MULTISCALE MODEL. SIMUL.
Vol. 7, No. 1, pp. 214–241

LEARNING MULTISCALE SPARSE REPRESENTATIONS FOR IMAGE AND VIDEO RESTORATION*

JULIEN MAIRAL†, GUILLERMO SAPIRO‡, AND MICHAEL ELAD§

Abstract. This paper presents a framework for learning multiscale sparse representations of color images and video with overcomplete dictionaries. A single-scale K-SVD algorithm was introduced in [M. Aharon, M. Elad, and A. M. Bruckstein, *IEEE Trans. Signal Process.*, 54 (2006), pp. 4311–4322], formulating sparse dictionary learning for grayscale image representation as an optimization problem, efficiently solved via orthogonal matching pursuit (OMP) and singular value decomposition (SVD). Following this work, we propose a multiscale learned representation, obtained by using an efficient quadtree decomposition of the learned dictionary and overlapping image patches. The proposed framework provides an alternative to predefined dictionaries such as wavelets and is shown to lead to state-of-the-art results in a number of image and video enhancement and restoration applications. This paper describes the proposed framework and accompanies it by numerous examples demonstrating its strength.

Key words. image and video processing, sparsity, dictionary, multiscale representation, denoising, inpainting, interpolation, learning

AMS subject classifications. 49M27, 62H35

DOI. 10.1137/070697653

1. Introduction.

Consider a signal $\mathbf{x} \in \mathbb{R}^n$. We say that it admits a sparse approximation over a dictionary $\mathbf{D} \in \mathbb{R}^{n \times k}$, composed of k columns, referred to as atoms, if one can find a linear combination of "few" atoms from $\mathbf{D}$ that is "close" to the signal $\mathbf{x}$. The so-called *Sparseland* model suggests that such dictionaries exist for various classes of signals and that the sparsity of a signal decomposition is a powerful model in many image and video processing applications [1, 20, 27, 36].

An important assumption, commonly and successfully used in image processing, is the existence of multiscale features in images. Attempting to design the best multiscale dictionary which fulfills a sparsity criterion has been a major challenge in recent years. Such attempts include wavelets [28], curvelets [5, 6], contourlets [15, 16], wedgelets [17], bandlets [24, 29], and steerable wavelets [21, 39]. These methods lead to many effective algorithms in image processing, e.g., image denoising [35]. In this paper, instead of designing the best predefined dictionary for image reconstruction, we propose to learn it from examples. In section 3 we provide further insights into the importance of the multiscale structure in such learned dictionaries.

In [1], the K-SVD algorithm is proposed for learning a single-scale dictionary for sparse representation of grayscale image patches. By means of a sparsity prior on all fixed-sized overlapping patches in the image, the K-SVD is used for removing white Gaussian noise, leading to a very effective algorithm [20]. This has been extended to

*Received by the editors July 18, 2007; accepted for publication (in revised form) December 26, 2007; published electronically April 16, 2008. This work was partially supported by NSF, ONR, NGA, DARPA, ARO, the McKnight Foundation, and Israeli Science Foundation grant 796/05.
http://www.siam.org/journals/mms/7-1/69765.html

†INRIA, WILLOW Project Team, UMR 8548 INRIA/ENS/CNRS, Département d'Informatique, Ecole Normale Supérieure, 45 rue d'Ulm, F-75230 Paris Cedex 05, France (julien.mairal@inria.fr). Part of this work was performed while this author was visiting the University of Minnesota.

‡Corresponding author. Department of Electrical and Computer Engineering, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455 (guille@umn.edu).

§Department of Computer Science, The Technion – Israel Institute of Technology, Haifa 32000, Israel (elad@cs.technion.ac.il).

color images, with state-of-the-art results in denoising, inpainting, and demosaicing applications [27], and more recently to video denoising [36]. In this paper, we extend the basic K-SVD work, providing a framework for learning multiscale and sparse image representation. In addition to the presentation of the new methodology, we apply it to various image and video processing tasks, obtaining results that outperform previous works. Our results for denoising grayscale images outperform, for instance, works such as [10, 19, 20, 22, 35, 38]. The proposed algorithm also competes favorably with the most recent and state-of-the-art result in this field [12], which is based on the nonlocal means algorithm [4]. Our framework for color image denoising also competes favorably with the best known algorithm in this field [11], and the results for the other presented applications such as color video denoising and inpainting of small holes in image and video are also among the best we are aware of.

The task of learning a multiscale dictionary has been addressed in [33] in the general context of sparsifying image content. Our approach differs from theirs in the numerical treatment, the multiscale structure formed, and the way the dictionaries found are deployed for denoising. These differences may explain the significantly superior performance we obtain. Other results on learning single-scale image dictionaries include, for example, [37, 38, 43].

While sparsity and multiscale techniques have been both widely applied and studied independently in signal and image processing, this paper is the first successful attempt to combine these two concepts in a learning fashion. Our work presents both algorithmic considerations and accompanying simulation results in several image processing tasks. Those are shown to outperform or equal the state-of-the-art results, thereby confirming that multiscale is crucial in these fields. Furthermore, the proposed framework is compatible with all the K-SVD prior work [20, 27, 36], which implies that the same extension could be found beneficial to many other image and video restoration problems.

The structure of this paper is as follows: In section 2 we briefly review relevant background, which includes the original K-SVD denoising algorithm [1], the extensions to color image denoising, nonhomogeneous noise, and inpainting [27], and the K-SVD for denoising videos [36]. Section 3 is devoted to the presentation of our novel proposed multiscale framework. This section is followed by a section that introduces further important algorithmic improvements to the original single-scale K-SVD. Section 5 presents some applications of the multiscale K-SVD, covering grayscale and color image denoising and image inpainting. In section 6 we further extend the framework and show the performance for video processing. Section 7 concludes this paper with a brief description of its contributions and some open questions for future work.

2. The single-scale K-SVD. The single-scale K-SVD has already shown very good performance for grayscale image denoising [19, 20], color image denoising [27], inpainting and demosaicing [27], and video denoising [36]. In this section, we briefly review these algorithms.

2.1. Grayscale image denoising algorithm. We start by briefly reviewing the main ideas of the K-SVD framework for sparse image representation and denoising. The reader is referred to [19, 20] for additional details.

Let $\mathbf{x}_0$ be a clean image and $\mathbf{y} = \mathbf{x}_0 + \mathbf{w}$ its noisy version with $\mathbf{w}$ being an additive zero-mean white Gaussian noise with a known standard deviation σ. The K-SVD bases its denoising approach on a local and shift-invariant sparsity prior, imposed on small overlapping patches in the image. This way, the dictionary learning task bypasses the prohibitive computation and memory requirements involved in a global

handling of complete images (e.g., 512×512 pixels). The parameter n, denoting the size of such patches, is fixed a priori (e.g., $n = 64$ for an 8×8 patch), and the algorithm aims at finding a sparse approximation of every $\sqrt{n} \times \sqrt{n}$ overlapping patch extracted from $\mathbf{y}$ (e.g., $(512 - 8 + 1)^2 = 255{,}025$ different patches for a 512×512 image). This representation is done using an adapted dictionary $\mathbf{D}$, learned for this set of patches. The patches' approximations are averaged to obtain the reconstructed image. This algorithm (shown in Figure 1) can be described as the minimization of an energy:

$$\left\{\{\hat{\alpha_{ij}}\}_{ij}, \hat{\mathbf{D}}, \hat{\mathbf{x}}\right\} = \underset{\mathbf{D}, \{\alpha_{ij}\}_{ij}, \mathbf{x}}{\arg\min} \lambda \|\mathbf{x} - \mathbf{y}\|_2^2 + \sum_{ij} \mu_{ij} \|\alpha_{ij}\|_0 + \sum_{ij} \|\mathbf{D}\alpha_{ij} - \mathbf{R}_{ij}\mathbf{x}\|_2^2. \tag{2.1}$$

In this equation, $\hat{\mathbf{x}}$ is the estimator of $\mathbf{x}_0$, and the dictionary $\hat{\mathbf{D}} \in \mathbb{R}^{n \times k}$ is an estimator of the optimal dictionary, which leads to the sparsest representation of the patches in the recovered image. The indices $[i, j]$ mark the location of the patch in the image (representing its top-left corner). The vector $\hat{\alpha}_{ij} \in \mathbb{R}^k$ is the sparse representation for the $[i, j]$th patch in $\hat{\mathbf{x}}$ using the dictionary $\hat{\mathbf{D}}$, such that $\mathbf{D}\alpha_{ij}$ is a linear combination of columns of $\mathbf{D}$ that is close to this $[i, j]$th patch. The notation $\|.\|_0$ is the ℓ^0 quasi norm, a sparsity measure, which counts the number of nonzero elements in a vector. The operator $\mathbf{R}_{ij}$ is a binary matrix that extracts a square patch of size $\sqrt{n} \times \sqrt{n}$ from location $[i, j]$ in the image, forming the output patch as a column vector. The main steps of the algorithm are (refer to Figure 1) the following:

- *Sparse Coding step*: This is performed with an orthogonal matching pursuit (OMP) [13, 14, 30], a greedy algorithm that proves to be very efficient for diverse approximation problems [18, 41, 42]. The approximation stops when the residual reaches a sphere of radius $\sqrt{n}C\sigma$ representing the probability distribution of the noise (C being a constant). More on this can be found in [27].

  Note that this algorithm provides only an approximated solution of (2.2), as the nonconvexity of the functional we are considering makes this problem difficult (in fact, NP-Hard) in general. A well-known and alternative approach is the basis pursuit [7], which suggests a convexification of the problem by using the ℓ^1-norm instead of ℓ^0. Nevertheless, when working with small patches, greedy algorithms prove to be far more efficient.
- *Dictionary Update*: This is a sequence of rank-one approximation problems that update both the dictionary atoms and the sparse representations that use it, one at a time.
- *Reconstruction*: The last step is a simple averaging between the patches' approximations and the noisy image. The denoised image is $\hat{x}$. Equation (2.4) emerges directly from the energy minimization in (2.1).

This algorithm provided state-of-the-art results at the time of its publication. A key issue in this work is the data to train on. One possibility is to learn a dictionary from a large database of images (the so-called *global* approach), thereby exploiting intrinsic information about natural images. The alternative, that proved to be more effective, is to learn a dictionary from all the overlapping patches of the given noisy image (the *adaptive* approach), adapting to the specific image content. As in [27], we typically learn off-line a global dictionary and then use it as an initialization for the iterative *adaptive* approach presented in Figure 1.

An extensive study presented in [19, 20] led to the following choice of parameters in the algorithm: $k = 256$, $n = 8 \times 8$, $C = 1.15$, $\lambda = \frac{30}{\sigma}$, and $J = 10$ (number of iterations). These were found to be good parameters for this algorithm, with a good compromise between speed and quality.

