Learning Sparse Dictionaries for Sparse Signal Approximation

Ron Rubinstein*, Michael Zibulevsky* and Michael Elad*

Abstract

An efficient and flexible dictionary structure is proposed for sparse and redundant signal representation. The structure is based on a sparsity model of the dictionary atoms over a base dictionary. 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 report 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.

1 Introduction

Sparse representation of signals over redundant dictionaries [1, 2, 3] is a rapidly evolving field, with state-of-the-art results in many fundamental signal and image processing tasks [4, 5, 6, 7, 8, 9, 10]. 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:

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

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 [11], the above problem is relatively easy to *approximate* using a wide variety of techniques [12, 13, 14, 15, 16, 17, 18, 19, 20].

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 divided to one of two main

*Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000 Israel.

categories: *analytic-based* and *learning-based*. 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 algorithmically rather than as an explicit matrix. Dictionaries of this type include Wavelets [21], Curvelets [22], Contourlets [23], Complex Wavelets [24], Bandelets [25], and more.

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 [26], the Method of Optimal Directions (MOD) [27], the K-SVD [28], and others. Advantages of this approach are the much finer-tuned dictionaries they produce compared to analytical approaches, and the 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 implicit *base dictionary*. The new parametric structure leads to a flexible and compact dictionary representation which is both adaptive and efficient. Advantages of the new structure include stabilizing and accelerating sparsity-based techniques, handling larger and higher-dimensional signals, and offering a more compact dictionary representation.

1.1 Related Work

The idea of training dictionaries with a specific structure has been proposed in the past, though the research is still in its early stages. Most of the work so far has focused specifically on developing adaptive *Wavelet* transforms, as in [29, 30, 31, 32]. These works attempt to adapt various parameters of the Wavelet transform, such as the mother wavelet or the scale and dilation operators, to better suite specific given data.

More recently, an algorithm for training unions of orthonormal bases was proposed in [33]. The suggested dictionary structure takes the form

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

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) [16], and the training algorithm itself is simple and relatively efficient. On the down side, the dictionary model 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:

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

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 interesting option for future study.

Another recent contribution is the *signature dictionary* proposed in [34]. 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 (both fixed and variable-sized sub-blocks can be considered). 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. The proposed sparse dictionary model in this paper improves the dictionary expressiveness by increasing the number of parameters per atom from 1 to $k > 1$, while maintaining other favorable properties of the dictionary.

1.2 Paper Organization

This report is organized as follows. We begin in Section 2 with a description of the dictionary model and its advantages. In Section 3 we consider the task of training the dictionary from examples, and present an efficient algorithm for doing so. Section 4 analyzes and quantifies the complexity of sparse dictionaries, and compares this to other dictionary forms. Simulation results are provided in Section 5. We summarize and conclude in Section 6.

1.3 Notation

This report uses the following 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 $\mathbf{M}$.

2 Sparse Dictionaries

2.1 Motivation

In selecting a dictionary for sparse signal representation, two elementary and seemingly competing properties immediately come to mind. The first is the *complexity* of the dictionary, as multiplication by the dictionary and its adjoint constitute the dominant components in most sparse-coding techniques, and these in turn form the core components in all dictionary training and sparsity-based signal processing algorithms. Indeed, techniques such as Matching Pursuit (MP) [2], Orthogonal Matching Pursuit (OMP) [12], Stagewise Orthogonal Matching Pursuit (StOMP) [15], and their variants, all involve costly dictionary-signal computations each iteration. Other common methods such as interior-point Basis Pursuit [1] and FOCUSS [14] minimize a quadratic function each iteration, which is commonly performed iteratively 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 [21], Curvelets [22], Contourlets [23], and Shearlets [35], 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, 8, 10, 6]. 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.[1]

[1]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 [29] and Bandelets [25], 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 (frequency content or geometry, respectively).

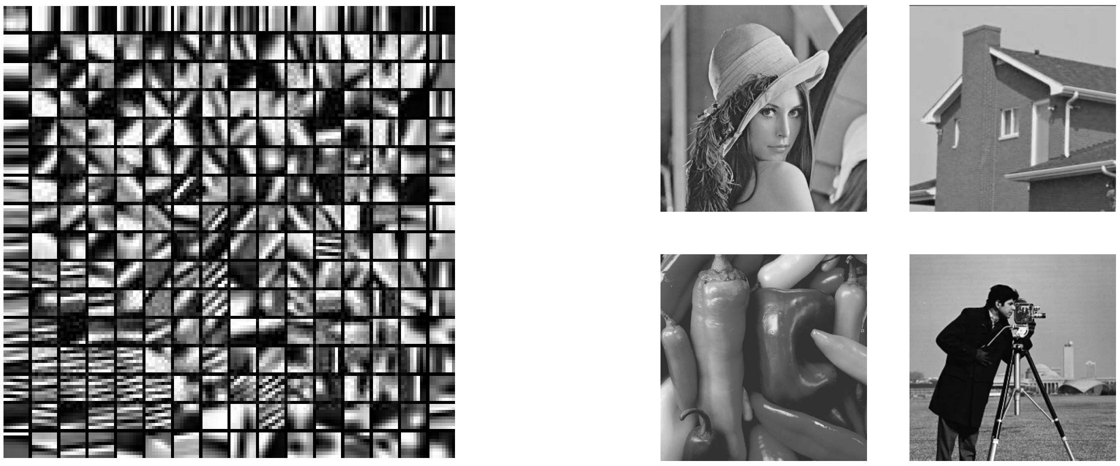


Figure 1: Left: dictionary for 8×8 image patches, trained using the K-SVD algorithm. Right: images used for the training. Each image contributed 25,000 randomly selected patches, for a total of 100,000 training signals.

