LEARNING DOUBLY SPARSE TRANSFORMS FOR IMAGE REPRESENTATION

Saiprasad Ravishankar and Yoram Bresler

Department of Electrical and Computer Engineering and the Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign, IL, USA

ABSTRACT

The sparsity of images in a fixed analytic transform domain or dictionary such as DCT or Wavelets has been exploited in many applications in image processing including image compression. Recently, synthesis sparsifying dictionaries that are directly adapted to the data have become popular in image processing. However, the idea of learning sparsifying transforms has received only little attention. We propose a novel problem formulation for learning doubly sparse transforms for signals or image patches. These transforms are a product of a fixed, fast analytic transform such as the DCT, and an adaptive matrix constrained to be sparse. Such transforms can be learnt, stored, and implemented efficiently. We show the superior promise of our approach as compared to analytical sparsifying transforms such as DCT for image representation.

Index Terms— Analysis transforms, Structured transforms, Image representation, Dictionary learning, Sparse representation.

1. INTRODUCTION

Sparse representation of signals in transform domains or dictionaries has become widely popular in recent years. It is well known that images (or image patches) admit a sparse representation (few significant non-zero coefficients) in transform domains such as Wavelets and discrete cosine transform (DCT). This property has been used in designing various compressions standards such as JPEG2000.

Alternatively, a signal $y \in \mathbb{R}^n$ may be represented as a linear combination $y \approx Dx$ of a small number of atoms/columns from a dictionary $D \in \mathbb{R}^{n \times K}$, where $x \in \mathbb{R}^K$ is sparse, i.e., $\|x\|_0 \ll K$ (the l_0 quasi norm counts the number of non-zeros). This is known as the synthesis model [1], which has received considerable recent attention. The idea of learning a synthesis dictionary from training signals has also been exploited in recent papers (cf. [2] for a discussion of the popular K-SVD algorithm). Such adaptive synthesis dictionaries have been shown to be useful in a variety of applications such as image denoising [3] and medical image reconstruction [4].

The well-known analysis model [1] model suggests that a signal y is sparsifiable using a transform $W \in \mathbb{R}^{m \times n}$, i.e., $Wy \approx x$, where $x \in \mathbb{R}^m$ is sparse with $\|x\|_0 \ll m$ (x is the sparse code for y). Well-known examples of such transforms for natural signals include Wavelets and DCT. These transforms are square, i.e., $m = n$, and orthonormal. We will focus on square transforms in this work. However, while analytical sparsifying transforms have been extensively used in many applications, the idea of learning the transform from data has received only little attention [5], [6].

Now, while one could learn an 'unstructured' transform, imposing additional structure on the learnt transform W could lead to faster learning and implementation. Several structures have been recently proposed for learning synthesis dictionaries [7]. In particular, Rubinstein et al. [8] proposed learning a doubly sparse synthesis dictionary. However, the idea of structured analysis learning has not been explored.

In this work, we propose learning 'doubly sparse' analysis transforms. Specifically, we learn a sparsifying transform $W = B\Phi$ that is a product of two different transforms: $\Phi \in \mathbb{R}^{n \times n}$ is an analytical transform with an efficient implementation; and $B \in \mathbb{R}^{n \times n}$ is sparse. This structure combines the advantages of trained and analytic transforms: adapting to the data, it performs better than analytical transforms, yet owing to the sparsity of B, it can be stored and applied efficiently. As we show, it can also be learnt more efficiently than unstructured transforms.

This work was supported in part by the National Science Foundation (NSF) under grant CCF 10-18660.

2. PROBLEM FORMULATION

Given a matrix $Y \in \mathbb{R}^{n \times N}$ whose columns represent training signals, a formulation for learning an 'unstructured' transform $W \in \mathbb{R}^{n \times n}$ has been proposed by us in another paper [9] as follows.

$$(\text{P1}) \quad \min_{W,X} \|WY - X\|_F^2 - \lambda \log\det W + \mu \|W\|_F^2 \quad s.t. \ \|X_i\|_0 \leq T_0 \ \forall\, i \tag{1}$$

Here, X is an unknown matrix containing the sparse codes of the training signals (as columns). Subscript i denotes the i^{th} column of X, which is allowed to have T_0 non-zeros. The first term in the cost denotes the 'sparsification error' in the transform domain, namely the difference between the approximately sparse WY and the sparse code X. The $-\log\det W$ penalty helps enforce full rank on the transform W and eliminates degenerate solutions such as those with zero rows, or repeated rows. The $\|W\|_F^2$ penalty helps remove a 'scale ambiguity' [9] in the solution, which occurs when the data admits an exactly sparse representation. The $-\log\det W$ and $\|W\|_F^2$ penalties additionally help control the condition number of the learnt transform [9]. Poorly conditioned transforms typically convey little information and may degrade performance in applications.

