Image Sequence Denoising via Sparse and Redundant Representations

Matan Protter and Michael Elad, *Senior Member, IEEE*

***Abstract*—In this paper, we consider denoising of image sequences that are corrupted by zero-mean additive white Gaussian noise. Relative to single image denoising techniques, denoising of sequences aims to also utilize the temporal dimension. This assists in getting both faster algorithms and better output quality. This paper focuses on utilizing sparse and redundant representations for image sequence denoising, extending the work reported in [1], [2]. In the single image setting, the K-SVD algorithm is used to train a sparsifying dictionary for the corrupted image. This paper generalizes the above algorithm by offering several extensions: i) the atoms used are 3-D; ii) the dictionary is propagated from one frame to the next, reducing the number of required iterations; and iii) averaging is done on patches in both spatial and temporal neighboring locations. These modifications lead to substantial benefits in complexity and denoising performance, compared to simply running the single image algorithm sequentially. The algorithm's performance is experimentally compared to several state-of-the-art algorithms, demonstrating comparable or favorable results.**

***Index Terms*—Denoising, K-SVD, OMP, sparse representations, video.**

I. Introduction

DENOISING of images is one of the most basic tasks of image processing, and as such, it has been extensively studied in the past several decades. This problem is the simplest among the family of problems known as *Inverse Problems*, aiming to recover a high quality signal from a degraded version of it. There is a wealth of single image denoising algorithms; a comprehensive review of these techniques can be found in [3] and in [4].

Denoising image sequences extends the above task to handle the temporal dimension as well. Such sequences can be TV broadcast, camcorder files, and more. In many cases, one can assume the noise to be an additive zero-mean white Gaussian noise, as common also in the still image denoising literature. Algorithms for the denoising of image sequences aim to remove the additive noise while utilizing both the spatial and the temporal domains. Such an approach is expected to lead to a gain both in the denoising performance and the computational load, when compared to applying a single image denoising algorithm to each image separately. These desired and expected gains emerge from the high temporal redundancy in image sequences. Indeed, in many cases, image sequences are noisier than single images due to the high capture rate, making the use of the temporal dimension that much more important.

Denoising of video sequences attracted some attention in the past decade, with various suggested algorithms and principles. One suggested approach that utilizes the temporal redundancy is motion estimation [5], [7], [9], [13]. The estimated trajectories are used to filter along the temporal dimension, either in the wavelet domain [5], [7], [13] or the signal domain [9]. Spatial filtering may also be used, with stronger emphasis in areas in which the motion estimation is not as reliable. A similar approach described in [10] detects for each pixel weather it has undergone motion or not. Spatial filtering is applied to each image, and for each pixel with no motion detected, the results of the spatial filtering are recursively averaged with results from previous frames. The method described in [15] employs a similar principle with a fuzzy logic used to replace the binary motion detection.

A different approach to video denoising is treating the image sequence as a 3-D volume, and applying various transforms to this volume in order to attenuate the noise. Such transforms can be a Fourier-wavelet transform [12], an adaptive wavelet transform [11], or a combination of 2-D and 3-D dual-tree complex wavelet transform [8].

A third approach towards video denoising employs spatio-temporal adaptive average filtering for each pixel. The method described in [6] uses averaging of pixels in the neighborhood of the processed pixel, both in the current frame and the previous one. The weights in this averaging are determined according to the similarity of 3×3 patches around the two pixels matched. The method in [14] extends this approach by considering a full 3-D neighborhood around the processed pixel (which could be as large as the entire sequence), with the weights being computed using larger patches and, thus, producing more accurate weights. An adaptive selection of the neighborhood size used for averaging is employed in [16] for obtaining improved results. The current state-of-the-art reported in [17] also finds similar patches in the neighborhood of each pixel; however, instead of a weighted averaging of the centers of these patches, noise attenuation is performed in the transform domain. More on some of these algorithms and their comparison to our proposed scheme is found in Section IV.

In this paper, we explore a method that utilizes sparse and redundant representations for image sequence denoising, extending the work reported in [1] and [2]. In the single image

Manuscript received May 19, 2007; revised June 09, 2008. First published December 2, 2008; current version published December 12, 2008. This work was supported in part by the Israel Science Foundation Grant 796/05. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mario A. T. (G. E.) Figueiredo.

The authors are with the Department of Computer Science, The Technion—Israel Institute of Technology, Haifa 32000, Israel (e-mail: matanpr@cs.technion.ac.il; elad@cs.technion.ac.il).

Digital Object Identifier 10.1109/TIP.2008.2008065

setting, the K-SVD algorithm, as presented in [18] and [19], is used to train a sparsifying dictionary for the corrupted image, forcing each patch in the image to have a sparse representation describing its content. Put in a maximum *a posteriori* probability (MAP) framework, the developed algorithm in [1] and [2] leads to a simple algorithm with state-of-the-art performance for the single image denoising application.

This paper extends the above algorithm by considering 3-D (spatio-temporal) patches, a propagation of the dictionary over time, and averaging that is done on neighboring patches both in space and time. As the dictionary of adjacent frames (belonging to the same scene) is expected to be nearly identical, the number of required iterations per frame can be significantly reduced. Utilizing patches in nearby frames for the denoising process is also examined. All of these modifications lead to substantial benefits both in complexity and denoising performance, outperforming all the recently published video denoising methods.

The structure of the paper is as follows. In Section II, we describe the principles of sparse and redundant representations and their deployment to single image denoising. Section III discusses the generalization to video, discussing various options of using the temporal dimension with their expected benefits and drawbacks. Each proposed extension is experimentally validated. Section IV surveys the literature, describing several leading and competitive video denoising algorithms. A performance comparison of these methods and the one introduced in this paper is given, demonstrating the superiority of the proposed approach. Section V summarizes and concludes the paper.

II. Image Denoising Using Sparsity and Redundancy

A method of denoising images based on sparse and redundant representations is developed and reported in [1] and [2]. In this section, we provide a brief description of this algorithm, as it serves as the foundation for the video denoising we develop in Section III.

