Learning Tensor-structured Dictionaries with Application to Hyperspectral Image Denoising
Cassio F. Dantas, Jeremy E Cohen, Rémi Gribonval

To cite this version:

HAL Id: hal-02126782
https://inria.hal.science/hal-02126782v2
Submitted on 5 Jun 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Learning Tensor-structured Dictionaries with Application to Hyperspectral Image Denoising

Cássio F. Dantas, Jérémie E. Cohen, Rémi Gribonval

Univ Rennes, Inria, CNRS, IRISA
Rennes, France

Abstract—Dictionary learning, paired with sparse coding, aims at providing sparse data representations, that can be used for multiple tasks such as denoising or inpainting, as well as dimensionality reduction. However, when working with large data sets, the dictionary obtained by applying unstructured dictionary learning methods may be of considerable size, which poses both memory and computational complexity issues. In this article, we show how a previously proposed structured dictionary learning model, HO-SuKro, can be used to obtain more compact and readily-applicable dictionaries when the targeted data is a collection of multiway arrays. We introduce an efficient alternating optimization learning algorithm, describe important implementation details that have a considerable impact on both algorithmic complexity and actual speed, and showcase the proposed algorithm on a hyperspectral image denoising task.

Index Terms—Dictionary learning, Tensor, Kronecker product, Hyperspectral imaging, Denoising

I. PROBLEM STATEMENT

Let $\mathcal{Y}$ be a collection of $n$ tensor data $\{Y_1, \ldots, Y_n\}$. For the sake of simplicity let us suppose $Y_i$ are three-mode arrays in $\mathbb{R}^{m_1 \times m_2 \times m_3}$ stacked along the fourth mode to form $Y \in \mathbb{R}^{m_1 \times m_2 \times m_3 \times n}$ - a generalization to higher-order tensors is straightforward. We are interested in the following problem, which is classically formulated for vector input data:

$$
\arg\min_{D \in S_D, x_i} \sum_{i=1}^n \| \operatorname{vec}(Y_i) - Dx_i \|_2^2 + g(x_i) 
$$

where $D \in \mathbb{R}^{m_1 m_2 m_3 \times d}$ is the dictionary which is often overcomplete ($d \geq m_1 m_2 m_3$) and belongs to a constraint set $S_D$. Function $g$ is a sparsity inducing penalty and $x_i$ is the sparse vector of coefficients describing the vectorized tensor $Y_i$ in the set of atoms $D$. Most often, $S_D$ is the set of matrices with unit Euclidean norm columns.

This problem is coined as Dictionary Learning [1], and has been extensively studied in signal processing over the last decade. When tensor data is considered, two main drawbacks emerge in this formulation: i) it is completely agnostic to the original multidimensional structure of the data; ii) data sizes $m_j$ may be relatively large (even more so the product $m_1 m_2 m_3$) and both the storage of matrix $D$ and the computation of products such as $Dx_i$ may be cumbersome.

One way to tackle the second issue is to restrict the class $S_D$ of dictionaries that are sought in problem (1). To also tackle the first mentioned issue, we previously proposed the High-Order Sum of Kronecker model (HO-SuKro) [2], a class of tensor-structured dictionaries which are particularly suited to tensorial input data and can be written as follows:

$$
D = \sum_{q=1}^r D_{1,q} \otimes D_{2,q} \otimes D_{3,q} 
$$

where $\otimes$ is the Kronecker product [4], $D_{j,q}$ are matrices in $\mathbb{R}^{m_j \times d_j}$ and $d_1 d_2 d_3 = d$. The unit-norm constraint in $D$ is handled as a post-processing step.

It can be shown that this parametrization corresponds to a low (operator-)rank constraint on the dictionary – seen as a trilinear operator acting on tensors – which appears as a natural structural assumption. To provide some extra intuition, for a fixed $q$ we can see the terms $D_{j,q}$ as linear operators acting independently in the $j$-th mode of the input data $Y$. For $r = 1$, it boils down to the classic separable operators, which include for instance the 3-dimensional discrete cosine transform (3D-DCT).