We formulate the doubly sparse transform learning problem by replacing W with $B\Phi$ in Problem (P1) and adding an extra sparsity constraint on B. Since the determinant is multiplicative, we have $\log\det(B\Phi) = \log\det(B) + C$ where $C = \log\det(\Phi)$, and we have assumed that Φ has positive determinant (else, it can be multiplied on the left with a diagonal ± 1 sign matrix to enforce positive determinant). Moreover, if matrix Φ is orthonormal (e.g. Wavelets, DCT), we get that $\|B\Phi\|_F^2 = \|B\|_F^2$. With these simplifications, the doubly sparse transform learning problem formulation becomes

$$(\text{P2}) \quad \min_{B,X} \left\|B\hat{Y} - X\right\|_F^2 - \lambda \log\det(B) + \mu \|B\|_F^2 \quad s.t. \ \|B\|_0 \leq T_1, \ \|X_i\|_0 \leq T_0 \ \forall\, i \tag{2}$$

where $\hat{Y} = \Phi Y$ and the matrix B is restricted to have T_1 non-zeros. Problem (P2) is, however, non-convex.

3. ALGORITHM

Our algorithm for solving Problem (P2) alternates between updating X and B. The initialization for B needs to be appropriately chosen, i.e., with positive determinant. In one step (Sparse Update step) of our algorithm, we solve Problem (P2) with fixed B.

$$\min_X \|B\hat{Y} - X\|_F^2 \ \ s.t. \ \ \|X_i\|_0 \leq T_0 \ \ \forall \ i \tag{3}$$

The solution X can be computed exactly and cheaply by thresholding $B\hat{Y}$, and retaining only the T_0 largest coefficients in each column. In the second step (Transform Update step) of our algorithm, we solve Problem (P2) with fixed X as follows.

$$\begin{aligned} &\min_B \|B\hat{Y} - X\|_F^2 - \lambda \, log \, det \, B + \mu \, \|B\|_F^2 \\ &s.t. \ \ \|B\|_0 \leq T_1. \end{aligned} \tag{4}$$

This problem does not have a closed-form solution and is non-convex. We could solve for B using iterative procedures such as the projected conjugate gradient algorithm. However, we found that an alternative strategy of employing the conjugate gradient (CG) algorithm followed by post-thresholding of B led to better empirical performance. Hence, we choose this alternative strategy. The iterative updates in the CG algorithm (initialized with the most recent B) are as follows.

$$B^{k+1} = B^k - \eta_k \, g^k \tag{5}$$

Here k denotes the iteration number; g^k is the CG; and $\eta_k > 0$ is the step size. We could use the Armijo step size rule which guarantees decreasing cost. However, a fixed step size rule (i.e., constant but small η_k) was also observed to work well and faster in practice.

The result from the CG algorithm is thresholded retaining only the T_1 largest elements. Note that for improperly chosen initializations, this hard thresholding step may result in a transform of zero or negative determinant. If the determinant becomes zero, in order to prevent the algorithm from getting stuck, we scale a non-zero entry B_{ij} of B such that the modified B has non-zero/reasonably positive determinant (det B is affine in B_{ij} when all other entries are fixed). Alternatively, one could differentially scale the entries of an entire column of B. These procedures maintain the support of B. On the other hand, if the hard thresholding results in det $(B) < 0$, then we can simply swap two rows of B along with the corresponding rows of X and the determinant of the permuted B is positive. Such a permutation results in a trivially equivalent transform or iterate.

In our experiments, when we initialized the alternating algorithm with an identity matrix B, the aforementioned extra steps were never invoked as the algorithm did not reach degenerate states. The computational cost per iteration (of sparse update and transform update) of the proposed algorithm scales as $O(Nn^2)$ for learning an $n \times n$ transform from N training vectors.

