Learning Sparsifying Transforms

Saiprasad Ravishankar, *Student Member, IEEE,* and Yoram Bresler, *Fellow, IEEE*

***Abstract*—The sparsity of signals and images in a certain transform domain or dictionary has been exploited in many applications in signal and image processing. Analytical sparsifying transforms such as Wavelets and DCT have been widely used in compression standards. Recently, synthesis sparsifying dictionaries that are directly adapted to the data have become popular especially in applications such as image denoising, inpainting, and medical image reconstruction. While there has been extensive research on learning synthesis dictionaries and some recent work on learning analysis dictionaries, the idea of learning sparsifying transforms has received no attention. In this work, we propose novel problem formulations for learning sparsifying transforms from data. The proposed alternating minimization algorithms give rise to well-conditioned square transforms. We show the superiority of our approach over analytical sparsifying transforms such as the DCT for signal and image representation. We also show promising performance in signal denoising using the learnt sparsifying transforms. The proposed approach is much faster than previous approaches involving learnt synthesis, or analysis dictionaries.**

***Index Terms*—Sparsifying transforms, Dictionary learning, Signal denoising, Sparse representation, Compressed Sensing.**

I. Introduction

Sparse representation of signals has become widely popular in recent years. It is well known that natural signals and images have an essentially sparse representation (few significant non-zero coefficients) in analytical transform domains such as Wavelets [1] and discrete cosine transform (DCT). This property has been exploited in designing various compression algorithms and in compression standards such as JPEG2000 [2].

While transforms are a classical tool in signal processing, alternative models have also been studied for sparse representation of data. Two such well-known models are the synthesis and analysis models [3]. More recently, attention has turned to adapting these models to data. These approaches are known in the literature as synthesis dictionary learning [4]–[9] and analysis dictionary learning [10]–[15]. Surprisingly, the adaptation or learning of transforms has received no attention. In this paper, we develop formulations aimed at learning sparsifying transforms from data. In the following, we discuss the synthesis and analysis models and their learning, and elucidate their drawbacks. This discussion will also serve to highlight the fundamental differences between the transform model and the synthesis/analysis models. The former will be shown to possess important advantages.

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

S. Ravishankar and Y. Bresler are with the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois, Urbana-Champaign, IL, 61801 USA e-mail: (ravisha3, ybresler)@illinois.edu.

A. Synthesis Model for Sparse Representation

A popular sparse representation model is the *synthesis model*, which states that a signal $y \in \mathbb{R}^n$ may be represented as a linear combination of a small number of atoms/columns from a dictionary $D \in \mathbb{R}^{n \times K}$ [3], [16]. This means $y = Dx$, where $x \in \mathbb{R}^K$ is sparse with $\|x\|_0 \ll K$, and the l_0 quasi-norm counts the number of non-zeros in x. In practice, "real world" signals are expected to deviate from this model. Therefore, the data is more generally assumed to satisfy $y = Dx + \xi$, where ξ is an error or noise term in the signal domain [16]. When $n = K$, and D is full rank, the dictionary D is said to be a basis, whereas when $K > n$, the dictionary is said to be overcomplete.

The approach of representing a signal in a synthesis dictionary D has received considerable attention recently. Specifically, the problem shown in (1), of extracting the sparse representation of a signal y in a synthesis dictionary D, has been studied in great detail in recent years [17]–[20], and is popularly known as the sparse coding problem [1].

$$\min_{x} \|y - Dx\|_2^2 \ \ s.t. \ \ \|x\|_0 \leq s \tag{1}$$

Here, parameter s represents the desired sparsity level. Although this problem is NP-hard (Non-deterministic Polynomial-time hard) and therefore computationally infeasible in general, under certain conditions it can be solved exactly using polynomial-time algorithms [17]–[23]. Other algorithms, for which similar theoretical performance guarantees may not be available, often provide even better empirical performance [24]–[28]. All these various algorithms are, however, computationally expensive in practice [16].

The process of obtaining a sparse representation for a signal/image requires explicit knowledge of the synthesis dictionary D. Many analytical dictionaries have been developed for sparse image representation such as the Ridgelet [29], Contourlet [30], and Curvelet [31] dictionaries.

The idea of learning a synthesis dictionary from training signals has also been exploited [4]–[7]. Given a matrix $Y \in \mathbb{R}^{n \times N}$ whose columns represent training signals, the problem of learning a dictionary D that gives a sparse representation for the signals in Y, can be formulated as follows [6].

$$\text{(P0s)} \ \ \min_{D,X} \|Y - DX\|_F^2 \ \ s.t. \ \ \|X_i\|_0 \leq s \ \forall \ i \tag{2}$$

Here, the columns X_i of matrix $X \in \mathbb{R}^{K \times N}$ with maximum allowed sparsity level s are the sparse representations, or sparse codes, of the signals/columns in Y, and $\|\cdot\|_F$ denotes the Frobenius norm. The columns of the learnt dictionary in (P0s) are typically constrained to be of unit norm (or normalized) in order to avoid the scaling ambiguity [32].

[1] Alternative formulations swap the constraint and cost, or use a Lagrangian formulation.

Problem (P0s) is to minimize the fitting error of the training signals with respect to the dictionary D subject to sparsity constraints. This problem too is NP-hard (for each fixed D, it requires the solution of N sparse coding problems), and unlike the sparse coding problem, there are no known polynomial-time algorithms for its exact solution under practically interesting conditions. Nonetheless, several popular heuristics have been developed for the solution of (P0s), or its variations, in recent years [5]–[9]. The algorithms typically alternate between finding the dictionary D (dictionary update step), and the sparse representations X (sparse coding step). The sparse coding step is always solved with fixed D. Of the various algorithms, the K-SVD algorithm [6] has been especially popular in applications.

Unfortunately, since Problem (P0s) is non-convex, methods such as K-SVD can get easily caught in local minima or even saddle points [33]. Moreover, the sparse coding step in such methods is computationally expensive. Although there has been some preliminary theoretical analysis of dictionary identification [32], no proof exists to date for the general global convergence of K-SVD or other synthesis dictionary learning algorithms.

Adaptive synthesis dictionaries have been shown to be useful in a variety of applications such as image/video denoising, image/video inpainting, deblurring, demosaicing [34]–[40], clustering and classification [41], and tomography [42]. Recently, we have shown impressive performance for learnt dictionaries in magnetic resonance imaging (MRI) involving highly undersampled k-space data [43], [44].

B. Analysis Model for Sparse Representation

Another model for sparse representation of data is the *analysis model* [3]. This model suggests that given a signal $y \in \mathbb{R}^n$ and operator $\Omega \in \mathbb{R}^{m \times n}$, the representation $\Omega y \in \mathbb{R}^m$ is sparse, i.e., $\|\Omega y\|_0 \ll m$ [12], [15]. Here, Ω is known as the analysis dictionary, since it "analyzes" the signal y to produce a sparse result. The zeros of Ωy essentially define the subspace to which the signal belongs, and the total number of these zeros is called co-sparsity [12], [15]. A well-known analysis dictionary is the finite difference dictionary. Elad et al. [3] derived conditions for the equivalence of analysis and synthesis based priors.

When the given signal $y \in \mathbb{R}^n$ is contaminated with noise, the analysis model is extended as $y = q + \xi$, with Ωq being sparse, and ξ representing the noise [12], [15]. We refer to this as the *noisy signal analysis model*. Given the noisy signal y and dictionary Ω, the problem of recovering the clean signal q is as follows [12], [15].

$$\min_{q} \|y - q\|_2^2 \;\; s.t. \;\; \|\Omega q\|_0 \leq m - t \tag{3}$$

Here, parameter t represents the minimum allowed co-sparsity. This problem is called the analysis sparse coding problem [15]. It can also be viewed as a denoising scheme. This problem too is NP-hard, just like sparse coding in the synthesis case. Approximate algorithms have been proposed in recent years for solving the analysis sparse coding problem [12], [15], which similar to the synthesis case are also computationally expensive. Note that when the analysis dictionary Ω is square and non-singular, we have $q = \Omega^{-1} v$ for some sparse v, and the problem of finding q above is identical to a synthesis sparse coding problem of finding v, where $D = \Omega^{-1}$ is the synthesis dictionary. While (3) uses the l_0 quasi norm for sparsity, some authors [14] alternatively consider a convex relaxation of the problem in which they replace it with an l_1 norm that they add as a penalty in the cost.

While there has been considerable research on the learning of synthesis dictionaries, the idea of learning analysis dictionaries has received only recent attention. Peyré et al. [10] learn a dictionary in a sparsity-promoting analysis-type prior. Analysis dictionary learning has been pursued by several authors. Given the training matrix $Y \in \mathbb{R}^{n \times N}$, Ophir et al. [13] and Yaghoobi et al. [11] minimize the l_0 and l_1 norm of ΩY respectively. However, the (global) convergence of these learning algorithms is unclear, even in the case when l_1 relaxation is used for sparsity.

Some authors alternatively consider analysis dictionary learning with the noisy signal analysis model [12], [14], [15]. Rubinstein et al. [12], [15] proposed the following formulation for the noisy signal case that involves the l_0 quasi norm for sparsity.

$$\text{(P0a)} \;\; \min_{\Omega, Q} \|Y - Q\|_F^2 \;\; s.t. \;\; \|\Omega Q_i\|_0 \leq m - t \;\; \forall \; i \tag{4}$$

The columns of matrix ΩQ are constrained to have a minimum co-sparsity of t. In (P0a), the rows of Ω are also typically constrained to be of unit norm [14], [15].

Problem (P0a) is non-convex and NP-hard. However, heuristics have been proposed for its solution such as the analysis K-SVD algorithm [15] which alternates between updating Q (analysis sparse coding), and Ω. Still, no convergence guarantees exist for these algorithms. Indeed, because these algorithms are similar to those proposed in the synthesis case, we expect that they can get similarly stuck in bad local minima.

The promise of learnt analysis dictionaries in signal and image processing applications has not been sufficiently explored. Furthermore, it is unclear whether the recent methods can outperform previous synthesis based approaches such as K-SVD [6]. On the contrary, in preliminary experiments, Yaghoobi et al. [14], [45] show that the learned operator denoises not much better than even a fixed analytical dictionary.

C. Transform Model for Sparse Representation

In this paper, we pursue a generalized analysis model for sparse representation of data, which we call the transform model. This model suggests that a signal $y \in \mathbb{R}^n$ is *approximately sparsifiable* using a transform $W \in \mathbb{R}^{m \times n}$, that is $Wy = x + e$ where $x \in \mathbb{R}^m$ is sparse, i.e., $\|x\|_0 \ll m$, and e is the representation error or residual in the transform domain that is assumed to be small (of small norm). The transform model suggests that signals can be approximately sparse in the transform domain. This can be viewed as a generalization of the analysis model with Ωy exactly sparse. Additionally, unlike the analysis model in which the sparse representation Ωy lies in the range space of Ω, the sparse representation x in the transform model is not constrained to lie in the range space of W. The generalization allows the transform model

to include a wider class of signals within its ambit than the analysis model.

The transform model $Wy \approx x$ with x sparse, is also more general than the notion of compressibility [46] applied to Wy, where the assumption would be that the coefficients of Wy, when arranged in non-increasing magnitude order, follow a power law decay. The transform model only assumes that the residual $Wy - x$ has small energy compared to the energy in x. Well-known examples of sparsifying transforms for natural signals and images include the Wavelets and DCT.

