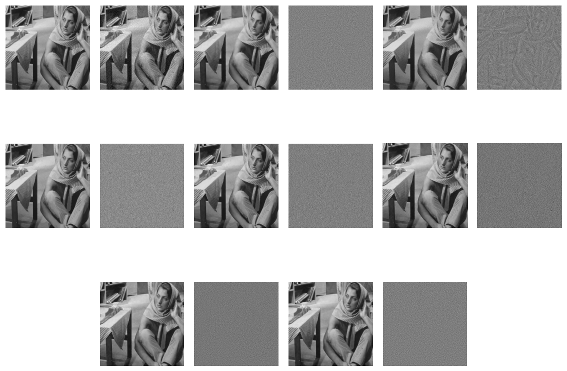


Fig. 5. Barbara image: left to right, top to bottom - clean image, noisy version with $\sigma = 20$/PSNR = 22, output and residual for NL-SVD, NL-Means, BM3D1, BM3D2, HOSVD, HOSVD2, oracle. Zoom into pdf file for a detailed view. PSNR values in Table 1, image 12.

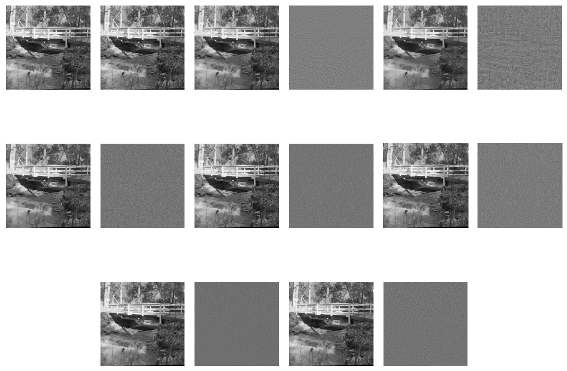


Fig. 7. Boat image: left to right, top to bottom - clean image, noisy version with $\sigma = 20$/PSNR = 22, output and residual for NL-SVD, NL-Means, BM3D1, BM3D2, HOSVD, HOSVD2, oracle. Zoom into pdf file for a detailed view. PSNR values in Table 1, image 2.

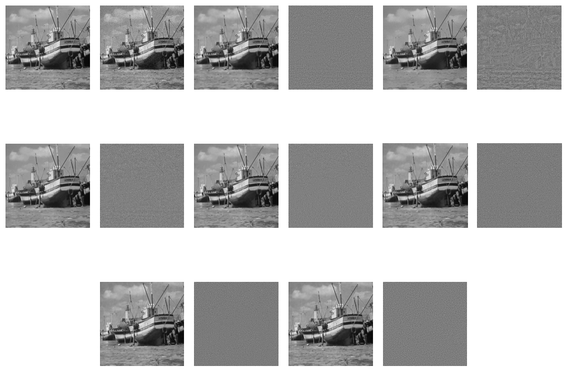


Fig. 6. Boat image: left to right, top to bottom - clean image, noisy version with $\sigma = 20$/PSNR = 22, output and residual for NL-SVD, NL-Means, BM3D1, BM3D2, HOSVD, HOSVD2, oracle. Zoom into pdf file for a detailed view. PSNR values in Table 1, image 11.

on finer edges and texture, however HOSVD does not produce the shock artifacts that BM3D does. The images produced by HOSVD, however, have a finely grainy appearance, because collective smoothing of multiple patches tends to undersmooth them slightly. Overall, among all methods, the images produced by the oracle are undoubtedly the best, especially for preserving fine shading effects that all other methods erase.

We performed a more detailed numerical comparison between the outputs of NL-SVD, BM3D1, BM3D2 and HOSVD on a portion of the Barbara image. For this, we computed the absolute difference images between the true image and the outputs of these algorithms, as shown in Figure 8. The mean absolute error values over the chosen subimage were 5.96 (for NL-SVD), 5.76 (BM3D1), 5.30 (BM3D2) and 5.50 (HOSVD). The mean ℓ_2 errors were 61.89, 58.13, 49.87 and 53.02 respectively. The errors produced by NL-SVD were greater than those by BM3D1/BM3D2 for only 46-50% of the pixels, and the corresponding range for HOSVD was only 43.5-46%. We also ran a Canny edge detector (with the default parameters from the MATLAB® implementation) on the true image, and computed the errors only on the edge pixels. The mean absolute errors on edge pixels were 6.42, 6.418, 6.13 and 6.128 for NL-SVD, BM3D1, BM3D2 and HOSVD respectively, whereas the mean ℓ_2 errors on edge pixels were 68.63, 68.65, 62.15 and 62.75 respectively. However, for only around 46-47% percent of the edge pixels was the error for NL-SVD greater than that for BM3D1/BM3D2, and the corresponding range for HOSVD was 42-43%. This further shows that HOSVD and BM3D yield comparable performance on edges and textured regions.

