Analysis SimCO Algorithms for Sparse Analysis Model Based Dictionary Learning

Jing Dong, Wenwu Wang, Wei Dai, Mark D. Plumbley, Zi-Fa Han, Jonathon Chambers

***Abstract*—In this paper, we consider the dictionary learning problem for the sparse analysis model. A novel algorithm is proposed by adapting the simultaneous codeword optimization (SimCO) algorithm, based on the sparse synthesis model, to the sparse analysis model. This algorithm assumes that the analysis dictionary contains unit ℓ_2-norm atoms and learns the dictionary by optimization on manifolds. This framework allows multiple dictionary atoms to be updated simultaneously in each iteration. However, similar to several existing analysis dictionary learning algorithms, dictionaries learned by the proposed algorithm may contain similar atoms, leading to a degenerate (coherent) dictionary. To address this problem, we also consider restricting the coherence of the learned dictionary and propose Incoherent Analysis SimCO by introducing an atom decorrelation step following the update of the dictionary. We demonstrate the competitive performance of the proposed algorithms using experiments with synthetic data and image denoising as compared with existing algorithms.**

***Index Terms*—Sparse representation, analysis model, SimCO, analysis dictionary learning.**

I. Introduction

MANY problems in signal processing can be regarded as inverse problems, for example, denoising [1], inpainting [2] and super-resolution [3]. These problems aim to reconstruct original signals from their observed measurements. Some prior knowledge or assumptions about the signals are required due to the lack of information or the presence of noise in the observations. One assumption that has attracted extensive attention in the past decade is that the signals to be restored are *sparse* in some domain. Two signal models to capture the sparse property of the signals have been proposed, namely, the *sparse synthesis model* [4] and *sparse analysis model* [5], [6]. More recently, the sparse analysis model has been extended to a more generalized model, referred to as the *sparsifying transform model* [7].

A. Sparse Synthesis Model

The most well-known model in sparse representation is the sparse synthesis model [4], [8], [9], [10]. This model assumes that a signal $\mathbf{y} \in \mathbb{R}^m$ can be linearly represented with some atoms (columns) of a *synthesis dictionary* $\mathbf{D} \in \mathbb{R}^{m \times d}$, where the dictionary is usually overcomplete with $d > m$. The number of atoms used to represent $\mathbf{y}$ is much smaller than the total number of atoms in the dictionary, which reflects the *sparsity* of the signal $\mathbf{y}$. Mathematically, this model can be written as $\mathbf{y} = \mathbf{D}\mathbf{a}$ with $\|\mathbf{a}\|_0 = s$, where the ℓ_0-norm $\|\cdot\|_0$ counts the number of non-zero elements of its argument, and $\mathbf{a}$ is the representation coefficient vector with s being its sparsity. The atoms corresponding to the non-zero elements of $\mathbf{a}$ are used to synthesize the signal $\mathbf{y}$ via their linear combination, which brings about the term "synthesis" in the name of this model.

One challenge related to this model is the *sparse coding problem* which aims to find the sparsest representation $\mathbf{a}$ of a given signal $\mathbf{y}$ with respect to a given dictionary $\mathbf{D}$. In order to tackle this problem, greedy algorithms have been proposed, such as matching pursuit (MP) [11], orthogonal matching pursuit (OMP) [12], stagewise orthogonal matching pursuit (StOMP) [13] and subspace pursuit (SP) [14], as well as relaxation methods such as basis pursuit (BP) [15] and focal underdetermined system solver (FOCUSS) [16].

A second challenge is to design or learn an appropriate dictionary $\mathbf{D}$ to represent a set of signals as sparsely as possible. Many analytical dictionaries have been developed, but dictionaries learned from a set of training signals have the potential to fit these signals better than the analytical dictionaries [4]. As a result, the *dictionary learning problem* for the sparse synthesis model has become one of the most popular topics in sparse representation. This problem aims to seek the dictionary $\mathbf{D}$ that leads to the best set of representations for a given set of training signals. Many algorithms have been proposed to address this problem, for example, method of optimal directions (MOD) [9], K-SVD [4] and SimCO [10]. These algorithms typically alternate iteratively between an update of the coefficients and an update of the dictionary. For the update of the coefficients, sparse coding algorithms are often used while keeping the dictionary fixed. The main difference between synthesis dictionary learning algorithms is the way in which the dictionary is updated.

B. Sparse Analysis Model and Sparsifying Transform Model

In contrast to the synthesis model, the sparse analysis model uses an *analysis dictionary* $\mathbf{\Omega} \in \mathbb{R}^{p \times m}$ with $p > m$ to "analyze" the signal $\mathbf{y} \in \mathbb{R}^m$. Specifically, it assumes that the product of $\mathbf{\Omega}$ and $\mathbf{y}$ is sparse, i.e. $\mathbf{x} = \mathbf{\Omega}\mathbf{y}$ with $\|\mathbf{x}\|_0 = p - l$, where $0 \leq l \leq p$ is the number of zeros in $\mathbf{x} \in \mathbb{R}^p$. The matrix $\mathbf{\Omega}$ is usually referred to as an *analysis dictionary* [17]

J. Dong, W. Wang and M. D. Plumbley are with the Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford GU2 7XH, U.K. (emails: {j.dong, w.wang, m.plumbley}@surrey.ac.uk).

W. Dai is with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K. (email: wei.dai1@imperial.ac.uk).

Z.-F. Han is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong. (email: zifahan@gmail.com).

J. Chambers is with the School of Electrical and Electronic Engineering, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K. (email: Jonathon.Chambers@newcastle.ac.uk).

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) Grant number EP/K014307/1 and the MOD University Defence Research Collaboration in Signal Processing.

or *analysis operator* [18], [19], with each row of $\mathbf{\Omega}$ being an *atom*. The vector $\mathbf{x}$ is the *analysis representation* of the signal $\mathbf{y}$ with respect to $\mathbf{\Omega}$. This model is also referred to as a *co-sparse analysis model*, and the number of zeros l is called the *co-sparsity* of the signal $\mathbf{y}$ with respect to $\mathbf{\Omega}$ [6]. Let $\Lambda = \{i : x_i = 0\}$ denote the index set of the rows in $\mathbf{\Omega}$ corresponding to the zero elements in $\mathbf{x}$ (thus, $\mathrm{card}(\Lambda) = l$) and let $\mathbf{\Omega}_\Lambda$ denote the sub-matrix of $\mathbf{\Omega}$ containing only the rows indexed by Λ. The set Λ is called the *co-support* of $\mathbf{y}$. For the analysis model, we have $\mathbf{\Omega}_\Lambda \mathbf{y} = \mathbf{0}$, meaning that the l atoms indexed by Λ are orthogonal to the signal $\mathbf{y}$. From the subspace point of view, $\mathbf{y}$ lies in the subspace which is orthogonal to the subspace spanned by the rows of $\mathbf{\Omega}_\Lambda$. Even though the description of the sparse analysis model may seem similar to its synthesis counterpart, these two models differ significantly if the dictionaries are overcomplete [5].

If the signal $\mathbf{y}$ is known, its analysis representation with respect to a given $\mathbf{\Omega}$ can be obtained via multiplying $\mathbf{y}$ by $\mathbf{\Omega}$. However, when the observed signal is contaminated by noise, the clean signal $\mathbf{y}$ has to be estimated first in order to get its analysis representation, which leads to the *analysis pursuit problem* [17]. Some algorithms like backward-greedy (BG) [17], optimized-backward-greedy (OBG) [17], and greedy analysis pursuit (GAP) [6] have been proposed to address this problem.

In a similar way to the dictionary in the synthesis model, the analysis dictionary $\mathbf{\Omega}$ also plays an important role in the analysis representation of the signal $\mathbf{y}$, and the dictionaries learned from a set of training signals show some advantages compared with pre-defined dictionaries [17]. In the past few years, the *analysis dictionary learning (ADL)* problem has begun to attract more attention [17], [18], [19]. In this paper, we focus on this ADL problem.

Recently, the so-called sparsifying transform model, which assumes that a signal can be approximately sparsified with an analysis transform operator, was introduced in [7]. This model can be regarded as a natural extension of the sparse analysis model. Learning a sparsifying transform has been investigated in [7], [20], [21]. These algorithms deal with the sparsification error in the transform domain rather than in the original signal domain as in the ADL algorithms [17], [18], by applying the transform operator to the training signals even if the signals contain noise. In the present paper, we intentionally ignore this subtle difference between these two formulations and regard the sparsifying transform learning algorithms as alternative solvers of the ADL problem, since the results of sparsifying transform learning can be regarded as dictionaries for the sparse analysis model.

C. Analysis Dictionary Learning

Several algorithms have been proposed for the ADL problem. The Analysis K-SVD algorithm [17] assumes that the training samples are noisy signals and minimizes the error between the training samples and the signals estimated using the learned dictionary. It applies the OBG [17] analysis pursuit algorithm to detect the co-support of each training signal with respect to an initial dictionary, and employs the singular value decomposition (SVD) to update the dictionary atoms one-by-one. After the update of all atoms, similar atoms, determined with inner-product of two atoms, are replaced by new randomly generated atoms. However, these new atoms cannot preserve the information of the atoms to be replaced because of the randomness. Besides, the computational complexity of Analysis K-SVD is quite high due to the involvement of the analysis pursuit problem [17].

The learning overcomplete sparsifying transforms (LOST) algorithm [20] minimizes the so-called sparsifying error which is defined in the transform or analysis domain rather than the original signal domain as in the formulation of Analysis K-SVD. As a result, the time consuming algorithm OBG is not used any more. Two penalty terms are added to the objective function of LOST to apply two constraints on the learned dictionary respectively, i.e. the full column rank constraint and the constraint on the correlation between the atoms. The coefficients of the penalty terms play an important role in the performance of LOST, but selecting proper coefficients is a practical challenge [20].

Transform K-SVD proposed recently in [21] combines the sparsifying error formulation of LOST with the dictionary update approach of Analysis K-SVD. This algorithm uses the same method as in Analysis K-SVD to avoid similar atoms, but overcomes its computational complexity issue with a formulation used in LOST. We have found that Transform K-SVD performs well in recovering a reference dictionary, but its denoising performance is relatively limited, as shown in our simulations (see Section VI-B later).

The analysis operator learning (AOL) algorithm reported in [18] addresses the ADL problem using a constrained optimization framework. In this algorithm, the ℓ_1-norm is used as the co-sparsity measurement. It restricts the dictionary to be a uniform normalized tight frame (UNTF) which is the intersection of uniform normalized (UN) frames manifold and tight frames (TF) manifold. The AOL algorithm learns a dictionary by combining an ℓ_1 optimization framework with projection of the dictionary onto the UNTF set. The algorithms (NL)AOL (noiseless AOL) and (NA)AOL (noise-aware AOL) have been developed for learning with noiseless and noisy training samples respectively. However, random dictionaries cannot be recovered by the AOL algorithms since the UNTF constraint limits the possible dictionaries to be learned (see Section VI later). We will see that the denoising performance of (NA)AOL is also limited when the noise level is relatively high.