Bridging this gap between complexity and adaptivity requires a parametric dictionary model having an inner structure with enough 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 we see an example of a dictionary trained using the K-SVD algorithm [28] for 8×8 natural image patches. The algorithm trains explicit, fully un-constrained dictionary matrices, and yet, the resulting dictionary appears highly structured, and its atoms have noticeable regularity. This gives rise to the hypothesis that the dictionary atoms *themselves* may have some underlying sparse structure over a more fundamental dictionary. As we will see, such a structure can indeed be recovered, and has several favorable properties.

2.2 Dictionary Model

The sparse dictionary model suggests that each of the atom signals in 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.

In comparison to implicit dictionaries, the dictionary model (4) is *adaptive* 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, which adds them a new layer of adaptability.

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 report, 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 a signal $\mathbf{x}$ is sparsely represented over the dictionary $\mathbf{D} = \mathbf{\Phi A}$, so $\mathbf{x} = \mathbf{\Phi A}\gamma$ 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 way beyond the success range of the 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.[2]

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.

3 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 $\mathbf{D} = \mathbf{\Phi A} \approx \mathbf{D}_0$. 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.

[2]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.

A more desirable approach would be to learn the sparse dictionary using a process that is aware of the dictionary's structure. We adopt an approach which continues the line of work in [28], 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 [36], which we briefly review.

3.1 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

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

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 ones fixed.

The atom update step 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

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

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) is computationally demanding, especially when the number of training signals is large. As an alternative, an approximate solution can be used to reduce the complexity of this task [36]. The simplified update step is obtained by applying a single iteration of alternate-optimization, 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,[3] and when truncated, supplies an approximation which still reduces the penalty term. Also, this process eliminates the need to explicitly compute $\mathbf{E}$, which is both time and memory consuming, and instead only requires its products with vectors.

3.2 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

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

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 $\mathbf{d} = \mathbf{\Phi a}$ with $\|\mathbf{a}\|_0^0 \le p$. The modified atom update is therefore given by

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

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 [37]. The SMA problem is formulated similar to problem (10), but replaces the rank-1 matrix $\mathbf{ag}^T$ 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 [37], 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 alternate-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 enough for our purposes.

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

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

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

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

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

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

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

The standard approach to this problem is to rewrite $\mathbf{E}$ as a column vector $\mathbf{e}$, and formulate the problem as an ordinary sparse-coding task for $\mathbf{e}$ (we use $\otimes$ to denote the Kronecker matrix product [38]):

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

This, however, leads to an intolerably large optimization problem, as the length of the signal to be sparse-coded 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

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

Here, the vector $\mathbf{Eg}$ 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. Furthermore, the vector $\mathbf{Eg}$ itself is faster to compute than the vector $\mathbf{e}$ in (14), as it can be computed using simple matrix-vector multiplications without having to explicitly compute the full matrix $\mathbf{E}$.

We establish the equivalence between the two problems using the following Lemma:

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

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

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

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

□

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

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

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 $\mathbf{g}$.

Combining the pieces, the final atom update process consists of (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.

4 Complexity of Sparse Dictionaries

Sparse dictionaries are generally much more efficient than explicit ones. However, the gains become more substantial for larger dictionaries and higher-dimensional signals. In this section we analyze and quantify the complexity of sparse dictionaries, and compare them to other dictionary forms. We consider specifically the case of Orthogonal Matching Pursuit (OMP), which is a widely used method which is relatively simple to analyze.

```
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 $\mathbf{X} \approx \mathbf{\Phi A \Gamma}$
3: Init: Set $\mathbf{A} := \mathbf{A}_0$
4: for $n = 1 \dots k$ do
5:    $\forall i : \mathbf{\Gamma}_i := \operatorname{Argmin}_{\gamma} \|\mathbf{x}_i - \mathbf{\Phi A}\gamma\|_2^2$ Subject To $\|\gamma\|_0^0 \le t$
6:    for $j = 1 \dots L$ do
7:       $\mathbf{A}_j := \mathbf{0}$
8:       $I := \{$indices of the signals in $\boldsymbol{X}$ whose reps. use $\boldsymbol{a}_j\}$
9:       $\mathbf{g} := \mathbf{\Gamma}_{j,I}^T$
10:      $\mathbf{g} := \mathbf{g}/\|\mathbf{g}\|_2$
11:      $\mathbf{z} := \mathbf{X}_I\mathbf{g} - \mathbf{\Phi A \Gamma}_I\mathbf{g}$
12:      $\mathbf{a} := \operatorname{Argmin}_{\mathbf{a}} \|\mathbf{z} - \mathbf{\Phi a}\|_2^2$ Subject To $\|\mathbf{a}\|_0^0 \le p$
13:      $\mathbf{a} := \mathbf{a}/\|\mathbf{\Phi a}\|_2$
14:      $\mathbf{A}_j := \mathbf{a}$
15:      $\mathbf{\Gamma}_{j,I} := (\mathbf{X}_I^T\mathbf{\Phi a} - (\mathbf{\Phi A \Gamma}_I)^T\mathbf{\Phi a})^T$
16:   end for
17: end for
```

Figure 2: The Sparse K-SVD Algorithm.

