Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation

Ron Rubinstein, *Student Member, IEEE*, Michael Zibulevsky, *Member, IEEE*, and Michael Elad, *Senior Member, IEEE*

***Abstract*—An efficient and flexible dictionary structure is proposed for sparse and redundant signal representation. The proposed *sparse dictionary* is based on a sparsity model of the dictionary atoms over a base dictionary, and takes the form $\mathbf{D} = \mathbf{\Phi}\mathbf{A}$ where $\mathbf{\Phi}$ is a fixed base dictionary and $\mathbf{A}$ is sparse. The sparse dictionary provides efficient forward and adjoint operators, has a compact representation, and can be effectively trained from given example data. In this, the sparse structure bridges the gap between implicit dictionaries, which have efficient implementations yet lack adaptability, and explicit dictionaries, which are fully adaptable but non-efficient and costly to deploy. In this paper we discuss the advantages of sparse dictionaries, and present an efficient algorithm for training them. We demonstrate the advantages of the proposed structure for 3-D image denoising.**

***Index Terms*—Sparse representation, sparse coding, dictionary learning, K-SVD, signal denoising, computed tomography.**

I. Introduction

SPARSE representation of signals over redundant dictionaries [1]–[3] is a rapidly evolving field, with state-of-the-art results in many fundamental signal and image processing tasks [4]–[11]. The basic model suggests that natural signals can be compactly expressed, or efficiently approximated, as a linear combination of prespecified *atom signals*, where the linear coefficients are *sparse* (i.e., most of them zero). Formally, letting $\mathbf{x} \in \mathbb{R}^N$ be a column signal, and arranging the atom signals as the columns of the *dictionary* $\mathbf{D} \in \mathbb{R}^{N \times L}$, the sparsity assumption is described by the following *sparse approximation* problem, for which we assume a sparse solution exists:

$$\hat{\gamma} = \underset{\gamma}{\text{Argmin}} \; \|\gamma\|_0^0 \quad \text{Subject To} \quad \|\mathbf{x} - \mathbf{D}\gamma\|_2 \leq \epsilon \; . \tag{1}$$

In this expression, $\hat{\gamma}$ is the *sparse representation* of $\mathbf{x}$, ϵ is the error tolerance, and the function $\|\cdot\|_0^0$, loosely referred to as the *ℓ^0-norm*, counts the non-zero entries of a vector. Though known to be NP-hard in general [12], the above problem is relatively easy to *approximate* using a wide variety of techniques [13]–[21].

A fundamental consideration in employing the above model is the *choice* of the dictionary $\mathbf{D}$. The majority of literature on this topic can be categorized into two basic approaches: the *analytic* approach and the *learning-based* approach. In the first approach, a mathematical model of the data is formulated, and an analytic construction is developed to efficiently represent the model. This generally leads to dictionaries that are highly structured and have a fast numerical implementation. We refer to these as *implicit* dictionaries as they are described by their algorithm rather than their explicit matrix. Dictionaries of this type include Wavelets [22], Curvelets [23], Contourlets [24], Shearlets [25], Complex Wavelets [26], and Bandelets [27], among others.

The second approach suggests using machine learning techniques to infer the dictionary from a set of examples. In this case, the dictionary is typically represented as an explicit matrix, and a training algorithm is employed to adapt the matrix coefficients to the examples. Algorithms of this type include PCA and Generalized PCA [28], the Method of Optimal Directions (MOD) [29], the K-SVD [30], and others. Advantages of this approach are the much finer-tuned dictionaries they produce compared to the analytic approaches, and their significantly better performance in applications. However, this comes at the expense of generating an unstructured dictionary, which is more costly to apply. Also, complexity constraints limit the size of the dictionaries that can be trained in this way, and the dimensions of the signals that can be processed.

In this paper, we present a novel dictionary structure that bridges some of the gap between these two approaches, gaining the benefits of both. The structure is based on a sparsity model of the dictionary atoms over a known *base dictionary*. The new parametric structure leads to a simple and flexible dictionary representation which is both adaptive and efficient. Advantages of the new structure include low complexity, compact representation, stability under noise and reduced overfitting, among others.

A. Related Work

The idea of training dictionaries with a specific structure has been proposed in the past, though research in this direction is still in its early stages. Much of the work so far has focused specifically on developing adaptive *Wavelet* transforms, as in [31]–[34]. These works attempt to adapt various parameters of the Wavelet transform, such as the mother wavelet or the scale and dilation operators, to better suit specific given data.

More recently, an algorithm for training unions of orthonormal bases was proposed in [35]. The suggested dictionary structure takes the form

$$\mathbf{D} = [\ \mathbf{D}_1 \ \mathbf{D}_2 \ \dots \ \mathbf{D}_k \ ] \ , \tag{2}$$

R. Rubinstein, M. Zibulevsky and M. Elad are with the Department of Computer Science, The Technion – Israel Institute of Technology, Haifa 32000, Israel.

This research was supported by the European Community's FP7-FET program, SMALL project, under grant agreement no. 225913. This research was partly supported by the Israel Science Foundation (ISF) grant number 1031/08

where the $\mathbf{D}_i$'s are unitary sub-dictionaries. The structure has the advantage of offering efficient sparse-coding via Block Coordinate Relaxation (BCR) [17], and its training algorithm is simple and relatively efficient. However, the dictionary model itself is relatively restrictive, and its training algorithm shows somewhat weak performance. Furthermore, the structure does not lead to quick forward and adjoint operators, as the dictionary itself remains explicit.

A different approach is proposed in [6], where a semi-multiscale structure is employed. The dictionary model is a concatenation of several scale-specific dictionaries over a dyadic grid, leading (in the 1-D case) to the form:

$$\mathbf{D} = \left(\begin{array}{c|cc|cccc|c} \mathbf{D}_1 & \mathbf{D}_2 & & \mathbf{D}_3 & & & & \cdots \\ & & & & \mathbf{D}_3 & & & \\ & & \mathbf{D}_2 & & & \mathbf{D}_3 & & \\ & & & & & & \mathbf{D}_3 & \end{array} \right) . \tag{3}$$

The multiscale structure is shown to provide excellent results in applications such as denoising and inpainting. Nonetheless, the explicit nature of the dictionary is maintained along with most of the drawbacks of such dictionaries. Indeed, the use of sparse dictionaries to replace the explicit ones in (3) is an exciting option for future study.

Another recent contribution is the *signature dictionary* proposed in [36]. According to the suggested model, the dictionary is described via a compact *signature image*, with each sub-block of this image constituting an atom of the dictionary.[1] The advantages of this structure include near-translation-invariance, reduced overfitting, and faster sparse-coding when utilizing spatial relationships between neighboring signal blocks. On the other hand, the small number of parameters in this model — one coefficient per atom — also makes this dictionary more restrictive than other structures. Indeed, the sparse dictionary model proposed in this paper enhances the dictionary expressiveness by increasing the number of parameters per atom from 1 to $p > 1$, while maintaining other favorable properties of the dictionary.

B. Paper Organization

This paper is organized as follows. We begin in Section II with a description of the dictionary model and its advantages. In Section III we consider the task of training the dictionary from examples, and present an efficient algorithm for doing so. Section IV analyzes and quantifies the complexity of sparse dictionaries, and compares it to other dictionary forms. Simulation results are provided in Section V. We summarize and conclude in Section VI.

C. Notation

- Bold uppercase letters designate matrices ($\mathbf{M}$, $\mathbf{\Gamma}$), and bold lowercase letters designate column vectors ($\mathbf{v}$, $\boldsymbol{\gamma}$). The columns of a matrix are referenced using the corresponding lowercase letter, e.g. $\mathbf{M} = [\,\mathbf{m}_1 \,|\, \ldots \,|\, \mathbf{m}_n\,]$; the elements of a vector are similarly referenced using standard-type letters, e.g. $\mathbf{v} = (v_1, \ldots, v_n)^T$. The notation $\mathbf{0}$ is used to denote the zero vector, with its length inferred from the context.
- Given a single index $I = i_1$ or an ordered sequence of indices $I = (i_1, \ldots, i_k)$, we denote by $\mathbf{M}_I = [\,\mathbf{m}_{i_1} \,|\, \ldots \,|\, \mathbf{m}_{i_k}\,]$ the sub-matrix of $\mathbf{M}$ containing the columns indexed by I, *in the order in which they appear in* I. For vectors we similarly denote the sub-vector $\mathbf{v}_I = (v_{i_1}, \ldots, v_{i_k})^T$. We use the notation $\mathbf{M}_{I,J}$, with J a second index or sequence of indices, to refer to the sub-matrix of $\mathbf{M}$ containing the rows indexed by I and the columns indexed by J, in their respective orders. This notation is used for both access and assignment, so if $I = (2, 4, 6, \ldots, n)$, the statement $\mathbf{M}_{I,j} := \mathbf{0}$ means nullifying the even-indexed entries in the j-th row of $\mathbf{M}$.

