MULTISCALE SPARSE IMAGE REPRESENTATION WITH LEARNED DICTIONARIES

Julien Mairal and Guillermo Sapiro
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455

Michael Elad
Department of Computer Science
Technion, Haifa 32000, Israel

ABSTRACT

This paper introduces a new framework for learning multiscale sparse representations of natural images with overcomplete dictionaries. Our work extends the K-SVD algorithm [1], which learns sparse single-scale dictionaries for natural images. Recent work has shown that the K-SVD can lead to state-of-the-art image restoration results [2, 3]. We show that these are further improved with a multiscale approach, based on a Quadtree decomposition. Our framework provides an alternative to multiscale pre-defined dictionaries such as wavelets, curvelets, and contourlets, with dictionaries optimized for the data and application instead of pre-modelled ones.

Index Terms— Image Restoration, Denoising, Multiscale, Sparsity

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 a "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 processing applications [1, 2, 3].

Another important assumption, commonly and successfully used in image processing, is the existence of multiscale features in images. Trying to design the best multiscale dictionary which fulfils a sparsity criterion has been a major challenge. Such attempts include the wavelets, curvelets, contourlets, wedgelets, bandlets, and steerable wavelets (see for example [4] and references therein). These methods lead to many effective algorithms in image processing, e.g., image denoising [5].

In [1] the K-SVD is proposed for learning a single-scale dictionary for sparse representation of 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 [2]. This has been recently extended to color images, with state-of-the-art results in denoising, inpainting, and demosaicing applications [3]. In this paper, we extend the basic K-SVD work, providing a framework for learning multiscale and sparse representation of images. In addition to the presentation of the new framework, we apply it to denoising, obtaining results that outperform reference works such as [2, 5, 6] and competes favorably with the most recent and state-of-the-art in this field [7].

The task of learning a multiscale dictionary has been addressed in [8] 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 [2]. This explains the superior performance we obtain.

Work partially supported by NSF, ONR, NGA, DARPA, and the McKnight Foundation.

2. THE SINGLE-SCALE K-SVD DENOISING ALGORITHM

In this section, we briefly review the main ideas of the K-SVD framework for sparse image representation and denoising. The reader is referred to [1, 2, 3] for more 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 reconstruct image. This algorithm (shown in Figure 1) can be described as the minimization of an energy:

$$\begin{aligned}\left\{\hat{\alpha_{ij}}, \hat{\mathbf{D}}, \hat{\mathbf{x}}\right\} &= \arg\min_{\mathbf{D}, \alpha_{ij}, \mathbf{x}} \lambda ||\mathbf{x} - \mathbf{y}||_2^2 \\ &+ \sum_{i,j} \mu_{ij} ||\alpha_{ij}||_0 + \sum_{ij} ||\mathbf{D}\alpha_{ij} - \mathbf{R}_{ij}\mathbf{x}||_2^2 .\end{aligned} \quad (1)$$

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 vectors $\hat{\alpha}_{ij} \in \mathbb{R}^k$ are the sparse representations 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 1):

Sparse Coding Step: This is performed with an Orthogonal Matching Pursuit (OMP) [9], which proves to be very efficient for diverse approximation problems [10]. The approximation stops when the residual reaches a sphere of radius $\sqrt{n}C\sigma$ representing the probability distribution of the noise. More on this is found in [3].

Dictionary Update: This is a sequence of one-rank approximation problems that update both the dictionary atom and the sparse representations that use it.

Reconstruction: The last step is a simple averaging between the patches approximations and the noisy image. The denoised image is $\hat{\mathbf{x}}$. Equation (4) emerges directly from the energy minimization in Equation (2).

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 goal of this paper.