The GeOmetric Analysis operator Learning (GOAL) algorithm [19] employs the ℓ_p-norm ($0 \leq p \leq 1$) as the co-sparsity measurement, which is different from the algorithms reviewed above. Similar to the LOST algorithm, the objective function of GOAL also incorporates two additional penalty terms to address the full rank and the correlation constraints, leading to the difficulty of setting the weights for the penalties. The conjugate gradient method on manifolds [22] is applied for optimization.

D. Contributions

In this work, we propose two new algorithms which can partly address the limitations of the ADL algorithms mentioned above. Firstly, we adapt the synthesis model based SimCO algorithm [10] to the analysis model and develop a new ADL algorithm which is referred to as the Analysis SimCO algorithm. In SimCO, the optimization method on manifolds is applied to update multiple dictionary atoms simultaneously, leading to a better performance compared with K-SVD where the atoms are updated one-by-one. Thus, we adapt the framework of SimCO to the ADL problem to enable the simultaneous update of multiple atoms via the optimization on manifolds. The preliminary results of this work have been presented in [23]. This dictionary update method is different from the methods used by the existing algorithms. Analysis K-SVD and Transform K-SVD only allow one atom to be updated in one iteration. In LOST, the dictionary is updated as a whole matrix by the standard conjugate gradient method. Compared with the AOL algorithms, the updated dictionary in our proposed method is more general without projection onto the UNTF set. Notice that the GOAL algorithm also employs an optimization method on manifolds, however, the objective function of our proposed algorithm is different from that of GOAL due to the different co-sparsity measure based on ℓ_0-norm, and the fewer penalty terms used. Besides, our proposed algorithm employs the gradient descent method on manifolds rather than the conjugate gradient method as in GOAL.

Secondly, we propose the Incoherent Analysis SimCO algorithm to avoid similar atoms appearing in the dictionaries learned by Analysis SimCO. In the Incoherent Analysis SimCO algorithm, a constraint restricting the correlations of two distinct atoms of the dictionary is considered and an atom decorrelation step is applied to enforce this constraint by rotating the highly-correlated atom pairs. In this way, the correlation of any two distinct atoms can be restricted to be below a given threshold explicitly. Compared with the methods used in existing ADL algorithms to avoid similar atoms, the decorrelation step applied in the Incoherent Analysis SimCO algorithm has some advantages. For example, this method avoids the coefficient selection problem of LOST since the constraint is tackled directly rather than applied as a penalty term of the objective function. Besides, the new atoms obtained by the decorrelation step are more likely to be closer to the atoms replaced than the atoms that are generated randomly in Analysis K-SVD and Transform K-SVD.

E. Notations

Bold capital letters are used to represent matrices. In particular, $\mathbf{I}$ denotes the identity matrix whose dimension can be decided from the context. The notation $\mathbf{X}_{i,:}$ is used to specify the ith row of the matrix $\mathbf{X}$ and $\mathbf{X}_{:,j}$ represents its jth column. Bold lowercase letters represent vectors. Scalars are either capital or lowercase letters. The norms $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the ℓ_2-norm and the Frobenius norm respectively. The notation $|\cdot|$ returns the absolute value of a scalar. The notation $\langle\cdot,\cdot\rangle$ is used to represent the canonical inner-product of two vectors.

F. Organization of the Paper

The remainder of the paper is organized as follows. In Section II the original SimCO algorithm is reviewed. In Section III we present our formulation for the ADL problem and the optimization framework. More details of our proposed Analysis SimCO algorithm and the discussions of its convergence and computational complexity are provided in Section IV. Section V introduces Incoherent Analysis SimCO where an atom decorrelation step is involved. Section VI provides experimental results of learning dictionaries with synthetic data and for image denoising. Conclusions are drawn in Section VII.

II. The SimCO Algorithm

The SimCO algorithm [10] was proposed to learn a synthesis dictionary from a set of signals so that the signals can each be represented by a few atoms of the dictionary. Let $\mathbf{Y} \in \mathbb{R}^{m\times n}$ denote the matrix of the training signals, where each column of $\mathbf{Y}$ is one training signal. In SimCO, the dictionary learning problem is formulated as

$$\arg\min_{\mathbf{D}\in\mathcal{D}} f(\mathbf{D}) = \arg\min_{\mathbf{D}\in\mathcal{D}} \underbrace{\min_{\mathbf{A}\in\mathcal{A}} \|\mathbf{Y}-\mathbf{D}\mathbf{A}\|_F^2}_{f(\mathbf{D})}, \tag{1}$$

where the columns of $\mathbf{A} \in \mathbb{R}^{d\times n}$ are the representation coefficient vectors and $\mathbf{D} \in \mathbb{R}^{m\times d}$ is the dictionary to be learned. In this formulation, $\mathbf{D}$ is assumed to contain unit ℓ_2-norm columns, which is addressed by the constraint $\mathbf{D} \in \mathcal{D}$ with $\mathcal{D}$ representing the set of all matrices that contain unit ℓ_2-norm columns. The positions of the non-zero elements of the coefficient matrix $\mathbf{A}$ are fixed, achieved with the constraint $\mathbf{A} \in \mathcal{A}$.

To solve the optimization problem (1), SimCO follows the conventional two-stage optimization process – a sparse coding stage and a dictionary update stage. The sparse coding stage determines the sparse representations $\mathbf{A}$ of the signals in $\mathbf{Y}$ for a given dictionary $\mathbf{D}$. Various sparse coding algorithms such as OMP [8] can be employed in this stage.

In the dictionary update stage, SimCO applies optimization methods on manifolds [24] to update the dictionary $\mathbf{D}$ under the unit ℓ_2-norm constraints on the columns of $\mathbf{D}$. According to the updated $\mathbf{D}$, the coefficient matrix $\mathbf{A}$ is also updated, while the positions of the non-zero elements are kept unchanged. This framework is able to update multiple atoms and the corresponding coefficients simultaneously, which gives rise to the term simultaneous codeword optimization (SimCO).

III. Problem Formulation and Optimization Framework

Given a set of training signals $\mathbf{Y} \in \mathbb{R}^{m\times n}$, the ADL problem can be written as [7]

$$\begin{aligned} \{\mathbf{\Omega}^*, \mathbf{X}^*\} &= \arg\min_{\{\mathbf{\Omega},\mathbf{X}\}} \|\mathbf{X}-\mathbf{\Omega}\mathbf{Y}\|_F^2 \\ &\text{s.t.} \quad \|\mathbf{X}_{:,i}\|_0 = p-l, \ \forall i. \end{aligned} \tag{2}$$

This is a general formulation without any additional constraint on $\mathbf{\Omega}$ apart from the co-sparsity constraints $\|\mathbf{X}_{:,i}\|_0 = p-l$, $\forall i$.

However, this formulation has ambiguities caused by scaling. In one case, when the training data $\mathbf{Y}$ admits exact sparse representations, there exists a dictionary $\mathbf{\Omega}$ with which the analysis representations of $\mathbf{Y}$, i.e. $\mathbf{X} = \mathbf{\Omega Y}$, satisfy the co-sparsity constraints. If the dictionary $\mathbf{\Omega}$ is scaled by multiplying a scalar $c \in \mathbb{R}$, the corresponding representations $c \cdot \mathbf{X} = c \cdot \mathbf{\Omega Y}$ will also satisfy the constraints. Thus, the problem (2) has infinite optimal solutions $c \cdot \mathbf{\Omega}$ and $c \cdot \mathbf{X}$. This may introduce difficulty in optimization. In the other case, if the data $\mathbf{Y}$ admits approximation representations and $\|\mathbf{X} - \mathbf{\Omega Y}\|_F^2 = \delta$, the value of the cost function with scaled $\mathbf{X}$ and $\mathbf{\Omega}$, i.e. $\|\mathbf{X} - \mathbf{\Omega Y}\|_F^2 = c^2 \cdot \delta$, can be arbitrarily small. In other words, the cost function is unbounded from below, which makes it impossible to find an optimal solution. In addition, (2) has trivial solutions where $\mathbf{\Omega}$ contains all-zero rows.

In order to avoid these problems, we apply the unit ℓ_2-norm constraints on the rows of $\mathbf{\Omega}$, leading to the following formulation of the ADL problem

$$\begin{aligned} \{\mathbf{\Omega}^*, \mathbf{X}^*\} = \arg\min_{\{\mathbf{\Omega}, \mathbf{X}\}} & \|\mathbf{X} - \mathbf{\Omega Y}\|_F^2 \\ \text{s.t.} \quad & \|\mathbf{X}_{:,i}\|_0 = p - l, \ \forall i \\ & \|\mathbf{\Omega}_{j,:}\|_2 = 1, \ \forall j. \end{aligned} \tag{3}$$

The unit ℓ_2-norm constraints on the rows of $\mathbf{\Omega}$ are able to eliminate the scaling ambiguity mentioned above. Besides, the trivial solutions where $\mathbf{\Omega}$ has zero rows can be excluded. The formulation (3) is different from that of Analysis K-SVD [17] which minimizes the error in the signal domain. It also differs from the objective function of LOST [20] where the penalty terms as described earlier in Section I are included.

The problem (3) can be addressed by an optimization framework alternating between two stages: analysis sparse coding and dictionary update. Given a dictionary $\mathbf{\Omega}$, the first stage finds $\mathbf{X}$ satisfying the co-sparsity constraints $\|\mathbf{X}_{:,i}\|_0 = p - l, \ \forall i$. In the dictionary update stage, $\mathbf{\Omega}$ is updated assuming known and fixed $\mathbf{X}$ obtained in the first stage.

Here we attempt to update the dictionary using a similar method as in SimCO and refer to our proposed algorithm as Analysis SimCO. The optimization framework of Analysis SimCO is presented in Algorithm 1. In our original algorithm SimCO, the use of the term "simultaneous" comes from the following two facts: (1) multiple dictionary atoms are updated simultaneously, and (2) their corresponding coefficients are also updated simultaneously with these atoms. In the analysis case, we borrow the term "SimCO" mainly because in the proposed algorithm the dictionary atoms are updated simultaneously.

A common problem with the popular analysis dictionary learning algorithms, such as Analysis K-SVD [17], is that the learned dictionary $\mathbf{\Omega}$ may contain similar atoms. Such a dictionary is regarded as a degenerate solution [7], [21]. This issue is also observed in the dictionary learned from (3) with the Analysis SimCO algorithm, as will be shown in Section VI. Thus, we develop an extended version of Analysis SimCO to avoid this kind of degenerate dictionary, which will be presented in Section V in detail.

Algorithm 1 Optimization Framework of Analysis SimCO

Input: $\mathbf{Y}$, p, l
Output: $\mathbf{\Omega}^\star$
Initialization:
Initialize the iteration counter $k = 1$ and the analysis dictionary $\mathbf{\Omega}^{(k)}$. Perform the following steps.
Main Iterations:
1) Analysis sparse coding: Compute the representations $\mathbf{X}^{(k)}$ with the fixed dictionary $\mathbf{\Omega}^{(k)}$ and the training signals in $\mathbf{Y}$.
2) Dictionary update: Update the dictionary $\mathbf{\Omega}^{(k+1)} \leftarrow \mathbf{\Omega}^{(k)}$.
3) If the stopping criterion is satisfied, $\mathbf{\Omega}^\star = \mathbf{\Omega}^{(k+1)}$ and quit the iteration. Otherwise, increase the iteration counter $k = k + 1$ and go back to step 1).