II. Sparse Dictionaries

A. Motivation

Selecting a dictionary for sparse signal representation involves balancing between two elementary and seemingly competing considerations. The first is the *complexity* of the dictionary, as the dictionary forward and adjoint operators form the dominant components of most sparse-coding techniques, and these in turn form the core of all sparsity-based signal processing methods. Indeed, techniques such as Matching Pursuit (MP) [2], Orthogonal Matching Pursuit (OMP) [13], Stagewise Orthogonal Matching Pursuit (StOMP) [16], and their variants, all involve costly dictionary-signal computations each iteration. Other common methods such as interior-point Basis Pursuit [1] and FOCUSS [15] minimize a quadratic function each iteration, which is commonly performed using repeated application of the dictionary and its adjoint. Many additional methods rely heavily on the dictionary operators as well.

Over the years, a variety of dictionaries with fast implementations have been designed. For natural images, dictionaries such as Wavelets [22], Curvelets [23], Contourlets [24], and Shearlets [25], all provide fast transforms. However, such dictionaries are *fixed* and limited in their ability to adapt to different types of data. *Adaptability* is thus a second desirable property of a dictionary, and in practical applications, adaptive dictionaries consistently show better performance than generic ones [5], [6], [8], [10], [11]. Unfortunately, adaptive methods usually prefer explicit dictionary representations over structured ones, gaining a higher degree of freedom in the training but sacrificing regularity and efficiency of the result.[2]

Bridging this gap between complexity and adaptivity requires a parametric dictionary model that provides sufficient degrees of freedom. In this work, we propose the *sparse dictionary* model as a simple and effective structure for achieving this goal, based on sparsity of the atoms over a known *base dictionary*. Our approach can be motivated as follows. In Fig. 1

[1] Indeed, both fixed and variable-sized sub-blocks can be considered, though in [36] mostly fixed-sized blocks are studied.

[2] We should note that in *adaptive dictionaries* we are referring to dictionaries whose content can be adapted to different families of signals, typically through a learning process. *Signal-dependent* representation schemes, such as Best Wavelet Packet Bases [31] and Bandelets [27], are another type of adaptive process, but of a very different nature. These methods produce an optimized dictionary for a *given* signal based on its specific characteristics (e.g. frequency content or geometry, respectively), and they are not considered here.

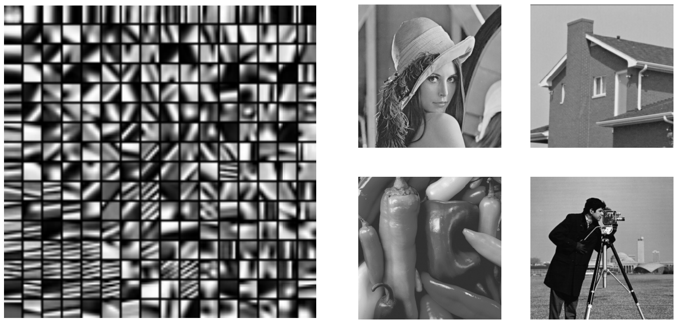


Fig. 1. Left: dictionary for 8×8 image patches, trained using the K-SVD algorithm. Right: images used for the training. Each image contributed 25,000 randomly selected patches, for a total of 100,000 training signals.

we see an example of a dictionary trained using the K-SVD algorithm [30] on a set of 8×8 natural image patches. The algorithm trains an explicit, fully un-constrained dictionary matrix, and yet, we see that the resulting dictionary is highly structured, with noticeably regular atoms. This gives rise to the hypothesis that the dictionary atoms *themselves* may have some underlying sparse structure over a more fundamental dictionary, and as we show in this paper, such a structure can indeed be recovered, and has several favorable properties.

B. Dictionary Model

The sparse dictionary model suggests that each atom of the dictionary has *itself* a sparse representation over some *prespecified* base dictionary $\mathbf{\Phi}$. The dictionary is therefore expressed as

$$\mathbf{D} = \mathbf{\Phi}\mathbf{A} \ , \tag{4}$$

where $\mathbf{A}$ is the atom representation matrix, assumed to be sparse. For simplicity, we focus on matrices $\mathbf{A}$ having a fixed number of non-zeros per column, so $\|\mathbf{a}_i\|_0^0 \leq p$ for some p. The base dictionary $\mathbf{\Phi}$ will generally be chosen to have a quick implicit implementation, and, while $\mathbf{\Phi}$ may have any number of atoms, we assume it to span the signal space. The choice of the base dictionary obviously affects the success of the entire model, and we thus prefer one which already incorporates some prior knowledge about the data. Indeed, if more than one possible base dictionary exists, one may benefit from experimenting with a few different options in order to determine the most suitable one.

In comparison to implicit dictionaries, the dictionary model (4) provides *adaptability* via modification of the matrix $\mathbf{A}$, and can be efficiently trained from examples. Furthermore, as $\mathbf{\Phi}$ can be any dictionary — specifically, any existing implicit dictionary — the model can be viewed as an *extension* to existing dictionaries, adding them a new layer of adaptivity.

In comparison to explicit dictionaries, the sparse structure is significantly more efficient, depending mostly on the choice of $\mathbf{\Phi}$. It is also more compact to store and transmit. Furthermore, as we show later in this paper, the imposed structure acts as a regularizer in dictionary learning processes, and reduces overfitting and instability in the presence of noise. Training a sparse dictionary requires less examples than an explicit one, and produces useable results even when only a few examples are available.

The sparse dictionary model has another interesting interpretation. Assume the signal $\mathbf{x}$ is sparsely represented over the dictionary $\mathbf{D} = \mathbf{\Phi}\mathbf{A}$, so $\mathbf{x} = \mathbf{\Phi}\mathbf{A}\gamma$ for some sparse γ. Therefore, $(\mathbf{A}\gamma)$ is the representation of $\mathbf{x}$ over $\mathbf{\Phi}$. Since both γ and the columns of $\mathbf{A}$ are sparse — having no more than, say, t and p non-zeros, respectively — this representation will have approximately tp non-zeros. However, such quadratic cardinality will generally fall beyond the success range of sparse-approximation techniques [3]. As such, it is no longer considered sparse in terms of the formulation (1), and sparse-coding methods will commonly fail to recover it. Furthermore, given a noisy version of $\mathbf{x}$, attempting to recover it directly over $\mathbf{\Phi}$ using tp atoms will likely result in capturing a significant portion of the noise along with the signal, due to the number of coefficients used.[3]

Through the sparse dictionary structure, we are able to accommodate denser signal representations over $\mathbf{\Phi}$ while essentially by-passing the related difficulties. The reason is that even though every t-sparse signal over $\mathbf{D}$ will generally have a denser tp-representation over $\mathbf{\Phi}$, not *every* tp-representation over $\mathbf{\Phi}$ will *necessarily* fit the model. The proposed model therefore acts as a *regularizer* for the allowed dense representations over $\mathbf{\Phi}$, and by learning the matrix $\mathbf{A}$, we are expressing in some form the complicated dependencies between its atoms.

III. Learning Sparse Dictionaries

We now turn to the question of *designing* a sparse dictionary for sparse signal representation. A straightforward approach would be to select some general (probably learned) dictionary $\mathbf{D}_0$, choose a base dictionary $\mathbf{\Phi}$, and sparse-code the atoms in $\mathbf{D}_0$ to obtain $\mathbf{D} = \mathbf{\Phi}\mathbf{A} \approx \mathbf{D}_0$. This naive approach, however, is clearly sub-optimal: specifically, the dictionary $\mathbf{\Phi}$ must be sufficiently compatible with $\mathbf{D}_0$, or else the representations in $\mathbf{A}$ may not be very sparse. Simulation results indicate that such dictionaries indeed perform poorly in practical signal processing applications.

A more desirable approach would be to learn the sparse dictionary using a process that is aware of the dictionary's specific structure. We adopt an approach which continues the line of work in [30], and develop a K-SVD-like learning scheme for training the sparse dictionary from examples. The algorithm is inspired by the Approximate K-SVD implementation presented in [37], which we briefly review.

A. K-SVD and Its Approximate Implementation

The K-SVD algorithm accepts an initial overcomplete dictionary matrix $\mathbf{D}_0 \in \mathbb{R}^{N \times L}$, a number of iterations k, and a set of examples arranged as the columns of the matrix $\mathbf{X} \in \mathbb{R}^{N \times R}$. The algorithm aims to iteratively improve the dictionary by approximating the solution to

$$\underset{\mathbf{D},\mathbf{\Gamma}}{\text{Min}} \ \|\mathbf{X} - \mathbf{D}\mathbf{\Gamma}\|_F^2 \quad \text{Subject To} \quad \begin{array}{l} \forall i \ \ \|\gamma_i\|_0^0 \leq t \\ \forall j \ \ \|\mathbf{d}_j\|_2 = 1 \end{array} . \tag{5}$$

[3]For white noise and a signal of length N, the expected remaining noise in a recovered signal using t atoms is approximately t/N the initial noise energy, due to the orthogonal projection.

