Open access • Journal Article • DOI:10.1137/070697653

Learning multiscale sparse representations for image and video restoration

Julien Mairal, Guillermo Sapiro, Michael Elad

Published on: 16 Apr 2008 - Multiscale Modeling & Simulation (Society for Industrial and Applied Mathematics)

Topics: K-SVD, Image restoration, Image processing, Matching pursuit and Grayscale

LEARNING MULTISCALE SPARSE REPRESENTATIONS FOR IMAGE AND VIDEO RESTORATION

By

Julien Mairal

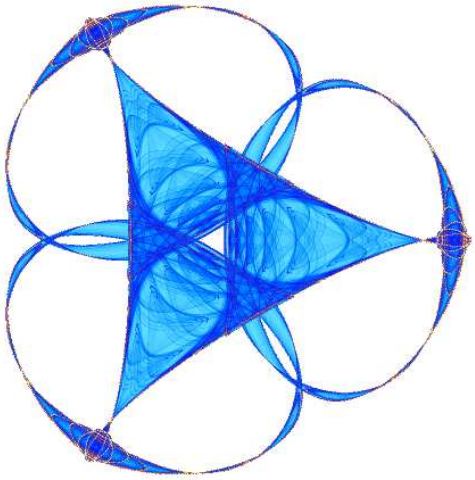
Guillermo Sapiro

and

Michael Elad

IMA Preprint Series # 2168

(July 2007)



INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

UNIVERSITY OF MINNESOTA

400 Lind Hall
207 Church Street S.E.
Minneapolis, Minnesota 55455–0436

Phone: 612/624-6066 Fax: 612/626-7370
URL: http://www.ima.umn.edu

LEARNING MULTISCALE SPARSE REPRESENTATIONS FOR IMAGE AND VIDEO RESTORATION*

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

Abstract. A framework for learning multiscale sparse representations of color images and video with overcomplete dictionaries is presented in this paper. Following the single-scale grayscale K-SVD algorithm introduced in [1], which formulates the sparse dictionary learning and image representation as an optimization problem efficiently solved via orthogonal matching pursuit and SVD, this proposed multiscale learned representation is obtained based on an efficient quadtree decomposition of the learned dictionary and overlapping image patches. The proposed framework provides an alternative to pre-defined dictionaries such as wavelets, and leads to state-of-the-art results in a number of image and video enhancement and restoration applications. The presentation of the framework here proposed is accompanied by numerous examples demonstrating its practical power.

Key words. Image and video processing, sparsity, dictionaries, multiscale representation, denoising, inpainting, interpolation, learning.

AMS subject classifications. 49M27, 62H35

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 elements 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, 19, 25, 35].

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 fulfils a sparsity criterion has been a major challenge in recent years. Such attempts include wavelets [26], curvelets [5, 6], contourlets [14, 15], wedgelets [16], bandlets [27, 28], and steerable wavelets [20, 38]. These methods lead to many effective algorithms in image processing, e.g., image denoising [34]. In this paper, instead of designing the best pre-defined dictionary for image reconstruction, we propose to learn it from examples.

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 highly efficient algorithm [19]. This has been extended to color images, with state-of-the-art results in denoising, inpainting, and demosaicing applications [25], and more recently to video denoising [35]. 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

*Work partially supported by NSF, ONR, NGA, DARPA, the McKnight Foundation, and the Israeli Science Foundation grant No. 796/05.

†WILLOW Team - ENS/INRIA/ENPC, Département d'Informatique, Ecole Normale Supérieure, 45 rue d'Ulm F-75230 Paris Cedex 05, France, Email: julien.mairal@m4x.org. Phone: +331-44-32-21-69. Fax: +331-44-32-21-56. Part of this work was performed while visiting the University of Minnesota.

‡Corresponding author. Department of Electrical and Computer Engineering, University of Minnesota, 200 Union Street SE Minneapolis, MN 55455 USA, E-mail: guille@ece.umn.edu. Phone: +1-612-6251343. Fax: +1-612-6254583.

§Department of Computer Science, The Technion – Israel Institute of Technology, Haifa 32000 Israel, Email: elad@cs.technion.ac.il. Phone: +972-4-829-4169. Fax: +972-4-829-4353.

various image and video processing tasks, obtaining results that outperform previous works. Our results for denoising grayscale images outperform for instance works such as [9, 18, 19, 21, 34, 37]. The proposed algorithm also competes favorably with the most recent and state-of-the-art result in this field [11], which is based on the non-local means algorithm [4]. Our framework for color image denoising also competes favorably with the best known algorithm in this field [10], 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 [32] in the general context of sparsifying image content. Our approach differs from this work in many ways, including: (i) their training algorithm employs a simple steepest descent while ours uses more effective iterations, thus leading to faster convergence; (ii) the structure of the multiscale process; and (iii) the way the found dictionaries are deployed for denoising is entirely different, as we base our algorithm on the energy minimization method introduced in [19]. This explains the significantly superior performance we obtain. Other results on learning single-scale image dictionaries include for example [36, 37, 42].

The structure of this paper is as follows: In Section 2, we briefly review relevant background: The original K-SVD denoising algorithm [1], the extensions to color image denoising, non-homogeneous noise, and inpainting [25], and the K-SVD for denoising videos [35]. Section 3 is devoted to the presentation of our multiscale scheme, and this section is followed by two sections that introduce important algorithmic improvements to the original single-scale K-SVD. Section 6 presents some applications of the multiscale K-SVD, covering grayscale and color image denoising and image inpainting. In Section 7 we show the performance for video processing. Section 8 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 [18, 19], color image denoising [25], inpainting and demosaicing [25], and video denoising [35]. In this section, we briefly review these algorithms.

2.1. The grayscale image denoising K-SVD algorithm. We now briefly review the main ideas of the K-SVD framework for sparse image representation and denoising. The reader is referred to [18, 19] 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 algorithm aims at finding a sparse approximation of every $\sqrt{n} \times \sqrt{n}$ overlapping patch of $\mathbf{y}$, where n is fixed a-priori. This representation is done over an adapted dictionary $\mathbf{D}$, learned for this set of patches. These approximations of patches are averaged to obtain the reconstructed image. This algorithm (shown in Figure 2.1) can be described as the minimization of an energy:

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

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 it's 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 $\hat{\mathbf{D}}$. The notation $||.||_0$ is the ℓ^0 quasi-norm, a sparsity measure, which counts the number of non-zero elements in a vector. The operator $\mathbf{R}_{ij}$ is a binary matrix which extracts the square $\sqrt{n} \times \sqrt{n}$ patch of coordinates $[i,j]$ from the image written as a column vector. The main steps of the algorithm are (refer to Figure 2.1):

- *Sparse Coding Step:* This is performed with an Orthogonal Matching Pursuit (OMP) [12, 13, 29], which proves to be very efficient for diverse approximation problems [17, 40, 41]. 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 [25].
- *Dictionary Update:* This is a sequence of one-rank 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 $\hat{\mathbf{x}}$. Equation (2.4) emerges directly from the energy minimization in Equation (2.1).