Parameters: λ (Lagrange multiplier); C (noise gain); J (number of iterations); k (number of atoms); n (size of the patches).
Initialization: Set $\hat{\mathbf{x}} = \mathbf{y}$; Initialize $\hat{\mathbf{D}} = (\hat{\mathbf{d}}_l \in \mathbb{R}^{n\times 1})_{l\in 1\dots k}$.
Loop: Repeat J times

- *Sparse Coding*: Fix $\hat{\mathbf{D}}$ and use OMP to compute coefficients $\hat{\alpha}_{ij} \in \mathbb{R}^{k\times 1}$ for each patch by solving

$$\forall ij \quad \hat{\alpha}_{ij} = \arg\min_{\alpha} \|\alpha\|_0 \text{ subject to } \|\mathbf{R}_{ij}\hat{\mathbf{x}} - \hat{\mathbf{D}}\alpha\|_2^2 \le n(C\sigma)^2. \tag{2.2}$$

- *Dictionary Update*: Fix all $\hat{\alpha}_{ij}$, and for each atom $\hat{\mathbf{d}}_l$, $l \in 1, 2, \dots, k$ in $\hat{\mathbf{D}}$,
  - Select the patches ω_l that use this atom, $\omega_l := \{[i,j] \mid \hat{\alpha}_{ij}(l) \neq 0\}$.
  - For each patch $[i,j] \in \omega_l$, compute its residual without the contribution of the atom $\hat{\mathbf{d}}_l$, i.e., $\mathbf{e}_{ij}^l = \mathbf{R}_{ij}\hat{\mathbf{x}} - \hat{\mathbf{D}}\hat{\alpha}_{ij} + \hat{\mathbf{d}}_l\hat{\alpha}_{ij}(l)$.
  - Set $\mathbf{E}_l = (\mathbf{e}_{ij}^l)_{[i,j]\in\omega_l} \in \mathbb{R}^{n\times|\omega_l|}$ as the matrix whose columns are the $\mathbf{e}_{ij}^l$, and $\hat{\alpha}^l = (\hat{\alpha}_{ij}(l))_{[i,j]\in\omega_l} \in \mathbb{R}^{|\omega_l|}$.
  - Update $\hat{\mathbf{d}}_l$ and the $\hat{\alpha}_{ij}(l)$ by minimizing

$$(\hat{\mathbf{d}}_l, \hat{\alpha}^l) = \arg\min_{\alpha, \|\mathbf{d}\|_2=1} \|\mathbf{E}_l - \mathbf{d}\alpha^T\|_F^2. \tag{2.3}$$

  This rank-one approximation is performed by a truncated SVD of $\mathbf{E}_l$. Here $\mathbf{E}_l - \mathbf{d}\alpha^T$ represents the residual error of the patches from ω_l if we replace $\hat{\mathbf{d}}_l\hat{\alpha}^{lT}$ by $\mathbf{d}\alpha^T$ in their decompositions.

Reconstruction: Perform a weighted average

$$\hat{\mathbf{x}} = \Big(\lambda\mathbf{I} + \sum_{ij}\mathbf{R}_{ij}^T\mathbf{R}_{ij}\Big)^{-1}\Big(\lambda\mathbf{y} + \sum_{ij}\mathbf{R}_{ij}^T\hat{\mathbf{D}}\hat{\alpha}_{ij}\Big), \tag{2.4}$$

which is the solution of the optimization problem of (2.1) with respect to $\mathbf{x}$ when $\mathbf{D}$ and α are assumed fixed. Note that the inversion described above is trivial, as the matrix involved is diagonal. Thus, this equation is in effect a simple weighted averaging of image patches, and the division plays a role of normalization.

FIG. 1. *The single-scale K-SVD-based grayscale image denoising algorithm.*

2.2. Extension to color image denoising. In [27], we have shown that we can apply the K-SVD for color images by denoising each RGB patch directly as a long concatenated RGB vector. Like in the grayscale denoising algorithm, each RGB channel is written as a column vector, and then the obtained RGB vectors are concatenated. Within this framework, the algorithm is able to learn the correlation between the RGB channels and exploit it effectively. This was shown to provide improved results over the denoising of each color channel separately.

Nevertheless, we observed on some images a color bias, especially when we used the *global* dictionaries. Our study has led us to the conclusion that this phenomenon happened because the dictionary redundancy was too small to represent the diversity of colors among natural images. Therefore, we introduced a different metric within the OMP that maintains the average color of the original image.[1] With this intention,

[1]The OMP selects in a greedy fashion the "closest" atom at each iteration, and thereby having a metric is an intrinsic component of this algorithm.

we introduced a new parameter γ, which de facto defines a new metric during the OMP, taking into account the average color of the patches. Additional details and numerous examples are given in [27], showing that the proposed framework leads to state-of-the-art results, which are further improved with the multiscale approach and additional algorithmic improvements to be exposed here.

2.3. Handling nonhomogeneous noise and applications. Handling nonhomogenous noise is very important, as nonuniform noise across color channels is very common in digital cameras. In [27], we have presented a variation of the K-SVD, which permits us to address this issue. Within the limits of this model, we were able to fill in relatively small holes in images, and we presented state-of-the-art results for image demosaicing, outperforming every specialized interpolation-based method, such as [8, 23, 25, 32].

Consider the case where $\mathbf{w}$ is a white Gaussian noise with a different standard deviation $\sigma_p > 0$ at each location p. Assuming these standard deviations are known, we introduce a vector β composed of weights for each location,

$$\beta_p = \frac{\min_{p' \in \text{Image}} \sigma_{p'}}{\sigma_p}. \tag{2.5}$$

This leads us to define a weighted K-SVD algorithm based on a different metric for each patch. We denote by $\otimes$ an elementwise multiplication between two vectors, used to apply the above vector β as a "mask." We aim at solving the following problem, which replaces (2.1):

$$\begin{aligned}\left\{\hat{\alpha_{ij}}, \hat{\mathbf{D}}, \hat{\mathbf{x}}\right\} = \underset{\mathbf{D},\alpha_{ij},\mathbf{x}}{\arg\min}\, & \lambda\|\beta \otimes (\mathbf{x}-\mathbf{y})\|_2^2 + \sum_{ij} \mu_{ij}\|\alpha_{ij}\|_0 \\ & + \sum_{ij} \|(\mathbf{R}_{ij}\beta) \otimes (\mathbf{D}\alpha_{ij} - \mathbf{R}_{ij}\mathbf{x})\|_2^2.\end{aligned} \tag{2.6}$$

There are two main modifications in the minimization of this energy. First, the *Sparse Coding* step takes the matrix β into account by using a different metric within the OMP. Second, the *Dictionary Update* variation is more delicate, and (2.3) is replaced by

$$(\hat{\mathbf{d}}_l, \hat{\alpha}^l) = \underset{\alpha, \|\mathbf{d}\|_2=1}{\arg\min} \|\beta^l \otimes (\mathbf{E}_l - \mathbf{d}\alpha^T)\|_F^2, \tag{2.7}$$

where β^l is the matrix whose size is the same as $\mathbf{E}_l$ and where each column corresponding to an index $[i,j]$ is $\mathbf{R}_{ij}\beta$. This problem is known as a weighted rank-one approximation matrix (see [40]).

Inpainting (see, e.g., [2, 9]) consists of filling in holes in images. Within the limits of our model, it becomes possible to address a particular case of inpainting. By considering small random holes as areas with infinite power noise, one can design a mask β with zero values for the missing pixels and apply it as a nonhomogeneous denoising problem. This approach proves to be very efficient. This inpainting case could also be considered as a specific case of interpolation. The mathematical formulation from (2.6) remains the same, but some values from the matrix β are just zeros. Details about this method are provided in [27], together with a discussion on how to handle the demosaicing problem that has a fixed and periodic pattern of missing values.

2.4. Video denoising algorithm. The video extension of the K-SVD has been developed and described in [36]. This work exploits the temporal correlation in video signals to increase the denoising performance of the algorithm, providing state-of-the-art results for removing white Gaussian noise. As explained in [36], applying the previously described K-SVD on the whole video volume as one signal is problematic due to the rapid changes in the video content, implying that one dictionary cannot fit well to the whole data. At the other extreme, an alternative method that applies the single-image K-SVD denoising algorithm to the image sequence one frame at a time is also expected to perform poorly, since we do not exploit the temporal correlation. Therefore, a different approach is proposed in [36], based on the following concepts:

- *Three-dimensional (3D) atoms*: Each frame should be denoised separately, but patches are constructed from more than one frame, grasping both spatial and temporal behaviors.
- *Dictionary propagation*: The initial dictionary for each frame is the one trained for the previous one. Fewer training iterations are thus required.
- *Extended temporal set of patches*: Patches in neighboring frames are also used for dictionary training and image cleaning for each frame.

Translating these three concepts into a mathematical formulation leads to the following modified version of (2.1):

(2.8)

$$\forall t \in 1 \dots T, \quad \left\{\hat{\alpha}_{ij\tau}, \hat{\mathbf{D}}_t, \hat{\mathbf{x}}_t\right\} = \underset{\mathbf{D}_t, \alpha_{ij\tau}, \mathbf{x}_t}{\arg\min} \lambda \|\mathbf{x}_t - \mathbf{y}_t\|_2^2 + \sum_{ij} \sum_{\tau=t-\Delta_t}^{t+\Delta_t} \mu_{ij\tau} \|\alpha_{ij\tau}\|_0 + \|\mathbf{D}_t \alpha_{ij\tau} - \mathbf{R}_{ij\tau}\mathbf{x}\|_2^2.$$

In this formulation, $\mathbf{x}$ is a video composed of T frames, and $\mathbf{y}_t$ denotes the tth frame of a noisy video $\mathbf{y}$. $\hat{\mathbf{x}}_t$ denotes the estimated clean tth frame. For each frame, the learning process is performed on an extended temporal set of patches $[t-\Delta_t; t+\Delta_t]$, as explained before, with Δ_t typically equal to 1 or 2. Then, 3D patches are used, which are constructed with patches at the same spatial location from more than one frame, that are concatenated one after another into a column vector. In [36], patches of size $n = 8 \times 8 \times 5$ were built with five adjacent frames. To that effect, $\mathbf{R}_{ij\tau}$ is again a binary matrix that extracts the 3D patch with the spatial location $[i, j]$ and with a temporal location centered at the time τ. Then, $\hat{\mathbf{D}}_t \in \mathbb{R}^{n \times k}$ is the adapted dictionary for the time t, and $\alpha_{ij\tau}$ is the representation of the 3D patch $\mathbf{R}_{ij\tau}\mathbf{x}$.

2.5. Brief summary. Having concluded the brief background presentation, we proceed to present a multiscale framework that permits us to improve all of the above-mentioned algorithms. We should note that the approach we are about to present is one among several possibilities for introducing multiscale analysis into the dictionary learning and sparse image/video representation framework. This means that further work could (and should) be done to explore alternative possibilities, in spite of the fact that the approach here presented already leads to state-of-the-art results.

3. Learned multiscale sparse representation. Since it is well accepted that image information spreads across multiple scales, designing a K-SVD type of algorithm that is able to adapt and simultaneously capture information at multiple scales is the main goal of this paper. This section discusses the main principles of our proposed approach.

The original, single-scale, K-SVD has proven to work very well with small image patches ($n = 8 \times 8$). Nevertheless, there is a strong incentive for multiscale extension of this method for various reasons. Intuitively speaking at first, in the single-scale denoising algorithm, every pixel filtered is influenced by a limited and quite small group of its local neighbors. The size of the relevant neighborhood is dictated by the patch size—17×17 pixels, with 8 pixels in each direction, in the case of [19, 20]. A multiscale treatment can help the pixels see "beyond the horizon" and thus get better treatment. As a simple example, if a pixel belongs to a wide segment of constant gray value, averaging all the pixels in this segment removes noise very effectively. When working with a limited neighborhood, as in the single-scale method, some of this power is lost. The introduction of multiscale dictionaries leads to an effective growth of the neighborhoods and thus the expected gain.