Note that in this formulation, the atom normalization constraint is commonly added for convenience, though it does not have any practical significance to the result.

The K-SVD iteration consists of two basic steps: (i) sparse-coding the signals in $\mathbf{X}$ given the current dictionary estimate, and (ii) updating the dictionary atoms given the sparse representations in $\mathbf{\Gamma}$. The sparse-coding step can be implemented using any sparse-approximation method. The dictionary update is performed one atom at a time, optimizing the target function for each atom individually while keeping the remaining atoms fixed.

The atom update is carried out while preserving the sparsity constraints in (5). To achieve this, the update uses only those signals in $\mathbf{X}$ whose sparse representations use the current atom. Denoting by I the indices of the signals in $\mathbf{X}$ that use the j-th atom, the update of this atom is obtained by minimizing the target function

$$\|\mathbf{X}_I - \mathbf{D}\mathbf{\Gamma}_I\|_F^2 \tag{6}$$

for both the atom and its corresponding coefficient row in $\mathbf{\Gamma}_I$. The resulting problem is a simple rank-1 approximation, given by

$$\{\mathbf{d},\mathbf{g}\} := \underset{\mathbf{d},\mathbf{g}}{\text{Argmin}}\ \|\mathbf{E}-\mathbf{d}\,\mathbf{g}^T\|_F^2 \quad \text{Subject To} \quad \|\mathbf{d}\|_2 = 1\ , \tag{7}$$

where $\mathbf{E} = \mathbf{X}_I - \sum_{i\neq j}\mathbf{d}_i\mathbf{\Gamma}_{i,I}$ is the error matrix without the j-th atom, and $\mathbf{d}$ and $\mathbf{g}^T$ are the updated atom and coefficient row, respectively. The problem can be solved directly via an SVD decomposition, or more efficiently using some numerical power method.

In practice, the exact solution of (7) can be quite computationally demanding, especially when the number of training signals is large. As an alternative, an approximate solution may be used to reduce the complexity of this task [37]. The simplified update step is obtained by applying a single iteration of alternated-optimization [17], [38], given by

$$\begin{array}{rcl} \mathbf{d} & := & \mathbf{E}\mathbf{g}/\|\mathbf{E}\mathbf{g}\|_2 \\ \mathbf{g} & := & \mathbf{E}^T\mathbf{d} \end{array} . \tag{8}$$

The above process is known to ultimately converge to the optimum,[4] and when truncated, supplies an approximation which still reduces the penalty term. Also, this process eliminates the need to explicitly compute the matrix $\mathbf{E}$, as only its products with vectors are required.[5]

B. The Sparse K-SVD Algorithm

To train a sparse dictionary, we use the same basic framework as the original K-SVD algorithm. Specifically, we aim to (approximately) solve the optimization problem

$$\underset{\mathbf{A},\mathbf{\Gamma}}{\text{Min}}\ \|\mathbf{X}-\mathbf{\Phi}\mathbf{A}\mathbf{\Gamma}\|_F^2$$
$$\text{Subject To} \quad \begin{cases} \forall i \quad \|\gamma_i\|_0^0 \le t \\ \forall j \quad \|\mathbf{a}_j\|_0^0 \le p\ ,\ \ \|\Phi\mathbf{a}_j\|_2 = 1 \end{cases} , \tag{9}$$

alternating sparse-coding and dictionary update steps for a fixed number of iterations. The notable change is in the atom update step: as opposed to the original K-SVD algorithm, in this case the atom is constrained to the form $\mathbf{d} = \mathbf{\Phi}\mathbf{a}$ with $\|\mathbf{a}\|_0^0 \le p$. The modified atom update is therefore given by

$$\{\mathbf{a},\mathbf{g}\} := \underset{\mathbf{a},\mathbf{g}}{\text{Argmin}}\ \|\mathbf{E}-\mathbf{\Phi}\mathbf{a}\,\mathbf{g}^T\|_F^2 \quad \text{Subject To} \quad \begin{array}{r} \|\mathbf{a}\|_0^0 \le p \\ \|\mathbf{\Phi}\mathbf{a}\|_2 = 1 \end{array} , \tag{10}$$

with $\mathbf{E}$ defined as in (7).

Interestingly, our problem is closely related to a different problem known as *Sparse Matrix Approximation* (here SMA), recently raised in the context of Kernel-SVM methods [39]. The SMA problem is formulated similar to problem (10), but replaces the rank-1 matrix $\mathbf{a}\mathbf{g}^T$ with a general matrix $\mathbf{T}$, and the sparsity constraint on $\mathbf{a}$ with a constraint on the number of non-zero rows in $\mathbf{T}$. Our problem is therefore essentially a *rank-constrained version* of the original SMA problem. In [39], the authors suggest a greedy OMP-like algorithm for solving the problem, utilizing randomization to deal with the large amount of work involved. Unfortunately, while this approach is likely extendable to the rank-constrained case, it leads to a computationally intensive process which is impractical for large problems.

Our approach therefore takes a different path to solving the problem, employing an alternated-optimization technique over $\mathbf{a}$ and $\mathbf{g}$ parallel to (8). We point out that as opposed to (8), the process here does *not* generally converge to the optimum when repeated, due to the non-convexity of the problem. Nonetheless, the method does guarantee a reduction in the target function value, which is essentially sufficient for our purposes.

To simplify the derivation, we note that (10) may be solved *without* the norm constraint on $\mathbf{\Phi}\mathbf{a}$, and adding a post-processing step which transfers energy between $\mathbf{a}$ and $\mathbf{g}$ to achieve $\|\mathbf{\Phi}\mathbf{a}\|_2 = 1$ while keeping $\mathbf{a}\mathbf{g}^T$ fixed. The simplified problem is given by

$$\{\mathbf{a},\mathbf{g}\} := \underset{\mathbf{a},\mathbf{g}}{\text{Argmin}}\ \|\mathbf{E}-\mathbf{\Phi}\mathbf{a}\,\mathbf{g}^T\|_F^2 \quad \text{Subject To} \quad \|\mathbf{a}\|_0^0 \le p\ . \tag{11}$$

We also note that the solution to this problem is guaranteed to be non-zero for all $\mathbf{E} \neq 0$, hence the described re-normalization of $\mathbf{a}$ and $\mathbf{g}$ is possible.

Optimizing over $\mathbf{g}$ in (11) is straightforward, and given by

$$\mathbf{g} := \mathbf{E}^T\mathbf{\Phi}\mathbf{a}/\|\mathbf{\Phi}\mathbf{a}\|_2^2\ . \tag{12}$$

Optimizing over $\mathbf{a}$, however, requires more attention. The minimization task for $\mathbf{a}$ is given by:

$$\mathbf{a} := \underset{\mathbf{a}}{\text{Argmin}}\ \|\mathbf{E}-\mathbf{\Phi}\mathbf{a}\,\mathbf{g}^T\|_F^2 \quad \text{Subject To} \quad \|\mathbf{a}\|_0^0 \le p\ . \tag{13}$$

The straightforward approach to this problem is to rewrite $\mathbf{E}$ as a column vector $\mathbf{e}$, and formulate the problem as an ordinary

[4] Applying two consecutive iterations of this process produces $\mathbf{d}^{j+1} = \mathbf{E}\mathbf{E}^T\mathbf{d}^j/\|\mathbf{E}\mathbf{E}^T\mathbf{d}^j\|_2$, which is the well-known power iteration for $\mathbf{E}\mathbf{E}^T$. The process converges, under reasonable assumptions, to the largest eigenvector of $\mathbf{E}\mathbf{E}^T$ — also the largest left singular vector of $\mathbf{E}$.

