remote sensing

MDPI

Article

An Efficient Compressive Hyperspectral Imaging Algorithm Based on Sequential Computations of Alternating Least Squares

Geunseop Lee

Division of Global Business and Technology, Hankuk University of Foreign Studies, Yongin 17035, Korea; geunseop.lee@hufs.ac.kr

Received: 23 October 2019; Accepted: 3 December 2019; Published: 6 December 2019

Abstract: Hyperspectral imaging is widely used to many applications as it includes both spatial and spectral distributions of a target scene. However, a compression, or a low multilinear rank approximation of hyperspectral imaging data, is required owing to the difficult manipulation of the massive amount of data. In this paper, we propose an efficient algorithm for higher order singular value decomposition that enables the decomposition of a tensor into a compressed tensor multiplied by orthogonal factor matrices. Specifically, we sequentially compute low rank factor matrices from the Tucker-1 model optimization problems via an alternating least squares approach. Experiments with real world hyperspectral imaging revealed that the proposed algorithm could compute the compressed tensor with a higher computational speed, but with no significant difference in accuracy of compression compared to the other tensor decomposition-based compression algorithms.

Keywords: hyperspectral imaging; HOSVD; alternating least squares

1. Introduction

Hyperspectral imaging (HSI) allows one to provide information on the spatial and spectral distributions of a target scene simultaneously by acquiring up to hundreds of narrow and adjacent spectral band images ranging from ultraviolet to far-infrared electromagnetic spectrum [1,2]. To do this, an imaging sensor such as a charged coupled device collects the different wavelengths dispersed from the incoming light. Then, the signals captured by the imaging sensor are digitized and arranged into pixels of a two-dimensional image $\mathbf{T} = \mathbf{R}^{I \times \lambda}$, where I denotes the size of X-directional spatial information, and λ is the number of quantized spectra of the signals. The procedure of capturing the pixels as an X-directional single line continues until the spatial range reaches J, which is the Y-directional size of the entire target scene. Finally, HSI constructs a three-dimensional data $\mathcal{T} \in \mathbf{R}^{I \times J \times \lambda}$. Once the HSI data are obtained, they can be used in many applications, such as detecting and identifying objects at a distance in environmental monitoring [3] or medical image processing [4], finding anomaly in automatic visual inspection [5], or detecting and identifying targets of interest [6,7]. However, as the area of the target scene $I \times J$ or the number of quantized spectra λ increase, the manipulation of $\mathcal{T}$ demands prohibitively large computational resources and storage space. To overcome this, efficient compression techniques must be employed as preprocessing for applications to filter out some redundancy along their adjacent spectral bands or spatial information, thereby reducing the size of $\mathcal{T}$.

Owing to the shape of HSI data, the typical mathematical basis for compression techniques is based on tensor decompositions, because it facilitates the simultaneous preservation and analysis of the spatial and spectral structures of data [8]. Fang et al. used canonical polyadic decomposition (CPD) for dimension reduction of HSI data, where CPD decomposes a tensor into several rank-one tensors [9,10]. De Lathauwer et al. suggested a Tucker decomposition-based low rank approximation

algorithm of a tensor, where Tucker decomposition decomposes a tensor into a smaller sized core tensor multiplied by a factor matrix along each mode [11]. In this study, we considered Tucker decomposition for compressing HSI data. Specifically, we focused on developing an efficient algorithm to compute a higher order singular value decomposition (HOSVD), which is a special case of Tucker decomposition with orthogonal constraints on the factor matrices. Subsequently, we applied it to compression problems from real world HSI data.

The remainder of this paper is organized as follows. Section 2 defines the notations and preliminaries frequently used in this paper. Section 3 briefly explains the well-known algorithms for computing HOSVD for a compression. Section 4 introduces the algorithm we propose. Section 5 provides the experimental results, and Section 6 concludes the paper.

2. Notations and Preliminaries

Here, we define the symbols and terminology for the simplicity of notation and presentation. We use calligraphic letters to denote tensors, e.g., $\mathcal{A}$; boldface capital letters for matrices, e.g., $\mathbf{A}$; boldface lowercase letters for vectors, e.g., $\mathbf{a}$; and lowercase letters for scalars, e.g., a. We define an operation of tensor-matrix multiplication between an arbitrary N-th order tensor $\mathcal{A} \in \mathbf{R}^{I_1 \times I_2 \cdots \times I_N}$ and a matrix $\mathbf{B} \in \mathbf{R}^{K \times I_n}$, $1 \leq n \leq N$ along mode-n, such that

$$\mathcal{C} = \mathcal{A} \times_n \mathbf{B},$$

where $\mathcal{C} \in \mathbf{R}^{I_1 \times \cdots \times I_{n-1} \times K \times I_{n+1} \times \cdots \times I_N}$ and the $(i_1 \ldots i_{n-1}, k, i_{n+1} \ldots i_N)$-th element of $\mathcal{C}$ is computed by

$$\mathcal{C}_{i_1 \ldots i_{n-1}, k, i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} a_{i_1, i_2 \ldots i_N} b_{k, i_n},$$

for arbitrary $1 \leq i_n \leq I_n$ and $1 \leq k \leq K$. Note that the values $a_{i_1, i_2 \ldots i_N}$ and b_{k, i_n} are the elements of $\mathcal{A}$ and $\mathbf{B}$, respectively [12].

A tensor $\mathcal{A}$ can be matricized with mode-n, after rearranging each element of $\mathcal{A}$ appropriately. For example, a third order tensor $\mathcal{A} \in \mathbf{R}^{I_1 \times I_2 \times I_3}$ is matricized along each mode, as shown in Figure 1.

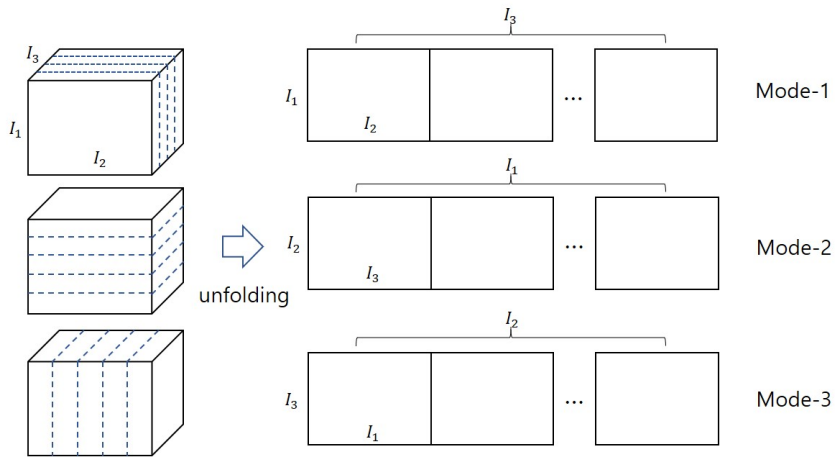


Figure 1. Example of third order tensor matricization along each mode.

The Frobenius norm of a tensor $\mathcal{A} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the square root of the sum of the squares of all elements in $\mathcal{A}$, such that

$$\|\mathcal{A}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} a_{i_1 i_2 \ldots i_N}^2}.$$

3. Related Works

We briefly revisit here, well-known algorithms to compute HOSVD of a tensor. HOSVD, which is a special case of Tucker decomposition that has an orthogonal constraint, decomposes an N-th order tensor $\mathcal{T} \in \mathbf{R}^{I_1 \times I_2 \ldots \times I_N}$ into factor matrices $\mathbf{U}_n \in \mathbf{R}^{I_n \times R_n}, 1 \leq n \leq N$ and a core tensor $\mathcal{G} \in \mathbf{R}^{R_1 \times R_2 \ldots \times R_N}$ such that

$$\mathcal{T} = \mathcal{G} \times_{n=1 \text{ to } N} \mathbf{U}_n, \tag{1}$$

while satisfying a constraint $\mathbf{U}_n^T \mathbf{U}_n = \mathbf{I}$, where a matrix $\mathbf{I} \in \mathbf{R}^{R_n \times R_n}$ represents an identity matrix [11]. Here, a factor matrix $\mathbf{U}_n$ is considered as the principal components in each mode, and elements of a core tensor $\mathcal{G}$ reveal the level of interactions between the different components. Note that $R_n \leq I_n$, and we denote $(R_1, R_2, \ldots, R_N)$ as multilinear ranks of $\mathcal{T}$. The representation of $\mathcal{T}$ with the form of (1) is not unique. Thus, there are many algorithms to compute (1). One of the simplest methods to obtain $\mathbf{U}_n$ and $\mathcal{G}$ is computing leading singular vectors of each unfolding matrix of $\mathcal{T}$ such that

