K-SVD meets Transform Learning: Transform K-SVD

Ender M. Eksioglu*, *Member, IEEE* and Ozden Bayir, *Student Member, IEEE*

***Abstract*—Recently there has been increasing attention directed towards the analysis sparsity models. Consequently, there is a quest for learning the operators which would enable analysis sparse representations for signals in hand. Analysis operator learning algorithms such as the Analysis K-SVD have been proposed. Sparsifying transform learning is a paradigm which is similar to the analysis operator learning, but they differ in some subtle points. In this paper, we propose a novel transform operator learning algorithm called as the Transform K-SVD, which brings the transform learning and the K-SVD based analysis dictionary learning approaches together. The proposed Transform K-SVD has the important advantage that the sparse coding step of the Analysis K-SVD gets replaced with the simple thresholding step of the transform learning framework. We show that the Transform K-SVD learns operators which are similar both in appearance and performance to the operators learned from the Analysis K-SVD, while its computational complexity stays much reduced compared to the Analysis K-SVD.**

***Index Terms*—Sparsifying transform learning; analysis operator learning; dictionary learning; sparse representation**

EDICS: MLSAS-SPARSE, DSP-SPARSE, IMD-SPAR

I. Introduction

Using the sparsity prior for signal representation tasks has been a popular approach which brings performance gain in various applications. The synthesis sparsity model has been the more common approach when compared to the analysis sparsity model. Various dictionary learning methods for the synthesis model have been proposed in the literature. The original (synthesis) K-SVD is one such method which allows the construction of an overcomplete dictionary suitable for sparse synthesis, by learning the dictionary from the data itself [1]. Recently, the analysis sparsity model has been receiving increasing attention as a lesser-known counterpart to the synthesis sparsity model [2], [3]. The analysis sparsity prior provides a signal model different from but related to the synthesis sparsity. Similar to the synthesis dictionary learning, methods for analysis operator or analysis dictionary learning have also been proposed. Analysis K-SVD [4] is one such method developed in parallel with its better known original synthesis counterpart [1]. There are other recent methods for analysis operator learning such as the ones given in [5], [6]. These methods try to solve the minimization of cost functions similar to the one proposed in [4]. Recently, transform sparsity model which is more general than the analysis sparsity model has been introduced in [7]. Transform learning deals with a cost function in which the modeling error called as the sparsification error is defined in the transform or analysis domain rather than the original signal domain. This slight variation in the problem formulation of transform learning results in some important advantages when compared to the analysis operator learning framework. In the transform sparsity model, the costly co-sparse representation step gets replaced with a simpler thresholding procedure, leading to substantial computational savings. In this paper, we develop a new transform learning method which assumes a structure similar to the Analysis K-SVD algorithm of [4]. The resulting algorithm which we call as the Transform K-SVD is shown to learn operators or transforms with performance and structure similar to the operators learned using Analysis K-SVD. However, Transform K-SVD has much reduced computational complexity as promised by the transform learning paradigm.

The authors are with the Electronics and Communications Engineering Department, Istanbul Technical University, Istanbul, Turkey (Phone/fax: +90-212-285 3623, e-mail: {eksioglue, bayiroz}@itu.edu.tr).

II. Transform Learning Formulation

We consider the transform learning framework which was introduced in [7] and further developed in [8]. The formulation for the overcomplete transform learning problem is given in [8] using the following minimization.

$$\min_{\mathbf{W},\mathbf{X}} \|\mathbf{W}\mathbf{Y}-\mathbf{X}\|_F^2 - \gamma \log\det\left(\mathbf{W}^T\mathbf{W}\right) + \eta \sum_{k\neq j} |\langle \boldsymbol{w}^k, \boldsymbol{w}^j\rangle|^p,$$
$$\text{s.t.} \quad \|\boldsymbol{x}_n\|_0 \leq s \ \forall n, \|\boldsymbol{w}^k\|_2 = 1 \ \forall k \quad (1)$$