[5] Specifically, $\mathbf{E}\mathbf{g} = \mathbf{X}_I\mathbf{g} - \sum_{i\neq j}\mathbf{d}_i(\mathbf{\Gamma}_{i,I}\mathbf{g})$ can be computed via a series of vector inner products $\xi_i = \mathbf{\Gamma}_{i,I}\mathbf{g}$, followed by a vector sum $\sum_{i\neq j}\xi_i\mathbf{d}_i$ and a matrix-vector product $\mathbf{X}_I\mathbf{g}$. This is significantly faster and more memory-efficient than the explicit computation of $\mathbf{E}$, which involves matrix-matrix operations. The same applies to the computation of $\mathbf{E}^T\mathbf{d}$.

sparse-coding task for $\mathbf{e}$ (we use $\otimes$ to denote the Kronecker matrix product [40]):

$$\mathbf{a} := \underset{\mathbf{a}}{\text{Argmin}} \; \|\mathbf{e} - (\mathbf{g} \otimes \mathbf{\Phi})\mathbf{a}\|_2^2 \quad \text{Subject To} \quad \|\mathbf{a}\|_0^0 \le p \; . \quad (14)$$

However, this leads to an intolerably large optimization problem, as the length of the signal to sparse-code is of the same order of magnitude as the entire dataset. Instead, we show that problem (13) is equivalent to a much simpler sparse-coding problem, namely

$$\mathbf{a} := \underset{\mathbf{a}}{\text{Argmin}} \; \|\mathbf{E}\mathbf{g} - \mathbf{\Phi}\mathbf{a}\|_2^2 \quad \text{Subject To} \quad \|\mathbf{a}\|_0^0 \le p \; . \quad (15)$$

Here, the vector $\mathbf{E}\mathbf{g}$ is of the same length as a single training example, and the dictionary is the base dictionary $\mathbf{\Phi}$ which is assumed to have an efficient implementation; therefore, this problem is significantly easier to handle than the previous one. Also, as discussed above, the vector $\mathbf{E}\mathbf{g}$ itself is much easier to compute than the vector $\mathbf{e}$, which is just a vectorized version of the matrix $\mathbf{E}$.

To establish the equivalence between the problems (13) and (15), we use the following Lemma:

Lemma 1: Let $\mathbf{X} \in \mathbb{R}^{N \times M}$ and $\mathbf{Y} \in \mathbb{R}^{N \times K}$ be two matrices, and $\mathbf{v} \in \mathbb{R}^M$ and $\mathbf{u} \in \mathbb{R}^K$ be two vectors. Also assume that $\mathbf{v}^T\mathbf{v} = 1$. Then the following holds:

$$\|\mathbf{X} - \mathbf{Y}\mathbf{u}\mathbf{v}^T\|_F^2 = \|\mathbf{X}\mathbf{v} - \mathbf{Y}\mathbf{u}\|_2^2 + f(\mathbf{X}, \mathbf{v}) \; .$$

Proof: The equality follows from elementary properties of the trace function:

$$\begin{aligned}
&\|\mathbf{X} - \mathbf{Y}\mathbf{u}\mathbf{v}^T\|_F^2 = \\
&= Tr((\mathbf{X} - \mathbf{Y}\mathbf{u}\mathbf{v}^T)^T(\mathbf{X} - \mathbf{Y}\mathbf{u}\mathbf{v}^T)) \\
&= Tr(\mathbf{X}^T\mathbf{X}) - 2Tr(\mathbf{X}^T\mathbf{Y}\mathbf{u}\mathbf{v}^T) + Tr(\mathbf{v}\mathbf{u}^T\mathbf{Y}^T\mathbf{Y}\mathbf{u}\mathbf{v}^T) \\
&= Tr(\mathbf{X}^T\mathbf{X}) - 2Tr(\mathbf{v}^T\mathbf{X}^T\mathbf{Y}\mathbf{u}) + Tr(\mathbf{v}^T\mathbf{v}\mathbf{u}^T\mathbf{Y}^T\mathbf{Y}\mathbf{u}) \\
&= Tr(\mathbf{X}^T\mathbf{X}) - 2\mathbf{v}^T\mathbf{X}^T\mathbf{Y}\mathbf{u} + \mathbf{u}^T\mathbf{Y}^T\mathbf{Y}\mathbf{u} \\
&= Tr(\mathbf{X}^T\mathbf{X}) - 2\mathbf{v}^T\mathbf{X}^T\mathbf{Y}\mathbf{u} + \mathbf{u}^T\mathbf{Y}^T\mathbf{Y}\mathbf{u} + \\
&\qquad + \mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} - \mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} \\
&= \|\mathbf{X}\mathbf{v} - \mathbf{Y}\mathbf{u}\|_2^2 + Tr(\mathbf{X}^T\mathbf{X}) - \mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} \\
&= \|\mathbf{X}\mathbf{v} - \mathbf{Y}\mathbf{u}\|_2^2 + f(\mathbf{X}, \mathbf{v}) \; .
\end{aligned}$$

■

The Lemma implies that, assuming $\mathbf{g}^T\mathbf{g} = 1$, then for every representation vector $\mathbf{a}$,

$$\|\mathbf{E} - \mathbf{\Phi}\mathbf{a}\mathbf{g}^T\|_F^2 = \|\mathbf{E}\mathbf{g} - \mathbf{\Phi}\mathbf{a}\|_2^2 + f(\mathbf{E}, \mathbf{g}) \; .$$

Clearly the important point in this equality is that the two sides differ by a constant independent of $\mathbf{a}$. Thus, the target function in (13) can be safely replaced with the right hand side of the equality (sans the constant), establishing the equivalence to (15).

When using the Lemma to solve (13), we note that the energy assumption on $\mathbf{g}$ can be easily overcome, as dividing $\mathbf{g}$ by a non-zero constant simply results in a solution $\mathbf{a}$ scaled by that same constant. Thus (13) can be solved for any $\mathbf{g}$ by normalizing it to unit length, applying the Lemma, and re-scaling the solution $\mathbf{a}$ by the appropriate factor. Conveniently, since $\mathbf{a}$ is independently re-normalized at the end of the process, this re-scaling can be skipped completely, scaling $\mathbf{a}$ instead to $\|\mathbf{\Phi}\mathbf{a}\|_2 = 1$ and continuing with the update of $\mathbf{g}$.

```
1: Input: Signal set $\mathbf{X}$, base dictionary $\mathbf{\Phi}$, initial dictionary representation $\mathbf{A}_0$, target atom sparsity $p$, target signal sparsity $t$, number of iterations $k$.
2: Output: Sparse dictionary representation $\mathbf{A}$ and sparse signal representations $\mathbf{\Gamma}$ such that $\mathbf{X} \approx \mathbf{\Phi}\mathbf{A}\mathbf{\Gamma}$
3: Init: Set $\mathbf{A} := \mathbf{A}_0$
4: for $n = 1 \ldots k$ do
5:     $\forall i : \mathbf{\Gamma}_i := \underset{\boldsymbol{\gamma}}{\text{Argmin}} \|\mathbf{x}_i - \mathbf{\Phi}\mathbf{A}\boldsymbol{\gamma}\|_2^2$ Subject To $\|\boldsymbol{\gamma}\|_0^0 \le t$
6:     for $j = 1 \ldots L$ do
7:         $\mathbf{A}_j := \mathbf{0}$
8:         $I := \{$indices of the signals in $\mathbf{X}$ whose reps. use $\mathbf{a}_j\}$
9:         $\mathbf{g} := \mathbf{\Gamma}_{j,I}^T$
10:        $\mathbf{g} := \mathbf{g}/\|\mathbf{g}\|_2$
11:        $\mathbf{z} := \mathbf{X}_I\mathbf{g} - \mathbf{\Phi}\mathbf{A}\mathbf{\Gamma}_I\mathbf{g}$
12:        $\mathbf{a} := \underset{\mathbf{a}}{\text{Argmin}} \|\mathbf{z} - \mathbf{\Phi}\mathbf{a}\|_2^2$ Subject To $\|\mathbf{a}\|_0^0 \le p$
13:        $\mathbf{a} := \mathbf{a}/\|\mathbf{\Phi}\mathbf{a}\|_2$
14:        $\mathbf{A}_j := \mathbf{a}$
15:        $\mathbf{\Gamma}_{j,I} := (\mathbf{X}_I^T\mathbf{\Phi}\mathbf{a} - (\mathbf{\Phi}\mathbf{A}\mathbf{\Gamma}_I)^T\mathbf{\Phi}\mathbf{a})^T$
16:    end for
17: end for
```

Fig. 2. The Sparse K-SVD Algorithm.

Combining the pieces, the final atom update process consists of the following steps: (i) normalizing $\mathbf{g}$ to unit length; (ii) solving (15) for $\mathbf{a}$; (iii) normalizing $\mathbf{a}$ to $\|\mathbf{\Phi}\mathbf{a}\|_2 = 1$; and ($iv$) updating $\mathbf{g} := \mathbf{E}^T\mathbf{\Phi}\mathbf{a}$. This process may generally be repeated, though we have found little practical advantage in doing so. The complete Sparse K-SVD algorithm is detailed in Fig. 2. Figs. 3,4 show an example result, obtained by applying this algorithm to the same training set as that used to train the dictionary in Fig. 1.

IV. Complexity of Sparse Dictionaries

Sparse dictionaries are generally much more efficient than explicit ones, and provide significant gains especially for larger dictionaries and higher-dimensional signals. In this section we discuss the complexity of sparse dictionaries and describe the cases where they are most advantageous. To focus the discussion, we concentrate on the case of Orthogonal Matching Pursuit (OMP) sparse-coding, which is a widely used method which is relatively simple to analyze.