$$\mathbf{T}_n = \mathbf{U}_n \mathbf{\Sigma}_n \mathbf{V}_n^T, \tag{2}$$

where $\mathbf{T}_n$ indicates the mode-n unfolding matrix of $\mathcal{T}$. Then a core tensor $\mathcal{G}$ is obtained from $\mathcal{G} = \mathcal{T} \times_{n=1 \text{ to } N} \mathbf{U}_n^T$, and $\mathcal{G}$ is regarded as a compressed tensor of $\mathcal{T}$. Despite its simple procedure of computation, a single step of applying singular value decomposition (SVD) is insufficient to improve approximations of $\mathcal{T}$ in many contexts. De Lathauwer et al. proposed a more accurate algorithm of computing HOSVD via iterative computations of SVD, called higher order orthogonal iterations (HOOI) [13]. HOOI is designed to solve the optimization problem of finding $\mathbf{U}_n$ and $\mathcal{G}$, such that

$$\min_{\mathcal{G}, \mathbf{U}_n, n=1 \text{ to } N} \|\mathcal{T} - \mathcal{G} \times_{n=1 \text{ to } N} \mathbf{U}_n\|_F, \quad \text{subject to } \mathbf{U}_n^T \mathbf{U}_n = \mathbf{I}. \tag{3}$$

Since $\|\mathcal{T} - \mathcal{G} \times_{n=1 \text{ to } N} \mathbf{U}_n\|_F = \|\mathcal{T}\|_F - \|\mathcal{G}\|_F$, the minimization problem (3) is identical to finding $\max \|\mathcal{G}\|_F$; thus, by definition,

$$\max \|\mathcal{T} \times_{n, n=1 \text{ to } N} \mathbf{U}_n^T\|_F. \tag{4}$$

Therefore, HOOI obtains each factor matrix $\mathbf{U}_n$ independently from the R_n, leading singular vectors of the unfolding matrix matricized from $\mathcal{T} \times_{n=1 \text{ to } N, n \neq k} \mathbf{U}_n$ along mode-k while fixing the other factor matrices. The iterations continue until the output converges. The procedure of HOOI is summarized in Algorithm 1 for $N = 3$. In practice, HOOI produces more accurate outputs than those from the algorithm based on (2); it is considered one of the most accurate algorithms for obtaining HOSVD from a tensor. Therefore, many hyperspectral compression techniques have been developed based on HOOI. For example, Zhang et. al applied HOOI to the compression of HSI [14]. An et al. suggested the method based on [11] with an adaptive, multilinear rank estimation, and applied it to HSI compression [15]. However, iterative computations of SVD require huge computational resources. Additionally, the number of iterations for convergence is difficult to predict.

To overcome these limitations, Elden et al. proposed a Newton–Grassman based algorithm that guaranteed quadratic convergence and fewer iteration steps than HOOI [16]. However a single iteration of the algorithm is much more expensive owing to the computation of the Hessian. Sorber et al. introduced a quasi-Newton optimization algorithm that iteratively improves the initial guess by using a quasi-Newton method from the nonlinear cost function [17]. Hassanzadeh and karami proposed a block coordinate descent search based algorithm [18], which updates the factor matrices initialized by using compressed sensing. Instead of employing SVD for unfolding matrices, Phan et al. proposed a fast algorithm based on a Crank–Nicholson-like technique, which has a lower computational cost in a single step compared with HOOI [19]. Lee proposed a HOSVD algorithm based on an alternating least squares method that recycles the intermediate results of computations for one factor matrix to the other computations [20]. In contrast to the algorithm proposed by [20], which computes the Tucker-1 model optimization individually along each mode, the proposed algorithm considers sub-problems of

computing a factor matrix simultaneously in a single iteration. This approach enables more accurate computation of the intermediate results in each iteration.

Algorithm 1 HOOI

Input: $\mathcal{T}, \mathcal{G}^0, \mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3, \epsilon$
Output: $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3, \mathcal{G}^l$

for l = 1,2,3,... **do**
  for n=1,2,3 **do**
    $k = [n, 1 : n-1, n+1 : 3]$
    $\mathcal{S} = \mathcal{T} \times_k \mathbf{U}_k^T$
    $\mathbf{U}_n$ = R_n leading singular vectors of $\mathbf{S}_n$
  end for
  $\mathcal{G}^l = \mathcal{T} \times_{n=1,2,3} \mathbf{U}_n^T$
  if $\|\mathcal{G}^l - \mathcal{G}^{l-1}\|_F / \|\mathcal{G}^{l-1}\|_F \leq \epsilon$ **then**
    break
  end if
end for

4. Sequential Computations of Alternating Least Squares for Efficient HOSVD

In the proposed algorithm, we sequentially compute low multilinear rank factor matrices from the Tucker-1 model optimization problems via an alternating least squares approach. For simplicity, and to consider the shape of a HSI data, we only used a third order tensor in this study. However, the extension of the algorithm for application to any dimensional tensors without loss of generality is straightforward.

Assume that we have a third order tensor $\mathcal{T} \in \mathbf{R}^{I_1 \times I_2 \times I_3}$. Our goal is to decompose $\mathcal{T}$ to

$$\mathcal{T} \approx \mathcal{G} \times_{n=1 \text{ to } 3} \mathbf{U}_n, \tag{5}$$

where $\mathbf{U}_n \in \mathbf{R}^{I_n \times R_n}$ represents an orthogonal factor matrix along mode-n, R_n is the appropriate truncation level or n-multilinear rank, and $\mathcal{G} \in \mathbf{R}^{R_1 \times R_2 \times R_3}$ denotes the core tensor. Then, we rewrite the formulation of the optimization problem of finding $\mathbf{U}_n, n = 1, 2, 3$ and $\mathcal{G}$ in (3) as

$$\min_{\mathcal{G}, U_n, n=1,2,3} \|\mathcal{T} - \mathcal{G} \times_{n=1,2,3} \mathbf{U}_n\|_F^2 \quad \text{subject to } \mathbf{U}_n^T \mathbf{U}_n = \mathbf{I}. \tag{6}$$

Before starting the explanation, we note that the order of computation for each factor matrix does not need to be fixed. However, for convenience we will present the procedure for solving our optimization problems from the order of (1,2,3) mode. Let $\mathcal{S}_1 = \mathcal{G} \times_{n=2,3} \mathbf{U}_n$. Then, the problem of finding $\mathbf{U}_1$ in (6) is equivalent to the problem of finding the solution from the Tucker-1 model optimization of $\mathcal{T}$, such that

$$\min_{\mathbf{U}_1, \mathcal{S}_1} \|\mathcal{T} - \mathcal{S}_1 \times_1 \mathbf{U}_1\|_F^2 \text{ subject to } \mathbf{U}_1^T \mathbf{U}_1 = \mathbf{I}, \tag{7}$$

where $\mathbf{I} \in \mathbf{R}^{R_1 \times R_1}$ denotes an identity matrix. Here, we can see that the tensor $\mathcal{S}_1$ is a Tucker-2 model, but it can be used to formulate another Tucker-1 optimization problem of finding $\mathbf{U}_2$, such that

$$\min_{\mathbf{U}_2, \mathcal{S}_2} \|\mathcal{S}_1 - \mathcal{S}_2 \times_2 \mathbf{U}_2\|_F^2 \text{ subject to } \mathbf{U}_2^T \mathbf{U}_2 = \mathbf{I}, \tag{8}$$

where $\mathcal{S}_2 = \mathcal{G} \times_3 \mathbf{U}_3$. Similarly, to find $\mathbf{U}_3$, we use the optimization problem of Tucker-1 model, which is given by

$$\min_{\mathbf{U}_3, \mathcal{G}} \|\mathcal{S}_3 - \mathcal{G} \times_3 \mathbf{U}_3\|_F^2 \text{ subject to } \mathbf{U}_3^T \mathbf{U}_3 = \mathbf{I}. \tag{9}$$