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 $\hat{\mathbf{D}} = (\hat{\mathbf{d}}_l \in \mathbb{R}^{n\times 1})_{l\in 1\ldots k}$ (e.g., redundant DCT).
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:

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

- *Dictionary Update:* Fix all $\hat{\alpha}_{ij}$, and for each atom $\hat{\mathbf{d}}_l$, $l \in 1, 2, \ldots, k$ in $\hat{\mathbf{D}}$,
  - Select the set of patches ω_l that use this atom, $\omega_l := \{[i,j] | \hat{\alpha}_{ij}(l) \neq 0\}$.
  - For each patch $[i,j] \in \omega_l$, compute its residual: $e_{ij}^l = \mathbf{R}_{ij}\hat{\mathbf{x}} - \hat{\mathbf{D}}\hat{\alpha}_{ij} + \hat{\mathbf{d}}_l\hat{\alpha}_{ij}(l)$.
  - Set $\mathbf{E}_l$ as the matrix whose columns are the $\mathbf{e}_{ij}^l$, and $\hat{\alpha}^l$ the vector whose elements are the $\hat{\alpha}_{ij}(l)$.
  - 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 one-rank approximation is performed by a truncated SVD of $\mathbf{E}_l$.

Reconstruction: Perform a weighted average:

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

FIG. 2.1. *The single-scale K-SVD-based grayscale image denoising algorithm.*

When it was designed, this algorithm provided state-of-the-art results. One of its main contributions was its possibility to learn a dictionary on a large database of images (the so-called *global* approach), thereby exploiting intrinsic information

of natural images, or to learn a dictionary over all the overlapping patches of one image (*adaptive* approach), exploiting gathered information from the whole image at a specific location. As in [25], we typically learn off-line a global dictionary and then use it as an initial dictionary at the beginning of the iterative *adaptive* approach presented in Figure 2.1.

2.2. The color image denoising extension. In [25], we showed that we can apply the K-SVD for color image by denoising each RGB patch directly as a long concatenated RGB vector. 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, and especially so when we used the *global* dictionaries. Our study has shown that this phenomenon happens because the dictionary redundancy was too small to represent the diversity of colors among natural images. We used therefore a different metric during the orthogonal matching pursuit that maintains the average color of the original image. With this intention, we introduced a new parameter γ that enforces the average color of the patches. Additional details and numerous examples are given in [25], showing that the proposed framework leads to state-of-the-art results, which are further improved with the multiscale approach and additional algorithmic improvements here introduced.

2.3. Handling non-homogeneous noise and inpainting. The problem of handling non-homogenous noise is very important as non-uniform noise across color channels is very common in digital cameras. In [25], we presented a variation of the K-SVD, which permits to address this issue. Within the limits of this model, we were able to fill-in relatively small holes in the images and we presented state-of-the-art results for image demosaicing, outperforming every specialized interpolation-based methods, such as [7, 22, 23, 31].

Let us now 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 introduced a vector β composed of weights for each location:

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

This leads us to define a weighted K-SVD algorithm based on a different metric for each patch. Since the fine details of this approach are given in [25], we restrict the discussion here to a rough coverage of the main idea. Denoting by $\otimes$ the element-wise multiplication between two vectors, we aim at solving the following problem, which replaces Equation (2.1):

$$\begin{aligned}\left\{\hat{\alpha_{ij}}, \hat{\mathbf{D}}, \hat{\mathbf{x}}\right\} = \underset{\mathbf{D},\alpha_{ij},\mathbf{x}}{\arg\min}\, \lambda||\beta \otimes \beta \otimes (\mathbf{x}-\mathbf{y})||_2^2 + \sum_{ij} \mu_{ij}||\alpha_{ij}||_0 + \\ \sum_{ij} ||(\mathbf{R}_{ij}\beta) \otimes (\mathbf{D}\alpha_{ij} - \mathbf{R}_{ij}\mathbf{x})||_2^2 \ . \end{aligned} \tag{2.5}$$

As explained in [25], 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 Equation (2.3) is replaced by

$$(\hat{\mathbf{d}}_l, \hat{\alpha}^l) = \underset{\alpha, ||\mathbf{d}||_2=1}{\arg\min}\, ||\beta^l \otimes (\mathbf{E}_l - \mathbf{d}\alpha^T)||_F^2, \tag{2.6}$$

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 one-rank approximation matrix (see [39]). The algorithm that we used to approximate a solution is presented Figure 2.2.

Input: $\mathbf{E}$ ($n \times m$ matrix), β ($n \times m$ matrix of weights), `iter` (number of iterations, typically 5).
Output: $\mathbf{d}_{\texttt{iter}}$ ($n \times 1$ vector), $\alpha_{\texttt{iter}}$ ($m \times 1$ vector)
Problem to solve:

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

Initialization: $\mathbf{d}_0 = 0$, $\alpha_0 = 0$.
Loop: For i from 1 to `iter`, use a truncated SVD to solve:

$$(\mathbf{d}_i, \alpha_i) = \underset{\alpha, \|\mathbf{d}\|_2=1}{\arg\min} \|\beta \otimes \mathbf{E} + (\mathbf{1}_{n\times m} - \beta) \otimes \mathbf{d}_{i-1}\alpha_{i-1}^T - \mathbf{d}\alpha^T\|_F^2,$$

where $\mathbf{1}_{n\times m}$ is an $n \times m$ matrix filled with ones.

FIG. 2.2. *A weighted one-rank approximation algorithm.*

Inpainting, e.g., [2, 8], 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, our weighted K-SVD algorithm proved to be very efficient. This inpainting case could also be considered as a specific case of interpolation. The mathematical formulation from Equation (2.5) remains the same, but some values from the matrix β are just equal to zero. Details about this are provided in [25], together with a discussion on how to handle the demosaicing problem that has a fixed and periodic pattern of missing values.

2.4. The video denoising algorithm. The video extension of the K-SVD has been developed and described in [35]. 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 [35], 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 will not be able to fit well to the whole data. At the other extreme, an alternative method that applies the K-SVD single-image denoising algorithm, as described above, to the image sequence is also expected to perform poorly, since we do not exploit the temporal correlation. Therefore, a different approach is proposed in [35], based on the following concepts:

- *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 this three concepts into a mathematical formulation leads to the following modified version of Equation (2.1), the reader should refer to [35] for additional

details:

$$\begin{aligned}&\forall t \in 1 \dots T \ , \\ &\left\{\hat{\alpha}_{ijk}, \hat{\mathbf{D}}_t, \hat{\mathbf{x}}_t\right\} = \underset{\mathbf{D}_t, \alpha_{ijk}, \mathbf{x}_t}{\arg\min} \ \lambda ||\mathbf{x}_t - \mathbf{y}_t||_2^2 + \\ &\qquad \sum_{ij} \sum_{k=t-\Delta_t}^{t+\Delta_t} \mu_{ijk} ||\alpha_{ijk}||_0 + ||\mathbf{D}_t \alpha_{ijk} - \mathbf{R}_{ijk}\mathbf{x}||_2^2 \ . \end{aligned} \tag{2.7}$$

Having concluded the brief background presentation, we proceed to present a multiscale framework that permits 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. The 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 capture information at multiple scales is the main goal of this paper. This section discusses the main principles of our proposed approach.

One simple and naive strategy to introduce multiscale analysis consists of using large patches with a high redundancy factor ($\frac{k}{n}$), and hope 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 $\{n = 16 \times 16, k = 1024\}$. A number of reasons might explain the "failure" of this direct approach. First, it might be that for low dimensions (small n) there is no need for multiscale structure for representation and denoising, becoming more crucial only as the dimension grows. In that respect, 16×16 blocks might not be enough for the original K-SVD algorithm to show the multiscale structure. Another explanation is that the K-SVD may be trapped in a local minima, avoiding the true multiscale result. By explicitly imposing such multiscale structure, we may help in this regard. This leads us naturally to the proposed framework. We note 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.