4. EXPERIMENTS

In this section, we first illustrate the convergence of the cost function and iterates for our alternating algorithm. We then show that highly sparse transforms B can be learnt for natural images with only a marginal loss in performance, and in fact, with dramatic gains in speed of learning. Such doubly sparse transforms will be shown to be substantially better than analytical ones such as DCT. Finally, we illustrate the generalizability of learnt doubly sparse transforms and their promise for image compression.

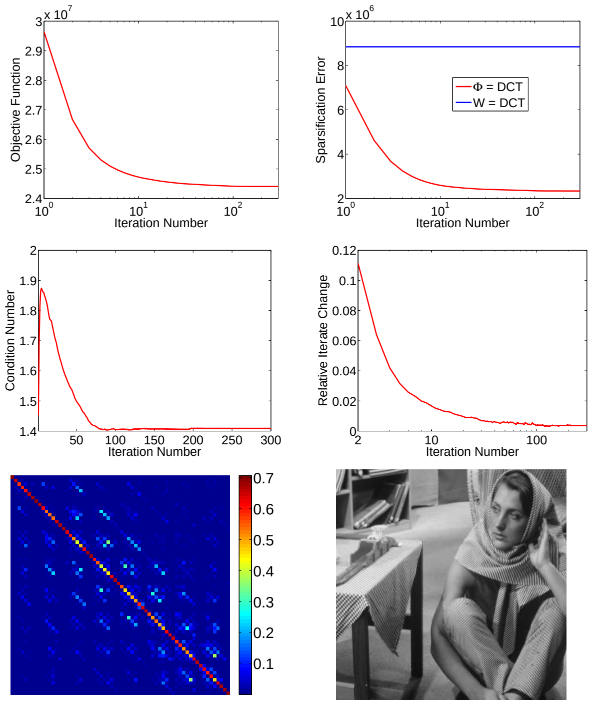

Fig. 1. Top: Objective function vs. iterations (Left), Sparsification error vs. iterations; horizontal line corresponds to sparsification error of $W = DCT$ (Right). Middle: Condition number of B vs. iterations (Left), Relative iterate change vs. iterations (Right). Bottom: Magnitude image of the learnt sparse matrix B (Left), Image recovered from sparse code (Right).

We employ our doubly sparse transform learning Problem formulation (P2) for learning adaptive sparse representations of image patches. The means (or DC values) of the patches are removed and we only sparsify the mean-subtracted patches which are stacked as columns of Y (patches represented as vectors). The means are added back for image display. Mean removal is typically adopted in image processing applications such as K-SVD based image denoising.

We introduce several metrics to evaluate the performance of learnt transforms. We define the *normalized sparsification error* as $\|B\hat{Y} - X\|_F^2 / \, \|B\hat{Y}\|_F^2$. This measures the fraction of energy lost in sparse fitting. We also define the *normalized recovery error* as $\left\|Y - W^{-1}X\right\|_F^2 / \ \|Y\|_F^2$. This measures the error (normalized) in recovering the data Y from their sparse codes obtained by thresholding WY with $W = B\Phi$. A closely related metric is the recovery peak signal to noise ratio (or *recovery PSNR*) defined as $255\sqrt{P} / \left\|Y - W^{-1}X\right\|_F$ in dB, where P is the number of image pixels. The *condition number* of W and the *sparsity* of B are the other measures of usefulness and efficiency of the doubly sparse transform.

Experiment 1. For the first experiment, we learn a doubly sparse transform for the non-overlapping 8×8 patches extracted from the Barbara image [2]. The (zero mean) data matrix Y has 4096 training signals and we work with $T_0 = 11$. We fix the transform Φ in the experiment to be the 2D DCT matrix ($\Phi = \Phi_0 \otimes \Phi_0$ where Φ_0 is the 8×8 1D DCT matrix and "$\otimes$" denotes the Kronecker product). The algorithm is initialized with an identity matrix for B. The other algorithm parameters are $\lambda = \mu = 4 \times 10^5$, and

$T_1 = 0.25 \times n^2$ (25% transform sparsity). The CG algorithm was run for 30 iterations with a fixed step size of 2×10^{-9}.