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 $\hat{\alpha}_{ij} \in \mathbb{R}^{1\times k}$ for each patch by solving:

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

- *Dictionary Update:* Fix all $\hat{\alpha}_{ij}$, and for each atom $l \in 1, 2, \ldots, k$ in $\hat{\mathbf{D}}$,
  - Select the set of patches which 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 row vector whose elements are the $\hat{\alpha}_{ij}(l)$.
  - Update $\hat{\mathbf{d}}_l$ and the $\hat{\alpha}_{ij}(l)$ by minimizing:

$$(\hat{\mathbf{d}}_l, \hat{\alpha}^l) = \arg\min_{\alpha, ||\mathbf{d}||_2=1} ||\mathbf{E}_l - \mathbf{d}\alpha||_F^2. \quad (3)$$

    This one-rank approximation is performed by a truncated SVD of $\mathbf{E}_l$.

Reconstruction: Perform a weighted average:

$$\hat{\mathbf{x}} = \left(\lambda\mathbf{I} + \sum_{ij}\mathbf{R}_{ij}^T\mathbf{R}_{ij}\right)^{-1}\left(\lambda\mathbf{y} + \sum_{ij}\mathbf{R}_{ij}^T\hat{\mathbf{D}}\hat{\alpha}_{ij}\right). \quad (4)$$

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

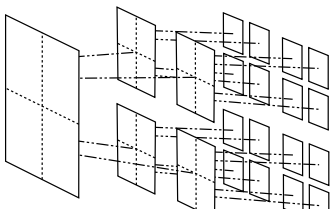


Fig. 2. *Quadtree model chosen for the multiscale.*

3. THE MULTISCALE SPARSE REPRESENTATION

One simple and naive strategy to introduce multiscale analysis consists of using big 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 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 it may be that the K-SVD is trapped in a local minima. By explicitly imposing such multiscale structure, we may help in this regard. This leads us naturally to the proposed framework. We note that learning multiscale dictionaries is important per se, also for applications beyond image denoising.

3.1. The basic model

In this paper we focus on the use of different sizes of atoms simultaneously.[1] Considering the design of a patch-based representation/denoising framework, we put forward a simple Quadtree model on large patches, Figure 2. This is a classical data structure, also used in wedgelets for example [11]. A fixed number of scales, N, is chosen that corresponds to N different sizes of atoms. A big patch of size n pixels is divided along the tree to sub-patches of sizes $n_s = \frac{n}{4^s}$, where s is the depth in the tree. Then, one different dictionary $\mathbf{D}_s$ composed of k_s atoms of size n_s is built at each scale. The original K-SVD exploits the overlapping/shift-invariant sparsity of the patches' representation, which has been found to be prominent for denoising [2, 3, 12]. One asset of our multiscale model is that it does not allow for all possible shifts for the sub-patches inside one large patch, preventing them from constantly adapting their position to the noisy patch. Therefore, this structure permits to force and exploit the overlapping/shift-invariance sparsity at each scale.

The overall idea of the multiscale algorithm we propose stays as close as possible to the original K-SVD algorithm, Figure 1, with an attempt to exploit the several existing scales. The following are the key modifications to the basic algorithm:

Sparse Coding: This remains unchanged if we introduce some new notations. In Equation (3) assume that $\mathbf{R}_{ij}$ remains the matrix that extracts the patch of size $n_0 = n$ with coordinates ij. The dictionary $\hat{\mathbf{D}}$ is a joint one, composed of all the atoms of all the dictionaries $\hat{\mathbf{D}}_s = (\hat{\mathbf{d}}_{sl} \in \mathbb{R}^{n\times 1})_{l\in 1\ldots k_s}$ located at every possible position in the Quadtree. For the scale s, there exists 4^s such positions, we denote their index as p. This makes a total of $\sum_{s=0}^{N-1} 4^s k_s$ atoms in $\hat{\mathbf{D}}$. The OMP is implemented efficiently using a Modified Gram-Schmidt algorithm [13]. For each patch, this step can be achieved in $\mathcal{O}((\sum_{i=0}^{N-1} k_s)n||\hat{\alpha}||_0)$ 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 $s = N-1$ downwards), by:

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

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

  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 $\mathbf{e}_{ijsp}^l$, and $\hat{\alpha}^{sl}$ the row 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:

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

[1] In a separate work we also consider using a multiscale pyramid and learning dictionaries at all the pyramid scales (see also [8]). Results along this direction will be reported elsewhere.