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 3.1. This is a classical data structure, also used in wedgelets for example [16]. A fixed number of scales, N, is chosen, such that it corresponds to N different sizes of atoms. A large patch of size n pixels is divided along the tree to sub-patches of sizes $n_s = \frac{n}{4^s}$, where $s = 0 \dots N-1$ is the depth in the tree. Then, one different dictionary $\mathbf{D}_s$ composed of k_s atoms of size n_s is learned and used per each scale s.

The overall idea of the multiscale algorithm we propose stays as close as possible to the original K-SVD algorithm, Figure 2.1, with an attempt to exploit the several existing scales. This aims at solving the same energy minimization problem of Equation (2.1), with a multiscale structure embedded within the dictionary $\mathbf{D}$, which is a joint one, composed of all the atoms of all the dictionaries $\mathbf{D}_s$ located at every possible position in the quadtree. For the scale s, there exists 4^s such positions. This

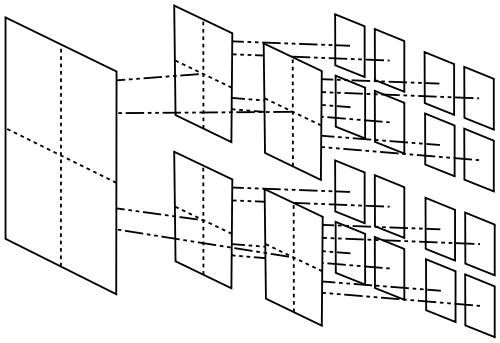


FIG. 3.1. *Quadtree model selected for the proposed multiscale framework.*

makes a total of $\sum_{s=0}^{N-1} 4^s k_s$ atoms in $\mathbf{D}$. This is illustrated in Figure 3.2, where an example of a possible multiscale decomposition is presented.

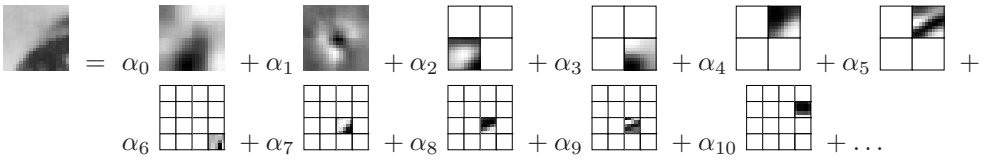


FIG. 3.2. *Possible decomposition of a* 20×20 *patch with a* 3*-scales dictionary.*

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

The original K-SVD exploits the overlapping/shift-invariant treatment of the patches' representation, which has been found to be critical for denoising [18, 19, 25, 37]. One characteristic of our multiscale model is that it permits to force and exploit this overlapping/shift-invariant sparsity at each scale: The use of the quadtree does not allow for all possible shifts for the sub-patches inside one large patch, by letting only 4^s different shifts at the scale s for a sub-patch. This prevents them from constantly adapting their position to a noisy pattern and thereby learning it.

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 Equation (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 $\hat{\mathbf{D}}_s = (\hat{\mathbf{d}}_{sl} \in \mathbb{R}^{n \times 1})_{l \in 1 \dots k_s}$ 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 implemented efficiently using a 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 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 $\hat{\mathbf{d}}_{sl}$ $(1 \leq l \leq k_s)$ in each scale (from $s = 0$ to $s = N-1$):
  - Select the set of sub-patches from the scale s that use the l-th atom,

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

    where $[i,j,s,p]$ denotes the sub-patch 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 sub-patch and the atom $\hat{\mathbf{d}}_{sl}$.
  - For each sub-patch $[i,j,s,p] \in \omega_{sl}$, 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 sub-patch $[i,j,s,p]$ from a patch $[i,j]$.
  - Set $\mathbf{E}_{sl}$ as the matrix whose columns are the $\mathbf{e}^l_{ijsp}$, and $\hat{\alpha}^{sl}$ the vector whose elements are the $\hat{\alpha}_{ij}(s,l,p)$.
  - Update $\hat{\mathbf{d}}_{sl}$ 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:* Remains the same as in Equation (2.4), while using the new notation just introduced. 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 $\mathcal{O}((\sum_{i=0}^{N-1} k_s)nLJM)$ where L is the average sparsity factor (number of coefficients obtained in the decomposition), and M is the number of patches processed.

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 case of the single-scale algorithm, extending the color framework to the multiscale version requires to consider a concatenated RGB vector. Then, the same quadtree structure and the same scheme is applied. The only difference respect to the grayscale algorithm is the use of the parameter γ, which was introduced to solve the bias-color problem that we described in [25]. We recall that it enforces the average color of the patches during the OMP, thereby creating a new metric. In the multiscale case, we can not enforce the average color of a patch, since it would introduce a bias for the sub-patches in the quadtree. Therefore, this enforcement of the average color should be done only for the smallest sub-patches. For instance, assume we have $N = 3$ scales, and a patch of size $n = 20 \times 20 \times 3$. Then, we enforce the average color of each $5 \times 5 \times 3$ sub-patch within the dictionaries and patches.
- *Non homogeneous denoising and inpainting:* For extending the non-homogeneous denoising and inpainting algorithm to the multiscale version, 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 matrix $\mathbf{T}_{sp}\mathbf{R}_{ij}\beta$ for each scale s and position p within the scale. Equation (3.1) has to be adapted to match Equation (2.6):

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

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 $\mathbf{T}_{sp}\mathbf{R}_{ij}\beta$. Again, the algorithm from Figure 2.2 remains relevant.

- *Video denoising:* We now show how to combine the multiscale color K-SVD algorithm and the video one. Both are using patches and atoms with many channels: The R,G,B layers for the color processing, and temporal frames for the video processing (4D-atoms). Concatenating the color channels and different frames in single vectors permits to address the color video denoising problem by minimizing the same energy as in Equation (2.7). For the multiscale extension for denoising image sequences, one can see the K-SVD for video as successive K-SVD for images applied to multi-channels images. It consists indeed of putting the quadtree structure on the considered channel images, and to consider the learning of the dictionaries at each scale, by proceeding exactly like it was done 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 warped metric that uses the parameter γ.
- *Non-homogeneous 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 non-homogeneous noise. This leads to the following energy minimization formulation:

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

Handling the inpainting problem for video via an extension of the previous algorithm is possible since we can regard the processing per each frame as separate, although involving adjacent frames. This permits 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, one should refer to [33, 43]. 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 important additional refinements, which further improve the results without increasing the computational cost.

First, for the grayscale K-SVD we find it useful to force the presence of a constant (DC) atom in each dictionary, and to 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 [13]). 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.

Secondly, as discussed in [25], 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 non-overlapping patches simultaneously and consider them as a larger patch (and therefore associated with a better stopping criterion). In practice, we choose m adjacent and non-overlapping 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.

Finally, we mention a numerical shortcut that was proposed in [35] and we found to be useful. In Figure 2.1, 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.

5. 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 more likely the K-SVD is 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 significantly improve the results. This might be explained by the increased risk of getting stuck in a local minima in the K-SVD training, or perhaps the reason is the reduced performance of the OMP, due to bad coherence of the resulting larger dictionary. This is why the K-SVD has been used so far with a fixed size and relatively small dictionary, which already provides excellent performance.