Figure 1 shows the progress of the algorithm over iterations. The objective function, sparsification error, and condition number all converge quickly. The condition number of B, which is also the condition number of $W = B\Phi$ for orthonormal Φ, converges to a low value of 1.41. The horizontal line in the sparsification error plot corresponds to the sparsification error of the patch-based 2D DCT transform (i.e., $W = \Phi = DCT$) at $T_0 = 11$. Our algorithm improves/decreases the sparsification error by 5.8 dB compared to the analytical DCT transform.

The normalized sparsification error and normalized recovery error for the learnt transform $W = B\Phi$ (with X obtained by thresholding WY) are 0.0450 (or 4.5%) and 0.0474 respectively. The two errors are similar for well-conditioned transforms. On the other hand, the normalized errors (both sparsification/recovery) corresponding to 2D DCT for the zero mean Y are much worse at 0.0676. The results here indicate that image patches admit reasonably good doubly sparse transform representations.

The learnt sparse matrix B is shown at the bottom of Figure 1. We also show the sparse approximation image recovered as $W^{-1}X$, which has a recovery PSNR of 34.39 dB. In contrast, the image recovered using the fixed, patch-based 2D DCT has a much worse PSNR of 32.85 dB.

Finally, we plot in Figure 1, the evolution of $\|B_i - B_{i-1}\|_F$ / $\|B_{i-1}\|_F$. This term measures the relative change between successive iterates, and can be used as a stopping condition for the algorithm. It decreases to a low value of 3.6×10^{-3} indicating convergence of the iterates.

Note that while the experiment here was performed with Φ as the DCT matrix, the behavior of the algorithm is similar for other Φ matrices such as Hadamard. The initial iterations of the algorithm can also be executed with few or no transform thresholding steps. This leads to a better initialization. Hence, in the following experiments, we perform no post-thresholding for the first 40 iterations.

Experiment 2. We now work with the same data as for Figure 1 and test the performance of the algorithm as a function of the sparsity levels T_0 and T_1. All parameters are fixed as in the experiment of Figure 1 (except that a larger step size of 10^{-8} is used for large T_1 such as 50% to ensure faster convergence). The algorithm terminates (here and in the rest of the experiments) when the relative iterate change between successive iterates B drops below a small threshold empirically chosen here as 0.1%. Figure 2 plots the variation of normalized sparsification error, recovery PSNR (of Barbara), and condition number for the learnt transform as a function of percentage sparsity of B (i.e., $\frac{T_1}{n^2}$). This is done at different data sparsity levels $T_0 = 10, 15, 20$. The normalized sparsification error increases monotonically as the transform sparsity level is reduced. The condition number, however, changes only slightly as a function of T_1 (its change/increase with decreasing T_1 is more drastic at lower values of λ, μ). Moreover, the recovery PSNR decreases when the transform sparsity level is lowered. This behavior is expected because as its sparsity level decreases, the transform has fewer degrees of freedom to adapt. Likewise, all the metrics also degrade when the data sparsity level T_0 is reduced. However, the values of all metrics (for fixed T_0) are reasonable even at low transform sparsities such as 10-15 % (i.e., the values at 10 % are not too different from those at 100%). This indicates that we can learn highly sparse transforms B with only a marginal loss in performance.

The normalized sparsification errors with the patch-based 2D DCT at T_0 values of 10, 15, and 20 are 0.0782, 0.0393, and 0.0211 respectively. The corresponding recovery PSNRs are 32.22 dB, 35.21 dB, and 37.91 dB respectively. For a fixed T_0, the learnt doubly sparse transforms have better normalized sparsification errors and recovery PSNRs than 2D DCT even at very low transform sparsities such as 2%. This indicates the promise of efficient adaptive transforms over analytical ones. As we will see later, the performance advantage of learnt transforms increases with patch size.

