The Iteration-Tuned Dictionary for Sparse Representations

Joaquin Zepeda [#1], Christine Guillemot [#2], Ewa Kijak [*3]

[#] *INRIA Centre Rennes - Bretagne Atlantique*
Campus de Beaulieu, 35042 Rennes Cedex, FRANCE
[1] joaquin.zepeda@irisa.fr
[2] christine.guillemot@irisa.fr

[*] *Université de Rennes 1, IRISA*
Campus de Beaulieu, 35042 Rennes Cedex, FRANCE
[3] ewa.kijak@irisa.fr

***Abstract*—We introduce a new dictionary structure for sparse representations better adapted to pursuit algorithms used in practical scenarios. The new structure, which we call an *Iteration-Tuned Dictionary* (ITD), consists of a set of dictionaries each associated to a single iteration index of a pursuit algorithm. In this work we first adapt pursuit decompositions to the case of ITD structures and then introduce a training algorithm used to construct ITDs. The training algorithm consists of applying a K-means to the $(i-1)$-th residuals of the training set to thus produce the i-th dictionary of the ITD structure. In the results section we compare our algorithm against the state-of-the-art dictionary training scheme and show that our method produces sparse representations yielding better signal approximations for the same sparsity level.**

I. Introduction

Sparse decompositions have become a very active research topic in recent years. A sparse decomposition system is comprised of (*i*) an overcomplete basis (called the *dictionary*) and (*ii*) an algorithm that selects a small number of basis vectors (called the *atoms*) and uses them to form a linear combination approximating the signal being decomposed. The representation is termed sparse because generally the selected atoms will not span the entire signal space. Indeed the sparsity of the representation can be quantified as the number of atoms selected.

Much research effort has been devoted to taking advantage of sparsity in various image processing applications. For example: Guleryuz shows how adequate selection of a sparse support can serve to reconstruct missing blocks in textured areas [1]. Elad *et al.* use sparse decompositions to achieve state-of-the art image and video denoising [2], [3]. Their proposed K-SVD dictionary is also an important building block for a facial image codec that outperforms JPEG2000 by a wide margin at low bit-rates [4]. All this applications of sparse representations share the assumption that the sparse decomposition system (including the dictionary and the atom-selection algorithm) will produce good approximations of the input signal with a very small number of atoms, an assumption referred to as the *sparseland* assumption [2].

In order to better satisfy the sparseland assumption, several papers have addressed the issue of training dictionaries to thus tailor them to specific signal classes. One of the more recent dictionary training algorithms is the K-SVD algorithm proposed by Aharon *et al.* [5]. The algorithm uses a pursuit scheme to classify training vectors. An error matrix is then assembled for each atom from the residuals associated to training vectors in the atom's class. Each atom is then updated to minimize the related representation error.

The K-SVD algorithm illustrates the standard formulation addressed by the various existing dictionary training schemes: The aim is always to build a pool of atoms that will then be made available to an oracle sparse representation algorithm that will extract the hopefully correct atoms from the pool. In practice, however, the sparse representation algorithm will proceed in an iterative manner, choosing one atom at a time while all previous atoms are fixed (examples of such algorithms include matching pursuit and its variants [6], [7], [8]). Note in particular that an iterative approach assumes that atoms selected in latter iterations will differ (be linearly independant, for the case of [7] and [8]) from those chosen in previous iterations as otherwise they will not add new information to the representation. This observation motivates our approach: instead of assuming an oracle target sparse representation algorithm when training a dictionary, we acknowledge the iterative nature of practical decomposition schemes and the fact that atoms chosen in latter iterations need to differ from those chosen previously. We thus propose building a layered dictionary structure where the i-th layer will contain a dictionary $\mathbf{D}^i$ (used during the i-th iteration) having atoms that differ from the atoms of dictionaries $\mathbf{D}^j,\ j=1,\ldots,i-1$ used in previous iterations. We refer to this structure as an *Iteration-Tuned Dictionary* (ITD) as it will be the case that the dictionary from the i-th ITD layer will display spatial frequency components that vary with i.