Nevertheless, one way of addressing the above mentioned problem is to handle separately different zones of the images. In this paper, we chose to naturally define a block denoising algorithm, but future work will consist of combining our denoising algorithm with a segmentation of the input image. More precisely, what we do is to consider blocks of the same size from one image, with a small overlap of the same width as the patches's size as illustrated Figure 5.1.[1] Given a judiciously adapted

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

block size, this approach introduces two advantages: First of all, the performance in terms of denoising results is better since the dictionaries are better adapted to their own regions, as we can notice on Figure 5.1. Secondly, this approach has the same computational complexity with a lower memory usage, since both are linear in the number of denoised pixels.[2]

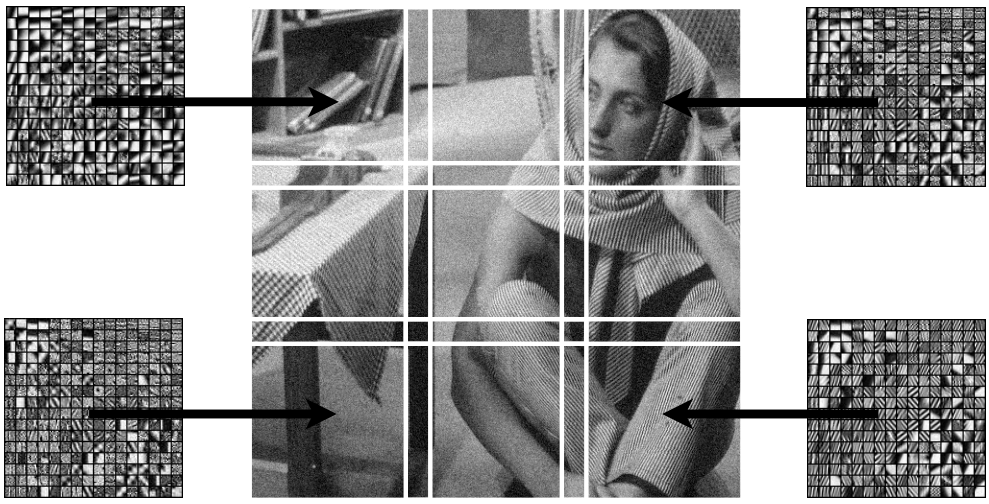


FIG. 5.1. *Illustration of the block denoising algorithm. Four dictionaries trained on four different blocks (out of the overall 9 that we have) of a noisy image* barbara *with* $\sigma = 15$ *during a denoising process are reported on the figure. As can be seen, each dictionary is more adapted to the 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.*

One natural question that is raised here is whether there exists a generic optimal size to choose for these blocks. Answering this requires to take 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 downsize, though, 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 over-fitting, where the dictionary is learning the given examples, and absorbs some of the noise in them as well.
- The bigger σ is, the more patches (and thus bigger blocks) are required to make the K-SVD robust to the noise.

As expected, our experiments showed that the best size for the block denoising algorithm was 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.

6. Application in image processing. Applying our multiscale scheme to some image processing tasks proved to noticeably improve the results compared to the

[2]Note that we neglect the small increase in the number of pixels due to the small overlapping of the blocks.

single-scale original algorithm, leading to state-of-the-art results in many image processing tasks. We turn to present such results below.

6.1. Grayscale image denoising. We now present denoising results obtained within the proposed multiscale sparsity framework and the algorithmic improvements that we have introduced. In Table 6.1, our results for $N = 1$ (single-scale) and $N = 2$ scales are carefully compared to the original K-SVD algorithm [18, 19] and the recent results reported in [11].[3] The best results are shared between our algorithm and [11]. As it can be observed, the differences are insignificant. Our average performance is better for $\sigma \leq 10$ and for $\sigma = 50$, while the results from [11] 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 by much these near-equivalent performance. Our framework is of course a general multiscale representation applicable to numerous image processing tasks, some of them here demonstrated.

The PSNR values in Table 6.1, corresponding to the results in [11, 18, 19] and our algorithm, are averaged over 5 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 [24], which is an extension of [34] 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 4 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 [11]. Nevertheless, for other values of noise, we have an improved average PSNR of 0.20dB over these five images.

During our experiments, the number of atoms k_s for each scale was set to 256, the parameter λ 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 6.2. The initial dictionaries are the results of an off-line training on a large generic database of images [18, 25]. Some of these dictionaries are shown in Figure 6.1. The so-called sparsity factor L for these off-line training was set to $L = 6$ for $N = 1$, $L = 20$ for $N = 2, 3$.

From these experiments, we draw two conclusions: First of all, 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. Secondly, the two-scales 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 Figure 6.2. Using $N = 3$ scales can provide further improvement at a higher computational cost, as illustrated in Table 6.3 for $\sigma = 10, 15$ and images of size 256×256. A visual comparison between the use of different scales is shown in Figure 6.3. 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 less artifacts in the flat areas of the image cameraman.

Some examples of multiscale learned dictionaries are presented in figures 6.1, 6.4, 6.5, and 6.6. As we can observe, the very strong structure from the image barbara can be observed through the different scales.

[3] The results in [11] are the best known denoising results at the time of writing this paper. These go beyond the performance reported in [18, 19, 21, 34], which until recently were the leading ones, each in its short period of time.

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 involved parameters. 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 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.[4]

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

TABLE 6.1

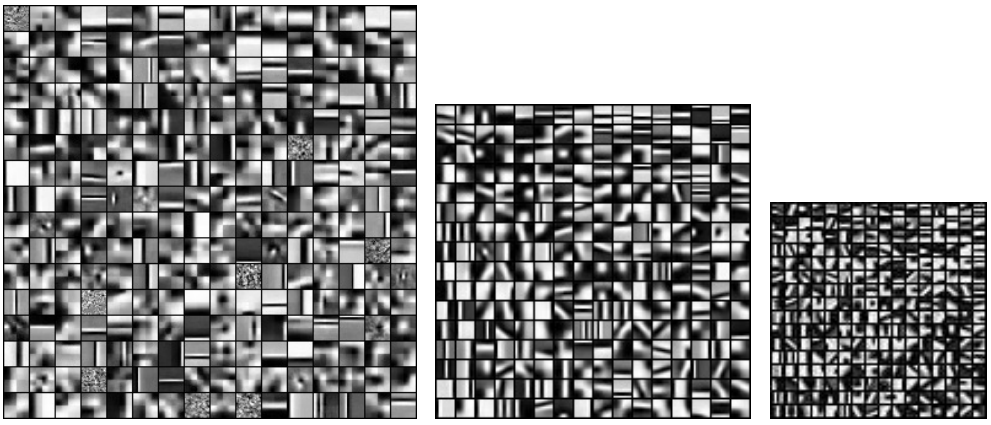
PSNR results of our denoising algorithm compared with some other ones. Each cell is divided into four parts. The top-left part shows the results from the original K-SVD [1], the top-right from the most recent state-of-the-art [11]. The bottom-left is devoted to our results for $N = 1$ scale and the bottom-right to $N = 2$ scales. Each time the best results are in bold.

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

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

TABLE 6.2

Parameters used for the grayscale denoising experiments presented on Figure 6.2 and Table 6.1. n is the size of the patches. J is the number of learning iterations. μ is the fraction of patches used during the training. m is the number of adjacent and non-overlapping patches processed at the same time (see Section 4). C is the parameter from Equation (2.2). 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.