An obvious remark about HO-SuKro is that the number of parameters to represent $D$ using (2), which is $r (m_1 d_1 + m_2 d_2 + m_3 d_3)$, is much smaller than the number of entries $m_1 m_2 m_3 d_1 d_2 d_3$ as long as $r$ is small. Moreover, matrix products with the dictionary can be computed markedly faster.

In this paper, we propose an efficient algorithm to learn one such dictionary from data (Section III) and scrutinize the involved computational complexities as well as how they translate into practical speedups (Section IV). Finally, in Section V, we apply the proposed technique to a Hyperspectral image denoising task and compare its performance to some similar approaches as well as the state-of-the-art.

II. STATE OF THE ART

Modeling dictionaries in the dictionary learning problem as a Kronecker product of several smaller matrices has become a somewhat standard technique, though typically only the rank one case is considered. Early models studied Kronecker products with two terms [4], and later three or any number of terms [5]-[7]. The choice of this Kronecker structure is seldom justified using tensor algebra, but rather as a trick to reduce the number of parameters and computational complexity.

Sums of Kronecker products have been proposed for estimating covariance matrices [8], [9], with only two terms in the Kronecker products. The closest work to our proposed SuKro [10] and HO-SuKro [2] is the Kronecker representation for matrices introduced in [11], which is meant as a solution
to computing a tensor Singular Value Decomposition for structured tensors, where blocks \( D_{j,q} \) are orthogonal. It doesn’t, therefore, particularly address the dictionary learning problem.

In what follows, we will show that rather simple algorithms can be designed to estimate matrices \( D_{j,q} \). This contrasts with sometimes intricate optimization schemes employed by the references mentioned above. In particular, in our previous work on HO-SuKro [3], we proposed a projected alternating algorithm and, despite the theoretical gains in parameter size and memory requirements for the learned dictionary, the learning process itself was quite time-demanding.

III. Proposed Algorithm

The dictionary learning problem in equation (1) is a non-convex problem. However, initially ignoring the normalization constraint, the minimization with respect to \( D \) with fixed \( x \) \((\forall i \in \{1, \ldots, n\})\) is a quadratic problem which has a closed-form solution, while estimating \( x \) with known \( D \) is a sparse coding problem, which has been extensively studied over the last two decades [12], [13]. Therefore, a usual way to solve (1) is to alternate between estimating only \( D \) and estimating only \( X \). We will refer to each subproblem respectively as dictionary update and sparse coding.

Now, using the proposed Kronecker structure on \( D \), the sparse coding problem is formally unchanged. Therefore, one may use greedy heuristics such as Orthogonal Matching Pursuit (OMP) [12] or convex relaxations based algorithms such as FISTA [13]. The Kronecker structure can be exploited to positively impact the running time and complexity of these methods, see Section IV for more details.

Dictionary update, on the other hand, is heavily modified in the HO-SuKro formulation. The partial cost function, with respect to the blocks \( \{D_{j,q}\}_{q=1}^{r} \), may be written as follows:

\[
 f(D_{j,q}) = \sum_{i=1}^{n} \|\text{vec}(Y_{i}) - \sum_{q=1}^{r} (D_{1,q} \otimes D_{2,q} \otimes D_{3,q}) x_{i} \|^{2}_{2} \tag{3}
\]

Of course, one may think of multiple techniques to minimize (3), for instance gradient-based approaches, second order methods, or block-coordinate descent. In fact, because of the similarity of (3) with the Tucker Decomposition problem (see [14] for an overview), a first approach we tried was to minimize (3) in an alternating fashion, fixing all parameters but one elementary block \( D_{j,q} \). This leads to a quadratic problem, which can be solved in closed form.