Put more generally, as some image structures are bigger than 8 pixels, we should expect a better performance from such a method that handles larger patches. Indeed, we have observed that different images may prefer different patch sizes (both globally and locally) for optimal performance, and thus having a simultaneous multiscale dictionary avoids this difficult task of selecting a patch size in advance. As we demonstrate in this paper, learning multiscale dictionaries leads to better restoration results, due to these reasons.

One simple and naive strategy to introduce multiscale analysis consists of using large patches with a high redundancy factor ($\frac{k}{n}$) and hoping for the appearance of intrinsic multiple scales among the learned dictionary's atoms. However, we have observed no significant differences between the results with the parameters $\{n = 8 \times 8,\, k = 256\}$ compared to $\{n = 16 \times 16,\, k = 1024\}$. Sometimes we have even observed oversmoothing artifacts when using the bigger patches. One explanation for the "failure" of this direct approach is that the K-SVD may be trapped in a local minimum, learning only the scale that corresponds to the size of the patches, avoiding the true multiscale result. In that respect, the use of small patches for learning fine details is unavoidable. By explicitly imposing such multiscale structure, we may help in this regard. This leads us naturally to the proposed framework. We note again that although we present a multiscale extension of the K-SVD for image and video enhancement, learning multiscale dictionaries is important, per se, also for other applications such as classification.

3.1. The basic model. In our proposed multiscale framework, we focus on the use of different sizes of atoms simultaneously. Considering the design of a patch-based representation and a denoising framework, we put forward a simple quadtree model of large patches, as shown on Figure 2. This is a classical data structure, also used, for example, in wedgelets [17]. A fixed number of scales, N, is chosen such that it corresponds to N different sizes of atoms. A large "root" patch of size n pixels is divided along the tree to subpatches of sizes $n_s = \frac{n}{4^s}$, where $s = 0 \dots N-1$ is the depth in the tree. Then, one different dictionary $\mathbf{D}_s \in \mathbb{R}^{n_s \times k_s}$ composed of k_s atoms of size n_s is learned and used per scale s.

The overall idea of the multiscale algorithm we propose stays as close as possible to the original K-SVD algorithm, as depicted in Figure 1, with an attempt to exploit the several existing scales jointly and wisely. More specifically, we aim at solving the same energy minimization problem of (2.1), with a multiscale structure embedded within the overall dictionary $\mathbf{D} \in \mathbb{R}^{n \times k}$. This overall dictionary is a joint one, composed of all the atoms of all the dictionaries $\mathbf{D}_s$ located at every possible position in the quadtree.

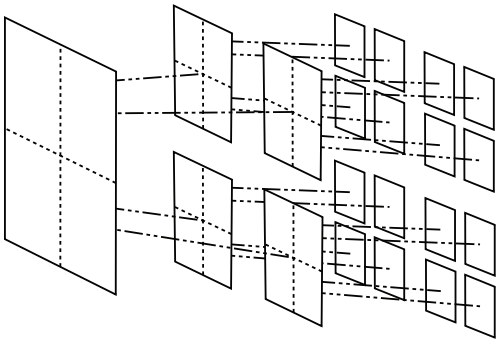


FIG. 2. *Quadtree model selected for the proposed multiscale framework.*

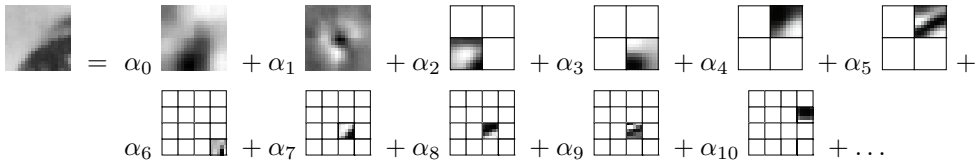


FIG. 3. *Possible decomposition of a* 20×20 *patch with a* 3*-scale dictionary.*

For the scale s, there exist 4^s such positions. This makes a total of $k = \sum_{s=0}^{N-1} 4^s k_s$ atoms in $\mathbf{D}$. This is illustrated in Figure 3, where an example of a possible multiscale decomposition is presented. One can see in this figure how the dictionary $\mathbf{D}$ has been built: The atoms of $\mathbf{D}_0$ of size n (for instance the ones associated with α_0 and α_1 on this picture) are atoms of $\mathbf{D}$. Then, the atoms of $\mathbf{D}_1$ of size $\frac{n}{4}$ are embedded into bigger atoms of size n with zero padding at four possible positions (see the atoms associated with $\alpha_2, \alpha_3, \alpha_4, \alpha_5$ in the figure). Then, the same idea applies to $\mathbf{D}_2$ and so on.

Addressing the minimization problem of (2.1) with a multiscale dictionary $\mathbf{D}$ implies the need to consider equally the atoms from the different scales. Therefore, we choose to normalize all the atoms of the dictionaries to unit norm. This policy is important during the *Sparse Coding* step and proves to provide better results than choosing a different norm per scale.

The original K-SVD exploits the overlapping/shift-invariant treatment of the patches' representation, which has been found to be critical for denoising [19, 20, 27, 38]. Exploiting this treatment at each scale of our multiscale model is therefore important. Due to the quadtree, however, each subpatch inside one "root" patch is restricted to 4^s different shifts at the scale s. Nevertheless, as every pixel in the original image is considered as the center of a root patch, shift invariance is employed in this scheme as well.

Integrating the multiscale structure requires the following key modifications to the basic algorithm:

- *Sparse Coding*: This remains unchanged if we introduce some simple notation. In (2.2), assume that $\mathbf{R}_{ij}$ remains the matrix that extracts the patch of size $n_0 = n$ with coordinates $[i, j]$. The multiscale dictionary $\hat{\mathbf{D}}$ is the joint one, composed of all the atoms of all the dictionaries $\hat{\mathbf{D}}_s = (\hat{\mathbf{d}}_{sl} \in \mathbb{R}^{n_s})_{l \in 1 \ldots k_s}$ located at every possible position in the quadtree structure. For the scale s, we denote their index as p among the 4^s possible shifts. The OMP is im-

plemented efficiently using the modified Gram–Schmidt algorithm [3]. During each selection procedure of the OMP, a scale s, a position p, and an atom $\hat{\mathbf{d}}_{sl}$ are chosen. For each "root" patch, this step can be achieved in $\mathcal{O}((\sum_{s=0}^{N-1} k_s)n\|\hat{\alpha}\|_0)$ operations.

- *Dictionary Update*: This step is slightly changed, as we update each atom $\hat{\mathbf{d}}_{sl}$ $(1 \le l \le k_s)$ in each scale (from $s = 0$ to $s = N-1$):
  - Select the set of subpatches from the scale s that use the lth atom,

$$\omega_{sl} := \{[i,j,s,p] \mid \hat{\alpha}_{ij}(s,l,p) \neq 0\},$$

    where $[i,j,s,p]$ denotes the subpatch at the scale s and position p from the patch $[i,j]$, and $\hat{\alpha}_{ij}(s,l,p)$ is the coefficient corresponding to this subpatch and the atom $\hat{\mathbf{d}}_{sl}$.
  - For each subpatch $[i,j,s,p] \in \omega_{sl}$, compute

$$e^l_{ijsp} = \mathbf{T}_{sp}(\mathbf{R}_{ij}\hat{\mathbf{x}} - \hat{\mathbf{D}}\hat{\alpha}_{ij}) + \hat{\mathbf{d}}_{sl}\hat{\alpha}_{ij}(s,l,p),$$

    where $\mathbf{T}_{sp} \in \{0,1\}^{n_s \times n_0}$ is a binary matrix which extracts the subpatch $[i,j,s,p]$ from a patch $[i,j]$.
  - Set $\mathbf{E}_{sl} \in \mathbb{R}^{n_s \times |\omega_{sl}|}$ as the matrix whose columns are the $\mathbf{e}^l_{ijsp}$ and $\hat{\alpha}^{sl} \in \mathbb{R}^{|\omega_{sl}|}$ as the vector whose elements are the $\hat{\alpha}_{ij}(s,l,p)$.
  - Update $\hat{\mathbf{d}}_{sl} \in \mathbb{R}^{n_s}$ and the $\hat{\alpha}_{ij}(s,l,p)$ using a SVD as before:

$$(\hat{\mathbf{d}}_{sl}, \hat{\alpha}^{sl}) = \underset{\alpha, \|\mathbf{d}\|_2=1}{\arg\min} \|\mathbf{E}_{sl} - \mathbf{d}\alpha^T\|_F^2. \tag{3.1}$$

- *Reconstruction*: This remains the same as in (2.4), while using the new notation just introduced. This is again the solution of the optimization problem from (2.1) with respect to $\mathbf{x}$ when the multiscale dictionary $\mathbf{D}$ and the coefficients α are fixed. Note that each patch is reconstructed from multiple scales, and since a pixel belongs to multiple (overlapping) patches, it is reconstructed with multiple scales and at multiple positions.

The computational time of the *Sparse Coding* stage is paramount compared to the *Dictionary Update* and the *Reconstruction* stages. The total complexity is therefore $\mathcal{O}(k_a nLJM)$, where L is the average sparsity factor (number of coefficients obtained in the decomposition), M is the number of patches processed, and $k_a = \sum_{i=0}^{N-1} k_s/4^s$ is the *effective* overall number of atoms used, taking into account the fact that some atoms are mostly zeros (due to their small size relative to the root patch).

3.2. Extension to various image and video enhancement tasks. We now show how this framework is extended to different applications.

- *Color image denoising*: As in the single-scale algorithm, extending the color framework to the multiscale version requires us to consider a concatenated RGB vector. Then, the same quadtree structure and the same scheme are applied. The only difference with respect to the grayscale algorithm is the use of the parameter γ, which was introduced to solve the bias-color problem, described in [27]. We recall that this mechanism enforces the average color of the patches during the OMP, thereby creating a new metric. In the multiscale case, we cannot enforce the average color of a patch, since it would introduce a bias for the subpatches in the quadtree. Therefore, this should be done only for the smallest subpatches. For instance, assume we have $N = 3$ scales and

a patch of size $n = 20 \times 20 \times 3$ (3 being the number of color channels). Then, we enforce the average color of each $5 \times 5 \times 3$ subpatch within the dictionaries and patches.