Here, $\boldsymbol{y}_n \in \mathbb{R}^M$, $n = 1, \ldots, N$ are the signal vectors, and they form the columns of the overall data matrix $\mathbf{Y} \in \mathbb{R}^{M\times N}$. $\mathbf{W} \in \mathbb{R}^{K\times M}$ is the trained overcomplete sparsifying transform, with $\boldsymbol{w}^k \in \mathbb{R}^M$ for $k = 1, \ldots, K$ being the row vectors of $\mathbf{W}$. In [8] the overcomplete transform case $K > M$ is studied, whereas [7] considers the simpler case of square transform learning with $K = M$. In (1), $\boldsymbol{x}_n \in \mathbb{R}^K$ for $n = 1, \ldots, N$ are the sparse signal representations in the transform domain, and they constitute the columns of the matrix $\mathbf{X} \in \mathbb{R}^{K\times N}$. The constant $s \ll K$ determines the signal sparsity in the transform domain. In the cost function of (1), there are three terms. Of these, $\|\mathbf{W}\mathbf{Y}-\mathbf{X}\|_F^2$ is the main sparsification error. The $\log\det\left(\mathbf{W}^T\mathbf{W}\right)$ and $\sum_{k\neq j}|\langle \boldsymbol{w}^k, \boldsymbol{w}^j\rangle|^p$ are secondary terms, which are used to avoid trivial and degenerate solutions. $\log\det\left(\mathbf{W}^T\mathbf{W}\right)$ enforces full rank constraint for the tall $\mathbf{W}$ matrix, whereas $\sum_{k\neq j}|\langle \boldsymbol{w}^k, \boldsymbol{w}^j\rangle|^p$ is included to avoid repeated rows in $\mathbf{W}$. Here, we use an alternate approach by simplifying the cost function when compared to (1). The core minimization problem we deal with is given below.

$$\min_{\mathbf{W},\mathbf{X}} \|\mathbf{W}\mathbf{Y}-\mathbf{X}\|_F^2, \text{ s.t. } \|\boldsymbol{x}_n\|_0 \le s\ \forall n, \|\boldsymbol{w}^k\|_2 = 1\ \forall k,$$
$$\mu\{\mathbf{W}^T\} = \max_{k\neq j} |\boldsymbol{w}^k \boldsymbol{w}^{j^T}| \le 1-\delta \quad (2)$$

Here, the unit norm constraint on the operator rows eliminates the trivial all zero solution. $\mu\{\mathbf{W}^T\}$ is the row-wise mutual coherence for $\mathbf{W}$. The mutual coherence constraint between the rows is for excluding too close or repeating rows. We will incorporate a row replacement step into the transform update procedure of our transform learning algorithm to enforce the mutual coherence constraint and hence to eliminate degenerate solutions with repeated rows. This approach is similar to the one held in the Analysis-KSVD algorithm of [4].

III. The Transform K-SVD Algorithm

To solve the minimization problem (2), we shall utilize the two-stage iterative approach, which has been extensively used in dictionary and analysis operator learning algorithms. Hence, we shall minimize on $\mathbf{W}$ and $\mathbf{X}$ separately. The first stage at the i^{th} iteration should solve the following minimization problem on $\mathbf{X}$.

$$\mathbf{X}_{(i)} = \operatorname*{argmin}_{\mathbf{X}} \|\mathbf{W}_{(i-1)}\mathbf{Y}-\mathbf{X}\|_F^2, \text{ s.t. } \|\boldsymbol{x}_n\|_0 \le s\ \forall n \quad (3)$$

This is equivalent to the sparse coding stage of the sparsifying transform learning algorithm as introduced in [7]. As detailed in [7], this problem is computationally much simpler to solve than the sparse representation steps of regular dictionary learning algorithms and also the co-sparse representation steps of recent analysis operator learning algorithms. The exact solution for this problem is simply given by column-wise thresholding of $\mathbf{WY}$ as to retain only s terms of highest magnitude in each column [7]. For example, Analysis K-SVD requires a backward greedy algorithm for the solution of its similar analysis pursuit step, which has much higher complexity compared to simple thresholding. As described in [7], if the ℓ_0 sparsity constraint in (3) is convexly relaxed with an ℓ_1 penalty, the exact solution in that case will be computed by a computationally even cheaper soft thresholding procedure. In our realizations in this paper we have stuck with s-term column-wise hard thresholding.