In the remainder of the paper, we begin with a background discussion on sparse decompositions that will further lay down the notational groundwork for subsequent developments. In

MMSP'10, October 4-6, 2010, Saint-Malo, France.
978-1-4244-8112-5/10/$26.00

Section III, we then introduce our new dictionary construction scheme. In the results section (Section IV) we present visual illustrations of the various ITD dictionary layers to show how frequency components are structured across layers. We also show that ITDs better satisfy the sparseland assumption than the state-of-the-art K-SVD dictionary. We then conclude our paper in Section V.

II. Background

Sparse representations are based on an overcomplete (full-rank and fat) matrix $\mathbf{D}$ referred to as the *dictionary*; the columns of $\mathbf{D}$ are called *atoms* [6], [7], [8], [9]. Given said dictionary, a sparse representation $\mathbf{x}$ of a signal vector $\mathbf{y} \in \mathbb{R}^d$ defines a linear combination of a few atoms of $\mathbf{D}$. The selection of the atoms is specified by the position of the non-zero coefficients of $\mathbf{x}$, while the coefficients themselves are the weights of the linear combination. We will let $|\mathbf{x}|_0$ denote the number of non-zero coefficients of $\mathbf{x}$. Using this notation, we can express the construction of the sparse representation $\mathbf{x}$ as follows:

$$\underset{\mathbf{x}}{\operatorname{argmin}} |\mathbf{y} - \mathbf{D}\mathbf{x}| \text{ s.t. } |\mathbf{x}|_0 = L_{\mathbf{y}}. \tag{1}$$

The formulation specifies an $\mathbf{x}$ that minimizes the representation error of the resulting reconstruction $\hat{\mathbf{y}}^{L_{\mathbf{y}}} = \mathbf{D}\mathbf{x}$ under a sparsity constraint $L_{\mathbf{y}}$ chosen, for example, to satisfy a predetermined maximum error. In our work, each non-zero entry of $\mathbf{x}$ will be denoted by an atom-index/coefficient pair (a_i, γ_i), $i = 1, \ldots, L_{\mathbf{y}}$. For convenience, we group the atom indices a_i to form the ordered set

$$\{a_i\} = \{a_1, \ldots, a_{L_{\mathbf{y}}}\}. \tag{2}$$

The coefficients can be grouped similarly to form the vector

$$\mathbf{\Gamma}^{L_{\mathbf{y}}} = \begin{bmatrix} \gamma_1 & \ldots & \gamma_{L_{\mathbf{y}}} \end{bmatrix}^{\mathrm{t}}. \tag{3}$$

Thus the representation $\mathbf{x}$ is uniquely determined by the set of atom indices and the corresponding coefficient vector:

$$\{a_i | a_i \in [1, \ldots N], i = 1, \ldots, L_{\mathbf{y}}\}, \mathbf{\Gamma}^{L_{\mathbf{y}}}, \tag{4}$$

where N is the number of atoms in the dictionary $\mathbf{D}$.

In practice, the solution to (1) is obtained using a greedy algorithm such as *Matching Pursuit* (MP) [6]. The algorithm proceeds by selecting a single atom-index/coefficient pair (a_i, γ_i) at each iteration i. Given the first $i \leq L_{\mathbf{y}}$ (a_i, γ_i) pairs, we can express the corresponding approximation $\hat{\mathbf{y}}^i$ as

$$\hat{\mathbf{y}}^i = \sum_{j=1}^{i} \gamma_j \cdot \mathbf{d}_{a_j} \tag{5}$$

and the corresponding residual $\mathbf{r}^i$ at the output of this iteration as

$$\mathbf{r}^i = \mathbf{y} - \hat{\mathbf{y}}^i, \tag{6}$$

$$= \mathbf{r}^{i-1} - \gamma_i \cdot \mathbf{d}_{a_i}, \tag{7}$$

where by convention we let $\mathbf{r}^0 = \mathbf{y}$. The MP algorithm proceeds in the i-th iteration by choosing the (a_i, γ_i) pair that minimizes the norm of the output residual $\mathbf{r}^i$. Formally, this can be expressed as:

$$\underset{(a_i, \gamma_i)}{\operatorname{argmin}} |\mathbf{r}^{i-1} - \gamma_i \cdot \mathbf{d}_{a_i}| \iff \tag{8a}$$

$$a_i = \underset{a}{\operatorname{argmax}} |\mathbf{d}_a^{\mathrm{t}} \cdot \mathbf{r}^{i-1}|, \tag{8b}$$

$$\gamma_i = \mathbf{d}_{a_i}^{\mathrm{t}} \cdot \mathbf{r}^{i-1}, \tag{8c}$$

where superscript t denotes the vector transpose.