Reconstruction: Remains the same as in Equation (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* 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. Additional Algorithmic Improvements

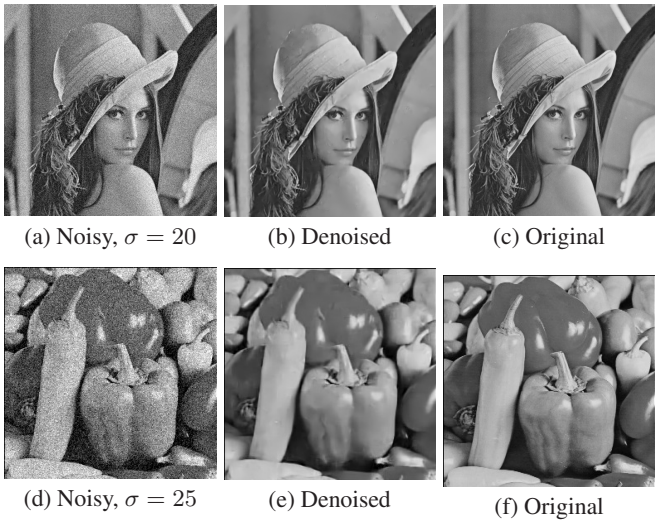
Compared to the original K-SVD algorithm [2], we introduce some additional refinements, which further improve the result without increasing the computational cost.

First, 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 (2.5 in our examples) during the selection procedure of the OMP (refer to [9]). This makes sense since a constant atom does not introduce any noise in a reconstruction.

Secondly, as discussed in [3], the stopping criterion during the OMP is based on the norm of an n-dimensional Gaussian vector which is distributed by the generalized Rayleigh law. This means that one has to stop the approximation when the residual reaches a fuzzy sphere. But 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 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 improvements.

4. EXPERIMENTAL RESULTS

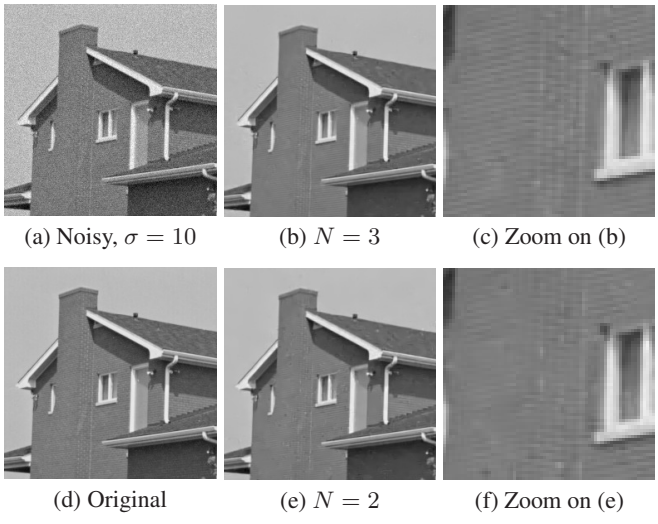
We now present denoising results obtained within the proposed multiscale sparsity framework. On Table 1, our results for $N = 1$ (single-scale) and $N = 2$ scales are compared to those presented in [2, 5, 6, 7]. The best results are shared between our algorithm and [7], where [7] performs better only for very high noise (beyond the normal expected one) and on the images "barbara" and "lena." For $N = 1$, $n = 8 \times 8$. For $N = 2$, n is 10×10 for $\sigma = 5$, 12×12 for $\sigma = 10$, 16×16 for $15 \leq \sigma \leq 25$, and 20×20 for $\sigma \geq 50$. The results from our experiments and [2, 7] reported in Table 1 are averaged over 5 experiments for each image and each level of noise. During our experiments, the number of iterations J was fixed to 20, the number of atoms k_s for each scale was set to 256 and the parameter λ to $0.45n^2/\sigma$. The parameter m, representing the number of patches simultaneously processed, and C, are reported within the table. The initial dictionaries used during these experiments are the results of off-line training on a large generic database of images [2, 3]. The so-called sparsity factor L for these off-line training was set to $L = 6$ for $N = 1$, $L = 20$ for $N = 2$, and $L = 30$ for $N = 3$. Some visual results for $N = 2$ are presented in Figure 3, while further improvements provided by the use of $N = 3$ scales and $n = 20 \times 20$ (PSNR = 36.93) compared to $N = 2$ and $n = 12 \times 12$ (PSNR = 36.57) are shown on Figure 4. One example of a multiscale learned dictionary is presented in Figure 5.