4.1 Complexity of OMP

The Orthogonal Matching Pursuit algorithm and its efficient implementation are reviewed in the Appendix. We focus on two OMP implementations — *OMP-Cholesky* and *Batch-OMP* [36]. The first is an implementation which uses progressive Cholesky decomposition to accelerate the orthogonal projection, and the second is an implementation designed for sparse-coding large sets of signals, and trades memory for speed by employing an initial precomputation step which substantially accelerates the subsequent sparse-coding tasks. In the following we quote the main complexity results, and leave additional details to the Appendix.

The implementation of OMP-Choleksy varies slightly depending on the dictionary representation — either explicit or implicit. Consequently, its complexity has two forms:

$$\begin{aligned} T_{omp}\{\textit{explicit-dict}\} &= tT_D + 2t^2N + 2t(L+N) + t^3 \\ T_{omp}\{\textit{implicit-dict}\} &= 4tT_D + 2t(L+N) + t^3 \end{aligned} \quad . \tag{16}$$

In these expressions, t is the number of OMP iterations (also the number of selected atoms), N and L are the dimensions of the dictionary, and T_D is the complexity of the dictionary operators ($T_D = 2NL$ in the explicit case).

The complexity of Batch-OMP is simpler and does not depend on the dictionary

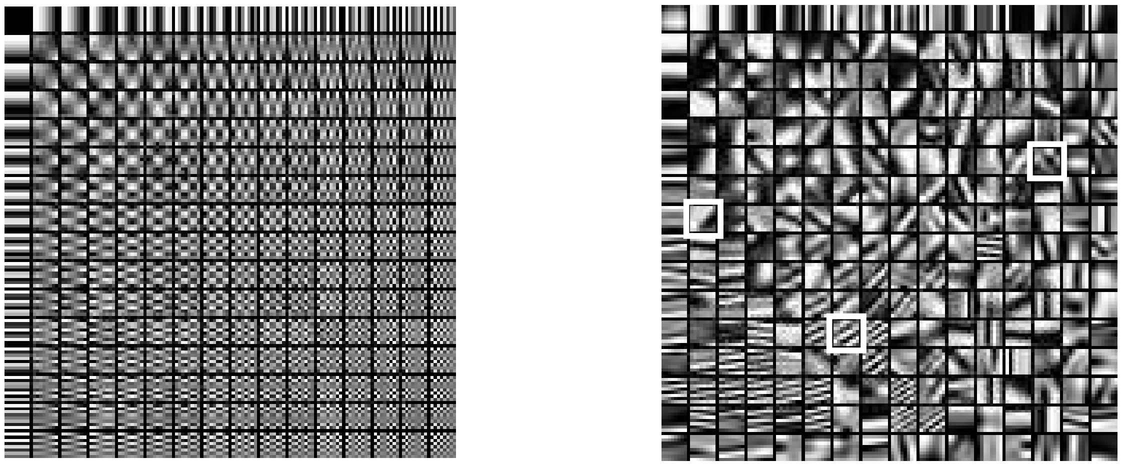


Figure 3: Left: overcomplete DCT dictionary for 8×8 image patches. Right: sparse dictionary trained over the overcomplete DCT using Sparse K-SVD. Dictionary atoms are represented using 6 coefficients each. Marked atoms are magnified in Fig. 4.

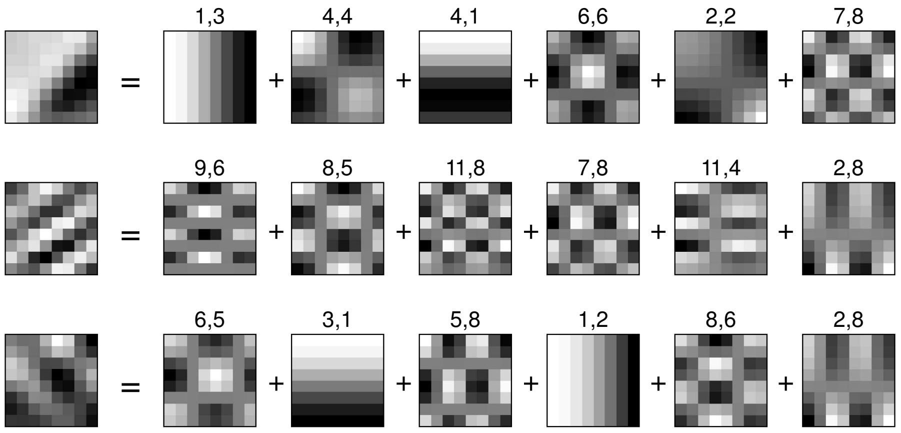


Figure 4: Some atoms from the trained dictionary in Fig. 3, and their overcomplete DCT components. The index pair above each overcomplete DCT atom denotes the wave number of the atom, with (1,1) corresponding to the upper-left atom, (16,1) corresponding to the lower-left atom, etc. In each row, the components 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.

implementation, as all dictionary-dependent computations are absorbed into the precomputation. The Batch-OMP complexity per-signal is given by:

$$T_{b-omp} = T_D + t^2 L + 3tL + t^3 \ . \tag{17}$$

The precomputation step involves computing the Gram matrix $\mathbf{G} = \mathbf{D}^T\mathbf{D}$. For an implicit dictionary, this will generally require $2L$ applications of the dictionary, or $T_G =$

$2LT_D$. For an explicit dictionary, we can exploit the symmetry of $\mathbf{G}$ to obtain $T_G = NL^2$.

4.2 Using Sparse Dictionaries

For the dictionary structure (4), the forward dictionary operator is implemented by multiplying the sparse representation by $\mathbf{A}$ and applying $\mathbf{\Phi}$; the adjoint operator is obtained by reversing and transposing the process.