We illustrate the sequential procedure for the Tucker-1 model optimization problems in Figure 2. Let us consider the optimization problems (7)–(9) simultaneously. Then, the optimal factor matrices of the tensor $\mathcal{T}$ are computed by solving the minimization problem of the cost function $C(\mathbf{U}_k, k = 1,2,3)$, which is defined as

$$\begin{aligned} C(\mathbf{U}_k, k = 1,2,3) = &\|\mathcal{T} - \mathcal{S}_1 \times_1 \mathbf{U}_1\|_F^2 + \|\mathcal{S}_1 - \mathcal{S}_2 \times_2 \mathbf{U}_2\|_F^2 \\ &+ \|\mathcal{S}_3 - \mathcal{G} \times_3 \mathbf{U}_3\|_F^2 + \sum_{n=1,2,3} \lambda_n \mathrm{tr}(\mathbf{U}_n^T \mathbf{U}_n - \mathbf{I}), \end{aligned} \tag{10}$$

where the parameters λ_n represent the Lagrangian multipliers, and the function $\mathrm{tr}(\mathbf{A})$ computes the trace of an arbitrary matrix $\mathbf{A}$. Because the cost function $C(\mathbf{U}_k, k = 1,2,3)$ in (10) has too many unknown variables, an alternating least squares method that optimizes one variable while leaving the others fixed, is a proper approach. First, by taking a derivative of $C(\mathbf{U}_k, k = 1,2,3)$ with respect to $\mathbf{U}_1$ while regarding the other variables as constant values, and matricizing those along mode-1, we obtain

$$\frac{\partial C(\mathbf{U}_k, k = 1,2,3)}{\partial \mathbf{U}_1} = -\frac{1}{2}(\mathbf{T}_1 - \mathbf{U}_1 \mathbf{S}_{1_1})\mathbf{S}_{1_1}^T + \frac{1}{2}\lambda_1 \mathbf{U}_1 = 0, \tag{11}$$

where $\mathbf{S}_{1_1}$ is the unfolding matrix of $\mathcal{S}_1$ along mode-1. Thus, from (11), we can compute $\mathbf{U}_1$ as follows:

$$\mathbf{U}_1 = \mathbf{T}_1 \mathbf{S}_{1_1}^T (\lambda_1 \mathbf{I} + \mathbf{S}_{1_1} \mathbf{S}_{1_1}^T)^{-1}. \tag{12}$$

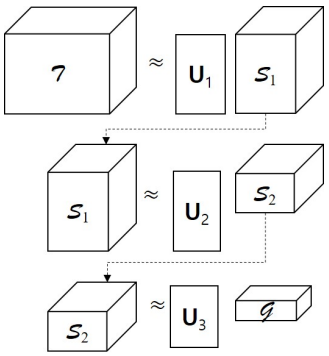


Figure 2. Sequential computations of factor matrices from Tucker-1 model optimization problems.

After $\mathbf{U}_1$ is obtained, the next step is to compute $\mathcal{S}_1$ while fixing $\mathbf{U}_1$ and the others to recycle the intermediate results for computing the others. After updating $\mathbf{U}_1$ in (12) and by taking a derivative to (10) with respect to $\mathcal{S}_1$, we can compute $\mathcal{S}_1$ such that

$$\mathcal{S}_1 = (\mathcal{T} \times_1 \mathbf{U}_1 + \mathcal{S}_2 \times_2 \mathbf{U}_2) \times_1 (\mathbf{I} + \mathbf{U}_1^T \mathbf{U}_1)^{-1}.$$

Because the orthogonal constraint must be satisfied, we reorthogonalize $\mathbf{U}_1$ by simply applying QR-decomposition.

Next, we find $\mathbf{U}_2$. After updating $\mathcal{S}_1$ and $\mathbf{U}_1$, we take a derivative of $C(\mathbf{U}_k, k = 1,2,3)$ with respect to $\mathbf{U}_2$ and rearrange the terms similar to the procedure of computing $\mathbf{U}_1$. Then, we obtain

$$\mathbf{U}_2 = \mathbf{S}_{1_2} \mathbf{S}_{2_2}^T (\lambda_2 \mathbf{I} + \mathbf{S}_{2_2} \mathbf{S}_{2_2}^T)^{-1}, \tag{13}$$

where $\mathbf{S}_{1_2}$ and $\mathbf{S}_{2_2}$ are the unfolding matrices of $\mathcal{S}_1$ and $\mathcal{S}_2$ along mode-2, respectively. Then, we can compute $\mathcal{S}_2$ with fixed $\mathbf{U}_2$ such that

$$\mathcal{S}_2 = (\mathcal{S}_1 \times_2 \mathbf{U}_2 + \mathcal{G} \times_3 \mathbf{U}_3) \times_2 (\mathbf{I} + \mathbf{U}_2^T \mathbf{U}_2)^{-1}.$$

Finally, $\mathbf{U}_3$ is obtained by solving

$$\mathbf{U}_3 = \mathbf{S}_{2_3}\mathbf{G}_3^T(\lambda_3\mathbf{I} + \mathbf{G}_3\mathbf{G}_3^T)^{-1}, \tag{14}$$

where $\mathbf{S}_{2_3}$ and $\mathbf{G}_3$ are the unfolding matrices from the tensor $\mathcal{S}_2$ and $\mathcal{G}$ along mode-3, respectively.

Algorithm 2 summarizes the procedure explained in this section. Note that the function $\mathbf{A}_i = \mathbf{unfolding}(\mathcal{A}, i)$ in steps 6, 10, and 14, returns the unfolding matrix $\mathbf{A}_i$ from an arbitrary tensor $\mathcal{A}$ along mode-i, and the function $\mathbf{C} = \mathbf{Reorth}(\mathbf{B})$ in steps 9, 13, and 16 returns the reorthogonalized matrix $\mathbf{C}$ from $\mathbf{B}$ by applying QR decomposition. If we assume that the size of an input tensor is (I, I, I), and its initial multilinear rank is (R, R, R), then the most expensive step in Algorithm 2 occurs at the computation of $\mathbf{T}_i\mathbf{S}_{ii}^T$ in step 7, and its computational complexity is approximately $O(I^3R)$ operations, which is similar to those of the other HOSVD algorithms. Additionally, unlike HOOI, we eliminate independence from the computation of each factor matrix by reusing intermediate tensors and factor matrices to find a specific factor matrix. Thus, we expected the proposed algorithm to achieve better convergence to the solution.

Algorithm 2 HOSVD_ALS

Input: $\mathcal{T}, \mathcal{G}^0, \mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3, \lambda_1, \lambda_2, \lambda_3, \epsilon$
Output: $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3, \mathcal{G}^l$

1: **for** l = 1,2,3,... **do**
2: **for** n = 1 to 3 **do**
3: Rearrange the order $[i, j, k]$ such that $[n, 1 : n-1, n+1 : 3]$
4: $\mathcal{S}_j = \mathcal{G}^{l-1} \times_k \mathbf{U}_k$
5: $\mathcal{S}_i = \mathcal{S}_j \times_j \mathbf{U}_j$
6: $\mathbf{T}_i = \mathbf{unfolding}(\mathcal{T}, i)$, and $\mathbf{S}_{ii} = \mathbf{unfolding}(\mathcal{S}_i, i)$
7: $\mathbf{U}_i = \mathbf{T}_i\mathbf{S}_{ii}^T(\lambda_i\mathbf{I} + \mathbf{S}_{ii}\mathbf{S}_{ii}^T)^{-1}$
8: $\mathcal{S}_i = (\mathcal{T} \times_i \mathbf{U}_i + \mathcal{S}_j \times_j \mathbf{U}_j) \times_i (\mathbf{I} + \mathbf{U}_i^T\mathbf{U}_i)^{-1}$
9: $\mathbf{U}_i = \mathbf{Reorth}(\mathbf{U}_i)$
10: $\mathbf{S}_{ij} = \mathbf{unfolding}(\mathcal{S}_i, j)$, and $\mathbf{S}_{jj} = \mathbf{unfolding}(\mathcal{S}_j, j)$
11: $\mathbf{U}_j = \mathbf{S}_{ij}\mathbf{S}_{jj}^T(\lambda_j\mathbf{I} + \mathbf{S}_{jj}\mathbf{S}_{jj}^T)^{-1}$
12: $\mathcal{S}_j = (\mathcal{S}_i \times_j \mathbf{U}_j + \mathcal{G}^{k-1} \times_k \mathbf{U}_k) \times_j (\mathbf{I} + \mathbf{U}_j^T\mathbf{U}_j j)^{-1}$
13: $\mathbf{U}_j = \mathbf{Reorth}(\mathbf{U}_j)$
14: $\mathbf{S}_{jk} = \mathbf{unfolding}(\mathcal{S}_j, k)$, and $\mathbf{G}_k = \mathbf{unfolding}(\mathcal{G}^{l-1}, k)$
15: $\mathbf{U}_k = \mathbf{S}_{jk}\mathbf{G}_k^T(\lambda_k\mathbf{I} + \mathbf{G}_k\mathbf{G}_k^T)^{-1}$
16: $\mathbf{U}_k = \mathbf{Reorth}(\mathbf{U}_k)$
17: $\mathcal{G}^l = \mathcal{T} \times_{m=1,2,3} \mathbf{U}_m^T$
18: **if** $\|\mathcal{G}^l - \mathcal{G}^{l-1}\|_F / \|\mathcal{G}^{l-1}\|_F \leq \epsilon$ **then**
19: break
20: **end if**
21: **end for**
22: **end for**