However, in the spirit of Alternating Least Squares algorithms (ALS), it is possible to gather all elementary blocks \( D_{j,q} \) for a fixed mode \( j \) and rather alternate between only three macro-blocks (one per mode) \( \Delta_{j} = \{D_{j,q}\}_{q=1}^{r} \), \( 1 \leq j \leq 3 \), and still have a closed-form solution.

A. Quadratic partial cost function explained

Using, for instance, the first-mode unfolding as in [3] denoted \( Y_{[1]} \in \mathbb{R}^{m_{1} \times m_{2} \times m_{3}} \) for \( Y \) and \( X_{[1]} \in \mathbb{R}^{d_{1} \times d_{2} \times d_{3}} \) for \( X \), where \( X \in \mathbb{R}^{d_{1} \times d_{2} \times d_{3} \times n} \) is a tensor built similarly to \( Y \) by stacking all coefficient tensors \( X_{i} = \text{tens}(x_{i}) \in \mathbb{R}^{d_{1} \times d_{2} \times d_{3}} \), one can rewrite (3) as

\[
 f(D_{j,q}) = \|Y_{[1]} - \sum_{q=1}^{r} D_{1,q} X_{[1]} (D_{2,q} \otimes D_{3,q} \otimes I_{n})^{T} \|_{F}^{2} \tag{4}
\]

The identity matrix \( I_{n} \) that appears in the Kronecker structure is an implicit way to apply the dictionary to \( X_{i} \) simultaneously for all \( i \in \{1, \ldots, n\} \). Equation (4), in turn, can be seen as a matrix factorization problem

\[
 f(D_{j,q}) = \|Y_{[1]} - \Delta_{1} U\|_{F}^{2} \tag{5}
\]

where \( \Delta_{1} = \{D_{1,1}, \ldots, D_{1,r}\} \in \mathbb{R}^{m_{1} \times m_{1}} \) contains the horizontally-stacked matrices \( D_{1,q} \) with \( 1 \leq q \leq r \), and \( U = [U_{1}; \ldots; U_{r}] \in \mathbb{R}^{rd_{1} \times m_{2} m_{3}} \) is the vertically-stacked right-hand terms with \( U_{q} = X_{[1]} (D_{2,q} \otimes D_{3,q} \otimes I_{n})^{T} \).

Note that if \( f \) is quadratic with respect to \( \Delta_{1} \), and an optimal solution for fixed \( U \) is given in closed form by

\[
 \hat{\Delta}_{1} = \text{argmin}_{\Delta} \|Y_{[1]} - \Delta_{1} U\|_{F}^{2} = Y_{[1]} U^{\dagger} \tag{6}
\]

where \( U^{\dagger} = U^{T} (U U^{T})^{-1} \) is the right inverse of \( U \). A similar reasoning applies to the other modes.

B. ALS for training

Since we have shown that \( f \) is quadratic with respect to blocks \( \Delta_{j} \), let us now derive formally an algorithm to estimate these blocks \( \Delta_{j} \) using Alternating Least Squares, as well as \( X \). ALS is reported as a baseline algorithm for unconstrained PARAFAC [14]. It is also a well studied algorithm for computing the Tucker decomposition, a decomposition to which our training problem reduces to when \( r = 1 \) [6]. The proposed ALS is summarized in Algorithm[7]. Factors \( \Delta_{j} \) are estimated in an alternating fashion using (6), while \( X \) is computed using any sparse coding method. Little can be said about convergence of Algorithm[1] since ALS for the training problem does not have a local convergence guarantee [14], and the sparse coding step is not even guaranteed to decrease the cost function. Yet, no convergence problems have been observed in practice.

Special attention has to be devoted to satisfy the unit-norm constraint. Like most dictionary learning algorithms, we normalize the estimated \( D = \sum_{q} D_{1,q} \otimes D_{2,q} \otimes D_{3,q} \) at each iteration. However, at line 17 of Algorithm[1] we store both the blocks \( \{D_{j,q}\}_{q=1}^{r} \) and the norms \( \Sigma \) separately to make use of the Kronecker structure in the sparse coding step. Even if this ad-hoc normalization step may marginally increase the cost function, it was not observed to decisively impact the algorithm’s performance.