A noisy image $\mathbf{Y}$ results from noise $\mathbf{V}$ superimposed on an original image $\mathbf{X}$. We assume the noise $\mathbf{V}$ to be white, zero-mean Gaussian noise, with a known standard deviation σ

$$\mathbf{Y} = \mathbf{X} + \mathbf{V}, \ \text{where} \ \mathbf{V} \sim \mathcal{N}\{\mathbf{0}, \sigma^2\mathbf{I}\}. \tag{1}$$

The basic assumption of the denoising method developed in [1] and [2] is that each image patch (of a fixed size) can be represented as a linear combination of a small subset of patches (*atoms*), taken from a fixed *dictionary*. Using this assumption, the denoising task can be described as an energy minimization procedure. The following functional describes a combination of three penalties to be minimized:

$$f_{Still}\left(\{\boldsymbol{\alpha}_{ij}\}_{i,j}, \mathbf{X}\right) = \lambda\|\mathbf{X}-\mathbf{Y}\|_2^2 + \sum_{ij\in\Omega}\|\mathbf{D}\boldsymbol{\alpha}_{ij} - \mathbf{R}_{ij}\mathbf{X}\|_2^2 + \sum_{ij\in\Omega}\mu_{ij}\|\boldsymbol{\alpha}_{ij}\|_0. \tag{2}$$

The first term demands a proximity between the measured image, $\mathbf{Y}$, and its denoised (and unknown) version $\mathbf{X}$. The second term demands that each patch from the reconstructed image (denoted by[1] $\mathbf{R}_{ij}\mathbf{X}$) can be represented up to a bounded error by a dictionary $\mathbf{D}$, with coefficients $\boldsymbol{\alpha}_{ij}$. The third part demands that the number of coefficients required to represent any patch is small. The values $\boldsymbol{\mu}_{ij}$ are patch-specific weights. Minimizing this functional with respect to its unknowns yields the denoising algorithm.

The choice of $\mathbf{D}$ is of high importance to the performance of the algorithm. In [1], [2] it is shown that training can be done by minimizing (2) with respect to $\mathbf{D}$ as well (in addition to $\mathbf{X}$ and $\boldsymbol{\alpha}_{ij}$). The proposed algorithm in [1] and [2] is an iterative block-coordinate relaxation method, that fixes all the unknowns apart from the one to be updated, and alternates between the following update stages.

1) *Update of the sparse representations* $\{\boldsymbol{\alpha}_{ij}\}$: Assuming that $\mathbf{D}$ and $\mathbf{X}$ are fixed, we solve a set of problems of the form
$$\hat{\boldsymbol{\alpha}}_{ij} = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{D}\boldsymbol{\alpha} - \mathbf{R}_{ij}\mathbf{X}\|_2^2 + \mu\|\boldsymbol{\alpha}\|_0 \tag{3}$$
per each location $[i, j]$. This means that we seek for each patch in the image the sparsest vector to describe it using atoms from $\mathbf{D}$. In [1] and [2], the orthogonal matching pursuit (OMP) algorithm is used for this task [20], [22], [23].
2) *Update the dictionary* $\mathbf{D}$: In this stage, we assume that $\mathbf{X}$ is fixed, and we update one atom at a time in $\mathbf{D}$, while also updating the coefficients in $\{\boldsymbol{\alpha}_{ij}\}_{ij}$ that use it. This is done via a rank-one approximation of a residual matrix, as described in [18], [19], and [24].
3) *Update the estimated image* $\mathbf{X}$: After several rounds of updates of $\{\boldsymbol{\alpha}_{ij}\}_{ij}$ and $\mathbf{D}$, the final output image is computed by fixing these unknowns and minimizing (2) with respect to $\mathbf{X}$. This leads to the quadratic problem
$$\hat{\mathbf{X}} = \arg\min_{\mathbf{X}} \quad \lambda\|\mathbf{X}-\mathbf{Y}\|_2^2 + \sum_{ij}\|\mathbf{D}\boldsymbol{\alpha}_{ij} - \mathbf{R}_{ij}\mathbf{X}\|_2^2 \tag{4}$$
which is solved by a simple weighting of the represented patches with overlaps, and the original image $\mathbf{Y}$.

The improved results obtained by training a dictionary based on the noisy image itself stem from the dictionary adapting to the content of the actual image to be denoised. An added benefit is that the K-SVD algorithm has noise averaging built into it, by taking a large set of noisy patches and creating a small, relatively clean representative set. More recently, the above described algorithm was generalized to handle color image denoising, demosaicing, and inpainting, leading in all these applications to state-of-the-art results [25].

III. Extension to Video Denoising

In this section, we describe, step by step, the proposed extensions to handling image sequences. We also provide experimental results for each proposed extension, validating its efficiency. A description of the test set and a comparison of the overall algorithm to other state-of-the-art methods appear in the next section. To validate the efficiency of the proposed extension steps, we super-impose white Gaussian noise on several sequences, which are then denoised and quantitatively compared.

[1]The matrix $\mathbf{R}_{ij}$ stands for an operator that extracts a patch of fixed size from the image in location $[i, j]$.

The measure of quality of the denoising result $\hat{\mathbf{X}}$ versus the original signal $\mathbf{X}$ is the Peak-Signal-to-Noise-Ratio (PSNR), given by

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2 \cdot p}{||\hat{\mathbf{X}} - \mathbf{X}||_2^2}\right) \quad [\mathrm{dB}]$$

where both signals use the scale 0–255. The PSNR is computed for each image in the sequence, and then averaged over the entire sequence, for assessment of the overall denoising quality.

A. Constructing the Algorithm

Considering the objective function in (2), extending it to handle image sequences might seem to be a simple task. By letting $\mathbf{Y}$ and $\mathbf{X}$ represent the noisy and clean videos respectively (instead of the noisy and clean images), and adding an index t in the range $[1, T]$ to account for the time dimension, we arrive at a desired penalty term that contains all the forces described in the single image denoising setting. This formulation forms one MAP energy function for the entire sequence

$$f_{\mathrm{Video}}^{All}(\{\boldsymbol{\alpha}_{ijt}\}_{ijt}, \mathbf{X}, \mathbf{D}) = \lambda||\mathbf{X}-\mathbf{Y}||_2^2 + \sum_{ij\in\Omega}\sum_{t=1}^{T}\mu_{ijt}||\boldsymbol{\alpha}_{ijt}||_0 + \sum_{ij\in\Omega}\sum_{t=1}^{T}||\mathbf{D}\boldsymbol{\alpha}_{ijt} - \mathbf{R}_{ijt}\mathbf{X}||_2^2. \quad (5)$$

The term $\mathbf{R}_{ijt}\mathbf{X}$ extracts a patch of a fixed size from the volume $\mathbf{X}$ in time t and spatial location $[i, j]$. This patch may be 3-D, in general, this way exploiting the temporal axis to our benefit.