5. Experiments

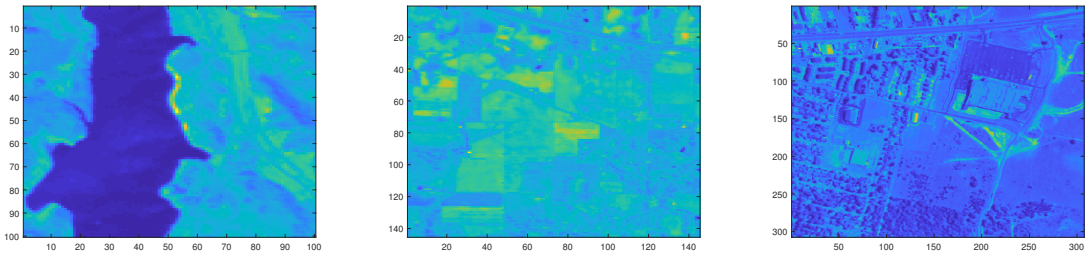
We begin this section by introducing the experimental settings. Then, we compare the performance of Algorithm 2 to those of the other well-known HOSVD algorithms by showing an application to real-world HSI data.

5.1. Experimental Settings

Our experiments were performed on Intel i9 processor with 32 GB of memory. We developed the software for the experiments using MATLAB version 9.6.0.1135713. For the real-world HSI dataset, we used three datasets which are widely used for testing classification or compression performance of HSI data. Details on these datasets are as follows.

- Jasper Ridge: Jasper Ridge dataset was captured by an airborne visible/infrared imaging spectrometer (AVIRIS) sensor by the Jet Propulsion Laboratory. The spatial size of the dataset is 100×100 with 224 channels, in which its quantized spectra range is from 380 nm to 2500 nm. There are four endmembers in this dataset, which include "road," "soil," "water," and "tree." Detailed information regarding Jasper Ridge dataset is provided in [21].
- Indian Pines: Indian pine dataset was captured by AVIRIS sensor over the Indian pine test site in North-western Indiana. The scenery is comprised of agriculture, forest or natural perennial vegetation. The spatial size of the dataset used in the experiments was 145×145 with 224 channels ranging from 400 nm to 2500 nm. Detailed information of Indian Pines dataset is provided in [22].
- Urban: Urban dataset was recorded by a hyperspectral digital image collection experiment (HYDICE) sensor; its location is an urban area at Copperas Cove in Texas. The spatial size of the dataset is 307×307 with 221 channels, in which its quantized spectral range is from 400 nm to 2500 nm. There are four endmembers to be classified; namely, "asphalt," "grass," "tree," and "roof." More information about Urban dataset is provided in [21].

Figure 3 depicts the average intensities of pixels throughout the spectrum. The datasets are represented as tensors; for example, the Jasper Ridge dataset is denoted by $\mathcal{T} \in \mathbf{R}^{100 \times 100 \times 224}$.



(**a**) Jasper Ridge (**b**) Indian Pines (**c**) Urban

Figure 3. Real-world HSI dataset for performance comparison of the proposed algorithm. Images display the average intensities of pixels throughout the spectrum.

To compare the performances of the proposed algorithm with previous algorithms, we evaluated the relative errors and the execution times with the algorithm from HOOI, a Crank–Nicholson-like algorithm for HOSVD (CrNc henceforce) [19]; a quasi-Newton-based nonlinear least squares algorithm (henceforce HOSVD_NLS) [17]; and a method based on block coordinate descent search [18] which is a slight modification of the algorithm described in [23] (henceforth BCD-CD). Here, the relative errors, denoted as relerr, are defined such that

$$\text{relerr} = \frac{\|\mathcal{T}_{gt} - \mathcal{T}_{output}\|_F}{\|\mathcal{T}_{gt}\|_F},$$

where $\mathcal{T}_{gt}$ and $\mathcal{T}_{output}$ are the tensors before and after applying compression algorithms, respectively. The programs implementing HOOI and HOSVD_NLS algorithms, are from [24]. Note that the stopping criterion of all the algorithms in the comparison is equal, and it is defined as

$$\frac{\|\mathcal{G}_{n+1} - \mathcal{G}_n\|_F}{\|\mathcal{G}_n\|_F} \leq \epsilon, \tag{15}$$

where ϵ indicates a user-defined threshold. The maximum iteration number is 100. To measure the execution time of each algorithm, we repeat the experiments 10 times and take the average from the results.

The initial factor matrices with low multilinear ranks of all algorithms are computed from SVD-based HOSVD algorithm described in (2) (henceforth, HOSVD) when the truncation level R_n in mode-n, $1 \leq n \leq 3$, satisfies the condition

$$\sqrt{\|\mathbf{T}_n\|_F^2 - \sum_{i=1}^{R_n} \sigma_i} \leq \gamma,$$

where σ_i represents the i-th largest singular value of the mode-n unfolding matrix $\mathbf{T}_n$ from $\mathcal{T}$, and γ is the user-defined threshold to adjust the compression rate of the spectral and spatial dimensions of $\mathcal{T}$. Specifically, we set the value γ to 0.05, 0.1, 0.2, and 0.3. Table 1 lists the low multilinear ranks in each case after applying HOSVD. Note that the sizes of the core tensors from all algorithms are identical.

Table 1. Multilinear ranks of the initial factor matrices computed from HOSVD when $\gamma = 0.05, 0.1, 0.2,$ and 0.3, respectively.

Dataset	γ **= 0.05**	**0.1**	**0.2**	**0.3**
Jasper Ridge	(71,75,5)	(48,54,3)	(28,32,2)	(17,20,2)
Indian Pines	(97,117,8)	(44,51,2)	(11,10,2)	(3,3,2)
Urban	(292,227,13)	(265,165,5)	(186,93,3)	(108,52,3)

5.2. Experimental Results

The first experiment measured the performances of the algorithms when the user-defined threshold ϵ in (15) was 1.0×10^{-6}. We measured relerrs and the execution times while changing the value of γ to 0.05, 0.1, 0.2, and 0.3. When $\gamma = 0.3$ we set the Lagrangian multipliers $\lambda_1 = \lambda_2 = \lambda_3 = 1.0 \times 10^{-8}$ in (10), and set 0 to the other cases of γ. The experimental results are given in Tables 2–4. For an unknown reason, HOOI failed to converged to the solutions occasionally; for example, when $\gamma = 0.05$, as shown Table 2. From these results, we can see that the overall execution speed of the proposed algorithm is the fastest with fewer iteration numbers, while its relative errors are very close to those of HOOI, which is the most accurate algorithm in this experiment. CrNc converged to the solutions with the fewest iteration numbers and produced the outputs with the fastest times occasionally; however, its relative errors are inaccurate compared to the other algorithms. HOSVD_NLS produced the closest relative errors to HOOI; however, its execution time was the slowest among all algorithms. The overall performance of BCD-CD appears to be unsatisfactory in all cases, especially with regard to convergence speed and relative errors. Excluding one case presented in Table 3, BCD-CD failed to converge to the solutions within the predefined maximum iteration number.

The second experiment measured the performances of the algorithms when ϵ in (15) was 1.0×10^{-8}. We provide the results of this experiment in Tables 5–7. Similar to the results from the first experiment, the proposed algorithm computes the compressed tensor more efficiently compared to the other algorithms.

The third experiment measured the performance of the algorithms under noisy conditions. We added white Gaussian noises with different signal-to-noise ratios to HSI imaging data such that

$$\mathcal{T}_{noise} = \mathcal{T}_{gt} + \sqrt{10^{-\sigma/20}} \frac{\|\mathcal{T}_{gt}\|_F}{\|\mathcal{N}\|_F} \cdot \mathcal{N},$$

where σ represents the signal-to-noise ratio, and $\mathcal{N}$ is the randomly generated tensor. We set σ = +60 dB, +30 dB, and +20 dB, respectively. Table 8 summarizes the outputs of the experiment, and similar to the first experiment, it shows that the most accurate algorithm in many cases is HOOI. However,

the proposed algorithm produces outputs with relative errors very similar to those of HOOI, while maintaining robust convergence to the solutions. In some cases, when using Indian Pines and Urban as the input data, the proposed algorithm produces even smaller relative errors than HOOI. Note that the numbers in the parenthesis represent the average iteration numbers required for convergence under noisy conditions. Additionally, Figure 4 shows the first channel to be compressed when Jasper Ridge dataset is used. There are no significant differences except the case of BCD-CD.