A. Sparse Dictionary Operator Complexity

The dictionary structure (4) is implemented by multiplying the sparse representation γ by $\mathbf{A}$ and applying $\mathbf{\Phi}$. In the following, we assume that $\mathbf{A}$ has a total of pL non-zeros, and that $\mathbf{\Phi}$ has an efficient implementation with complexity T_{Φ}.

Operations with sparse matrices are not immediate to analyze, with many factors affecting actual performance (see [41] for some insights on the topic). In this paper we make the simplifying assumption that the complexity of such operations

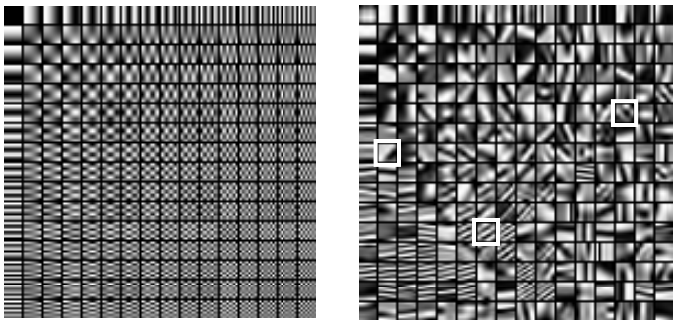


Fig. 3. Left: overcomplete DCT dictionary for 8×8 image patches. Right: sparse dictionary trained over the overcomplete DCT using Sparse K-SVD. Dictionary atoms are represented using 6 coefficients each. Marked atoms are magnified in Fig. 4.

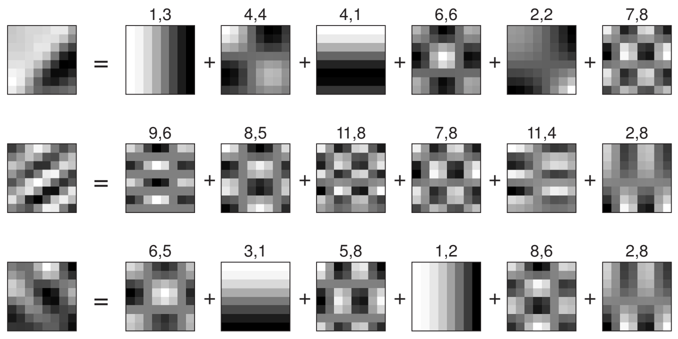


Fig. 4. Some atoms from the trained dictionary in Fig. 3, and their overcomplete DCT components. The index pair above each overcomplete DCT atom denotes the wave number of the atom, with (1,1) corresponding to the upper-left atom, (16,1) corresponding to the lower-left atom, etc. In each row, the components are ordered by decreasing magnitude of the coefficients, the most significant component on the left. The coefficients themselves are not shown due to space limitations, but are all of the same order of magnitude.

is proportional to the number of non-zeros in the sparse matrix, so multiplying a vector by a sparse matrix with Z non-zeros is equivalent to multiplying it by a full matrix with αZ ($\alpha \geq 1$) coefficients (a total of $2\alpha Z$ multiplications and additions). For a concrete figure, we use $\alpha = 7$, which is roughly what our machine (an Intel Core 2 running *Matlab 2007a*) produced. With this assumption, the complexity of the sparse dictionary $\mathbf{D} = \mathbf{\Phi A}$ is given by

$$T_D \{\textit{sparse-dict}\} = 2\alpha pL + T_\Phi \ . \quad (16)$$

The base dictionary $\mathbf{\Phi}$ will usually be chosen to have a compact representation and sub-N^2 implementation. Indeed, most implicit dictionaries provide these properties, with complexities ranging from linear to low-degree (< 2) polynomial. In the following analysis we focus on two very common types of base dictionaries, which roughly represent this range:

Separable dictionaries: Dictionaries which are the Kronecker product of several 1-dimensional dictionaries. Assuming $\mathbf{\Phi}_0 \in \mathbb{R}^{n\times m}$ is a dictionary for 1-D signals of length n, the dictionary $\mathbf{\Phi} = \mathbf{\Phi}_0 \otimes \mathbf{\Phi}_0 \in \mathbb{R}^{n^2\times m^2}$ can be constructed for representing $n \times n$ signals arranged in column-major order as vectors of length n^2. The dictionary adjoint is separable as well and given by $\mathbf{\Phi}^T = \mathbf{\Phi}_0^T \otimes \mathbf{\Phi}_0^T$. The dictionary and its adjoint are efficiently implemented by applying $\mathbf{\Phi}_0$ or $\mathbf{\Phi}_0^T$ (respectively) along each of the signal dimensions, in any order. Denoting $a = m/n$, and assuming $\mathbf{\Phi}_0$ is applied via explicit matrix multiplication, the complexity of this dictionary in the 2-D case is

$$T_\Phi = 2N\sqrt{M}(1 + a) \quad (17)$$

where $N = n^2$ and $M = m^2$ are the dictionary dimensions. Examples of separable dictionaries include the DCT (Fourier), overcomplete DCT (Fourier), and Wavelet dictionaries, among others. Generalizations to higher dimensions are straightforward to derive.

Linear-time dictionaries: Dictionaries which are implemented with a constant number of operations per sample, so

$$T_\Phi = \beta N \quad (18)$$

for some constant value β. Examples include the Wavelet, Contourlet, and Complex Wavelet dictionaries, among others.

B. Complexity of OMP

OMP is a greedy sparse-coding algorithm which has several efficient implementations. One of the most common ones is *OMP-Cholesky* [21], [37], [42] which employs a progressive Cholesky decomposition to perform efficient matrix inversions.

When the dictionary is represented explicitly, the number of operations performed by OMP-Cholesky can be shown to be [37]

$$T_{omp}\{\textit{explicit-dict}\} = 2tNL + 2t^2N + 2t(L+N) + t^3 \ , \quad (19)$$

where t is the number of OMP iterations (also the number of selected atoms), and N and L are the dictionary dimensions. Note that since $N \sim L \gg t$, the dominant term in this expression is the first one, which is associated with the explicit dictionary operator.

With a sparse dictionary, one can show that the complexity of OMP-Cholesky becomes

$$T_{omp}\{\textit{sparse-dict}\} = 4tT_\Phi + 2\alpha tpL + 2t(L+N) + t^3 \ , \quad (20)$$

where p is the sparsity of the dictionary atoms over the base dictionary, and α is the sparse operation overhead factor discussed above (for a derivation of this result we refer the reader to [43]). We observe that the term proportional to tNL in (19) has been replaced by terms proportional to tT_Φ and tpL in this expression. Therefore, when the base dictionary Φ has an efficient implementation, and assuming $p \ll N$, the sparse dictionary indeed provides an order-of-magnitude complexity advantage over an explicit one.

The complexity gain of OMP-Cholesky with a sparse dictionary is depicted in Fig. 5. The Figure shows the speedup factor of OMP-Cholesky with a sparse dictionary compared to an explicit one, for 2-D and 3-D signals, and using either a separable or linear base dictionary. The x-axis corresponds to the signal length N, where $N = n^d$ for $d = 2, 3$.

As can be seen, sparse dictionaries provide a pronounced performance increase compared to explicit ones, especially in the 3-D case where the speedup is around $\times 5 - \times 10$ for the separable case and $\times 10 - \times 30$ for the linear case. We also see that the speedup continues to increase as the signal becomes

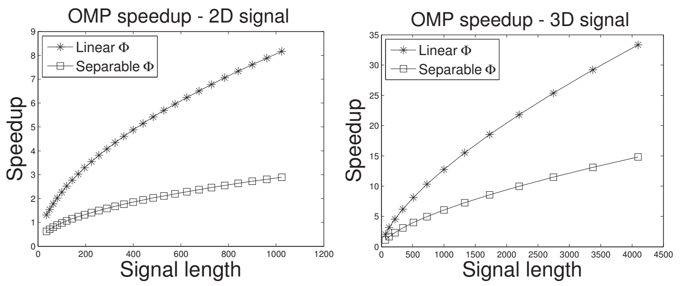


Fig. 5. Speedup of OMP-Cholesky using a sparse dictionary compared to an explicit dictionary. Left: speedup for 2-D signals. Right: speedup for 3-D signals. Signal length is $N = n^d$ where n is the block size and $d = 2, 3$ is the number of dimensions. Dictionary size is chosen to be $n^d \times (n+3)^d$ (base dictionary is of the same size, and the matrix $\mathbf{A}$ is square). Atom sparsity is set to $p = n/2$ in the 2-D case and to $p = n$ in the 3-D case. Complexity of linear dictionary is $T_\Phi = 8N$.

larger. In a practical signal processing application, where large numbers of signals are involved, this difference may make sparse dictionaries the only feasible option.