The residual images for various algorithms are shown in the bottom rows of Figures 5, 6 and 7. Note that the residual is calculated as the difference between the noisy and denoised image, with the difference image normalized between 0 and 255. Ideally, the residual should obey the properties of the noise model and therefore necessarily be devoid of structure. NL-Means produces undesirably structured residuals. Some structure is visible in the residuals produced by NL-SVD and BM3D1, whereas those by produced by BM3D2 and HOSVD are the noisiest, and visually quite similar to those produced by the oracle.

5 Comparison of Time Complexity

Assume that the number of image pixels is N, that the average time to compute similar patches per reference patch is T_S, that the average number of patches similar to the reference patch is K and that the size of the patch is $p \times p$. The complexity of NL-SVD is

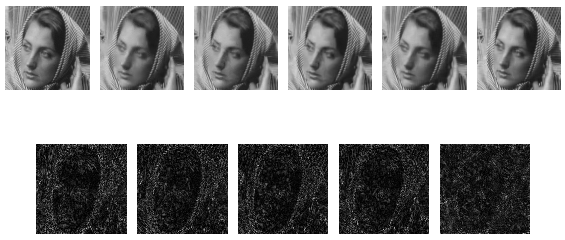


Fig. 8. Top Row: Barbara's face from (left to right) the original image, and denoised versions (from noise in $\mathcal{N}(0,20)$) produced by NL-SVD, BM3D1, BM3D2, HOSVD, Oracle. Bottom row: Absolute difference between the image of Barbara's face and the denoised image produced by (left to right) NL-SVD, BM3D1, BM3D2, HOSVD and Oracle. Zoom into pdf file for a detailed view.

$\mathcal{O}([T_S + Kp^3]N)$ because the eigendecomposition of a $p \times p$ matrix and the multiplication of two $p \times p$ matrices are both $\mathcal{O}(p^3)$ operations. BM3D requires $\mathcal{O}(Kp^3)$ time for the 2D transforms and $\mathcal{O}(K^2p^2)$ time for the 1D transforms, if the transforms are implemented using simple matrix multiplication. This leads to a total complexity of $\mathcal{O}([T_S + Kp^3 + K^2p^2]N)$. If algorithms such as the fast Fourier transform are used, this complexity reduces to $\mathcal{O}([T_S + Kp^2 \log p + p^2 K \log K]N)$. If we assume that $p \ll K$ (which is desirable for good performance of a non-local algorithm), then NL-SVD is slightly faster than the more efficient version of BM3D. The complexity of HOSVD is obtained as follows. Given a patch stack of size $p \times p \times K$, the size of two of its unfoldings is $p \times pK$, the SVD of which is $\mathcal{O}(Kp^3)$. The third unfolding has size $K \times p^2$, the SVD of which is $\mathcal{O}(\min(K^2p^2, Kp^4))$. Hence, the total complexity of BM3D is $\mathcal{O}([T_S + Kp^3 + \min(K^2p^2, Kp^4)]N)$. Note again that NL-SVD and HOSVD follow the concept of matrix based patch representations as followed by [26], [12] and [14]. PCA-based methods such as [21] and [43] represent each $p \times p$ patch as a $p^2 \times 1$ vector and build a $p^2 \times p^2$ covariance matrix to produce the spatially adaptive bases, leading to a time complexity of $\mathcal{O}([T_S + Kp^4 + p^6]N)$ which is far greater than that of NL-SVD or HOSVD. The KSVD technique also follows a vector-based patch representation and the K learned bases have size $p^2 \times 1$ (with $K \gg p^2$), leading to a time complexity of $\mathcal{O}(p^2 KLNJ)$ for sparsity factor L and J iterations for the optimization.

6 Selection of Global Patch-Size and Estimation of Noise Standard Deviation

All results so far were reported for a fixed patch-size of 8×8, a commonly used parameter value in patch-based algorithms (including JPEG). Here, we present an objective criterion for selecting the patch-size that will yield the best denoising performance. For this, we consider the residual images after denoising with NL-SVD using a fixed patch-size $p \times p$, with a threshold of $\sigma\sqrt{2 \log p^2}$ for hard-thresholding the transform coefficients. Each residual image is divided into non-overlapping patches of size $q \times q$ where $q \in \{8, 9, ..., 15, 16\}$. For each value of q, we compute the average, absolute correlation coefficient between all pairs of patches in the residual image, and then calculate the sum of these average values. The absolute correlation coefficient between vectors v_1 and v_2 (of size $q^2 \times 1$) is defined as follows:

$$\rho_{pq}(v_1, v_2) = \frac{1}{q^2} \frac{|(v_1 - \mu_1)^T (v_2 - \mu_2)|}{\sigma_{v_1} \sigma_{v_2}} \tag{7}$$