In practice, operations involving sparse matrices introduce an overhead compared to the equivalent operations with explicit matrices. The performance of the sparse-matrix operations will vary depending on the architecture, data structures, and algorithms used (see [39] for some insights on the topic). In this paper, we make the simplifying assumption that the complexity of these operations is proportional to the number of non-zeros in the sparse matrix. Specifically, we consider multiplying a vector by a sparse matrix with Z non-zeros as 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. Thus, the complexity of the sparse dictionary operators is

$$T_D = 2\alpha pL + T_\Phi \ , \tag{18}$$

where pL is the number of non-zeros in $\mathbf{A}$, and T_Φ is the complexity of the base dictionary.

OMP-Cholesky with a sparse dictionary is implemented slightly differently than with either an explicit or implicit dictionary. We derive the complexity of this implementation in the Appendix, where we arrive at

$$T_{omp}\left\{\textit{sparse-dict}\right\} = 4tT_\Phi + 2\alpha tpL + 2t(L+N) + t^3 \ . \tag{19}$$

As to Batch-OMP, deriving its complexity is straightforward as (18) can be readily substituted in (17), leading to

$$T_{b-omp}\left\{\textit{sparse-dict}\right\} = 2\alpha pL + T_\Phi + t^2L + 3tL + t^3 \ . \tag{20}$$

The precomputation step, where the dictionary implementation has more influence, will generally be implemented using a series of applications of the base dictionary and its adjoint. Taking into account the symmetry of $\mathbf{G}$, this sums up to

$$T_G\left\{\textit{sparse-dict}\right\} = 2LT_\Phi + \alpha pL^2 \ . \tag{21}$$

In specific cases, this process can be further accelerated. For instance, if $\mathbf{G}_\Phi = \mathbf{\Phi}^T\mathbf{\Phi}$ is assumed to be known (a reasonable assumption when $\mathbf{\Phi}$ is some standard transform), the computation of $\mathbf{G}$ reduces to $\mathbf{G} = \mathbf{A}^T\mathbf{G}_\Phi\mathbf{A}$.

Table 1 summarizes the complexities of the various OMP configurations.

4.3 Some Example Base Dictionaries

The base dictionary $\mathbf{\Phi}$ will usually be chosen to have a compact representation and sub-N^2 implementation, which characterizes most implicit dictionaries. For concrete quantitative results, we consider specifically the following types of base dictionaries:

Separable dictionaries: Dictionaries which are the Kronecker product of several 1-dimensional dictionaries. When the signal $\mathbf{x} \in \mathbb{R}^{N_1 \cdot N_2 \cdots N_d}$ represents a d-dimensional patch of size $N_1 \times N_2 \times \ldots \times N_d$, sorted in column-major order, a dictionary can be constructed for this signal by combining d separate 1-dimensional dictionaries $\mathbf{\Phi}^i \in \mathbb{R}^{N_i \times M_i}$, where each $\mathbf{\Phi}^i$ is designed to represent the signal's behavior along its i-th dimension. The combined dictionary is given by $\mathbf{\Phi} = \mathbf{\Phi}^d \otimes \mathbf{\Phi}^{d-1} \cdots \otimes \mathbf{\Phi}^1 \in \mathbb{R}^{N \times M}$, with $N = \prod_i N_i$ and $M = \prod_i M_i$. The dictionary adjoint is given by $\mathbf{D}^T = (\mathbf{\Phi}^d)^T \otimes (\mathbf{\Phi}^{d-1})^T \otimes \cdots \otimes (\mathbf{\Phi}^1)^T$. The forward and adjoint operators can be efficiently implemented by applying each of the 1-D transforms $\mathbf{\Phi}^i$ or $(\mathbf{\Phi}^i)^T$ (respectively) along the i-th dimension, in any order. Assuming the dictionaries are applied in order (from 1 to d), and denoting $a_i = M_i/N_i$, the complexity of this dictionary is given by

$$T_\Phi = 2N\big(M_1 + a_1 M_2 + a_1 a_2 M_3 + \ldots + (a_1 a_2 \cdots a_{d-1}) M_d\big) \ . \tag{22}$$

Note that in the above we assumed that the 1-D dictionaries are applied via explicit matrix multiplications, and complexity may further decrease if the $\mathbf{\Phi}^i$'s have efficient implementations. The separable dictionary model generalizes many common dictionaries such as the DCT (Fourier), overcomplete DCT (Fourier), and Wavelet dictionaries, among others.

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

$$T_\Phi = \beta N \ . \tag{23}$$

for some constant value β. Indeed these are the most efficient dictionaries possible, though they are restricted to a linear number of atoms in N. Examples include the Wavelet, Contourlet, and Complex Wavelet dictionaries, among others.

Other types of dictionaries could also be considered. We restrict the discussion to these two dictionary families as they roughly represent two extremes — a near-N^2 polynomial dictionary and a linear dictionary. Many other dictionaries used in practice, such as the Curvelet and Undecimated Wavelet dictionaries, fall in-between the two, with complexities of around $\Theta(N \log N)$.

4.4 Quantitative Comparison and Analysis

Table 2 lists some operation counts of OMP with explicit and sparse dictionaries, for a few sizes of signal sets. For each set, the fastest method is designated in bold numerals. Note that the complexity of OMP-Cholesky scales linearly with the number of the signals, whereas Batch-OMP gains its efficiency as the number of signals increases.