C. Dictionary Training

Seeing the complexity gain in sparse-coding, it is unsurprising that Sparse K-SVD is similarly much faster than the standard and approximate K-SVD methods. Indeed, the gain mostly stems from the acceleration in the sparse-coding step (line 5 of the algorithm). In the asymptotic case where $t \sim p \ll M \sim L \sim N \ll R$, with R the number of training signals, the complexity of the approximate K-SVD becomes proportional to the complexity of its sparse-coding method [37]. Indeed, this result is easily extended to Sparse K-SVD as well; consequently, Sparse K-SVD is faster than the approximate K-SVD by *approximately the sparse-coding speedup*.

As we will see in the experimental section, a more significant (though less obvious) advantage of Sparse K-SVD is the reduction in overfitting. This results in a substantially smaller number of examples required for the training process, and leads to a further reduction in training complexity.

V. Applications and Simulation Results

The sparse dictionary structure has several advantages. It enables larger dictionaries to be trained, for instance to fill-in bigger holes in an image inpainting task [6]. Specifically of interest are dictionaries for *high-dimensional* data. Indeed, employing sparsity-based techniques to high-dimensional signal data is challenging, as the complicated nature of these signals limits the availability of analytic transforms for them, while the complexity of the training problem constrains the use of existing adaptive techniques as well. The sparse dictionary structure — coupled with the Sparse K-SVD algorithm — makes it possible to process such signals and design rich dictionaries for representing them.

Another application for sparse dictionaries is signal compression. Using an adaptive dictionary to code signal blocks leads to sparser representations than generic dictionaries, and therefore to higher compression rates. Such dictionaries, however, must be stored alongside the compressed data, and this becomes a limiting factor when used with explicit dictionary representations. Sparse dictionaries significantly reduce this overhead. In essence, wherever a prespecified dictionary is used for compression, one may introduce adaptivity by training a sparse dictionary over this predesigned one. The facial compression algorithm in [8] makes a good candidate for such a technique, and research in this direction is currently undergoing.

In the following experiments we focus on a specific type of signal, namely 3-D computed tomography (CT) imagery. We compare the sparse and explicit dictionary structures in their ability to adapt to specific data and generalize from it. We also provide concrete CT denoising results for the two dictionary structures, and show that the sparse dictionary consistently outperforms the explicit one, while operating substantially faster. Our simulations make use of the CT data provided by the NIH *Visible Human Project*[6].

A. Training and Generalization

Training a large dictionary generally requires increasing the number of training signals accordingly. Heuristically, we expect the training set to grow *at least* linearly with the number of atoms, to guarantee sufficient information for the training process. Uniqueness is in fact only known to exist for an *exponential* number of training signals in the general case [44]. Unfortunately, large numbers of training signals quickly become impractical when the dictionary size increases, and it is therefore highly desirable to develop methods for reducing the number of required examples.

In the following experiments we compare the generalization performance of K-SVD versus Sparse K-SVD with small to moderate training sets. We use both methods to train a 512×1000 dictionary for $8 \times 8 \times 8$ signal patches. The data is taken from the *Visible Male - Head* CT volume. We extract the training blocks from a noisy version of the CT volume (PSNR=17dB), while the validation blocks are extracted directly from the original volume. Training is performed using 10,000, 30,000, and 80,000 training blocks, randomly selected from the noisy volume, and with each set including all the signals in the previous sets. The validation set consists of 20,000 blocks, randomly selected from the locations not used for training. The initial dictionary for both methods is the overcomplete DCT dictionary[7]. For Sparse K-SVD, we use the overcomplete DCT as the base dictionary, and set the initial $\mathbf{A}$ matrix to identity. The sparse dictionary is trained using either 8, 16, or 24 coefficients per atom.

Fig. 6 shows our results. The top and bottom rows show the performance of the K-SVD and Sparse K-SVD dictionaries on the training and validation sets (respectively) during the algorithm iterations. Following [5], we code the noisy training

[6]http://www.nlm.nih.gov/research/visible/.

[7]The 1-D $N \times L$ overcomplete DCT dictionary is essentially a cropped version of the orthogonal $L \times L$ DCT dictionary matrix. The k-D overcomplete DCT dictionary is simply the Kronecker product of k 1-D overcomplete DCT dictionaries. Note that the number of atoms in such a dictionary is L^k, and must have a whole k-th root (in our case, $10^3 = 1000$ atoms).

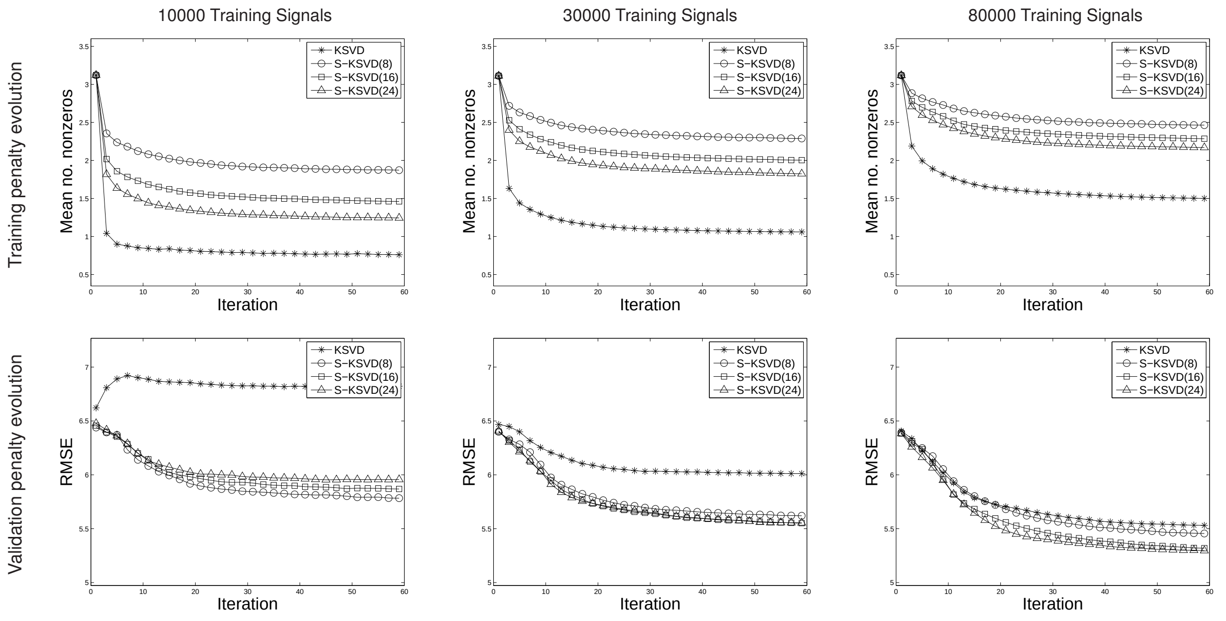


Fig. 6. Training and validation results for patches from *Visible Male - Head*. Training signals are taken from the noisy volume (PSNR=17dB), and validation signals are taken from the original volume. Block size is $8 \times 8 \times 8$, and dictionary size is 512×1000. Training signals (noisy) are sparse-coded using an error stopping criterion proportional to the noise; validation signals (noiseless) are sparse-coded using a fixed number of atoms. Shown penalty functions are respectively the average number of non-zeros in the sparse representations and the coding RMSE. Sparse K-SVD with atom-sparsity p is designated in the legend as S-KSVD(p).

signals using an error target proportional to the noise, and have the ℓ^0 sparsity of the representations as the training target function. We evaluate performance on the validation signals (which are noiseless) by sparse-coding with a fixed number of atoms, and measuring the resulting representation RMSE.

We can see that the average number of non-zeros for the training signals decreases rapidly in the K-SVD case, especially for smaller training sets. However, this phenomena is mostly an indication of overfitting, as the drop is greatly attenuated when adding training data. The overfitting consequently leads to degraded performance on the validation set, as can be seen in the bottom row.

In contrast, the sparse dictionary shows much more stable performance. Even with only 10,000 training signals, the learned dictionary performs reasonably well on the validation signals. As the training set increases, we find that the performance of the sparsest ($p = 8$) dictionary begins to weaken, indicating the limits of the constrained structure. However, for $p = 16$ and $p = 24$ the sparse dictionary continues to gradually improve, and consistently outperforms the standard K-SVD. It should be noted that while the K-SVD dictionary is also expected to improve as the training set is increased — possibly surpassing the Sparse K-SVD at some point — such large training sets are extremely difficult to process, to the point of being impractical.