IV. Analysis SimCO Algorithm

As the dictionary update stage in our algorithm is based on optimization on matrix manifolds, we begin this section with a brief introduction to the optimization on matrix manifolds to make this paper self-contained. The details of the analysis sparse coding and dictionary update are then presented respectively, followed by the convergence and computational complexity analysis of our proposed algorithm.

A. Optimization on Matrix Manifolds

The Stiefel manifold $\mathrm{St}(p, m)(p \leq m)$ is defined as $\mathrm{St}(p, m) := \{\mathbf{U} \in \mathbb{R}^{m \times p} : \mathbf{U}^T\mathbf{U} = \mathbf{I}\}$ [24, pp. 26]. For $p = 1$, the Stiefel manifold $\mathrm{St}(p, m)$ reduces to the unit sphere, i.e., $\mathcal{S} = \{\mathbf{u} \in \mathbb{R}^m : \mathbf{u}^T\mathbf{u} = 1\}$. At each point $\mathbf{u} \in \mathcal{S}$, there exists a tangent space $T_{\mathbf{u}}\mathcal{S}$ which consists of all vectors orthogonal to $\mathbf{u}$ in $\mathbb{R}^m$, i.e. $T_{\mathbf{u}}\mathcal{S} = \{\mathbf{v} \in \mathbb{R}^m : \mathbf{u}^T\mathbf{v} = 0\}$. The vectors in $T_{\mathbf{u}}\mathcal{S}$ are tangent vectors to $\mathcal{S}$ at the point $\mathbf{u}$. The tangent space $T_{\mathbf{u}}\mathcal{S}$ can be regarded as a vector space approximation of the manifold $\mathcal{S}$ at the point $\mathbf{u}$ [24, pp. 34].

Before dealing with the optimization problem on manifolds, we consider a more general class of problems, i.e., the unconstrained optimization problem, from which the optimization methods on matrix manifolds can be adapted,

$$\min_{\mathbf{u}} f(\mathbf{u}), \tag{4}$$

where $\mathbf{u} \in \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}$ is a differentiable function. This problem can be addressed by the standard line search method. In the kth iteration, the standard line search method selects a descent direction $\mathbf{p}$ along which the current point $\mathbf{u}_k$ is moved to a new point $\mathbf{u}_{k+1}$ leading to a smaller or equal objective function value, i.e.

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha \cdot \mathbf{p} \tag{5}$$

with $f(\mathbf{u}_{k+1}) \leq f(\mathbf{u}_k)$. Here α is the scalar step size which can be selected carefully to guarantee the reduction of the cost function [25]. In order to determine the search direction $\mathbf{p}$, the value and the derivatives of the objective function can be used. The most obvious choice is the steepest descent direction $\mathbf{p}_k = -\nabla f(\mathbf{u}_k)$ along which the objective function value decreases most rapidly among all the directions [25, pp. 20].

Now we consider the optimization problem where the variable $\mathbf{u}$ is restricted on the manifold $\mathcal{S}$, i.e.

$$\min_{\mathbf{u}\in\mathcal{S}} f(\mathbf{u}). \tag{6}$$

Analogous line search methods on manifolds have been developed by generalizing the standard line search methods for the unconstrained optimization problem (4). Specifically, in the kth iteration, the search direction $\mathbf{q}_k$ should be chosen as a tangent vector to $\mathcal{S}$ at $\mathbf{u}_k$, i.e., $\mathbf{q}_k \in T_{\mathbf{u}_k}\mathcal{S}$. Thus the search direction $\mathbf{q}_k$ is the projection of the search direction $\mathbf{p}_k$ of the unconstrained optimization methods to the tangent space $T_{\mathbf{u}_k}\mathcal{S}$ [24, pp. 49], that is

$$\mathbf{q}_k = (\mathbf{I} - \mathbf{u}_k\mathbf{u}_k^T)\mathbf{p}_k. \tag{7}$$

The new point $\mathbf{u}_{k+1}$ obtained by moving $\mathbf{u}_k$ in the direction of $\mathbf{q}_k$ should stay on $\mathcal{S}$. As a result, the line search path (5) is replaced by a curve on $\mathcal{S}$ [24, pp. 103] , i.e.

$$\mathbf{u}_{k+1} = \mathbf{u}_k\cos(\alpha\|\mathbf{q}_k\|_2) + \frac{\mathbf{q}_k}{\|\mathbf{q}_k\|_2}\sin(\alpha\|\mathbf{q}_k\|_2). \tag{8}$$

B. Analysis Sparse Coding Stage

The purpose of the analysis sparse coding stage is to get the sparse representations $\mathbf{X}$ of the training signals in $\mathbf{Y}$ based on a given dictionary $\mathbf{\Omega}$. Unlike the corresponding problem of the synthesis model, here the exact representations $\mathbf{X}$ can be calculated directly by simply multiplying the signals in $\mathbf{Y}$ by the dictionary $\mathbf{\Omega}$, that is

$$\mathbf{X} = \mathbf{\Omega}\mathbf{Y}. \tag{9}$$

Since the initial dictionary is an arbitrary one, the representations obtained in this way may not satisfy the co-sparsity constraints on $\mathbf{X}$ in (3). A hard thresholding operation is therefore applied to enforce the co-sparsity

$$\hat{\mathbf{X}} = HT_l(\mathbf{X}), \tag{10}$$

where $HT_l(\mathbf{X})$ is the non-linear operator that sets the smallest l elements (in magnitude) of each column of $\mathbf{X}$ to zero. The representations $\hat{\mathbf{X}}$ obtained via equation (10) are the best approximation of the exact representations $\mathbf{X}$ in terms of the error in Frobenius norm among all the matrices satisfying the co-sparsity constraints.

C. Dictionary Update Stage

The dictionary update stage aims at optimizing the following problem (by fixing $\mathbf{X}$ in (3))

$$\arg\min_{\mathbf{\Omega}} f(\mathbf{\Omega}) = \|\mathbf{X}-\mathbf{\Omega}\mathbf{Y}\|_F^2 \quad \text{s.t.} \quad \|\mathbf{\Omega}_{j,:}\|_2 = 1,\ \forall j. \tag{11}$$

The cost function can be rewritten as a function of the rows of $\mathbf{\Omega}$. Besides, the constraint that $\mathbf{\Omega}$ only contains unit ℓ_2-norm rows restricts the transposes of the rows of $\mathbf{\Omega}$ to lie on the unit sphere $\mathcal{S}$, i.e. $\mathbf{\Omega}_{j,:}^T \in \mathcal{S},\ \forall j$. Thus the problem (11) can be rewritten as

$$\arg\min_{\mathbf{\Omega}} f(\mathbf{\Omega}) = \sum_{j=1}^{p} \|\mathbf{X}_{j,:} - \mathbf{\Omega}_{j,:}\mathbf{Y}\|_2^2 \quad \text{s.t.} \quad \mathbf{\Omega}_{j,:}^T \in \mathcal{S},\ \forall j. \tag{12}$$

As a result, the "line" search methods on manifolds can be utilized in this stage. Here we use the first order optimization procedures as in SimCO [10], i.e. the gradient descent line search method. We explain below the key points of this method including search direction, line search path, and step size respectively. The dictionary update stage is summarized in Algorithm 2.

Algorithm 2 Dictionary Update Stage

Input: $\mathbf{\Omega}^{(k)}$, $\mathbf{X}^{(k)}$, $\mathbf{Y}$
Output: $\mathbf{\Omega}^{(k+1)}$
Main Steps:

1) Calculate the search direction, based on equations (13) and (14).
2) Find a proper step size α using golden section search.
3) Update the dictionary $\mathbf{\Omega}^{(k+1)} \leftarrow \mathbf{\Omega}^{(k)}$, based on equation (15).

1) Search direction: We use the steepest descent direction as the search direction, i.e. the negative gradient of the objective function with respect to $\mathbf{\Omega}$ as follows

$$\begin{aligned}\mathbf{H} &= -\nabla f(\mathbf{\Omega}) \\ &= -\frac{\partial\|\mathbf{X}-\mathbf{\Omega}\mathbf{Y}\|_F^2}{\partial\mathbf{\Omega}} \\ &= 2\mathbf{X}\mathbf{Y}^T - 2\mathbf{\Omega}\mathbf{Y}\mathbf{Y}^T.\end{aligned} \tag{13}$$

2) Line search path: The search direction of the jth row of $\mathbf{\Omega}$, i.e. the projection of each row of $\mathbf{H}$ onto the tangent space of $\mathcal{S}$, is [24, pp. 49]

$$\bar{\mathbf{h}}_j = \mathbf{H}_{j,:}(\mathbf{I} - \mathbf{\Omega}_{j,:}^T\mathbf{\Omega}_{j,:}). \tag{14}$$

According to equation (8), the line search path for the jth row of $\mathbf{\Omega}$ can be written as

$$\mathbf{\Omega}_{j,:}(\alpha) = \begin{cases} \mathbf{\Omega}_{j,:} & \text{if } \|\bar{\mathbf{h}}_j\|_2 = 0, \\ \mathbf{\Omega}_{j,:}\cos(\alpha\|\bar{\mathbf{h}}_j\|_2) + (\bar{\mathbf{h}}_j/\|\bar{\mathbf{h}}_j\|_2)\sin(\alpha\|\bar{\mathbf{h}}_j\|_2) & \\ & \text{otherwise,} \end{cases} \tag{15}$$

where α is the step size.

3) Step size: In order to find a proper step size α, we apply the golden section search method [10]. This method consists of two stages. In the first stage, it finds a range which contains a local minimum and within which the objective function is unimodal. In the second stage, the golden section ratio is used to successively narrow the range until the minimizer is located and thus α is determined.

D. Convergence

Our proposed algorithm alternates between the analysis sparse coding stage and the dictionary update stage. For a fixed dictionary $\mathbf{\Omega}$, $\hat{\mathbf{X}}$ obtained in the analysis sparse coding stage is the optimal solution under the constraint of co-sparsity. Thus, the cost function can only decrease in this stage. In the dictionary update stage, since the update of $\mathbf{\Omega}$ is along a descent direction and the step size is chosen to guarantee that the updated $\mathbf{\Omega}$ will not increase the cost function. Thus, the cost function is decreasing monotonically in our proposed

algorithm. In addition, the cost function of our formulation (3) is lower bounded by zero, i.e. $\|\mathbf{X}-\mathbf{\Omega Y}\|_F^2 \geq 0$. According to the monotone convergence theorem [26], given the cost function decreases monotonically and is lower bounded, the algorithm must converge. The convergence will also be demonstrated experimentally in Section VI-A.

E. Computational Complexity

The time complexity of the Analsysis SimCO algorithm can be analyzed as follows. The time complexity of the sparse coding stage is dominated by the calculation of $\mathbf{\Omega Y}$, at $O(pmn)$, in terms of the analysis in [20]. In the dictionary update stage, the calculation of $\mathbf{H}$ is the dominant part. Computing the product $\mathbf{XY}^T$ requires $O(pmn)$ operations. The time complexity of $\mathbf{\Omega YY}^T$ is $O(pm^2)$ with pre-computed $\mathbf{YY}^T$. As a result, the dictionary update stage requires $O(pmn)$ operations with the usual case $n > m$. The total time complexity of each iteration of the Analysis SimCO algorithm thus scales as $O(pmn)$.