The reason we have chosen the name "transform model" for our approach is because the assumption $Wy \approx x$ has been traditionally used in transform coding (with orthonormal transforms), and the concept of transform coding is older [47] and pre-dates the terms analysis and synthesis [48].

The transform model is more general not only than the analysis model, but also than the noisy signal analysis model. To demonstrate this point, we use for convenience, the same symbol W for both the transform and analysis operator.

First, if we assume that the signal y satisfies the noisy signal analysis model, we have the representation $y = q + \xi$ with $z' = Wq$ being sparse. This implies that

$$Wy = W(q + \xi) = Wq + W\xi = z' + e' \tag{5}$$

where $e' = W\xi$ is the error in the transform domain of W. Thus, z' is a transform code with error e' for the noisy y. Hence, it is true that if a signal y satisfies the noisy signal analysis model, it also satisfies a corresponding transform model. However, the converse is not true, in general. That is, for a given signal y, if the transform model $Wy = x + e$ holds with sparse x, then the relation $y = q' + \xi'$, with $Wq' = x$ need not hold for any error ξ'. This is because the transform model does not restrict its sparse code x to lie in the range space of W (i.e., $Wq' = x$ need not be true for any q', especially for a tall/overcomplete transform W). Hence, if a signal satisfies the transform model, it need not satisfy a corresponding noisy signal analysis model. In this sense, the noisy signal analysis model is restrictive compared to the transform model, or, in other words, the transform model is more general. In this paper, we will also consider an explicit extension of the transform model for the "noisy signal" case, that makes it even more general.

We note that in the aforementioned arguments, the relation $Wq' = x$ can be satisfied when the transform W is square and full rank (invertible), in which case $q' = W^{-1}x$. However, although for the square and invertible case, the noisy signal analysis model translates to a corresponding transform model and vice-versa, the error ξ in the signal domain in the noisy signal analysis model becomes $e' = W\xi$ in the transform domain and conversely, the error e in the transform domain for the transform model becomes $\xi' = W^{-1}e$ in the signal domain. Thus, even small errors in one model can be amplified, when translated to the other model for a poorly conditioned matrix W. (That is, $\|W\xi\|_2$ or its "normalized" version $\|W\xi\|_2 / \|Wy\|_2$, and $\left\|W^{-1}e\right\|_2$ or its "normalized" version $\left\|W^{-1}e\right\|_2 / \|y\|_2$, can be large.) This indicates that the noisy signal analysis model can translate to a bad transform model and vice-versa.

When a sparsifying transform W is known for the signal y, the process of obtaining a sparse code x of given sparsity s involves solving the following problem, which we call the transform sparse coding problem.

$$\min_x \|Wy - x\|_2^2 \;\; s.t. \;\; \|x\|_0 \leq s \tag{6}$$

This problem minimizes the transform domain residual. The solution $\hat{x}$ is obtained exactly by thresholding the product Wy and retaining the s largest coefficients. In contrast, sparse coding with the synthesis or analysis dictionaries involves in general an NP-hard problem. Thus, a sparsifying transform is much simpler and faster to use in practice. Given the transform W and sparse code x, we can also recover a least squares estimate of the true signal y by minimizing $\|Wy - x\|_2^2$ over all $y \in \mathbb{R}^n$. The recovered signal is then simply $W^\dagger x$, where $W^\dagger$ is the pseudo-inverse of W.

While analytical sparsifying transforms such as the Wavelets, DCT, and discrete fourier transform (DFT) have been extensively used in many applications such as compression, denoising, and compressed sensing [49], [50], adapting the sparsifying transform to the data could make it more effective in these applications. In this adaptive setting, given a matrix Y of training signals, we would like to learn a transform W such that the *sparsification error* $\|WY - X\|_F^2$ is minimized. Here, X is again a matrix (unknown) containing the sparse codes of the training signals as columns. This idea of learning a sparsifying transform, is the subject of this paper. In this paper, we restrict ourselves to learning square transforms, i.e., $W \in \mathbb{R}^{n \times n}$.

We propose a novel framework for transform learning that aims to learn well-conditioned transforms, and provide algorithms to solve the proposed problems. The learnt transforms will be shown to be useful in applications such as signal denoising. In contrast to the prior work on synthesis and analysis dictionary learning, the transform model allows WY to be approximated by a sparse matrix X, which makes the learning of this model easier. Specifically, with the transform model, sparse coding is cheap and exact. Furthermore, unlike synthesis or analysis dictionary learning, the proposed transform learning formulation does not include a highly non-convex function involving the product of two unknown matrices. Moreover, while the convergence analysis in this paper is only partial, it shows monotone convergence of the objective function in the proposed algorithm. Empirical results demonstrate convergence of the iterates regardless of initial conditions. This desirable convergence behavior contrasts with that of current synthesis or analysis learning algorithms, for which no such results, theoretical or empirical, are available.

The rest of the paper is organized as follows. Our problem formulations for learning sparsifying transforms are described in Section II. Section III details the proposed algorithms for transform learning and discusses their relevant properties such as convergence and computational cost. In Section IV, we introduce a novel signal denoising framework incorporating sparsifying transforms. Section V demonstrates the performance of our algorithms including promising performance of learnt transforms in denoising. In Section VI, we conclude with proposals for future work.

II. Transform Learning Problems and Properties

A. Trivial Solutions, and Full Rank Constraint

Given the training matrix $Y \in \mathbb{R}^{n \times N}$, we propose to minimize the sparsification error given by $\|WY - X\|_F^2$, where $W \in \mathbb{R}^{m \times n}$ and X is column-sparse.

$$\text{(P1)} \quad \min_{W,X} \|WY - X\|_F^2 \;\; s.t. \;\; \|X_i\|_0 \leq s \;\; \forall \; i \qquad (7)$$

The minimization corresponds to fitting the transform model parametrized by W and X to the data Y, by selecting the best (matrix) parameters W and X. The notion of sparsification error can be justified by the fact that natural signals and images can be reasonably approximated in a transform domain (such as Wavelets) using few (here s) significant non-zero transform coefficients. The rest of the coefficients contribute only marginally to the sparsification error.

Problem (P1), although intuitive, has a glaring defect. It has a trivial solution $W = 0, X = 0$. In order to avoid the trivial solution, we need to enforce additional constraints or penalties. Introducing a constraint on the norm of W or of its rows (similar to constraints in synthesis [6], [7], [32] or analysis [14], [15] dictionary learning) may seem apt for this setting. However, such a constraint does not preclude the possibility of repeated rows in W. That is, if a particular non-trivial row vector can individually provide the lowest sparsification error for Y (compared to any other row vector), the best solution would be to form W by repeating this row. This problem is exacerbated in the case when Y has rank $n-1$ or lower (i.e., is rank deficient). In this case, all the rows of W can be chosen as scaled versions of a left null vector (left null space has dimension 1 or higher) of Y, leading to a zero sparsification error. However, the resulting W has repeated rows and is not useful.

Therefore, an appropriate constraint must preclude zero rows, repeated rows, or even linearly dependent rows (except when W is a tall matrix, in which case linear dependence of the rows cannot be avoided but may be minimized). In the following, we consider a formulation for the case of square W ($m = n$).

We propose a full rank constraint to address the ambiguities in the square case. We note that many well-known square sparsifying transforms, such as the DCT, Wavelets, and Hadamard, are non-singular. The one-dimensional finite difference transform, which typically has fewer rows than columns, can be appended with a linearly independent row(s), thereby making it non-singular. The two-dimensional finite difference transform can be obtained as the Kronecker product of two such non-singular (one-dimensional) matrices.

However, full rank is a non-convex and non-smooth constraint. Hence, in the square W setting, we propose to add a negative log-determinant (of W) penalty [51] in the cost. Such a constraint penalizes transforms that have a small determinant (note that full rank is trivially imposed by this constraint).

$$\text{(P2)} \quad \min_{W,X} \|WY - X\|_F^2 - \lambda \log\det W \qquad (8)$$
$$s.t. \;\; \|X_i\|_0 \leq s \;\; \forall \; i$$

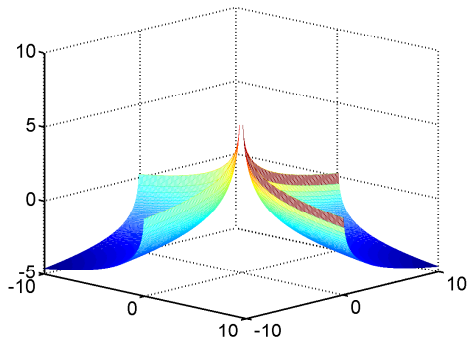


Fig. 1. Plot of $-\log\det W$ for 2×2 diagonal matrices W. The horizontal axes indicate the first and second diagonal entry values.

Problem (P2) is non-convex. To illustrate this fact, we plot in Figure 1, $-\log\det W$ for 2×2 diagonal matrices W as a function of the diagonal elements. The function is well-defined and symmetric in the first and third quadrants (W is positive definite in the first quadrant and negative definite in the third). While the function is convex within each of these quadrants, it is non-convex overall. Note that the domain of the function, which is the union of the first and third quadrants, is also a non-convex set.

One could constrain W in (P2) to be positive definite, $W \succ 0$. If in addition the l_0 quasi norm for X were relaxed to an l_1 norm, the resulting problem would be jointly convex in W and X. However, enforcing the transform to be positive definite is too restrictive in general. Many well-known sparsifying transforms such as the DCT and Wavelets are not positive definite. Hence, a positive definite constraint will preclude many good candidate transforms from the class of solutions. Therefore, we do not use such a constraint.

B. Scale Ambiguity and Conditioning

When the data Y admits an exact representation, i.e., there exists a pair $(\tilde{W}, \tilde{X})$ with $\tilde{X}$ sparse such that $\tilde{W}Y = \tilde{X}$, then the cost in (P2) can be made arbitrarily small by pre-multiplying $\tilde{W}$ and $\tilde{X}$ by a scalar α (or equivalently, by a diagonal matrix Γ with entries $\pm\alpha$ chosen such that $\det(\Gamma\tilde{W}) > 0$) with $\alpha \to \infty$. The cost becomes unbounded from below in this case, which spells trouble for optimization algorithms. We refer to this phenomenon as the 'scale' ambiguity.

Moreover, the negative log determinant penalty although it enforces full rank, does not necessarily imply good-conditioning. To elucidate the importance of conditioning, consider again the case of Y with rank $n-1$ or lower. Let $\hat{W}$ be a matrix that has the left singular vectors of Y as its rows. We can scale a row of $\hat{W}$ that contains a singular vector corresponding to a zero singular value by $\frac{1}{\epsilon^p}$, where $\epsilon > 0$ is small and p is an integer. All the other rows of $\hat{W}$ can be scaled by ϵ. In this case, such a scaled transform $\hat{W}$ is full rank (has non-zero $\det\hat{W} = \epsilon^{n-1-p}$, which is large for $p \gg n-1$), and provides a sparsification error (computed using the X, whose columns are obtained by thresholding the columns of $\hat{W}Y$) that is close to zero for sufficiently small ϵ. However, this transform is poorly conditioned (condition number $\kappa(\hat{W}) = \frac{1}{\epsilon^{p+1}}$, which is large for large p), and

produces only about as much good/non-redundant information as the single row with the large scaling (since the other rows are almost zero in scale). With better conditioning, we would expect the transform to produce more information and less degeneracies. The conditioning of a transform is also an important property for applications such as denoising.