- *Nonhomogeneous denoising and inpainting*: In order to extend the nonhomogeneous denoising and inpainting algorithms to multiscale, one should first notice that for a patch of index $[i, j]$, the matrix $\mathbf{R}_{ij}\beta$ that we introduced in section 2.3 can be used directly during the *Sparse Coding* stage, since it operates as a single-scale one with a large dictionary. Then, the *Dictionary Update* step requires a decomposition of $\mathbf{R}_{ij}\beta$ in a quadtree structure, providing a set of matrices $\mathbf{T}_{sp}\mathbf{R}_{ij}\beta$ for each scale s and position p within the scale. Equation (3.1) has to be adapted to match (2.7):

$$(\hat{\mathbf{d}}_{sl}, \hat{\alpha}^{sl}) = \underset{\alpha, \|\mathbf{d}\|_2=1}{\arg\min} \|\beta_{sl} \otimes (\mathbf{E}_{sl} - \mathbf{d}\alpha^T)\|_F^2, \tag{3.2}$$

  where β_{sl} is a matrix whose size is the same as $\mathbf{E}_{sl}$, and where each column corresponding to an index $[i, j]$ and position p within the scale s is $\mathbf{T}_{sp}\mathbf{R}_{ij}\beta$.
- *Video denoising*: Both color images and image sequences (video) are using patches and atoms with many channels: the RGB layers for the color processing, and temporal frames for the video processing (four-dimensional atoms). Concatenating the color channels and different frames in single vectors permits us to address a color video denoising problem by minimizing the same energy as in (2.8). To do so, each RGB channel from each patch of the considered frames is written as column a vector, and those are concatenated together. For the multiscale extension for denoising image sequences, one can regard the K-SVD for video as a successive K-SVD employed to multichannel images. It consists of putting the quadtree structure on the considered channel images, and considering the learning of the dictionaries at each scale, by proceeding exactly as for the single-image case. Here extending the grayscale video denoising to color consists of handling concatenated RGB vectors. Interestingly, we found that when handling a video, we can omit the use of a warped metric that uses the parameter γ.
- *Nonhomogeneous denoising and inpainting for video*: Using the same matrix β_t introduced for the weighted K-SVD algorithm for the frame at time t, the video inpainting problem can be treated as suffering from nonhomogeneous noise. This leads to the following energy minimization formulation:

$$\begin{aligned}\forall t \in 1 \dots T, \quad \left\{\hat{\alpha}_{ij\tau}, \hat{\mathbf{D}}_t, \hat{\mathbf{x}}_t\right\} = \underset{\mathbf{D}_t, \alpha_{ij\tau}, \mathbf{x}_t}{\arg\min} \; & \lambda\|\beta_t \otimes (\mathbf{x}_t - \mathbf{y}_t)\|_2^2 \\ & + \sum_{ij} \sum_{\tau=t-\Delta_t}^{t+\Delta_t} \mu_{ij\tau}\|\alpha_{ij\tau}\|_0 + \|(\mathbf{R}_{ij}\beta_\tau) \otimes (\mathbf{D}_t\alpha_{ij\tau} - \mathbf{R}_{ij\tau}\mathbf{x})\|_2^2.\end{aligned}$$

  Handling the inpainting problem for video via an extension of the previous algorithm is possible since we can regard the processing per frame as separate, although involving adjacent frames. This permits us to use the matrix β exactly the same way as we already did for single images. This handles inpainting of relatively small holes. For addressing the general video inpainting problem, the reader should refer to [34, 44]. Due to the multiscale nature of the proposed scheme, somewhat larger holes can be treated successfully, compared to the single-scale algorithm.

4. Additional algorithmic improvements. We now introduce several important additional refinements, which further improve the results without increasing the computational cost.

4.1. Treatment of the DC. For the grayscale K-SVD we find it useful to force the presence of a constant (DC) atom in each dictionary (an atom that has the same value for every pixel) and give it a preference by multiplying this atom by a constant η (2.5 for example) during the selection procedure of the OMP (refer to [14]). This makes sense since a constant atom does not introduce any noise in the reconstruction. For the color extension, we introduced one constant atom per channel, i.e., one red, one green, and one blue atom, and for the video K-SVD algorithm, one constant atom (or constant per channel in the color case) per frame in the 3D patches.

4.2. The stopping rule. As discussed in [27], the stopping criterion during the OMP is based on the norm of an n-dimensional Gaussian vector which is distributed following the generalized Rayleigh law. This means that one has to stop the approximation when the residual reaches a fuzzy sphere. According to this law, the bigger n is, the thinner the sphere is, and the more accurate the stopping criterion $\sqrt{n}C(n)\sigma$ becomes (C is a parameter that depends on n). Thus one asset of increasing n through our multiscale scheme is to provide an improved stopping criterion.

It is actually not necessary to perform a complete multiscale algorithm to take advantage of this property. During the *Sparse Coding* stage, instead of processing each patch separately, one can choose to process some adjacent sets of nonoverlapping patches simultaneously and consider them as a larger patch (and therefore associated with a better stopping criterion). In practice, we choose m adjacent and nonoverlapping patches of size n, and we first process them independently using their own stopping criterion $\sqrt{n}C(n)\sigma$. Then, as long as the cumulative error of the m patches is larger than the (better) stopping criterion $\sqrt{nm}C(nm)\sigma$, we refine the approximation by progressively adding terms, one at a time, to the sparse expansion of the worse of the m patches. Then we consider a new set of m patches and continue the sparse approximation. This does not increase the complexity of the algorithm and provides noticeable improvement.

4.3. Reduced training set. In Figure 1, the *Sparse Coding* and *Dictionary Update* stages are performed over the full set of overlapping patches. Performing these steps over a partial and random subset of these patches during all the iterations (apart from the last one) leads to a substantial reduction in the computational time and the memory requirements. This idea was proposed and used successfully in [36], and we found it to be useful in our applications as well.

4.4. A block denoising variation. Analyzing the performance of the K-SVD-based image denoising algorithm raises some interesting questions. Let us consider an "homogeneous" image that can be represented reliably using one dictionary $\mathbf{D}_{\text{opt}}$. Then, the bigger the image is, the better the denoising results are, since we get more examples to train on, and thus the K-SVD is more likely to find $\mathbf{D}_{\text{opt}}$. When the image has a wide variability of content, one could try to use a larger dictionary (with more redundancy), but our extensive experiments show that this does not improve the results significantly. This might be explained by the increased risk of getting stuck in a local minimum in the K-SVD training, or perhaps the reason is the reduced performance of the OMP in such cases.

Nevertheless, a way to address the above-mentioned problem is to handle different zones of the images separately. In this paper, we choose to define a block denoising

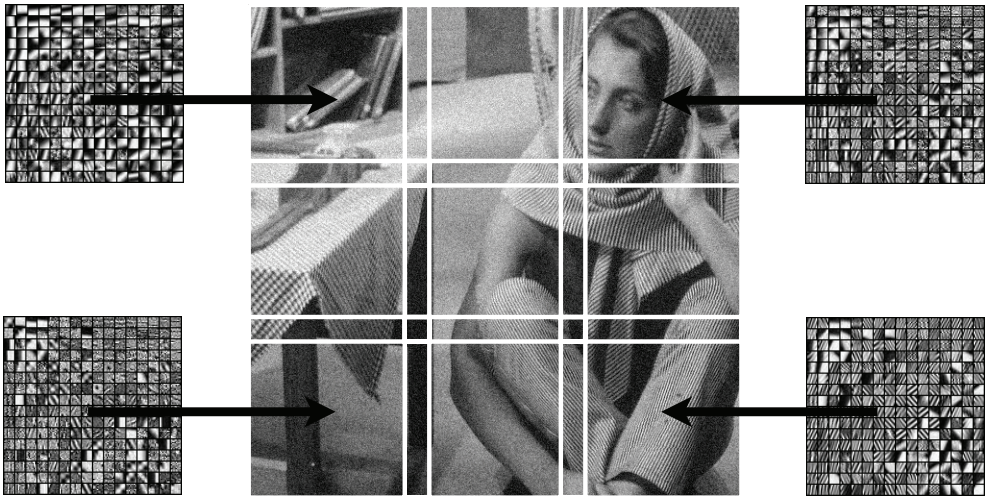


FIG. 4. *Illustration of the block denoising algorithm. Four dictionaries are shown. Those are trained on four different blocks (out of the nine overall that we have) from a noisy* barbara *with* $\sigma = 15$. *As can be seen, each dictionary is more adapted to the content it is serving. E.g., the top-left dictionary does not contain textured atoms, as those are not needed in this part of the image. On the other hand, the bottom-right dictionary is practically loaded with such textured atoms, as those are crucial and dominant in that part of the image.*

algorithm, but future work may combine our denoising algorithm with a segmentation of the input image. More precisely, we consider blocks of the same size from one image, with a small overlap of the same width as the patches' size, as illustrated in Figure 4.[2] Given a judiciously adapted block size, this approach introduces two advantages: First, the performance in terms of denoising results is better since the dictionaries are better adapted to their own regions, as can be noticed in Figure 4. Second, this approach has the same computational complexity with a lower memory usage, since both are linear in the number of denoised pixels.[3]

A natural question raised is whether there exists a generic optimal size to choose for these blocks. Answering this requires taking into account several considerations:

- The bigger the blocks are, the more information from the image is taken into account each time the K-SVD is performed. On the down side, bigger blocks imply more diversity of the image content and less flexibility of the dictionary to handle this content well.
- The smaller a block is, the better the K-SVD can adapt the dictionary to it. However, smaller blocks imply a risk of overfitting, where the dictionary is learning the given examples and absorbs some of the noise in them as well.
- The bigger σ (the noise power) is, the more patches (and thus bigger blocks) are required to make the K-SVD robust to the noise.

As expected, our experiments show that the best size for the block denoising algorithm is linked to the amount of noise in the image. The smaller the noise variance is, the smaller the average best size to get the best denoising performance.

[2]Note that since pixels are recovered as a linear combination of overlapping patches, this will attenuate the common artifacts at the boundary of the segments.

[3]We neglect the small increase in the number of pixels due to the small overlapping of the blocks.

TABLE 1

Comparison of PSNR results of several denoising algorithms. Each cell is divided into four parts. The top-left cell shows the results from the original K-SVD [20], and the top-right cell presents the most recent state-of-the-art results [12]. The bottom-left cell is devoted to our results for $N = 1$ scale and the bottom-right cell to $N = 2$ scales. Each time the best results are in bold.

σ	house		peppers		cameraman		lena		barbara	
5	39.37	39.82	37.78	38.09	37.87	38.26	38.60	38.73	38.08	38.30
	39.81	**39.92**	38.07	**38.20**	38.12	**38.32**	38.72	**38.78**	**38.34**	38.32
10	35.98	36.68	34.28	**34.68**	33.73	34.07	35.47	**35.90**	34.42	**34.96**
	36.38	**36.75**	34.58	34.62	34.01	**34.17**	35.75	35.84	34.90	34.86
15	34.32	34.97	32.22	**32.70**	31.42	**31.83**	33.70	**34.27**	32.37	**33.08**
	34.68	**35.00**	32.53	32.47	31.68	31.72	34.00	34.14	32.82	32.96
20	33.20	**33.79**	30.82	**31.33**	29.91	**30.42**	32.38	**33.01**	30.83	**31.77**
	33.51	33.75	31.15	31.08	30.32	30.37	32.68	32.88	31.37	31.53
25	32.15	**32.87**	29.73	**30.19**	28.85	**29.40**	31.32	**32.06**	29.60	**30.65**
	32.39	32.83	30.03	30.04	29.28	39.37	31.63	31.92	30.17	30.29
50	27.95	**29.45**	26.13	26.35	25.73	25.86	27.79	**28.86**	25.47	**27.14**
	28.24	29.40	26.34	**26.64**	26.06	**26.17**	28.15	28.80	26.08	26.78
100	23.71	**25.43**	21.75	**22.91**	21.69	22.62	24.46	**25.51**	21.89	**23.49**
	23.83	24.84	21.94	22.64	22.05	**22.84**	24.49	25.06	22.07	22.95