III. Iteration-tuned dictionaries

The crux of our algorithm is that, when decomposing a signal $\mathbf{y}$ using MP, a different dictionary $\mathbf{D}^i$ is used at each iteration i, one that is *tuned* to the spatial frequency components associated to the space of residual vectors $\mathbf{r}^{i-1}$. We will refer to the dictionary $\mathbf{D}^i$ as belonging to the i-th *layer* of an *Iteration-Tuned Dictionary* (ITD).

If we assume for now that we are given a set of ITD dictionaries $\mathbf{D}^i$ $\forall i$, ITD signal decomposition will follow the same procedure as (8) by letting $\mathbf{d}_{a_i}$ instead denote the a_i-th atom of dictionary $\mathbf{D}^i$. Adapting the expression (4) corresponding to fixed-dictionary sparse representations, those obtained using ITDs will have the form

$$\{a_i | a_i \in [1, \ldots N_i], i = 1, \ldots, L_{\mathbf{y}}\}, \mathbf{\Gamma}^{L_{\mathbf{y}}}, \tag{9}$$

where N_i is the number of atoms in the dictionary of the i-th layer.

Regarding the ITD layer dictionaries: We will tune layer dictionaries $\mathbf{D}^i$ by training them on the residuals $\{\mathbf{r}^{i-1}\}$ at the output of the previous layer. We derive the procedure by first formulating the problem optimally in the subsequent discussion.

A. Construction of iteration-tuned dictionaries

Given some representative training set $\{\mathbf{y}\}$, the construction of the ITD layer dictionaries $\mathbf{D}^i$ can be expressed as the minimization of the aggregate representation error of the training vectors $\mathbf{y}$:

$$\underset{\{\mathbf{D}^i | i=1,\ldots,L_M\}}{\operatorname{argmin}} \sum_{\{\mathbf{y}\}} \min_{\{(a_i, \gamma_i)\}} \left| \mathbf{y} - \begin{bmatrix} \mathbf{d}_{a_1} & \ldots & \mathbf{d}_{a_{L_{\mathbf{y}}}} \end{bmatrix} \mathbf{\Gamma}^{L_{\mathbf{y}}} \right|^2, \tag{10}$$

where we let L_M denote the maximum layer index (we discuss the selection of L_M shortly).

The above formulation does not have an evident solution: the difficulty in jointly selecting a large number of layer dictionaries $\mathbf{D}^i$ is further complicated by the optimal ITD sparse decomposition in the inner minimization. Since in practice the inner minimization will be solved using a greedy approach such as MP in (8), we substitute the inner minimization in (10) by the simplified MP decomposition problem in (8a). The MP cost function has the interesting property that it is only concerned with a single layer dictionary $\mathbf{D}^i$ at a time. Since the MP formulation requires the residual $\mathbf{r}^{i-1}$ at the output of the previous layer, we need to assume that, when constructing a given dictionary $\mathbf{D}^i$, all previous dictionaries $\mathbf{D}^1, \ldots, \mathbf{D}^{i-1}$ have already been built and the corresponding

atom-index/coefficient pairs $(a_j, \gamma_j),\ j = 1, \ldots, i-1$, have already been selected. The resulting simplification of the dictionary construction scheme in (10) is given below:

$$\underset{\mathbf{D}^i}{\operatorname{argmin}} \sum_{\{\mathbf{r}^{i-1}\}} \min_{(a_i,\gamma_i)} \left|\mathbf{r}^{i-1} - \gamma_i \cdot \mathbf{d}_{a_i}\right|^2, \tag{11}$$

B. The optimal dictionary for single-atom representations

Inspection of the ITD formulation in (11) reveals that each dictionary $\mathbf{D}^i$ will be used to obtain a single (a_i, γ_i) pair for each $\mathbf{r}^{i-1}$. As we now show, a dictionary thus employed can be constructed optimally using the K-means algorithm under the projection distance.