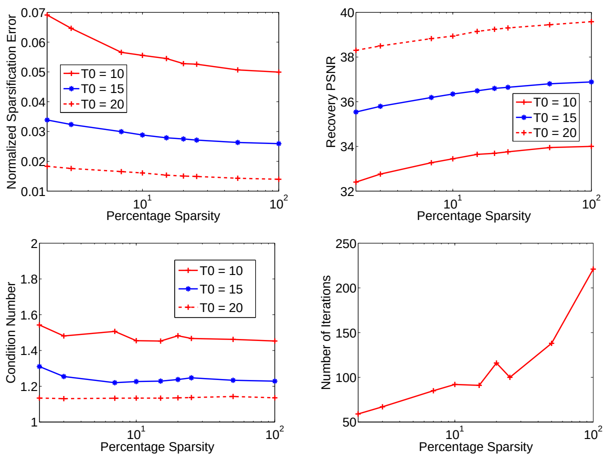

Fig. 2. Normalized Sparsification error (Top Left), Recovery PSNR (Top Right), and Condition number (Bottom Left), vs. percentage transform sparsity at different T_0 values. Algorithm iteration count vs. percentage transform sparsity for $T_0 = 15$ (Bottom Right).

Next, turning to computational cost, we plot the number of algorithm iterations as a function of transform sparsity for $T_0 = 15$ (the behavior is similar for other T_0 values). The number of required algorithm iterations is much lower at lower transform sparsity levels as compared to 100% transform sparsity (which corresponds to the 'unstructured' transform). The reduction in iteration counts is 2-4 fold depending on the value of T_1 (this reduction is not a monotone function of T_1, possibly an artifact due to the specific choice of the stopping criterion). Moreover, the computational cost per algorithm iteration is lower at smaller values of T_1 since operations with sparse matrices are faster. Currently, the run time per iteration is upto about 2 times lower at smaller T_1 compared to $T_1 = n^2$. We expect the speedups to be much greater with code optimization. Thus, doubly sparse transforms (which have fewer free parameters) can be learnt much faster than 'unstructured' ones.

Experiment 3. In the final experiment, designed to study the generalizing power of learnt transforms, we consider a magnetic resonance image dataset [10]. Of the 16 images in the dataset, 6 are used for training. A total of 41,334 overlapping patches of size 12×12 ($n = 144$) are extracted from the training images using a patch overlap stride of $r = 3$ (cf. [4] for the definition of overlap stride). The formulation of Problem (P2) is used to learn a doubly sparse transform adapted to these 41,334 patches after removing their means. The algorithm parameters are set as $\lambda = \mu = 10^7$, $T_0 = 0.17 \times n$, and $T_1 = 0.15 \times n^2$ (rounded to nearest integers). The 'globally' (called so because it is learnt over a variety of images and tested on others) learnt doubly sparse transform has a condition number of 1.19 and normalized sparsification error of 0.0302 for training patches. We also learnt a 'global' transform at T_1 of 100%, which had a normalized sparsification error and condition number of 0.0292 and 1.17 respectively.

Table 1 considers three unrelated test images from the dataset [10] labeled as R1, R2, and R3 (not used in training, shown in Figure 3), and lists the normalized sparsification errors, recovery PSNRs,

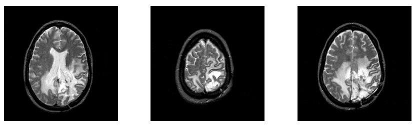

Fig. 3. Left to Right: The three test images - R1, R2, and R3.

	CN-I	NSE-I	RP-I	NSE-D	RP-D	NSE-G	RP-G
R1	1.15	0.0207	36.84	0.0403	33.98	0.0308	34.86
R2	1.11	0.0143	39.97	0.0347	36.11	0.0267	36.93
R3	1.14	0.0184	37.15	0.0378	34.04	0.0300	34.77

Table 1. Normalized sparsification errors (NSE-G) and recovery PSNRs (RP-G) for the globally adapted transform at T_1 of 15%, along with the corresponding quantities for 2D DCT (NSE-D/RP-D), and for image-specific transforms (NSE-I/RP-I) at T_1 of 15%, and the condition numbers (CN-I) of the image-specific transforms.

and condition numbers of doubly sparse transforms of transform sparsity 15% learnt on 12×12 non-overlapping patches for each of the test images (hence, also called image-specific transforms). The normalized sparsification errors and recovery PSNRs obtained by applying the globally learnt transform at T_1 of 15% on each of these images are also listed along with the corresponding values for the 144×144 2D DCT.

Both the global and image-specific transforms (which are also well-conditioned) perform significantly better than 2D DCT for all the test images. The image-specific transform provides upto 3.85 dB better normalized sparsification error and 3.86 dB better recovery PSNR than the 2D DCT. The global transform, although adapted to a particular training set, provides promising improvements of upto 0.88 dB in recovery PSNR and 1.17 dB in normalized sparsification error over 2D DCT on the test images. This indicates that the global transform is able to learn the common properties of the magnetic resonance images, and is hence able to represent the test images more effectively. The results point to the promise of both the global and image-specific doubly sparse transforms for image compression (the latter have the overhead of learning and encoding/storing the learnt sparse transform for each image).