IV. Complexity analysis and practical speedups

The advantages of HO-SuKro are threefold. First, as the number of parameters to estimate in \( D \) is smaller than in the usual dictionary learning framework, HO-SuKro is expected to better resist a diminution of sample size. This behavior has been observed in [2]. To be exact, the number of parameters in the Kronecker-structured \( D \) is \( r(\sum_{j} m_{j} d_{j}) \) while the full \( D \) has \( \prod_{j} m_{j} d_{j} \) entries. Therefore asymptotically this dimensionality reduction is significant if \( r \) is smaller than the product...
Algorithm 1 Alternating Least Squares for HO-SuKro

1: INPUTS: Data $\mathcal{X}$, initial $D_{j,q}$ ($j\in\{1,2,3\}, q\in\{1,\ldots,r\}$)
2: Write $D_{4,q} = I_{d_4}$ for all $q \leq r$
3: while stopping criterion is not met do
4:  \textbf{Sparse Coding}
5:  Solve $X_r = \text{argmin}_X \left\| Y_r - D X \right\|_F + g(\mathcal{X})$ using any sparse coding algorithm.
6:  \textbf{Dictionary Update}
7:  \textbf{Dictionary Update}  
7:  while update on $D_{j,q}$ is significant do
8:  for $j$ from 1 to $3$ do
9:  Set $U_q = X_{[j]} (\mathbb{E}_{\neq j} D_{j,q})^T$ for all $q \leq r$
10:  Set $U = [U_1;\ldots;U_r]$ stacked vertically
11:  $\Delta_j = Y_{[j]} U^T (U U^T)^{-1}$
12:  Set $[D_{j,1},\ldots,D_{j,r}] = \Delta_j$ stacked horizontally
13: end for
14: end while
15: \textbf{Dictionary Column Normalization}
16: $\Sigma(k,k) = 1/\|D(:,k)\|_2$ for all $k \in \{1,\ldots,d\}$
17: $D = (\sum_{q=1} D_{1,q} \otimes D_{2,q} \otimes D_{3,q}) \Sigma$
18: end while
19: OUTPUTS: Estimated blocks $D_{j,q}$, norms $\Sigma$, sparse $\mathcal{X}$

TABLE I: Computational complexity: dictionary update

<table>
<thead>
<tr>
<th>Operation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_q \otimes q \leq r$</td>
<td>$r \sum_{j\neq q} m_j 1 \prod_{l \neq q} d_l n$</td>
</tr>
<tr>
<td>$Y_j U^T$</td>
<td>$r \prod_{l \neq j} m_l d_l n$</td>
</tr>
<tr>
<td>$U U^T$</td>
<td>$r^2 \prod_{l \neq j} m_l d_l^2 n$</td>
</tr>
<tr>
<td>$(U U^T)^{-1}$</td>
<td>$(r d_j)^3$</td>
</tr>
</tbody>
</table>

B. Structured sparse coding

Sparse coding algorithms, either greedy (like OMP and its variants) or convex-relaxation-based (like Iterative Soft-Thresholding algorithm – ISTA – and its variants), can also profit from the proposed structure in the dictionary. Basically, any product with the dictionary can be replaced by a mode-product with the smaller factors $D_{j,q}$ as shown in eq. (7).

In OMP-like algorithms, the complexity of an iteration is dominated (see Table I for some empirical evidence) by the calculation of the inner product between the dictionary and the current residual in $O(\prod_j m_j d_j)$. For an HO-SuKro dictionary, this operation can be accelerated using (7) to a complexity of $O(r(\sum_j m_j)(\prod_j d_j))$. This operation is followed by a normalization using $\Sigma$ with a minor computation cost.