Table 2. Experimental results of algorithms when the Jasper Ridge dataset was used and $\epsilon = 1.0 \times 10^{-6}$.

γ		HOOI	CrNc	HOSVD_NLS	BCD-CD	Algorithm 2
0.05	iteration	100	2	29	100	2
	relerr	0.0291179	0.0291398	0.0291181	0.155959	0.0291264
	time (s)	8.1602	0.2674	8.3305	4.6889	0.1905
0.1	iteration	5	2	26	100	2
	relerr	0.0591755	0.0592419	0.0591757	0.152589	0.0591861
	time (s)	0.3509	0.2197	4.4906	3.4722	0.1401
0.2	iteration	8	4	68	100	6
	relerr	0.118573	0.118831	0.118572	0.161462	0.118703
	time (s)	0.4058	0.2921	6.3683	2.4399	0.2031
0.3	iteration	6	6	23	100	6
	relerr	0.141258	0.141271	0.141258	0.179646	0.141280
	time (s)	0.2339	0.2978	1.9192	3.4341	0.1442

Table 3. Experimental results of algorithms when the Indian Pines dataset was used and $\epsilon = 1.0 \times 10^{-6}$.

γ		HOOI	CrNc	HOSVD_NLS	BCD-CD	Algorithm 2
0.05	iteration	100	2	57	100	4
	relerr	0.0303899	0.0304928	0.0303906	0.071481	0.0303906
	time (s)	20.6298	0.5488	45.4292	12.0304	0.7744
0.1	iteration	6	2	36	100	2
	relerr	0.0527313	0.0528268	0.0527315	0.073079	0.0527553
	time (s)	0.6524	0.2927	7.5470	4.6133	0.2011
0.2	iteration	7	4	48	100	6
	relerr	0.0768619	0.0770075	0.0768605	0.124355	0.0769648
	time (s)	0.2839	0.2696	6.1851	2.0955	0.1630
0.3	iteration	2	2	9	50	2
	relerr	0.105915	0.105915	0.105915	0.127037	0.105915
	time (s)	0.0814	0.1522	1.1482	0.6996	0.0662

Table 4. Experimental results of algorithms when the Urban dataset was used and $\epsilon = 1.0 \times 10^{-6}$.

γ		HOOI	CrNc	HOSVD_NLS	BCD-CD	Algorithm 2
0.05	iteration	100	2	47	100	2
	relerr	0.0310416	0.0311142	0.0310424	0.298475	0.0310626
	time (s)	84.2434	1.8922	303.6298	63.0052	1.9077
0.1	iteration	100	2	91	100	5
	relerr	0.0617698	0.0621153	0.0617709	0.290838	0.0617950
	time (s)	63.7905	1.5078	242.1523	45.4830	2.9882
0.2	iteration	12	4	42	100	6
	relerr	0.120992	0.121752	0.120992	0.299642	0.12102
	time (s)	4.7328	1.8160	39.5247	27.2452	1.8032
0.3	iteration	12	6	44	100	8
	relerr	0.180622	0.181341	0.180624	0.295660	0.180645
	time (s)	3.0067	1.7068	27.0944	1.8032	27.2452

Table 5. Experimental results of algorithms when Jasper Ridge dataset was used and $\epsilon = 1.0 \times 10^{-8}$.

γ		HOOI	CrNc	HOSVD_NLS	BCD-CD	Algorithm 2
0.05	iteration	100	2	81	100	6
	relerr	0.0291179	0.0291398	0.0291179	0.155959	0.0291195
	time (s)	8.1761	0.2747	22.4445	4.7015	0.4218
0.1	iteration	10	2	63	100	11
	relerr	0.0591755	0.0592419	0.0591755	0.152589	0.0591765
	time (s)	0.6939	0.2399	10.3936	3.4566	0.5311
0.2	iteration	97	6	100	100	39
	relerr	0.118564	0.118782	0.118571	0.174686	0.118575
	time (s)	4.4129	0.6047	9.3648	2.4832	1.1770
0.3	iteration	9	15	39	100	15
	relerr	0.141258	0.141259	0.141258	0.179646	0.141259
	time (s)	0.3618	0.5357	3.1429	1.8633	0.3152

Table 6. Experimental results of algorithms when Indian Pines dataset was used and $\epsilon = 1.0 \times 10^{-8}$.

γ		HOOI	CrNc	HOSVD_NLS	BCD-CD	Algorithm 2
0.05	iteration	100	2	100	100	14
	relerr	0.0303899	0.0304928	0.0303903	0.071481	0.0303932
	time (s)	20.4660	0.5455	79.4811	11.8359	2.5751
0.1	iteration	9	8	89	100	12
	relerr	0.0527312	0.0527693	0.0527312	0.073079	0.0527339
	time (s)	0.9499	0.9088	18.6477	4.6742	0.8240
0.2	iteration	12	16	76	100	27
	relerr	0.0768604	0.0768700	0.0768604	0.124355	0.0768864
	time (s)	0.4698	0.8457	9.6352	2.1600	0.5941
0.3	iteration	4	4	18	100	6
	relerr	0.105915	0.105915	0.105915	0.126850	0.105915
	time (s)	0.1358	0.2284	2.2056	1.3507	0.1252

Table 7. Experimental results of algorithms when Urban dataset was used and $\epsilon = 1.0 \times 10^{-8}$.

γ		HOOI	CrNc	HOSVD_NLS	BCD-CD	Algorithm 2
0.05	iteration	100	2	100	100	9
	relerr	0.0310416	0.0311142	0.0310421	0.298475	0.0310468
	time (s)	84.8450	1.9226	665.8282	63.4775	6.4988
0.1	iteration	100	8	100	100	18
	relerr	0.0617698	0.0620033	0.0617706	0.290838	0.0617754
	time (s)	64.3367	5.3712	268.656	45.8471	9.2198
0.2	iteration	100	13	100	100	24
	relerr	0.120991	0.121355	0.120993	0.299642	0.120993
	time (s)	39.3584	5.5898	95.7108	27.6234	6.8905
0.3	iteration	38	13	100	100	27
	relerr	0.180622	0.180855	0.180622	0.29566	0.180625
	time (s)	9.5898	3.5682	61.3466	16.7007	4.4569

Table 8. Experimental results of algorithms when HSI dataset when Gaussian white noise was used and $\epsilon = 1.0 \times 10^{-6}$. The numbers in the parenthesis represent the average iteration numbers.

Dataset	γ	σ	HOOI	CrNc	HOSVD_NLS	BCD-CD	Algorithm 2
Jasper Ridge	0.1	+60 dB	0.0591755 (5)	0.0592419 (2)	0.0591757 (26)	0.153469 (100)	0.0591861 (2)
		+30 dB	0.0449392 (100)	0.0449707 (2)	0.0449392 (25.33)	0.152897 (100)	0.0449440 (2.67)
		+20 dB	0.0788476 (100)	0.0788524 (2)	0.0788469 (100)	0.118547 (100)	0.0788477 (2.67)
	0.3	+60 dB	0.141258 (5.67)	0.141296 (5.17)	0.141258 (23)	0.179359 (100)	0.141280 (6)
		+30 dB	0.137677 (5.83)	0.137713 (4.67)	0.137677 (24.33)	0.175483 (100)	0.137699 (5)
		+20 dB	0.113076 (6.33)	0.113200 (2.83)	0.113077 (32.5)	0.166953 (100)	0.113094 (4.33)
Indian Pines	0.1	+60 dB	0.0527313 (6.33)	0.0528347 (2)	0.0527315 (36)	0.0731127 (100)	0.0527554 (2)
		+30 dB	0.0491485 (6.83)	0.0492972 (2)	0.0491491 (100)	0.0739519 (100)	0.0491542 (5.16)
		+20 dB	0.0791087 (100)	0.0790856 (2)	0.0790949 (100)	0.0962606 (100)	0.0790843 (2.83)
	0.3	+60 dB	0.105915 (2)	0.105915 (2)	0.105915 (9)	0.127037 (50)	0.105915 (2)
		+30 dB	0.1053231 (2.67)	0.103234 (2)	0.103231 (11)	0.124117 (100)	0.103233 (5)
		+20 dB	0.0618722 (7.33)	0.0620632 (2.5)	0.0618740 (41.83)	0.0746809 (100)	0.0619321 (2.33)
Urban	0.1	+60 dB	0.0617699 (100)	0.0621156 (2)	0.0617710 (90.67)	0.289897 (100)	0.0618057 (5)
		+30 dB	0.0539840 (71.67)	0.0543273 (2)	0.0539852 (52.33)	0.296169 (100)	0.0540272 (4.5)
		+20 dB	0.0853106 (70.67)	0.08527384 (2)	0.0853042 (100)	0.302653 (100)	0.0852731 (2.83)
	0.3	+60 dB	0.180622 (13)	0.181286 (5.83)	0.180624 (43.5)	0.295660 (100)	0.180645 (8)
		+30 dB	0.178234 (12.83)	0.179252 (4.33)	0.178236 (36.17)	0.298176 (100)	0.178262 (5.5)
		+20 dB	0.158404 (68.33)	0.159271 (2)	0.158406 (20.83)	0.297824 (100)	0.158410 (7.16)