Next, we compare the generalizability of unstructured (T_1 of 100 %) vs. doubly sparse (T_1 of 15%) globally learnt transforms. Table 2 lists performance metrics of image-specific unstructured transforms along with the corresponding values for the global unstructured transform when applied to each of the test images. The image-specific unstructured transforms perform better than the corresponding doubly sparse (image-specific) transforms of Table 1 while the latter are much cheaper to learn. In the case of the global transform, the doubly sparse transform of Table 1 performs almost as well as an unconstrained one on test images. The results also indicate a larger performance gap between the image-specific and global transforms at T_1 of 100% than the corresponding gap at T_1 of 15%. Apparently, reducing the number of degrees of freedom in the global transform by the sparsity constraint prevents over-fitting to a particular training image and enables good generalization. The results here can be improved further by optimal choice of λ and μ.

The performance of all our adaptive transforms improves with increasing patch size. For example, with $n = 196$, when evaluated over the test images, the globally learnt doubly sparse transform (at T_1 of 15%) provides upto 1.16 dB and 1.47 dB improvements in recovery PSNR and in normalized sparsification error, respectively, over 2D DCT. The corresponding improvements over 2D DCT by the image-specific doubly sparse transforms (at T_1 of 15%) are 5.76 dB and 5.72 dB respectively.

	CN-I	NSE-I	RP-I	NSE-G	RP-G
R1	1.07	0.0127	38.94	0.0303	34.95
R2	1.04	0.0066	43.32	0.0260	37.06
R3	1.08	0.0118	39.05	0.0294	34.85

Table 2. Normalized sparsification errors (NSE-G) and recovery PSNRs (RP-G) for the globally adapted transform at T_1 of 100%, along with the corresponding quantities for image-specific transforms (NSE-I/RP-I) at T_1 of 100%, and the condition numbers (CN-I) of the image-specific transforms.

When a global doubly sparse transform is used in applications, the computational cost of applying it on a test vector is $CT_1 + T_\Phi$ where C is a constant and T_Φ is the cost of applying Φ (e.g. $T_\Phi = n \, log \, n$ for DFT). For small T_1, this cost is much lower than the cost of applying an unstructured transform which scales as n^2 and moreover, only involves a small overhead above the fast analytical Φ. Using a sparse linear system solver and a fast implementation of the analytical transform Φ^{-1}, the cost of recovery from sparse code x by $W^{-1}x = \Phi^{-1}B^{-1}x$ is also lower for doubly sparse transforms compared to unstructured ones.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel problem formulation for learning 'doubly sparse' transforms. Our formulation gives rise to significantly sparse and reasonably-conditioned transforms with much lower sparsification errors than analytical transforms. Moreover, imposing the doubly sparse property leads to faster learning and faster computations with the sparse transform. The adapted transform also has a reduced storage requirement and generalizes better than the 'unstructured' (or non-sparse) transform. The usefulness of doubly sparse transform learning in image compression and other applications merits further study.

6. REFERENCES

[1] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," *Inverse Problems*, vol. 23, no. 3, pp. 947–968, 2007.

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

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

[4] S. Ravishankar and Y. Bresler, "MR image reconstruction from highly undersampled k-space data by dictionary learning," *IEEE Trans. Med. Imag.*, vol. 30, no. 5, pp. 1028–1041, 2011.

[5] M. Yaghoobi, S. Nam, R. Gribonval, and M. Davies, "Analysis operator learning for overcomplete cosparse representations," in *EUSIPCO*, 2011.

[6] R. Rubinstein and M. Elad, "K-SVD dictionary-learning for analysis sparse models," in *Proc. SPARS11*, June 2011.

[7] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[8] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1553–1564, 2010.

[9] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms for image processing," in *IEEE Int. Conf. Image Process.*, 2012, to appear.

[10] "CVG-UGR-Image database," `http://decsai.ugr.es/cvg/dbimagenes/index.php`, 2012, [Online; accessed 29-April-2012].