We will find it useful in the subsequent discussion to rewrite (11) by grouping the summation into N_i terms according to the atom $\mathbf{d}_{a_i},\ a_i \in \{1, \ldots, N_i\}$, selected for each $\mathbf{r}^{i-1}$. We first define the *class matrix* $\mathbf{R}_{a_i}$ consisting of those $\mathbf{r}^{i-1}$ that use atom $\mathbf{d}_{a_i}$ in their single-atom representation,

$$\mathbf{R}_{a_i} \triangleq \operatorname{cols}(\{\mathbf{r}^{i-1} \big| \operatorname*{argmax}_{\alpha_i} \left|(\mathbf{d}_{\alpha_i})^{\mathrm{t}} \cdot \mathbf{r}^{i-1}\right| = a_i\}), \tag{12}$$

where the notation $\operatorname{cols}(\{\mathbf{a}\})$ denotes the matrix of size $\dim(\mathbf{a}) \times |\{\mathbf{a}\}|$ with vectors $\mathbf{a}$ stacked side-by-side as columns. With the help of the class matrix $\mathbf{R}_{a_i}$, the resulting form of (11) is

$$\underset{\mathbf{D}^i}{\operatorname{argmin}} \sum_{a_i=1}^{N_i} \min_{\boldsymbol{\omega}} |\mathbf{R}_{a_i} - \mathbf{d}_{a_i}\boldsymbol{\omega}^{\mathrm{t}}|_F^2, \tag{13}$$

where (*i*) the square Frobenius norm $|\cdot|_F^2$ sums the square of all matrix elements and (*ii*) the vector $\boldsymbol{\omega}$ regroups the coefficients γ_i (*cf.* (11)) of each training vector $\mathbf{r}^{i-1}$ contained in $\mathbf{R}_{a_i}$. Note that $\mathbf{R}_{a_i}$ itself depends on the optimization parameter $\mathbf{D}^i$ through (12).

A practical solution of the problem in (13) can be seen to correspond to an instance of the K-means algorithm under the projection distance: The dictionary is initialized using, for example, a random selection of the training vectors $\mathbf{r}^{i-1}$ without repetition. The method then proceeds iteratively where each iteration consists of two steps: (*i*) classifying each and all vectors $\mathbf{r}^{i-1}$ into class matrices $\mathbf{R}_{a_i}$ and (*ii*) updating the corresponding atom $\mathbf{d}_{a_i}$ so as to best represent the corresponding class $\mathbf{R}_{a_i}$.

The classification stage consists of stacking all the vectors $\mathbf{r}^{i-1}$ along the columns of one of N_i class matrices $\mathbf{R}_{a_i}$ following (12).

In the atom update stage, the new atom $\mathbf{d}_{a_i}$ associated to each $\mathbf{R}_{a_i}$ is chosen to best represent the columns of $\mathbf{R}_{a_i}$:

$$\underset{\mathbf{d}_{a_i}}{\operatorname{argmin}} \min_{\boldsymbol{\omega}} |\mathbf{R}_{a_i} - \mathbf{d}_{a_i}\boldsymbol{\omega}^{\mathrm{t}}|_F^2. \tag{14}$$

The term $\mathbf{d}_{a_i}\boldsymbol{\omega}^{\mathrm{t}}$ can be seen to be a rank-1 approximation of $\mathbf{R}_{a_i}$ and thus the solution $\mathbf{d}_{a_i}$ is just the left singular vector of $\mathbf{R}_{a_i}$ associated to the strongest singular value [10].

C. Construction of subsequent layers

Following the construction of a given dictionary $\mathbf{D}^i$, the input training residuals $\mathbf{r}^{i-1}$ are decomposed following (8) using the newly constructed dictionary. The residual set $\{\mathbf{r}^i\}$ thus obtained is then used to train the subsequent dictionary $\mathbf{D}^{i+1}$.