The mentioned accelerations can be used to speedup the learning process (sparse coding step on line 5 of Alg. 1) and also after it, once the dictionary is learned and applied repeatedly to some targeted data. In considered application (patch-based Hyperspectral image denoising), the latter corresponds to the denoising itself, using the learned dictionary.

TABLE II: Time spent in operation $D^T \rho$ for 10 iterations

<table>
<thead>
<tr>
<th>$m$</th>
<th>$d$</th>
<th>$\text{OMP Cholesky}$</th>
<th>$\text{OMP Cholesky}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6, 6</td>
<td>12, 12</td>
<td>63 %</td>
<td>93 %</td>
</tr>
<tr>
<td>8, 8</td>
<td>16, 16</td>
<td>83 %</td>
<td>93 %</td>
</tr>
<tr>
<td>10, 10</td>
<td>20, 20</td>
<td>73 %</td>
<td>93 %</td>
</tr>
</tbody>
</table>

Considering that typically $n \gg \prod_j d_j > \prod_j m_j \gg r$, the first and the third operations are the most costly. Nevertheless, thanks to the tensor structure we manage to completely avoid complexities scaling with $(\prod_j m_j)(\prod_j d_j)n$ as it would be the case for a single product $DX$ or $D^T Y$ with an unstructured dictionary. Whenever multiple mode-products are to be performed, the mode ordering can be chosen wisely to minimize the total complexity.

C. From theoretical to practical speedups

The proposed accelerations, in both dictionary update and sparse coding steps, rely on the theoretical complexity of tensor mode-products. In this section, we evaluate to which extent such theoretical gains translate into actual speedups.

As explained, a product with the HO-SuKro dictionary becomes a sequence of mode-products. When implementing such
TABLE III: Speedups in matrix-vector products

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Theoretical speedup</th>
<th>Empirical speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( r = 1 )</td>
<td>( r = 3 )</td>
</tr>
<tr>
<td>6, 6, 6</td>
<td>20.3</td>
<td>0.8</td>
</tr>
<tr>
<td>8, 8, 8</td>
<td>36.6</td>
<td>4.5</td>
</tr>
<tr>
<td>10, 10, 10</td>
<td>57.1</td>
<td>13.2</td>
</tr>
</tbody>
</table>

operations, it is necessary to repeatedly unfold the data tensor along each of its modes. Given a tensor \( X \in \mathbb{R}^{d_1 \times d_2 \times d_3 \times n} \), the mode-\( j \) unfolding is obtained by cascading two operations: (i) a permutation of the indexes of \( X \) so that mode \( j \) becomes the first dimension of the permuted tensor, and (ii) a reshape operation that unfolds all modes but the first along the second dimension. Although such operations are neglected in theory (considered \( O(1) \)), this is not the case in practice as a reorganization of the tensor entries in memory might be required.

This seemingly harmless overhead actually creates a considerable gap between theoretical and empirical speedups, especially for smaller dimensions, as exemplified in Table III. Nevertheless, some considerable acceleration is still achieved in practice, in higher dimensions. A lower-level implementation can be considered to try and tighten this gap, which would be of great interest since there is still a large acceleration potential to be exploited.

V. Experiments

A. Hyperspectral image denoising

We now evaluate the proposed Algorithm 1 in a Hyperspectral image (HSI) denoising task. The correlation along spectral bands in HSI have been found very useful to improve HSI denoising, as opposed to conventional denoising techniques based solely on 2D modeling. As a result, modern HSI denoising techniques have evolved to incorporate spectral information. Our structured dictionary framework allows us to manipulated directly the 3D data and properly exploit its original structure. Patch-based approaches, like ours, have rarely been explored in the literature so far. For a more extensive survey of this domain, we refer the reader to [17].