Note that the transform $\hat{W}$ in the aforementioned example also suffers from the scale ambiguity (the objective function decreases without bound as $\epsilon \rightarrow 0$). While the example considers an extreme case of rank deficient Y, Problem (P2) was also empirically observed to give rise to ill-conditioned transforms for matrices Y constructed from image patches.

In order to address the scale ambiguity and the conditioning of the transform, we introduce an additional norm-constraint $\|W\|_F \leq 1$, which is convex.[2] Alternatively, the constraint can be added as a penalty in the cost to result in the following formulation.

$$\text{(P3)} \quad \min_{W,X} \|WY - X\|_F^2 - \lambda \log\det W + \mu \|W\|_F^2 \qquad (9)$$
$$s.t. \quad \|X_i\|_0 \leq s \ \forall \ i$$

While we use the l_0 quasi-norm for sparsity in Problem (P3), it could be substituted by a convex l_1 penalty in the cost (i.e., penalize $\|X\|_1 = \sum_{i=1}^{N} \|X_i\|_1$).

In the following, we will prove that the cost function of Problem (P3) is lower bounded and encourages well-conditioning. For convenience, we define

$$Q \triangleq -\log\det W + c\|W\|_F^2 \qquad (10)$$

Then, the cost function of Problem (P3) can be written as follows

$$\|WY - X\|_F^2 + \lambda Q \qquad (11)$$

with $c = \frac{\mu}{\lambda}$. Since, $\|WY - X\|_F^2 \geq 0$, we proceed to lower bound Q. We define

$$Q_0 \triangleq \frac{n}{2} + \frac{n}{2}\log(2c) \qquad (12)$$

Lemma 1. *Suppose $W \in \mathbb{R}^{n\times n}$ has positive determinant, and let $Q \triangleq -\log\det W + c\|W\|_F^2$, for some $c > 0$. Then,*

$$Q \geq \frac{n}{2} + \frac{n}{2}\log(2c) - \log\frac{2\kappa}{1+\kappa^2} \qquad (13)$$

where κ is the 2-norm condition number of W.

Proof: See Appendix A.

Corollary 1. *$Q \geq Q_0$, with equality if and only if $\kappa = 1$ and the singular values of W are all equal to $\sqrt{\frac{1}{2c}}$.*

Proof: See Appendix B.

It follows from corollary 1 that the cost function in equation (11) is lower bounded by λQ_0. Furthermore, Problem (P3) attains this lower bound if and only if there exists a pair $(\hat{W}, \hat{X})$ with $\hat{X}$ sparse (and $\det \hat{W} > 0$) such that $\hat{W}Y = \hat{X}$, and the singular values of $\hat{W}$ are all equal to $\sqrt{0.5\lambda/\mu}$ (hence, the condition number $\kappa(\hat{W}) = 1$). Thus, formulation (P3) does not suffer from the scale ambiguity (due to finite lower bound on cost), and favors both a low sparsification error and good conditioning.

Next, we demonstrate that the function Q can be used to derive an upper bound for $\kappa(W)$.

Proposition 1. *Suppose $W \in \mathbb{R}^{n\times n}$ has positive determinant, then*

$$1 \leq \kappa \leq e^{Q-Q_0} + \sqrt{e^{2(Q-Q_0)} - 1}$$

Proof: Inequality (13) in Lemma 1 can be rewritten as $\kappa^2 - 2\kappa e^{Q-Q_0} + 1 \leq 0$. Solving for κ, we get

$$e^{Q-Q_0} - \sqrt{e^{2(Q-Q_0)} - 1} \leq \kappa \leq e^{Q-Q_0} + \sqrt{e^{2(Q-Q_0)} - 1}$$

Since, the lower bound above is ≤ 1, we instead use the trivial lower bound of 1 for κ. ■

In Proposition 1, the upper bound on κ is a monotonically increasing function of Q. Hence, we conclude that in general, the minimization of the proposed cost function (11) (of Problem (P3)) encourages reduction of condition number.

Moreover, the following corollary shows that the solution to Problem (P3) is perfectly conditioned in the limit of $\lambda \rightarrow \infty$. Let the value of Q corresponding to the minimum/optimum value of the cost function (11) of Problem (P3) be denoted by Q^*, and the condition number of the minimizing transform(s) be denoted by κ^*. We then have the following result.

Corollary 2. *For a fixed $\frac{\mu}{\lambda}$, as $\lambda \rightarrow \infty$ in Problem* (P3)*, $\kappa^* \rightarrow 1$.*

Proof: By (12), for a fixed $c = \frac{\mu}{\lambda}$, Q_0 is fixed too. Suppose we were to minimize the function Q alone, then by corollary 1, the minimum value is Q_0 which is attained by any transform $\hat{W}$ (of positive determinant) that has all of its singular values equal to $\sqrt{\frac{\lambda}{2\mu}}$. Now, as λ is increased in (P3) with fixed $\frac{\mu}{\lambda}$, Q^* can only decrease because the Q part of the cost function (11) is weighted more heavily. In the limit as $\lambda \rightarrow \infty$, we get $Q^* \searrow Q_0$. By proposition 1, we then have that as $Q^* \searrow Q_0$, $\kappa^* \rightarrow 1$ (the upper bound on κ^* goes to 1 while the lower bound is 1). ■

The upper bound on κ^* decreases as λ is increased with fixed $\frac{\mu}{\lambda}$, suggesting that the minimizing transform(s) can be better conditioned at larger λ. This is also shown empirically in Section V.

Our Problem formulation (P3) thus, aims to minimize the sparsification error while controlling the condition number and eliminating the scale ambiguity, with the goal of estimating a "good" transform W that provides the best fit to the data. No particular performance metric in an application is directly optimized here. However, as we will show in Sections IV and V, W produced as a solution to Problem (P3), either by itself or via its extensions, performs well in applications such as signal representation/recovery, and denoising.

We note that a cost function similar to that in Problem (P3), but lacking the $\|W\|_F^2$ penalty (or in other words, similar to (P2)) has been derived under certain assumptions in a very different setting of blind source separation [52]. However, the

[2]The Frobenius-norm constraint could be alternatively replaced by constraining each row of W to unit norm. However, this alternative was found empirically to produce inferior transforms than those resulting from Problem (P3).

transform learning Problem (P3) performs poorly in signal processing applications in the absence of the crucial $\|W\|_F^2$ penalty, which as discussed earlier, helps overcome the scale ambiguity and control the condition number. The superiority of (P3) over (P2) is also illustrated empirically in Section V.

We also note that penalty terms similar to $-\log\det W$ and $\|W\|_F^2$ (of (P3)) can be used to regularize synthesis and analysis dictionary learning in order to enforce full-rank and well-conditioning, and overcome scale ambiguities.

C. Positive Determinant and Equivalent Solutions

As discussed earlier, Problem (P3) is non-convex. Moreover, there is an implicit constraint of $\det W > 0$ for $\log\det W$ to be well-defined. However, this constraint is non-restrictive, because if $(\tilde{W}, \tilde{X})$ is a non-trivial minimizer of the sparsification error term in (P3) (or equivalently solves Problem (P1)) with $\det\tilde{W} < 0$, then we can always form an equivalent pair $(\Gamma\tilde{W}, \Gamma\tilde{X})$ (where Γ is a diagonal "sign matrix" with ± 1 on the diagonal and $\det\Gamma < 0$), which provides the same sparsification error, but has $\det\Gamma\tilde{W} > 0$. Therefore, it suffices to only consider transforms that have $\det W > 0$.

Furthermore, the $\det W > 0$ constraint need not be enforced explicitly. This is because the cost function in (P3) has log-barriers (see Figure 1) in the space of matrices at W for which the determinant is less than or equal to zero. These log-barriers help prevent an iterative minimization algorithm initialized with W satisfying $\det W > 0$ from getting into the infeasible regions, where $\det W \leq 0$.

The problem has one remaining inherent ambiguity. Similar to synthesis dictionary learning [32], Problem (P3) admits an equivalence class of solutions/minimizers. Given a particular minimizer $(\tilde{W}, \tilde{X})$, we can form equivalent minimizers by simultaneously permuting the rows of $\tilde{W}$ and $\tilde{X}$ (only permutations that retain the sign of the determinant of $\tilde{W}$ are permitted). Pre-multiplying a minimizer by a sign matrix Γ with $\det\Gamma > 0$ also provides an equivalent minimizer. This ambiguity between completely equivalent solutions is of no concern, and we do not attempt to eliminate it. Which of the equivalent solutions will be obtained by an iterative algorithm for the solution of (P3) will depend on initialization.

III. Algorithm and Properties

A. Algorithm

Our algorithm for solving Problem (P3) alternates between updating X and W.

1) Sparse Coding Step: In this step, we solve Problem (P3) with fixed W.

$$\min_X \|WY - X\|_F^2 \;\; s.t. \;\; \|X_i\|_0 \leq s \;\; \forall \; i \tag{14}$$

The solution X can be computed exactly by thresholding WY, and retaining only the s largest coefficients in each column. Contrast this with the sparse coding step in synthesis dictionary learning [6], which can only be solved approximately using techniques such as OMP. If instead the l_0 quasi norm is relaxed to an l_1 norm and added as a penalty in the cost, we solve the following problem.

$$\min_X \|WY - X\|_F^2 + \eta \sum_{i=1}^{N} \|X_i\|_1 \tag{15}$$

The solution for X in this case can again be exactly computed by soft thresholding as follows.

$$X_{ij} = \begin{cases} (WY)_{ij} - \frac{\eta}{2} & , \;\; (WY)_{ij} \geq \frac{\eta}{2} \\ (WY)_{ij} + \frac{\eta}{2} & , \;\; (WY)_{ij} < -\frac{\eta}{2} \\ 0 & , \;\; else \end{cases} \tag{16}$$

Here, subscript ij indexes matrix entries. We will show that such soft thresholding is even cheaper than the projection onto the l_0 ball (14) by hard thresholding.

2) Transform Update Step: In this step, we solve Problem (P3) with fixed X. This involves the unconstrained minimization

$$\min_W \|WY - X\|_F^2 - \lambda \log\det W + \mu \|W\|_F^2 \tag{17}$$

This problem can be solved using methods such as steepest descent, or conjugate gradients. We can employ the conjugate gradient method with backtracking line search [53] (also known as Armijo rule), which typically converges faster than steepest descent. Fixed step size rules were also observed to work well and faster in practice.

The gradient expressions for the various terms in the cost [54] are as follows. We assume that $\det W > 0$ on some neighborhood of W, otherwise $\log()$ would be discontinuous.

$$\nabla_W \log\det W = W^{-T} \tag{18}$$

$$\nabla_W \|W\|_F^2 = 2W \tag{19}$$

$$\nabla_W \|WY - X\|_F^2 = 2WYY^T - 2XY^T \tag{20}$$

Various stopping rules can be used for the conjugate gradient iterations, such as the norm of the gradient of the objective function dropping below a threshold. However, the conjugate gradient algorithm typically converges quickly, and a fixed number of iterations were empirically observed to work well.

As described above, the update of W is performed with fixed X. Alternative strategies such as updating W jointly with the non-zero values in X (using the conjugate gradient method) for a fixed sparsity pattern of X (similarly to K-SVD), yielded similar empirical performance, albeit at a considerably higher computational cost.