Minimizing this functional with respect to its unknowns generates a single dictionary for the entire sequence and cleans all the images at once with such a dictionary. The transition to three dimensions appears in the finer details of the algorithm. The patches are transformed into 3-D ones, in the sense that they can contain pixels from more than one image. A 3-D patch is created by taking a block around the pixel (i, j, t) that extends by $(\Delta i, \Delta j, \Delta t)$ in each axis respectively. This makes the patch symmetrical around the center image and, therefore, is not causal.[2] This structure also requires paying special attention to the dictionary initialization. In our tests, the basis for the initial dictionary is the same 2-D overcomplete DCT dictionary. Each atom is then replicated $(2\Delta t + 1)$ times to create a 3-D atom of the wanted temporal width.

As already mentioned, in the penalty term in (5) all the patches in the sequence are used for training a single dictionary, that is then applied to the entire sequence. However, training a single dictionary for the entire sequence is problematic; The scene is expected to change rapidly, and objects that appear in one frame might not be there five or ten frames later. This either means that the dictionary will suit some images more than others, or that it would suit all of the images but only moderately so. Obtaining state-of-the-art denoising results requires better adaptation.

[2]Causality can be enforced, but was found to lead to a slightly inferior performance.

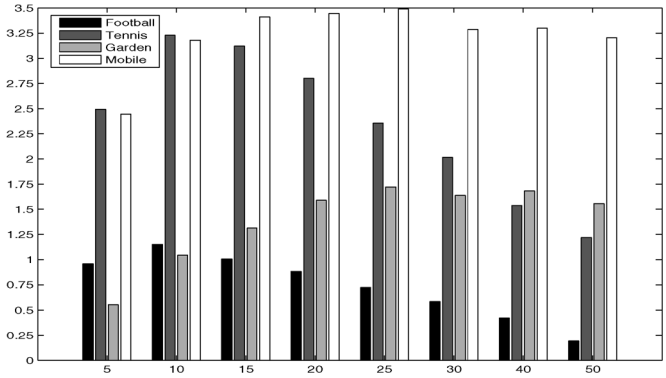

Fig. 1. PSNR gain (difference in decibels relative to the 2-D atoms' method) achieved by using 3-D atoms versus 2-D atoms.

An alternative approach could be proposed by defining a locally temporal penalty term, that on one hand allows the dictionary to adapt to the scene, and on the other hand, exploits the benefits of the temporal redundancy. A natural such attempt is rewriting the penalty in (5) for each image separately

$$f_{\mathrm{Video}}^{(t)}(\{\boldsymbol{\alpha}_{ij}\}_{ij}, \mathbf{X}_t, \mathbf{D}_t) = \lambda||\mathbf{X}_t - \mathbf{Y}_t||_2^2 + \sum_{ij\in\Omega}\mu_{ij}||\boldsymbol{\alpha}_{ij}||_0 + \sum_{ij\in\Omega}||\mathbf{D}_t\boldsymbol{\alpha}_{ij} - \mathbf{R}_{ijt}\mathbf{X}||_2^2 \quad (6)$$

defined for $t = 1, 2, \ldots, T$.

The temporal repetitiveness of the video sequence can be further used to improve the algorithm. As consecutive images $\mathbf{X}_t$ and $\mathbf{X}_{t-1}$ are similar, their corresponding dictionaries are also expected to be similar. This temporal coherence can help speed-up the algorithm. Fewer training iterations are necessary if the initialization for the dictionary $\mathbf{D}_t$ is the one trained for the previous image.

Returning to the first step of migrating from 2-D to 3-D patches, how important is this change? To answer this question we test the effects of 2-D and 3-D atoms on the performance of the overall video denoising algorithm (with a temporally adaptive and propagated dictionaries). The results of such a comparison appears in Fig. 1. For the 3-D case, one set of parameters (noise-level dependent) is used for all movies. The 3-D atoms used in these tests are 5 images wide (two images away in each direction). For the 2-D case, it is difficult to find one set of parameters that does justice to all movies. We, therefore, use the optimal set found for each movie, as this does not change the conclusions drawn from this comparison.

The performance gain achieved by using 3-D atoms is very noticeable. A better understanding of the reasons the 3-D atoms outperform the 2-D ones can be gained by looking at the content of the dictionary. Fig. 2 shows several atoms from the trained 3-D dictionary, for image #10 of the "garden" sequence (several frames from this sequence appear at the end of this section). This sequence has a camera motion to the right, so the entire scene moves to the left. The described motion can be seen clearly in the atoms. The central part of the atom (coming from the current image) has moved to the left compared to the top part (coming

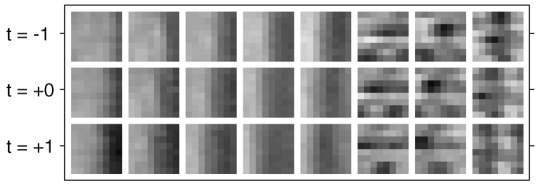

Fig. 2. Several (8) 3-D atoms (each of size $8 \times 8 \times 5$, but only the temporal center is shown) trained for image #10 of the garden sequence. Each vertical column is one atom.

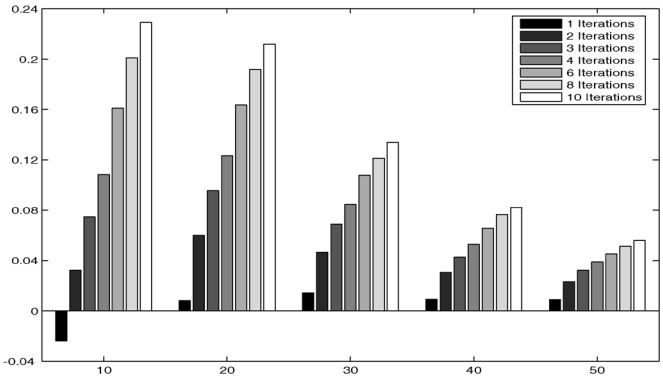

Fig. 3. PSNR gain (average of all sequences) by propagating the dictionary and using the specified number of training iterations for each image. The reference option is with no propagation, and 15 K-SVD iterations.