where μ_1 and μ_2 are the mean values of vectors v_1 and v_2, and σ_{v_1} and σ_{v_2} are their corresponding standard deviations. Our intuition is that an optimal denoiser will produce residual patches that are highly decorrelated with one another as measured by ρ_{pq}. However, ρ_{pq} is certainly dependent upon the patch-size $q \times q$ that is used for computation of the statistics. Hence, we sum the cross-correlation values over q and over all patch pairs, thus giving us

$$\rho_p = \sum_{i \in \Omega, j \in \Omega, q} \rho_{pq}(v_i, v_j) \tag{8}$$

as the final measure. Here, v_i and v_j denote patches (in vector form) with their upper left corner at locations i and j (respectively) in the image domain Ω. The patch-size $p \times p$ which produces the least value of ρ_p is selected as the optimal parameter value. In our experiments, we varied p from 3 to 16. We have observed that the PSNR corresponding to the optimal ρ_p is very close to the optimal PSNR. This can be seen in Table 3, where for each image in the benchmark database, we report the following: (1) the highest PSNR across $p \in \{3, 4, 5, ..., 15, 16\}$, (2) the patch-size which produced that PSNR, (3) the lowest ρ_p value across p, (4) the patch-size which produced the lowest ρ_p value and (5) the PSNR for the best patch-size as per the criterion ρ_p. One can see from Table 3 that the drop in PSNR (if any) is very low. The denoised images and their residuals for different patch-sizes are also shown in Figure 9. The noise-level for all these results is $\sigma = 20$. Ideally, there may not be a single optimal patch-size for the entire image. A better approach would be to adapt the patch-size based on the local structure of the image. However, given the aggregation of hypotheses from (and consequent dependence on) neighboring patches, this turns out to be a non-trivial problem.

We applied the criterion ρ also for estimation of noise standard deviation from the residuals of the NL-SVD algorithm, assuming a fixed patch-size of 8×8. We experimented on the 'boats' image under noise levels of $\sigma = 5$ to $\sigma = 50$ in steps of 5. Let us denote the particular value of noise standard deviation provided as input parameter to the algorithm to be σ_t.

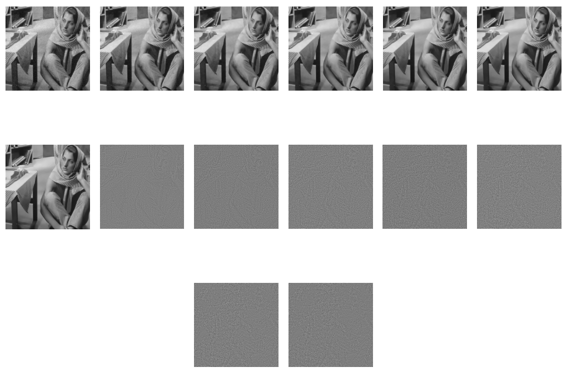


Fig. 9. Reconstructed images when Barbara (with noise $\sigma = 20$) is denoised with NL-SVD run on patch-sizes (from left to right) 4×4, 6×6, 8×8, 10×10, 12×12, 14×14 and 16×16. The corresponding residuals are in the next row (left to right). Zoom into pdf file for a detailed view.

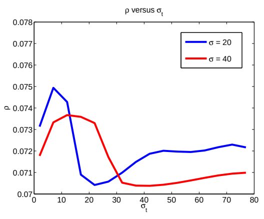


Fig. 10. Plot of ρ versus σ_t for $\sigma = 20$ and $\sigma = 40$, see Section 6.

For each noise level σ, we computed the residuals by using $\sigma_t = 2$ to $\sigma_t = 82$ in steps of 5 (the thresholds for patch similarity and nullification of coefficients were computed using σ_t). The estimate of the noise standard deviation $\hat{\sigma}$ was chosen to be the particular σ_t for which the value of ρ was the least. For all noise levels, our criterion yielded us accurate estimates. We show a sample plot of ρ versus σ_t in Figure 10 for $\sigma = 20$ and $\sigma = 40$.

7 Experiments on Color Images

We show PSNR and SSIM values for denoising all 24 RGB images from the Kodak gallery[3], under independent $\mathcal{N}(0, \sigma)$ noise on each channel (where $\sigma \in \{30, 40, 50\}$), in Tables 4, 5 and 6 respectively. As for grayscale images and for all σ values, the noisy color images were generated by adding Gaussian noise to the original image and converting the result to an image file ([0-255] range). We compare our results using HOSVD on 4D stacks (termed '4D-HOSVD', with the different channels representing the fourth dimension) with NL-Means and the color version of BM3D in YCbCr color space (using the authors' respective implementations). We also implemented a

3. http://r0k.us/graphics/kodak/

TABLE 3
Patch-size selection for $\sigma = 20$ as per criterion ρ, Section 6

Image #	Best PSNR	Best patch-size (by PSNR)	Best ρ	Best patch-size (by ρ)	Best PSNR by ρ
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	6	28.110
7	30.190	8	9.639	14	30.024
6	32.350	10	9.641	14	32.190
5	29.190	5	9.644	9	28.890
4	26.166	4	9.756	6	26.044
3	32.020	8	9.639	12	31.808
2	27.020	5	9.673	6	27.020
1	33.510	10	9.635	11	33.499