One question that comes to mind in the above-described ITD training scheme is the stopping criterion that determines the maximum layer index L_M. As for the case of the fixed-dictionary MP scheme [6], the i-th residual $\mathbf{r}^i$ of an ITD decomposition will not reduce to zero even for arbitrarily large values of i. As done for the OMP [7] and OOMP [8] versions of the MP decomposition rules in (8), one could force null residuals at $i = d$ (where d is the input signal dimension) by constraining each residual $\mathbf{r}^i,\ i = 1, \ldots, d$ to be orthogonal to all previously selected atoms. This approach would set a practical bound $L_M = d$ on the number of ITD layers. Another approach consists of choosing a number of atoms $L_{\mathbf{y}}$ independently for each $\mathbf{y}$ so as to satisfy a predetermined approximation error. The residual training sets $\{\mathbf{r}^i\}$ can be pruned accordingly by discarding those residuals with $i \geq L_{\mathbf{y}}$. The maximum ITD layer index will then be $L_M = \max_{\mathbf{y}}(L_{\mathbf{y}})$. In practice a combination of these two approaches can be used.

IV. Results

We use to 545 frontal-pose facial images of different individuals taken from the FERET database [11]. A subset of 100 of these images will comprise the test set and the remaining images will comprise a dictionary training set. We chose the FERET database because it provides a large number of uncompressed images and because it is comparable to the facial database used by the K-SVD authors [5] (we compare ITDs to K-SVD dictionaries). Throughout our experiments we use signal vectors $\mathbf{y}$ consisting of vectorized versions of 8×8 blocks taken from each image on a regular grid. All ITDs will be built with a number of atoms that is fixed for all layers, $N_i = N\ \forall\ i$. The maximum layer index will be set to $L_M = 32 \geq \max_{\mathbf{y}}(L_{\mathbf{y}})$ (we do not prune residual training sets).

We carry out experiments with the following two purposes: Firstly, we verify (through visual inspection) that the spatial frequencies comprising ITD layer dictionaries $\mathbf{D}^i$ increase with the iteration index i. Secondly, we show that ITDs better satisfy the sparseland assumption (with lower computational complexity) when compared to the state-of-the-art K-SVD dictionary [5]. Both of these qualities where motivating design goals presented in the introduction (Section I).

In Fig. 1 we illustrate the first six layers of an ITD constructed following (11). The ITD layer dictionaries $\mathbf{D}^i$ consist each of $N = 128$ atoms, and thus each dictionary is 2-times overcomplete ($128/8^2 = 2$). Note that the training process induces a structuring of spatial frequencies across the various layers: earlier layers have stronger low-frequency content, with high-frequency content increasing along with the layer index i.