For the proposed alternating algorithm to work, W must be initialized to have a positive determinant. The alternating algorithm itself typically converges quickly, and as shown in Section V, a fixed number of iterations suffices in practice.

B. Convergence

The algorithm for solving Problem (P3) alternates between sparse coding and transform update steps. The solution for the sparse coding step is exact/analytical. Thus, the cost function can only decrease in this step. For the transform update step, the solution is obtained by conjugate gradients (for instance with Armijo step size rule). Thus, in this step too, the cost function can again only decrease. The cost function being monotone decreasing and lower bounded, it must converge. While convergence of the iterates themselves for the proposed alternating minimization of the non-convex problems does not follow from this argument, the iterates are found empirically to converge as well.

C. Computational Cost

The algorithm for Problem (P3) involves Sparse coding steps and Transform Update steps. In the sparse coding step, when the ℓ_0 quasi norm is used for sparsity of X, then the projection onto the ℓ_0 ball by hard thresholding, if done by full sorting [55], involves $O(n \log n)$ comparisons per training signal, or a total of $O(nN \log n)$ operations. Using instead the ℓ_1 norm penalty for sparsity, the soft thresholding in (16) requires only $O(nN)$ operations, and is therefore cheaper than projecting onto the ℓ_0 ball. Either way, computing WY (prior to thresholding) requires $O(Nn^2)$ operations, and dominates the computation in the sparse coding step.

To estimate the cost of the Transform Update step, assume that YY^T has been pre-computed (at a total cost of $O(Nn^2)$ for the entire algorithm). The gradient evaluation in equation (20) involves the matrix products WYY^T and XY^T. Computing $W(YY^T)$ requires n^3 multiply-add operations. Furthermore, when X is sparse with Ns non-zero elements and $s = \alpha n$ (where $\alpha \ll 1$ typically), then computing the product XY^T requires αNn^2 multiply-add operations. (Note that when the ℓ_1 norm is used for sparsity of X, one would need to carefully choose the parameter η in (16), to get the αNn^2 cost for computing XY^T.) Next, the gradient evaluations in (18) and (19) are dominated (in computations) by $C_3 n^3$ (for the matrix inverse), where C_3 is a constant. Thus, the transform update step roughly has computational cost $\alpha NLn^2 + (1 + C_3)Ln^3$, where L is the number of conjugate gradient steps (typically fixed). Assuming $(1 + C_3)n < \alpha N$, and that αL is approximately constant, the cost per Transform Update step scales as $O(n^2 N)$.

The total cost per iteration (of Sparse coding and Transform Update) of the algorithms thus scales as $O(Nn^2)$. Contrast this with the cost per iteration of the synthesis dictionary learning algorithm K-SVD [6], which roughly scales as $O(sNn^2)$ (cost dominated by the sparse coding step) for the square dictionary case [34], [43]. Since $s = \alpha n$, the K-SVD cost scales as $O(Nn^3)$. The cost per iteration of the analysis K-SVD algorithm [12], [15] also scales similarly. Our transform-based algorithm thus provides a reduction of the computational cost relative to synthesis/analysis K-SVD in the order, by factor n. Computation times shown in the next section confirm the significant speed-ups of transform learning over K-SVD.

IV. Application to Signal Denoising

We introduce a novel problem formulation for signal denoising using adaptive sparsifying transforms. In this application, we are given N data signals/vectors arranged as columns of matrix $Y \in \mathbb{R}^{n \times N}$ that are corrupted by noise, i.e., $Y = Y^* + H$, where Y^* are the original noiseless data and H is the corrupting noise (a matrix). The goal is to estimate Y^* from Y. We propose the following formulation for denoising signals using learnt sparsifying transforms

$$(P4) \quad \min_{W,X,\hat{Y}} \left\| W\hat{Y} - X \right\|_F^2 + \lambda Q(W) + \tau \left\| Y - \hat{Y} \right\|_F^2 \quad (21)$$
$$s.t. \quad \|X_i\|_0 \leq s \ \forall \ i$$

where the functional Q was defined in Section II (with $c = \mu/\lambda$), and represents the portion of the cost depending only on W. The formulation assumes that the noisy Y can be approximated by $\hat{Y}$ that is approximately sparsifiable by a learnt sparsifying transform W (i.e., $W\hat{Y} \approx X$, with X being column-sparse). This assumption for noisy signals can also be viewed as a generalization of the transform model to allow for explicit denoising. The assumptions $Y \approx \hat{Y}$, $W\hat{Y} \approx X$, with X being column-sparse, can then be regarded as a *"noisy signal" version* of the transform model [3].

The parameter τ in Problem (P4) is typically inversely proportional to the noise level σ [34]. When $\sigma = 0$, the optimal $\hat{Y} = Y = Y^*$, and Problem (P4) reduces to Problem (P3). Note that while (P4) is aimed at explicit denoising, Problem (P3) is aimed at signal representation, where explicit denoising may not be required.

Problem formulation (P4) for simultaneously solving W, X, and $\hat{Y}$ is non-convex even when the l_0 quasi norm is relaxed to an l_1 norm. We propose a simple and fast alternating algorithm to solve Problem (P4). In one step of this alternating algorithm, $\hat{Y}$ and W are fixed (in (P4)), and X is obtained by thresholding $W\hat{Y}$ (same as the sparse coding step in the algorithm for (P3)). In the second step, $\hat{Y}$ and X are fixed (in (P4)), and W is updated using the conjugate gradient method (same as the Transform Update step in the algorithm for (P3)). In the third step, W and X are fixed (in (P4)), and $\hat{Y}$ is updated by solving the following simple least squares problem.

$$\min_{\hat{Y}} \left\| W\hat{Y} - X \right\|_F^2 + \tau \left\| Y - \hat{Y} \right\|_F^2 \quad (22)$$

The promise of the proposed denoising framework will be demonstrated in the next section.

V. Numerical Experiments

We demonstrate the promise of our adaptive formulations in terms of their ability to provide low sparsification errors, good denoising, etc. While the algorithms can be initialized with various useful/relevant transforms, such initializations must have positive determinant (or else be pre-multiplied with a sign matrix to ensure positive determinant). We work with the l_0 quasi norm for sparsity of X in the experiments, but this can be easily substituted by an l_1 norm penalty. We use a fixed step size in the transform update step of our algorithms here. Backtracking line search (employing the Armijo search rule) gives similar performance, but is slower. All implementations were coded in Matlab v7.8 (R2009a). Our algorithm is compared to the synthesis K-SVD [6] using the implementation available from Michael Elad's website [56]. Computations were performed with an Intel Core i5 CPU at 2.27GHz and 4GB memory, employing a 64-bit Windows 7 operating system.

We first demonstrate the ability of our formulation (P3) to learn a sparsifying transform. We consider both synthetic and real data in our experiments. Synthetic data are generated (without noise) as sparse linear combinations of the atoms of a square random (full rank) synthesis dictionary D [6]. Such data are also exactly sparsifiable by D^{-1}. Real data are generated as non-overlapping patches of natural images. Since

[3] We do not explore the usefulness of Problem (P4) with fixed W in this work, focusing instead on adaptivity.

real data are not exactly sparsifiable, the transform model is well-suited to such data. For this case, we compare the sparsification errors of transforms learnt using our algorithms with the corresponding errors of analytical transforms. The data, both synthetic and real, will be used to demonstrate the properties of our algorithm such as convergence, well-conditioned final transform, insensitivity to initialization, etc. Finally, we illustrate the promise of our signal denoising formulation and algorithm through simple experiments.

The quality of the learnt transforms in our experiments will be judged based on their condition number and sparsification error. We will argue and illustrate in this section that the performance of the learnt transform in applications depends on the trade-off between the sparsification error and condition number.

We also define the 'normalized sparsification error' as $\|WY - X\|_F^2 / \|WY\|_F^2$. This measures the fraction of energy lost in sparse fitting in the transform domain. In other words, it indicates the degree of energy compaction achieved in the transform domain, or how well the transform model holds for the signals. This is an interesting property to observe for the adaptive transforms.

For images, another useful 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. This measures the error in recovering the patches Y (or equivalently, the image in the case of non-overlapping patches) as $W^{-1}X$ from their sparse codes X obtained by thresholding WY. While we consider here the option of image recovery from the sparse code, the sparse representations are in general, used differently in various applications such as denoising.

A. Sparsification of Synthetic Data

*1) **Case 1 - Generating Transform Not Unit-Conditioned.**:* We generate a synthetic 20×20 synthesis dictionary with zero mean and unit variance i.i.d. gaussian entries. The data matrix Y has 200 training signals each of which is generated as linear combination of a random set of $s = 4$ atoms of the synthetic dictionary. The coefficients in the linear combination are chosen to be zero mean and unit variance i.i.d. gaussian random variables. Problem (P3) is solved with parameters $\lambda = 50$, $\mu = 10^{-4}$, to learn a transform W that is adapted to the data. The inital transform is the identity matrix. In each transform update step, the conjugate gradient algorithm was run for 60 iterations with a fixed step size of 10^{-4} (although we use more CG iterations here, even 30 or fewer iterations suffice typically).

Figure 2 shows the progress of the algorithm over iterations. The objective function of (P3) (Figure 2(a)) converges monotonically over the iterations (cannot decrease below the bound λQ_0 derived in Section II which is -5.7146×10^3 for this case). The sparsification error and normalized sparsification error (Figure 2(b)) also decrease/converge quickly. The normalized error decreases below 0.01 (or 1%) by the 23^{rd} iteration and below 0.001 by the 44^{th} iteration. Thus, a small number of algorithm iterations seem to suffice to reach a good sparsification error.

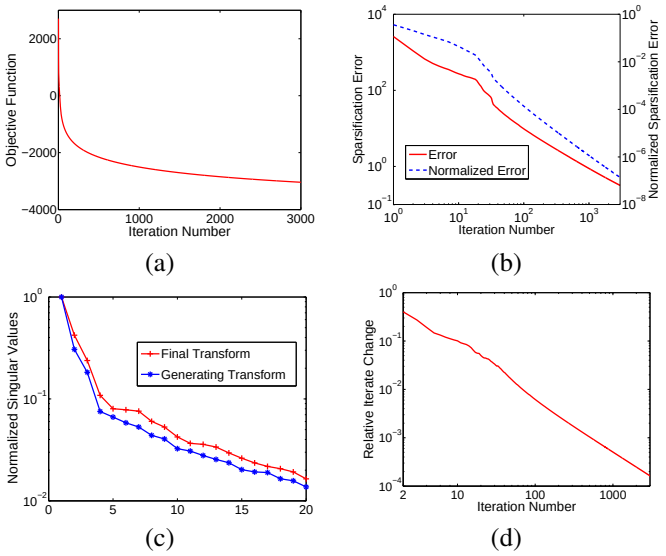


Fig. 2. Algorithm with synthetic data: (a) Objective function vs. iterations, (b) Sparsification error and normalized sparsification error vs. iterations, (c) Normalized singular values of final and generating transforms, (d) Relative iterate change $\|W_i - W_{i-1}\|_F / \|W_{i-1}\|_F$ vs. iterations beginning with iteration 2.