The second step of the iterative learning process will start with a minimization over $\mathbf{W}$ as described below.

$$\mathbf{W}_{(i)} = \operatorname*{argmin}_{\mathbf{W}} \|\mathbf{W}\mathbf{Y}-\mathbf{X}_{(i)}\|_F^2, \text{ s.t. } \|\boldsymbol{w}^k\|_2 = 1\ \forall k \quad (4)$$

We omit the $\mu\{\mathbf{W}^T\}$ constraint of (2) here. We will enforce this mutual coherence constraint via row replacement operations while solving (4). One possible solution of (4) would be calculating the least squares solution followed by row-wise normalization. However, here we adopt an approach similar to operator update step of the Analysis K-SVD. We solve the minimization problem sequentially for each row of $\mathbf{W}$. Following the argument put forward in both Synthesis and Analysis K-SVD algorithms, we maintain that the update of the k^{th} row $\boldsymbol{w}^k$ should only be determined by the columns of the data matrix $\mathbf{Y}$ which have already been found to be approximately orthogonal to $\boldsymbol{w}^k$. Hence, the transform operator update step should only include the data signals which have the particular row vector which is to be updated in their co-support. The data signals, which have the row vector $\boldsymbol{w}^k$ in their co-support, are determined by the zeros of the row vector $\boldsymbol{x}^k$ as found in the sparse coding step. Taking these arguments into consideration, the minimization problem (4) transforms into the sequential minimization problem as given below for $k = 1,\dots,K$.

$$\boldsymbol{w}^k_{(i)} = \operatorname*{argmin}_{\boldsymbol{w}} \|\boldsymbol{w}\mathbf{Y}_k^{(i)}\|_2^2, \text{ s.t. } \|\boldsymbol{w}\|_2 = 1 \quad (5)$$

Here, $\mathbf{Y}_k^{(i)}$ is a submatrix of $\mathbf{Y}$ containing only a selection of columns. The columns selected correspond to the positions of zeros in the k^{th} row of $\mathbf{X}_{(i)}$, $\boldsymbol{x}^k_{(i)}$. By using only the learned co-support information rather than the overall $\mathbf{X}_{(i)}$ matrix, we preserve the co-support structure as learned in the i^{th} sparse coding step.

The resulting minimization problem in (5) is exactly the same as the one derived in the analysis operator update step of Analysis K-SVD. This is interesting, because our transform learning approach (4) and the Analysis K-SVD operator learning approach start from two very different cost functions, but end up with the same minimization problem for the update of the operator. We should note that in [4] during the development of the Analysis K-SVD, the analysis comes up with exactly the same minimization problem as given in (5) only through a rigorous procedure involving various approximations. In [4], the original operator update minimization problem evolves into the form in (5), because it provides a simpler to solve, but still well-performing approximation. However, here (5) is the exact and natural form of the minimization problem when we require row-wise sequential updates for the transform in (4). Therefore, the formulation here provides further support for the use of (5) as an efficient approximation in [4].