Wiener filter step on top of 4D-HOSVD, termed '4D-HOSVD2' (with details similar to Equation 6). We also implemented the following variant of our HOSVD technique: we learned a decorrelated color space from the noisy R, G, B values by principal component analysis (PCA) and then applied the HOSVD denoising algorithm for grayscale images independently on the three resulting PCA-transformed channels. We refer to this as the 'independent 3D HOSVD' or '3D-IHOSVD'. For 3D-IHOSVD, we compute the patch similarity independently on the three channels obtained after PCA using the distance threshold $\tau_d = 3\sigma^2 n^2$; whereas in BM3D [9], the patch similarity is computed only over the Y channel from the YCbCr color space, which ignores chrominance information. For 4D-HOSVD/4D-HOSVD2, the distance threshold used was $\tau_d = 3 \times 3\sigma^2 n^2$. The patch-size used for all algorithms was 8×8.

The 4D-HOSVD method clearly outperformed NL-Means, 3D-IHOSVD and often BM3D1. Its PSNR values are very slightly lower than those of BM3D2, however it outperformed BM3D2 on some images. 4D-HOSVD2 outperformed BM3D2 on 18 out of 24 images at $\sigma = 40$ and 15 out of 24 images at $\sigma = 50$ in terms of PSNR. Beyond PSNR comparisons, we have observed that at higher noise levels, 3D-IHOSVD and BM3D2 produce color artifacts that alter the hue, unlike 4D-HOSVD/4D-HOSVD2 (see Figures 11 and 12). Thus, 4D-HOSVD2 is clearly a state of the art color denoising algorithm. Sample results using 4D-HOSVD are shown in Figure 12 and also in Figure 1, all under $\mathcal{N}(0, 30)$. (Please see the *supplemental material* for more results.) On color images, we consistently outperformed LPG-PCA by almost 1 to 1.5 dB at several noise levels. This is because the LPG-PCA implementation for color images in [43] denoises the R, G, B channels independently (section 3.5 of [43]), ignoring the coupling.

8 Conclusion

We have presented an extremely simple denoising algorithm - the HOSVD of similar image patches in

TABLE 4
PSNR results for color images (and SSIM values on *gray-scale versions*) corrupted by $\mathcal{N}(0, 30)$

Image #	BM3D1	BM3D2	NLM	3D-IHOSVD	4DHOSVD	4DHOSVD2
1	27.734, 0.799	28.097, 0.814	26.155, 0.766	27.527, 0.797	27.341, 0.771	27.903, 0.798
2	30.821, 0.802	31.254, 0.816	28.958, 0.732	30.532, 0.798	30.827, 0.796	31.088, 0.807
3	32.241, 0.870	33.070, 0.899	29.433, 0.764	31.742, 0.868	32.337, 0.878	32.787, 0.892
4	30.957, 0.815	30.957, 0.815	28.626, 0.732	30.697, 0.816	31.068, 0.809	31.425, 0.823
5	27.804, 0.838	28.383, 0.859	26.110, 0.786	27.787, 0.846	27.557, 0.834	28.348, 0.861
6	28.640, 0.818	29.015, 0.840	26.820, 0.759	28.431, 0.822	28.378, 0.798	28.890, 0.825
7	31.519, 0.899	32.375, 0.922	28.455, 0.792	31.063, 0.899	31.726, 0.911	32.258, 0.920
8	28.058, 0.871	28.408, 0.881	25.979, 0.824	27.953, 0.876	28.028, 0.864	28.607, 0.879
9	32.158, 0.871	33.040, 0.898	28.985, 0.764	31.725, 0.870	32.447, 0.883	32.912, 0.892
10	31.779, 0.856	32.664, 0.882	28.719, 0.748	31.425, 0.859	32.005, 0.862	32.554, 0.875
11	29.469, 0.800	29.995, 0.822	27.536, 0.734	29.399, 0.807	29.282, 0.791	29.781, 0.812
12	31.711, 0.828	32.388, 0.854	29.074, 0.737	31.252, 0.832	31.980, 0.832	32.276, 0.844
13	25.887, 0.744	26.350, 0.763	24.832, 0.739	25.761, 0.745	25.476, 0.713	26.078, 0.749
14	28.410, 0.783	28.878, 0.799	26.927, 0.741	28.267, 0.787	28.273, 0.767	28.834, 0.793
15	30.395, 0.839	30.670, 0.854	28.642, 0.777	30.094, 0.839	30.527, 0.844	30.711, 0.855
16	30.616, 0.809	31.335, 0.840	28.192, 0.716	30.420, 0.817	30.621, 0.807	31.126, 0.828
17	30.755, 0.846	31.212, 0.861	28.426, 0.771	30.607, 0.849	30.686, 0.850	31.081, 0.864
18	27.923, 0.783	28.489, 0.807	26.555, 0.726	27.822, 0.790	27.548, 0.773	28.178, 0.806
19	30.312, 0.815	30.782, 0.833	28.110, 0.746	30.106, 0.818	30.315, 0.814	30.669, 0.826
20	28.407, 0.886	28.189, 0.892	27.253, 0.840	28.002, 0.881	28.507, 0.877	28.456, 0.884
21	29.280, 0.846	29.822, 0.872	27.340, 0.765	29.032, 0.845	29.028, 0.848	29.610, 0.866
22	29.370, 0.781	29.898, 0.802	27.648, 0.714	29.220, 0.783	29.234, 0.766	29.697, 0.789
23	32.414, 0.887	33.243, 0.911	29.597, 0.779	32.032, 0.886	33.056, 0.903	33.378, 0.907
24	28.056, 0.828	28.503, 0.849	26.552, 0.759	27.930, 0.831	27.834, 0.81	28.505, 0.840