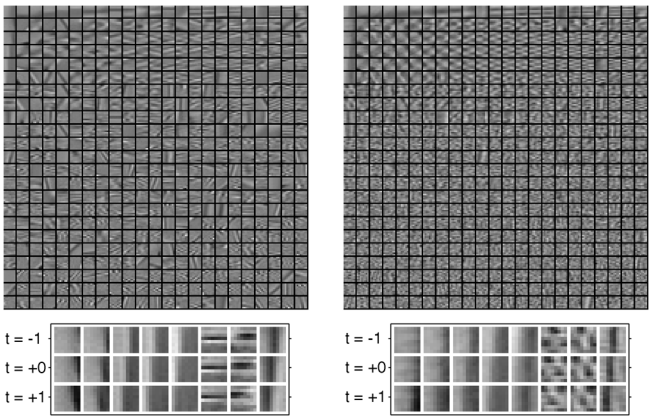

Fig. 4. Dictionaries trained by propagating (left) and not propagating (right) the dictionary between images. Top: Central part (time-wise) of each dictionary. Bottom: Several enlarged atoms, showing the three center temporal layers.

from the previous image). In the same manner, the bottom part (coming from the next image) has moved to the left relative to the center part. The question of what happens to the dictionary when the motion is not global naturally arises. In such cases, the dictionary has several atoms reflecting each of the patterns and motions.

To gauge the possible speed-up and improvement achieved by temporally adaptive and propagated dictionary, we test the required number of iterations to obtain similar results to the non-propagation alternative. Fig. 3 presents the results of such an experiment. Several options for the number of training iterations that follow the dictionary propagation are compared to the non-propagation (using 15 training iterations per image) option.

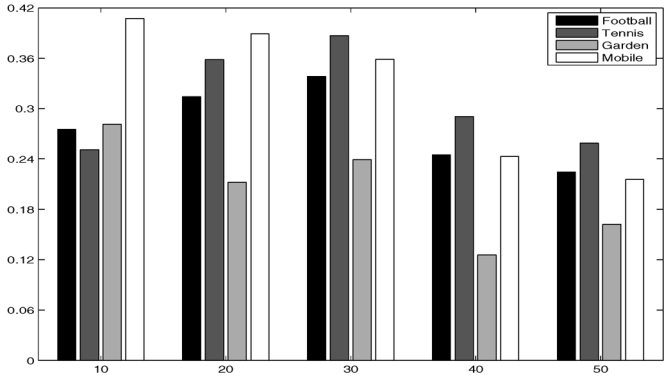

Fig. 5. PSNR gain achieved by also using patches that are one image away for training and cleaning.

The clear conclusion from these results is that propagation is crucial and leads to improved denoising performance.[3] More insight to the reasons for such an improvement can be seen in Fig. 4, that shows the dictionaries trained for frame #30 of the "garden" sequence. The left dictionary is the one trained after propagating the dictionary (from image #10), using 4 training iterations for each image. The right part of the figure shows the trained dictionary from the DCT using 15 training iterations. This comparison shows that propagation of the dictionary leads to a cleaner version with clearer and sharper texture atoms. These benefits are attributed to the memory induced by the propagation. Indeed, when handling longer sequences, we expect this memory feature of our algorithm to further benefit in denoising performance. This was verified in tests on longer sequences.

A second conclusion is that the number of training iterations should not be constant, but rather depend on the noise level. It appears that the less noisy the sequence, the more training iterations are needed for each image. In higher noise levels, adding more iterations hardly results in any denoising improvement.

So far, we discussed the use of 3-D atoms and a temporally adaptive dictionaries. However, in the formulation written in (6), only patches centered in the current image are used for training the dictionary and cleaning the image. In the global temporal term as in (5), all the patches in the sequence were used for these tasks. A compromise between temporal locality and exploiting the temporal redundancy is again called for. This compromise is achieved by using patches centered in a limited number of neighboring images of the image currently denoised, both in training and cleaning. Introducing this into the penalty term in (6) leads to the modified version

$$\begin{aligned} f_{Video}^{(t\pm\Delta t)}&\left(\{\boldsymbol{\alpha}_{ijk}\}_{ijk}, \mathbf{X}_t, \mathbf{D}_t\right) \\ &= \lambda\|\mathbf{X}_t - \mathbf{Y}_t\|_2^2 + \sum_{ij\in\Omega}\sum_{k=t-\Delta t}^{t+\Delta t}\mu_{ijk}\|\boldsymbol{\alpha}_{ijk}\|_0 \\ &\quad + \sum_{ij\in\Omega}\sum_{k=t-\Delta t}^{t+\Delta t}\|\mathbf{D}_t\boldsymbol{\alpha}_{ijk} - \mathbf{R}_{ijk}\mathbf{X}\|_2^2 \end{aligned} \tag{7}$$

defined for $t = 1, 2, \ldots, T$.

[3]In a real system, employing dictionary propagation requires some scene-cut detection algorithm to reset the dictionary when the scene changes at once.

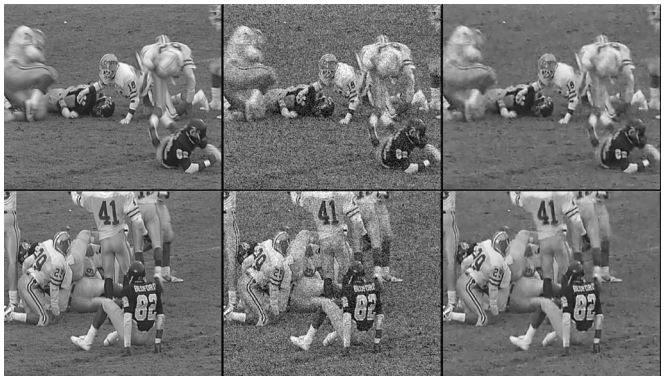

Fig. 6. Football sequence (frames 15 and 70) with $\sigma = 30$. Left: Original frame. Middle: Noisy frame. Right: Cleaned frame.

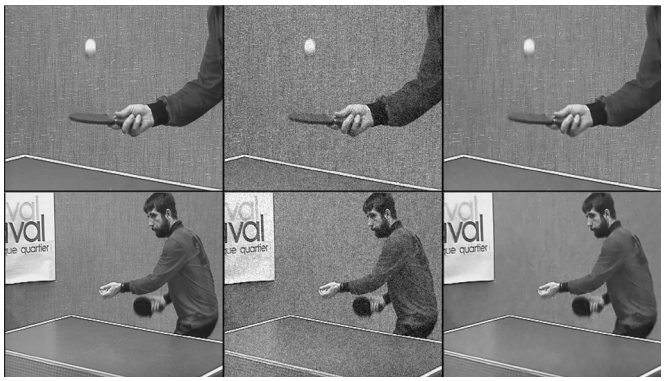

Fig. 7. Tennis sequence (frames 15 and 60) with $\sigma = 20$. Left: Original frame. Middle: Noisy frame. Right: Cleaned frame.