Figure 2(c) shows the normalized singular values (normalized by the largest singular value) of the final learnt transform, and generating transform (i.e., inverse of the synthetic dictionary used to generate the data) on a log-scale. The condition number of the learnt transform is 60.8, while that of the generating transform is 73. This indicates the ability of our algorithm to converge to well-conditioned (not just full rank) transforms. Moreover, the algorithm provides better conditioning compared to even the generating transform/dictionary. In Figure 2(d), we plot the relative change between successive iterates/transforms defined as $\|W_i - W_{i-1}\|_F / \|W_{i-1}\|_F$, where i denotes the iteration number. This quantity decreases to a low value of 10^{-4} over the iterations, indicating convergence of the iterates.

Since the synthetic data are exactly sparsifiable (i.e., they satisfy the transform model $WY = X + E$ with $E = 0$, or equivalently, they satisfy the analysis model since WY is column-sparse), the results of Figure 2 indicate that when the analysis model itself holds, our transform learning algorithm for (P3) can easily and cheaply find it.

*2) **Case 2 - Unit-Conditioned Generating Transform.**:* We now consider the case when the generating dictionary has equal singular values and evaluate the performance of the algorithm for this case. The goal is to investigate whether the algorithm can achieve the lower bounds derived in Section II.

For this experiment, a synthetic 20×20 dictionary with i.i.d. gaussian entries is generated and its singular value decomposition (SVD) of form $U\Sigma V^H$ is computed. An orthonormal dictionary is then generated as UV^H. The data Y are generated using this dictionary similarly to the experiment of Figure 2. The transform learning algorithm parameters are set as $\lambda = 0.5$, $\mu = 0.25$. The inital transform is another random matrix with i.i.d. gaussian entries. All other algorithm parameters are set similarly to the experiment of Figure 2.

The objective function, sparsification error, and condition number (of W) are plotted over iterations for our algorithm

(for (P3)) in Figures 3(a) and 3(b), repectively. In this case, the objective function converges to a value of 5 which is the lower bound (λQ_0) predicted in Section II. The sparsification error and condition number converge to 0 and 1 respectively. The normalized sparsification error (not shown here) drops below 0.01 (or 1%) by the 14^{th} iteration. Thus, the learnt transform provides zero sparsification error, and has equal singular values as predicted in Section II (when the lower bound of the objective is achieved). Moreover, since $\sqrt{\frac{\lambda}{2\mu}} = 1$ (by choice of parameters), the singular values of the learnt transform turn out to be all equal to 1 as predicted in Section II. Thus, when a transform that satisfies the lower bound of the objective function exists, the algorithm converges to it.

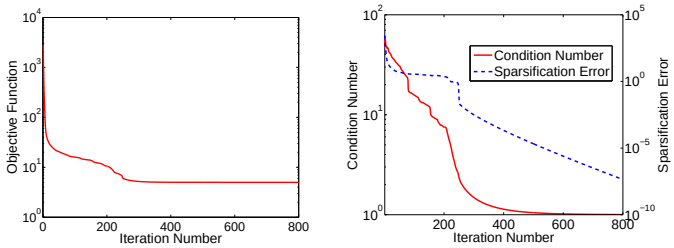
Note that when the data are generated using a synthetic dictionary with condition number $\kappa_{gen} > 1$, the algorithm tends to produce a transform $\hat{W}$ with condition number $\kappa(\hat{W}) < \kappa_{gen}$ lower than that of the generating dictionary. A condition number $\kappa(\hat{W}) = 1$ may not be achieved, since a transform that gives both low sparsification error and a condition number of 1 may not exist in this case. This was also observed in the experiment of Figure 2.

B. Sparsification of Real Data

*1) **Insensitivity to Initialization.**:* Here, we extract the 8×8 ($n = 64$) non-overlapping patches from the image Barbara [6]. The data matrix Y in this case has 4096 training signals (patches represented as vectors) and we work with $s = 11$. The means (or DC values) of the patches are removed and we only sparsify the mean-subtracted patches (mean removal is typically adopted in image processing applications such as K-SVD based image denoising). The means can be added back for display purposes. Problem (P3) is solved to learn a square transform W that is adapted to this data. The algorithm parameters are $\lambda = \mu = 4 \times 10^5$. The conjugate gradient algorithm was run for 128 iterations in each transform update step, with a fixed step size of 10^{-8}. (The performance is similar even with 20 or less conjugate gradient iterations.)

We consider four different initializations (initial transforms) for the algorithm. The first is the 64×64 2D DCT matrix (defined as $W_0 \otimes W_0$, where W_0 is the 8×8 1D DCT matrix, and "$\otimes$" denotes the Kronecker product). The second initialization is obtained by inverting/transposing the left singular matrix of Y. This initialization is also popularly known as the Karhunen-Loève Transform (KLT). The third and fourth initializations are the identity matrix, and a random matrix with i.i.d. gaussian entries (zero mean and standard deviation 0.2), respectively. Note that any initial matrix with a negative determinant can be made to have a positive determinant by switching the signs of the entries on one row, or by exchanging two rows.

Figure 4(a) shows the objective function of Problem (P3) over the iterations of the algorithm for the different initializations. The objective function converges monotonically, and although different initializations lead to different initial rates of convergence, the objective functions have nearly identical final values in all cases. This indicates that our alternating algorithm is robust to initialization. The sparsification error (Figures 4(b) and 4(d)) too converges quickly and to similar values for all



(a) (b)

Fig. 3. Algorithm attains lower bound: (a) Objective function vs. iterations, (b) Condition number and sparsification error vs. iterations.

initializations. The horizontal lines in Figures 4(b) and 4(d) denote the sparsification errors of the 2D DCT, KLT, identity, and random gaussian transforms (i.e., the sparsification error at iteration zero). Our algorithm reduces the sparsification error by 5.98 dB, 7.14 dB, 14.77 dB, and 18.72 dB, respectively, from the values for the initial transforms. The normalized sparsification error for the learnt transform W with the 2D DCT initialization is 0.0437. The values corresponding to the other initializations are only slightly different.

The recovery PSNR for the learnt W is 34.59 dB with the 2D DCT initialization. The corresponding values for the other initializations differ only by hundredths of a dB. (These small gaps could be reduced further with better choice of step size and the number of conjugate gradient iterations.)

Finally, the condition number (Figure 4(c)) with the random gaussian initialization, also converges quickly to a low value of 1.46. This illustrates that image patches are well sparsified by well-conditioned transforms.

To study further the effect of different initializations, Figure 4(e) compares the singular values of the transforms learnt with 2D DCT and with random gaussian initializations. The singular values for the two cases are almost identical, with condition numbers of 1.40 and 1.46, respectively. For direct comparison, the transforms learnt with the two different initializations (DCT and random gaussian) are compared in Figures 4(f) and 4(g), respectively. Figures 4(h) and 4(i) provide a different view of the learnt transforms, with each row of W displayed as an 8×8 patch, which we call the 'transform atom'. While the two learnt transforms appear different, they are essentially equivalent in the sense that they produce very similar sparsification errors, and have almost identical singular values (and thus, almost identical condition numbers). In both cases, the atoms exhibit geometric and frequency like structures. Apparently, the transform learning algorithm is able to discover the structure of image patches, and provide dramatically lower sparsification errors than analytical transforms.

Based on the results of Figure 4, we conjecture that in general, Problem (P3) may admit multiple global minima that are not related by only row permutations and sign changes. Which of these essentially equivalent solutions is actually achieved by the algorithm depends on the initialization. The existence of alternative but essentially equivalent solutions to (P3) also suggests that additional application-specific performance criteria may be used to select between equivalent transforms, or the problem formulation may be modified to incorporate additional preferences.

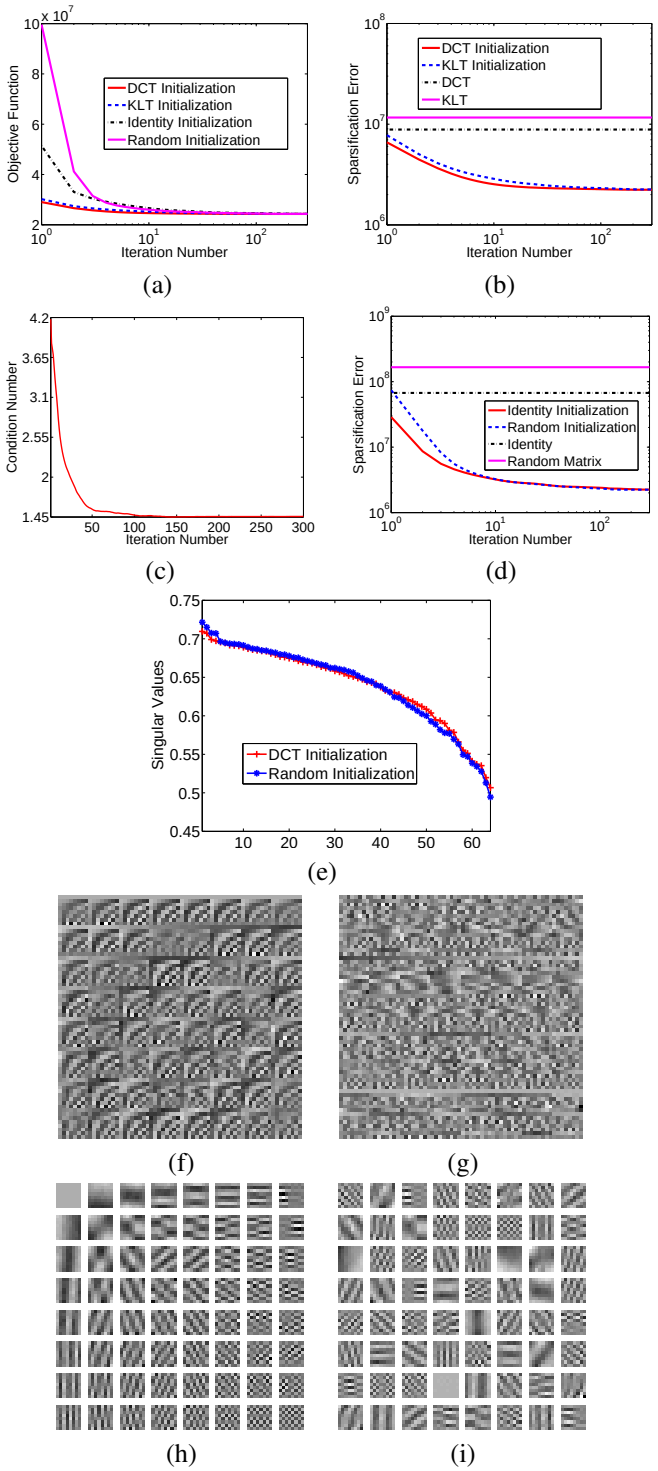


Fig. 4. Real data - Effect of different Initializations: (a) Objective function vs. iterations, (b) Sparsification error vs. iterations for DCT and KLT initializations, along with the sparsification errors (horizontal lines) of the DCT and KLT transforms, (c) Condition number vs. iterations for random gaussian initialization, (d) Sparsification error vs. iterations for identity and random gaussian initializations, along with the sparsification errors (horizontal lines) of the identity and random gaussian matrices themselves, (e) Singular values of the transforms learnt with DCT and random gaussian initializations, (f) Transform learnt with DCT initialization, (g) Transform learnt with random gaussian initialization, (h) Rows of learnt transform shown as patches for DCT initialization, (i) Rows of learnt transform shown as patches for the case of random gaussian initialization.