σ	boat		couple		hill		**Average**	
5	37.22	37.28	37.31	37.50	37.02	37.13	37.91	38.14
	37.35	**37.35**	37.42	**37.54**	37.11	**37.17**	38.12	**38.20**
10	33.64	33.90	33.52	**34.03**	33.37	33.60	34.30	34.73
	33.93	**33.98**	33.84	33.97	33.59	**33.70**	34.62	**34.74**
15	31.73	32.10	31.45	**32.10**	31.47	31.86	32.34	**32.86**
	32.04	**32.13**	31.83	31.94	31.78	**31.88**	32.67	32.78
20	30.36	**30.85**	30.00	**30.74**	30.18	**30.70**	30.96	**31.57**
	30.74	30.82	30.42	30.59	30.53	30.66	31.34	31.46
25	29.28	**29.84**	28.90	**29.68**	29.18	**29.82**	29.88	**30.56**
	29.67	29.82	29.31	29.51	29.52	29.78	30.25	30.45
50	25.95	26.56	25.32	26.32	26.27	**27.04**	26.33	27.20
	26.36	**26.74**	25.78	**26.36**	26.52	**27.04**	26.69	**27.24**
100	22.81	23.64	22.60	**23.39**	23.98	**24.44**	22.86	**23.93**
	22.96	**23.67**	22.73	23.16	23.92	24.16	23.00	23.67

5. Image processing applications. Applying our multiscale scheme to some image processing tasks proves to noticeably improve the results compared to the single-scale original algorithm, leading to state-of-the-art results in several image processing tasks. We turn to present such results below.[4]

5.1. Grayscale image denoising. We present denoising results obtained with the proposed multiscale sparsity framework and the algorithmic improvements that we have introduced. In Table 1 our results for $N = 1$ (single-scale) and $N = 2$ scales

[4]The initial dictionaries for all the examples presented in this paper are either a standard DCT one or a dictionary learned from a set of natural images, which did not include the tested set.

are carefully compared to the original K-SVD algorithm [20] and the recent results reported in [12].[5] The best results are shared between our algorithm and [12]. As can be observed, the differences are insignificant. Our average performance is better for $\sigma \leq 10$ and for $\sigma = 50$, while the results from [12] are slightly better or similar to ours for other noise levels. Tuning more carefully the parameters of these two algorithms is not expected to change these near-equivalent performances by much. Our framework is, of course, a general multiscale representation, and, as such, it is applicable to other image processing tasks, some of them demonstrated hereafter.

The peak signal-to-noise ratio (PSNR) values in Table 1, corresponding to the results in [12, 20] and our algorithm, are averaged over five experiments for each image and each level of noise, to cope with the variability of the PSNR with the different noise realizations. We also compared our results with a very recent paper [26], which is an extension of [35] with noticeable improvements. In this work, the authors presented some experiments over a data set that has five images in common with the one we chose (house, peppers, lena, barbara, boat) and four standard deviations for the noise (10, 25, 50, 100). For very high noise ($\sigma = 100$), their algorithm performs better than ours and slightly better than [12]. Nevertheless, for other values of noise, we have an improved average PSNR of 0.2dB over these five images.

During our experiments, the number of atoms k_s for each scale was set to 256, the parameter λ was set to $0.45n^2/\sigma$, and η, which gives a preference of the constant atom during the OMP, was set to 2.5. The other parameters used are reported in Table 2. The initial dictionaries are the results of an off-line training on a large generic database of images [19, 27]. Some of these dictionaries are shown in Figure 5. The so-called sparsity factor L for this off-line training was set to $L = 6$ for $N = 1$ and $L = 20$ for $N = 2, 3$.

From these experiments, we draw two conclusions: First, the algorithmic improvements and the block denoising approach with $N = 1$ lead to better performance than the original K-SVD, and this is achieved without increasing the computational cost. Second, the 2-scale algorithm provides further noticeable improvement over the single-scale K-SVD, which makes $N = 2$ a relevant choice, although it introduces a higher computational cost. A few examples for $N = 2$ are presented in Figure 6. Using $N = 3$ scales can provide further improvement at a higher computational cost, as illustrated in Table 3 for $\sigma = 10, 15$ and images of size 256×256. A visual comparison between the use of different scales is shown in Figure 7. In these images, as the denoising performance is already very good for one and two scales, the visual improvements are difficult to observe. Nevertheless, on the zoomed parts of the images, one can notice that $N = 3$ provides a more precise brick texture on the image house and fewer artifacts in the flat areas of the image cameraman.

Some examples of multiscale learned dictionaries are presented in Figures 5, 8, 9, and 10. As can be observed, the very strong structure from the image barbara can be observed through the different scales.

With $N > 3$, our multiscale scheme proves not to be flexible enough to be used, since it leads to significant computational cost and optimization problems of the parameters involved. Further work is required to modify this scheme to allow such flexibility. Using image pyramids is a topic we are currently considering.

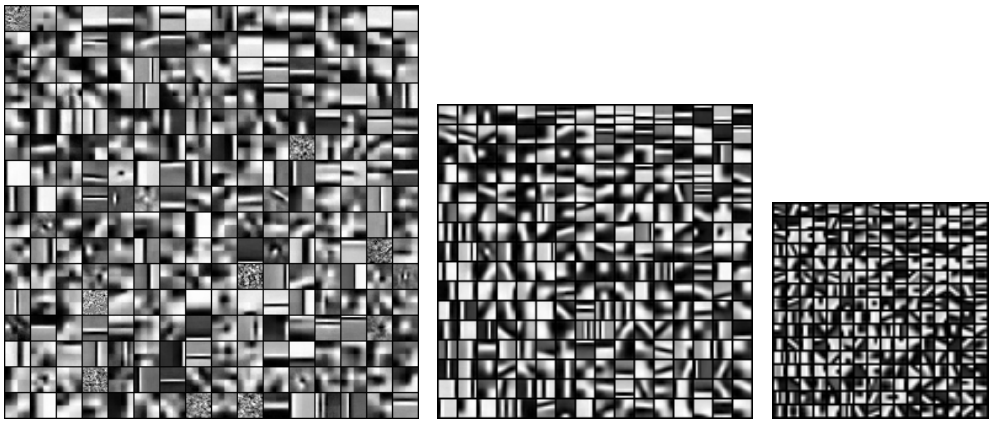
We implemented a parallel version of the algorithm in C++ using OpenMP for

[5] The results in [12] are the best known denoising results at the time of writing this paper. These go beyond the performance reported in [19, 20, 22, 35], which until recently were the leading ones, each for a short period of time.

TABLE 2

Parameters used for the grayscale denoising experiments presented in Figure 6 *and Table* 1*:* (i) N *is the number of scales;* n *is the size of the patches;* (ii) J *is the number of learning iterations;* (iii) μ *is the fraction of patches used during the training;* (iv) m *is the number of adjacent and nonoverlapping patches processed at the same time (see section* 4*);* (v) C *is the parameter from* (2.2)*;* (vi) *the block denoising algorithm has been applied to* $\sqrt{S_b} \times \sqrt{S_b}$ *blocks when* $\sqrt{S_b}$ *was smaller than the size of the input image.*

N	$N=1$						
σ	5	10	15	20	25	50	100
$\sqrt{n}$	8	8	8	8	8	8	8
J	30	30	30	30	30	15	15
μ	0.5	0.5	0.5	0.5	0.5	1.0	1.0
m	1	1	64	64	64	64	64
C	1.128	1.128	1.041	1.023	1.023	1.018	1.018
$\sqrt{S_b}$	150	150	200	200	200	512	768
N	$N=2$						
σ	5	10	15	20	25	50	100
$\sqrt{n}$	10	12	16	16	16	20	20
J	30	30	30	30	30	15	15
μ	0.5	0.5	0.5	0.5	0.5	1.0	1.0
m	4	4	16	16	16	64	64
C	1.069	1.042	1.026	1.026	1.020	1.010	1.008
$\sqrt{S_b}$	150	200	200	250	400	512	768



(a) $s = 0$ (b) $s = 1$ (c) $s = 2$

FIG. 5. *A learned* 3*-scale global dictionary, which has been trained over a large database of natural images.*

parallelism and the Intel Math Kernel Library for the matrix computation. On a recent quad-core Intel Xeon 2.33 GHz, $J = 30$ iterations for one 200×200 block of the image lena with $\sigma = 15$ took approximately 8 seconds for $N = 1$ scale and 58 seconds for $N = 2$ scales, using the parameters from the above experiments.[6]

[6]The code will be made publicly available upon publication.

(a) *Original image* boat (b) *Noisy,* $\sigma = 50$ (c) *Result, PSNR*=26.74*dB*

(d) *Original image* hill (e) *Noisy,* $\sigma = 10$ (f) *Result, PSNR*=33.68*dB*

(g) *Original image* lena (h) *Noisy,* $\sigma = 10$ (i) *Result, PSNR*=35.85*dB*

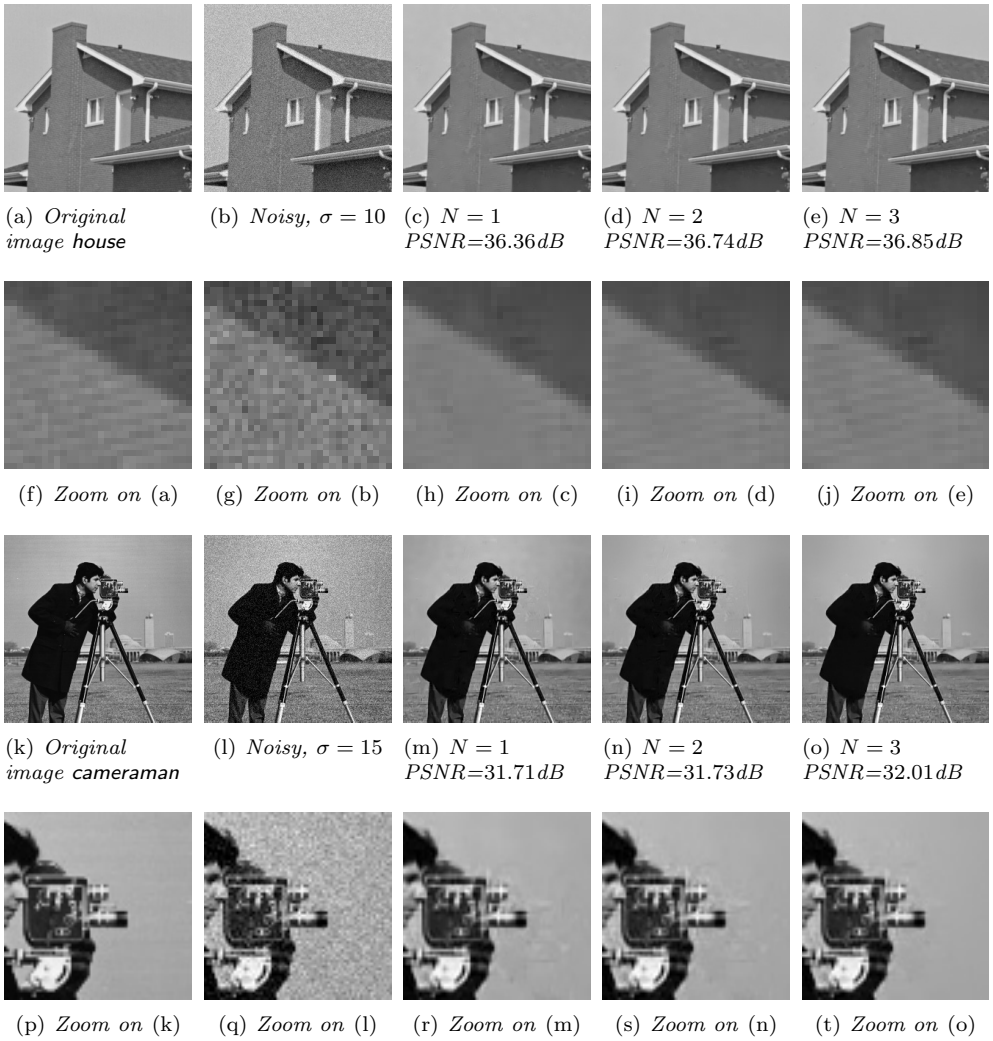
FIG. 6. *Examples of denoising results for* $N = 2$ *scales.*