TABLE 5
PSNR results for color images (and SSIM values on *gray-scale versions*) corrupted by $\mathcal{N}(0, 40)$

Image #	BM3D1	BM3D2	NLM	3D-IHOSVD	4DHOSVD	4DHOSVD2
1	26.314, 0.739	26.478, 0.747	24.400, 0.682	26.143, 0.742	26.03, 0.712	26.479, 0.740
2	29.197, 0.766	29.457, 0.777	27.206, 0.670	29.043, 0.761	29.526, 0.771	29.710, 0.778
3	30.541, 0.834	30.892, 0.864	27.468, 0.687	29.947, 0.827	30.737, 0.848	31.029, 0.857
4	29.567, 0.773	29.818, 0.791	26.927, 0.656	29.227, 0.774	29.785, 0.777	30.088, 0.790
5	25.946, 0.774	26.258, 0.788	24.181, 0.697	26.009, 0.787	25.735, 0.776	26.430, 0.812
6	27.034, 0.764	27.165, 0.777	25.074, 0.680	26.816, 0.769	26.828, 0.748	27.239, 0.774
7	29.667, 0.860	30.234, 0.891	26.206, 0.696	29.246, 0.859	29.875, 0.882	30.381, 0.892
8	26.256, 0.828	26.339, 0.832	24.067, 0.753	26.248, 0.838	26.192, 0.829	26.781, 0.844
9	30.644, 0.834	31.190, 0.868	27.071, 0.678	30.056, 0.830	30.970, 0.853	31.402, 0.865
10	30.133, 0.812	30.661, 0.843	26.778, 0.659	29.726, 0.815	30.413, 0.831	30.895, 0.843
11	27.822, 0.745	28.090, 0.759	25.739, 0.652	27.732, 0.754	27.706, 0.750	28.120, 0.771
12	30.219, 0.789	30.581, 0.816	27.349, 0.666	29.642, 0.792	30.654, 0.807	30.829, 0.815
13	24.227, 0.662	24.572, 0.668	23.368, 0.653	24.196, 0.672	23.782, 0.635	24.412, 0.675
14	26.875, 0.726	27.133, 0.735	25.184, 0.664	26.808, 0.734	26.780, 0.715	27.243, 0.741
15	28.518, 0.804	28.499, 0.815	26.714, 0.720	28.175, 0.801	28.688, 0.819	28.719, 0.829
16	29.230, 0.753	29.599, 0.780	26.511, 0.628	28.938, 0.764	29.292, 0.762	29.663, 0.782
17	28.815, 0.800	28.968, 0.814	26.386, 0.692	28.610, 0.800	28.761, 0.817	29.056, 0.831
18	26.273, 0.716	26.637, 0.733	24.939, 0.641	25.377, 0.674	25.942, 0.710	26.467, 0.747
19	28.926, 0.770	29.143, 0.789	26.339, 0.668	28.386, 0.767	29.070, 0.784	29.378, 0.794
20	26.257, 0.861	25.871, 0.863	25.059, 0.788	25.699, 0.840	26.378, 0.856	26.257, 0.861
21	27.695, 0.796	28.093, 0.827	25.558, 0.675	26.616, 0.774	27.530, 0.810	28.045, 0.829
22	28.046, 0.726	28.394, 0.744	26.065, 0.632	27.456, 0.704	28.053, 0.725	28.422, 0.745
23	30.653, 0.860	31.242, 0.890	27.814, 0.710	30.113, 0.865	31.283, 0.884	31.568, 0.888
24	26.381, 0.769	26.639, 0.784	24.856, 0.676	25.178, 0.730	26.122, 0.762	26.745, 0.788

Fig. 11. Left to right: Original image, Noisy image under $\mathcal{N}(0, 40)$, outputs of BM3D2 [PSNR: 26.339], 4D-HOSVD [PSNR: 26.192] and 4D-HOSVD2 [PSNR: 26.7]. Zoom into pdf file for better view. Notice the color artifacts in BM3D2, which are absent in 4D-HOSVD/4D-HOSVD2.