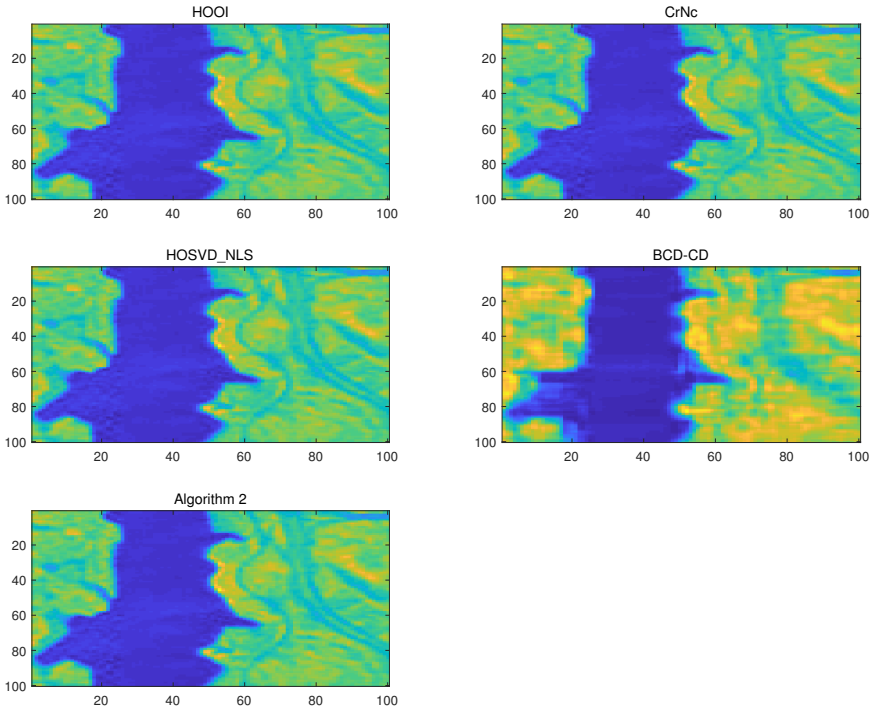
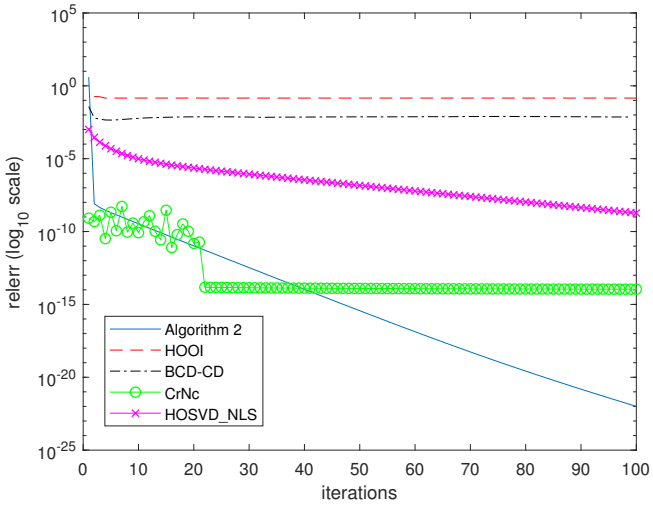


Figure 4. The first channel image of compressed HSI when the Jasper Ridge dataset was used; $\sigma = +20$ dB, $\gamma = 0.3$, and $\epsilon = 1.0 \times 10^{-6}$.

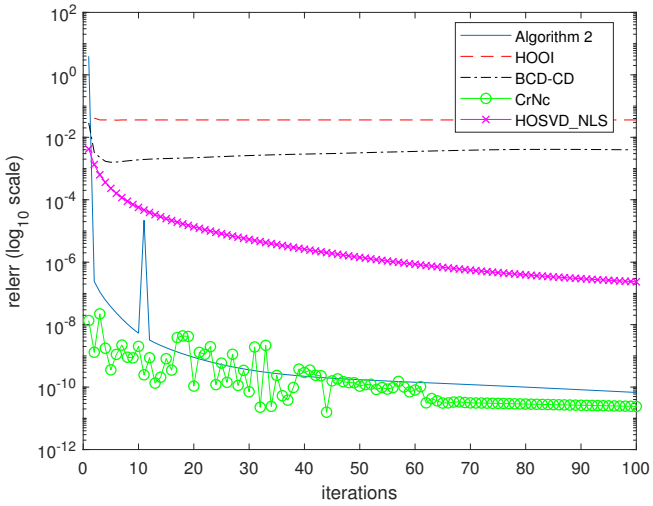
The last experiment examined how the algorithms would converge to the solutions. In the experiment, the algorithms were forced to continue until 100 iterations without considering the stopping criterion. Figures 5–8 depict the histories of convergences when $\gamma = 0.05$, $\gamma = 0.1$, $\gamma = 0.2$, $\gamma = 0.3$, respectively. Additionally, Table 9 shows the relative errors of outputs after the iterations reached 100 steps. In this experiment, even though the overall shapes of convergence histories from CrNc appear better than the others, the outputs of CrNc seem to converge to the relatively inaccurate local minimum, as shown in Table 9. The convergence speed of HOSVD_NLS is very slow according to this experiment, but no-meaningful differences with HOOI in terms accuracy were generated. Furthermore, HOOI produces unstable convergence history occasionally. In any case, Algorithm 2 produces robust outputs with stable convergence histories and with the accuracy close to that of HOOI.

Table 9. Relative errors when algorithms continue for 100 iterations.

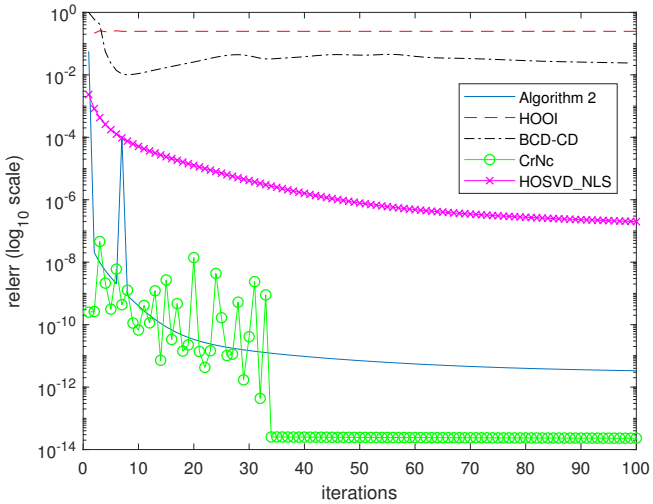
γ		HOOI	CrNc	HOSVD_NLS	BCD-CD	Algorithm 2
0.05	Jasper Ridge	0.0291179	0.0291283	0.0291179	0.155959	0.0291179
	Indian Pines	0.033899	0.0304179	0.0303900	0.071481	0.0303903
	Urban	0.0310416	0.0310716	0.0310418	0.298475	0.0310421
0.1	Jasper Ridge	0.0591755	0.0591779	0.0591755	0.152589	0.0591755
	Indian Pines	0.0527312	0.0527466	0.0527312	0.073079	0.0527312
	Urban	0.0617698	0.0618683	0.0617698	0.290838	0.0617706
0.2	Jasper Ridge	0.118564	0.118621	0.118564	0.161462	0.118571
	Indian Pines	0.0768604	0.0768700	0.0768604	0.124355	0.0768604
	Urban	0.120991	0.121298	0.120991	0.299642	0.120992
0.3	Jasper Ridge	0.141258	0.141258	0.141258	0.179646	0.141258
	Indian Pines	0.105915	0.105915	0.105915	0.126850	0.105915
	Urban	0.180622	0.180855	0.180622	0.295660	0.180622