TABLE 3

PSNR improvements obtained using $N = 3$ *scales for* $\sigma = 10$ *and* $\sigma = 15$ *compared to the case of* $N = 2$ *scales. For* $N = 3$*, a dictionary with* $k_s = 256$ *for all* $s = 0, 1, 2$*,* $m = 4$*, and* $C = 1.018$ *were used.*

σ	n	house	peppers	cameraman	**Average**
10	20×20	+0.10dB	+0.03dB	+0.00dB	**+0.04dB**
15	20×20	−0.07dB	+0.18dB	+0.28dB	**+0.13dB**
15	24×24	+0.02dB	+0.13dB	+0.15dB	**+0.10dB**

5.2. Color image denoising. In [27], we presented state-of-the-art results for color image denoising using the previously described modified version of the K-SVD. These results have recently been slightly surpassed [11]. Here we apply our multiscale



(a) *Original image* house (b) *Noisy,* $\sigma = 10$ (c) $N = 1$ $PSNR$=36.36dB (d) $N = 2$ $PSNR$=36.74dB (e) $N = 3$ $PSNR$=36.85dB

(f) *Zoom on* (a) (g) *Zoom on* (b) (h) *Zoom on* (c) (i) *Zoom on* (d) (j) *Zoom on* (e)

(k) *Original image* cameraman (l) *Noisy,* $\sigma = 15$ (m) $N = 1$ $PSNR$=31.71dB (n) $N = 2$ $PSNR$=31.73dB (o) $N = 3$ $PSNR$=32.01dB

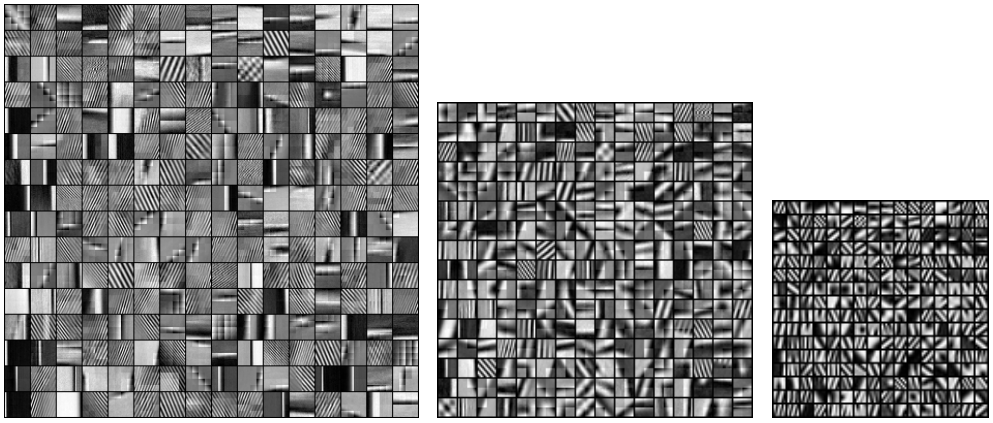
(p) *Zoom on* (k) (q) *Zoom on* (l) (r) *Zoom on* (m) (s) *Zoom on* (n) (t) *Zoom on* (o)

FIG. 7. *A comparison between* $N = 1, 2, 3$ *scales.*

framework and our algorithmic improvements to the color denoising K-SVD to show that it can compete and again provide state-of-the-art results. Like in [27], we use a data set composed of natural images from the Berkeley Segmentation Database [31]; see Figure 11.

Numerical results are presented in Table 4 and some visual results in Figure 12. All the numbers presented here are averaged over five experiments for each image and each level of noise. The parameters used during the experiments are reported in Table 5, where we can observe that our experiments indicate that for $N = 2$ the parameter γ proves to be useful only for high noise levels ($\sigma \geq 25$).

As can be seen, our model with $N = 1$ is already close to [11] (-0.03dB on average) and even slightly better for $\sigma \leq 5$ ($+0.05$dB). With $N = 2$ scales, we have an average improvement of $+0.06$dB over the single-scale algorithm and $+0.04$dB over [11]. One can also note that our color denoising algorithm is a lot more efficient than handling



(a) $s = 0$ (b) $s = 1$ (c) $s = 2$

FIG. 8. *A learned 3-scale dictionary, which has been trained over a noisy version of the image barbara, with $\sigma = 15$. This image is presented in Figure 4. The initial dictionary is a global one, presented in Figure 5.*

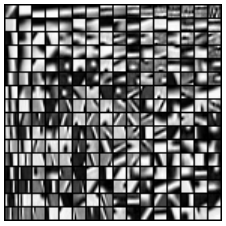
each RGB channel separately, providing a very important average improvement of 2.65dB on our data set. For illustrative purposes, some color multiscale dictionaries are presented in Figure 13. Very interestingly, the color information seems to be present mainly at the coarse scale.

5.3. Image inpainting. Filling in small holes in images was presented in [27] using the K-SVD algorithm. Here we show that using more than one scale can lead to visually impressive results. For illustrative purposes, we show an example obtained with $N = 2$ scales in Figure 14, compared with $N = 1$. This result is quite impressive, bearing in mind that it is able to retrieve the brick texture of the wall, something that our visual system is not able to do. In this example, the multiscale version provides an improvement of 2.24dB over the single-scale algorithm. Within our inpainting framework, the OMP process should stop only when it perfectly reconstructs the input data. Nevertheless, it proved to be more efficient in our experiments to set a maximum number of atoms in the decomposition of the patches, which we denote as the sparsity factor L.

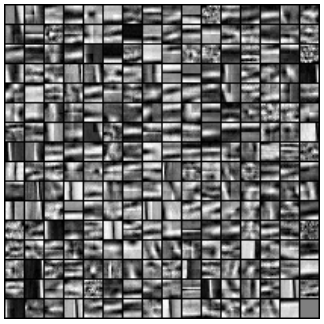
6. Video processing applications. We demonstrate the proposed framework on several video processing applications—color video denoising and video inpainting.

6.1. Color video denoising. Figure 15 presents a result obtained on a sequence of five images taken from a classical video sequence, with added white Gaussian noise of standard deviation $\sigma = 25$. On the third column, we present the results obtained by denoising each frame separately using the multiscale K-SVD algorithm for color images using the same parameters as in subsection 5.2. In the last column, we present the output of our multiscale K-SVD algorithm for denoising color videos that takes into account the temporal correlation as well. As can be seen, the multiscale temporal algorithm provides both PSNR and visual improvements. The raw performance difference in terms of PSNR between this two methods is +1.14dB. Looking carefully at the images, we see fewer artifacts and sharper details in the last column.

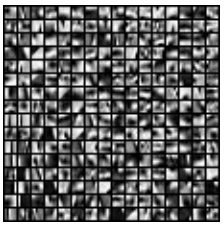
In these experiments, we used patches and atoms of size $n = 10 \times 10 \times 3 \times 3$ with $N = 2$ scales. This means that we used three successive frames to build each patch



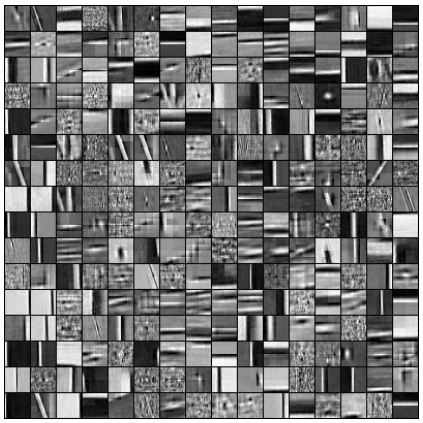
(a) $N = 1$, $s = 0$



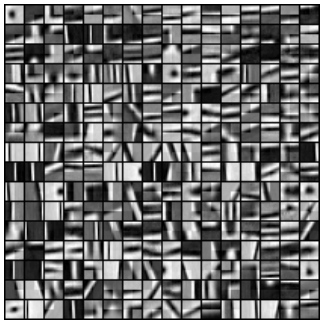
(b) $N = 2$, $s = 0$



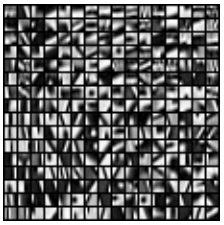
(c) $N = 2$, $s = 1$



(d) $N = 3$, $s = 0$



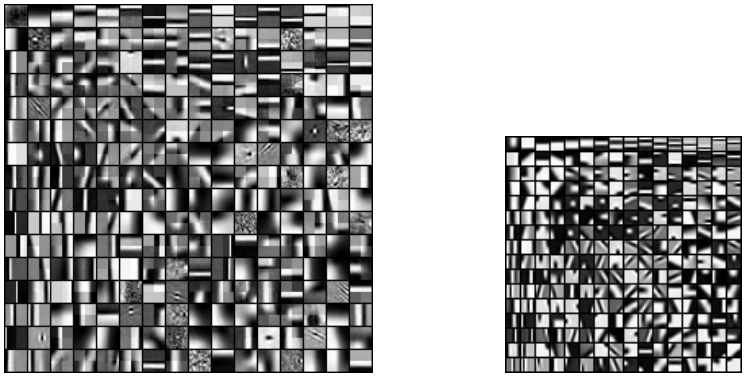
(e) $N = 3$, $s = 1$



(f) $N = 3$, $s = 2$

FIG. 9. *Multiscale dictionaries that have been trained over a noisy version of the image* boat, *with* $\sigma = 15$, $N = 1$, $N = 2$, *and* $N = 3$.

and dictionaries with three temporal channels. The initial dictionary is a *global* one, trained on a large database of videos, with a sparsity factor $L = 20$. The parameters γ and η are not used ($\gamma = 0.0$ and $\eta = 1.0$), but it proved to be important to introduce some constant red, green, and blue atoms for each temporal channel. The parameters m and C are set, respectively, to 1 and 1.04. $J = 30$ iterations are used during the denoising for the first algorithm (that skips proper treatment of the temporal domain). As we propagate the dictionary, the number of iterations J during the denoising of the next frames is set to 10. 15 frames of the test video were processed, but only 5 are shown in Figure 15.