conjunction with hard thresholding and averaging. We have demonstrated its excellent empirical performance in comparison with state of the art algorithms through a large number of experiments on two full databases (as opposed to arbitrary image subsets from image databases). Most of the competing algorithms are either more computationally expensive or more complex in terms of implementation. As the parameters in our technique are all tied to the noise model, our method can be elegantly and easily extended to handle non-Gaussian noise models. We have specified principled criteria for the selection of patch-size or the estimation of the noise standard deviation when unknown. Augmented with a Wiener filter step, the HOSVD method outperforms state of the art algorithms such as BM3D2 and LPG-PCA on color images. On gray-scale images, the HOSVD with the Wiener filter outperforms LPG-PCA and comes very close to BM3D2. But we have observed that it falls short of the shape adaptive implementation of BM3D [10].

TABLE 6
PSNR results for color images (and SSIM values on *gray-scale versions*) corrupted by $\mathcal{N}(0, 50)$

Image #	BM3D1	BM3D2	4DHOSVD	4DHOSVD2
1	24.976, 0.658	25.330, 0.683	24.995, 0.646	25.372, 0.671
2	27.575, 0.706	27.761, 0.719	28.278, 0.711	28.426, 0.718
3	29.105, 0.780	29.600, 0.829	29.365, 0.803	29.551, 0.811
4	28.284, 0.707	28.719, 0.748	28.654, 0.736	28.900, 0.745
5	24.226, 0.689	24.663, 0.718	24.312, 0.689	24.884, 0.724
6	25.703, 0.701	26.005, 0.732	25.577, 0.688	25.890, 0.714
7	28.124, 0.794	28.844, 0.855	28.342, 0.837	28.801, 0.850
8	24.478, 0.771	24.931, 0.794	24.661, 0.769	25.222, 0.793
9	29.421, 0.780	30.228, 0.846	29.713, 0.821	30.084, 0.833
10	28.844, 0.751	29.564, 0.815	29.120, 0.791	29.546, 0.805
11	26.366, 0.668	26.732, 0.705	26.457, 0.684	26.757, 0.701
12	29.058, 0.743	29.467, 0.789	29.435, 0.775	29.496, 0.783
13	22.852, 0.567	23.191, 0.584	22.622, 0.544	23.158, 0.581
14	25.356, 0.643	25.671, 0.666	25.616, 0.648	26.003, 0.674
15	26.727, 0.765	26.754, 0.779	26.959, 0.768	26.875, 0.776
16	28.109, 0.686	28.574, 0.739	28.265, 0.714	28.489, 0.732
17	26.990, 0.730	27.172, 0.761	27.132, 0.745	27.323, 0.759
18	24.782, 0.632	25.097, 0.657	24.766, 0.624	25.127, 0.653
19	27.640, 0.712	28.059, 0.755	27.959, 0.749	28.224, 0.757
20	24.371, 0.813	24.204, 0.821	24.608, 0.811	24.432, 0.819
21	26.339, 0.727	26.809, 0.782	26.344, 0.754	26.773, 0.773
22	26.959, 0.657	27.379, 0.693	27.114, 0.675	27.412, 0.691
23	29.255, 0.820	29.749, 0.862	29.676, 0.849	29.883, 0.854
24	24.920, 0.693	25.296, 0.731	24.901, 0.697	25.422, 0.724

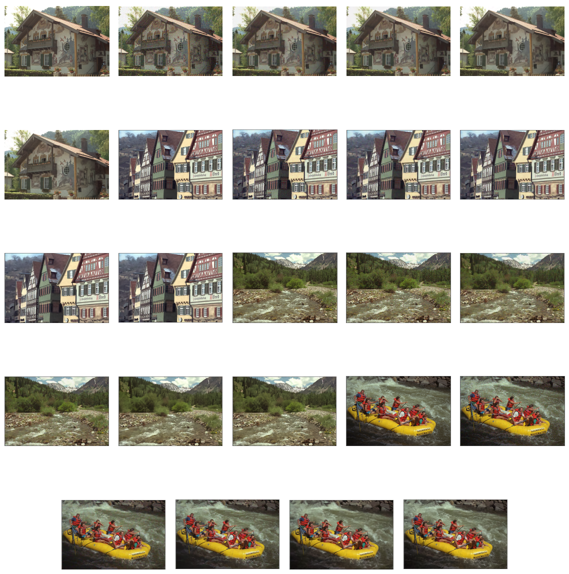


Fig. 12. Four images from Kodak database (# 24,8,13,14): from left to right, original, noisy ($\sigma = 30$ on R, G, B) and denoised with 3D-IHOSVD, BM3D2, 4D-HOSVD and 4D-HOSVD2 [PSNR values in Table 4]. Please zoom into pdf file for a detailed view. BM3D2 shows subtle color artifacts (especially in the 2nd and 3rd image) which are absent in 4D-HOSVD/4D-HOSVD2.