We will initially compare our method to other techniques based on the sparse modeling assumption: fixed sparsifying transforms (2D and 3D [18] Wavelet) and patch-based unstructured learned dictionaries (K-SVD [15]). We will see that we manage to outperform both of them. In the Hyperspectral imaging literature, another assumption besides sparsity proved to be very useful: the low-rankness of the image itself. Since we don’t take this assumption into account, we don’t manage to match the performance of state-of-the-art methods which combine both sparsity and low-rank assumptions. We still manage, nevertheless, to approach their performance, which is quite encouraging. The integration of the low-rank assumption in our model is envisioned for future work.

B. Simulation setup: a patch-based approach

Given a hyperspectral image corrupted with random Gaussian noise with standard deviation \( \sigma \) uniform over all spectral modes, we collect \( n \) 3D-patches with dimensions \( \{m_1, m_2, m_3\} \) from the noisy image to form our training data \( Y \in \mathbb{R}^{m_1 m_2 m_3 \times n} \). As a pre-treatment, each patch is re-centered to have a zero mean along its pixel values. The patches are taken uniformly-spaced and partially overlapping.

A dictionary is learned from this data with 20 iterations of alternating optimization. Sparse coding is performed by OMP with an error threshold \( \tau \) proportional to the noise level \( \sigma : \tau = \sigma \sqrt{\tau_j} \). The learned dictionary is then used to reconstruct patches with a one-pixel step. The recovered patches are averaged in the overlapping pixels along with the noisy image itself to form the final denoised image.

Two standard HSI images, from San Diego and Houston datasets, were cropped to \( 256 \times 256 \times 100 \) pixels and used in the experiments. Reported results were obtained using 100000 training samples, which corresponds roughly to a 4-pixel step between adjacent patches. Although some performance improvement was observed when increasing this number, it does not compensate for the resulting rise in training time. Dictionaries were initialized with a 3D-DCT. In HO-SuKro’s \( r > 1 \) case, the remaining terms were initialized with unit-norm Gaussian random matrices, and revealed to be quite robust to the initialization. The ALS loop (line 7 in Alg. 1) was carried over until the update on the blocks \( D_{j,q} \) fell below a threshold empirically set to \( 10^{-1} \), i.e.: \( \frac{1}{r \sqrt{d}} \sum_j \|D_{j,q}^{old} - D_{j,q}^{new}\|_F < 10^{-1}, \forall j \).

C. Denoising performance

Table IV shows the denoising performance of the proposed structured dictionaries compared to K-SVD as an unstructured counterpart. In order not to overburden the analysis, we report only the HO-SuKro results with rank \( r = 3 \), which we judged to represent a good performance-complexity compromise.

Increasing the patch size improves significantly the performance of HO-SuKro, while that of K-SVD doesn’t benefit as much and may even deteriorate, indicating the onset of overfitting. This can be attributed to the growing number of parameters to estimate (as the dictionary size grows) making the number of available training data insufficient. HO-SuKro, thanks to its structured nature, avoids this issue.

More than simply increasing the patch size, what was empirically observed to drastically improve performance was increasing the patch dimension in the spectral mode, as shown in the last line of Table IV. Naturally, bigger patches also imply higher learning and denoising times. It is, thus, a compromise to be considered according to the available resources.

TABLE IV: Output SNR for various patch sizes – San Diego

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Patch size</th>
<th>Input SNR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m ( d )</td>
<td>10 15 20 25</td>
</tr>
<tr>
<td>HO-SuKro ( (r = 3) )</td>
<td>[6, 6, 6] [12,12,12]</td>
<td>24.10 27.07 30.09 32.32</td>
</tr>
<tr>
<td>K-SVD</td>
<td>[8, 8, 8] [16,16,16]</td>
<td>21.77 25.31 29.09 32.68</td>
</tr>
<tr>
<td>K-SVD ( (r = 2) )</td>
<td>[6, 6, 20] [12,12,20]</td>
<td>22.49 26.06 29.91 33.21</td>
</tr>
<tr>
<td>K-SVD ( (r = 1) )</td>
<td>[8, 8, 8] [16,16,16]</td>
<td>20.35 25.34 29.42 33.24</td>
</tr>
<tr>
<td>K-SVD ( (r = 1) )</td>
<td>[10,10,10] [20,20,20]</td>
<td>23.08 26.35 29.73 33.27</td>
</tr>
</tbody>
</table>
Table V: Output SNR [dB] comparison with literature