(a) $s = 0$ (b) $s = 1$

FIG. 10. *A learned 2-scale dictionary, which has been trained on a large set of clean patches from a database of natural images. Compare with Figure 5.*

FIG. 11. *Data used for evaluating the color denoising experiments. (This is a color figure.)*

6.2. Video inpainting. Figure 16 presents results obtained with the multiscale K-SVD for video inpainting and compares these to the results obtained when applying the single-image K-SVD algorithm. As can be observed, taking into account the temporal behavior permits us to achieve better results in terms of PSNR and visual quality. The parameters used when we applied the K-SVD for images on each frame separately were the same as in the experiments in Figure 14, with $J = 60$. For the multiscale K-SVD for the video inpainting algorithm, we used patches and atoms of size $n = 10 \times 10 \times 5$ with $N = 2$ scales (with five temporal channels). The initial dictionary is a *global* one, trained on a large database of videos, with a sparsity factor $L = 20$. The parameter η is set to 2.0. $J = 30$ iterations are used during the processing of the first multiframe and then only 10. We present 5 out of the 15 processed frames.

7. Conclusion and future directions. In this paper we presented a K-SVD based algorithm that is able to learn multiscale sparse image representations. Using a shift-invariant sparsity prior on natural images, the proposed framework achieves state-of-the-art image restoration results. We have shown that this framework can be adapted to video processing, exploiting temporal information. We have observed that our multiscale framework provides noticeable improvements over the original single-scale approach, especially for highly damaged images (high level of noise or missing data). All of the experiments reported in this paper can be reproduced with a C++ software, which will be freely available on the authors' webpage. Our current efforts are devoted in part to the design of faster algorithms, which can be used with any number of scales. One direction we are pursuing is to combine the K-SVD with image

TABLE 4

PSNR results for our color image denoising experiments. Each cell is composed of four parts: The top-left cell is devoted to [11]*, the top-right cell to our* 2*-scale gray image denoising method applied to each RGB channel independently, the bottom-left cell to the color denoising algorithm with* $N = 1$ *scale, and the bottom-right cell to our algorithm with* $N = 2$ *scales. Each time the best results are in bold.*

σ	castle		mushroom		train	
5	**40.84**	38.27	40.20	37.65	39.91	36.52
	40.77	40.79	**40.26**	**40.26**	**40.04**	40.03
10	36.61	34.25	**35.94**	33.46	34.85	31.37
	36.51	**36.65**	35.88	35.92	34.90	**34.93**
15	**34.39**	31.95	**33.61**	31.21	31.95	28.53
	34.22	34.37	33.51	33.58	31.98	**32.04**
20	**32.84**	30.52	**31.99**	29.74	29.97	26.79
	32.63	32.77	31.86	31.97	29.97	**30.01**
25	**31.68**	29.47	**30.84**	28.69	28.45	26.55
	31.45	31.59	30.67	30.75	28.50	**28.53**

σ	horses		kangaroo		**Average**	
5	**40.46**	37.17	39.13	35.73	40.11	37.07
	40.44	40.45	**39.26**	39.25	40.15	**40.16**
10	**35.78**	32.70	34.29	31.20	35.49	32.60
	35.67	35.75	34.31	**34.34**	35.45	**35.52**
15	33.18	30.48	31.63	29.05	32.95	30.24
	33.11	**33.19**	31.71	**31.75**	32.91	**32.99**
20	31.44	29.13	29.85	27.77	31.22	28.79
	31.35	**31.47**	29.99	**30.07**	31.16	**31.26**
25	30.19	28.21	28.65	26.90	29.96	27.96
	30.19	**30.28**	28.82	**28.87**	29.93	**30.00**

pyramids. Results along this direction will hopefully be reported soon.

At the more general level, we should ask ourselves how far we are from the performance limits for some image and video enhancement problems, such as image denoising and demosaicing. Understanding these limits is critical for evaluating the importance of future efforts in these challenging problems, and this stands today as a major challenge.

Acknowledgments. We would like to thank the authors of [11, 12] for providing very efficient and intuitive implementations of the BM3D and CBM3D algorithms.

(a) *Original* (b) $\sigma = 10$ (c) *Denoised*

(d) *Original* (e) $\sigma = 25$ (f) *Denoised*

(g) *Original* (h) $\sigma = 25$

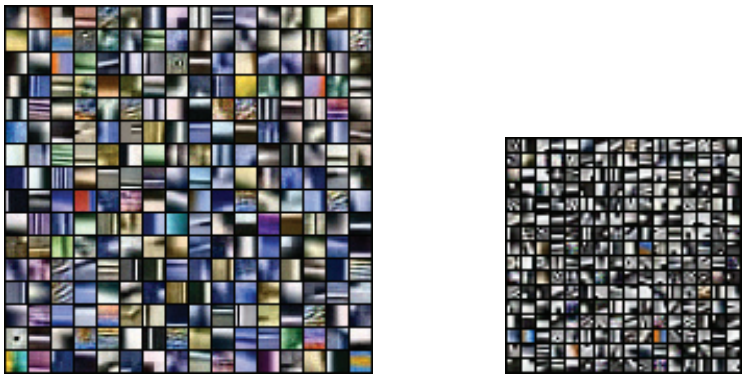
(i) *Denoised*

FIG. 12. *Results for color image denoising with two scales. For the* castle *image, the resulting PSNR is* 36.65*dB, for the* mushroom 30.78*dB, and for the* horses 30.25*dB. (This is a color figure.)*

TABLE 5

Parameters used for the color denoising algorithm: (i) N *is the number of scales;* n *is the size of the patches;* (ii) J *is the number of learning iterations;* (iii) μ *is the fraction of patches used during the training;* (iv) η *gives a preference to the constant atom during the OMP;* (v) γ *enforces the average color of the patches (see* [27]*);* (vi) m *is the number of patches processed at the same time (see section* 4*);* (vii) C *is the parameter from* (2.2)*;* (viii) *the block denoising algorithm has been applied to* $\sqrt{S_b} \times \sqrt{S_b}$ *blocks when* $\sqrt{S_b}$ *was smaller than the size of the input image.*

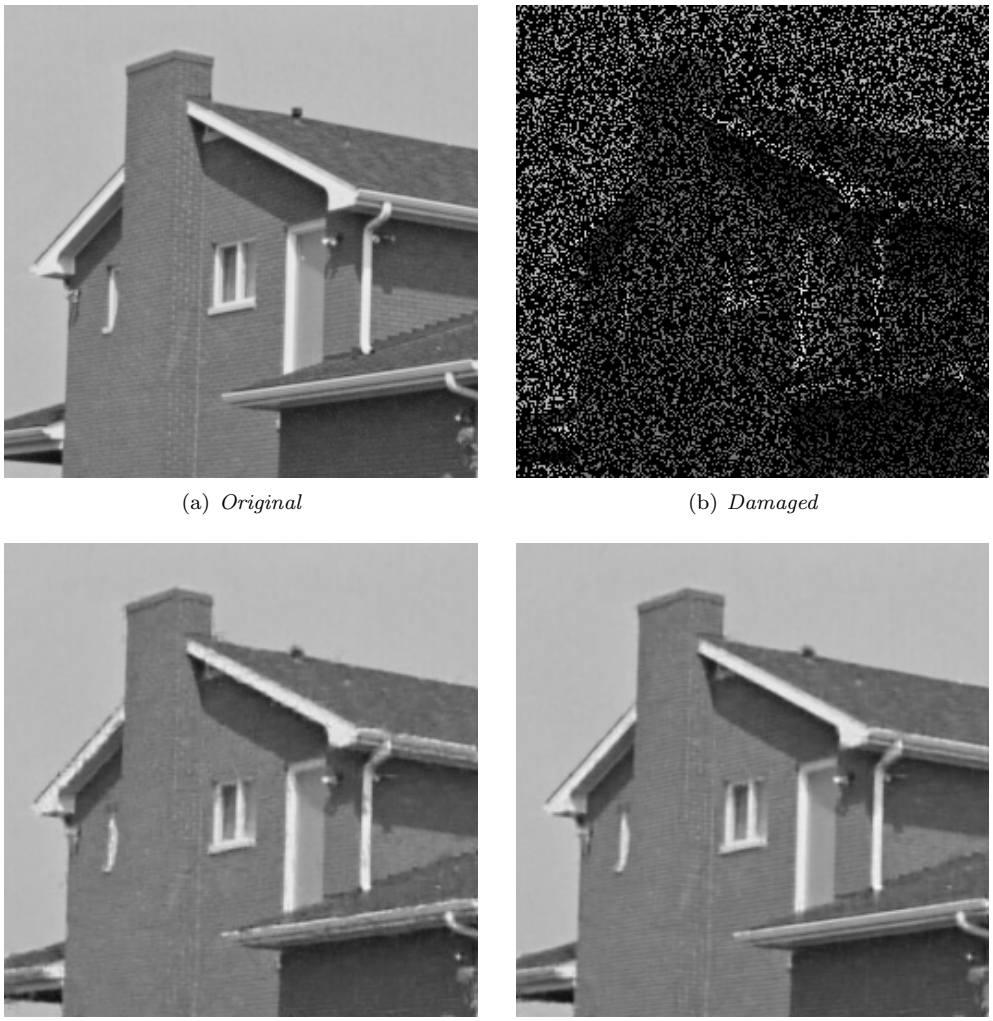
N	$N=1$					$N=2$				
σ	5	10	15	20	25	5	10	15	20	25
$\sqrt{\frac{n}{3}}$	6	6	7	7	8	10	10	12	14	14
J	30	30	30	30	30	30	30	30	30	30
μ	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
η	2.0	2.0	2.0	2.0	2.0	1.5	1.5	1.5	1.5	1.5
γ	0.0	1.25	3.0	5.25	5.25	0.0	0.0	0.0	0.0	1.25
m	64	64	64	64	64	4	16	64	64	64
C	1.016	1.016	1.014	1.014	1.012	1.019	1.01	1.004	1.003	1.003
$\sqrt{S_b}$	300	300	300	300	300	300	300	300	300	300



(a) $s=0$ (b) $s=1$

(c) $s=0$ (d) $s=1$

FIG. 13. *Two learned* 2*-scale color dictionaries. The top one has been trained over a noisy version of the image* castle*, with* $\sigma = 10$*, and the initial dictionary was a global one. The bottom dictionary has been trained on a large set of clean patches from a database of natural images. Since the atoms can have negative values, the vectors are presented scaled and shifted to the* $[0, 255]$ *range per channel. (This is a color figure.)*