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

FIG. 6.1. *A learned 3-scales global dictionary, which has been trained over a large database of natural images.*

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**

TABLE 6.3

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.

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

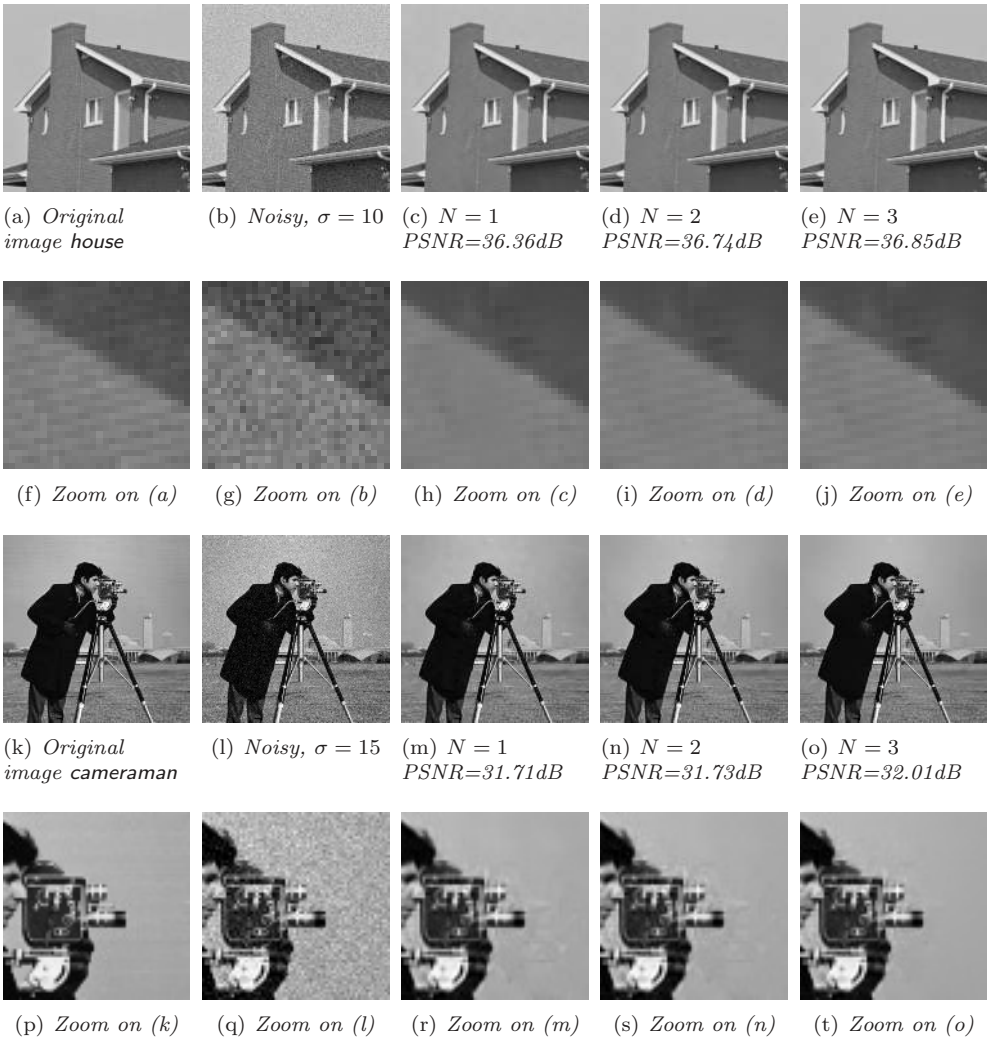
(d) *Original image* **hill** (e) *Noisy,* $\sigma = 10$ (f) *Result, PSNR=33.68dB*

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

FIG. 6.2. *Examples of denoising results for* $N = 2$ *scales.*

6.2. Color image denoising. In [25], 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 [10]. Here we apply our multiscale framework and our algorithmic improvements to the color denoising K-SVD to show that it can compete and provide again state-of-the-art results. Like in [25], we use a data set composed of natural images from the Berkeley Segmentation Database [30], see Figure 6.7.

Numerical results are presented on Table 6.4 and some visual results in Figure 6.8. All the numbers presented here are averaged over 5 experiments for each image and each level of noise. The parameters used during the experiments are reported in Table 6.5, where we can observe that our experiments indicate that for $N = 2$, the parameter γ proves to be useful only for high noise levels ($\sigma \geq 25$).



(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)*

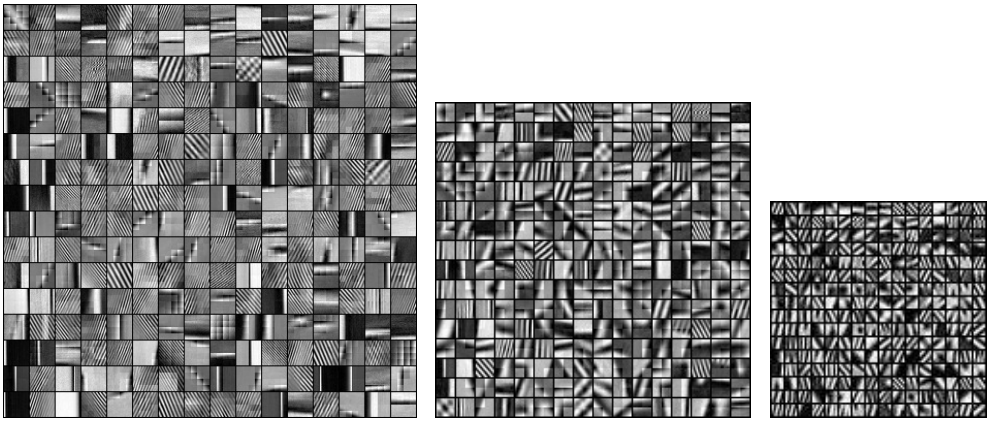
(k) *Original image* cameraman (l) *Noisy,* $\sigma = 15$ (m) $N = 1$ *PSNR=31.71dB* (n) $N = 2$ *PSNR=31.73dB* (o) $N = 3$ *PSNR=32.01dB*

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

FIG. 6.3. *A comparison between* $N = 1, 2, 3$ *scales.*

As we can see, our model with $N = 1$ is already close to [10] (-0.03dB on average) and even slightly better for $\sigma \leq 5$ (+0.05dB). With $N = 2$ scales, we have an average improvement of +0.06dB over the single-scale algorithm and +0.04dB over [10]. One can note also that our color denoising algorithm is a lot more efficient than handling each R,G,B channel separately, providing a very important average improvement of 2.65dB on our dataset. For illustrative purpose, some color multiscale dictionaries are presented Figure 6.9. Very interestingly, the color information seems to be present mainly at the coarse scale.