The computational complexity of Analysis SimCO, similar to those of LOST [20], (NL)AOL [18], and Transform K-SVD [21], shows a reduction compared with those of Analysis K-SVD and (NA)AOL. The complexity of Analysis K-SVD is $O(pm^2n)$ using BG or $O(pm^3n)$ using OBG, and (NA)AOL requires $O(pmnk)$ operations with k being the number of dictionary update per iteration. The running time of these algorithms in practice will be given in Section VI.

V. Incoherent Analysis SimCO

As mentioned in Section III, dictionaries learned by the existing ADL algorithms may contain similar atoms, which can degrade the representation performance for signal recovery. To address this problem, several methods have been proposed. For example, in Analysis K-SVD and Transform K-SVD, the similar atoms are replaced by randomly generated atoms, as mentioned in Section I. In LOST [20], a penalty term is used in their objective function to restrict the correlations between atoms. As will be observed in the experiments of Section VI, Analysis SimCO has the same issue, where some of the atoms in the learned dictionary may appear similar. Here, we present an alternative solution to this problem based on [27]. The method in [27] was developed to mitigate the correlations between atoms learned by a synthesis model. Here we adapt this method to our model and optimization problem.

In the context of the sparse synthesis model, the coherence of the dictionary has been defined as a measure of the similarities between the atoms [28]. We extend this definition for an analysis dictionary $\mathbf{\Omega}$ and define the coherence $\mu(\mathbf{\Omega})$ in a row-wise way as

$$\mu(\mathbf{\Omega}) = \max_{\forall i,j,i\neq j} \left| \left\langle \frac{\mathbf{\Omega}_{i,:}}{\|\mathbf{\Omega}_{i,:}\|_2}, \frac{\mathbf{\Omega}_{j,:}}{\|\mathbf{\Omega}_{j,:}\|_2} \right\rangle \right|, \tag{16}$$

From this definition, we have $0 \leq \mu(\mathbf{\Omega}) \leq 1$. With the unit ℓ_2-norm constraints on the rows of $\mathbf{\Omega}$, the coherence $\mu(\mathbf{\Omega})$ can be simplified as

$$\mu(\mathbf{\Omega}) = \max_{\forall i,j,i\neq j} |\langle \mathbf{\Omega}_{i,:}, \mathbf{\Omega}_{j,:} \rangle|. \tag{17}$$

The coherence $\mu(\mathbf{\Omega})$ reflects the maximum correlation of two distinct atoms in $\mathbf{\Omega}$. If $\mu(\mathbf{\Omega})$ is close to 1, it means that there are very similar rows in $\mathbf{\Omega}$, which is the case we attempt to avoid. Thus, we add a coherence constraint $\mu(\mathbf{\Omega}) \leq \mu_0$ to the formulation (3) , i.e.

$$\begin{aligned} \{\mathbf{\Omega}^*, \mathbf{X}^*\} = \arg\min_{\mathbf{\Omega},\mathbf{X}} & \|\mathbf{X}-\mathbf{\Omega Y}\|_F^2 \\ \text{s.t.} \quad & \|\mathbf{X}_{:,i}\|_0 = p-l, \ \forall i \\ & \|\mathbf{\Omega}_{j,:}\|_2 = 1, \ \forall j \\ & \mu(\mathbf{\Omega}) \leq \mu_0, \end{aligned} \tag{18}$$

where μ_0 is the coherence limit for the learned dictionary $\mathbf{\Omega}$.

To enforce the incoherence constraint, we add an extra step in the dictionary update stage, aiming to find the closest dictionary $\hat{\mathbf{\Omega}}$ to $\mathbf{\Omega}$ in Frobenius norm, with the coherence of the dictionary $\hat{\mathbf{\Omega}}$ bounded by a threshold μ_0, that is

$$\begin{aligned} \arg\min_{\hat{\mathbf{\Omega}}} & \|\hat{\mathbf{\Omega}}-\mathbf{\Omega}\|_F^2 \\ \text{s.t.} \quad & \|\hat{\mathbf{\Omega}}_{i,:}\|_2 = 1, \ \forall i \\ & \mu(\hat{\mathbf{\Omega}}) \leq \mu_0. \end{aligned} \tag{19}$$

Here the unit ℓ_2-norm constraints for the atoms in the dictionary are also considered to guarantee that the transposes of the atoms in the output dictionary are still on the manifold. This problem is addressed by applying the decorrelation method [27] in a row-wise fashion, as presented in Algorithm 3. The general idea is to determine the atom pairs whose correlations are greater than μ_0, via a labeling process (from line 5 to line 9 of Algorithm 3), and decorrelate these atom pairs, via a decorrelation process (from line 10 to line 20). This method keeps alternating between the two processes until the coherence of the estimated dictionary $\hat{\mathbf{\Omega}}$ reaches the threshold μ_0. Although this is a heuristic algorithm, it typically involves only a few loops to output an incoherent dictionary. The convergence and the effectiveness of this algorithm will be numerically demonstrated in Section VI.

In the labeling process, the atoms of $\mathbf{\Omega}$ are labeled as either the atom pairs to be decorrelated or atoms that do not need to be modified. An index-pair set F is used to store the index pairs of atom pairs labeled to be decorrelated and an index set E is employed to save the indices of the remaining atoms. $\hat{\mathbf{\Omega}}_E$ represents the submatrix of $\hat{\mathbf{\Omega}}$ only containing the rows indexed by the set E. In each iteration, the correlations of any two distinct rows belonging to $\hat{\mathbf{\Omega}}_E$ are calculated to determine the most correlated pair. The indices of these two atoms will be saved, as an index-pair, into the set F, i.e., $F \leftarrow F \bigcup \{(i,j)\}$, indicating that these two atoms are labeled as an atom pair to be decorrelated in the following decorrelation process. Their indices will be removed from E to avoid being detected again, i.e. $E \leftarrow E \backslash \{i,j\}$.

In the decorrelation process, the atom pairs indexed by the members of F are decorrelated successively. The decorrelation of each atom pair is achieved by rotating the two atoms symmetrically with respect to their mean so that their correlation reaches μ_0 [27]. The rotated atoms are determined based on the orthonormal basis $\{\mathbf{b}_1, \mathbf{b}_2\}$ developed using the atoms to be decorrelated (line 11 and 12) and the angle θ determined by the coherence limit μ_0, i.e., $\theta = \frac{1}{2}\arccos \mu_0$ [27].

Algorithm 3 Atom Decorrelation Step

Input: $\mathbf{\Omega}$, μ_0
Output: $\hat{\mathbf{\Omega}}$
Initialization:
$\hat{\mathbf{\Omega}} = \mathbf{\Omega}$, $\theta = \frac{1}{2}\arccos\,\mu_0$, $c_1 = \cos\,\theta$, $c_2 = \sin\,\theta$
while $\mu(\hat{\mathbf{\Omega}}) > \mu_0$ **do**
  $E = \{1, 2, ..., p\}$ // line 5-9: labeling process
  $F = \emptyset$
  while $\mu(\hat{\mathbf{\Omega}}_E) > \mu_0$ **do**
    $(i, j) = \arg\,\max_{\forall i,j \in E, i \neq j}\ |\hat{\mathbf{\Omega}}_{i,:}\hat{\mathbf{\Omega}}_{j,:}^T|$
    $F \leftarrow F \bigcup \{(i, j)\}$
    $E \leftarrow E \backslash \{i, j\}$
  end while // line 10-20: decorrelation process
  for $\forall (i, j) \in F$ **do**
    $\mathbf{b}_1 = (\hat{\mathbf{\Omega}}_{i,:} + \hat{\mathbf{\Omega}}_{j,:})/(\|\hat{\mathbf{\Omega}}_{i,:} + \hat{\mathbf{\Omega}}_{j,:}\|_2)$
    $\mathbf{b}_2 = (\hat{\mathbf{\Omega}}_{i,:} - \hat{\mathbf{\Omega}}_{j,:})/(\|\hat{\mathbf{\Omega}}_{i,:} - \hat{\mathbf{\Omega}}_{j,:}\|_2)$
    if $\langle \hat{\mathbf{\Omega}}_{i,:},\ \hat{\mathbf{\Omega}}_{j,:} \rangle > 0$ **then**
      $\hat{\mathbf{\Omega}}_{i,:} = c_1\mathbf{b}_1 + c_2\mathbf{b}_2$
      $\hat{\mathbf{\Omega}}_{j,:} = c_1\mathbf{b}_1 - c_2\mathbf{b}_2$
    else
      $\hat{\mathbf{\Omega}}_{i,:} = c_1\mathbf{b}_2 + c_2\mathbf{b}_1$
      $\hat{\mathbf{\Omega}}_{j,:} = -c_1\mathbf{b}_2 + c_2\mathbf{b}_1$
    end if
  end for
end while
return $\hat{\mathbf{\Omega}}$

In order to address the problem (18), the atom decorrelation step is inserted after the dictionary update stage in the loop of Analysis SimCO (Algorithm 1), as summarized in Algorithm 4. We referred to this extended version of Analysis SimCO as Incoherent Analysis SimCO. Actually, Analysis SimCO can be regarded as the special case of Incoherent Analysis SimCO if $\mu_0 = 1$.

Algorithm 4 Incoherent Analysis SimCO

Input: $\mathbf{Y}$, p, l, μ_0
Output: $\mathbf{\Omega}^\star$
Initialization:
Initialize the iteration counter $k = 1$ and the analysis dictionary $\mathbf{\Omega}^{(k)}$. Perform the following steps.
Main Iterations:

1) Analysis sparse coding: Compute the representations $\mathbf{X}^{(k)}$ with the fixed dictionary $\mathbf{\Omega}^{(k)}$ and the training signals in $\mathbf{Y}$, based on equations (9) and (10).
2) Dictionary update: Update the dictionary $\mathbf{\Omega}^{(k+1)} \leftarrow \mathbf{\Omega}^{(k)}$, using Algorithm 2.
3) Atom decorrelation: Decorrelate the atoms $\hat{\mathbf{\Omega}}^{(k+1)} \leftarrow \mathbf{\Omega}^{(k+1)}$, using Algorithm 3.
4) If the stopping criterion is satisfied, $\mathbf{\Omega}^\star = \hat{\mathbf{\Omega}}^{(k+1)}$, quit the iteration. Otherwise, increase the iteration counter $k = k + 1$ and go back to step 1).

It is worth noting that other alternative methods could also be used to promote incoherent dictionaries. As mentioned earlier, Analysis K-SVD replaces the similar atoms with vectors generated in a random way. Compared with this method, the decorrelation step applied in Incoherent Analysis SimCO can better preserve the information in the dictionary atoms since the new atoms are generated by rotating the existing atoms to be replaced. The method in Incoherent K-SVD (IKSVD) [29], proposed for the synthesis model, can also be used to decorrelate the atoms of the analysis dictionary, which can be achieved by minimizing $\|\mathbf{\Omega}\mathbf{\Omega}^T - \mathbf{I}\|_F^2$ after the update of the dictionary. However, this method cannot directly control the degree of the coherence of the dictionary. For comparison, we modify the Incoherent Analysis SimCO by replacing the decorrelation step (i.e. step 3 in Algorithm 4) with these two decorrelation methods, which we refer to as Analysis SimCO-Random (ASimCO-Random) and Analysis SimCO-IKSVD (ASimCO-IKSVD) respectively.