Dictionary Type	OMP-Cholesky (Per Signal)		Batch-OMP Per Signal		Batch-OMP Precomputation
Implicit	$4tT_D$	$+\,2t(L+N)+t^3$	T_D	$+\,t^2L+3tL+t^3$	$2LT_D$
Explicit	$2tNL+2t^2N$	$+\,2t(L+N)+t^3$	$2NL$	$+\,t^2L+3tL+t^3$	NL^2
Sparse	$4tT_\Phi+2\alpha tpL$	$+\,2t(L+N)+t^3$	$T_\Phi+2\alpha pL$	$+\,t^2L+3tL+t^3$	$2LT_\Phi+\alpha pL^2$

Dictionary: $N \times L$	Base dictionary: $N \times M$
# of iterations: t	Atom sparsity (sparse dictionary): p

Table 1: Complexity of OMP for implicit, explicit and sparse dictionaries.

As can be seen in the Table, sparse dictionaries provide a pronounced performance increase compared to explicit ones for any number of signals. The largest gains are achieved for high-dimensional signals, where the complexity gap between explicit and implicit dictionaries is more significant. For large sets of signals (rightmost columns in these tables) we see that the operation counts may increase to the point where sparse dictionaries become the only feasible option.

A graphical comparison of these complexities is provided in Fig. 5. The Figure depicts the operation counts for coding $100,000$ signals using the two OMP variants and all dictionary types (explicit, implicit and sparse). We see that for Batch-OMP (dark bars), the significant complexity gap is between explicit and non-explicit dictionaries, with the non-explicit dictionaries all having very close complexities; in this case sparse dictionaries clearly offer a more flexible and useful structure than the other options, imposing only a small computational toll. Moving to OMP-Cholesky (light bars), we see that the complexity difference between the various dictionary options here is larger, however sparse dictionaries still provide the best balance between efficiency and adaptability. Indeed, they are significantly faster than the explicit option, and more flexible than the implicit options. We also see that the complexity advantage of sparse dictionaries is much more pronounced for larger dictionaries and higher-dimensional signals.

4.5 Dictionary Training

Unsurprisingly, the Sparse K-SVD algorithm is much faster than the standard and approximate K-SVD implementations. The gain stems mostly from the acceleration in the sparse-coding step (line 5 in the algorithm). In the case of Batch-OMP, the precomputation step is also accelerated as it can be done by precomputing $\mathbf{G}_\Phi = \mathbf{\Phi}^T\mathbf{\Phi}$ prior to the training ($\mathbf{\Phi}$ is fixed through the training process).

In the asymptotic case $t \sim p \ll M \sim L \sim N \ll R$, with R the number of training signals, the complexity of the approximate K-SVD can be shown to be proportional to the complexity of its sparse-coding method [36]. This result is easily extended to the Sparse K-SVD case, and consequently we find that the 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 the Sparse

	# of Signals:	1	10	10^2	10^3	10^4	10^5
	Exp	1.6	16.0	160.1	1,600.7	16,007.3	160,072.8
OMP-Chol	Sep	0.9 (x1.9)	8.5 (x1.9)	85.3 (x1.9)	852.7 (x1.9)	8,527.3 (x1.9)	85,272.6 (x1.9)
	Lin	0.3 (x5.3)	3.0 (x5.3)	30.4 (x5.3)	304.3 (x5.3)	3,043.4 (x5.3)	30,434.4 (x5.3)
	Exp	67.4	70.0	96.1	357.1	2,967.2	29,067.9
B-OMP	Sep	36.1 (x1.9)	36.8 (x1.9)	44.4 (x2.2)	120.5 (x2.0)	881.0 (x3.4)	8,486.5 (x3.4)
	Lin	10.0 (x6.7)	10.6 (x6.6)	15.9 (x6.1)	69.1 (x5.2)	600.9 (x4.9)	5,919.6 (x4.9)

Signal: 16×16	Dictionary: 256×512	Base dictionary: 256×512
# of atoms: $t = 6$	Atom sparsity: $p = 4$	$\alpha = 7,\ \beta = 10$

	# of Signals:	1	10	10^2	10^3	10^4
	Exp	143.0	1,439.5	14,395.1	143,951.0	1,439,510.4
OMP-Chol	Sep	14.4 (x9.0)	144.2 (x9.0)	1,442.0 (x9.0)	14,420.4 (x9.0)	144,203.6 (x9.0)
	Lin	6.4 (x22.4)	64.3 (x22.4)	642.0 (x22.4)	6,429.9 (x22.4)	64,298.9 (x22.4)
	Exp	20,651.7	20,764.8	21,895.9	33,206.9	146,316.6
B-OMP	Sep	2,059.6 (x10.0)	2,070.5 (x10.0)	2,179.6 (x10.0)	3,270.3 (x10.2)	14,177.4 (x10.3)
	Lin	789.3 (x26.2)	798.6 (x26.0)	891.1 (x24.6)	1,816.5 (x18.3)	11,069.9 (x13.2)

Signal: $12 \times 12 \times 12$	Dictionary: 1728×3456	Base dictionary: 1728×3456
# of atoms: $t = 12$	Atom sparsity: $p = 8$	$\alpha = 7,\ \beta = 10$

Table 2: Operation counts (in millions of operations) of OMP-Cholesky and Batch-OMP for explicit, sparse+separable and sparse+linear dictionaries (denoted *Exp*, *Sep*, and *Lin*, respectively). Values in parentheses are the speedups relative to an explicit dictionary. Bold numerals designate the fastest result for each set size. Separable dictionaries assumes $N_1 = \ldots = N_d = M_1 = \ldots = M_d$.

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.