Since (5) is identical to the approximate minimization problem in [4], we can adopt the solution proposed in Analysis K-SVD for the operator row update. The exact solution for (5) is the left-singular vector corresponding to the smallest singular value of $\mathbf{Y}_k^{(i)}$. Given that we have to calculate the SVD (or at least one particular singular vector) for each submatrix $\mathbf{Y}_k^{(i)}$ for $k = 1,\dots,K$, it is appropriate to name the resulting overall algorithm as Transform K-SVD. This algorithm possesses the advantages of both the transform learning approach and the Analysis K-SVD. First of all, the sparse coding step has been simplified compared to the Analysis K-SVD. On the other hand, using the row-wise updates as in Analysis K-SVD, the rows of $\mathbf{W}$ are modified separately and in parallel. After updating a row, we need to include an additional step to enforce the mutual coherence constraint of (2) and to avoid repeating rows. Similar to the procedure in [4], we replace a learned row $\boldsymbol{w}^k$ with a randomly generated one if $|\boldsymbol{w}^k \boldsymbol{w}^{j^T}| > 1-\delta$, for any $j < k$. The novel Transform K-SVD algorithm for learning a sparsifying transform is outlined in a structured form in Alg.1.

Another point recommended in [4] is the explicit modification of the row update rule for special applications, such as natural images. In [4], it is recommended to modify the

Algorithm 1 Transform K-SVD

```
Input: Data record of length $N$, $\mathbf{Y} = \{\boldsymbol{y}_n\}_{n=1}^N$; the desired sparsity in the transform domain given by $s$; $\delta$.
Goal: $\{\hat{\mathbf{W}}, \hat{\mathbf{X}}\} = \arg\min_{\mathbf{W},\mathbf{X}} \|\mathbf{W}\mathbf{Y} - \mathbf{X}\|_F^2$,
    s.t. $\|\boldsymbol{x}_n\|_0 \le s \;\forall n$; $\|\boldsymbol{w}^k\|_2 = 1 \;\forall k$; $\mu\{\mathbf{W}^T\} \le 1 - \delta$.

1: Initialize the transform operator, $\mathbf{W}_{(0)} = \mathbf{W}_0$.
2: for $i := 1, 2, \ldots$ do                                  ▷ main iteration
3:     $\mathbf{X}_{(i)} = \arg\min_{\mathbf{X}} \|\mathbf{W}_{(i-1)}\mathbf{Y} - \mathbf{X}\|_F^2$, s.t. $\|\boldsymbol{x}_n\|_0 \le s \;\forall n$
       ▷ transform sparse coding step, solved by column-wise thresholding of $\mathbf{W}_{(i-1)}\mathbf{Y}$.
4:     for $k := 1, 2, \ldots, K$ do        ▷ sequential update of the rows of the transform operator
5:         Form the matrix $\mathbf{Y}_k^{(i)}$, by selecting the columns of $\mathbf{Y}$ which correspond to the zero positions of $\boldsymbol{x}_{(i)}^k$.
6:         $\boldsymbol{w}_{(i)}^k = \arg\min_{\boldsymbol{w}} \|\boldsymbol{w}\mathbf{Y}_k^{(i)}\|_2^2$, s.t. $\|\boldsymbol{w}\|_2 = 1$
           ▷ calculate for $\mathbf{Y}_k^{(i)}$ the singular vector with the minimum singular value.
7:         Replace $\boldsymbol{w}_{(i)}^k$ if $|\boldsymbol{w}_{(i)}^k {\boldsymbol{w}^j}_{(i)}^T| > 1-\delta$, for any $j < k$.
8:     end for                                  ▷ end of row update iteration
9: end for                                      ▷ end of main iteration
```

row update in the following fashion when working on natural image patches.

$$\boldsymbol{w}^k = \arg\min_{\boldsymbol{w}} \|\boldsymbol{w}\mathbf{Y}_k\|_2^2 + \gamma \sum_{k \neq j} |I_k^C \cap I_j| (\boldsymbol{w}^j \boldsymbol{w}^T)^2, \quad \text{s.t.} \quad \|\boldsymbol{w}\|_2 = 1 \tag{6}$$

Here, I_k and I_j denote the sets of the indices of data vectors which have the kth and jth rows in their co-supports, respectively. I^C is a complementary set [4]. It is discussed in [4] that (6) forces pairs of atoms which do not jointly appear in the co-supports of signals to become orthogonal to each other. This should result in a more stable recovery in the applications of the learned operator/transform. The closed-form solution for this modified problem is given in [4] as the least eigenvector of the following matrix.

$$\mathbf{Y}_k {\mathbf{Y}_k}^T + \gamma \sum_{k \neq j} |I_k^C \cap I_j| {\boldsymbol{w}^j}^T \boldsymbol{w}^k \tag{7}$$

Hence, when we deal with natural image patches as the data vectors, we employ transform update step as given by (6).