(**a**) Jasper Ridge

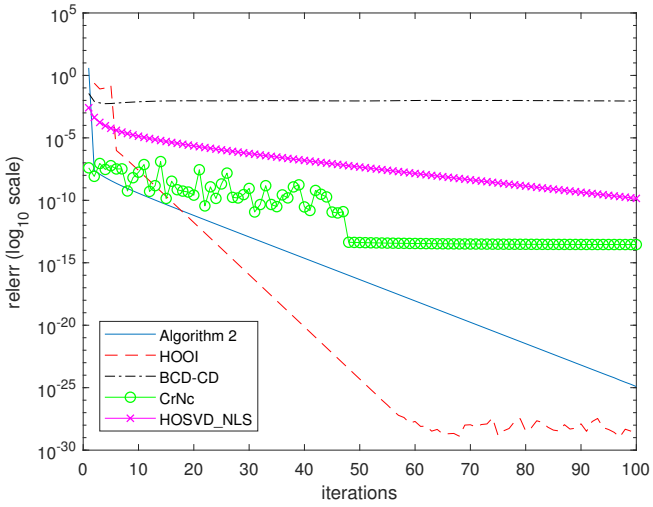


(**b**) Indian Pines

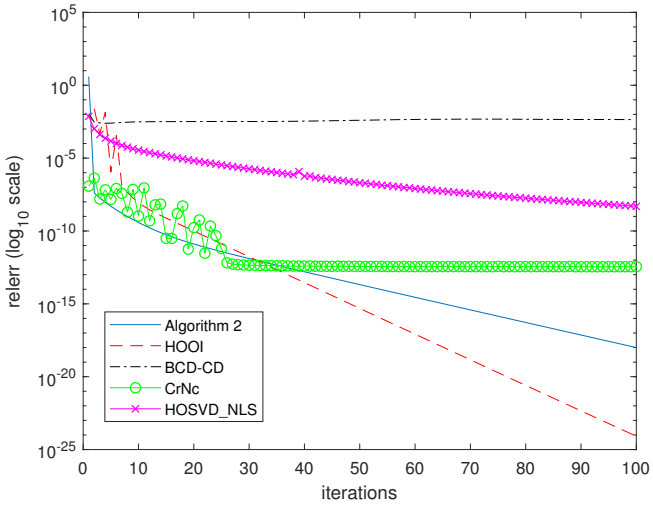


(**c**) Urban

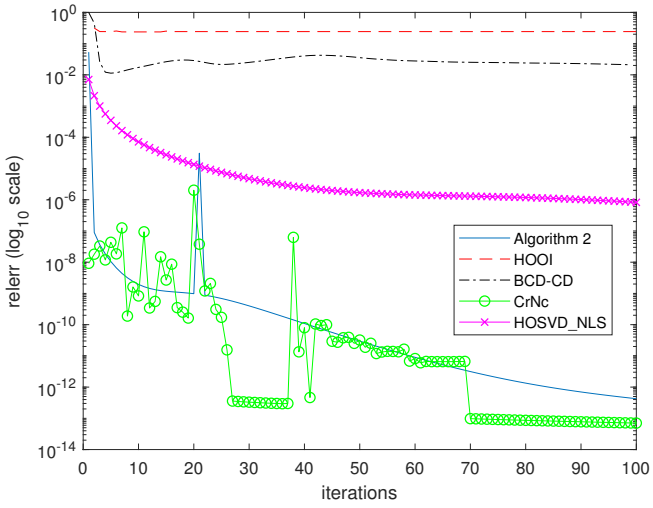
Figure 5. Convergence history of the algorithms when $\gamma = 0.05$.



(**a**) Jasper Ridge

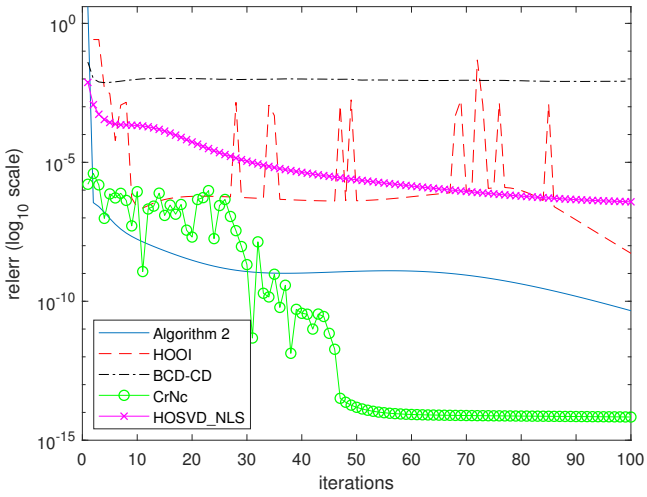


(**b**) Indian Pines

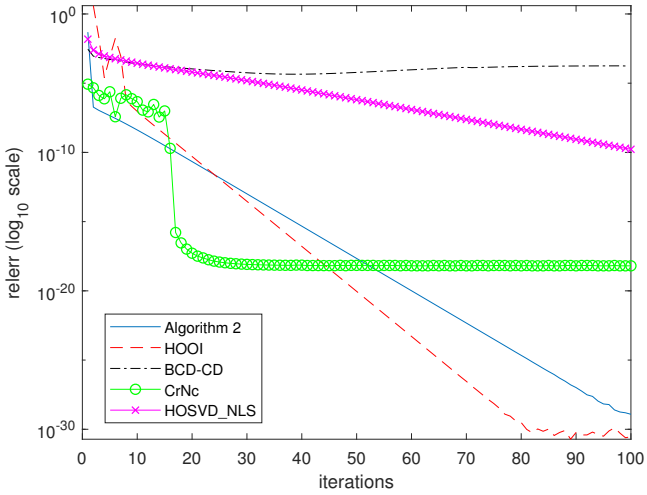


(**c**) Urban

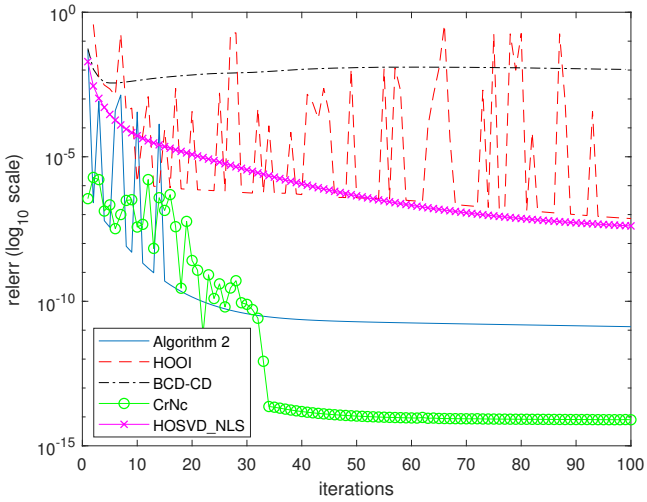
Figure 6. Convergence history of the algorithms when $\gamma = 0.1$.