B. CT Volume Denoising

We used the adaptive K-SVD denoising algorithm [5] to evaluate CT volume denoising performance. The algorithm trains an overcomplete dictionary using blocks from the noisy signal, and then denoises the signal using this dictionary, averaging the denoised blocks when they overlap in the result. We should mention that newer, state-of-the-art variants of the K-SVD denoising scheme, such as multi-scale K-SVD denoising [6] and non-local simultaneous sparse-coding [11], could also be used here to further improve the results, however in this work we focus on the original denoising formulation for simplicity.

We performed our experiments on the *Visible Male - Head* and *Visible Female - Ankle* volumes. The intensity values of each volume were first fitted to the range [0,255] for compatibility with image denoising results, and then subjected to additive white Gaussian noise with varying standard deviations of $5 \leq \sigma \leq 100$. We tested both 2-D denoising, in which each CT slice is processed separately, and 3-D denoising, in which the volume is processed as a whole. The atom sparsity for these experiments was heuristically set to $p = 6$ for the 2-D case and $p = 16$ for the 3-D case, motivated by results such as those in Fig. 6. Our denoising results are actually expected to improve as these values are increased, up to a point where overfitting becomes a factor. However, we preferred to limit the atom sparsity in these experiments to maintain the complexity advantage of the sparse dictionary. Further work may establish a more systematic way of selecting these values.

Our denoising results are summarized in Table I. Table II shows the running times obtained by our Intel Core 2 machine for the different algorithms in the 3-D case. For completeness, Table III lists the full set of parameters used in these experiments. Some actual denoising results are shown in Fig. 7.

The most evident result in Table I is that 3-D denoising is indeed substantially more effective than 2-D denoising for this task, with significant gains of 1.5dB-4dB in all cases. These results provide further motivation for the move towards larger dictionaries and higher-dimensional signals, where sparse dic-

TABLE II
RUNNING TIMES OF K-SVD, SPARSE K-SVD, AND OVERCOMPLETE DCT DENOISING FOR THE RESULTS IN TABLE II (3-D CASE). TIMINGS INCLUDE DICTIONARY TRAINING. SIMULATIONS WERE PERFORMED ON AN INTEL CORE 2 PROCESSOR, UTILIZING A SINGLE CORE. NOTE: RUNNING TIMES LISTED HERE CAN BE SIGNIFICANTLY IMPROVED, AND THE READER IS REFERRED TO SECTION V-C FOR A DISCUSSION.

	Vis. F. Ankle							Vis. M. Head						
Dictionary / σ	5	10	20	30	50	75	100	5	10	20	30	50	75	100
K-SVD	22:06:27	10:11:06	4:07:33	2:27:47	1:24:23	57:48	45:36	25:32:37	11:58:59	4:54:04	3:00:27	1:39:32	1:04:29	46:32
Sparse K-SVD	1:08:49	33:44	13:05	8:07	5:15	4:26	3:54	1:14:37	34:26	14:11	9:44	5:56	4:47	4:04
O-DCT	24:51	13:27	4:51	2:59	1:45	1:17	1:03	31:45	14:15	6:10	4:01	2:25	1:30	1:12

TABLE I
CT DENOISING RESULTS USING K-SVD, SPARSE K-SVD, AND OVERCOMPLETE DCT DICTIONARIES. VALUES REPRESENT PEAK SNR (DB), AND ARE AVERAGED OVER 4 EXECUTIONS. BOLD NUMERALS DENOTE THE BEST RESULT IN EACH TEST UP TO A 0.1DB DIFFERENCE.

Test	σ / PSNR	2-D Denoising			3-D Denoising		
		ODCT	KSVD	S-KSVD	ODCT	KSVD	S-KSVD
Vis. F. Ankle	5 / 34.15	43.07	43.23	43.15	44.42	**44.64**	**44.64**
	10 / 28.13	39.25	39.70	39.45	40.91	**41.24**	**41.22**
	20 / 22.11	35.34	36.12	35.87	37.57	**37.98**	**38.03**
	30 / 18.59	33.01	33.76	33.67	35.62	36.02	**36.21**
	50 / 14.15	30.15	30.43	30.48	33.07	33.48	**33.85**
	75 / 10.63	27.88	27.84	27.92	31.18	31.63	**31.98**
	100 / 8.13	26.42	26.31	26.39	29.89	30.08	**30.46**
Vis. M. Head	5 / 34.15	43.61	43.94	43.72	**45.11**	**45.12**	**45.17**
	10 / 28.13	39.34	40.13	39.70	41.46	**41.56**	**41.57**
	20 / 22.11	34.97	36.08	35.81	37.77	**38.02**	**38.10**
	30 / 18.59	32.48	33.13	33.08	35.54	35.91	**36.18**
	50 / 14.15	29.62	29.67	29.74	32.79	33.08	**33.56**
	75 / 10.63	27.84	27.75	27.82	30.73	30.69	**31.09**
	100 / 8.13	26.51	26.40	26.48	29.60	29.47	**29.72**

TABLE III
PARAMETERS OF THE K-SVD DENOISING ALGORITHM (SEE [5] FOR MORE DETAILS). NOTE THAT A LAGRANGE MULTIPLIER OF 0 MEANS THAT THE NOISY IMAGE IS NOT WEIGHTED WHEN COMPUTING THE FINAL DENOISED RESULT.

	2-D Denoising	3-D Denoising
Block size	8×8	$8 \times 8 \times 8$
Dictionary size	64×100	512×1000
Atom sparsity (Sparse K-SVD)	6	16
Initial dictionary	Overcomplete DCT	Overcomplete DCT
Training signals	30,000	80,000
K-SVD iterations	15	15
Noise gain	1.15	1.04
Lagrange multiplier	0	0
Step size	1	2

tionaries are truly advantageous.

Turning to the 3-D denoising results, we find that the Sparse K-SVD matches or outperforms the standard K-SVD in all test cases. Indeed, in the low noise range ($\sigma \leq 10$), both methods perform essentially the same, and provide only marginal improvement over the fixed overcomplete DCT dictionary. However in the medium and high noise ranges ($\sigma \geq 20$), the training process becomes beneficial, and leads to improved recovery compared to the fixed dictionary. In this noise range, the increased stability of the Sparse K-SVD in the presence of noise and limited training data becomes advantageous, and it performs consistently better than standard K-SVD. We note that in some cases of very high noise, the standard K-SVD actually performs *worse* than its initial overcomplete DCT dictionary, due to overfitting and its weakness in the presence of noise.

Reviewing the results in Table I, we note that the raw PSNR gain of Sparse K-SVD over standard K-SVD, while consistent, is typically small. Indeed, the main appeal of the Sparse K-SVD here is its substantially better complexity, as depicted in Table II. As can be seen, the complexity advantage of Sparse K-SVD translates to a $\times 10 - \times 20$ reduction in denoising time compared to the standard K-SVD, and in fact, the long running time of standard K-SVD makes it practically useless for this task. In contrast, the Sparse K-SVD is much faster, performing especially reasonably in the interesting noise range of $\sigma \geq 20$ (in the next section we discuss methods to further reduce running time in practical applications). Thus, we conclude that the Sparse K-SVD is indeed able to introduce adaptivity where the standard K-SVD is impractical, making sparse dictionaries an appealing alternative to both fixed dictionaries and explicit learned dictionaries alike.

C. Further Acceleration and Practical Considerations

The running times in Table II may be significantly improved to allow incorporation of the Sparse K-SVD in practical applications. First, analysis of the Sparse K-SVD denoising run-time shows that it is mostly dedicated to training, while the actual denoising requires similar time to the overcomplete DCT option. In many cases, training time may be decreased (and denoising results improved) by pre-training an initial sparse dictionary on a large set of generic data of the same type as handled by the application. This method, employed e.g. in [11], reduces the number of training iterations required, and can substantially accelerate the process.

Another source of acceleration is replacing the OMP-Cholesky implementation with a more efficient OMP implementation such as Batch-OMP [37]. This option, which is not discussed here due to its relative technicality, is analyzed in detail in [43]. Experiments done with Batch-OMP show that it achieves a $\times 2 - \times 3$ speedup in Sparse K-SVD and overcomplete DCT denoising over the running times shown in Table II, reducing the Sparse K-SVD denoising time to less than 5 minutes for the $\sigma \geq 20$ noise range. The software package published with this paper (see below) implements both OMP-Cholesky and Batch-OMP options.

Finally, we should mention that all algorithms discussed here are highly parallelizeable, with an expected near-linear speedup with the number of processors. Thus we expect an 8-core processor, combined with the Batch-OMP implementation, to carry out the entire 3-D Sparse K-SVD denoising process in less than a minute for any $\sigma \geq 20$.