Moreover, we observe two definite shortcomings of our technique, as also of all reported techniques in the current state of the art. First, there is no method, to the best of our knowledge, for the selection of the optimal patch-size, when it is allowed to vary across the image. Second, and more importantly, the **oracle** denoiser clearly outperforms all methods discussed in this paper, including ours, by as much as 4-5 dB in terms of PSNR. This amply illustrates the fact that there is tremendous scope for improvement in the field of image denoising, contrary to emerging belief.

Recently, other successful denoising techniques such as [46] and [41] (which use Gaussian mixture models), or [19], [44] and [45] (which learn overcomplete dictionaries), have emerged. In comparison to these methods, our technique is quite simple and also obtains separable bases (as opposed to vectorized patch representations). The approach in [45], (an extension of the beautiful, non-parametric Bayesian dictionary learning method in [44]), removes spiky noise in addition to Gaussian noise. Extending our method to handle such varied noise models is one important avenue for future research.

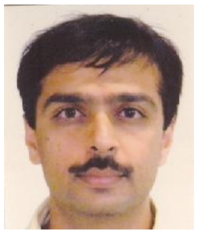
Acknowledgments

This work is partially supported by NSF IIS 1143963. We acknowledge helpful conversations with Brett Presnell, Baba Vemuri, Yair Weiss and Andrew Zisserman.

References

[1] 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.
[2] H. Andrews and C. Patterson, "Singular value decompositions and digital image processing," *IEEE Trans. Acoust., Speech and Signal Process.*, vol. 24, no. 1, pp. 425–432, 1976.
[3] S. Awate and R. Whitaker, "Unsupervised, information-theoretic, adaptive image filtering for image restoration," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 28, no. 3, pp. 364–376, 2006.
[4] A. Buades, B. Coll, and J.-M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale modelling and simulation*, vol. 4, no. 2, pp. 490–530, 2005.
[5] ——, "Neighborhood filters and PDEs," *Numerische Mathematik*, vol. 105, no. 1, pp. 1–34, 2006.
[6] P. Chatterjee and P. Milanfar, "Clustering-based denoising with locally learned dictionaries," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1438–1451, 2009.
[7] R. Coifman and D. Donoho, "Translation-invariant denoising," Yale University, Tech. Rep., 1995.
[8] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.
[9] 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.
[10] ——, "BM3D image denoising with shape-adaptive principal component analysis," in *Workshop on Signal Processing with Adaptive Sparse Structured Representations*, 2009.
[11] L. de Lathauwer, "Signal processing based on multilinear algebra," Ph.D. dissertation, Katholieke Universiteit Leuven, Belgium, 1997.
[12] C. Ding and J. Ye, "Two-dimensional singular value decomposition (2DSVD) for 2D maps and images," in *SIAM Intl. Conf. Data Mining*, 2005, pp. 32–43.
[13] D. Donoho and I. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, pp. 425–455, 1993.
[14] K. Gurumoorthy, A. Rajwade, A. Banerjee, and A. Rangarajan, "A method for compact image representation using sparse matrix and tensor projections onto exemplar orthonormal bases," *IEEE Trans. Image Process.*, vol. 19, no. 2, pp. 322–334, 2010.