6.3. Image inpainting. Filling-in small holes in images was presented in [25] 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 6.10, compared with $N = 1$. This result is quite impressive



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

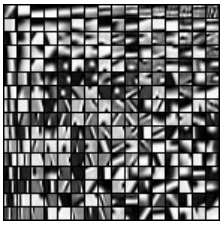
FIG. 6.4. *A learned 3-scales dictionary, which has been trained over a noisy version of the image* barbara*, with* $\sigma = 15$*. This image is presented Figure 5.1. The initial dictionary is a global one, presented in Figure 6.1*

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**

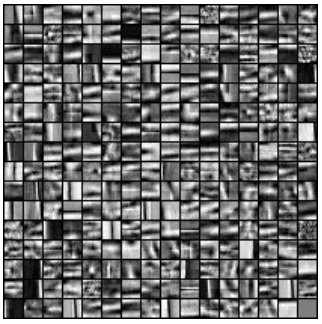
σ	horses		kangaroo		**Average**	
5	**40.46**	37.17	39.13	35.73	40.11	37.07
	40.44	40.45	**39.26**	39.25	40.15	**40.16**
10	**35.78**	32.70	34.29	31.20	35.49	32.60
	35.67	35.75	34.31	**34.34**	35.45	**35.52**
15	33.18	30.48	31.63	29.05	32.95	30.24
	33.11	**33.19**	31.71	**31.75**	32.91	**32.99**
20	31.44	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**

TABLE 6.4

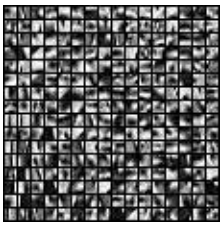
PSNR results for our color image denoising experiments. Each cell is composed of four parts: The top-left is devoted to [10], the top-right to our 2-scales gray image denoising method applied to each R,G,B channel independently, the bottom-left to the color denoising algorithm with $N = 1$ *scale and the bottom-right to our algorithm with* $N = 2$ *scales. Each time the best results are in bold.*



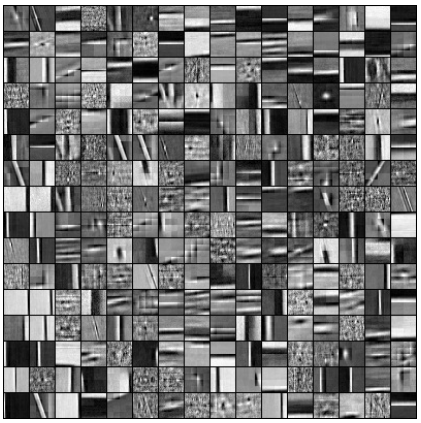
(a) $N = 1, s = 0$



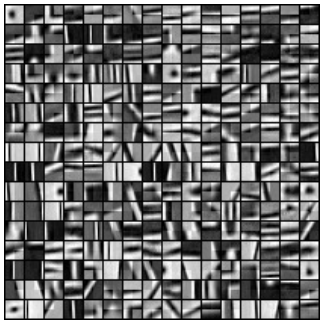
(b) $N = 2, s = 0$



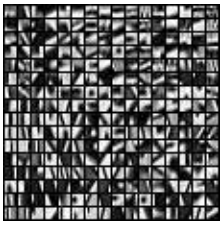
(c) $N = 2, s = 1$



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



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



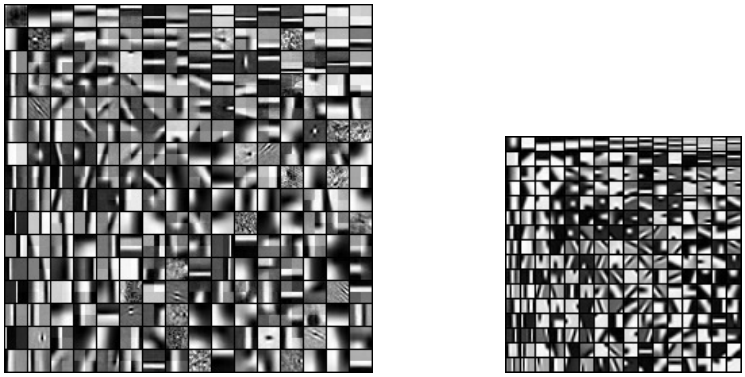
(f) $N = 3, s = 2$

FIG. 6.5. *Multiscale dictionaries that have been trained over a noisy version of the image* boat, *with* $\sigma = 15$, $N = 1$, $N = 2$ *and* $N = 3$.

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.

7. Applications to video processing. We show now that our framework can be extended to video processing. For illustrative purposes, we choose to give two examples: Color video denoising and video inpainting.

7.1. Color video denoising. We now present results obtained by combining the color extension of the multiscale K-SVD and the video one. Figure 7.1 presents a result obtained on a sequence of 5 images taken from a classical video sequence,



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

FIG. 6.6. *A learned 2-scales dictionary, which has been trained on a large set of clean patches from a database of natural images. Compare with Figure 6.1.*

FIG. 6.7. *Data used for evaluating the color denoising experiments. (This is a color figure.)*

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 6.2. On 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 we can see, the multiscale and temporal algorithm can provide 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 less artifacts and sharper details on 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 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 parameter γ and η are not used ($\gamma = 0.0$ and $\eta = 1.0$), but it proved to be important to introduce some constant atoms red, green and blue 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 7.1.

7.2. Video inpainting. Figure 7.2 presents results obtained with the multiscale K-SVD for video inpainting, and compare to the result obtained when applying the K-SVD for image inpainting applied to each frame separately. As we can observe,

(a) *Original*

(b) $\sigma = 10$

(c) *Denoised*

(d) *Original*

(e) $\sigma = 25$

(f) *Denoised*

(g) *Original*

(h) $\sigma = 25$

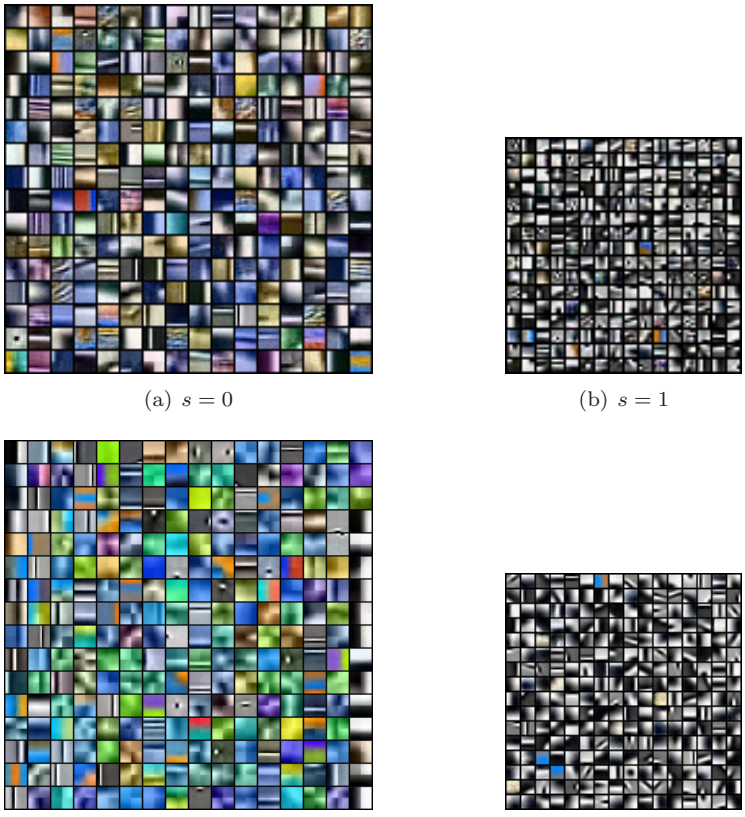
(i) *Denoised*

FIG. 6.8. *Results for color image denoising with* 2 *scales. For the* castle *image, the resulting PSNR is 36.65dB, for the* mushroom *30.78dB, and for the* horses *30.25dB. (This is a color figure.)*

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

TABLE 6.5

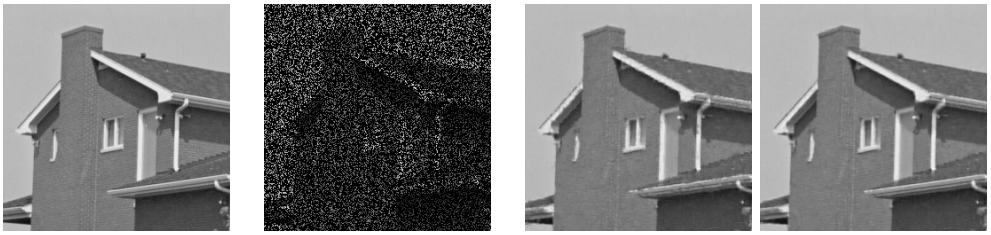
Parameters used for the color denoising algorithm. n is the size of the patches. J is the number of learning iterations. μ is the fraction of patches used during the training. η gives a preference to the constant atom during the OMP. γ enforces the average color of the patches (see [25]). m is the number of patches processed at the same time (see Section 4). C is the parameter from Equation (2.2). 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.