VI. Simulation Results

In this section we present two categories of experiments to demonstrate the performance of our proposed algorithms. The first category contains experiments with synthetic data, and the second one provides image denoising results using the dictionaries learned with different ADL algorithms.

A. Experiments with Synthetic Data

Now we test the ADL algorithms with synthetic data. First of all, the approach to generating the synthetic data sets and the performance metrics employed are introduced. Second, we test our proposed algorithms with different initial dictionaries, showing their convergence and robustness to initializations. Third, the effect of the atom decorrelation step of the Incoherent Analysis SimCO algorithm is demonstrated. Fourth, experiments with different parameters are conducted to provide a more comprehensive comparison between our proposed algorithms and other ADL algorithms.

1) Synthetic data generation and performance metrics: A set of synthetic data consists of a reference analysis dictionary $\mathbf{\Omega} \in \mathbb{R}^{p \times m}$ and a set of signals in $\mathbf{Y} \in \mathbb{R}^{m \times n}$ that is sparse with respect to $\mathbf{\Omega}$ with co-sparsity l. The reference dictionary $\mathbf{\Omega}$ is generated as detailed in the settings of the experiments. The generation of the signals in $\mathbf{Y}$ is based on the fact that the sparse analysis model can be used as a generative model with a given dictionary [6]. For generating each signal, l rows of $\mathbf{\Omega}$ are selected randomly and a basis for the null space of these l rows is determined. Multiplying this basis by a random vector gives a vector which can be one member of the signal set, i.e. one column of $\mathbf{Y}$. In the following experiments, $\mathbf{Y}$ and its noisy version will both be used as the training samples. The noisy training signals are obtained by adding Gaussian noise with zero mean and standard deviation 0.04 to $mathbfY$, set as in [17]. Learning with the original signals $\mathbf{Y}$ is referred to as the noiseless case and learning with the noisy signals as the noisy case.

An advantage of using synthetic data is that the reference dictionary which can sparsify the signals exactly is available, and therefore the quality of a learned dictionary can be evaluated by comparing it with the reference dictionary. We use the recovery rate of the atoms to measure the performance

of the algorithms for recovering the reference dictionary, following the experiments in [17]. An atom $\mathbf{\Omega}_{j,:}$ of the reference dictionary is regarded as recovered if

$$\min_i (1 - |\hat{\mathbf{\Omega}}_{i,:} \mathbf{\Omega}_{j,:}^T|) < \tau, \tag{20}$$

where $\hat{\mathbf{\Omega}}_{i,:}$ are the atoms of the learned dictionary and τ is the threshold value to determine whether the atoms are recovered. The value of τ is typically set as 0.01 [17].

Another way for evaluating a learned dictionary is to consider the average co-sparsity of the original signals in $\mathbf{Y}$ with respect to this dictionary since the final goal of ADL is to acquire a dictionary with which the analysis representations of the signals are sparse. We introduce an operator $\|\mathbf{x}\|_0^\epsilon$ counting the number of the elements of $\mathbf{x} \in \mathbb{R}^p$, which are below the threshold ϵ, i.e.

$$\|\mathbf{x}\|_0^\epsilon = \text{card}(\{i : |x_i| < \epsilon, i = 1, 2, ..., p\}), \tag{21}$$

where x_i denotes the ith element of $\mathbf{x}$ and $\epsilon > 0$. The threshold value ϵ should be close to zero and it is set as $\epsilon = 0.001$ throughout our experiments. The co-sparsity of a signal can be obtained by applying this operator to the product of the learned dictionary and this signal. The average co-sparsity of all signals are used as the second metric to evaluate the learned dictionaries.

2) Convergence of the proposed algorithms: Different initial dictionaries are used to demonstrate the convergence of our proposed algorithms. The reference dictionaries were generated with the random variables satisfying the i.i.d. Gaussian distribution with zero mean and unit variance and then the rows of the dictionaries were normalized. The size of the reference dictionaries was 50×25 (i.e. $p = 50$, $m = 25$). The number of training signals was 50000 (i.e. $n = 50000$) and their co-sparsity was 21 (i.e. $l = 21$) set as in [17]. Analysis SimCO and Incoherent Analysis SimCO were applied to learn analysis dictionaries respectively. The co-sparsity parameters of these two algorithms were both set as the reference co-sparsity. The coherence limit of Incoherent Analysis SimCO was set as $\mu_0 = 0.6$, based on our empirical tests.

Three types of matrices were used as initial dictionaries, following the experiments of [7]. The first type is the random matrix consisting of i.i.d. zero mean and unit variance Gaussian elements. The other two types are vertical concatenations of two matrices. One type is the vertical concatenations of two 25×25 2D DCT matrices (defined as the Kronecker product of two 5×5 1D DCT matrix), and the other is composed of two 25×25 identity matrices. We have used 100 independent runs to test the proposed algorithms, the change of the objective function (3) shows similar patterns in different runs. Fig. 1 shows the objective function value averaged from ten independent tests of the proposed algorithms over the iterations in noiseless case and noisy case. The objective function decreases monotonically for all the initializations in both the noiseless case and noisy case. Though the dictionaries were initialized in different ways, the algorithms converge to a similar final value. This indicates that our proposed algorithms can converge robustly with different initializations.

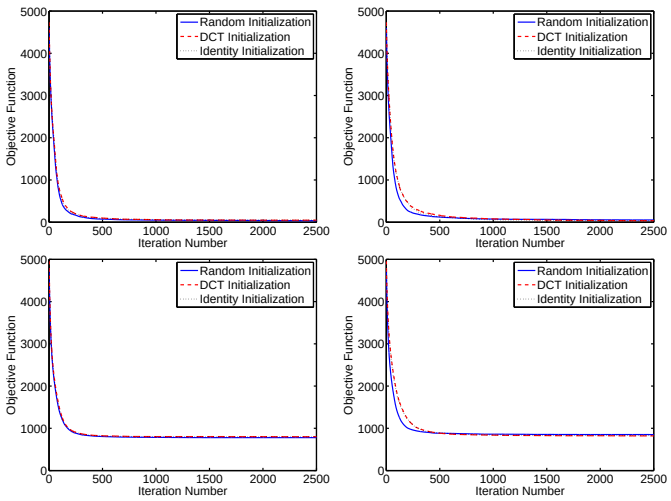


Fig. 1. Objective function value with different initializations in the noiseless case (top) and the noisy case (bottom). Left column: Analysis SimCO. Right column: Incoherent Analysis SimCO.

3) Effect of the atom decorrelation step: Now we compare the correlations of the atoms in the dictionaries learned by Analysis SimCO and Incoherent Analysis SimCO to show the effect of the atom decorrelation step. The initial dictionaries were set as random Gaussian matrices with normalized rows. Other settings were the same as those in the experiments of Fig. 1.

In order to observe the correlations of the atoms of a learned dictionary $\mathbf{\Omega}$, we define its mutual correlation matrix $\mathbf{M}$ as follows

$$\mathbf{M}(\mathbf{\Omega}) = \text{abs}(\mathbf{I} - \mathbf{\Omega}\mathbf{\Omega}^T), \tag{22}$$

where operator $\text{abs}(\cdot)$ takes the element-wise absolute value of a matrix. The non-diagonal elements of $\mathbf{M}(\mathbf{\Omega})$ represent the correlations between atoms of $\mathbf{\Omega}$ and thus the coherence of $\mathbf{\Omega}$ is the maximum value of all the elements of $\mathbf{M}$, i.e. $\mu(\mathbf{\Omega}) = \max(\mathbf{M}(\mathbf{\Omega}))$. The histograms of the mutual correlation matrices of the dictionaries learned by Analysis SimCO and Incoherent Analysis SimCO in one test are presented in Fig. 2, in both the noiseless case and noisy case. In the mutual correlation matrix obtained by Analysis SimCO, there are some elements close to 1, which means that highly-correlated atoms exist in the learned dictionary. These highly-correlated atoms disappear in the dictionary learned by Incoherent Analysis SimCO, as shown in the right plot of Fig. 2. This demonstrates that the atom decorrelation step can effectively avoid the highly-correlated atoms in the learned dictionary.

The recovery rate and average co-sparsity averaged from ten independent tests are shown in Fig. 3. It can be seen that the recovery rate is higher in both the noiseless case and noisy case, when the Incoherent Analsysis SimCO algorithm is applied. The average co-sparsity in the noisy case also increases due to the atom decorrelation step. In the noiseless case, the average co-sparsity obtained by Incoherent Analysis SimCO is lower than that obtained by Analysis SimCO. This is because some atoms which can sparsify the training signals with high co-sparsity are replaced because of their high correlation. Even though the dictionaries learned by Analysis SimCO can reach

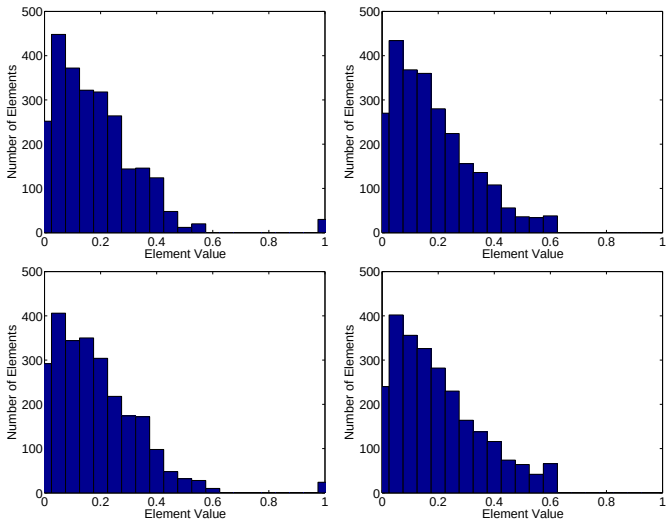


Fig. 2. The histograms of the elements in the mutual correlation matrices of the dictionaries learned in the noiseless case (top) and the noisy case (bottom). Left column: Analysis SimCO. Right column: Incoherent Analysis SimCO.

higher average co-sparsity, Incoherent Analysis SimCO can learn the dictionaries without highly-correlated atoms.

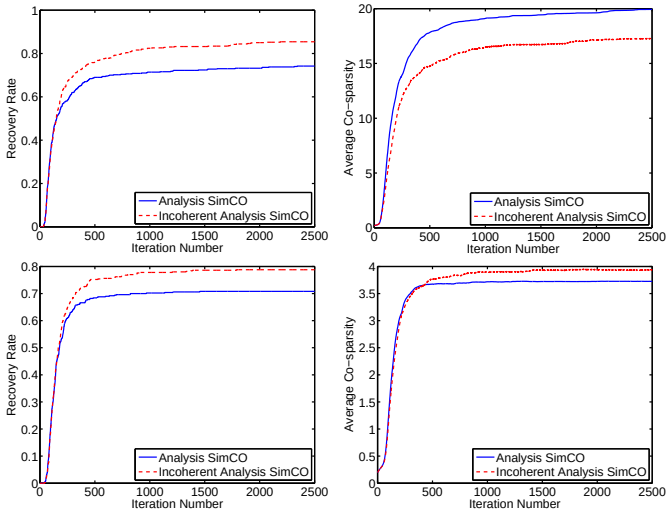


Fig. 3. Recovery Rate and Average Co-sparsity over iterations in the noiseless case (top) and the noisy case (bottom).