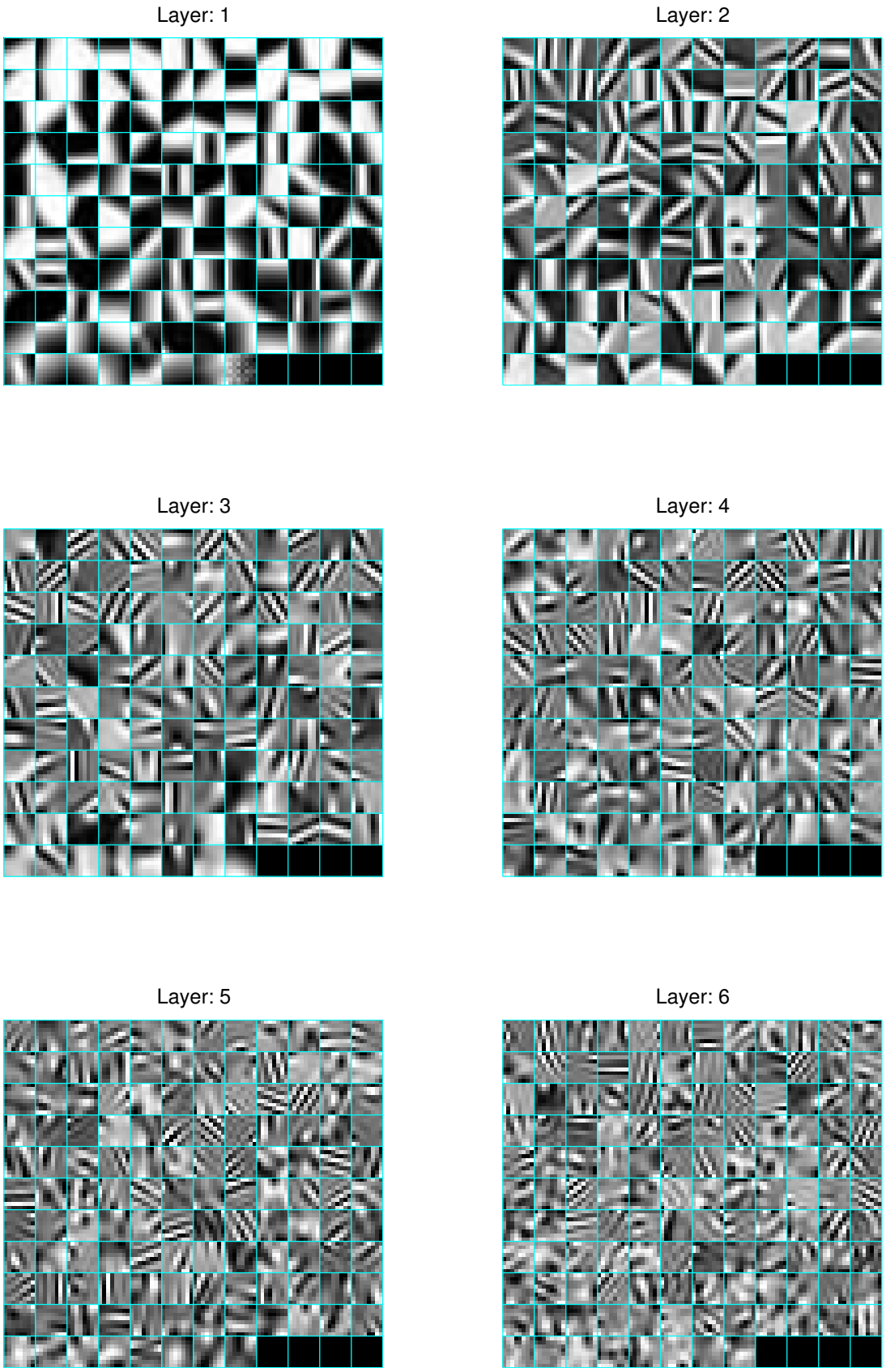

Fig. 1. The first six layers of an ITD dictionary having $N = 128$ atoms in each layers.

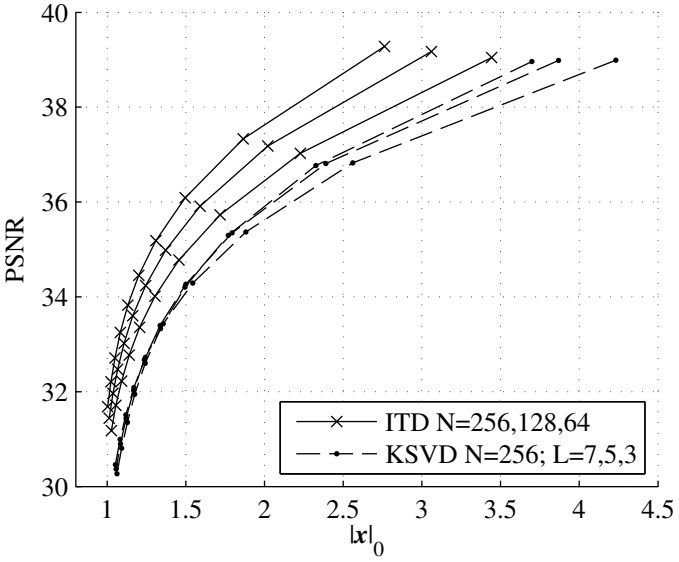

Fig. 2. ITD vs. K-SVD: Mean PSNR vs. mean l-0 norm when using the error threshold stopping criteria in (15) ($\epsilon = 4, 6, 8, \ldots, 22$). Signal decomposition is carried using OMP for K-SVD and MP for ITD. All K-SVD dictionaries have $N = 256$ atoms and were trained using OMP with stopping criterion $L = 7, 5, 3$ (dashed curves from top to bottom, at right). The ITD dictionaries have $N = 256, 128, 64$ atoms (solid curves from top to bottom), with N fixed for all layers i.