*2) **Performance for Various Images vs. DCT.**:* Next, we study the behavior of our algorithm on different images, by solving (P3) to learn a transform for each of four different 512×512 images. The transforms are directly adapted to the non-overlapping patches of the images. All algorithm parameters are the same as for the experiment of Figure 4 (and we use the DCT initialization).

	CN-L	NSE-L	RP-L	NSE-D	RP-D
Barbara	1.40	0.0437	34.59	0.0676	32.85
Lena	1.16	0.0376	37.64	0.0474	36.91
Peppers	1.17	0.0343	36.97	0.0448	36.15
Cameraman	1.13	0.0088	42.50	0.0191	39.43

TABLE I
NORMALIZED SPARSIFICATION ERRORS (NSE-L) AND RECOVERY PSNRS (RP-L) FOR THE LEARNT TRANSFORMS, ALONG WITH THE CORRESPONDING VALUES FOR THE 2D DCT (NSE-D/RP-D), AND THE CONDITION NUMBERS OF THE LEARNT TRANSFORMS (CN-L).

Table I lists the normalized sparsification errors and recovery PSNRs for the learnt transforms and the patch-based 2D DCT (at $s = 11$), along with the condition numbers of the learnt transforms. The learnt transforms are seen to be well-conditioned for all the images. The corresponding normalized sparsification errors are small and, moreover, better than those of the 2D DCT by upto 3.4 dB for the tested images. The learnt transforms also provide up to 3.1 dB better recovery PSNRs than the 2D DCT. All these results indicate the promise of the adaptive transform model for natural signals.

Note that while we considered adapting the transform to specific images, a transform adapted to a data-set of training images also gives promising improvements over fixed transforms such as the DCT and Wavelets on test images – a property that can be exploited for image compression.

C. Performance as Function of Parameters

Next, we work with the same data as for Figure 4, and test the performance of our algorithm as a function of the parameter λ at different data sparsity levels $s = 7, 11, 15$ (with $\mu = \lambda$, and all other parameters fixed as in the experiment of Figure 4). The algorithm is initialized with the 2D DCT for the experiments.

For fixed s, the condition number of the learnt transform (Figure 5(b)) decreases as a function of λ. For high values ($> 10^6$) of λ (and $\mu = \lambda$), the algorithm favors a condition number of 1. For low values of λ, the condition number is quite high (condition number of 70 at $\lambda = 10^3$ when $s = 11$). This behavior of the condition number for fixed $\frac{\mu}{\lambda}$, was predicted in Section II.

On the other hand, the normalized sparsification error (Figure 5(a)) increases with λ for fixed s. The error is lowest for smaller λ. This is because smaller λ values give higher preference to the sparsification error term in the cost (11) of Problem (P3). Moreover, even at high values of λ, the normalized sparsification error tends to be reasonable. (For example, it is 0.0457 at $\lambda = 10^6$ when $s = 11$.) Very high λ values tend to increase the normalized sparsification error only slightly. The plots indicate that we can get reasonable normalized sparsification errors simultaneously with good condition numbers.

Figure 5(c) plots the recovery PSNRs with the learnt transforms for Barbara. For fixed s, the recovery PSNR is best at λ values corresponding to intermediate conditioning

or 'well-conditioning'. (For example when $s = 11$, the best recovery PSNR is 34.65 dB at $\lambda = 10^5$, or at $\kappa = 2.29$.) At unit conditioning, or bad conditioning, the recovery PSNR is lower. This indicates that natural images tend to prefer well-conditioned transforms, as far as recovery PSNR is concerned.

All the metrics, however, degrade when the data sparsity level s is reduced (for fixed λ, μ). This behavior is expected since at lower s values, we have fewer degrees of freedom to represent the data (i.e., the learning is more constrained at lower data sparsity levels). For a particular choice of condition number (from Figure 5 (b)), we get a lower normalized sparsification error at a larger value of s, and vice-versa.

As the data sparsity level $s \nearrow n$ (where n is the number of pixels in a patch), we can expect all the metrics to improve, because the problem becomes less and less constrained. In the limit when $s = n$, we can have $WY = X$ exactly and thus, infinite recovery PSNR with orthonormal transforms such as the DCT, or even with the trivial identity matrix/transform. Thus, when $s = n$, Problem (P3) attains the lower bound (with any W satisfying Corollary 1) of its cost function.

The normalized sparsification errors for the (patch-based) 2D DCT at $s = 7, 11, 15$ are 0.1262, 0.0676, and 0.0393, respectively. The corresponding values for the recovery PSNR are 30.14 dB, 32.85 dB, and 35.21 dB, respectively. At reasonable condition numbers, the learnt transforms at $s = 7, 11, 15$ perform much better than the DCT at those sparsity levels.

Thus, it is evident from the results of Figure 5 that the parameters can provide a trade-off between the various metrics such as sparsification error and condition number. The choice of parameters would also depend on the specific application (i.e., on how sparsification error and condition number affect the performance in the specific application). For example, when the goal is recovery from sparse code, we see that since the transform model suggests $WY = X + E$, where E is the sparsification error term in the transform domain, we have $Y = W^{-1}X + W^{-1}E$, when W is non-singular. The recovery PSNR depends on the quantity $W^{-1}E$, which in turn depends on both the sparsification error and conditioning of W. Thus, the trade-off between sparsification error and condition number can be expected to determine the best recovery PSNR. Therefore, although the formulation of Problem (P3) does not directly optimize for the recovery PSNR, it attempts to do so indirectly by allowing control over the sparsification error and condition number, which in turn serve as good surrogates for the recovery PSNR.

An interesting point to note is that since our algorithm can learn well-conditioned transforms, the quantities $\|WY - X\|_F^2$ and $\left\|Y - W^{-1}X\right\|_F^2$ are both well-behaved. In contrast, K-SVD [6] typically provides very poorly conditioned dictionaries D. The term $\left\|D^{-1}Y - X\right\|_F^2$ is typically too high for K-SVD (even much worse than the sparisfication errors of analytical transforms). Thus, the inverse of the K-SVD dictionary hardly qualifies as a sparsifying transform.

Note that setting $\frac{\mu}{\lambda} = 1$ (with λ chosen appropriately) worked well for most experiments except for synthetic data cases with poor generating conditioning for which the parameters were set manually.

Next, in order to illustrate the importance of the Frobenius-norm regularization in our formulation (P3), we repeat the experiment of Figure 4 (with DCT initialization), but with $\mu = 0$ (or, in other words, we solve Problem (P2)). In this case, the constant atom (row of all 1's) is an exact sparsifier (or, orthogonal) for the zero-mean image patches. Thus, a transform (of positive determinant) that has a constant row scaled by $\alpha \to \infty$, and some other linearly independent rows scaled by $\hat{\alpha} \to 0$, would be a good sparsifier as per Problem (P2). The Frobenius-norm term in the cost of (P3) is necessary to overcome such scaling ambiguities, and for encouraging well-conditioning. Hence, when our algorithm is executed for many iterations (5000 iterations) with $\mu = 0$, we obtain a badly conditioned transform, whose condition number is $\kappa = 1037$. The learnt transform also has a high Frobenius norm (of 460) which is mostly concentrated on the constant atom that is learnt by our algorithm. (Note that a few other rows of the learnt W also have high norms, but the constant row has a much higher norm.) The normalized sparsification error and recovery PSNR for this case (after 5000 iterations) are 0.003 and 32.59 dB, respectively. The low value for the normalized sparsification error is expected, since the lack of the Frobenius-norm regularization term in (P3) favors better sparsification. However, the recovery PSNR is much worse (by 2 dB) than that obtained in the experiment of Figure 4, due to the poor conditioning of the learnt W [4]. Thus, the Frobenius-norm regularization is crucial for removing various ambiguities and for the success of sparsifying transform learning in applications.

Note that while the experiment for the $\mu = 0$ case was

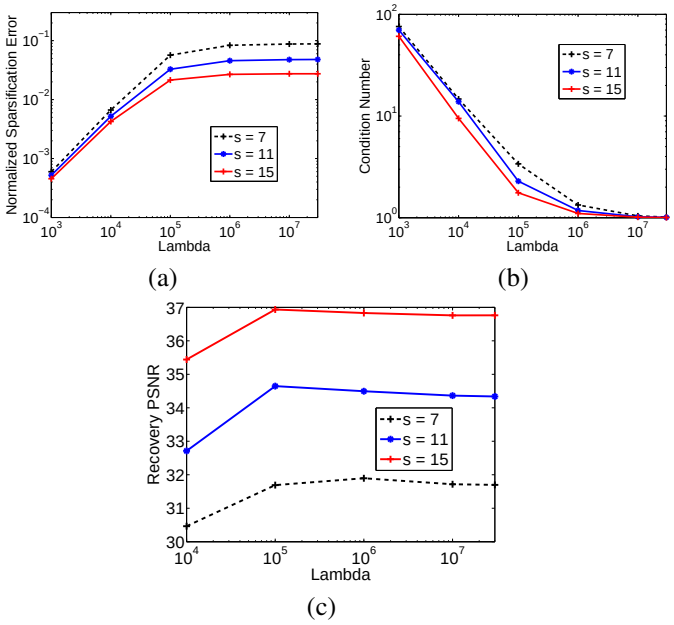


Fig. 5. Parameter Selection: (a) Normalized sparsification error vs. λ for different values of s with $\mu = \lambda$, (b) Condition number vs. λ for different s values with $\mu = \lambda$, (c) Recovery PSNR vs. λ for different values of s with $\mu = \lambda$.

[4]Note that, although the condition number obtained with $\mu = 0$ (after 5000 iterations) is substantially higher than that obtained with $\mu = \lambda$, the drop in the recovery PSNR is not as significant. This is because, the high norm of the constant row has no effect on the sparsification error term $E = WY - X$. Hence, the quantity $W^{-1}E$ (which determines the recovery PSNR), although degraded by the poor conditioning of W, does not degrade as significantly as the condition number, due to the higher norm of W.

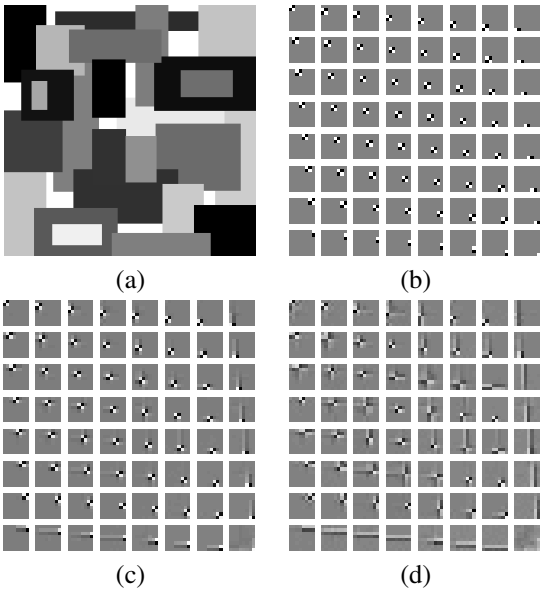


Fig. 6. Piecewise-constant Case: (a) The Piecewise-constant image, (b) 2D Finite difference transform with $\kappa = 113$, (c) Learnt transform with $\kappa = 15.35$, (d) Learnt transform with $\kappa = 5.77$.

performed with zero-mean patches, we also observed inferior performance with $\mu = 0$ (compared to $\mu = \lambda$), when the means of the patches were retained.