4) Simulations with different parameters: Our proposed algorithms are compared with seven baseline algorithms: ASimCO-Random, ASimCO-IKSVD, Analysis K-SVD [17], LOST [20], AOL [18], Transform K-SVD [21] and GOAL [19][1].

The algorithms were tested with different parameters, i.e. co-sparsity l, the number of training signals n and the number of atoms p. In each test, one parameter was changed while the others were fixed, as shown in Table I. These parameters are selected empirically to show the trends of the learning results of the algorithms in terms of recovery rate and average co-sparsity. The reference dictionaries were generated with random variables satisfying i.i.d. Gaussian distribution with zero mean and unit variance and their rows are normalized. The initial dictionaries used in all the algorithms were also generated in the same way.

TABLE I
PARAMETERS USED IN THE COMPARISON OF DIFFERENT ADL ALGORITHMS WITH SYNTHETIC DATA.

1	Fixed parameters	$p = 50$, $m = 25$, $n = 50000$
	Varying parameter l	$l \in \{4,\ 8,\ 12,\ 16,\ 20,\ 24\}$
2	Fixed parameters	$p = 50$, $m = 25$, $l = 18$
	Varying parameter n	$n \in \{0.5,\ 1,\ 2,\ 4,\ 6,\ 8\} \times 10^4$
3	Fixed parameters	$m = 25$, $l = 18$, $n = 50000$
	Varying parameter p	$p \in \{30,\ 40,\ 50,\ 60,\ 70,\ 80\}$

Analysis SimCO and Incoherent Analysis SimCO were applied for 2500 iterations. For Incoherent Analysis SimCO, the coherence limit was $\mu_0 = 0.6$. The parameters of Analysis K-SVD were set as the experiments with synthetic data in [17]. We found that the LOST algorithm fails to recover any atom of the reference dictionary if the parameters as in the original paper [20] are used. This may be because the experiments with synthetic data scale differently from the experiments with image patches in [20]. Extensive experiments were conducted to find good parameters of LOST for the experiments with synthetic data. The coefficients of the penalty terms in the objective function were chosen as 50 and the index parameter in the correlation penalty term was 20. The step size and the iteration number of the inner gradient conjugate algorithm were 10^{-4} and 30 respectively. The number of iterations for LOST was fixed to 1000. For the AOL algorithms, its noiseless version (NL)AOL and noise-aware version (NA)AOL were applied to the noiseless case and the noisy case respectively. The iteration numbers of (NL)AOL and (NA)AOL were 50000 and 10 respectively, according to the settings in [18]. The coefficient of the objective function of (NA)AOL was $\lambda = 0.3$. Other parameters of these two algorithms were the same as suggested in [18]. The parameters of Transform K-SVD were set at their default values as in [21]. The parameters of GOAL were set as in the original code[2]. The threshold used to replace similar rows in ASimCO-Random is also set as μ_0 to be consistent with the coherence limit of Incoherent Analysis SimCO. The parameters for the decorrelation method in ASimCO-IKSVD are set as recommended in [29].

The recovery rate and average co-sparsity averaged from five independent tests with different l, n and p are presented in Figs. 4, 5 and 6 respectively. Abbreviations are used in the legends because of space limitation (IN-ASimCO, ASimCO, AKSVD and TKSVD are short for Incoherent Analysis SimCO, Analysis SimCO, Analysis K-SVD and Transform K-SVD respectively). In general, our proposed algorithms, Analysis K-SVD, LOST and Transform K-SVD show similar trends over the varying parameters. This may result from the same measurements used for co-sparsity, i.e. ℓ_0-norm, and their similar optimization procedure which alternates between

[1]The code of GOAL was downloaded from http://www.gol.ei.tum.de/index.php?id=25&type=98.

[2]We should note that, in GOAL the values of the parameters set in the code downloaded are different from those presented in the original paper. Therefore, we tested both sets of parameters, the values of the parameters set in the code were used in our experiments as we observed that they usually lead to better results.

the update of the analysis representation and the update of the dictionary. For these five algorithms, better dictionaries can be learned with larger co-sparsities (cf. Fig. 4) and more training samples (cf. Fig. 5). With the increase of the number of atoms, the recovery rates obtained by these algorithms decrease (cf. Fig. 6). The results of Incoherent Analysis SimCO are similar to the results of Analysis K-SVD and Transform K-SVD, which are better than the results of LOST. The recovery rates of the dictionaries obtained by Incoherent Analysis SimCO are higher than Analysis SimCO in all cases due to the restriction of the coherence of the learned dictionary. The average co-sparsities obtained by Analysis SimCO are closer to the reference co-sparsities than those obtained by Incoherent Analysis SimCO in the noiseless case, but the Incoherent Analysis SimCO algorithm shows advantage for the average co-sparsity in the noisy case. The results of (NL)AOL, (NA)AOL and GOAL appear to be quite different from the other methods compared. This might be due to the "ℓ_1-norm" or "ℓ_p-norm" ($0 \leq p \leq 1$) used to estimate the co-sparsity of the coefficients, as opposed to the "ℓ_0-norm" used in the other algorithms. The relatively limited performances of (NL)AOL and (NA)AOL may result from the application of the UNTF constraint to the learned dictionaries, that the reference dictionaries do not satisfy. The results of Incoherent Analysis SimCO and ASimCO-Random are very similar to each other, and they both outperform the ASimCO-IKSVD algorithm.

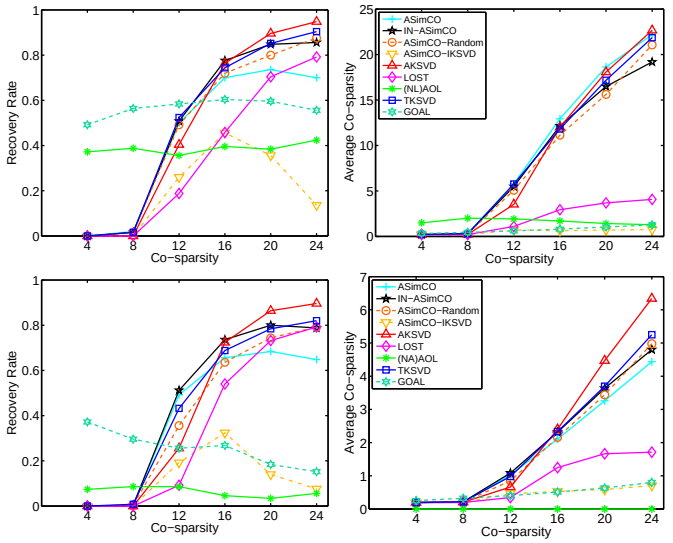


Fig. 4. Recovery Rate (left) and Average Co-sparsity (right) with different co-sparsities ($l \in \{4, 8, 12, 16, 20, 24\}$) in the noiseless case (top) and the noisy case (bottom).

The time (in seconds) of one test with different parameters is presented in Table II [3]. From Table II, we can see that our proposed algorithms are faster than Analysis K-SVD, LOST and (NL)AOL, but slower than Transform K-SVD and GOAL. It seems that Transform K-SVD is the best choice to learn dictionaries with synthetic data considering its good performance and efficient computation. However, for

[3]All algorithms were implemented in Matlab R2012a and performed with an Intel Core i5 CPU at 3.30GHz and 8GB memory.

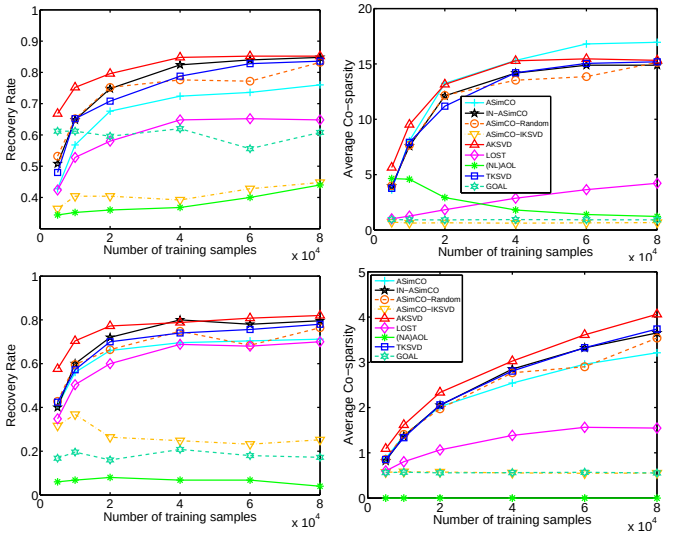


Fig. 5. Recovery Rate (left) and Average Co-sparsity (right) with different numbers of training samples ($n \in \{0.5, 1, 2, 4, 6, 8\} \times 10^4$) in the noiseless case (top) and the noisy case (bottom).

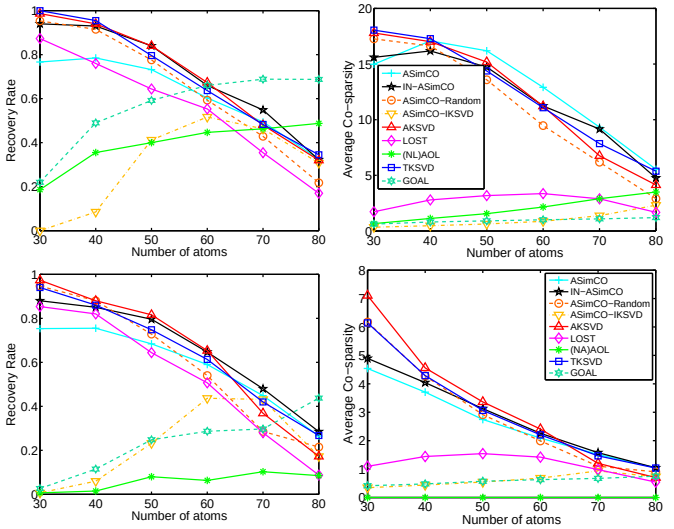


Fig. 6. Recovery Rate (left) and Average Co-sparsity (right) with different numbers of atoms ($p \in \{30, 40, 50, 60, 70, 80\}$) in the noiseless case (top) and the noisy case (bottom).

the application to image denoising, our proposed algorithm outperforms Transform K-SVD, which will be presented in the next two subsections. The running time of (NA)AOL changes substantially in different cases since sometimes the subgradient algorithm applied to update the dictionary requires longer time to converge. (NA)AOL seems to be faster than our proposed algorithms as shown in Table II. It should be noted that the running time presented here is the time for one test, however, the termination conditions for the algorithms are different. For each iteration, our proposed algorithms are faster than (NA)AOL which is consistent with the analysis of the computational complexities in Section IV-E.

B. Experiments for Image Denoising

We apply the learned dictionaries to image denoising which has become a common application for demonstrating ADL

TABLE II
TIME OF ONE TEST WITH DIFFERENT PARAMETERS (IN SECONDS).