5 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, 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 — make 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,

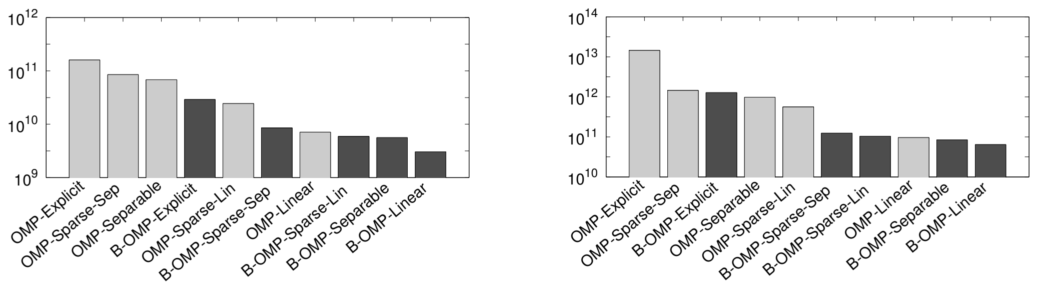


Figure 5: Operation counts of OMP-Cholesky and Batch-OMP for coding 10^5 signals, using either explicit, sparse+linear, sparse+separable, implicit linear, or implicit separable dictionaries. Bars are shown on a logarithmic scale. The parameters of the plots are identical to those in Table 2. Left: operation counts for signals of size 16×16. Right: operation counts for signals of size $12 \times 12 \times 12$. Dark bars correspond to Batch-OMP implementations.

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 readily 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 we are currently researching this option further.

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 produces superior results to the explicit one, especially in the higher noise range. Our simulations make use of the CT data provided by the NIH *Visible Human Project*[4].

5.1 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 [40]. 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

[4]`http://www.nlm.nih.gov/research/visible/visible_human.html`

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[5]. 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 evolution of the K-SVD and Sparse K-SVD performance on the training and validation sets (respectively) during the algorithm iterations. Following [5], we code the noisy training 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 them with a fixed number of atoms, and measuring the resulting representation RMSE.

We can see that 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 sparsest ($p = 8$) dictionary begins to slightly lag behind, 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.

5.2 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

[5]A 1-dimensional $N \times L$ overcomplete DCT dictionary is essentially a cropped version of the orthogonal $L \times L$ DCT dictionary matrix. A k-D overcomplete DCT dictionary is obtained as the Kronecker product of k 1-D overcomplete DCT dictionaries. Note that the number of atoms in the resulting dictionary is L^k and has an integer k-th root (in our case, $10^3 = 1000$ atoms).

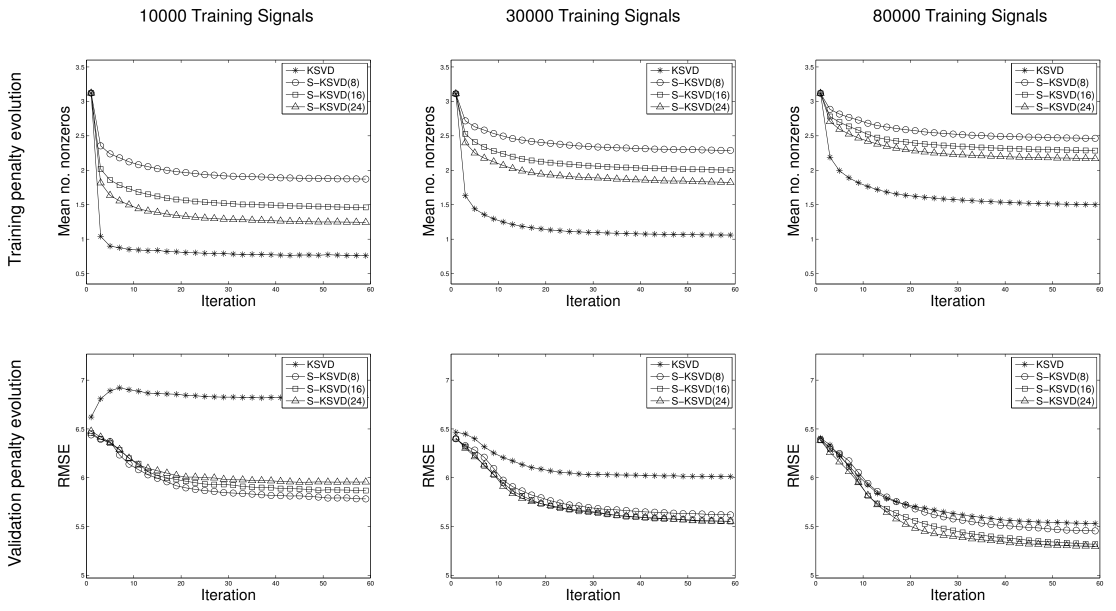


Figure 6: Training and validation results for patches from *Visible Male - Head*. Training signals are taken from the noisy volume (PSNR=17dB), and validation signals are taken from the original volume. Block size is $8\times8\times8$, and dictionary size is 512×1000. Training signals (noisy) are sparse-coded 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).

noisy signal, and then denoises the signal using this dictionary, averaging denoised blocks when they overlap in the result.

We performed the 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 comparability 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 $p = 6$ in the 2-D case and $p = 16$ in 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 the point where overfitting becomes a factor. However, we preferred to limit the atom sparsity in these experiments in order to maintain the complexity advantage. Further work may establish a more systematic way of selecting these values.