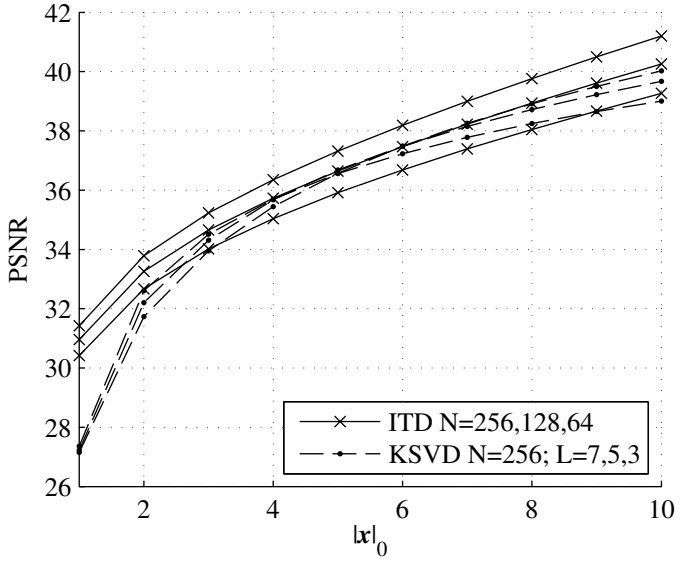

Fig. 3. ITD vs. K-SVD: Mean PSNR vs. mean l-0 norm when using the sparsity stopping criteria in (16) ($L_{\mathbf{y}} = 1, 2, \ldots, 10$). Signal decomposition is carried using OMP for K-SVD and MP for ITD. All K-SVD dictionaries have $N = 256$ atoms and were trained using OMP with stopping criterion $L = 7, 5, 3$ (dashed curves from top to bottom, at right). The ITD dictionaries have $N = 256, 128, 64$ atoms (solid curves from top to bottom), with N fixed for all layers i.

In Fig. 2 and Fig. 3 we evaluate how well ITDs satisfy the sparseland model by plotting the mean l-0 norm $|\mathbf{x}|_0$ versus the mean approximation PSNR $20\log_{10}(\frac{255}{|\mathbf{y}-\hat{\mathbf{y}}^{L_{\mathbf{y}}}|})$. The mean is taken over all patches of all test images (a total of close to 6.5×10^5 patches). Three different curves are shown for the ITD scheme. They correspond to three different values of the number N of atoms per layer. From top to bottom in either figure (solid-line curves), these values are $N = 256, 128, 64$.

As a reference, in both Fig. 2 and Fig. 3 we also plot the same curves corresponding to the state-of-the-art K-SVD dictionary [5]. We trained K-SVD dictionaries consisting of $N = 256$ atoms using the code provided by the authors. The training is carried out using OMP [7] with a sparsity-based stopping criterion $|\mathbf{x}|_0 = L$. The three K-SVD curves shown correspond to three different values of L. From top to bottom in either figure (dashed-line curves, with order according to the rightmost point in the curves), these values are $L = 7, 5, 3$.

The difference between Fig. 2 and Fig. 3 pertains to the stopping criterion used to determine $L_{\mathbf{y}}$ during sparse decomposition of the test images. In Fig. 2 we choose the smallest $L_{\mathbf{y}}$ that satisfies a maximum RMSE threshold ϵ:

$$\underset{L_{\mathbf{y}}}{\operatorname{argmin}} L_{\mathbf{y}} \text{ s.t. } |\mathbf{y} - \hat{\mathbf{y}}^{L_{\mathbf{y}}}|^2 \leq d \cdot \epsilon^2, \tag{15}$$

where we recall that d is the signal dimension, $\mathbf{y} \in \mathbb{R}^d$. In Fig. 3 we use an $L_{\mathbf{y}}$ that is fixed to an integer constant C:

$$L_{\mathbf{y}} = C \ \forall \ \mathbf{y}. \tag{16}$$

For clarity we point out that $L_{\mathbf{y}}$ denotes the sparsity of test image decompositions while L (indicated in figure legends) denotes the sparsity of decompositions used during K-SVD training. Regarding the choice of stopping criterion (15) or (16), we note that in practical scenarios the error threshold stopping criterion in (15) is more likely to be used because it adapts the sparsity level to each signal vector. Regardless of the stopping criterion used, ITD decompositions are carried out using the ITD adaptation of the MP algorithm in (8). K-SVD decompositions are carried out using the more powerful OMP algorithm.