Number of co-sparsities l						
Algorithm	4	8	12	16	20	24
ASimCO	1373	1232	1323	1290	1329	1488
IN-ASimCO	1411	1264	1227	1219	1233	1064
ASimCO-Random	1234	1136	1022	1051	1027	1025
ASimCO-IKSVD	1188	1054	938	946	981	1002
AKSVD	1376	2940	4697	6532	8677	11037
LOST	1523	1500	1559	1716	1521	1448
TKSVD	209	222	232	242	252	260
(NL)AOL	4196	4489	4108	4255	4430	4167
(NA)AOL	13	13	346	347	15	348
GOAL	28	34	31	34	28	32
Number of training signals n (10^4)						
Algorithm	0.5	1	2	4	6	8
ASimCO	87	263	531	1055	1580	2201
IN-ASimCO	105	222	504	862	1506	1989
ASimCO-Random	135	268	537	993	1458	1874
ASimCO-IKSVD	121	240	463	839	1214	1704
AKSVD	807	1557	3075	6164	9243	12090
LOST	354	449	721	1305	2039	2953
TKSVD	132	146	175	219	272	323
(NL)AOL	563	1017	1775	3356	5459	7131
(NA)AOL	59	52	57	71	18	24
GOAL	5	13	18	29	14	57
Number of atoms p						
Algorithm	30	40	50	60	70	80
ASimCO	1120	1264	1491	1530	1726	1787
IN-ASimCO	766	874	1344	1360	1454	1528
ASimCO-Random	908	1000	1106	1231	1356	1518
ASimCO-IKSVD	799	884	1001	1069	1170	1386
AKSVD	6700	7069	7561	8081	8697	9127
LOST	1168	1229	1552	2138	2068	2305
TKSVD	162	202	244	283	309	338
(NL)AOL	2610	3349	4690	5255	6134	6893
(NA)AOL	36	53	68	101	21	95
GOAL	23	31	32	45	48	60

algorithms [17], [21], [20]. In this section, the image denoising framework, the performance evaluation index, and the parameter selection are introduced first. After that, the denoising results for face images and natural images are presented.

1) Image denoising framework: The image denoising framework employed in our experiments consists of dictionary learning and image recovering, which are both based on small image patches [4], [17]. To denoise a large image of size $\sqrt{N} \times \sqrt{N}$, small image patches of size $\sqrt{m} \times \sqrt{m}$ with $m < N$ are used as the training signals to learn an analysis dictionary $\mathbf{\Omega} \in \mathbb{R}^{p \times m}$. These training patches are extracted from the image to be denoised or from other clean images. In the image recovering process, the noisy image is also handled as overlapping patches of the same size. Specifically, $\sqrt{m} \times \sqrt{m}$ patches extracted from the noisy image are reshaped as column vectors which are concatenated as a matrix $\mathbf{Z} \in \mathbb{R}^{m \times n}$, where n is the number of patches. The recovering operation is directly applied to $\mathbf{Z}$ using the learned dictionary $\mathbf{\Omega}$, resulting in a noiseless estimation $\mathbf{Y} \in \mathbb{R}^{m \times n}$. Overlapping patches are used to mitigate the blockiness artifacts caused by this patch-based framework. The denoised image can be obtained by reshaping the columns of $\mathbf{Y}$ as image patches and averaging these overlapping patches.

The key idea of estimating $\mathbf{Y}$ is to solve an optimization problem where the learned analysis dictionary $\mathbf{\Omega}$ serves in the regularization term reflecting the co-sparsity prior of $\mathbf{Y}$, that is

$$\arg\min_{\mathbf{Y}} \|\mathbf{\Omega Y}\|_1 + \frac{\lambda}{2}\|\mathbf{Z} - \mathbf{Y}\|_F^2, \tag{23}$$

where λ is the Lagrangian multiplier to balance the data fidelity term $\|\mathbf{Z} - \mathbf{Y}\|_F^2$ and the regularization term $\|\mathbf{\Omega Y}\|_1$. The alternating direction method of multipliers (ADMM) [30], [18] is applied to tackle this problem.

It should be noted that the methods used for image recovery in the experiments of LOST [20], Analysis K-SVD [18], Transform K-SVD [21] and GOAL [19] are different, which makes it difficult to evaluate the dictionaries learned by different algorithms consistently. To make a fair comparison, the same image recovering method, formulated as (23), is used in our experiments. This method is selected because of its high computational efficiency.

2) Denoising performance evaluation and parameter selection: The images to be denoised were artificially corrupted by additive white Gaussian noise with the standard deviation being either $\sigma = 12.8$ or $\sigma = 45$, choosing empirically to represent the case of a relatively low or high level of noise respectively. Peak signal to noise ratio (PSNR) was used to measure the denoising performance. For an N-pixel noise-free image $\mathbf{y} \in \mathbb{R}^N$, the PSNR in decibels (dB) of its denoised version $\hat{\mathbf{y}} \in \mathbb{R}^N$ is defined as

$$\text{PSNR} = 10\log_{10}\frac{255^2 N}{\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2}, \tag{24}$$

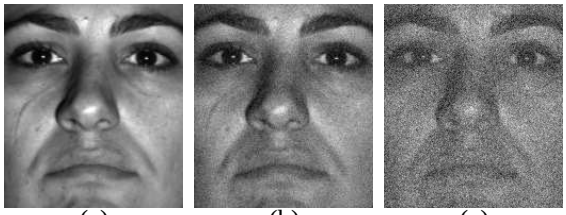
where $\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2$ is the mean squared error between the original image and its denoised version.

Throughout our experiments, we followed the same set up as in [18], [21] and fixed the size of the image patches to 8×8, i.e. $m = 64$. The overlap of the patches was set as 7. The size of the learned dictionaries was 128×64, i.e. $p = 128$. In the image recovering process, the proper selection of the Lagrangian multiplier λ is related to the noise level. In general, λ needs to be smaller when the noise level is higher. The method to choose optimal λ is out of the scope of our work. Herein a set of different λ's was tested, and only the results of $\lambda \in \{0.002,\ 0.01,\ 0.05,\ 0.1,\ 0.3,\ 0.5\}$ are presented to show the trends of the denoising results.

We still compare our proposed algorithms with the baseline algorithms as employed in the experiments for synthetic data. The parameters about co-sparsity were set as follows. For Analysis SimCO and Incoherent Analysis SimCO, the co-sparsities were set as either $l = 40$ or $l = 80$. The corresponding parameters of the baseline algorithms were set based on the value of l, in order to ensure the equal co-sparsity. For Analysis K-SVD, only the $l = 40$ case was tested since l cannot be greater than the signal dimension m in its parameter settings [17]. There is no parameter related to the co-sparsity l in (NA)AOL and GOAL. Other parameters

were set as those employed in their original papers except for GOAL whose parameters are the same as in the experiments for synthetic data. The coherence limit of Incoherent Analysis SimCO and the correlation threshold of ASimCO-IKDVD were both set as 0.2, which was lower than the value used in the experiments with synthetic data, since we found that, in general, the image dictionaries learned have a relatively lower coherence, as compared with that in the synthetic case. The same initial dictionaries, generated with i.i.d. Gaussian distribution with zero mean and unit variance, were used for different algorithms.

3) Face image denoising: Now we denoise face images using the learned analysis dictionaries, following the experiments in [18]. The face images are centred and cropped [31] and can be modelled as piecewise smooth signals approximately. The original face and the noisy face images are shown in Fig. 7. Two types of training data were tested: Type I consists of the patches extracted from 13 other clean face images [18]; Type II includes the patches extracted from the face image to be denoised. 16384 patches were randomly selected as training data in both cases [18].



(a) (b) (c)

Fig. 7. Face images. (a) Original face. (b) Noisy face with $\sigma = 12.8$ (PSNR = 26.00 dB). (c) Noisy face with $\sigma = 45$ (PSNR = 15.13 dB).

The PSNR (in dB) values of the denoised face images averaged from five independent tests with varying λ are presented in Fig. 8 ($\sigma = 12.8$) and Fig. 9 ($\sigma = 45$). In each of these two figures, the top and bottom sub-figures show the results using the Type I and Type II training data, respectively. The left and right sub-figures present the denoising results with $l = 40$ and $l = 80$, obtained by our proposed algorithms, LOST and Transform K-SVD. The results of Analysis K-SVD are only shown in the left sub-figures. The results of (NA)AOL and GOAL, which are not related to the co-sparsity setting, are plotted without modifications in both the left and right sub-figures. The best denoising results obtained via different algorithms with varying λ, i.e. the peak PSNR values of the lines in Fig. 8 and Fig. 9, are summarized in Table III.

Fig. 8 and Fig. 9 reveal that the denoising results change considerably with various λ, except for ASimCO-Random. The best λ of the set tested in the case $\sigma = 45$ is smaller than that of the case $\sigma = 12.8$ due to the increase of the noise level. Some common features of the ADL algorithms can also be observed from Table III. The dictionaries learned with higher co-sparsity ($l = 80$) perform better in general. In terms of the types of training data, the image patches from the noisy face image itself (Type II training data) seem to be more suitable to be the training data. As shown in Table III, the performance of GOAL is competitive, compared with other baseline algorithms. For the lower noise level with

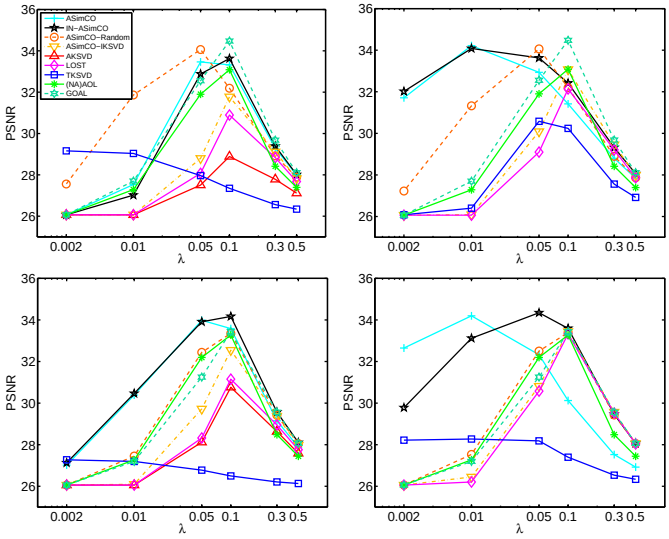


Fig. 8. Face image denoising ($\sigma = 12.8$). Top row: training patches are extracted from 13 other clean face images, with the co-sparsity $l = 40$ (left) and $l = 80$ (right). Bottom row: training patches are extracted from the face image to be denoised, with the co-sparsity $l = 40$ (left) and $l = 80$ (right).

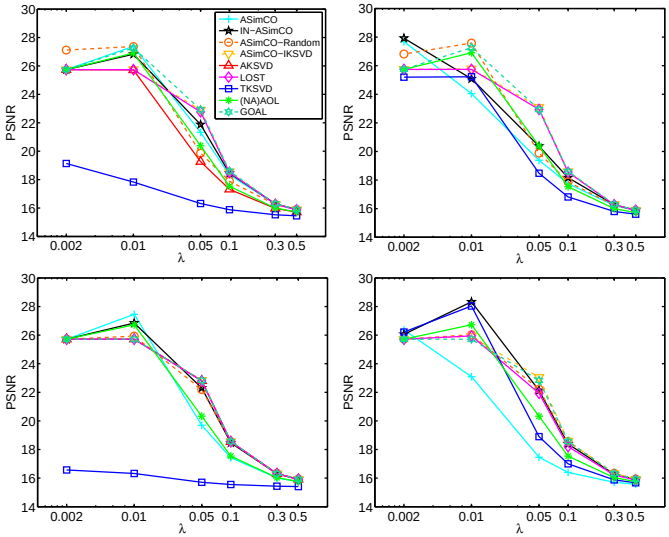


Fig. 9. Face image denoising ($\sigma = 45$). Top row: training patches are extracted from 13 other clean faces, with the co-sparsity $l = 40$ (left) and $l = 80$ (right). Bottom row: training patches are extracted from the face image to be denoised, with the co-sparsity $l = 40$ (left) and $l = 80$ (right).