The computational complexity of the Transform K-SVD per iteration can be analyzed as follows. The transform sparse coding step is dominated by the $\mathbf{WY}$ multiplication rather than the thresholding procedure, and its complexity is given as $\mathcal{O}(NMK)$. This is much reduced compared to to the sparse coding step of the Analysis K-SVD. The backward greedy (BG) and the optimized backward greedy (OBG) analysis pursuit algorithms used in Analysis K-SVD have complexities $\mathcal{O}(NM^2K)$ and $\mathcal{O}(NM^3K)$, respectively. In the transform update step, the matrix $\mathbf{Y}_k$ is of size $\left(M \times N \frac{K-s}{K}\right) \approx (M \times N)$ on average. We can use Lanczos bidiagonalization based methods for the calculation of the minimal left-singular vector of each such matrix [9], requiring $\mathcal{O}(NM)$ operations. Hence, the total computational complexity for K matrices will scale as $\mathcal{O}(NMK)$. Therefore, the overall computational complexity of Transform K-SVD is also given as $\mathcal{O}(NMK)$. This result shows a reduction on the order of M or M^2 when compared to the Analysis K-SVD depending on whether the BG or the OBG algorithm is used. The $\mathcal{O}(NMK)$ computational complexity of Transform K-SVD is the same as the complexity of the overcomplete transform learning algorithm in [8]. We should note that in our simulations we realized the transform update using (7), where the eigenvector calculation is for a matrix of size $M \times M$. Since the eigenvector calculation dominates the computation time for Transform K-SVD, its required computation times come out to be sublinear in N.

IV. Simulation Results

In this section we study the natural image denoising performance of the learned operators. We compare operators learned via the Analysis K-SVD algorithm, our novel Transform K-SVD and also the constant, analytic finite difference (FD) operator [6], [7]. The simulations are geared towards facilitating a comparison between the Analysis K-SVD and Transform K-SVD operators, rather than giving best or state-of-the-art results. The denoising procedure is adapted from [4] using the Analysis K-SVD realization available from the authors' webpages. The implementations were performed in Matlab on a system with an Intel Core i7 CPU at 2.4GHz, 12GB memory and 64-bit Windows 8 operating system.

Each noisy image is used to generate $N = 20,000$ possibly overlapping patches of size 8×8, and these patches form the training data $\mathbf{Y}$ used by the two algorithms. The operators are of size 128×64 with $K = 128$, and the number of iterations is 20. The operators are employed to denoise all the overlapping patches obtained from the noisy image using analysis sparsity regularization with error-based OBG for sparse coding. The denoised patches are merged to form the denoised image. Both algorithms use (6) and the corresponding update rule with $\gamma = 1000$ and $\delta = 0.05$. Both algorithms utilize the exact same operator initialization and row replacement schemes as detailed in [4]. In operator learning, analysis K-SVD employs OBG with target subspace dimension 7, whereas Transform K-SVD uses hard thresholding with $s = 30$. The standard deviation of the AWG noise changes as $\sigma = 5$, $\sigma = 10$ and $\sigma = 20$, which result in a PSNR of about 34.1, 28.1 and 22.1 dB in the noisy images, respectively.