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

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

FIG. 6.9. *Two learned 2-scales color dictionaries. The top one has been trained over a noisy version of the image* castle*, with $\sigma = 10$, 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. 6.10. *Inpainting using* $N = 2$ *and* $n = 16 \times 16$ *(third image), or* $N = 1$ *and* $n = 8 \times 8$ *(fourth image).* $J = 100$ *iterations were performed. 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* $N = 1$*.*

taking into account the temporal behavior permits 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 6.10, with $J = 60$. For the multiscale K-SVD for video inpainting algorithm, we used patches and atoms of size $n = 10 \times 10 \times 5$ with $N = 2$ scales (with 5 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 multi-frame, and then only 10. We present 5 out of the 15 processed frames.

8. 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. All of the experiments reported in this paper can be reproduced with a C++ software, which will be freely available in 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 pyramids. Results along this direction will be hopefully reported soon.

At the more general level, we ask ourself if we are reaching the performance limit for many image and video enhancement tasks such as the image denoising and demosaicing results presented here and in [25]. Understanding these limits is critical to evaluate the importance of future efforts in these challenging problems.

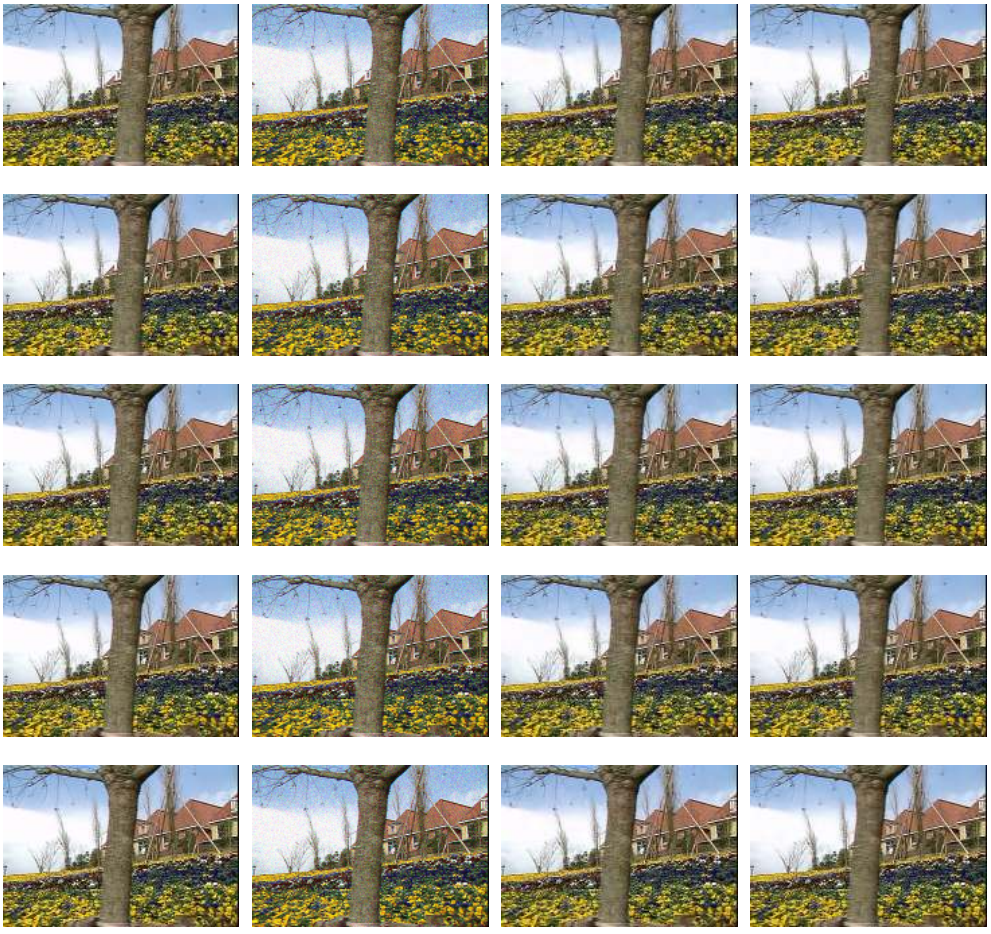
Acknowledgments. We would like to thank the authors of [11, 10], for providing very efficient and intuitive implementations of the BM3D and CBM3D algorithms.

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, 2000, pp. 417–424.
[3] A. Björck, *Numerical Methods for Least-Squares Problems*, SIAM, 1996.
[4] A. Buades, B. Coll, and J.M Morel, *A review of image denoising algorithms, with a new one*, SIAM Journal of Multiscale Modeling and Simulation, 4 (2005), pp. 490–530.
[5] E. Candes and D. L. Donoho, *Recovering edges in ill-posed inverse problems: Optimality of curvelet frames*, Annals of statistics, 30 (2002), pp. 784–842.
[6] ———, *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] K-H. Chung and Y-H. Chan, *Color demosaicing using variance of color differences*, IEEE Trans. Image Process., 15 (2006), pp. 2944–2955.
[8] 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.
[9] 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, USA, January 2006.
[10] ———, *Color image denoising by sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space*, in Proc. IEEE International Conference on Image Processing (ICIP), San Antonio, Texas, USA, September 2007. to appear.
[11] ———, *Image denoising by sparse 3d transform-domain collaborative filtering*, IEEE Trans. Image Process., 16 (2007), pp. 2080–2095. to appear `http://www.cs.tut.fi/~foi/GCF-BM3D/BM3D_TIP_2006.pdf`.
[12] G. M. Davis, S. Mallat, and M. Avellaneda, *Adaptive greedy approximations*, J. Construct. Approx., 13 (1997), pp. 57–98.
[13] G. M. Davis, S. Mallat, and Z. Zhang, *Adaptive time-frequency decompositions*, SPIE J. of Opt. Engin., 33 (1994), pp. 2183–2191.
[14] M. Do and M. Vetterli, *Contourlets, Beyond Wavelets*, Academic Press, New York, 2003.
[15] ———, *Framing pyramids*, IEEE Trans. Signal Process., 51 (2003), pp. 2329–2342.
[16] D. Donoho, *Wedgelets: Nearly minimax estimation of edges*, Annals of statistics, 27 (1998), pp. 859–897.
[17] 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.
[18] M. Elad and M. Aharon, *Image denoising via learned dictionaries and sparse representation*, in Proc. IEEE Computer Vision and Pattern Recognition (CVPR), New York, NY, USA, June 2006.
[19] ———, *Image denoising via sparse and redundant representations over learned dictionaries*, IEEE Trans. Image Process., 54 (2006), pp. 3736–3745.
[20] W. T. Freeman and E. H. Adelson, *The design and the use of steerable filters*, IEEE Pat. Anal. Mach. Intell., 13 (1991), pp. 891–906.
[21] C. Kervrann and J. Boulanger, *Optimal spatial adaptation for patch-based image denoising*, IEEE Trans. Image Process., 15 (2006), pp. 2866–2878.
[22] R. Kimmel, *Demosaicing: image reconstruction from color ccd samples.*, IEEE Trans. Image Process., 8 (1999), pp. 1221–1228.
[23] X. Li, *Demosaicing by successive approximations*, IEEE Trans. Image Process., 14 (2005), pp. 267–278.
[24] S. Lyu and E. P. Simoncelli, *Statistical modeling of images with fields of gaussian scale mixtures*, in Advances in Neural Information Processing Systems (NIPS), Vancouver, Canada, December 2006.
[25] J. Mairal, M. Elad, and G. Sapiro, *Sparse representation for color image restoration*, (2006). submitted, `http://www.ima.umn.edu/preprints/oct2006/2139.pdf`.
[26] S. Mallat, *A Wavelet Tour of Signal Processing, Second Edition*, Academic Press, New York, September 1999.
[27] S. Mallat and E. Le Pennec, *Bandelet image approximation and compression*, SIAM Journal of Multiscale Modeling and Simulation, 4 (2005), pp. 992–1039.
[28] ———, *Sparse geometric image representation with bandelets*, IEEE Trans. Image Process., 14 (2005), pp. 423–438.
[29] S. Mallat and Z. Zhang, *Matching pursuit in a time-frequency dictionary*, IEEE Trans. Signal Process., 41 (1993), pp. 3397–3415.
[30] 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 Proc. 8th Int'l Conf. Computer Vision, vol. 2, July 2001, pp. 416–423.