Type I training data, it obtains the best result. However, our proposed algorithms can obtain better results than the baseline algorithms in other cases. Incoherent Analysis SimCO is able to get higher PSNR than Analysis SimCO in some cases. The results of ASimCO-Random reflect that the decorrelation method employed in Analysis K-SVD is not suitable for the face denoising task.

Fig. 10. Test images for natural image denoising.

TABLE III
THE BEST FACE IMAGE DENOISING RESULTS (PSNR IN DECIBELS).

σ = 12.8				
Training data type	Type I		Type II	
co-sparsity l	40	80	40	80
ASimCO	33.46	34.22	33.98	34.20
IN-ASimCO	33.63	34.08	34.17	34.35
ASimCO-Random	34.06	34.06	33.35	33.39
ASimCO-IKSVD	31.78	33.07	32.54	33.44
AKSVD	28.89	—	30.76	—
LOST	30.90	32.14	31.15	33.33
TKSVD	29.16	30.58	27.27	28.27
(NA)AOL	33.08		33.27	
GOAL	34.47		33.43	

σ = 45				
Training data type	Type I		Type II	
co-sparsity l	40	80	40	80
ASimCO	27.38	27.69	27.46	26.37
IN-ASimCO	26.83	27.92	26.85	28.33
ASimCO-Random	27.36	27.58	25.93	26.03
ASimCO-IKSVD	25.74	25.78	25.72	25.92
AKSVD	25.74	—	25.72	—
LOST	25.74	25.74	25.72	25.94
TKSVD	19.13	25.22	16.57	28.04
(NA)AOL	26.91		26.72	
GOAL	27.27		25.72	

4) Natural image denoising: Now we examine the denoising of the natural images shown in Fig. 10. The size of the images is 256×256. Similar to the denoising of face images, two types of training data are tested: Type I contains image patches extracted from 5 other clean images shown in Fig. 11; Type II includes the patches of the image to be denoised. The number of training patches is 20000 [17]. The results (PSNR in dB) averaged from the denoised versions of the four test images are plotted in Fig. 12 ($\sigma = 12.8$) and Fig. 13 ($\sigma = 45$), and the peak results of each curve are listed in Table IV.

Fig. 11. Training images used for learning analysis dictionaries.

According to Fig. 12 and Fig. 13, the best λ is bigger for the lower noise level, which is consistent with the objective function (23). Table IV indicates that the patches extracted from other clean natural images are preferred to the patches of the image to be denoised. Our proposed algorithms outperform the baseline problems, except in the $\sigma = 12.8$ case with Type I training data where (NA)AOL and GOAL gives slightly better results. The results obtained by Analysis SimCO and Incoherent Analysis SimCO are similar to each other.

VII. CONCLUSION

In this paper we have proposed an analysis dictionary learning algorithm: Analysis SimCO. The dictionary learning process is formulated as an optimization problem with the co-sparsity and unit ℓ_2-norm constraints on the atoms of the dictionary. This algorithm iteratively solves this problem by hard thresholding and the gradient descent method on manifolds. We have also presented an extension of Analysis SimCO: Incoherent Analysis SimCO, by incorporating an atom decorrelation step after the dictionary update step. Extensive experiments on synthetic data, face and natural image data have confirmed the competitive performance of our proposed algorithms. The applications of learned analysis dictionaries in other signal processing tasks merit more study, which we leave for future work.

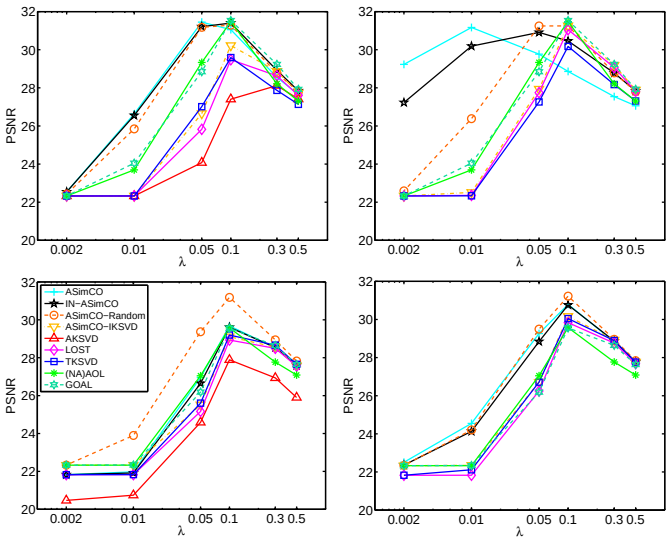


Fig. 12. Natural image denoising ($\sigma = 12.8$). Top row: training patches are extracted from the images in Fig. 11, with the co-sparsity $l = 40$ (left) and $l = 80$ (right). Bottom row: training patches are extracted from the natural image to be denoised, with the co-sparsity $l = 40$ (left) and $l = 80$ (right).

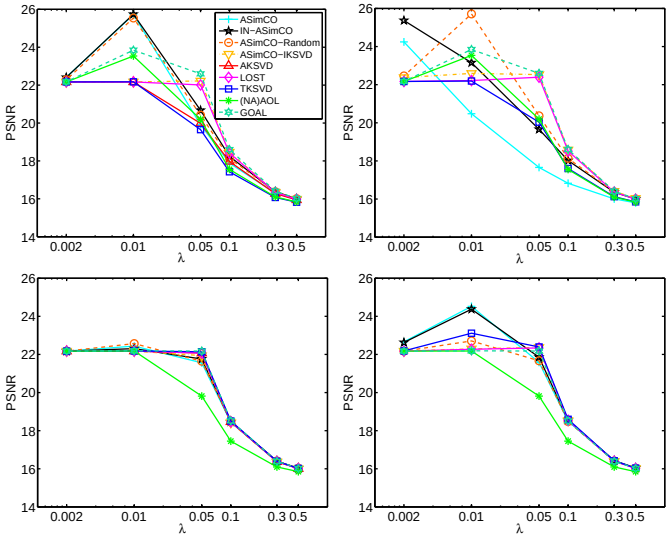


Fig. 13. Natural image denoising ($\sigma = 45$). Top row: training patches are extracted from the images in Fig. 11, with the co-sparsity $l = 40$ (left) and $l = 80$ (right). Bottom row: training patches are extracted from the natural image to be denoised, with the co-sparsity $l = 40$ (left) and $l = 80$ (right).

TABLE IV
THE BEST NATURAL IMAGE DENOISING RESULTS (PSNR IN DECIBELS).

σ = 12.8				
Training data type	Type I		Type II	
co-sparsity l	40	80	40	80
ASimCO	31.45	31.17	29.68	30.79
IN-ASimCO	31.40	30.91	29.60	30.76
ASimCO-Random	31.29	31.25	31.18	31.22
ASimCO-IKSVD	30.22	31.26	29.11	30.17
AKSVD	28.12	—	27.89	—
LOST	29.47	31.05	28.93	29.84
TKSVD	29.59	30.17	29.20	30.05
(NA)AOL	31.47		29.57	
GOAL	31.53		29.56	

σ = 45				
Training data type	Type I		Type II	
co-sparsity l	40	80	40	80
ASimCO	25.73	24.24	22.44	24.52
IN-ASimCO	25.74	25.37	22.30	24.37
ASimCO-Random	25.54	25.71	22.57	22.71
ASimCO-IKSVD	22.22	22.53	22.17	22.37
AKSVD	22.17	—	22.18	—
LOST	22.17	22.39	22.17	22.27
TKSVD	22.17	22.19	22.18	23.11
(NA)AOL	23.54		22.18	
GOAL	23.85		22.19	

VIII. ACKNOWLEDGEMENT

The authors thank the Associate Editor and the anonymous reviewers for their contributions to improving the quality of the paper.

REFERENCES

[1] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of Gaussians in the wavelet domain," *IEEE Trans. Image Process.*, vol. 12, no. 11, pp. 1338–1351, 2003.

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proc. Assoc. Comput. Mach. Spec. Interest Group Comput. Graph.*, 2000, pp. 417–424.

[3] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Comput. Graph. Appl.*, vol. 22, no. 2, pp. 56–65, 2002.

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

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

[6] S. Nam, M. E. Davies, M. Elad, and R. Gribonval, "The cosparse analysis model and algorithms," *Appl. Comput. Harmon. Anal.*, vol. 34, no. 1, pp. 30–56, 2013.

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

[8] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[9] K. Engan, S. Aase, and J. Hakon-Husoy, "Method of optimal directions for frame design," in *Proc. Int. Conf. Acoust., Speech, and Signal Process.*, vol. 5, 1999, pp. 2443–2446.

[10] W. Dai, T. Xu, and W. Wang, "Simultaneous codeword optimization (SimCO) for dictionary update and learning," *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6340–6353, 2012.

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

[12] Y. C. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. 27th Asilomar Conf. Signals, Syst., Comput.*, 1993, pp. 40–44.

[13] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 1094–1121, 2012.

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

[15] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.

[16] 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.

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

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

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

[20] S. Ravishankar and Y. Bresler, "Learning overcomplete sparsifying transforms for signal processing," in *Proc. Int. Conf. Acoust., Speech, and Signal Process.*, 2013, pp. 3088–3092.

[21] E. M. Eksioglu and O. Bayir, "K-SVD meets transform learning: Transform K-SVD," *IEEE Signal Process. Lett.*, vol. 21, no. 3, pp. 347–351, 2014.

[22] M. Kleinsteuber and H. Shen, "Blind source separation with compressively sensed linear mixtures," *IEEE Signal Process. Lett.*, vol. 19, no. 2, pp. 107–110, 2012.

[23] J. Dong, W. Wang, and W. Dai, "Analysis SimCO: A new algorithm for analysis dictionary learning," in *Proc. Int. Conf. Acoust., Speech, and Signal Process.*, 2014, pp. 7193–7197.

[24] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[25] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.

[26] T. Tao, *Analysis (Texts and Readings in Mathematics)*. Hindustan Book Agency, 2006.

[27] B. Mailhé, D. Barchiesi, and M. D. Plumbley, "INK-SVD: Learning incoherent dictionaries for sparse representations," in *Proc. Int. Conf. Acoust., Speech, and Signal Process.*, 2012, pp. 3573–3576.

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

[29] V. Abolghasemi, S. Ferdowsi, and S. Sanei, "Fast and incoherent dictionary learning algorithms with application to fMRI," *Signal, Image and Video Processing*, vol. 9, no. 1, pp. 147–158, 2015.

[30] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[31] K.-C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, 2005.