(a) Noisy, $\sigma = 20$ (b) Denoised (c) Original

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

Fig. 3. *Denoising results for* $N = 2$.



(a) Noisy, $\sigma = 10$ (b) $N = 3$ (c) Zoom on (b)

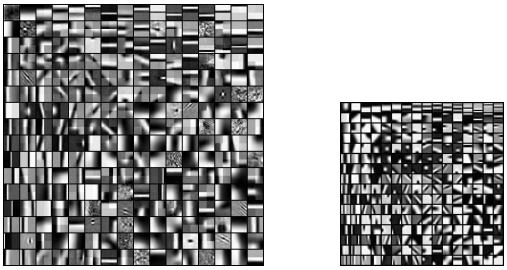
(d) Original (e) $N = 2$ (f) Zoom on (e)

Fig. 4. *Denoising results with* $N = 3$ *and* $N = 2$.

Our implementation was coded in C++ using the Intel Math Kernel Library. For $N = 1$, during one experiment on the 256×256 image "house", for $\sigma = 25$, one *Sparse Coding* step takes approximately 3s on an Opteron 2.4GHz. With the same image and same level of noise, with $N = 2$, this time becomes 60s. In both cases, the *Dictionary Update* takes less than 0.5s. Thus our algorithm is slower than [7], and improving on this is part of ongoing efforts in our group. To achieve this goal one could define a criterion to deactivate some scales during the OMP. Code profiling shows that more

σ	C	m	house			peppers			lena			barbara			boat		
5	1.128	1	38.65	37.62	39.56	37.31	37.34	37.83	38.49	37.91	38.62	37.79	37.12	**38.16**	36.97	36.14	37.19
	1.069	3	39.37	39.62	**39.84**	37.78	37.94	**38.14**	38.60	38.60	**38.70**	38.08	37.59	38.11	37.22	37.13	**37.26**
10	1.128	1	35.35	35.26	36.37	33.77	34.07	34.38	35.61	35.18	**35.81**	34.03	33.79	**34.86**	33.58	33.09	33.76
	1.042	3	35.98	36.24	**36.54**	34.28	34.49	**34.60**	35.47	35.63	35.75	34.42	34.35	34.57	33.64	33.81	**33.87**
15	1.041	4	33.64	34.08	34.75	31.74	32.13	32.35	33.90	33.70	**34.20**	31.86	31.80	**33.05**	31.70	31.44	31.92
	1.026	4	34.32	34.59	**34.87**	32.22	**32.41**	**32.41**	33.70	33.90	34.08	32.37	32.47	32.58	31.73	31.99	**32.02**
20	1.023	4	32.39	32.90	33.54	30.31	30.59	30.84	32.66	32.64	**33.02**	30.32	30.37	**31.71**	30.38	30.12	30.61
	1.026	4	33.20	33.45	**33.67**	30.82	31.10	**31.11**	32.38	32.69	32.86	30.83	31.11	31.24	30.36	30.69	**30.77**
25	1.023	4	31.40	32.44	32.66	29.21	29.95	29.82	31.69	31.66	**32.06**	29.13	29.96	**30.68**	29.37	29.66	29.64
	1.020	4	32.15	32.44	**32.75**	29.73	29.95	**30.05**	31.32	31.66	31.89	29.60	29.95	30.17	29.28	29.66	**29.79**
50	1.018	4	28.26	28.67	**29.68**	25.90	25.29	26.45	28.61	28.38	**29.10**	25.48	24.09	**27.50**	26.38	25.93	26.63
	1.010	5	27.95	28.25	29.43	26.13	26.40	**26.62**	27.79	28.11	28.75	25.47	26.04	26.80	25.95	26.34	**26.74**
100	1.018	4	25.11	23.08	**25.96**	22.66	20.51	**23.06**	25.64	23.32	**25.91**	22.61	20.64	**24.11**	23.75	21.78	**23.88**
	1.008	5	23.71	23.69	24.73	21.75	22.05	22.57	24.46	24.48	25.13	21.89	22.04	22.88	22.81	22.95	23.65