From Fig. 2 we can conclude that ITD better satisfies the sparseland model relative to K-SVD: For the same number of atoms $N = 256$, the gain can be as high as 2 dB. Note that, when using the same number of dictionary atoms N in both ITD and K-SVD dictionaries, the resulting decompositions will have comparable computational complexities (K-SVD decomposition complexity will be higher because the OMP coefficient calculation rule is more complex than the MP rule). Yet the ITD performance benefit holds even for values of N that are 2 and 4 times smaller ($N = 128, 64$, respectively) than that corresponding to K-SVD ($N = 256$). For these smaller $N = 128, 64$, ITD will enjoy an even higher complexity gain (besides the performance gain illustrated by the curves).

In Fig. 3 we show that the ITD performance benefit over K-SVD also holds when using the sparsity stopping criterion in (16), but only for the ITD having the same number of atoms per layer $N = 256$ as the K-SVD dictionary (*i.e.,* for the case of comparable decomposition complexity). For smaller ITD dictionaries ($N = 128, 64$), the performance falls below that of K-SVD for sparsity values in the neighborhood of 5. Note however that, for smaller sparsity values in the range $1 - 2$,

ITD still outperforms K-SVD even for $N = 128, 64$. One would expect that, given the PSNR difference between ITD and K-SVD at these lower sparsity values, ITD will result in a greater number of weak residual vectors (*i.e.,* residual vectors having small magnitude) starting at iteration $i = 3$. Adding (a_i, γ_i) pairs to weak residuals results in marginal PSNR improvement. Given the greater number of ITD weak residuals from $i = 3$ and onwards, we can expect (and indeed observe) a reduced curve slope for ITD when compared to K-SVD. Note that this is not observed in Fig. 2 because, when building the sparse decompositions corresponding to that figure, weak residuals will no longer produce (a_i, γ_i) pairs as a consequence of the error threshold stopping criterion in (15).

Finally we note how Fig. 2 and Fig. 3 make evident the dependance of fixed-dictionary schemes on the target sparsity level: training K-SVD dictionaries with different values of L results in dictionaries having performance that depends on the desired sparsity. In the figures this results in K-SVD curves with relative positions that become inverted when moving from left to right along the horizontal axis. Real life scenarios often require algorithms that are flexible and capable of performing well at all sparsity levels. In the context of image and video compression, for example, such flexibility enables representations with bit-streams that scale with the available bandwidth. The ITD algorithm manages to better address scalability because each dictionary is optimized to a single iteration index i and hence does not depend on a pre-specified signal sparsity.

V. Conclusion

In this paper we introduced the *Iteration-Tuned Dictionary* (ITD) for sparse representations. ITDs are better suited for sparse representations using pursuit algorithms because they provide a different dictionary $\mathbf{D}^i$ for each pursuit iteration i. Each $\mathbf{D}^i$ is tuned to the spatial frequency components of the residual class at the output of the previous iteration $i - 1$. We first presented an ITD adaptation of matching pursuit and then proposed a corresponding ITD training algorithm. Our resulting setup produced reconstructions displaying up to 2 dBs of improvement over reconstructions based on state-of-the-art trained dictionary schemes at the same sparsity level.

References

[1] O. Guleryuz, "Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising-part I: theory," *IEEE Transactions on Image Processing*, vol. 15, pp. 539–554, March 2006.

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

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

[4] O. Bryt and M. Elad, "Compression of facial images using the K-SVD algorithm," *J. Vis. Comun. Image Represent.*, vol. 19, no. 4, pp. 270–282, 2008.

[5] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, pp. 4311–4322, 2006.

[6] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on*, vol. 41, no. 12, pp. 3397–3415, Dec 1993.

[7] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. 27th Asilomar Conference on Signals, Systems and Computers*, A. Singh, Ed. IEEE Comput. Soc. Press, Los Alamitos, CA, 1993.

[8] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach," *IEEE Signal Processing Letters*, vol. 9, no. 4, pp. 137–140, April 2002.

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

[10] V. C. Klema and A. J. Laub, "The singular value decomposition: Its computation and some applications," *IEEE Transactions on Automatic Control*, vol. AC-25, no. 2, pp. 164–176, April 1980.

[11] P. J. Phillips, H. Moon, P. J. Rauss, and S. Rizvi, "The FERET evaluation methodology for face recognition algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, October 2000.