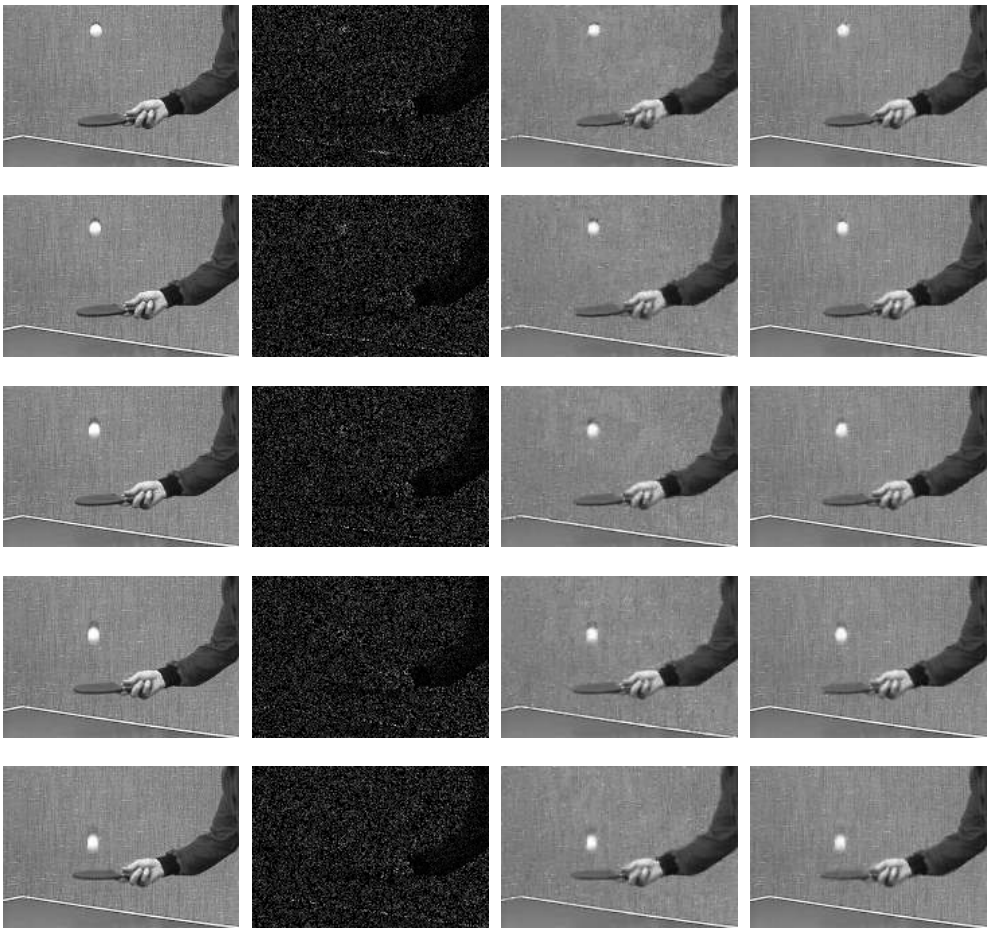
[31] D. D. Muresan and T. W. Parks, *Demosaicing using optimal recovery*, IEEE Trans. Image Process., 14 (2005), pp. 267–278.
[32] B.A. Olshausen, P. Sallee, and M.S. Lewicki, *Learning sparse multiscale image representations*, Advances in Neural Information Processing Systems, 15 (2003), pp. 1327–1334.
[33] K. A. Patwardhan, G. Sapiro, and M. Bertalmio, *Video inpainting under constrained camera motion*, IEEE Trans. Image Process., 16 (2007), pp. 545–553.
[34] 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.
[35] M. Protter and M. Elad, *Image sequence denoising via sparse and redundant representations*, (2007). submitted, `http://www.cs.technion.ac.il/~elad/publications/journals/2007/VideoDenoising_IEEETIP.pdf`.
[36] 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.
[37] S. Roth and M. J. Black, *Fields of experts: A framework for learning image priors.*, in Proc. IEEE Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, June 2005.
[38] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, *Shiftable multi-scale transforms*, IEEE Trans. on Inform. Theory, 38 (1992), pp. 587–607.
[39] N. Srebro and T. Jaakkola, *Weighted low-rank approximations*, in ICML, 2003, pp. 720–727.
[40] J. A. Tropp, *Greed is good: Algorithmic results for sparse approximation*, IEEE Trans. on Inform. Theory, 50 (2004), pp. 2231–2242.
[41] ———, *Just relax: Convex programming methods for identifying sparse signals*, IEEE Trans. Inform. Theory, 51 (2006), pp. 1030–1051.
[42] Y. Weiss and W. T. Freeman, *What makes a good model of natural images ?*, in Proc. IEEE Computer Vision and Pattern Recognition (CVPR), Minneaplis, MN, USA, June 2007.
[43] Y. Wexler, E. Shechtman, and M. Irani, *Space-time completion of video*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 463–476.



(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. 7.1. *Results obtained with the proposed multiscale K-SVD for video denoising. From left to right:* 5 *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: 27.14dB), and the result of the proposed color video denoising multiscale K-SVD (PSNR: 28.28dB). 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. 7.2. *Results obtained with the proposed multiscale K-SVD for video inpainting. From left to right:* 5 *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: 24.38dB), and the result of the new video inpainting K-SVD (PSNR: 28.49dB). The last row presents a zoomed version of one part of the last frame.*