Table 1. *PSNR results of our denoising algorithm. Each case (image and noise level) is divided into six parts: The top row for each part presents the results from, respectively, [5, 6, 7] (from left to right). The bottom row presents successively the original K-SVD [2], our results for $N = 1$ (single-scale), and then $N = 2$ scales. Each time the best results is in bold. The values of the parameters C and m are reported in the second and third columns: Inside these ones, the top part of each cell is devoted to $N = 1$ and the low part to $N = 2$.*



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

Fig. 5. *One learned multiscale dictionary.*

than 85% of the computational time is usually devoted to matrix-vector multiplication due to the computation of scalar products in the OMP. This can be significantly improved using standard nearest-neighborhood approximation algorithms, which often provide two or more orders of magnitude improvement. In addition, NVIDIA is at the moment developing a parallel linear algebra library which takes advantage of graphic cards and could potentially provide a speedup magnitude of more than 20 for these multiplications. We plan to provide a parallel version of the algorithm which will be able to take advantage of the new multi-core processors. To conclude, we do not anticipate the computational cost of the algorithm to be a bottleneck in the near future.

5. 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 denoising results. Our current efforts are devoted in part to the speed-up of the algorithm following the approaches mentioned above, and to the extension to multiscale sparse representation of color images, see [3] for the single-scale case. Another direction we are pursuing is to combine the K-SVD with image pyramids. Results in these directions will be reported soon.

6. 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. on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, November 2006.

[2] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. on Image Processing*, vol. 54, no. 12, pp. 3736–3745, December 2006.

[3] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," 2006, submitted, `http://www.ima.umn.edu/preprints/oct2006/2139.pdf`.

[4] S. Mallat, *A Wavelet Tour of Signal Processing, Second Edition*, Academic Press, September 1999.

[5] J. Portilla, V. Strela, M. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Trans. on Image Processing*, vol. 13, no. 4, pp. 496–508, 2004.

[6] C. Kervrann and J. Boulanger, "Optimal spatial adaptation for patch-based image denoising," *IEEE Trans. on Image Processing*, vol. 15, no. 10, pp. 2866–2878, 2006.

[7] 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*, San Jose, CA, USA, January 2006, vol. 6064.

[8] B.A. Olshausen, P. Sallee, and M.S. Lewicki, "Learning sparse multiscale image representations," *Advances in Neural Information Processing Systems*, vol. 15, pp. 1327–1334, 2003.

[9] G. M. Davis, S. Mallat, and Z. Zhang, "Adaptive time-frequency decompositions," *SPIE J. of Opt. Engin.*, vol. 33, no. 7, pp. 2183–2191, July 1994.

[10] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. on Information Theory*, vol. 50, no. 10, October 2004.

[11] D. Donoho, "Wedgelets: Nearly minimax estimation of edges," *Annals of statistics*, vol. 27, no. 3, pp. 859–897, June 1998.

[12] 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, USA, June 2005.

[13] A. Björck, *Numerical Methods for Least-Squares Problems*, 1996.