[15] A. Hyvarinen, P. Hoyer, and E. Oja, "Image denoising by sparse code shrinkage," in *Intelligent Signal Processing*, 1999, pp. 1–6.
[16] S. Lansel, "DenoiseLab," Available from http://www.stanford.edu/~slansel/DenoiseLab/documentation.htm, 2006.
[17] D. Letexier and S. Bourennane, "Adaptive flattening for multidimensional image restoration," *IEEE Signal Processing Letters*, vol. 15, pp. 229–232, 2008.
[18] M. Lewicki, T. Sejnowski, and H. Hughes, "Learning overcomplete representations," *Neural Computation*, vol. 12, pp. 337–365, 1998.
[19] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *IEEE Intl. Conf. Computer Vision (ICCV)*, 2009, pp. 2272–2279.
[20] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *Multiscale Modeling and Simulation*, vol. 7, no. 1, pp. 214–241, 2008.
[21] D. Muresan and T. Parks, "Adaptive principal components and image denoising," in *IEEE Intl. Conf. Image Process. (ICIP)*, 2003, pp. 101–104.
[22] B. Olshausen and D. Field, "Emergence of simple-cell receptive-field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.
[23] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, 1990.
[24] K. Popat and R. Picard, "Cluster-based probability model and its application to image and texture processing," *IEEE Trans. Image Process.*, vol. 6, no. 2, pp. 268–284, 1997.
[25] J. Portilla, V. Strela, M. Wainwright, and E. Simoncelli, "Image denoising using scale mixtures of Gaussians in the wavelet domain," *IEEE Trans. Image Process.*, vol. 12, no. 11, pp. 1338–1351, 2003.
[26] A. Rangarajan, "Learning matrix space image representations," in *Energy Min. Methods Computer Vision Pattern Recognition (EMMCVPR)*, 2001, pp. 153–168.
[27] A. Rosenfeld and A. Kak, *Digital Picture Processing*. Orlando, USA: Academic Press, 1982.
[28] L. Rudin and S. Osher, "Total variation based image resoration with free local constraints," in *IEEE Intl. Conf. Image Process. (ICIP)*, 1994, pp. 31–35.
[29] L. Sendur and I. Selesnick, "Bivariate shrinkage functions for wavelet-based denoising exploiting interscale dependency," *IEEE Trans. Signal Process.*, vol. 50, no. 11, pp. 2744–2756, 2002.
[30] E. Simoncelli, "Bayesian denoising of visual images in the wavelet domain," in *Bayesian inference in wavelet based models*, ser. Lecture Notes in Statistics. Springer, 1999, vol. 141, pp. 291–308.
[31] O. Subakan, B. Jian, B. Vemuri, and E. Vallejos, "Feature preserving image smoothing using a continuous mixture of tensors," in *IEEE Intl. Conf. Computer Vision (ICCV)*, 2007, pp. 1–6.
[32] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 349–366, 2007.
[33] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *IEEE Intl. Conf. Computer Vision (ICCV)*, 1998, pp. 839–846.
[34] D. Tschumperlé and R. Deriche, "Vector-valued image regularization with PDEs : A common framework for different applications," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 27, no. 4, pp. 506–517, 2005.
[35] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear analysis of image ensembles: Tensorfaces," in *Intl. Conf. Pattern Recognition (ICPR)*, 2002, pp. 511–514.
[36] Z. Wang, E. Simoncelli, and A. Bovik, "Multi-scale structural similarity for image quality assessment," in *IEEE Asilomar Conf. Signals, Sys. Comp.*, 2003, pp. 1398–1402.
[37] J. Weickert, *Anisotropic Diffusion in Image Processing*. Stuttgart, Germany: Teubner, 1998.
[38] L. Yaroslavsky, K. Egiazarian, and J. Astola, "Transform domain image restoration methods: review, comparison and interpretation," in *SPIE Proceedings Series, Nonlinear Processing and Pattern Analysis*, 2001, pp. 1–15.
[39] J. Ye, "Generalized low rank approximations of matrices," *Mach. Learning*, vol. 61, no. 1, pp. 167–191, 2005.
[40] Y. You and M. Kaveh, "Fourth order partial differential equations for noise removal," *IEEE Trans. Image Process.*, vol. 9, no. 10, pp. 1723–1730, 2000.
[41] G. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity," in *CoRR*, 2010.
[42] D. Zhang and Z. Wang, "Image information restoration based on long-range correlation," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 12, no. 5, pp. 331–341, 2002.
[43] L. Zhang, W. Dong, D. Zhang, and G. Shi, "Two-stage image denoising by principal component analysis with local pixel grouping," *Pattern Recognition*, vol. 43, no. 4, pp. 1531–1549, 2010.
[44] M. Zhou *et al.*, "Nonparametric Bayesian dictionary learning for analysis of noisy and incomplete images," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 130–144, 2012.
[45] M. Zhou, H. Yang, G. Sapiro, D. B. Dunson, and L. Carin, "Dependent hierarchical beta process for image interpolation and denoising," *Journal of Machine Learning Research - Proceedings Track*, vol. 15, pp. 883–891, 2011.
[46] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *IEEE Intl. Conf. Computer Vision (ICCV)*, 2011.



Ajit Rajwade Ajit Rajwade obtained his bachelors degree in computer engineering from the University of Pune in 2001, the masters degree in computer science from McGill University, Montreal, Canada in 2004 and the Ph.D. degree in computer science from the University of Florida, Gainesville, USA in 2010. He thereafter worked as a postdoctoral associate in the electrical and computer engineering department at Duke University and will be joining Dhirubhai Ambani Institute of Information and Communication Technology (DA-IICT), Gandhinagar, India as an assistant professor in July 2012. His research interests are in image and signal processing (especially restoration, compression and compressive sensing), computer vision, machine learning and computational geometry.

Anand Rangarajan Anand Rangarajan's research interests are at the intersections of machine learning, computer vision, medical imaging and the scientific study of consciousness. He is an Associate Professor in the Department of Computer and Information Science and Engineering, University of Florida. Prior to this, we was an Assistant Professor in the Departments of Diagnostic Radiology and Electrical Engineering, Yale University. He obtained his Ph.D. from the University of Southern California.

Arunava Banerjee Arunava Banerjee is an Associate Professor in the Department of Computer and Information Science and Engineering at the University of Florida, Gainesville. He received his M.S. and Ph.D. degrees in the Department of Computer Science at Rutgers, the State University of New Jersey in 1995 and 2001, respectively. His research interests include computational neuroscience, computer vision, and algorithms.