D. Performance for Piecewise-Constant Images

Next, we consider learning a transform for a 512×512 piecewise-constant image (Figure 6(a)). Piecewise-constant images are well-sparsified by the finite difference transform. We work with 8×8 non-overlapping image patches in this experiment. The two-dimensional finite difference transform shown in Figure 6(b) is obtained as a Kronecker product of two one-dimensional finite difference matrices. (Each made square and non-singular by appending a row that has all 0's and a 1 on the last entry.) Note that this finite difference transform is square rather than overcomplete. It is an exact sparsifier for the patches (with means removed) of the image in Figure 6(a) for sparsity levels of $s \geq 5$. However, this transform is poorly conditioned with $\kappa = 113$. We investigate the behavior of our algorithm for this interesting example. We solve Problem (P3) to learn transforms at various values of the parameter λ (with $\mu = \lambda$), with $s = 5$ and all other parameters fixed as in the experiment of Figure 4. We initialize the algorithm with the 2D finite difference transform itself, to check whether the algorithm converges to the same structure.

The learnt transforms at $\lambda = 4 \times 10^2$ (Figure 6(c)) and $\lambda = 8 \times 10^3$ (Figure 6(d)) are well-conditioned with condition numbers 15.35 and 5.77, respectively. Both these transforms provide almost zero normalized sparsification errors for the data (normalized sparsification errors of 1.6×10^{-5} and 5×10^{-4} when κ is 15.35 and 5.77, respectively). Thus, our transform learning algorithm is able to learn well-conditioned transforms that sparsify almost as well as the poorly conditioned finite difference transform. Such well-conditoned adaptive transforms also perform better than poorly conditioned ones in applications such as image denoising [57]. Note that the learnt transforms do appear somewhat different from the finite difference transform, since they are adapted to the specific data.

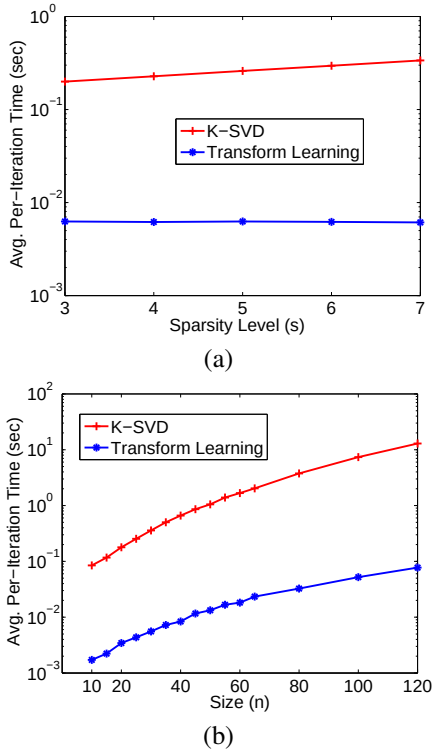


Fig. 7. Algorithm Execution Time: (a) Average per-iteration execution time vs. sparsity level (s) for the synthesis K-SVD and our transform learning scheme, (b) Average per-iteration execution time vs. dictionary/transform size (n) for both the synthesis K-SVD and our transform learning scheme.

When the transform learning algorithm was executed for the same data, but with lower sparsity levels $s < 5$, the learnt transforms apart from being well-conditioned, also provided significantly better sparsification at the same sparsity level, than the finite difference transform. The results here, much like previous ones, indicate the promise of our algorithm for (P3) to adapt to data and obtain significantly better sparse representations than analytical transforms.

E. Run Time

Next, we test the execution times of the iterations of our algorithm (for Problem (P3)) and compare it with the execution times of the iterations of the popular synthesis dictionary learning algorithm, K-SVD [6], [56]. The goal here is to demonstrate the promising speed-ups of transform learning, which can prove advantageous in applications.

First, we study the behavior of run times as a function of the sparsity level s. At each sparsity level, we generate synthetic data similarly to the data of Figure 2, and execute the algorithm of Problem (P3). The algorithm parameters are set similarly to the experiment of Figure 2 (note that $\lambda = 50$ for most of the experiments, and it is optimally set for the others). We also execute the synthesis K-SVD algorithm [6] with square dictionaries for the same data. The algorithms are executed for a fixed number of (700) iterations, and moreover the simulation is repeated five times (with fresh data) at each sparsity level.

Figure 7(a) plots the average per-iteration execution times

of our algorithm and of the synthesis K-SVD as a function of the sparsity level. It can be seen that the iterations of our algorithm are significantly faster (by at least a factor of 32) than K-SVD. The per-iteration execution time of K-SVD increases quickly with sparsity level. In contrast, the per-iteration execution times of our algorithm are approximately the same at the various sparsity levels. The iterations of our algorithm are thus, nearly 50-60 times faster than the synthesis K-SVD iterations at higher sparsity levels.

We also tested the speed of our algorithm as a function of transform size. The data for this experiment were generated synthetically at varying problem sizes (n). The number of training signals $N = 10n$, and the sparsity at which data are generated is $s = n/5$ (rounded to nearest integer). Both the synthesis K-SVD and our algorithm (for Problem (P3)) are executed for 100 iterations at the different problem sizes (both transform and dictionary are square). The conjugate gradient method within our algorithm is executed for 30 iterations, and the parameters such as μ, λ are set appropriately.

Figure 7 (b) plots the average per-iteration execution times of the algorithms as a function of problem size. The per-iteration execution times for our algorithm are at least about 50 times better than those for K-SVD. Moreover, the gap widens as the problem size increases. At $n = 120$, the iterations of the transform learning algorithm are 167 times faster than those of K-SVD. This was also predicted by the expressions for computational cost in Section III. The K-SVD iteration time also increases at a much faster rate with problem size compared to our algorithm. The results indicate that the proposed transform learning can be easily implemented at large problem sizes.

Since our algorithm converges very quickly, we can say that sparsifying transforms can in general, also be learnt (using Problem (P3)) much faster than synthesis dictionaries (using K-SVD). Note that while we compared to the case of a square synthesis dictionary here, the speed-ups are much greater when compared to the overcomplete K-SVD.

We expect the run times of our algorithm to decrease substantially with conversion of the code to C/C++, and code optimization. Efficient implementation of sparse matrix multiplications (in the transform update step (20)) and efficient thresholding (i.e., thresholding training vectors in parallel and without resorting to full sorting in the sparse coding step) are just some of the ways to reduce run times.

F. Preliminary Denoising Experiments

Here, we test the effectiveness of our denoising algorithm that solves Problem (P4). For comparison, we also consider a denoising formulation involving synthesis dictionary learning as follows [34]

$$(P5) \quad \min_{D,X,\hat{Y}} \left\| \hat{Y} - DX \right\|_F^2 + \rho \left\| Y - \hat{Y} \right\|_F^2 \qquad (23)$$
$$s.t. \quad \|X_i\|_0 \leq s \ \ \forall \ i$$

The parameter ρ is chosen similarly (i.e., inversely proportional to σ) to τ of Problem (P4). The synthesis denoising formulation (P5) is also solved by alternating minimization [34]. In one step, D and X are learnt via K-SVD with fixed $\hat{Y}$ while in the other step, $\hat{Y}$ is updated by a least squares procedure.

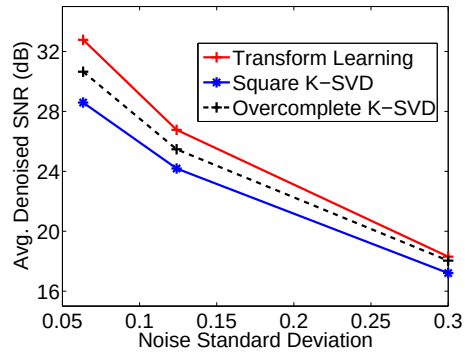


Fig. 8. Signal denoising: Average denoised SNR (in dB) vs. noise standard deviation (σ) for our algorithm, and for K-SVD with square and overcomplete dictionaries.

The data for the comparison of problems (P4) and (P5) were generated synthetically using a random 20×20 synthesis dictionary with a sparsity level of 5 ($s = 5$). Note that the synthetic data obey both the synthesis dictionary model and the transform model exactly. The generated synthetic data were further corrupted by additive i.i.d. gaussian noise. The number of noisy signals is $N = 10n$. The parameters τ and ρ for (P4) and (P5) were both optimally chosen (empirically) as $\frac{0.5}{\sigma}$. The parameter $\lambda = 20$ for our transform-based algorithm, and all other parameters for our algorithm (such as μ, etc.) were the same as for Figure 2.

Problems (P4) and (P5) were solved at various noise levels with fresh data generated at each noise level. In the case of Problem (P5), we learnt both a square 20×20 dictionary, and an overcomplete 20×80 dictionary. At each noise level, the denoised signal-to-noise ratio (SNR), i.e., $20 \log_{10} \left(\|\hat{Y}\|_F / \|Y^* - \hat{Y}\|_F \right)$ is computed for both algorithms. Here, Y^* denotes the original noiseless data to which noise was added, and $\hat{Y}$ is the output of the denoising algorithm (using either (P4) or (P5)). The denoised SNR is averaged over five trials at each noise level.

Figure 8 plots the average denoised SNR (in decibels) as a function of the noise standard deviation (σ). At a low noise level (or noise standard deviation) of 0.063, our algorithm provides nearly 4.2 dB better denoised SNR compared to the square K-SVD, and 2.1 dB better denoised SNR compared to the four fold overcomplete dictionary. (As expected, the overcomplete K-SVD performs better than the square K-SVD.) The improvement drops at higher noise levels as both algorithms degrade in performance. However, even at high noise levels ($\sigma = 0.3$), our algorithm still provides 1.1 dB of improvement over the square K-SVD, and 0.3 dB improvement over the overcomplete K-SVD, respectively. At mid-noise levels such as $\sigma = 0.124$, we obtain an improvement of 2.6 dB over the square K-SVD, and 1.3 dB over the overcomplete K-SVD, respectively. These results indicate that learnt sparsifying transforms can provide promising denoising.

We have also applied sparsifying transforms to image denoising [57], [58] showing better denoising compared to both adaptive overcomplete synthesis and analysis dictionaries.

VI. Conclusions and Future Work

In this paper, a novel problem formulation for learning sparsifying transforms was presented. The alternating algorithm for square transform learning involves two steps - thresholding and gradient descent. Our proposed framework gives rise to well-conditioned transforms with much lower sparsification errors than analytical transforms. Results with natural images demonstrate that well-conditioning (but not necessarily unit conditioning) of the transforms is compatible with good sparsification and good performance in applications. Even for piecewise constant images, for which a difference operator provides optimal sparsification, but at high condition number, our well-conditioned learnt transforms provide essentially identical, or even better sparsification. Our algorithm was shown to provide monotonic convergence of the cost function, and is insensitive to initialization. Moreover, the computational cost of our transform learning is nearly two orders of magnitude lower than that of synthesis dictionary learning algorithms such as K-SVD. We also introduced a signal denoising formulation involving sparsifying transform learning, and demonstrated promising performance for our proposed algorithm. The usefulness of transform learning in signal and image processing applications merits further study.