Our results are summarized in Table 3. For convenience, Table 4 lists the complete set of parameters used in these experiments. As can be seen, in the low noise range ($\sigma \leq 20$) the two methods perform essentially the same. However, in the medium and high noise ranges ($\sigma \geq 30$) the sparse dictionary clearly surpasses the explicit one, with some cases where the standard K-SVD actually performs *worse*, or just negligibly better, than the

Test	σ / PSNR	2-D Denoising			3-D Denoising		
		ODCT	KSVD	S-KSVD	ODCT	KSVD	S-KSVD
Vis. M.	5 / 34.15	43.61	43.94	43.72	45.10	45.12	**45.16**
Head	10 / 28.13	39.34	40.13	39.70	41.43	**41.55**	41.54
	20 / 22.11	34.97	36.08	35.81	37.67	38.01	**38.11**
	30 / 18.59	32.48	33.13	33.08	35.34	35.80	**36.12**
	50 / 14.15	29.62	29.67	29.74	32.56	32.85	**33.38**
	75 / 10.63	27.84	27.75	27.82	30.44	30.43	**30.74**
	100 / 8.13	26.51	26.40	26.48	29.40	29.27	**29.46**
Vis. F.	5 / 34.15	43.07	43.23	43.15	44.38	**44.61**	44.59
Ankle	10 / 28.13	39.25	39.70	39.45	40.87	**41.21**	41.18
	20 / 22.11	35.34	36.12	35.87	37.52	37.99	**38.01**
	30 / 18.59	33.01	33.76	33.67	35.47	35.94	**36.11**
	50 / 14.15	30.15	30.43	30.48	32.86	33.40	**33.77**
	75 / 10.63	27.88	27.84	27.92	30.98	31.44	**31.84**
	100 / 8.13	26.42	26.31	26.39	29.65	29.82	**30.18**

Table 3: CT denoising results using K-SVD, Sparse K-SVD, and overcomplete DCT. Bold numerals denote the best result in each test.

initial overcomplete DCT dictionary, due to the weakness of the K-SVD when used with insufficient training data and high noise. The sparse structure thus shows effective regularization in the presence of noise, while maintaining the required flexibility to rise above the performance of its base dictionary.

In conclusion, when considering the combination of complexity, stability and structure, sparse dictionaries form an appealing alternative to the existing dictionary forms in sparsity-based techniques. The pronounced difference between 2-D and 3-D denoising provides further motivation for the move towards bigger patches and dictionaries, where sparse dictionaries are truly advantageous. Some actual denoising results are shown in Fig. 7.

6 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 performance of

	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	1.05
Lagrange multiplier	0	0
Step size	1	2

Table 4: 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.

the sparse structure was found to exceed that of 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.

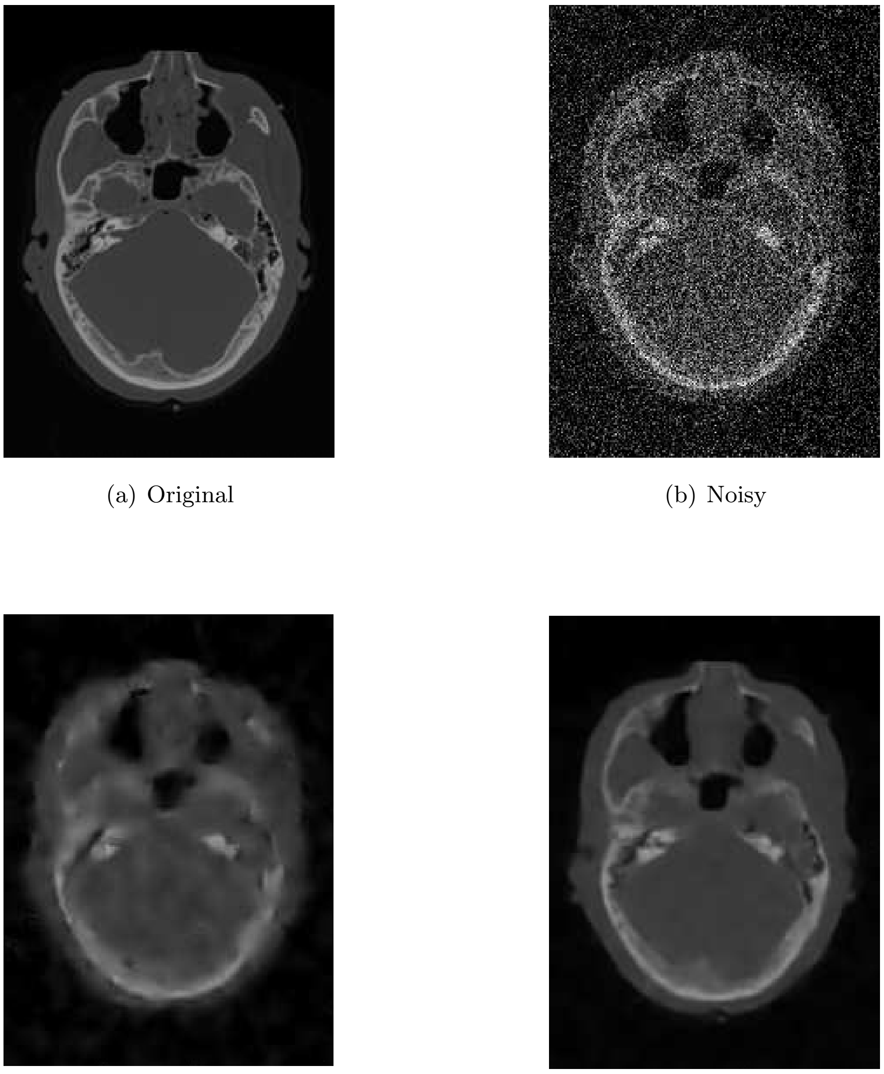
The full potential of the new dictionary structure is yet to be realized. We have provided preliminary results for CT denoising, however other signal processing tasks are expected to benefit from the new structure as well, and additional work is required to establish these gains. As noted in the introduction, 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.