(**a**) Jasper Ridge

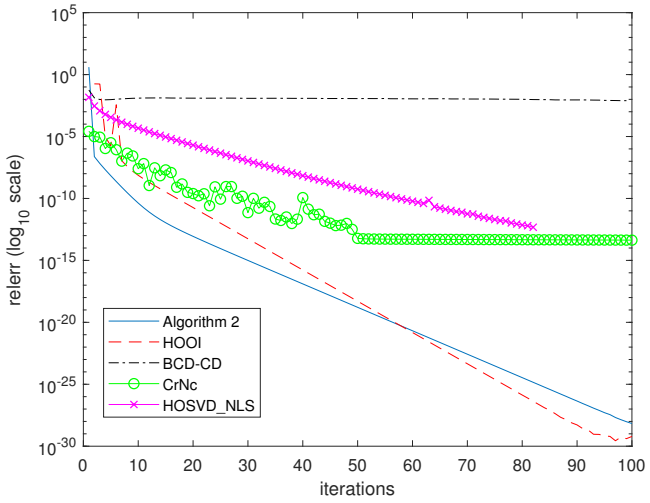


(**b**) Indian Pines

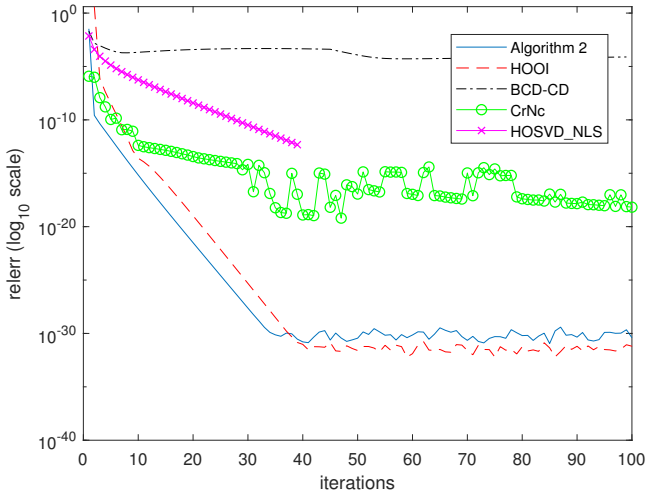


(**c**) Urban

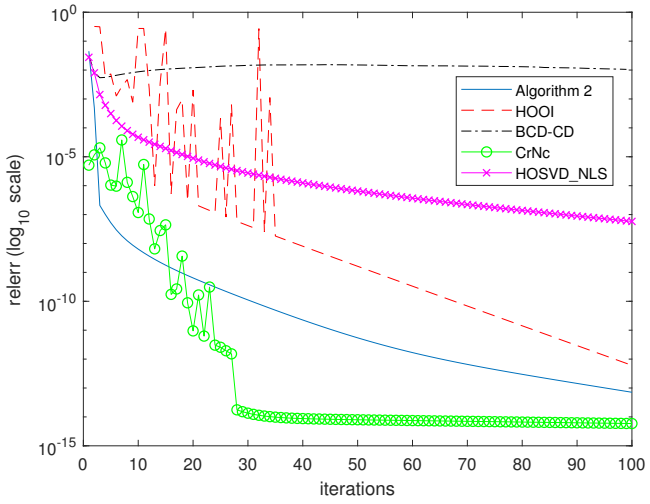
Figure 7. Convergence history of the algorithms when $\gamma = 0.2$.



(**a**) Jasper Ridge



(**b**) Indian Pines



(**c**) Urban

Figure 8. Convergence history of the algorithms when $\gamma = 0.3$.

6. Conclusions

Hyperspectral imaging is widely used, as it enables the simultaneous manipulation of the spatial and spectral distribution information of a target scene. Owing to the massive amount of information, tensor compression techniques such as higher order singular value decomposition must be applied. In this paper, we suggested an efficient computation method of higher order singular value decomposition by using sequential computations of an alternating least squares approach. Experiments on real-world hyperspectral imaging datasets highlight the faster computation of the proposed algorithm with no-meaningful difference in accuracy compared to higher order orthogonal iteration, which is typically known as the most accurate algorithm for computing higher order singular value decomposition.

Funding: This work was supported by Hankuk University of Foreign Studies Research Fund and the National Research Foundation of Korea (NRF) grant funded by the Korean government (2018R1C1B5085022).

Conflicts of Interest: The author declares no conflict of interest.

References

1. Bioucas-Dias, J.M.; Plaza, A.; Dobigeon, N.; Parente, M.; Du, Q.; Gader, P.; Chanussot, J. Hyperspectral Unmixing Overview: Geometrical, Statistical, and Sparse Regression-Based Approaches. *IEEE Sel. Top. Appl. Earth Obs. Remote. Sen.* **2012**, *5*, 354–379. [CrossRef]
2. Grahn, H.; Geladi, P. *Techniques and Applications of Hyperspectral Image Analysis*; John Wiley & Sons: New York, NY, USA, 2007.
3. Transon, J.; Andrimont, R.; Maugnard, A. Survey of Hyperspectral Earth Observation Applications from Space in the Sentinel-2 Context. *Remote Sens.* **2018**, *10*, 157. [CrossRef]
4. Lu, G.; Fei, B. Medical hyperspectral imaging: A review. *Biomed. Opt.* **2014**, *19*, 1–23. [CrossRef] [PubMed]
5. Lee, H.; Yang, C.; Kim, M.; Lim, J.; Cho, B.; Lefcourt, A.; Chao, K.; Everard, C. A Simple Multispectral Imaging Algorithm for Detection of Defects on Red Delicious Apples. *J. Biosyst. Eng.* **2014**, *39*, 142–149. [CrossRef]
6. Nasrabadi, N.M. Hyperspectral Target Detection. *IEEE Signal Process. Mag.* **2013**, *31*, 34–44. [CrossRef]
7. Poojary, N.; D'Souza, H.; Puttaswamy, M.R.; Kumar, G.H. Automatic target detection in hyperspectral image processing: A review of algorithms. In Proceedings of the 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Zhangjiajie, China, 15–17 August 2015; pp. 1991–1996.
8. Renard, N.; Bourennane, S. Dimensionality Reduction Based on Tensor Modeling for Classification Methods. *IEEE Trans. Geosci. Remote Sens.* **2009**, *47*, 1123–1131. [CrossRef]
9. Fang, L.; He, N.; Lin, H. CP tensor-based compression of hyperspectral images. *J. Opt. Soc. Am. A Opt. Image Sci. Vis.* **2017**, *34*, 252–258. [CrossRef] [PubMed]
10. Yan, R.; Peng, J.; Wen, D.; Ma, D. Denoising and dimensionality reduction based on PARAFAC decomposition for hyperspectral imaging. *Proc. SPIE Opt. Sens. Image. Tech. Appl.* **2018**, *10846*, 538–549.
11. Lathauwer, L.D.; Moor, B.D.; Vandewalle, J. A Multilinear Singular Value Decomposition. *SIAM J. Matrix Anal. Appl.* **2000**, *21*, 1253–1278. [CrossRef]
12. Kolda, T.G.; Bader, B.W. Tensor Decomposition and Applications. *Siam Rev.* **2009**, *51*, 455–500. [CrossRef]
13. Lathauwer, L.D.; Moor, B.D.; Vandewalle, J. On the Best Rank-1 and Rank-(R_1, R_2, ..., R_N) Approximation of Higher-Order Tensors. *SIAM J. Matrix Anal. Appl.* **2000**, *21*, 1324–1342. [CrossRef]
14. Zhang, L.; Zhang, L.; Tao, D.; Huang, X.; Du, B. Compression of hyperspectral remote sensing images by tensor approach. *Neurocomputing* **2015**, *147*, 358–363. [CrossRef]
15. An, J.; Lei, J.; Song, Y.; Zhang, X.; Guo, J. Tensor Based Multiscale Low Rank Decomposition for Hyperspectral Images Dimensionality Reduction. *Remote Sens.* **2019**, *11–12*, 1485. [CrossRef]
16. Eldén, L.; Savas, B. A Newton-Grassmann method for computing the Best Multi-Linear Rank-(r_1, r_2, r_3) Approximation of a Tensor. *Siam Matrix Anal. Appl.* **2009**, *31*, 248–271. [CrossRef]
17. Sorber, L.; Barel, M.V.; Lathauwer, L.D. Structured Data Fusion. *IEEE Sel. Top. Sig. Proc.* **2015**, *9*, 586–600. [CrossRef]
18. Hassanzadeh, S.; Karami, A. Compression and noise reduction of hyperspectral images using non-negative tensor decomposition and compressed sensing. *Eur. J. Remote Sens.* **2016**, *49*, 587–598. [CrossRef]

19. Phan, A.; Cichocki, A.; Tichavsky, P. On Fast Algorithms for Orthogonal Tucker Decomposition. In Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014; pp. 6766–6770.
20. Lee, G. Fast computation of the compressive hyperspectral imaging by using alternating least squares methods. *Sig. Proc. Image Commun.* **2018**, *60*, 100–106. [CrossRef]
21. Zhu, F.; Wang, Y.; Xiang, S.; Fan, B.; Pan, C. Structured Sparse Method for Hyperspectral Unmixing. *ISPRS J. Photogramm. Remote Sens.* **2014**, *88*, 101–118. [CrossRef]
22. Baumgardner, M.F.; Biehl, L.L.; Landgrebe, D.A. 220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3. *Purdue Univ. Res. Repos.* **2015**, *10*, R7RX991C.
23. Xu, Y.; Yin, W. A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion. *Siam Imag. Sci.* **2013**, *6*, 1758–1789. [CrossRef]
24. Vervliet,N.; Devals, O.; Sorber, L.; Van Barel, M.; De Lathauwer, L. *Tensorlab 3.0*. Available online: https://www.tensorlab.net/ (accessed on 10 March 2016).