Table 1 details the denoising results of the various operators for five commonly used images. The operators learned via Transform K-SVD perform in general a little better than the operators learned from Analysis K-SVD and the analytic FD operator. It is also observed that with increasing noise the performance gap between Transform K-SVD and Analysis K-SVD increases. Fig.1, on the other hand, visualizes the operators learned by the Transform and Analysis K-SVD algorithms from the noisy 'Barbara' and 'Lena' images. We can see similar structures appearing in the Transform and Analysis K-SVD operators learned from the same images, and we can deduce that although they start from different formulations of

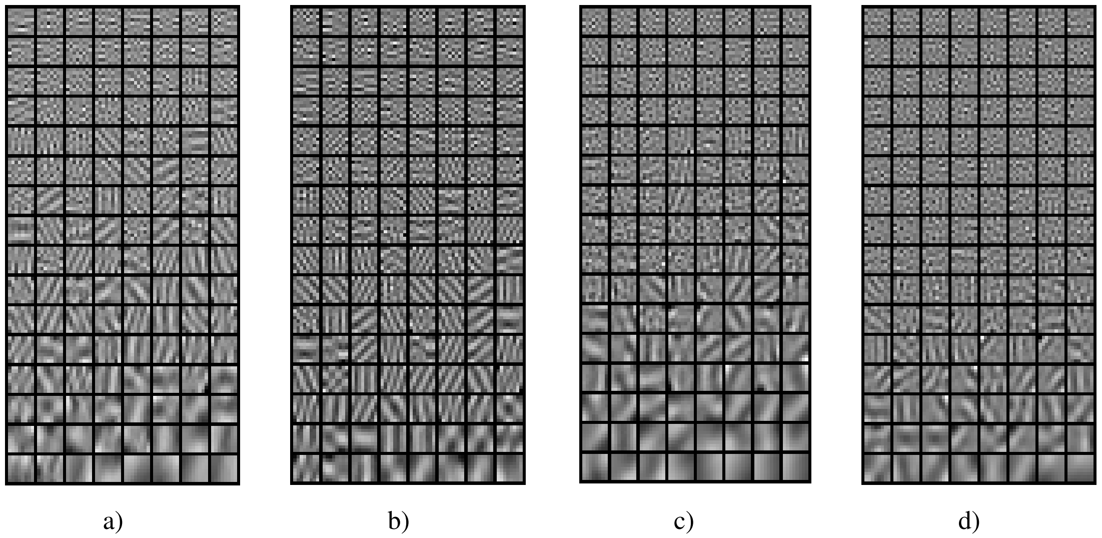

Fig. 1: Examples of the operators learned for $\sigma = 10$: a) Analysis K-SVD for 'Barbara', $\mu = 0.940$, b) Transform K-SVD for 'Barbara', $\mu = 0.946$, c) Analysis K-SVD for 'Lena', $\mu = 0.918$, d) Transform K-SVD for 'Lena', $\mu = 0.942$.

TABLE I: Denoised image PSNRs in dB for five natural images, noise standard deviation $\sigma = 5, 10$ and 20.

σ	operator	Lena	Barbara	Peppers	Boats	House
5	T. K-SVD	37.63	37.04	36.90	36.62	38.27
	A. K-SVD	37.33	37.02	36.94	36.70	38.18
	FD	36.38	34.69	35.93	35.13	36.96
10	T. K-SVD	33.61	32.28	32.46	32.30	34.16
	A. K-SVD	33.34	32.09	32.27	32.06	33.91
	FD	33.01	30.47	32.17	31.65	33.55
20	T. K-SVD	31.27	28.53	29.42	29.32	30.83
	A. K-SVD	29.48	27.28	27.87	27.95	29.66
	FD	30.07	26.81	28.86	28.42	30.28

TABLE II: Computation times (in seconds) of the two algorithms for transform/operator learning with a single iteration.

# of patches, N	1,000	2,000	4,000	8,000	16,000	20,000
T. K-SVD	2.4	2.8	3.2	4.0	5.2	5.8
A. K-SVD	51.4	100.0	202.1	396.6	793.0	963.6

the problem, the Transform and Analysis K-SVD algorithms provide related solutions to the sparsifying operator learning problem. The corresponding mutual coherence values are also listed in the figure caption. Fig.2 shows the noisy and denoised images for the image 'Lena' with $\sigma = 10$.