<table>
<thead>
<tr>
<th>Image</th>
<th>Algorithm</th>
<th>Input SNR [dB]</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Diego</td>
<td>Wavelet 2D</td>
<td></td>
<td>14.75</td>
<td>18.00</td>
<td>21.70</td>
<td>25.92</td>
</tr>
<tr>
<td></td>
<td>Wavelet 3D</td>
<td></td>
<td>23.11</td>
<td>26.04</td>
<td>28.91</td>
<td>31.68</td>
</tr>
<tr>
<td></td>
<td>FORPDN</td>
<td></td>
<td>22.23</td>
<td>24.17</td>
<td>26.42</td>
<td>29.00</td>
</tr>
<tr>
<td></td>
<td>HyRes</td>
<td></td>
<td>25.38</td>
<td>28.60</td>
<td>31.75</td>
<td>34.70</td>
</tr>
<tr>
<td></td>
<td>HO-SuKro</td>
<td></td>
<td>24.10</td>
<td>27.07</td>
<td>30.09</td>
<td>33.22</td>
</tr>
<tr>
<td>Houston</td>
<td>Wavelet 2D</td>
<td></td>
<td>14.22</td>
<td>17.67</td>
<td>21.43</td>
<td>25.80</td>
</tr>
<tr>
<td></td>
<td>Wavelet 3D</td>
<td></td>
<td>22.35</td>
<td>25.54</td>
<td>28.65</td>
<td>31.86</td>
</tr>
<tr>
<td></td>
<td>FORPDN</td>
<td></td>
<td>22.80</td>
<td>25.46</td>
<td>28.09</td>
<td>30.74</td>
</tr>
<tr>
<td></td>
<td>HyRes</td>
<td></td>
<td>26.00</td>
<td>29.35</td>
<td>33.24</td>
<td>37.05</td>
</tr>
<tr>
<td></td>
<td>HO-SuKro</td>
<td></td>
<td>23.29</td>
<td>26.63</td>
<td>29.93</td>
<td>33.20</td>
</tr>
</tbody>
</table>

Fig. 1: Example of denoised images (100\textsuperscript{th} spectral band)

Table V compares HO-SuKro’s performance to that of other techniques from the literature. By directly applying the proposed structured dictionary learning algorithm to Hyperspectral data, without any further domain-specific adaptation, we already obtain results comparable to the literature. Both HO-SuKro and K-SVD outperform the wavelet-based approaches (2D and 3D [18]), corroborating the interest of learning the sparsifying dictionary from data. HO-SuKro also consistently outperforms FORPDN [19] which exploits the correlation within spectral bands. Naturally, we don’t manage to reach state-of-the-art performance (HyRes [20]) for this task, which makes use of a meaningful low-rank prior on the HSI. The performance gap is around 1.5dB and 3dB for San Diego and Houston images respectively. An example of denoised image is provided in Figure 1. A closer look reveals that our approach may over-smooth some details.

VI. CONCLUSION

An alternate minimization algorithm was proposed in this paper for learning tensor-structured dictionaries designed to tackle the inherent multidimensional structure of the data. We analyzed the computational complexity of the algorithm and made some considerations on the expected empirical speedups.

Hyperspectral image denoising experiments were performed and the tensor structure constraint on the dictionary proved to be beneficial in terms of performance compared to a completely unstructured dictionary. Some adaptations are envisioned to improve our performance in this specific task and try to reach the state-of-the-art, notably the incorporation of the low-rank constraint on the data.

REFERENCES