The effectiveness of the extended training set can be seen by observing Fig. 5, in which we show the gain (or loss) in PSNR achieved by also using patches centered one image away (i.e., the patches are taken from three frames). All the tests in this experiment are done on one quarter of the patches in the spatio-temporal region (chosen randomly) so as to lead to a reduced complexity algorithm.

It is visible from the graph that using an extensive set indeed results in an improved performance. Tests taking patches also from two images away were also run; however, there was no significant advantage in performance to justify this additional computational burden.

B. Overall Algorithm and Parameter Selection

The penalty term in (7) is the penalty term we target in the algorithm that follows. The algorithm for minimizing this functional is founded on the same principles as the ones described in Section II, with the obvious modifications due to the 3-D nature of the treatment done here. Three visual examples of the denoising results can be seen in Figs. 6–8.

We have run many tests to tune the various parameters of the proposed algorithm, which have a crucial effect on the overall denoising performance. These tests have resulted in a selection of a single set of parameters (as a function of noise level). In conjunction with the previously described experiments, this set of parameters includes 3-D atoms (which are five images wide),

Fig. 8. Garden sequence (frames 10 and 20) with $\sigma = 50$. Left: Original frame. Middle: Noisy frame. Right: Cleaned frame.

propagation of the dictionary with the number of training iterations being noise-level dependent, and a training set which extends one image in each temporal direction. During the experimentation with other parameters, we have found, for example, that higher noise levels require that the spatial size of the blocks is slightly smaller than at low noise levels.

We have also found that at high noise levels, the redundancy factor (the ratio between the number of atoms in the dictionary to the size of an atom) should be smaller. At high noise levels, obtaining a clean dictionary requires averaging of a large number of patches for each atom. This is why only a relatively small number of atoms is used. At low noise levels, many details in the image need to be represented by the dictionary. Noise averaging takes a more minor role in this case. This calls for a large number of atoms, so they can represent the wealth of details in the image.

A comparison between the final 3-D algorithm and the original single image algorithm can be seen in Table IV, which appears in the next section.

C. Complexity of the Algorithm

The algorithm is divided into two parts: (i) dictionary training and (ii) image cleaning. The dictionary training is an iterative process, of repeatedly running sparse coding (OMP) followed by a dictionary update (SVD step). The image cleaning is composed of a simple per-pixel averaging. In order to analyze the complexity of the algorithm, we present the following notations: n—the number of pixels in an atom; d—the number of atoms in the dictionary; ℓ—the average number of atoms used in the representation of a patch; p—the number of pixels in one frame; $2\Delta t + 1$—the number of frames taken into account in the training and denoising; and J—the number of training iterations.

The sparse-coding stage—solving the problem posed in (3) using the OMP algorithm—requires $n \cdot d \cdot \ell$ operations for one patch [20]. Applying this to each of the $(2\Delta t+1)p$ patches in the spatio-temporal window requires $n \cdot d \cdot \ell \cdot (2\Delta t+1) \cdot p$ operations. The update of the dictionary requires $n \cdot \ell \cdot (2\Delta t+1) \cdot p$ operations [18], [19]. The image cleaning is done by a simple averaging of patches, requiring $n \cdot (2\Delta t+1) \cdot p$ operations. Since there are J iterations of sparse coding and dictionary update, and one final

estimate of the output image, the overall algorithm complexity is given by

$$\text{Complexity} = [J(d+1)\ell + 1] \cdot n \cdot (2\Delta t + 1) \cdot p \quad \text{Operations/Frame}.$$

Let us illustrate this complexity for a nominal case: The value of ℓ is noise-level dependant, being ≈ 1 for $\sigma = 10$–25. The number of training iterations J is 2 when propagating the dictionary. Assuming that we process patches of size $n = 125$ pixels each ($5 \times 5 \times 5$ patches) using a dictionary with $d = 300$ atoms, over a window of one frame ($\Delta t = 0$), there are $\approx$75,000 operations per pixel. This very demanding algorithm is essentially such because of the need to multiply the patches by the dictionary—a matrix of size $d \times n$.

Reducing the complexity of the algorithm is a necessary step in turning the algorithm into a reasonable one. There are several methods of reducing its complexity.

1) The image can be divided into several parts (with small overlaps), and a smaller dictionary (i.e., with a smaller number of atoms) can be trained for each part. This saves computations, as each patch considers less atoms, thus reducing the value of the effective d in the above formula. Further, one part's dictionary can be used for initialization for all other parts, saving more computations. Several tests using this approach indicate that not only does this lead to a speedup factor of 4–10, it actually leads to an improvement (approx. 0.1–0.2 dB in our tests) in the denoising performance. We believe that delicate parameter tuning for this approach will result in a more substantial improvement, especially in weak noise scenarios.
2) Choosing nonoptimal values for the parameters of the algorithm is another way to control complexity. One of the parameters is the temporal extent (Δt), i.e., from which images are patches drawn for the training process (it is important not to confuse with the temporal extent of the atoms, which should remain 3-D). Using only patches from the current image (and not from neighboring images) cuts complexity by a factor of 3 and costs approx. 0.2–0.4 dB. Reducing the overlap between patches, by using only every other patch in each axis, reduces complexity by a factor of 4, and costs only around 0.1 dB. Combining them causes a reduction of more than 0.5 dB, probably because too few patches are left for use.
3) Running all iterations except maybe the last one (which is also the cleaning iteration) on a subset of the patches does not cause noticeable degradation in performance. When using $J = 2$, this can gain almost a factor of 2 in the overall complexity.
4) The core operation of matrix multiplication can be replaced by an approximation of it (e.g., using singular value decomposition (SVD) on the dictionary [24]). We have not explored this option in detail and, thus, cannot report on its effectiveness.
5) The facts that OMP is done independently on each patch, and the K-SVD independently on each atom, lend this algorithm easily to a parallel implementation on any number of processors, again leading to a substantial speedup. Since 8-CPU structures or equivalent FPGA are currently available at affordable prices, and due to the parallel nature of almost the entire algorithm, this can be viewed as a viable option for reducing the overall run-time by about one order of magnitude.