The computation times of the two algorithms for different training data size N are listed in Table 2. These are the required times only for the transform/operator learning section for a single iteration with multicore pooling in MATLAB off. The results are averaged over 20 realizations. Transform K-SVD decreases the required time substantially by realizing the sparse coding step via simple thresholding.

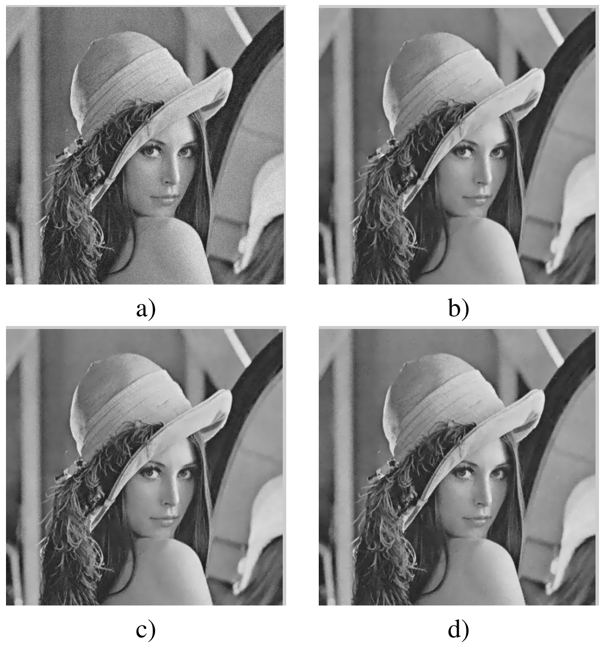

Fig. 2: Denoised images for $\sigma = 10$: a) Noisy image (PSNR = 28.12dB), b) Transform K-SVD (PSNR = 33.61dB), c) Analysis K-SVD (PSNR = 33.34dB), d) FD (PSNR = 33.01dB).

V. Conclusions

We have presented Transform K-SVD, a novel algorithm for solving the transform learning problem. Transform K-SVD extends the sequential update paradigm of the K-SVD synthesis dictionary and analysis operator learning algorithms into the newly introduced sparsifying transform learning realm. Transform K-SVD successfully merges the computationally efficient sparse coding step of the transform learning with the row-wise operator update of the K-SVD, which allows parallel implementation. Simulations with natural images show that Transform K-SVD can learn transforms which are structurally and performance-wise similar to and even better than the operators learned from Analysis K-SVD, while retaining a substantial computational saving.

Acknowledgments

We thank the anonymous reviewers for their insightful comments.

References

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

[2] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," *Inverse Problems*, vol. 23, no. 3, pp. 947, 2007.

[3] S. Vaiter, G. Peyré, C. Dossal, and J. Fadili, "Robust sparse analysis regularization," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2001–2016, 2013.

[4] R. Rubinstein, T. Peleg, and M. Elad, "Analysis K-SVD: a dictionary-learning algorithm for the analysis sparse model," *IEEE Trans. Signal Process.*, vol. 61, no. 3, pp. 661–677, 2013.

[5] S. Hawe, M. Kleinsteuber, and K. Diepold, "Analysis operator learning and its application to image reconstruction," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2138–2150, 2013.

[6] M. Yaghoobi, Sangnam Nam, R. Gribonval, and M.E. Davies, "Constrained overcomplete analysis operator learning for cosparse signal modelling," *IEEE Trans. Signal Process.*, vol. 61, no. 9, pp. 2341–2355, 2013.

[7] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Trans. Signal Process.*, vol. 61, no. 5, pp. 1072–1086, 2013.

[8] S. Ravishankar and Y. Bresler, "Learning overcomplete sparsifying transforms for signal processing," in *IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, 2013.

[9] A. Korobeynikov, "Computation- and space-efficient implementation of SSA," *Statistics and its Interface*, vol. 3, no. 3, pp. 357–368, 2010.