D. Reproducible Research

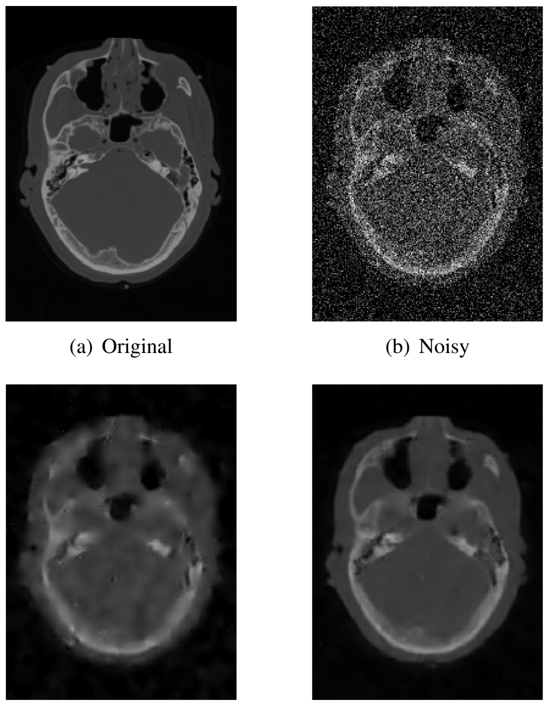
The complete K-SVD and Sparse K-SVD code reproducing the results in this paper, along with the original CT volumes used, are made available for download at http://www.cs.technion.ac.il/~ronrubin/software.html. The code is provided as a set of Matlab packages that combine Matlab code and compilable C MEX functions. The packages implement both the OMP-Cholesky and the Batch-OMP options. See the `README` files and the accompanying documentation in each of the packages for more information.

VI. Summary and Future Work

We have presented a novel dictionary structure which is both adaptive and efficient. The sparse structure is simple and can be easily integrated into existing sparsity-based methods. It provides fast forward and adjoint operators, enabling its use with larger dictionaries and higher-dimensional data. Its compact form is beneficial for tasks such as compression, communication, and real-time systems. It may be combined with any implicit dictionary to enhance its adaptability, with very little overhead.

We developed an efficient K-SVD-like algorithm for training the sparse dictionary, and showed that the structure provides better generalization abilities than the non-constrained one. The algorithm was applied to noisy CT data, where the sparse structure was found to outperform and operate significantly faster than the explicit representation under moderate and high noise. The proposed dictionary structure is thus a compelling alternative to existing explicit and implicit dictionaries alike, offering the benefits of both.

The full potential of the new dictionary structure is yet to be realized. We have provided preliminary results for CT denoising, however other signal processing tasks are expected to benefit from the new structure as well, and additional work is required to establish these gains. As noted in the introduction, the generality of the sparse dictionary structure allows it to be easily combined with other dictionary forms. As dictionary design receives increasing attention, the proposed structure can become a valuable tool for accelerating, regularizing, and enhancing adaptability in future dictionary structures.


(a) Original (b) Noisy

(c) 2-D Sparse KSVD (d) 3-D Sparse KSVD

Fig. 7. Denoising results for *Visible Male - Head*, slice #137 ($\sigma = 50$). Images are mainly provided for qualitative evaluation, and are best viewed by zooming-in using a computer display.

References

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

[2] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, 1993.

[3] 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.

[4] M. Elad, J. L. Starck, P. Querre, and D. L. Donoho, "Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)," *Applied and Computational Harmonic Analysis*, vol. 19, pp. 340–358, 2005.

[5] 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.

[6] 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.

[7] M. Protter and M. Elad, "Image sequence denoising via sparse and redundant representations," *IEEE Trans. Image Process.*, 2008. To appear.

[8] O. Bryt and M. Elad, "Compression of facial images using the K-SVD algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 270–283, 2008.

[9] 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.

[10] H. Y. Liao and G. Sapiro, "Sparse representations for limited data tomography," in *5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008*, pp. 1375–1378, 2008.

[11] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2009.

[12] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Constructive Approximation*, vol. 13, no. 1, pp. 57–98, 1997.

[13] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," *1993 Conference Record of The 27th Asilomar Conference on Signals, Systems and Computers*, pp. 40–44, 1993.

[14] D. L. Donoho and M. Elad, "Optimal sparse representation in general (nonorthogonal) dictionaries via L_1 minimization," *Proceedings of the National Academy of Sciences*, vol. 100, pp. 2197–2202, 2003.

[15] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm," *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 600–616, 1997.

[16] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," Submitted.

[17] S. Sardy, A. G. Bruce, and P. Tseng, "Block coordinate relaxation methods for nonparametric wavelet denoising," *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 361–379, 2000.

[18] M. Elad, “Why simple shrinkage is still relevant for redundant representations?,” *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5559–5569, 2006.

[19] M. Elad, B. Matalon, and M. Zibulevsky, “Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization,” *Applied and Computational Harmonic Analysis*, vol. 23, no. 3, pp. 346–367, 2007.

[20] K. Schnass and P. Vandergheynst, “Dictionary preconditioning for greedy algorithms,” *IEEE Trans. Signal Process.*, 2007.

[21] T. Blumensath and M. Davies, “Gradient pursuits,” *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2370–2382, 2008.

[22] S. Mallat, *A wavelet tour of signal processing*. Academic Press, 1999.

[23] E. J. Candès and D. L. Donoho, “Curvelets – a surprisingly effective nonadaptive representation for objects with edges,” *Curves and Surfaces*, 1999.

[24] 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.

[25] D. Labate, W. Lim, G. Kutyniok, and G. Weiss, “Sparse multidimensional representation using shearlets,” in *Proc. SPIE: Wavelets XI*, vol. 5914, pp. 254–262, 2005.

[26] I. W. Selesnick, R. G. Baraniuk, and N. C. Kingsbury, “The dual-tree complex wavelet transform,” *IEEE Signal Process Mag.*, vol. 22, no. 6, pp. 123–151, 2005.

[27] E. LePennec and S. Mallat, “Sparse geometric image representations with bandelets,” *IEEE Trans. Image Process.*, vol. 14, no. 4, pp. 423–438, 2005.

[28] R. Vidal, Y. Ma, and S. Sastry, “Generalized principal component analysis (GPCA),” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1945–1959, 2005.

[29] K. Engan, S. O. Aase, and J. Hakon Husoy, “Method of optimal directions for frame design,” *Proceedings ICASSP’99 – IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 2443–2446, 1999.

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

[31] R. R. Coifman and M. V. Wickerhauser, “Entropy-based algorithms for best basis selection,” *IEEE Trans. Inf. Theory*, vol. 38, no. 2(2), pp. 713–718, 1992.

[32] H. H. Szu, B. A. Telfer, and S. L. Kadambe, “Neural network adaptive wavelets for signal representation and classification,” *Optical Engineering*, vol. 31, p. 1907, 1992.

[33] W. J. Jasper, S. J. Garnier, and H. Potlapalli, “Texture characterization and defect detection using adaptive wavelets,” *Optical Engineering*, vol. 35, p. 3140, 1996.

[34] M. Nielsen, E. N. Kamavuako, M. M. Andersen, M. F. Lucas, and D. Farina, “Optimal wavelets for biomedical signal compression,” *Medical and Biological Engineering and Computing*, vol. 44, no. 7, pp. 561–568, 2006.

[35] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya, “Learning unions of orthonormal bases with thresholded singular value decomposition,” *Proceedings ICASSP’05 – IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 293–296, 2005.

[36] 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.

[37] R. Rubinstein, M. Zibulevsky, and M. Elad, “Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit,” *Technical Report – CS Technion*, 2008.

[38] J. C. Bezdek and R. J. Hathaway, “Some notes on alternating optimization,” *Lecture Notes in Computer Science*, vol. 2275, pp. 187–195, 2002.

[39] A. J. Smola and B. Scholkopf, “Sparse greedy matrix approximation for machine learning,” *Proceedings of the 17th International Conference on Machine Learning*, pp. 911–918, 2000.

[40] R. A. Horn and C. R. Johnson, *Topics in matrix analysis*. Cambridge University Press, 1991.

[41] E. J. Im, *Optimizing the Performance of Sparse Matrix-Vector Multiplication*. PhD thesis, University of California, 2000.

[42] S. F. Cotter, R. Adler, R. D. Rao, and K. Kreutz-Delgado, “Forward sequential algorithms for best basis selection,” *IEEE Proceedings – Vision, Image and Signal Processing*, vol. 146, no. 5, pp. 235–244, 1999.

[43] R. Rubinstein, M. Zibulevsky, and M. Elad, “Accelerating sparse-coding techniques using sparse dictionaries,” *Technical Report – CS Technion*, 2009.

[44] M. Aharon, M. Elad, and A. M. Bruckstein, “On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them,” *Linear Algebra and Its Applications*, vol. 416, no. 1, pp. 48–67, 2006.