To summarize, by making slight adaptations to the algorithm (e.g., dividing the image to parts, controlling the overlap between patches, and more), 1–2 orders of magnitude in the number of computations can be gained without a noticeable drop in performance. This leads to a rough estimate of 2,000 operations per pixel, which is definitely more reasonable. Note that another one order of magnitude can be gained by a parallel implementation.

The entire simulations described in this and the next section were run with a nonoptimized Matlab implementation of the proposed algorithm on a 2.4-GHz Pentium with 2-Gb RAM, with the required variations for each test. Using our implementation, processing a CIF (approx. 280×360) frame with this Matlab implementation requires 5–120 s, depending on the noise level (higher noise level requires less time) and the content of the scene (more textured scenes requiring more time).

IV. Comparison to State-of-the-Art

A. Overview of Other Methods

We compare the proposed algorithm to four methods that have been shown to display state-of-the-art results and are reported in [13], [14], [16], and [17]. We do not provide a comprehensive comparison to the algorithm reported in [6] since it is a simplified version of the one described in [14].

The method described in [13] operates fully in the wavelet domain. Motion estimation and adaptive temporal filtering (along the estimated trajectories) are applied recursively, followed by an intraframe spatially adaptive filter. Two types of motion reliability measures are estimated. One is a reliability measure per orientation, applied in the motion estimation stage. The other is a reliability measure per wavelet band, effecting the parameters of the temporal filter. The subsequent spatial filtering is designed to have an increased effect where the temporal filtering had been less effective due to low reliability.

The Nonlocal Means (NL-Means) algorithm reported in [3] and [14] takes an alternative approach to the problem. Instead of an explicit motion estimation for each patch, all the patches in its 3-D neighborhood are considered and their center pixels are averaged. Each patch is weighted according to its similarity to the center patch, making the motion-estimation a fuzzy and implicit one. Instead of computing a single motion vector for each patch, several possible vectors are allowed to co-exist, each with a different probability. As contributing patches may also appear within the same image, the interpretation of this approach as fuzzy motion estimation is inaccurate; still, it provides some intuition into the success of this approach. This approach focuses on the fusion of noisy estimates rather than obtaining accurate motion estimation.

The method described in [16] extends the NL-Means approach. Statistical measures are used for optimal adaptive selection of the neighborhood size for each pixel. Furthermore,

TABLE I
RESULTS OF THE PROPOSED ALGORITHM COMPARED TO THOSE REPORTED IN [13]. THE CHOSEN SEQUENCES AND NOISE POWERS ARE THOSE REPORTED IN [13]

Test Sequence	Input Noise Level	Results from [13] [dB]	Proposed Method Results [dB]	Difference [dB]
Garden	10	30.86	32.22	+1.36
	15	28.24	29.78	+1.54
	20	26.46	28.08	+1.62
Tennis	10	32.08	33.78	+1.70
	15	29.46	31.65	+2.19
	20	28.29	30.18	+1.89
Salesman	10	35.84	37.95	+2.11
	15	33.91	35.17	+1.26
	20	32.37	33.33	+0.96

TABLE II
RESULTS OF THE PROPOSED ALGORITHM COMPARED TO THOSE REPORTED IN [16]. THE CHOSEN SEQUENCES AND NOISE POWERS ARE THOSE REPORTED IN [16]

Test Sequence	Input PSNR [dB]	[16] Results [dB]	Our Results [dB]	Difference [dB]
Salesman	28	35.13	37.91	+2.78
	24	32.60	35.59	+2.99
Garden	28	31.33	32.13	+0.80
Miss America 1	28	39.39	40.49	+1.10
Suzie	28	37.07	37.96	+0.89
	24	35.11	35.95	+0.84
Trevor	28	36.68	38.10	+1.42
	24	34.79	35.97	+1.18
Foreman	28	34.94	37.86	+2.92
	24	32.90	35.86	+2.96

the parameters for computing the weights assigned to each pixel in the neighborhood are also adaptively selected. These powerful tools result in improved denoising performance compared to the original NL-Means approach.

The VBM3D, reported in [17], also uses a multitude of patches in the 3-D neighborhood of each pixel for attenuating the noise. However, the patches are used in a different manner. The most similar patches in the neighborhood are collected and stacked into a 3-D array. A 3-D wavelet transform is then applied, with hard-thresholding used for noise suppression. After the inverse transform is applied, the patches are returned to their original locations, and averaged. A second iteration follows, with Wiener filtering used to improve denoising results.

We now turn to present a comprehensive performance comparison between these four methods and the proposed algorithm. Note that since the above papers chose different video sequences to test on, we provide several groups of tests, to address each.

B. Comparison Results

The comparison to the method reported in [13] was done on the sequences appearing in that paper (and also found in the first author's website). The mean PSNR results of our proposed method, the results of [13], and the differences between them (for frames 5–35 of each sequence) all appear in Table I. Averaging over all these tests, the proposed method outperforms the one reported in [13] by 1.63 dB, and specifically on each and every test.

We now turn to compare the proposed method to the work reported in [16], which displayed superior denoising results relative to other methods it was compared to. We synthesized the same experiments as those in [16], and report the results in Table II. Again, it is clear that the proposed method performs better in these tests with an average gain of 1.79 dB.

We also compare the proposed method to the results of the classic NL-Means [3], [14] and to the current benchmark in video denoising, the VBM3D [17]. The tests were run on a set of four different image sequences—"Football," "Tennis," "Flower Garden," and "Mobile." Each of the four test sets is superimposed with synthetic white Gaussian noise, using noise levels $\sigma = 5$, 10, 15, 20, 25, 30, 35, 40, and 50. The translation between noise level and mean PSNR of the noisy sequences appears in Table III, as the clipping of out-of-range gray-values causes some variation, especially noticed in the strong noise cases.

For the NL-Means algorithm, we used our implementation to obtain denoising results for the test sequences. For fairness, we have varied the parameters, searching for the optimal configuration. It is worth noting that there was no one set of parameters fitting all sequences at a defined noise level. Instead, we report the results when all the parameters have been optimized for each specific test, keeping in mind that this gives the NL-Means algorithm an advantage in these comparisons.