(a) *Original* (b) *Damaged*

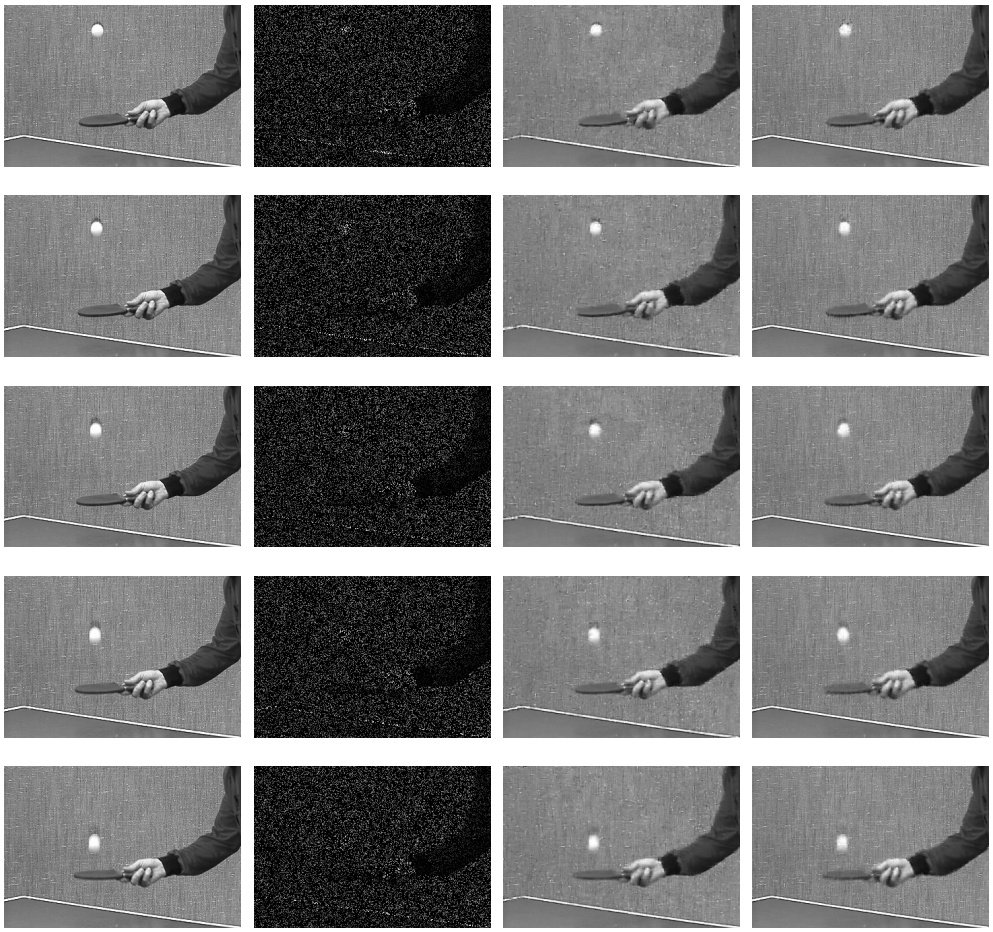
(c) *Restored,* $N = 1$ (d) *Restored,* $N = 2$

FIG. 14. *Inpainting using* $N = 2$ *and* $n = 16 \times 16$ *(bottom-right image), or* $N = 1$ *and* $n = 8 \times 8$ *(bottom-left).* $J = 100$ *iterations were performed, producing an adaptive dictionary. During the learning,* 50% *of the patches were used. A sparsity factor* $L = 10$ *has been used during the learning process and* $L = 25$ *for the final reconstruction. The damaged image was created by removing* 75% *of the data from the original image. The initial PSNR is* $6.13dB$*. The resulting PSNR for* $N = 2$ *is* $33.97dB$ *and* $31.75dB$ *for* $N = 1$*.*

(a) *Original* (b) *Damaged* (c) *Image Denoising* (d) *Video Denoising*

(e) *Zoom on* (a) (f) *Zoom on* (b) (g) *Zoom on* (c) (h) *Zoom on* (d)

FIG. 15. *Results obtained with the proposed multiscale K-SVD for video denoising. From left to right: five frames of an original video, the same frames with Gaussian additive noise ($\sigma = 25$), the results obtained when applying the color image denoising algorithm working on each frame separately (PSNR:* 27.14*dB), and the result of the proposed color video denoising multiscale K-SVD (PSNR:* 28.28*dB). The last row presents a zoomed version of one part of the last frame. (This is a color figure.)*



(a) *Original* (b) *Damaged* (c) *Image Inpainting* (d) *Video Inpainting*

(e) *Zoom on* (a) (f) *Zoom on* (b) (g) *Zoom on* (c) (h) *Zoom on* (d)

FIG. 16. *Results obtained with the proposed multiscale K-SVD for video inpainting. From left to right: five frames of a video are shown, the same sequence with* 80% *of data missing, the results obtained when applying the image inpainting algorithm to each frame separately (PSNR:* 24.38*dB), and the result of the new video inpainting K-SVD (PSNR:* 28.49*dB). The last row presents a zoomed version of one part of the last frame. Each time, an adaptive dictionary was used.*

REFERENCES

[1] M. Aharon, M. Elad, and A. M. Bruckstein, *The k-svd: An algorithm for designing of overcomplete dictionaries for sparse representations*, IEEE Trans. Signal Process., 54 (2006), pp. 4311–4322.

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, *Image inpainting*, in SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, ACM, New York, 2000, pp. 417–424.

[3] A. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[4] A. Buades, B. Coll, and J. M. Morel, *A review of image denoising algorithms, with a new one*, Multiscale Model. Simul., 4 (2005), pp. 490–530.

[5] E. Candes and D. L. Donoho, *Recovering edges in ill-posed inverse problems: Optimality of curvelet frames*, Ann. Statist., 30 (2002), pp. 784–842.

[6] E. Candes and D. L. Donoho, *New tight frames of curvelets and the problem of approximating piecewise* $\mathbf{C}^2$ *images with piecewise* $\mathbf{C}^2$ *edges*, Comm. Pure Appl. Math., 57 (2004), pp. 219–266.

[7] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., 20 (1998), pp. 33–61.

[8] K.-H. Chung and Y.-H. Chan, *Color demosaicing using variance of color differences*, IEEE Trans. Image Process., 15 (2006), pp. 2944–2955.

[9] A. Criminisi, P. Perez, and K. Toyama, *Region filling and object removal by exemplar-based image inpainting*, IEEE Trans. Image Process., 13 (2004), pp. 1200–1212.

[10] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *Image denoising with block-matching and 3D filtering*, in Proc. SPIE Electronic Imaging: Algorithms and Systems V, Vol. 6064, San Jose, CA, 2006.

[11] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *Color image denoising by sparse* 3*d collaborative filtering with grouping constraint in luminance-chrominance space*, in Proceedings of the IEEE International Conference on Image Processing (ICIP), San Antonio, TX, 2007, pp. 313–316.

[12] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *Image denoising by sparse* 3*d transform-domain collaborative filtering*, IEEE Trans. Image Process., 16 (2007), pp. 2080–2095.

[13] G. M. Davis, S. Mallat, and M. Avellaneda, *Adaptive greedy approximations*, Construct. Approx., 13 (1997), pp. 57–98.

[14] G. M. Davis, S. Mallat, and Z. Zhang, *Adaptive time-frequency decompositions*, SPIE J. Opt. Engin., 33 (1994), pp. 2183–2191.

[15] M. Do and M. Vetterli, *Contourlets, Beyond Wavelets*, Academic Press, New York, 2003.

[16] M. Do and M. Vetterli, *Framing pyramids*, IEEE Trans. Signal Process., 51 (2003), pp. 2329–2342.

[17] D. Donoho, *Wedgelets: Nearly minimax estimation of edges*, Ann. Statist., 27 (1998), pp. 859–897.

[18] D. Donoho, M. Elad, and V. Temlyakov, *Stable recovery of sparse overcomplete representations in the presence of noise*, IEEE Trans. Inform. Theory, 52 (2006), pp. 6–18.

[19] M. Elad and M. Aharon, *Image denoising via learned dictionaries and sparse representation*, in Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), New York, 2006, pp. 895–900.

[20] M. Elad and M. Aharon, *Image denoising via sparse and redundant representations over learned dictionaries*, IEEE Trans. Image Process., 54 (2006), pp. 3736–3745.

[21] W. T. Freeman and E. H. Adelson, *The design and the use of steerable filters*, IEEE Trans. Patt. Anal. Mach. Intell., 13 (1991), pp. 891–906.

[22] C. Kervrann and J. Boulanger, *Optimal spatial adaptation for patch-based image denoising*, IEEE Trans. Image Process., 15 (2006), pp. 2866–2878.

[23] R. Kimmel, *Demosaicing: Image reconstruction from color ccd samples*, IEEE Trans. Image Process., 8 (1999), pp. 1221–1228.

[24] E. Le Pennec and S. Mallat, *Bandelet image approximation and compression*, Multiscale Model. Simul., 4 (2005), pp. 992–1039.

[25] X. Li, *Demosaicing by successive approximations*, IEEE Trans. Image Process., 14 (2005), pp. 267–278.

[26] S. Lyu and E. P. Simoncelli, *Statistical modeling of images with fields of Gaussian scale mixtures*, in Advances in Neural Information Processing Systems 19, B. Schölkopf, J. Platt, and T. Hoffmann, eds., MIT Press, Cambridge, MA, 2007, pp. 945–952.

[27] J. Mairal, M. Elad, and G. Sapiro, *Sparse representation for color image restoration*, IEEE Trans. Image Process., 17 (2008), pp. 53–69.

[28] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed., Academic Press, New York, 1999.
[29] S. Mallat and E. Le Pennec, *Sparse geometric image representation with bandelets*, IEEE Trans. Image Process., 14 (2005), pp. 423–438.
[30] S. Mallat and Z. Zhang, *Matching pursuit in a time-frequency dictionary*, IEEE Trans. Signal Process., 41 (1993), pp. 3397–3415.
[31] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, in Proceedings of the 8th International Conference on Computer Vision, Vol. 2, 2001, pp. 416–423.
[32] D. D. Muresan and T. W. Parks, *Demosaicing using optimal recovery*, IEEE Trans. Image Process., 14 (2005), pp. 267–278.
[33] B. A. Olshausen, P. Sallee, and M. S. Lewicki, *Learning sparse multiscale image representations*, in Advances in Neural Information Processing Systems 15, MIT Press, Cambridge, MA, 2003, pp. 1327–1334.
[34] K. A. Patwardhan, G. Sapiro, and M. Bertalmio, *Video inpainting under constrained camera motion*, IEEE Trans. Image Process., 16 (2007), pp. 545–553.
[35] J. Portilla, V. Strela, M. Wainwright, and E. P. Simoncelli, *Image denoising using scale mixtures of Gaussians in the wavelet domain*, IEEE Trans. Image Process., 13 (2004), pp. 496–508.
[36] M. Protter and M. Elad, *Image sequence denoising via sparse and redundant representations*, IEEE Trans. Image Process., submitted.
[37] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, *Efficient learning of sparse representations with an energy-based model*, in Advances in Neural Information Processing Systems 19, B. Schölkopf, J. Platt, and T. Hoffman, eds., MIT Press, Cambridge, MA, 2007, pp. 1137–1144.
[38] S. Roth and M. J. Black, *Fields of experts: A framework for learning image priors*, in Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, 2005, pp. 860–867.
[39] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, *Shiftable multi-scale transforms*, IEEE Trans. Inform. Theory, 38 (1992), pp. 587–607.
[40] N. Srebro and T. Jaakkola, *Weighted low-rank approximations*, in Proceedings of the 20th International Conference on Machine Learning (ICML), AAAI Press, Menlo Park, CA, 2003, pp. 720–727.
[41] J. A. Tropp, *Greed is good: Algorithmic results for sparse approximation*, IEEE Trans. Inform. Theory, 50 (2004), pp. 2231–2242.
[42] J. A. Tropp, *Just relax: Convex programming methods for identifying sparse signals*, IEEE Trans. Inform. Theory, 51 (2006), pp. 1030–1051.
[43] Y. Weiss and W. T. Freeman, *What makes a good model of natural images?*, in Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Minneapolis, MN, 2007, pp. 1–8.
[44] Y. Wexler, E. Shechtman, and M. Irani, *Space-time completion of video*, IEEE Trans. Patt. Anal. Mach. Intell., 29 (2007), pp. 463–476.