Appendix

In this Appendix we briefly review the Orthogonal Matching Pursuit (OMP) algorithm, and discuss the OMP-Cholesky and Batch-OMP implementations along with their complexities.

The OMP algorithm aims to solve the sparse approximation problem (1), by taking a greedy approach. The algorithm begins with an empty set of atoms, and each iteration, selects the atom with highest correlation to the current residual, and adds it to the set. Once the atom is added, the signal is orthogonally projected to the span of the selected atoms, the residual is recomputed, and the process repeats with the new residual. The method terminates once the desired number of atoms is reached, or the predetermined error target is met.

The two computationally demanding processes in the OMP iteration are the computation of the atom-residual correlations $\mathbf{D}^T\mathbf{r}$, and the orthogonal projection of the signal on the span of the selected atoms. Beginning with the orthogonal projection task, we


(a) Original (b) Noisy

(c) 2-D Sparse K-SVD (d) 3-D Sparse K-SVD

Figure 7: Denoising results for *Visible Male - Head*, slice #137 ($\sigma = 50$).

let I denote the set of selected atoms, and thus write the projection as

$$\gamma_I \; = \; (\mathbf{D}_I)^{+}\mathbf{x} \; , \tag{24}$$

where γ_I are the non-zero coefficients in the sparse representation γ, and $\mathbf{D}_I$ are the selected atoms. The computation is efficiently implemented by employing a progressive matrix decomposition such as a Cholesky or QR decomposition [20, 36, 41]. The progressive Cholesky approach is obtained by writing

$$\gamma_I \; = \; (\mathbf{D}_I^T\mathbf{D}_I)^{-1}\mathbf{D}_I^T\mathbf{x} \; , \tag{25}$$

and using a progressive update of the Cholesky decomposition of $\mathbf{D}_I^T\mathbf{D}_I$ to perform the inversion.

The progressive Cholesky update involves two operations which assume direct access to the dictionary atoms. Namely, it requires the computation of $\mathbf{D}_I^T\mathbf{d}_j$ where $\mathbf{d}_j$ is the selected atom, and the computation of $\mathbf{D}_I\gamma_I$. For explicit dictionaries such operations are straightforward, and the complexity of the entire OMP-Cholesky algorithm is easily determined to be

$$T_{omp}\left\{\textit{explicit-dict}\right\} \;=\; tT_D + 2t(L+N) + t^3 + 2t^2N \; . \tag{26}$$

In this expression, the first three terms are common to all OMP-Cholesky implementations, whereas the final term arises from the two mentioned dictionary-specific computations.

For implicit dictionaries, where only *full* forward and adjoint operators are supported, these two computations involve *three* dictionary applications — one for computing the atom $\mathbf{d}_j$ and two for multiplying by $\mathbf{D}$ and $\mathbf{D}^T$. Thus, the complexity of OMP-Cholesky in this case becomes

$$T_{omp}\left\{\textit{implicit-dict}\right\} \;=\; 4tT_D + 2t(L+N) + t^3 \; . \tag{27}$$

For sparse dictionaries, operations with sub-dictionaries are straightforward as in the explicit case. Specifically, computing $\mathbf{D}_I\gamma_I = \mathbf{\Phi}\mathbf{A}_I\gamma_I$ at the n-th iteration requires multiplying by $\mathbf{A}_I$ (np non-zeros) and applying $\mathbf{\Phi}$; computing $\mathbf{D}_I^T\mathbf{d}_j = \mathbf{A}_I^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{a}_j$ requires multiplying by $\mathbf{A}_I^T$ plus two applications of $\mathbf{\Phi}$. Summing over all $n = 1 \ldots t$ iterations, we find that the OMP-Cholesky complexity with a sparse dictionary is

$$\begin{aligned} T_{omp}\left\{\textit{sparse-dict}\right\} &= \\ &= \; tT_D + 2t(L+N) + t^3 + 3tT_\Phi + 2\alpha t^2 p \\ &\approx \; 4tT_\Phi + 2\alpha tpL + 2t(L+N) + t^3 \; . \end{aligned} \tag{28}$$

Note that in the above we neglected $2\alpha t^2 p$ relative to $2\alpha tpL$ as we assume $t \ll L$.

In all these cases, the complexity of OMP-Cholesky is generally dominated by the dictionary operators, which emerge both in the correlation step and in the progressive Cholesky update. Batch-OMP [36] significantly reduces the complexity of these computations by adding an initial precomputation phase in which the Gram matrix $\mathbf{G} = \mathbf{D}^T\mathbf{D}$ is computed. Given this matrix, a quick update of the atom-residual correlations each iteration is possible, and the computation of $\mathbf{D}_I^T\mathbf{d}_j$ in the Cholesky update becomes unnecessary as well.

Performing the Batch-OMP precomputation requires a substantial amount of work and memory, and is only worthwhile when a large number of signals needs to be coded. The complexity of the precomputation is obviously dictionary-dependent, and expressions for a few types of dictionaries are mentioned in the text. Per-signal, Batch-OMP involves a single computation of $\mathbf{D}^T\mathbf{x}$, after which the cost of each iteration is small. The complexity of Batch-OMP per-signal can be determined to be

$$T_{b-omp} = T_D + t^2L + 3tL + t^3 \; , \tag{29}$$

and we note that this result is independent of the dictionary implementation, apart from the value of T_D.

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

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

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

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

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

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

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

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

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

[40] 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.

[41] 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.