Appendix A
Proof of Lemma 1

We now derive a lower bound for $Q = -\log\det W + c\|W\|_F^2$ that depends on the condition number κ of W. We work with the case $\det W > 0$. Denoting the singular values of W by $\beta_i, 1 \leq i \leq n$, we have $\|W\|_F^2 = \sum_{i=1}^n \beta_i^2$ and $-\log\det W = -\log(\prod_{i=1}^n \beta_i) = -\sum_{i=1}^n \log\beta_i$. Therefore,

$$Q = \sum_{i=1}^{n} \left(-\log\beta_i + c\beta_i^2\right) \tag{24}$$

We now bound the terms on the right hand side of (24). Since, $-\log\beta_i + c\beta_i^2$ is a strictly convex function of β_i for each i, we lower bound it using its minimum value (achieved when $\beta_i = \sqrt{\frac{1}{2c}}$) as follows.

$$-\log\beta_i + c\beta_i^2 \geq \frac{1}{2} + \frac{1}{2}\log(2c) \tag{25}$$

We substitute the above bound in equation (24) only for the indices $2 \leq i \leq n-1$ to get

$$Q \geq \frac{n-2}{2} + \frac{n-2}{2}\log(2c) - \log(\beta_1\beta_n) + c(\beta_1^2 + \beta_n^2) \tag{26}$$

We now express β_1 in terms of β_n in equation (26).

$$\beta_1 = \kappa\beta_n \tag{27}$$

Next, since, $-\log(\kappa\beta_n^2) + c\beta_n^2(1+\kappa^2)$ is a strictly convex function, we get the following lower bound using the minimum value (achieved when $\beta_n = \sqrt{\frac{1}{c(1+\kappa^2)}}$) of the function.

$$-\log(\kappa\beta_n^2) + c\beta_n^2(1+\kappa^2) \geq 1 - \log\frac{\kappa}{c(1+\kappa^2)} \tag{28}$$

Upon substitution of the preceding bound in equation (26) and simplification, we get the following lower bound on Q.

$$Q \geq \frac{n}{2} + \frac{n}{2}\log(2c) - \log\frac{2\kappa}{1+\kappa^2} \qquad \blacksquare$$

which completes the proof.

Appendix B
Proof of Corollary 1

The inequality $Q \geq Q_0$ follows from Lemma 1, because $-\log\frac{2\kappa}{1+\kappa^2}$ has a minimum value of zero (achieved for $\kappa = 1$). For the result to hold with equality, we require that $\kappa = 1$. We also require (26) and (28) in the proof of Lemma 1 to hold with equality, which can happen if and only if

$$\beta_1 = \sqrt{\frac{\kappa^2}{c(1+\kappa^2)}}\,, \quad \beta_n = \sqrt{\frac{1}{c(1+\kappa^2)}} \tag{29}$$

$$\beta_i = \sqrt{\frac{1}{2c}}\,, \;\; 2 \leq i \leq n-1$$

Now, using $\kappa = 1$ in (29), we get the required result. $\blacksquare$

References

[1] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[2] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of jpeg-2000," in *Proc. Data Compression Conf.*, 2000, pp. 523–541.

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

[4] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[5] K. Engan, S. Aase, and J. Hakon-Husoy, "Method of optimal directions for frame design," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1999, pp. 2443–2446.

[6] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[7] M. Yaghoobi, T. Blumensath, and M. Davies, "Dictionary learning for sparse approximations with the majorization method," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2178–2191, 2009.

[8] K. Skretting and K. Engan, "Recursive least squares dictionary learning algorithm," *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2121–2130, 2010.

[9] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, 2010.

[10] G. Peyré and J. Fadili, "Learning analysis sparsity priors," in *Proc. of Sampta'11*, 2011. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00542016/

[11] M. Yaghoobi, S. Nam, R. Gribonval, and M. Davies, "Analysis operator learning for overcomplete cosparse representations," in *European Signal Processing Conference (EUSIPCO)*, 2011.

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

[13] B. Ophir, M. Elad, N. Bertin, and M. Plumbley, "Sequential minimal eigenvalues - an approach to analysis dictionary learning," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2011.

[14] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, "Noise aware analysis operator learning for approximately cosparse signals," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5409–5412.

[15] R. Rubinstein, T. Faktor, and M. Elad, "K-SVD dictionary-learning for the analysis sparse model," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5405–5408.

[16] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009.

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

[18] E. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[19] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[20] D. Donoho, "Compressed sensing," *IEEE Trans. Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[21] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, pp. 407–499, 2004.

[22] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution," *Comm. Pure Appl. Math*, vol. 59, pp. 797–829, 2004.

[23] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[24] I. F. Gorodnitsky, J. George, and B. D. Rao, "Neuromagnetic source imaging with FOCUSS: A recursive weighted minimum norm algorithm," *Electrocephalography and Clinical Neurophysiology*, vol. 95, pp. 231–251, 1995.

[25] G. Harikumar and Y. Bresler, "A new algorithm for computing sparse solutions to linear inverse problems," in *ICASSP*, may 1996, pp. 1331–1334.

[26] G. Harikumar, "Blind image deconvolution from multiple blurs, and sparse approximation," Ph.D. dissertation, University of Illinois at Urbana-Champaign, mar 1997, Yoram Bresler, adviser.

[27] Y. Bresler, C. Couvreur, and G. Harikumar, "Fast optimal and suboptimal algorithms for sparse solutions to linear inverse problems," in *Proc. IEEE Int. Conf. Acoust. Speech, Sig. Proc.*, vol. 3, apr 1998, pp. 1877–1880.

[28] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," *Signal Processing Letters, IEEE*, vol. 14, no. 10, pp. 707–710, 2007.

[29] E. J. Candès and D. L. Donoho, "Ridgelets: A key to higher-dimensional intermittency?" *Phil. Trans. R. Soc. Lond. A*, vol. 357, no. 1760, pp. 2495–2509, 1999.

[30] M. N. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2091–2106, 2005.

[31] E. J. Candès and D. L. Donoho, "Curvelets - a surprisingly effective nonadaptive representation for objects with edges," in *Curves and Surfaces*. Vanderbilt University Press, 1999, pp. 105–120.

[32] R. Gribonval and K. Schnass, "Dictionary identification–sparse matrix-factorization via l_1 -minimization," *IEEE Trans. Inform. Theory*, vol. 56, no. 7, pp. 3523–3539, 2010.

[33] 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.

[34] 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.

[35] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. on Image Processing*, vol. 17, no. 1, pp. 53–69, 2008.

[36] M. Protter and M. Elad, "Image sequence denoising via sparse and redundant representations," *IEEE Trans. on Image Processing*, vol. 18, no. 1, pp. 27–36, 2009.

[37] M. Aharon and M. Elad, "Sparse and redundant modeling of image content using an image-signature-dictionary," *SIAM Journal on Imaging Sciences*, vol. 1, no. 3, pp. 228–247, 2008.

[38] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *SIAM Multiscale Modeling and Simulation*, vol. 7, no. 1, pp. 214–241, 2008.

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

[40] G. Yu, G. Sapiro, and S. Mallat, "Image modeling and enhancement via structured sparse model selection," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 1641–1644.

[41] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2010*, 2010, pp. 3501–3508.

[42] H. Y. Liao and G. Sapiro, "Sparse representations for limited data tomography," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2008, pp. 1375–1378.

[43] 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.

[44] ——, "Multiscale dictionary learning for MRI," in *Proc. ISMRM*, 2011, p. 2830.

[45] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, "Constrained overcomplete analysis operator learning for cosparse signal modelling," *IEEE Trans. Signal Process.*, 2012, submitted. [Online]. Available: http://arxiv.org/abs/1205.4133

[46] E. J. Candès and J. Romberg, "Practical signal recovery from random projections," in *SPIE International Symposium on Electronic Imaging: Computational Imaging III*, 2005.

[47] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proceedings of the IEEE*, vol. 57, no. 1, pp. 58–68, 1969.

[48] J. B. Allen and L. R. Rabiner, "A unified approach to short-time fourier analysis and synthesis," *Proc. IEEE*, vol. 65, no. 11, pp. 1558–1564, 1977.

[49] M. Lustig, D. Donoho, and J. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[50] M. Lustig, J. M. Santos, D. L. Donoho, and J. M. Pauly, "k-t SPARSE: High frame rate dynamic MRI exploiting spatio-temporal sparsity," in *Proc. ISMRM*, 2006, p. 2420.

[51] C. Wang, D. Sun, and K.-C. Toh, "Solving log-determinant optimization problems by a newton-CG primal proximal point algorithm," *SIAM J. Optim*, vol. 20, no. 6, pp. 2994–3013, 2010.

[52] M. Zibulevsky and B. A. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary," *Neural Computation*, vol. 13, no. 4, pp. 863–882, 2001.

[53] R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization*. Springer-Verlag, 2009.

[54] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing USA, 2005.

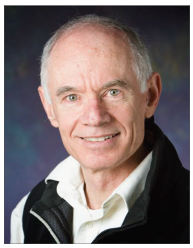
[55] J. A. Fill and S. Janson, "Quicksort asymptotics," *Journal of Algorithms*, vol. 44, no. 1, pp. 4–28, 2002.

[56] M. Elad, "Michael Elad personal page," http://www.cs.technion.ac.il/~elad/Various/KSVD_Matlab_ToolBox.zip, 2009.

[57] S. Ravishankar and Y. Bresler, "Learning doubly sparse transforms for image processing," *IEEE Trans. Image Process.*, 2012, submitted. [Online]. Available: https://netfiles.uiuc.edu/ravisha3/shared/tip_s1.pdf

[58] ——, "Learning sparsifying transforms for image processing," in *IEEE Int. Conf. Image Process.*, 2012, to appear.

Saiprasad Ravishankar received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology Madras, in 2008. He received the M.S. degree in Electrical and Computer Engineering, in 2010, from the University of Illinois at Urbana-Champaign, where he is currently a Ph.D candidate. His current research interests are in signal and image processing, and medical imaging.



Yoram Bresler received the B.Sc. (cum laude) and M.Sc. degrees from the Technion, Israel Institute of Technology, in 1974 and 1981 respectively, and the Ph.D degree from Stanford University, in 1986, all in Electrical Engineering. In 1987 he joined the University of Illinois at Urbana-Champaign, where he is currently a Professor at the Departments of Electrical and Computer Engineering and Bioengineering, and at the Coordinated Science Laboratory. Yoram Bresler is also President and Chief Technology Officer at InstaRecon, Inc., a startup he co-founded to commercialize breakthrough technology for tomographic reconstruction developed in his academic research. His current research interests include multi-dimensional and statistical signal processing and their applications to inverse problems in imaging, and in particular compressed sensing, computed tomography, and magnetic resonance imaging.

Dr. Bresler has served on the editorial board of a number of journals, and on various committees of the IEEE. Currently he serves on the editorial boards for the SIAM Journal on Imaging Science. Dr. Bresler is a fellow of the IEEE and of the AIMBE. He received two Senior Paper Awards from the IEEE Signal Processing society, and a paper he coauthored with one of his students received the Young Author Award from the same society in 2002. He is the recipient of a 1991 NSF Presidential Young Investigator Award, the Technion (Israel Inst. of Technology) Fellowship in 1995, and the Xerox Senior Award for Faculty Research in 1998. He was named a University of Illinois Scholar in 1999, appointed as an Associate at the Center for Advanced Study of the University in 2001-02, and Faculty Fellow at NCSA in 2006.