For the VBM3D, the authors of [17] were very kind to provide us with an implementation of their algorithm (which later was made available in their website). We compare in Table IV the results obtained by four algorithms: VBM3D [17], NL-Means [14], the original K-SVD denoising applied on single images [1], and the proposed algorithm that is proposed in this paper.

Averaging the above results, we get an average performance (from the best downwards) of 29.23 dB for our method,

TABLE IV
COMPARISON OF THE DENOISING RESULTS OF SEVERAL METHODS ON A NUMBER OF TEST SEQUENCES AND NOISE LEVELS. TOP LEFT: VBM3D; TOP RIGHT: NL-MEANS; BOTTOM LEFT: K-SVD SINGLE IMAGE; AND BOTTOM RIGHT: THE PROPOSED ALGORITHM. THE BEST RESULT FOR EACH SET IS WRITTEN IN BOLD. RESULTS ARE FOR IMAGES 10–20 OF EACH SET, USING OTHER IMAGES (FOR TEMPORAL FILTERING) AS NECESSARY

σ	Football		Tennis		Garden		Mobile		MeanPSNR	
5	36.96	35.95	**38.32**	37.12	**36.84**	36.03	36.81	36.22	37.23	36.33
	36.03	**37.42**	35.45	38.16	35.74	36.73	35.52	**38.55**	35.69	**37.72**
10	32.87	31.82	**34.60**	32.05	**32.38**	31.47	32.56	31.72	33.10	31.77
	31.77	**33.29**	30.89	34.33	30.72	32.16	30.55	**34.22**	30.98	**33.50**
15	30.74	29.61	**32.46**	29.56	**30.16**	29.04	30.27	29.43	30.91	29.41
	29.66	**30.96**	28.81	32.10	28.03	29.69	27.82	**31.62**	28.58	**31.09**
20	29.22	28.14	**30.76**	28.14	**28.58**	27.56	28.58	27.73	29.29	27.89
	28.25	**29.39**	27.64	30.51	26.25	28.03	25.97	**29.78**	27.03	**29.43**
25	28.07	27.12	29.10	27.35	**27.33**	26.38	27.19	26.37	27.92	26.81
	27.24	**28.24**	26.83	**29.32**	24.89	26.80	24.52	**28.35**	25.87	**28.18**
30	27.19	26.52	27.86	26.68	**26.26**	25.33	25.98	25.36	26.82	25.97
	26.47	**27.23**	26.31	**28.43**	23.84	25.56	23.43	**26.87**	25.01	**27.02**
35	26.40	25.80	26.98	26.29	**25.39**	24.43	25.01	24.44	25.95	25.24
	25.81	**26.50**	25.83	**27.75**	22.96	24.75	22.49	**25.91**	24.27	**26.23**
40	25.84	24.97	26.45	25.93	**24.49**	23.51	24.04	23.60	25.20	24.50
	25.22	**25.86**	25.52	**27.22**	22.20	24.02	21.75	**25.11**	23.67	**25.55**
50	**24.81**	24.11	25.79	25.34	22.48	22.16	21.71	21.93	23.70	23.38
	24.12	24.73	24.92	**26.34**	21.03	**22.80**	20.38	**23.64**	22.61	**24.38**

TABLE III
PSNR OF NOISY SEQUENCES FOR EACH SEQUENCE AND NOISE LEVEL COMBINATION. THE DIFFERENCE IS DUE TO THE OUT-OF-RANGE VALUES

Noise Sigma	Football	Tennis	Garden	Mobile
5	34.1446	34.1596	34.1623	34.1785
10	28.1398	28.1263	28.1567	28.1715
15	24.6191	24.6201	24.7063	24.6875
20	22.1287	22.1398	22.2673	22.2590
25	20.2348	20.2096	20.3964	20.4008
30	18.6824	18.6517	18.8939	18.9238
35	17.3685	17.3242	17.6543	17.6862
40	16.2616	16.2014	16.5809	16.6440
50	14.4705	14.3856	14.8762	14.9481

28.9 dB for the VBM3D, 27.92 dB for the NL-Means, and, finally, 27.08 dB for the single-frame K-SVD algorithm. The proposed method is slightly favorable compared to the VBM3D, obtaining better mean PSNR at every noise level. We also note that these results mean that the proposed extension of [1] and [2] to handle video yields about 2 dB better results on average than the single image method. This comes only to prove the necessity and potential of using the temporal axis in video denoising.

As a final experiment, we compare the visual quality of the results produced by the three algorithms—the VBM3D, the NL-Means, and the proposed method. This comparison appears in Fig. 9, along with the original high-quality image. We deliberately show results for very strong noise ($\sigma = 40$), since all these methods are very effective and low-noise cases appear to be near-perfect and with only delicate differences.

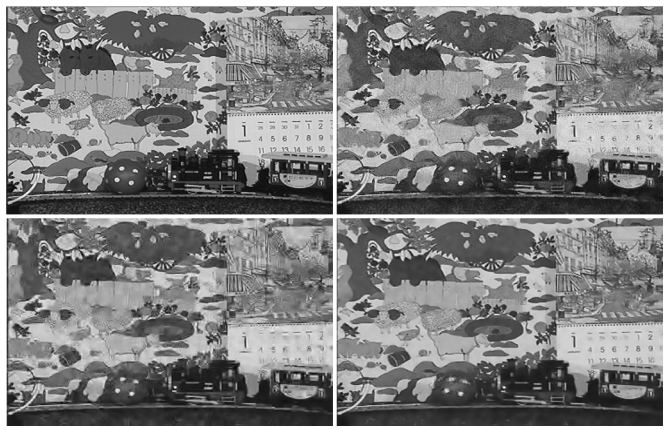

Fig. 9. Visual comparison of denoising results for one image of the Mobile sequence with noise level 40. Top Left: Original. Top Right: NL-Means. Bottom Left: VBM3D. Bottom Right: Proposed Method.

V. CONCLUSION

In this paper, we propose an image sequence denoising algorithm based on sparse and redundant representations. This algorithm is based on the single image denoising algorithm introduced in [1] and [2]. The extension of this basic algorithm to handle image sequences is discussed both on the mathematical level and in practical terms. Three extensions are proposed: the use of spatio-temporal (3-D) atoms, dictionary propagation coupled with fewer training iterations, and an extended patch-set for dictionary training and image cleaning. All these extensions are thoroughly tested on an extensive set of image sequences and noise levels, and are found to dramatically improve denoising performance. The proposed algorithm is also compared to other state-of-the-art methods, and shown to produce comparable or favorable results.

ACKNOWLEDGMENT

The authors would like to thank D. Rusanovskyy, K. Dabov, and Prof. K. Egiazarian (the authors of [17]) for their willingness to provide a working implementation of the SW3D algorithm, especially K. Dabov, who provided the source code of the improved SW3D (coined VBM3D) and the test sequences used to test it on. They would also like to thank J. Boulanger, C. Kervrann, and P. Bouthemy (the authors of [16]), for their

willingness to help in comparing the denoising methods, as well as the reviewers of this paper for various suggestions and ideas that helped improved the manuscript.

REFERENCES

[1] M. Elad and M. Aharon, "Image denoising via learned dictionaries and sparse representation," presented at the Int. Conf. Computer Vision and Pattern Recognition, New York, Jun. 17–22, 2006.

[2] M. Elad and M. Aharon, "Image denoising via sparse and redundant representation over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[3] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms with a new one," *Multiscale Model. Simul.*, vol. 4, no. 2, pp. 490–530, 2005.

[4] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Trans. Image Process.*, vol. 12, no. 11, pp. 1338–1351, Nov. 2003.

[5] V. Zlokolica, A. Pizurica, and W. Philips, "Recursive temporal denoising and motion estimation of video," presented at the Proc. Int. Conf. Image Processing, Singapore, Oct. 2004.

[6] V. Zlokolica, A. Pizurica, and W. Philips, "Video denoising using multiple class averaging with multiresolution," in *Proc. Lecture Notes in Compute Science (VLVB03)*, 2003, vol. 2849, pp. 172–179.

[7] F. Jin, P. Fieguth, and L. Winger, "Wavelet video denoising with regularized multiresolution motion estimation," *EURASIP J. Appl. Signal Process.*, vol. 2006, pp. 01–11, 2006.

[8] I. Selesnick and K. Y. Li, "Video denoising using 2D and 3D dual-tree complex wavelet transforms," presented at the Wavelet Applications in Signal and Image Processing X (SPIE), San Diego, CA, Aug. 2003.

[9] R. Dugad and N. Ahuja, "Video denoising by combining Kalman and Wiener estimates," presented at the Proc. Int. Conf. Image Processing, Kobe, Japan, Oct. 1999.

[10] A. Pizurica, V. Zlokolika, and W. Philips, "Combined wavelet domain and temporal video denoising," presented at the IEEE Int. Conf. Advanced Video and Signal Based Surveillance (AVSS), Jul. 2003.

[11] N. M. Rajpoot, Z. Yao, and R. G. Wilson, "Adaptive wavelet restoration of noisy video sequences," presented at the Proc. Int. Conf. Image Processing, Singapore, Oct. 2004.

[12] R. G. Wilson and N. M. Rajpoot, "Image volume denoising using a fourier-wavelet basis," presented at the 6th Baiona Workshop on Signal Processing in Communications, Baiona, Spain, Sep. 2003.

[13] V. Zlokolica, A. Pizurica, and W. Philips, "Wavelet-domain video denoising based on reliability measures," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 8, pp. 993–1007, Aug. 2006.

[14] A. Buades, B. Coll, and J. M. Morel, "Denoising image sequences does not require motion estimation," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, Sep. 2005, pp. 70–74.

[15] V. Zlokolica, M. D. Geyer, S. Schulte, A. Pizurica, W. Philips, and E. Kerre, "Fuzzy logic recursive change detection for tracking and denoising of video sequences," in *Proc. Image and Video Communications and Processing (SPIE)*, Mar. 2005, vol. 5685, pp. 771–782.

[16] J. Boulanger, C. Kervrann, and P. Bouthemy, "Space-time adaptation for patch based image sequence restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 1096–1102, Jun. 2007.

[17] K. Dabov, A. Foi, and K. Egiazarian, "Video denoising by sparse 3D transform-domain collaborative filtering," presented at the Eur. Signal Processing Conf., Poznan, Poland, Sep. 2007.

[18] M. Aharon, M. Elad, and A. M. Bruckstein, "On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them," *J. Lin. Algebra Appl.*, vol. 416, pp. 48–67, Jul. 2006.

[19] M. Aharon, M. Elad, and A. M. Bruckstein, "The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[20] S. Mallat and Z. Zhang, "Matching pursuit in a time-frequency dictionary," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[21] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.

[22] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[23] D. L. Donoho, M. Elad, and V. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Trans. Inf. Theory*, vol. 52, pp. 6–18, Jan. 2006.

[24] G. H. Golub and C. F. Van-Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: John Hopkins Univ. Press, 1996.

[25] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. Image Process.*, to be published.

Matan Protter received the B.Sc. degree in mathematics, physics, and computer sciences from the Hebrew university of Jerusalem, Israel, in 2003. He is currently pursuing the PhD. degree in computer sciences from the Computer Sciences Department, The Technion—Israel Institute of Technology, Haifa.

Since 2003, he has concurrently served in the Israeli Air Force, as a senior R&D engineer. Matan's research interests are in the area of image processing, focusing on example-based image models and their application to various inverse problems.

Michael Elad (SM'08) received the B.Sc (1986), M.Sc.(supervision by Prof. D. Malah, 1988), and D.Sc. (supervision by Prof. A. Feuer 1997) degrees from the Department of Electrical engineering, The Technion—Israel Institute of Technology, Haifa.

From 1988 to 1993, he served in the Israeli Air Force. From 1997 to 2000, he was with Hewlett-Packard Laboratories as an R&D engineer. From 2000 to 2001, he headed the research division at Jigami corporation, Israel. From 2001 to 2003, he spent a postdoctorate period as a research associate with the Computer Science Department, Stanford University (SCCM Program), Stanford, CA. In September 2003, he returned to The Technion, assuming a tenure-track assistant professorship position in the Department of Computer Science. In May 2007, he was tenured to an associate professorship. He currently works in the field of signal and image processing, specializing, in particular, in inverse problems, sparse representations, and overcomplete transforms.

Prof. Elad received The Technion's best lecturer award five times (1999, 2000, 2004, 2005, and 2006), he is the recipient of the Solomon Simon Mani award for excellence in teaching (2007), and he is also the recipient of the Henri Taub Prize for academic excellence (2008). He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING.