IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 13, 2021, accepted October 22, 2021, date of publication November 2, 2021, date of current version November 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3125069

Cross Tensor Approximation Methods for Compression and Dimensionality Reduction

SALMAN AHMADI-ASL[1], CESAR F. CAIAFA[2], ANDRZEJ CICHOCKI[1,3] (Life Fellow, IEEE), ANH HUY PHAN[1], (Member, IEEE), TOSHIHISA TANAKA[4], (Senior Member, IEEE), IVAN OSELEDETS[1], AND JUN WANG[1]

[1]CDISE, Skolkovo Institute of Science and Technology (SKOLTECH), 121205 Moscow, Russia
[2]Instituto Argentino de Radioastronomía—CCT La Plata, CONICET/CIC-PBA/UNLP, Villa Elisa 1894, Argentina
[3]RIKEN Center for Advanced Intelligence Project (AIP), Tokyo 103-0027, Japan
[4]Department of Electrical and Electronic Engineering, Tokyo University of Agriculture and Technology, Tokyo 183-8509, Japan

Corresponding author: Salman Ahmadi-Asl (s.asl@skoltech.ru)

This work was supported in part by the Ministry of Education and Science of the Russian Federation under Grant 075.10.2021.068. The work of Cesar F. Caiafa was supported in part by the Grant PICT 2017-3208 and Grant UBACYT 20020170100192BA (Argentina).

ABSTRACT Cross Tensor Approximation (CTA) is a generalization of Cross/skeleton matrix and CUR Matrix Approximation (CMA) and is a suitable tool for fast low-rank tensor approximation. It facilitates interpreting the underlying data tensors and decomposing/compressing tensors so that their structures, such as nonnegativity, smoothness, or sparsity, can be potentially preserved. This paper reviews and extends state-of-the-art deterministic and randomized algorithms for CTA with intuitive graphical illustrations. We discuss several possible generalizations of the CMA to tensors, including CTAs: based on fiber selection, slice-tube selection, and lateral-horizontal slice selection. The main focus is on the CTA algorithms using Tucker and tubal SVD (t-SVD) models while we provide references to other decompositions such as Tensor Train (TT), Hierarchical Tucker (HT), and Canonical Polyadic (CP) decompositions. We evaluate the performance of the CTA algorithms by extensive computer simulations to compress color and medical images and compare their performance.

INDEX TERMS CUR algorithms, cross approximation, tensor decomposition, tubal SVD, randomization.

I. INTRODUCTION

Tensor decompositions are efficient and widely used tools for multi-way data processing (analysis) and, in particular, they can be utilized to compress the data tensors without destroying their intrinsic multidimensional structure. In the past few decades, several types of tensor decompositions have been introduced, such as CANDECOMP/PARAFAC also called Canonical Polyadic Decomposition (CPD) [1], [2], Tucker decomposition [3]–[5] and its special case, Higher-Order SVD (HOSVD) [6], Block Term decomposition (BTD) [7]–[9], Hierarchical Tucker (HT) decomposition [10], [11], Tensor Train/Tensor Chain (TT-TC) decomposition [12]–[14], tubal SVD (t-SVD) [15]–[17], each of which generalizes the notion of matrix rank to tensors in an efficient way. They have been successfully applied in many applications such as signal processing [18], machine learning [19], blind source separation [18], [20], chemometrics, [21], feature extraction/selection [22]. For a comprehensive study on tensors and their applications, we refer to [23]–[25]. It is of interest to decompose a data tensor in such a way that some parameters of the underlying tensor decomposition, e.g., the factor matrices or the core tensor of the Tucker decomposition, preserve the structure of the original data tensor, e.g., smoothness, nonnegativity, or sparsity. The main reasons for such interests are 1- fast low tensor rank approximation, 2- achieving higher compression ratio, and 3- data interpretation issues. *Skeleton* or *Cross tensor/matrix approximation* (CTA/CMA) or equivalently tensor/matrix CUR approximation is an efficient framework for attaining the mentioned goals. The main characteristic of the CTA algorithms, which makes them useful for handling very large-scale data tensors, is to occupy less memory and reduce computational complexity. Regarding the higher compression capability, for example, in the case of matrices (second-order tensors), the SVD of sparse matrices does not provide sparse factor matrices while

The associate editor coordinating the review of this manuscript and approving it for publication was Manuel Rosa-Zurera.

the cross-skeleton approximation achieves this goal, leading to more compact data representation. The CTA methods have similar properties for compressing structured data tensors. Besides, regarding the interpretation issue, for example, as mentioned in [26], a vector $[1/2]\,age - (1/\sqrt{2})\,height + (1/2)\,income$ being as one of the significant uncorrelated factors for a data set of people's features is difficult to be interpreted. Kuruvilla *et al.* in [27] also claimed: "it would be interesting to try to find basis vectors for all experiment vectors, using actual experiment vectors and not artificial bases that offer little insight." This motivates computing a low-rank approximation as close as SVD but with individual columns/rows of the original data matrix; see [26], [27], for a detailed discussion on this topic.

In most types of tensor decompositions, there are no guarantees that the factor matrices or core tensors involved in the decompositions preserve the structure of the underlying data tensors unless additional constraints are imposed. As a result, the decomposition procedure should be formulated as constrained optimization problems. For instance, the l_1-norm as the tightest convex surrogate of l_0 norm imposes sparsity constraint on factor matrices and/or core tensor [28]. An alternative to the optimization approach is the Cross/Skeleton technique which uses a part of the data tensor to compute a tensor decomposition. The CMA methods compute a low-rank approximation based on a part of individual columns and rows. They have found applications in deep learning [29], signal processing [30], [31], scientific computing [32]–[34] and machine learning [35]–[37]. A closely related matrix approximation is called *matrix column selection* or *interpolative matrix decomposition*. Here, just a part of columns are sampled [38]–[44], and it can be considered as a special CMA. The CMA/CTA can be used similarly to approximate continuous functions; however, throughout the paper we only focus on the discrete variables (for details, please see [45]–[47]).

For the CMA methods, the problem is not formulated as a constrained optimization problem; but instead, a matrix is represented based on a part of its columns and rows. Generally, this can be done based on a part of its components which includes rows and columns as special cases. The sampled matrix is also called sketched matrix. Four main algorithms for the CMA are, (see Figure 1):

- Maxvol algorithm [48], [49]
- Cross2D algorithm [50], [51]
- Discrete Empirical Interpolation Method (DEIM) [52], [53]
- Random sampling algorithms [54].

The main difference between these algorithms is the procedure of column or row selection. The Maxvol, Cross2D, and DEIM algorithms are deterministic, while the random sampling algorithms are probabilistic (e.g., using uniform distribution). The DEIM algorithm needs estimations of the top left/right singular vectors to proceed, and this is also required in a kind of probabilistic sampling technique, i.e., leverage scores [54]. The Maxvol and Cross2D algorithms do not need the top singular vectors, and they heuristically look for optimal or the most representative columns and rows.

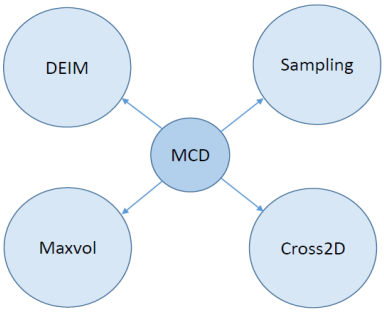

FIGURE 1. Four main categories for CMA approaches.

The main building block of all algorithms for tensor decomposition is computing low-rank approximation of unfolding matrices [23], [24]. Here, of course, the CMA algorithms (either deterministic or randomized algorithms) can be utilized instead of expensive alternative techniques, e.g., SVD. For a comprehensive study on randomized algorithms for the Tucker decomposition and TT-TR decomposition, we refer to [55]–[57]. Motivated by the limitation of algorithms for handling big data tensors and the need for fast low-rank approximation methods, the classical CMA was generalized to tensors in [58]–[63]. The work [60] is related to the TT decomposition, whereas the works [59], [61], [63] are for the Tucker decomposition and [62] is associated with the t-SVD. These generalizations have been made from different perspectives and to the best of our knowledge, there is no survey paper discussing their similarities and differences. In particular, it seems that it is rather difficult to distinguish one CTA from another. This paper attempts to achieve this goal where different CTA algorithms and their properties are unified and discussed in detail. Moreover, there is a lack of graphical illustrations in the literature for describing the CTA algorithms making the topic difficult to understand, especially for those who have an engineering background. Here, we provide some useful and intuitive illustrations with unifying notations. We mainly discuss three possible generalizations of the CMA to tensors including, see Figure 2:

- Fiber selection
- Slice and tube selection
- Lateral and horizontal slices selection

where the last one deals with tubal product (t-product) [15]–[17], [64].

In general, fiber or slice selection can be performed in either a deterministic or randomized way. If fibers or slices are sampled based on prior probability distributions such as uniform, length-squared, or leverage score probabilities [54], then the algorithms are referred to as randomized CTA; otherwise, they are called deterministic CTA. However, for the brevity of the presentation, in the rest of the paper, we only use the CTA term and highlight when the underlying selection is randomized. In the randomized CMA case, the accuracy of

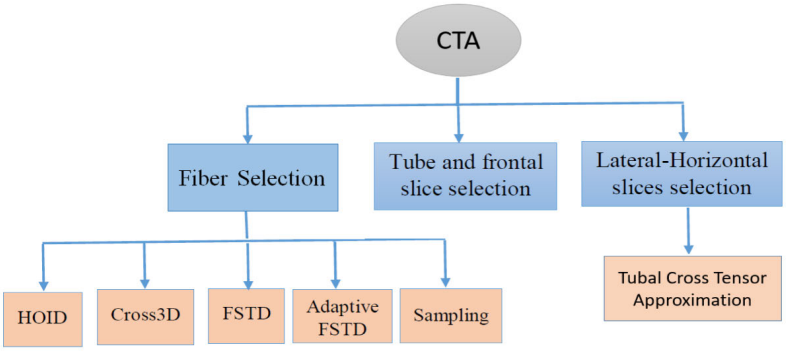

FIGURE 2. Different types of CTA approaches.

obtained approximations highly depends on the prior probability distributions [54]. Actually, it turns out that this is also true for slice-fiber selection. Please note that some properties of the underlying data matrix determine which sampling strategy should be used. For example, uniform sampling works well for matrices with low coherence [54]. The best existing random sampling algorithms [54] use leverage score probabilities, for which the relative-error approximation is achieved. However, they require the computation of top singular values, which is computationally expensive. It is known that randomized algorithms facilitate the tensor decomposition procedure not only by reducing the computational complexity of deterministic algorithms but also by reducing the communication among different levels of memories, which is the main bottleneck in modern computing environments and architectures for handling large-scale data tensors.

The structure of this paper is organized as follows. In Section II, we introduce basic definitions and concepts used throughout the paper. In Section III, we discuss the CTA algorithms and several generalizations of the CMA to tensors. Section IV, describes the tubal CTA, which is defined based on t-product. The number of parameters of the CTA models and the space/time complexity of the CTA algorithms are discussed in Section V. Section VI, compares the CTA algorithms in the sense of the number of passes to compute a low-rank tensor approximation. The main challenges and future research directions are discussed in Section VIII. Simulations are provided in Section IX to validate and evaluate the performance of the presented algorithms. Finally, a conclusion is drawn in Section X.

II. PRELIMINARIES

In this section, we present notations and concepts used throughout the paper. Tensors, matrices, and vectors are denoted by underlined bold upper case letters, e.g., $\underline{\mathbf{X}}$, bold upper case letters, e.g., $\mathbf{X}$, and bold lower case letters, e.g., $\mathbf{x}$, respectively. Fibers are first-order tensors produced by fixing all modes except one, while slices are second-order tensors generated by fixing all modes except two of them. For a third-order tensor $\underline{\mathbf{X}}$, the slices $\underline{\mathbf{X}}(:,:,k)$, $\underline{\mathbf{X}}(:,j,:)$, $\underline{\mathbf{X}}(i,:,:)$ are called frontal, lateral and horizontal slices, respectively. Fibers $\mathbf{X}(i,:,j)$, $\mathbf{X}(:,j,k)$ and $\mathbf{X}(i,j,:)$ are called rows, columns, and tubes, respectively [23], see Figure 3 for graphical illustration of slice and fiber concepts for a 3rd order tensor. The rows, columns, and tubes are also called mode-1, mode-2, and mode-3 fibers. In general, for an N order tensor, N types of fibers can be defined, i.e., n-mode fibers for $n = 1, 2, \ldots, N$. The notations $\mathbf{X}^{+}$ and $\mathbf{X}^{T}$ denote the Moore-Penrose pseudoinverse (MP[1]) and the transpose of matrix $\mathbf{X}$, respectively. The Frobenius and Chebyshev (infinity) norms of tensors/matrices are denoted by $\|.\|_F$ and $\|.\|_\infty$. The first norm is the square root of the sum of the square of tensor components, while the second is the maximum of the absolute value of tensor components. We use $|\mathbf{X}|$ to indicate a matrix whose components are the absolute value of the components of the data matrix $\mathbf{X}$, i.e., $y_{i,j} = |x_{i,j}|$ where $\mathbf{Y} = |\mathbf{X}|$. The notation $[N]$ denotes the set of natural numbers $[1, 2, \ldots, N]$ and $n(A)$ means the number of elements of the set A. For the simplicity of the presentation, all notations used in the paper are summarized in Table 1.

Definition 1: n-unfolding[2] matrices: Given an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, then the n-unfolding of the sketched data tensor $\underline{\mathbf{X}}$ is denoted by $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$, whose components are

$$\mathbf{X}_{(n)}(i_n, j) = \underline{\mathbf{X}}(i_1, i_2, \ldots, i_N),$$

where

$$j = 1 + \sum_{k=1,\, k\neq n}^{N} (i_k - 1) J_k, \quad J_k = \prod_{m=1,\, m\neq n}^{k-1} I_m.$$

This is an element-wise representation of the n-unfolding transformation.

The n-unfolding operator can be introduced in a more understandable way by stacking the mode-n fibers of a tensor. See Figure 4 for a graphical illustration for this concept for a 3rd order tensor. The reverse of this procedure is called

[1]The MP concept is also defined for tensors based on the t-product. (see Section IV.)

[2]It is also called matricization or flattening.

TABLE 1. The notations and symbols used in the paper.

Symbol	Definition
$\underline{\mathbf{X}}$, $\mathbf{X}$, $\mathbf{x}$	Tensor, matrix, vector
$\underline{\mathbf{X}} * \underline{\mathbf{Y}}$	t-product of tensors $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$
$\mathbf{X}_{(n)}$	n-mode unfolding of a matrix
$\underline{\mathbf{X}} \times_n \mathbf{B}$	n-mode product of tensor $\underline{\mathbf{X}}$ and matrix $\mathbf{B}$
$(.)^+$	Moore-Pseudo inverse of tensors (matrices)
$(.)^T$	Transpose of tensors (matrices)
$\lvert\mathbf{X}\rvert$	Absolute value of matrix components
$\lVert\mathbf{X}\rVert_F^2$	Frobenius norm of matrix X
$\underline{\mathbf{X}}(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_N)$	Intersection subtensor of $\underline{\mathbf{X}}$ with indices in mode k
[I]	The set of natural numbers [1,2,...,I]
n(A)	The number of elements of set A
fold_n	n-folding of a matrix
$\lVert\underline{\mathbf{X}}\rVert_F$	Frobenious norm of tensor $\underline{\mathbf{X}}$
$\lVert\underline{\mathbf{X}}\rVert_\infty$	Infinity norm of tensor $\underline{\mathbf{X}}$

n-folding which transforms a matrix into a tensor. There are MATLAB libraries to perform these operations, such as tensorlab [65] and tensor toolbox [66].

For a given data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, an intersection subtensor is produced by considering only a part of indices in the modes of the original data tensor $\underline{\mathbf{X}}$. For example, if $\mathcal{I}_n \subset [I_n]$, $n = 1, 2, \ldots, N$, then the subtensor $\underline{\mathbf{X}}(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_N) \in \mathbb{R}^{n(\mathcal{I}_1) \times n(\mathcal{I}_2) \times \cdots \times (\mathcal{I}_N)}$ can be generated.

Tensors and matrices can be multiplied in modes that have the same dimensions. This is called tensor *mode-n product* and is a generalization of matrix-matrix multiplication. To be precise, let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathbf{B} \in \mathbb{R}^{J \times I_n}$, then the tensor-matrix multiplication along mode n is denoted by

$$\underline{\mathbf{X}} \times_n \mathbf{B} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N},$$

and defined as follows

$$\left(\underline{\mathbf{X}} \times_n \mathbf{B}\right)_{i_1, \ldots, i_{n-1}, j, i_{n+1}, \ldots, i_N} = \sum_{i_n=1}^{I_N} x_{i_1, i_2, \ldots, i_N} b_{j, i_n},$$

for $j = 1, 2, \ldots, J$. It can be shown that $\underline{\mathbf{X}} \times_m \mathbf{A} \times_n \mathbf{B} = \underline{\mathbf{X}} \times_n \mathbf{B} \times_m \mathbf{A}$ if $m \neq n$ and when $m = n$ we have

$$\underline{\mathbf{X}} \times_n \mathbf{A} \times_n \mathbf{B} = \underline{\mathbf{X}} \times_n (\mathbf{BA}).$$

The tensor-matrix multiplication can be done in the unfolding form. Given a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\mathbf{A} \in \mathbb{R}^{K \times I_n}$, then we have

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_n \mathbf{A} \Longleftrightarrow \mathbf{Y}_{(n)} = \mathbf{A}\mathbf{X}_{(n)}. \tag{1}$$

Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be a given data tensor, then the Tucker decomposition model is defined as follows: [3]–[5]

$$\underline{\mathbf{X}} \cong \underline{\mathbf{S}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)}, \tag{2}$$

where $\underline{\mathbf{S}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ is called the core tensor, and the matrices $\mathbf{A}_n \in \mathbb{R}^{I_n \times R_n}$, $R_n \leq I_n$, $n = 1, 2, \ldots, N$ are called factor matrices. A shorthand notation for the Tucker decomposition is

$$\underline{\mathbf{X}} = \left[\!\left[\underline{\mathbf{S}}; \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N\right]\!\right]. \tag{3}$$

The N-tuple $(R_1, R_2, \ldots, R_N)$ is called multilinear or Tucker rank where R_n is the rank of the mode-n unfolding matrix $\mathbf{X}_{(n)}$, $n = 1, 2, \ldots, N$. For a matrix $\mathbf{X}$ as a second-order tensor, the rank of $\mathbf{X}_{(1)}$ and $\mathbf{X}_{(2)}$ are the same because $\mathbf{X}_1 = \mathbf{X}_2^T$. For tensors of order higher than 2, the components of the multilinear rank can be different [6]. Higher-order SVD (HOSVD) [6] is a special Tucker decomposition in which the factor matrices are orthogonal, and the core tensor $\underline{\mathbf{S}}$ satisfies two properties called *all-orthogonality* and *pseudo-diagonality* [6]. It is not difficult to see that for the case of matrices, the SVD of a matrix $\mathbf{X}$ can be regarded as a special case of the HOSVD, $\mathbf{U}_R \Sigma_R \mathbf{V}_R^T = [\![\Sigma_R; \mathbf{U}_R, \mathbf{V}_R]\!] \equiv \Sigma_R \times_1 \mathbf{U}_R \times_2 \mathbf{V}_R$, where $\mathbf{U}_R$ and $\mathbf{V}_R$ are the matrices containing the top left and right singular vectors, respectively, and Σ is a diagonal matrix containing the top singular values.

A. CROSS MATRIX APPROXIMATION (CMA) AND MATRIX COLUMN SELECTION

Theory and algorithms for low-rank matrix approximation based on a part of its individual columns and rows were first developed in [67]. This framework for low-rank approximation of matrices is known as *skeleton/cross matrix approximation*. It is called cross approximation because the matrix intersecting the columns and rows is used within the approximation procedure. As we will discuss later, the procedure of column or row selection highly affects the approximation error. In the cross/skeleton approximation, the columns and rows are selected in heuristic ways and using deterministic algorithms. If the columns or rows are selected based on random sampling techniques, this framework is often known as randomized CUR decomposition.

Given a data matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$, let us select R_1 columns and R_2 rows from $\mathbf{X}$ with corresponding indices $\mathcal{I}$ and $\mathcal{J}$,

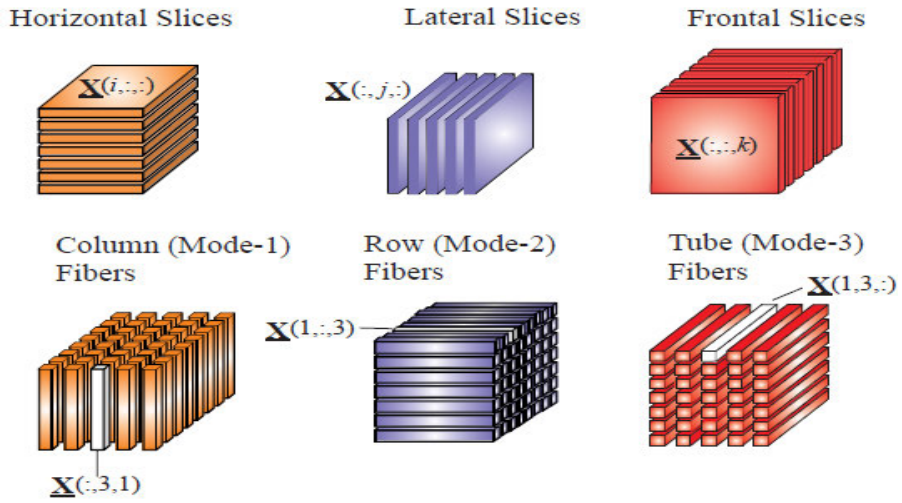

FIGURE 3. Illustration of different types of slices/fibers for a 3rd order tensor [23].

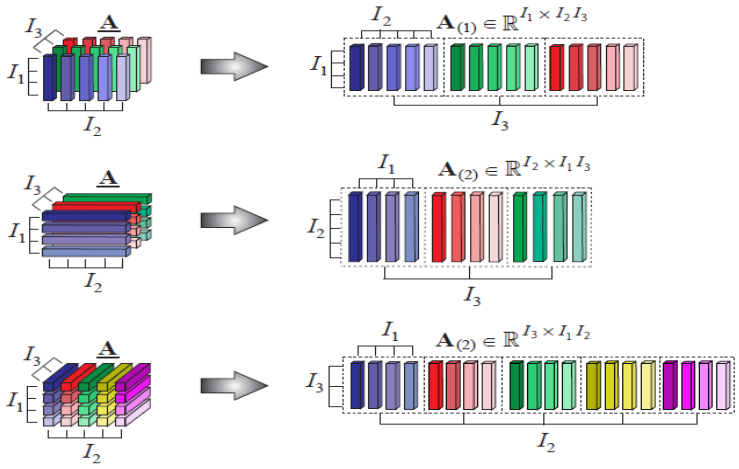

FIGURE 4. Illustration of *n*-unfolding operator for a 3rd order tensor [23].

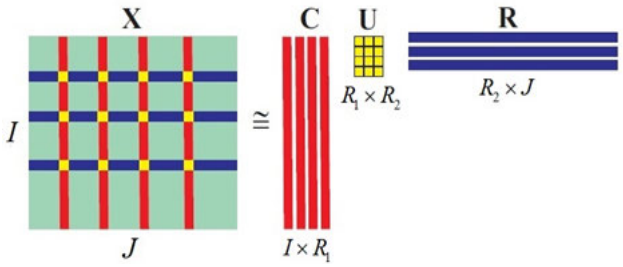

FIGURE 5. Illustration of CMA for low-rank matrix approximation $\mathbf{X} \cong \mathbf{CUR}$ where $\mathbf{U} = \mathbf{W}^{+}$ [23].

respectively. Assume that the selected columns and rows are stored as $\mathbf{C} \in \mathbb{R}^{I \times R_1}$, $\mathbf{R} \in \mathbb{R}^{J \times R_2}$ respectively, then the intersection submatrix is $\mathbf{W} = \mathbf{X}(\mathcal{I}, \mathcal{J}) \in \mathbb{R}^{R_1 \times R_2}$, see Figure 5. The low-rank CMA is computed as follows:

$$\mathbf{X} \cong \mathbf{CUR} = \mathbf{U} \times_1 \mathbf{C} \times_2 \mathbf{R}, \tag{4}$$

where $\mathbf{U} \in \mathbb{R}^{R_1 \times R_2}$ should be computed to yield the smallest error. Two main motivations for using such an approximation are

- Fast low-rank approximation
- Compression and interpretability issues in which the factors $\mathbf{C}$ and $\mathbf{R}$ should have the same structure as the original data matrix $\mathbf{X}$.

Concerning the second motivation, note that the middle matrix $\mathbf{U}$ does not necessarily preserve the structure of the data matrix, and only $\mathbf{C}$ and $\mathbf{R}$ have this property. As a result, for nonnegative matrices, one should distinguish this with the nonnegative matrix factorization in which everything should be nonnegative. However, sometimes it is desirable that only $\mathbf{C}$ and $\mathbf{R}$ have the same structure as the original data matrix and not the middle matrix $\mathbf{U}$. For example, for sparse data matrices, the factor matrices $\mathbf{C}$ and $\mathbf{R}$ are sparse and a higher compression rate than SVD can be achieved. The interpretability issue is another application of

these techniques where the data matrix is represented as a linear combination of individual columns, and it can be better interpreted. The best middle matrix $\mathbf{U}$ in the least-squares sense is [67]

$$\mathbf{U} = \mathbf{C}^{+}\mathbf{X}\mathbf{R}^{+}, \tag{5}$$

for which the approximation is exact if $\text{rank}(\mathbf{X}) \leq \min\{R_1, R_2\}$. However, this is of less practical interest because we need to access the whole data matrix $\mathbf{X}$. This causes expensive communication costs, especially when the data matrix is stored out-of-core and the cost of communication may exceed our main computations. An alternative and more efficient method is to use the intersection submatrix $\mathbf{W}$ and compute $\mathbf{U} = \mathbf{W}^{+}$. The two mentioned techniques are not necessarily equivalent, but for a data matrix with exact rank (noiseless) in [68], the conditions under which they will be the same are discussed. For the noisy data matrices, Formula (5) provides a better approximation. When the intersection submatrix is square and nonsingular, the approximation is known as *skeleton*, but when it is singular, the algorithm is called *pseudo-skeleton* approximation. Here, the CMA $\mathbf{X} \cong \mathbf{C}\mathbf{W}^{+}\mathbf{R}$ is used, which is computationally less expensive than the approach using Formula (5). The first approximation is called a two-passes algorithm, while the second is called a single-pass algorithm. Computing the Moore-Penrose (MP) pseudoinverse of matrices incurs instability issues, especially when they are ill-conditioned. Because of this, a well-conditioned intersection submatrix should be selected. It is known that the quality of the intersection submatrix quite depends on the module of the determinant of the intersection submatrix, which is called *matrix volume*.[3] More precisely, a set of columns and rows should be selected whose intersection submatrix has as much volume as possible. Clearly, this is an NP-hard problem because we need to check the volume of all possible intersection matrices produced by different selections of columns on rows. However, heuristic algorithms do exist for computing suboptimal solutions [69]–[72]. Three known heuristic CMA algorithms are:

- Maxvol-based algorithm [48], [49]
- Cross2D algorithm [50], [51]
- Discrete Empirical Interpolatory Method (DEIM) [52], [53].

As mentioned earlier, these selection techniques are deterministic, and one should distinguish them from the random sampling approaches [54], [73], [74] or the random projection approaches [75], which basically are randomized algorithms. The DEIM algorithm was first introduced in model order reduction [76], and deals only with column selection in which an estimation of the top right singular vectors are required, while in the Cross2D and the maxvol-based low-rank approximations, both columns, and rows are involved.

[3]The volume of a square matrix $\mathbf{X}$ is defined as $|det(\mathbf{X})|$ where "*det*" denotes the matrix determinant. For the rectangular matrices, the volume is defined as the absolute value of the product of singular values.

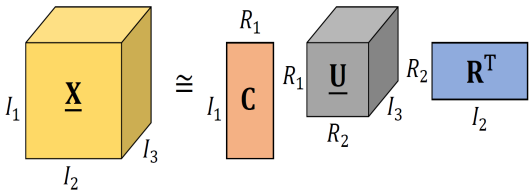

FIGURE 6. Illustration of the CTA based on Tucker-2 decomposition for a 3rd-order tensor ((R_1, R_2) denotes the Tucker-2 rank).

Remark 1: If a given matrix $\mathbf{X}$ is positive semi-definite, then the CMA approach is called the Nyström method, i.e., $\mathbf{C} = \mathbf{R}$, and has several applications in machine learning and data science [77]–[80].

A special case of the CMA in which only columns are sampled is called *matrix column selection*, *interpolative matrix decompositions*, and in some contexts, it is also referred to as CX decomposition. This problem is known as column (feature) selection in the field of machine learning and data analysis. Let $\mathbf{X} \in \mathbb{R}^{I\times J}$ is a given data matrix and $\mathbf{C} \in \mathbb{R}^{I\times R}$ is a matrix containing the R selected columns from the matrix $\mathbf{X}$. Then the matrix column selection is formulated as follows

$$\mathbf{X} \cong \mathbf{C}\widetilde{\mathbf{X}}, \tag{6}$$

where $\mathbf{C} \in \mathbb{R}^{I\times R}$ is a matrix containing the selected columns and $\widetilde{\mathbf{X}} \in \mathbb{R}^{R\times J}$ should be computed in such a way that the approximate error should be as small as possible. The best solution to the problem (6) in the least-squares (LS) sense is $\widetilde{\mathbf{X}} = \mathbf{C}^{+}\mathbf{X}$, and if $rank(\mathbf{X}) = R$, then this approximation is exact, i.e., $\mathbf{X} = \mathbf{C}\mathbf{C}^{+}\mathbf{X}$.

III. CROSS TENSOR APPROXIMATION (CTA)

Existing tensor decompositions such as HOSVD, CPD, TT/TR decompositions without constraints do not preserve the natural structure of the original data tensors, such as non-negativity or sparsity. To solve this problem, we can impose additional constraints or use cross-types tensor decompositions, which are based on a part of its fibers or slices (or both of them). Here, the latter approach is taken. In the next subsequent sections, we explain how the CMA can be generalized to tensors.

A. CTA BASED ON FIBER SELECTION

A possible generalization of the CMA to tensors in the sense of selecting only columns and rows can be performed based on the Tucker-2 decomposition in which the compression is performed only in the first and second modes. Given a 3rd order data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I\times J\times K}$, here, the factor matrices $\mathbf{C} \in \mathbb{R}^{I\times R_1}$ and $\mathbf{R} \in \mathbb{R}^{J\times R_2}$ contain the selected columns and rows and the middle core tensor $\underline{\mathbf{U}} \in \mathbb{R}^{R_1\times R_2\times K}$ should be computed to yield the smallest error (see Figure 6). The Tucker-2 rank is accordingly defined as (R_1, R_2). The optimal middle core tensor can be computed as follows

$$\underline{\mathbf{U}} = \underline{\mathbf{X}} \times_1 \mathbf{C}^{+} \times_2 \mathbf{R}^{+}.$$

Natural and straightforward generalizations of the CMA to tensors are proposed in [59], [81], and [61]. Quite similar to the CMA in which parts of columns and rows of a given matrix are sampled, here a set of fibers (along different modes) are selected, and the goal is computing a Tucker approximation based on these sampled fibers. For instance, for 3rd-order tensors, we should sample columns, rows, and tubes. The approach proposed in [81] is randomized, while those proposed in [59], [61] are deterministic. These works can be considered as the starting points of generalization of the CMA to tensors, and in the sequel, we explain the idea behind each of these Tucker approximations.

For noiseless data tensor, the existence of an exact Tucker model whose factor matrices are taken from the fibers of the original data tensor is straightforward. To be more precise, let $\underline{\mathbf{X}}$ be an Nth-order tensor of size $I_1 \times I_2 \times \cdots \times I_N$ and of Tucker rank $(R_1, R_2, \ldots, R_N)$. Now, if we generate any full-rank factor matrices $\mathbf{A}_n \in \mathbb{R}^{I_n \times R_n}$, $n = 1, 2, \ldots, N$, by sampling fibers in each mode and computing the core tensor as

$$\underline{\mathbf{S}} = \underline{\mathbf{X}} \times_1 \mathbf{A}_1^+ \times_2 \mathbf{A}_2^+ \cdots \times_N \mathbf{A}_N^+ \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}, \quad (7)$$

then the obtained Tucker decomposition has the exact Tucker rank $(R_1, R_2, \ldots, R_N)$. So an exact Tucker decomposition of the tensor $\underline{\mathbf{X}}$ whose factor matrices were taken from the original data tensor is computed. For noisy tensors, similar to the CMA, the accuracy of approximation depends on the number of selected fibers and the list of sampled fibers. The idea of sampling fibers and considering them as the factor matrices, first was proposed in [81]. In the first step, the factor matrices are generated, after which the core tensor is computed through (7) as described above. This is summarized in Algorithm 1. It is possible to sample fibers only in some modes and not in all of them. Motivated by the matrix case where it is possible to only select columns and not rows, in [82] it is suggested to sample fibers only in some of the modes and not all of them. This is considered as a generalization of the CX matrix approximation to tensors.

B. LOW-RANK APPROXIMATIONS BASED ON MULTILINEAR PROJECTIONS

If one needs the Tucker approximation of a data tensor for several multilinear ranks, then formulation (7) needs passing the original data tensor $\underline{\mathbf{X}}$ multiple times. This burdens high communication costs. To resolve this problem, consider the following relation

$$\underline{\mathbf{W}} = \underline{\mathbf{X}} \times_1 \Phi_1 \times_2 \Phi_2 \cdots \times_N \Phi_N, \quad (8)$$

where $\Phi_n \in \mathbb{R}^{S_n \times I_n}$, $n = 1, 2, \ldots, N$ are random matrices preserving the structure of the data tensor $\underline{\mathbf{X}}$ and the tensor $\underline{\mathbf{W}} \in \mathbb{R}^{S_1 \times S_2 \times \cdots \times S_N}$ is the compressed or sketched tensor. For example, for non-negative data tensors, we can use uniform random matrices. The reduction dimension, S_n should be selected carefully to capture the range of the unfolding matrices. From the theoretical point of view, it is required to have $S_n > 2R_n$, for a detailed discussion on this, see [83], [84].

Algorithm 1: The Sampling Tucker (STucker) Algorithm [81]

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, positive integer numbers R_n, $n = 1, 2, \ldots, N$
Output: Tucker approximation $\underline{\mathbf{X}} \cong [\![\underline{\mathbf{S}}; \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]$

```
for n = 1, 2, ..., N do
    Sample R_n Mode-n Fibers and Generate
      Approximate Factor Matrix A_n ∈ R^{I_n × R_n}
end
Compute the Core Tensor S ∈ R^{R_1 × R_2 × ··· × R_N} as in (7)
```

Algorithm 2: The Sampling Single-Pass Tucker (SPSTucker) Algorithm [84]

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, positive integer numbers R_n, $n = 1, 2, \ldots, N$
Output: Tucker approximation $\underline{\mathbf{X}} \cong [\![\underline{\mathbf{S}}; \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]$

```
for n = 1, 2, ..., N do
    Sample R_n Mode-n Fibers in n-Th Mode and
      Generate Approximate Factor Matrix A_n ∈ R^{I_n × R_n}
end
Compute the Core Tensor S ∈ R^{R_1 × R_2 × ··· × R_N} as in (10)
```

The size of the tensor $\underline{\mathbf{W}}$ is smaller than the original data tensor $\underline{\mathbf{X}}$ and is easier to be handled. By substituting (2) in (8) we have

$$\underline{\mathbf{W}} \cong \underline{\mathbf{S}} \times_1 (\Phi_1 \mathbf{A}_1) \times_2 (\Phi_2 \mathbf{A}_2) \cdots \times_N (\Phi_N \mathbf{A}_N), \quad (9)$$

From (9), the core tensor of the Tucker decomposition can be computed by [83], [84]

$$\underline{\mathbf{S}} \cong \underline{\mathbf{W}} \times_1 (\Phi_1 \mathbf{A}_1)^+ \times_2 (\Phi_2 \mathbf{A}_2)^+ \cdots \times_N (\Phi_N \mathbf{A}_N)^+, \quad (10)$$

which is independent of the original data tensor $\underline{\mathbf{X}}$, i.e., it is a single-pass algorithm. This procedure is summarized in Algorithm 2. The most expensive part of Algorithm 2 is computing the sketching tensor $\underline{\mathbf{W}}$ in (9), which can be accelerated by exploiting the structured random matrices such as sparse random matrices [85], [86], subsampled random Fourier transform (or SRFT) [87], [88] subsampled Hadamard transforms, and sequence of Givens rotations [89].

Algorithm 2 combines sampling and multilinear random projection techniques for tensor decomposition since in the first step, fibers are selected based on some probability distribution (sampling step) while the compression step (8) is performed via multilinear projection technique (random projection step). It is also possible to perform the first step by using randomized QR decomposition [90], [91].

Another approach is based on multilinear projections, either random or deterministic, was proposed in [92], which generalizes the idea of Compressed Sensing (CS) [93], [94] for data tensors. This approach, is called *Multi-Linear Projection (MLProj) method*, and allows one to recover some

potentially big data tensor from a few multilinear projection measurements when the original data tensor has a low Tucker-rank $(R_1, R_2, \ldots, R_N)$ structure. More specifically, in this method, it is assumed that the following multilinear projection measurement are available:

$$\underline{\mathbf{Z}}^{(n)} = \underline{\mathbf{X}} \times_1 \Phi_1 \cdots \times_{n-1} \Phi_{n-1} \times_{n+1} \Phi_{n+1} \cdots \times_N \Phi_N \tag{11}$$

for $n = 1, 2, \ldots, N$, where $\Phi_n \in \mathbb{R}^{R_n \times I_n}$. Then, an approximation of the original tensor $\underline{\mathbf{X}}$ can be obtained by the following formula:

$$\underline{\mathbf{X}} \cong \underline{\mathbf{W}} \times_1 \mathbf{Z}_1 \mathbf{W}_{(1)}^{+} \cdots \times_N \mathbf{Z}_N \mathbf{W}_{(N)}^{+}, \tag{12}$$

where $\mathbf{Z}_n = (\underline{\mathbf{Z}}^{(n)})_{(n)}$ and $\underline{\mathbf{W}}$ was defined in (8). This low Tucker-rank reconstruction is proved to be numerically stable and robust [92]. Matrices Φ_n can be random (e.g., Gaussian distributed) or deterministic. For example, by choosing Φ_n as a subset of columns in the identity matrix, the obtained projections in eq. (11) are actual fibers in mode-n of the original data tensor. Another option is to construct matrices Φ_n composed by subsets of columns in the Fourier transform matrix. In this case, the obtained measurements are fibers in the multidimensional Fourier domain. Section IX, illustrates how to apply the MLProj method to reconstruct cardiac fMRI images by sub-sampling the Fourier domain, which allows a significant speedup in signal acquisition with negligible loss of information.

The first question regarding the sampling approaches is: what will be the approximation error by fiber sampling algorithms? On one side, the method based on multilinear projections [92] is equipped with error bounds for the 2D and 3D cases. On the other side, using the existing theory of randomized sampling techniques on unfolding matrices, an additive-error bound on the accuracy of the solution is proven in [81]. It is known that the relative error accuracy is also achievable if the leverage score probabilities are used [54]. This idea is used in [91]. The computation of leverage scores is expensive because of the requirement of computation of the SVD, but fast algorithms for their computations are proposed in [95]. The procedure of column selection can be performed using the leverage scores sampling or the DEIM algorithm. These techniques were studied in [91].

C. THE CROSS3D ALGORITHM

In [59], the Tucker decomposition is tackled by applying the Cross2D algorithms to unfolding matrices, and also efficient variants with linear complexity are developed. They use the theory of cross-approximation of matrices to tensors, and in particular, it is proven that if a data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ admits the following Tucker model

$$\underline{\mathbf{X}} = \underline{\mathbf{S}} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3 + \underline{\mathbf{E}}, \quad \left\|\underline{\mathbf{E}}\right\|_F = \varepsilon,$$

with low-rank (R_1, R_2, R_3) where $\underline{\mathbf{S}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$, and $\mathbf{A}_n \in \mathbb{R}^{I_n \times R_n}$, $n = 1, 2, 3$. Then it is possible to find a new Tucker approximation whose factor matrices $\widehat{\mathbf{A}}_n$, $n = 1, 2, 3$ are taken from some fibers of the original data tensor $\underline{\mathbf{X}}$ and satisfying

$$\underline{\mathbf{X}} = \widehat{\underline{\mathbf{S}}} \times_1 \widehat{\mathbf{A}}_1 \times_2 \widehat{\mathbf{A}}_2 \times_3 \widehat{\mathbf{A}}_3 + \widehat{\underline{\mathbf{E}}},$$

where

$$\left\|\widehat{\underline{\mathbf{E}}}\right\|_\infty \leq (R_1 R_2 R_3 + 2R_1 R_2 + 2R_1 + 1)\, \varepsilon. \tag{13}$$

The above result can be easily generalized to higher-order tensors. Later on, tighter error bounds compared to (13) were presented in [96]. Depending on the factor matrices $\mathbf{A}_i$, $i = 1, 2, 3$ and the core tensor $\underline{\mathbf{S}}$ used for the approximation, the error term $\left\|\underline{\mathbf{E}}\right\|_F$ (expressed as the Frobenius norm), is different. For example, in the case of matrices for which the HOSVD is reduced to the SVD, it is known that the truncated SVD, $\mathbf{X} \cong \mathbf{U}_R \Sigma_R \mathbf{V}_R^T$, provides the best rank R matrix approximation, where $\mathbf{U}_R$ and $\mathbf{V}_R$ are the top left and right singular vectors and Σ_R is a diagonal matrix containing the R largest singular values σ_r, $r = 1, 2, \ldots, R$. To be more precise, let $\mathbf{X} \in \mathbb{R}^{I \times J}$ be a given data matrix with rank $R < min(I, J)$, then we have $\|\mathbf{E}\|_F = \left(\sum_{i=R+1}^{\min(I,J)} \sigma_i^2\right)^{1/2}$ where σ_i, $i = R+1, R+2, \ldots, \min(I, J)$ are the last (smallest) $\min(I, J) - R$ nonzero singular values of the matrix $\mathbf{X}$. It is clear that the error term achieved by the truncated SVD provides a lower bound for the CMA (4) obtained by sampling columns and rows. However, in the case of tensors, the truncated HOSVD does not provide the best multilinear rank approximation in the least-squares sense while a quasi-best approximation can be achieved [6] as follows

$$\left\|\underline{\mathbf{E}}\right\|_F \leq \sqrt{N} \left\|\underline{\mathbf{X}} - \underline{\mathbf{X}}_{Best}\right\|_F,$$

where $\underline{\mathbf{X}}_{best}$ is the best multilinear rank approximation of the tensor $\underline{\mathbf{X}}$. The error term $\|\underline{\mathbf{E}}\|_F$ can be represented in terms of the singular values of the unfolding matrices; see [6], [97] for details.

The proposed algorithm in [59] for computing the Tucker approximation with error accuracy (13) is constructive and the idea is applying the Cross2D algorithm to unfolding matrices sequentially (to be discussed later) and this algorithm is called Cross3D. The main superiority of the Cross3D algorithm over fiber sampling technique [81] is that it can find a set of good fibers in a heuristic and efficient way. This is crucial, especially when the data tensor does not have exact Tucker rank, i.e., the data tensor is corrupted by noise and the selection of columns affects the approximation error. The cross approximation is able to handle this issue by heuristically looking for good columns and rows. In the sequel, we describe the Cross3D algorithm. For the first unfolding matrix $\mathbf{X}_{(1)}$, the following cross approximation can be computed by applying the Cross2D algorithm to the first unfolding matrix as

$$\mathbf{X}_{(1)} = \mathbf{CUR} + \mathbf{E}_1, \tag{14}$$

where $\mathbf{C} \in \mathbb{R}^{I_1 \times R}$, $\mathbf{U} \in \mathbb{R}^{R \times R}$, $\mathbf{R} \in \mathbb{R}^{R \times I_2 I_3}$ and the error term $\mathbf{E}_1 \in \mathbb{R}^{I_1 \times I_2 I_3}$. It is not difficult to see that each row of the matrix $\mathbf{R}$ is a vectorization of a horizontal slice of the

Algorithm 3: The Cross3D Algorithm [59]

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, a Tucker rank $(R_1, R_2, \ldots, R_N)$

Output: Tucker approximation $\underline{\mathbf{X}} \cong [\![\underline{\mathbf{U}}; \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]$

Set $\underline{\mathbf{R}} = \underline{\mathbf{X}}$
$\mathbf{z}_1 = [R_1, I_2, \ldots, I_N]$
for $n = 1, 2, \ldots, N$ **do**
  $[\mathbf{C}_n, \mathbf{U}_n, \mathbf{R}_n] = \text{Cross2D}(\mathbf{R}_{(n)}, R_n)$;
  $\underline{\mathbf{R}} = \text{fold\_}n(\mathbf{R}_n, \mathbf{z}_n)$
  $\underline{\mathbf{R}} = \underline{\mathbf{R}} \times_n (\mathbf{C}_n \mathbf{U}_n)$
  if $n < N$ **then**
    $\mathbf{z}_{n+1} = [R_1, \ldots, R_n, I_{n+1}, \ldots, I_N]$
  end
end
Set $\underline{\mathbf{U}} = \underline{\mathbf{R}}$
Set $\mathbf{A}_n = \mathbf{C}_n, \ n = 1, 2, \ldots, N$

original tensor $\underline{\mathbf{X}}$. Applying the n-folding operator on both sides of (14) and using the identity (1), we have

$$\underline{\mathbf{X}} = \underline{\mathbf{R}} \times_1 (\mathbf{CU}) + \underline{\mathbf{E}}_1,$$

where $\underline{\mathbf{R}} = \text{fold\_1}(\mathbf{R}, [R, I_2, I_3]) \in \mathbb{R}^{R \times I_2 \times I_3}$ and $\underline{\mathbf{E}}_1 = \text{fold\_1}(\mathbf{E}_1, [I_1, I_2, I_3]) \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. In the next step, a cross approximation of the unfolding matrix $\mathbf{R}_{(2)} \in \mathbb{R}^{I_2 \times RI_3}$ is computed as

$$\mathbf{R}_{(2)} = \widehat{\mathbf{C}}\widehat{\mathbf{U}}\widehat{\mathbf{R}} + \mathbf{E}_2,$$

where $\widehat{\mathbf{C}} \in \mathbb{R}^{I_2 \times R}$, $\widehat{\mathbf{U}} \in \mathbb{R}^{R \times R}$, $\widehat{\mathbf{R}} \in \mathbb{R}^{R \times RI_3}$ and the error term $\mathbf{E}_2 \in \mathbb{R}^{I_2 \times RI_3}$ or equivalently

$$\underline{\mathbf{R}} = \widehat{\underline{\mathbf{R}}} \times_2 (\widehat{\mathbf{C}}\widehat{\mathbf{U}}) + \underline{\mathbf{E}}_2, \tag{15}$$

where $\widehat{\underline{\mathbf{R}}} = \text{fold\_2}(\widehat{\mathbf{R}}, [R, R, I_3]) \in \mathbb{R}^{R \times R \times I_3}$ and $\underline{\mathbf{E}}_2 = \text{fold\_2}(\mathbf{E}_2, [R, I_2, I_3]) \in \mathbb{R}^{R \times I_2 \times I_3}$. Finally, for the last unfolding matrix $\widehat{\mathbf{R}}_{(3)} \in \mathbb{R}^{I_3 \times R^2}$, we have

$$\widehat{\mathbf{R}}_{(3)} = \widetilde{\mathbf{C}}\widetilde{\mathbf{U}}\widetilde{\mathbf{R}} + \mathbf{E}_3,$$

where $\widetilde{\mathbf{C}} \in \mathbb{R}^{I_3 \times R}$, $\widetilde{\mathbf{U}} \in \mathbb{R}^{R \times R}$, $\widetilde{\mathbf{R}} \in \mathbb{R}^{R \times R^2}$ and the error term $\mathbf{E}_3 \in \mathbb{R}^{R \times RI_3}$ or equivalently

$$\widehat{\underline{\mathbf{R}}} = \widetilde{\underline{\mathbf{R}}} \times_3 (\widetilde{\mathbf{C}}\widetilde{\mathbf{U}}) + \underline{\mathbf{E}}_3, \tag{16}$$

where $\widetilde{\underline{\mathbf{R}}} = \text{fold\_3}(\widetilde{\mathbf{R}}, [R, R, R]) \in \mathbb{R}^{R \times R \times R}$, and $\underline{\mathbf{E}}_3 = \text{fold\_3}(\mathbf{E}_3, [R, R, I_3]) \in \mathbb{R}^{R \times R \times I_3}$. Ignoring the error terms in (14)-(16), and combing all terms, we have

$$\underline{\mathbf{X}} \cong (\widetilde{\underline{\mathbf{R}}} \times_1 \mathbf{U} \times_2 \widehat{\mathbf{U}} \times_3 \widetilde{\mathbf{U}}) \times_1 \mathbf{C} \times_2 \widehat{\mathbf{C}} \times_3 \widetilde{\mathbf{C}}.$$

Next, the core tensor can be computed as

$$\underline{\mathbf{S}} = \widehat{\underline{\mathbf{R}}} \times_1 \mathbf{U} \times_2 \widehat{\mathbf{U}} \times_3 \widetilde{\mathbf{U}}. \tag{17}$$

This procedure is summarized in Algorithm 3.

In Algorithm 3, at each iteration, the size of unfolding matrices to which the Cross2D algorithm is applied is reduced. This is the same as the Sequentially Truncated HOSVD (ST-HOSVD) algorithm [98] except that instead of the SVD, the cross approximation is used. Please note that unlike (7) where we need to access the whole data tensor, the core tensor in the Cross3D is computed automatically in the final step via (17). The computational complexity of the Cross3D algorithm is $\mathcal{O}(I^2R^2)$ for $I_1 = I_2 = I_3 = I$ because the size of rows in the unfolding matrices are very large. However, each row represents a horizontal slice of the original tensor and it can be approximated by the cross approximation again. An efficient algorithm based on this with computational complexity $\mathcal{O}(IR^4)$ (for Tucker rank (R, R, R)) is developed in [59], where each long row is treated as a matrix and cross approximation of it is computed. A multilevel version of Algorithm 3 is developed in [99]. The CMA approaches can also be combined with other types of tensor decompositions. For example, the so-called TT-cross approximation developed in [60] is for computation of the TT decomposition, while the algorithms developed in [100], [101] are for the HT decomposition, which includes TT decomposition as a special case. The TR-ALS algorithm [13] and the CP-ALS algorithm [25] are known algorithms for computation of the TR [13] and the CP decompositions [2], respectively, where each iteration solves an over-determined least-squares problem. The idea of solving the underlying over-determined least-squares problem by sampling a part of rows of the coefficient matrix and solving the smaller least-squares problems, was proposed in [102] and [103], respectively.

D. FAST SAMPLING TUCKER DECOMPOSITION (FSTD) ALGORITHM

Now we introduce the third kind of CTA methods. For the simplicity of presentation, we first study the 3rd-order tensors and then outline generalization to higher-order tensors. Unlike the matrices where each column and row always intersect, columns, rows, and tubes for a 3-th order tensor may not intersect each other. As a result, an analogous intersection subtensor like what we have in the matrix case may not arise. The idea in [61] is to first produce an intersection subtensor $\underline{\mathbf{W}}$ by sampling some indices in different modes and then selecting some fibers. Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ be a given 3rd-order tensor and $\mathcal{I}_1 \subseteq I_1$, $\mathcal{I}_2 \subseteq I_2$, $\mathcal{I}_3 \subseteq I_3$, be subsets of indices I_1, I_2, I_3 where $|\mathcal{I}| = P_1$, $|\mathcal{I}_2| = P_2$, $|\mathcal{I}_3| = P_3$ and the core intersection subtensor is denoted by $\underline{\mathbf{W}} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$.

The question is, how to compute an approximate Tucker decomposition for the tensor $\underline{\mathbf{X}}$ based on the intersection subtensor $\underline{\mathbf{W}}$? Motivated by the fact that

$$\mathbf{U} = \mathbf{W} \times_1 \mathbf{W}_{(1)}^+ \times_2 \mathbf{W}_{(2)}^+ = \mathbf{W}^+\mathbf{W}\mathbf{W}^+ = \mathbf{W}^+, \tag{18}$$

which is used as the middle matrix in the CMA, it is suggested [61] to compute the approximate core tensor in the Tucker decomposition as

$$\begin{aligned} \underline{\mathbf{U}} &= \underline{\mathbf{W}} \times_1 \mathbf{W}_{(1)}^+ \times_2 \mathbf{W}_{(2)}^+ \times_3 \mathbf{W}_{(3)}^+ \\ &\equiv [\![\underline{\mathbf{W}}; \mathbf{W}_{(1)}^+, \mathbf{W}_{(2)}^+, \mathbf{W}_{(3)}^+]\!]. \end{aligned} \tag{19}$$

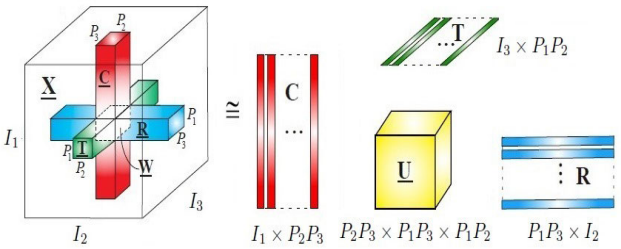

FIGURE 7. Illustration of the FSTD algorithm for a 3rd-order low-rank tensor. For simplicity of presentation, we assume that all fibers build up to block sub-tensors, [23].

Algorithm 4: Fast Sampling Tucker Decomposition (FSTD) Algorithm for 3rd-Order Tensors [61]

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, indices $\mathcal{I}_n \subseteq [I_n],\ n = 1, 2, 3$

Output: Tucker approximation of the tensor $\underline{\mathbf{X}}$

1 Generate the Intersection Subtensor $\underline{\mathbf{W}} = \underline{\mathbf{U}}\,(\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3)$

2 Generate the Subsampled Matrices $\mathbf{A}_1 = \mathbf{X}_{(1)}(:, \mathcal{I}_2, \mathcal{I}_3),\ \mathbf{A}_2 = \mathbf{X}_{(2)}(\mathcal{I}_1, :, \mathcal{I}_3)$ and $\mathbf{A}_3 = \mathbf{X}_{(3)}(\mathcal{I}_1, \mathcal{I}_2, :)$

3 $\underline{\mathbf{X}} \cong \left[\!\left[\underline{\mathbf{W}}, \mathbf{A}_1\mathbf{W}_{(1)}^{+}, \mathbf{A}_2\mathbf{W}_{(2)}^{+}, \mathbf{A}_3\mathbf{W}_{(3)}^{+} \right]\!\right]$

This is a direct generalization of (18) to 3rd-order tensors. In view of (19), the core tensor $\underline{\mathbf{U}}$ of the Tucker approximation is of size $P_2P_3 \times P_1P_3 \times P_1P_2$, and as a result, we need to sample P_2P_3 columns, P_1P_3 rows, and P_1P_2 tubes, see Figure 7. They should be selected in an appropriate way. It is shown in [61] that the corresponding factor matrices $\mathbf{A}_1 \in \mathbb{R}^{I_1 \times P_2P_3}$, $\mathbf{A}_2 \in \mathbb{R}^{I_2 \times P_1P_3}$, $\mathbf{A}_3 \in \mathbb{R}^{I_3 \times P_1P_2}$ are the subsampled matrices from the unfolding matrices $\mathbf{X}_{(1)}(:, \mathcal{I}_2, \mathcal{I}_3)$, $\mathbf{X}_{(2)}(\mathcal{I}_1, :, \mathcal{I}_3)$ and $\mathbf{X}_{(3)}(\mathcal{I}_1, \mathcal{I}_2, :)$ respectively and CTA can be found as

$$\begin{aligned} \underline{\mathbf{X}} &\cong [\![\underline{\mathbf{U}}; \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3]\!] \\ &\equiv \left[\!\!\left[\underline{\mathbf{W}}; \underbrace{\mathbf{A}_1\mathbf{W}_{(1)}^{+}}_{\widetilde{\mathbf{C}}_1}, \underbrace{\mathbf{A}_2\mathbf{W}_{(2)}^{+}}_{\widetilde{\mathbf{C}}_2}, \underbrace{\mathbf{A}_3\mathbf{W}_{(3)}^{+}}_{\widetilde{\mathbf{C}}_3} \right]\!\!\right]. \end{aligned} \tag{20}$$

The procedure of this approach is summarized as follows

- Consider indices $\mathcal{I}_n \in [I_n],\ n = 1, 2, 3$ and produce the intersection subtensor $\underline{\mathbf{W}}$ and corresponding sampled columns, rows, and tubes $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_3$.
- Compute the Tucker approximation (20).

We refer to this algorithm as Fast Sampling Tucker Decomposition (FSTD) and summarize it in Algorithm 4 [61]. It is interesting to note that the FSTD is obtained as a particular case of the MLProj method, eq. (12), when projection matrices Φ_n are built upon subsets of columns of the identity matrix. A similar approach is developed in [104] in the sense of the number of selected fibers in each mode.

This approach can be straightforwardly generalized to higher-order tensors. The main difference between formulas (7) and (20) is that components of factor matrices $\mathbf{A}^{(n)}$ in (7) are taken from the original data tensor $\underline{\mathbf{X}}$, while this is the case for the core tensor $\underline{\mathbf{W}}$ in (20). A drawback of this approach is that the number of sampling fibers i.e., $\mathbf{A}_n,\ n = 1, 2, 3$ required in each mode is relatively high, and the matrix-matrix multiplications in (20) may be expensive. As a matter of fact, the CTA for N-th order tensors needs P^{N-1} fibers in each mode which still increases exponentially with the tensor order. To resolve this issue, an adaptive algorithm was suggested in [61], which is a generalization of the Cross2D algorithm [50], [51] to tensors in the sense of just selecting maximum absolute values of fibers while in the Cross3D algorithm both fibers and slices are involved. This procedure for a 3rd order tensor is described in Algorithm 5.

Note that Algorithm 5 is not a randomized algorithm because, except Line 1, all computations are performed deterministically. In contrary to other algorithms, the indices are not given to the algorithm in advance, and they are selected adaptively. To be more precise, let $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ be a given data tensor and $\mathcal{I}_p = \left[i_1, i_2, \ldots, i_p\right] \in [I],\ \mathcal{J}_q = \left[j_1, j_2, \ldots, j_q\right] \in [J], \mathcal{K}_r = [k_1, k_2, \ldots, k_r] \in [K]$ be the indices which have been already selected. Based on the already selected indices $(\mathcal{I}_p, \mathcal{J}_q, \mathcal{K}_r)$, let us choose a new index to be added, for example, to the set $\mathcal{I}_p$. To this end, first the (p, q, r)-approximation is computed using the FSTD as follows

$$\widehat{\underline{\mathbf{X}}}^{(p,q,r)} = \left[\!\left[\underline{\mathbf{U}}^{(p,q,r)}; \mathbf{C}_1^{(q,r)}, \mathbf{C}_2^{(p,r)}, \mathbf{C}_3^{(p,q)} \right]\!\right],$$

where the core tensor $\underline{\mathbf{U}}^{(p,q,r)}$ is computed from the intersection subtensor $\underline{\mathbf{W}}^{(p,q,r)} = \underline{\mathbf{X}}(\mathcal{I}_p, \mathcal{J}_q, \mathcal{K}_r)$ using (19) and $\mathbf{C}_1^{(q,r)} = \mathbf{X}_{(1)}(:, \mathcal{J}_q, \mathcal{K}_r)$, $\mathbf{C}_2^{(p,r)} = \mathbf{X}_{(2)}(\mathcal{I}_p, :, \mathcal{K}_r)$, $\mathbf{C}_3^{(p,q)} = \mathbf{X}_{(3)}(\mathcal{I}_p, \mathcal{J}_q, :)$. Then the residual is defined as

$$\underline{\mathbf{E}}^{(p,q,r)} = \underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}^{(p,q,r)}, \tag{21}$$

and the index of the residual fiber $\underline{\mathbf{E}}^{(p,q,r)}(:, j_q, k_r)$, with maximum absolute value, e.g. i_{p+1}, is selected and added to the index set, i.e., $\mathcal{I}_{p+1} = \mathcal{I}_p \cup [i_{p+1}]$. Similarly new indices can be added to the index sets $\mathcal{J}_q, \mathcal{K}_r$ for which the residual fibers $\underline{\mathbf{E}}^{(p,q,r)}(i_p, :, k_r)$ and $\underline{\mathbf{E}}^{(p,q,r)}(i_p, j_q, :)$ should be considered, respectively. Note that the whole residual $\underline{\mathbf{E}}^{(p,q,r)}$ is not necessary to be computed and only a fiber to which a maximum component is required is calculated; see [61] for details.

Algorithm 5 is fast for tensors of very low Tucker ranks and its complexity increased exponentially if the Tucker rank is large, see Table 2. It is shown in [61] that the linear complexity is achievable and the Tucker approximation of an Nth-order tensor of size $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with an exact Tucker rank $(R, R, \ldots, R)$ can be computed just by taking R mode-n fibers in each mode for $n = 1, 2, \ldots, N$. This result is based on a modification of the (20). In this modified version, the indices $\mathcal{I}_n \subset [I_n],\ n = 1, 2, \ldots, N$ under which the intersection core tensor $\underline{\mathbf{W}}$ is constructed are selected in a special way and not randomly. More precisely, in the first step, from each of the n-unfolding matrices $(n = 1, 2, \ldots, N)$, R fibers

Algorithm 5: Adaptive Fiber Selection (AFS) Algorithm [61]

Input : Initial column fiber selection (j_1, k_1) and the number of fibers to be selected P

Output: Indices of selected P rows/columns/tubes $\mathcal{I}_P$, $\mathcal{J}_P$ and $\mathcal{K}_P$

$\mathcal{I}_0 = [\,]$, $\mathcal{J}_1 = [j_1]$, $\mathcal{K}_1 = [k_1]$
Choose i_1 as the index of Max absolute value in the column fiber
$\mathcal{I}_1 = \mathcal{I}_0 \cup [i_1]$
$p = 2$
while $p < P$ **do**
  if $p = 2$ **then**
    Choose j_p as the index of Max absolute value in $\underline{\mathbf{Y}}\left(i_{p-1}, :, k_{p-1}\right)$;
  else
    Choose j_p as the index of Max absolute value in $\underline{\mathbf{E}}^{(p-1,p-1,p-1)}\left(i_{p-1}, :, k_{p-1}\right)$;
  end
  $\mathcal{J}_p = \mathcal{J}_{p-1} \cup \left[j_p\right]$
  Choose k_p as the index of Max absolute value in $\underline{\mathbf{E}}^{(p-1,p,p-1)}\left(i_{p-1}, j_p, :\right)$
  $\mathcal{K}_p = \mathcal{K}_{p-1} \cup \left[j_p\right]$
  Choose i_p as the index of Max absolute value in $\underline{\mathbf{E}}^{(p-1,p,p)}\left(:, j_p, k_p\right)$
  $\mathcal{I}_p = \mathcal{I}_{p-1} \cup \left[i_p\right]$
  $p = p + 1$
end

in mode n (mode-n fibers) are selected and they are stored as matrices $\mathbf{C}_n \in \mathbb{R}^{I_n \times R}$. As $R \leq I_n$, and in the next step, a subset of indices $\mathcal{I}_n$, $n = 1, 2, \ldots, N$, $(\mathcal{I}_n \subset [I_n])$, is selected for which the intersection submatrix $\mathbf{W}_n = \mathbf{C}_n(\mathcal{I}_n, :) \in \mathbb{R}^{R \times R}$, is nonsigular. Then it is shown that the following exact approximation can be obtained

$$\underline{\mathbf{X}} = \left[\!\left[\underline{\mathbf{U}}; \mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_N\right]\!\right],$$

where

$$\underline{\mathbf{U}} = \left[\!\left[\underline{\mathbf{W}}; \mathbf{W}_1^{-1}, \mathbf{W}_2^{-1}, \ldots, \mathbf{W}_N^{-1}\right]\!\right], \tag{22}$$

and $\underline{\mathbf{W}} = \underline{\mathbf{X}}(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_N) \in \mathbb{R}^{R \times R \times R}$ is the intersection subtensor. Note that formulation (22) differs from the formulation (19) because in the former, the unfolding matrices $\mathbf{W}_{(n)}$, $n = 1, 2, 3$ are used, while in the latter, the submatrices $\mathbf{W}_n = \mathbf{C}_n(\mathcal{I}_n, :)$, $n = 1, 2, \ldots, N$. The fibers can be sampled using the pivoted QR decomposition applied to the unfolding matrices $\mathbf{X}_{(n)}$. More precisely, inspired by the matrix case [90], *Higher-Order Interpolatory Decomposition (HOID)* is proposed in [91], and this will be discussed in detail in the next Section.

E. RANDOMIZED HIGHER-ORDER INTERPOLATORY DECOMPOSITION (HOID)

The fibers can be sampled using the pivoted QR decomposition applied to the unfolding matrices $\mathbf{X}_{(n)}$. The *Higher-Order Interpolatory Decomposition (HOID)* is developed in [91] and is a generalization of the DEIM algorithm [90]. Given an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the pivoted QR factorization is applied to the unfolding matrix $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times J}$, $(J = \prod_{i \neq n} I_i)$ as

$$\mathbf{X}_{(n)}\Pi = \mathbf{Q}\mathbf{R}, \tag{23}$$

where $\Pi \in \mathbb{R}^{J \times J}$ is a permutation matrix, and $\mathbf{Q} \in \mathbb{R}^{I_n \times I_n}$ is an orthogonal matrix. The pivoted QR decomposition can be computed using the strong rank-revealing QR (RRQR) algorithm [105]. The permutation matrix Π and the corresponding orthogonal matrix $\mathbf{Q}$ are partitioned as follows

$$\Pi = [\Pi_1 \ \Pi_2], \quad \mathbf{Q} = [\mathbf{Q}_1 \ \mathbf{Q}_2], \tag{24}$$

where $\Pi_1 \in \mathbb{R}^{J \times K}$, $\Pi_2 \in \mathbb{R}^{J \times (J-K)}$, $\mathbf{Q}_1 \in \mathbb{R}^{I_n \times K}$, $\mathbf{Q}_2 \in \mathbb{R}^{I_n \times (I_n - K)}$. Here Equation (23) can be rewritten as

$$\mathbf{X}_{(n)}[\Pi_1 \ \Pi_2] = [\mathbf{Q}_1 \ \mathbf{Q}_2]\begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix}, \tag{25}$$

where $\mathbf{R}_{11} \in \mathbb{R}^{K \times K}$, $\mathbf{R}_{12} \in \mathbb{R}^{K \times (I_n - K)}$, $\mathbf{R}_{22} \in \mathbb{R}^{(I_n - K) \times (J - K)}$ and $\mathbf{0} \in \mathbb{R}^{(I_n - K) \times K}$ is a null matrix. From (25), by straightforward computations, we have

$$\begin{aligned} \mathbf{Q}^{(n)} &\equiv \mathbf{X}_{(n)}\Pi_1 = \mathbf{Q}_1\mathbf{R}_{11}, \\ \mathbf{X}_{(n)}\Pi_2 &= \mathbf{Q}_1\mathbf{R}_{12} + \mathbf{Q}_2\mathbf{R}_{22} \cong \mathbf{Q}_1\mathbf{R}_{12}, \end{aligned}$$

if $\|\mathbf{R}_{22}\|_2$ is small enough. It can be seen that

$$\mathbf{X}_{(n)} \cong [\mathbf{Q}_1\mathbf{R}_{11}, \ \mathbf{Q}_1\mathbf{R}_{12}]\,\Pi^T, \tag{26}$$

and substituting $\mathbf{Q}_1 = \mathbf{Q}^{(n)}\mathbf{R}_{11}^{-1}$ in (26), we have

$$\mathbf{X}_{(n)} \cong \mathbf{Q}^{(n)}\mathbf{F}^T, \quad \mathbf{F}^T = \left[\mathbf{I} \ \ \mathbf{R}_{11}^{-1}\mathbf{R}_{12}\right]\Pi^T,$$

which is a low-rank approximation for the matrix $\mathbf{X}_{(n)}$. The matrix $\mathbf{Q}^{(n)}$ is of full-rank because it is a multiplication of nonsingular and orthogonal matrices and can be used as an approximation for the basis of the range of $\mathbf{X}_{(n)}$. Let denote the indices of the columns of the matrix $\mathbf{X}_{(n)}$ which are selected based on the permutation matrix[4] Π_1 as $\mathbf{p}$. Then, we have $\mathbf{Q}^{(n)} = \mathbf{X}_{(n)}(:, \mathbf{p})$ and as a result, it may not be necessarily orthonormal. The HOID algorithm applies the earlier procedure to all unfolding matrices $\mathbf{X}_{(n)}$ and computes the basis matrices $\mathbf{Q}^{(n)}$. Afterward, the core tensor $\underline{\mathbf{S}}$ can be computed as follows

$$\underline{\mathbf{S}} = \underline{\mathbf{X}} \times_1 \mathbf{Q}^{(1)+} \times_2 \mathbf{Q}^{(2)+} \cdots \times_N \mathbf{Q}^{(N)+}.$$

Algorithm 6 summarizes a deterministic algorithm that computes the QR decomposition of unfolding matrices. A

[4]In MATLAB, the command $[\mathbf{Q}, \mathbf{R}, \mathbf{p}] = qr(\mathbf{X}, 'vector')$ provides the permutation of the indices of the columns of the matrix $\mathbf{X}$.

Algorithm 6: HOID Algorithm [91]

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and a multilinear rank $(R_1, R_2, \ldots, R_N)$

Output: Tucker approximation $\underline{\mathbf{X}} \cong [\![\underline{\mathbf{S}}; \mathbf{Q}^{(1)}, \mathbf{Q}^{(2)}, \ldots, \mathbf{Q}^{(N)}]\!]$

1 **for** $n = 1, 2, \ldots, N$ **do**

2 Compute an Interpolatory Decomposition of Unfolding Matrix $\mathbf{X}_{(n)} \cong \mathbf{Q}^{(n)}\mathbf{F}_n^T$ {where $\mathbf{Q}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ are columns of $\mathbf{X}_{(n)}$}

3 **end**

4 Compute Core Tensor $\underline{\mathbf{S}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ as

$$\underline{\mathbf{S}} \equiv \underline{\mathbf{X}} \times_1 \mathbf{Q}^{(1)+} \times_2 \mathbf{Q}^{(2)+} \cdots \times_N \mathbf{Q}^{(N)+}$$

randomized variant of this algorithm replaces randomized QR decomposition instead of the QR decomposition. To this end, we should first make a reduction in the first mode of the unfolding matrices using random projection as $\mathbf{Y} = \Omega\mathbf{X}_{(n)}$ where $\Omega \in \mathbb{R}^{R+P \times I_n}$ (P is the oversampling parameter) is a random matrix after which the procedure described above is applied to the matrix $\mathbf{Y}$ for column the selection. It is shown in [106] that the selected column indices of the matrix $\mathbf{Y}$ can be used for the original data matrix $\mathbf{X}$. More precisely, if $\mathbf{Y} \cong \mathbf{Y}(:, \mathbf{p})\mathbf{F}^T$ then the indices $\mathbf{p}$ of the selected columns can be used for the matrix $\mathbf{X}_{(n)}$, i.e., $\mathbf{X}_{(n)} \cong \mathbf{X}_{(n)}(:, \mathbf{p})\mathbf{F}^T$.

In Algorithm 6, firstly, the interpolatory decompositions of all unfolding matrices $\underline{\mathbf{X}}_{(n)}$ are computed, and then the core tensor is constructed. It is possible to accelerate this algorithm using the idea of a Sequentially truncated HOSVD algorithm [98] where at each iteration, the size of the unfolding matrices is reduced. The accelerated version of Algorithm 6 based on the mentioned idea is developed in [91].

F. CTA BASED ON SLICE-FIBER SELECTION

Motivated by some applications in hyperspectral medical image analysis and consumer recommender system analysis where one of the modes is qualitatively different from others, an alternative CTA is proposed in [58]. Some examples of such datasets which have a ''qualitatively different'' or ''distinguished'' mode, are time-evolving internet graph or a set of hyperspectrally-resolved biopsy images or user product- product preference data for consumers [58]. The distinguished modes in the mentioned data sets are temporal evolution of the graph, the frequency or spectral variation in the images, and the users, respectively.

Here, the procedure is based on slice-tube selection, see Figure 8 for an illustrative explanation. We first briefly describe this idea for 3rd-order tensors. Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ be a given data tensor, and without loss of generality, we assume that the last mode is qualitatively different from the others.

Given prior probability distributions for sampling frontal slices as $\{p_i\}_{i=1}^{I_3}$ and tubes as $\{q_j\}_{j=1}^{I_1 I_2}$, in the first step, some frontal slices, say L_1, are sampled, and they are stored in $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times I_2 \times L_1}$. In the second step, we sample some tubes,

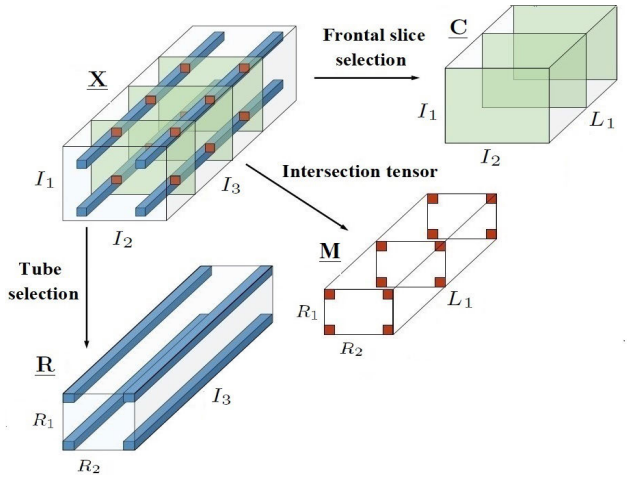

FIGURE 8. Procedure of frontal slices and tubes selections.

say $L_2 = R_1 R_2$ and store them in $\underline{\mathbf{R}} \in \mathbb{R}^{R_1 \times R_2 \times I_3}$, or a matrix $\mathbf{R} \in \mathbb{R}^{L_2 \times I_3}$, (see Figure 9 (a)). The CTA is then defined as (see Figure 9 (b))

$$\underline{\mathbf{X}} \cong \underline{\mathbf{C}} \times_3 (\mathbf{UR})^T, \tag{27}$$

where the tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times I_2 \times L_1}$ and the matrix $\mathbf{R} \in \mathbb{R}^{L_2 \times I_3}$ contain the sampled frontal slices and tubes respectively. The matrix $\mathbf{U} \in \mathbb{R}^{L_1 \times L_2}$ is defined as

$$\mathbf{U} = \mathbf{D}_1(\mathbf{D}_2\mathbf{M}\mathbf{D}_1)^+\mathbf{D}_2 \in \mathbb{R}^{L_1 \times L_2},$$
$$\mathbf{M} = reshape(\underline{\mathbf{M}}, [L_2, L_1]),$$

where $\mathbf{D}_1 \in \mathbb{R}^{L_1 \times L_1}$ and $\mathbf{D}_2 \in \mathbb{R}^{L_2 \times L_2}$ are scaling diagonal matrices corresponding to the slice and fiber sampling respectively and defined as follows

$$(\mathbf{D}_1)_{tt} = \frac{1}{\sqrt{L_1 p_{i_t}}}, \quad t = 1, 2, \ldots, L_1,$$
$$(\mathbf{D}_2)_{tt} = \frac{1}{\sqrt{L_2 q_{i_t}}}, \quad t = 1, 2, \ldots, L_2,$$

where $\{p_i\}_{i=1}^{I_3}$ are $\{q_j\}_{j=1}^{I_1 I_2}$ are probability distributions under which the frontal slices and fibers are sampled. This procedure is summarized in Algorithm 7. The length-squared probability distributions defined as follows

$$p_i = \frac{\|\underline{\mathbf{X}}(:, :, i_3)\|_F^2}{\|\underline{\mathbf{X}}\|_F^2}, \quad i_3 = 1, 2, \ldots, I_3,$$
$$q_j = \frac{\underline{\mathbf{X}}(j_1, j_2, :)}{\|\underline{\mathbf{X}}\|_F^2}, \quad j_1, j_2 \in J_1, J_2 \tag{28}$$

where J_1 and J_2 are subsets of the indices I_1 and I_2, are used in [58] for selecting the slices/tubes.

Remark 2: As discussed in [63], the model (27) can be considered as a special case of (20) when $\underline{\mathbf{W}} = \underline{\mathbf{C}}$ and $\mathbf{A}_i = \mathbf{W}_{(i)}$, $i = 1, 2$. Here, $\underline{\mathbf{W}}$ is the intersection subtensor in model (20), and $\underline{\mathbf{C}}$ is a tensor containing the selected frontal slices.

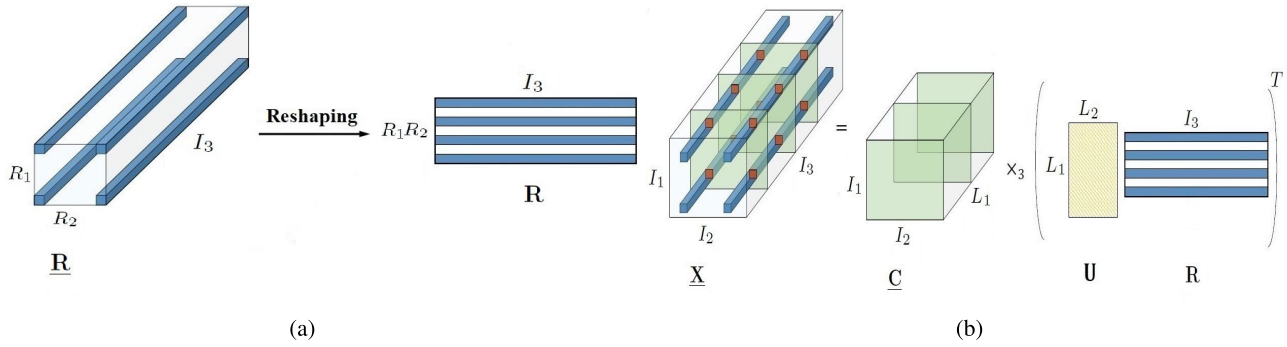

FIGURE 9. (a) Matricization of the selected tubes, (b) Illustration of the CTA based on frontal slice and tube selection.

A generalized version of Algorithm 7 to higher-order tensors is proposed in [58]. Here, the notion of the slab is used which means that all modes of a given tensor are free except one and this definition for 3rd-order tensors reduces to the slices. Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be a given data tensor whose n-th mode is the corresponding qualitatively different mode. We first sample L_1 slabs in the mode n from the original data tensor $\underline{\mathbf{X}}$ denoted by the tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times L_1 \times I_{n+1} \times \cdots \times I_N}$. In the next step, we sample L_2 fibers in mode n from the original data tensor $\underline{\mathbf{X}}$ and also L_2 fibers from the sampled tensor $\underline{\mathbf{C}}$ denoted by $\mathbf{R} \in \mathbb{R}^{L_2 \times I_n}$ and $\Psi \in \mathbb{R}^{L_2 \times L_1}$ respectively. Similar to the 3rd-order tensors, the diagonal scaling matrices $\mathbf{D}_1 \in \mathbb{R}^{L_1 \times L_1}$ and $\mathbf{D}_2 \in \mathbb{R}^{L_2 \times L_2}$ are defined as follows

$$(\mathbf{D}_1)_{tt} = \frac{1}{\sqrt{L_1 p_{i_t}}}, \quad t = 1, 2, \ldots, L_1,$$
$$(\mathbf{D}_2)_{tt} = \frac{1}{\sqrt{L_2 q_{i_t}}}, \quad t = 1, 2, \ldots, L_2,$$

where $\{p_i\}_{i=1}^{I_n}$ are $\{q_j\}_{j=1}^{\prod_{k \neq n}^{k} I_k}$ are probability distributions under which the slabs and fibers are sampled. Then the CTA is defined as follows

$$\underline{\mathbf{X}} \cong \underline{\mathbf{C}} \times_n (\mathbf{UR})^T,$$

where

$$\mathbf{U} = \Phi(\mathbf{D}_2 \Psi)^T \in \mathbb{R}^{L_1 \times L_2},$$

and $\Phi \in \mathbb{R}^{L_1 \times L_1}$ is the best rank-L_1 approximation of the MP of the matrix $\mathbf{H}_{(n)}\mathbf{H}_{(n)}^T$ where

$$\underline{\mathbf{H}} = \underline{\mathbf{C}} \times_n \mathbf{D}_1.$$

For detailed theoretical theorems and results, we refer to [58].

IV. CTA BASED ON TUBAL PRODUCT (T-productT)

The CMA and matrix column selection can be generalized to tensors based on the concept of t-product. We first introduce preliminary concepts and operations related to the t-product and then explain how to define the cross approximation using the t-product.

Algorithm 7: CTA Based on Fiber-Slice Selection (CTA-FS) for 3rd-Order Tensors [58]

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, a probability distribution $\{p_i\}_{i=1}^{I_3}$, a probability distribution $\{q_j\}_{j=1}^{I_1 I_2}$ and positive integers L_1 and L_2

Output: A tensor $\underline{\mathbf{C}}$ of size $I_1 \times I_2 \times L_1$, a matrix $\mathbf{U}$ of size $L_1 \times L_2$ and a matrix $\mathbf{R}$ of size $L_2 \times I_3$

1 Select L_1 Frontal Slices of Tensor $\underline{\mathbf{X}}$ i.i.d. Trials According to $\{p_i\}_{i=1}^{I_3}$ and Produce Tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times I_2 \times L_1}$;

2 Generate Diagonal Scaling Matrix D_1 of Size $L_1 \times L_1$ Where $(\mathbf{D}_1)_{tt} = \frac{1}{\sqrt{L_1 p_{i_t}}}$ for $t = 1, 2, \ldots, L_1$

3 Select L_2 Tubes of Tensor $\underline{\mathbf{X}}$ in L_2 i.i.d. Trials According to $\{q_j\}_{j=1}^{I_1 I_2}$ and Produce Unfolding Matrix $\mathbf{R} \in \mathbb{R}^{L_2 \times I_3}$

4 Generate Diagonal Scaling Matrix of Size $L_2 \times L_2$ Where $(\mathbf{D}_2)_{tt} = \frac{1}{\sqrt{l_2 q_{j_t}}}$ for $t = 1, 2, \ldots, L_2$

5 Compute the 3rd-Order Tensor Intersecting the Sampled Tubes and Frontal Slices as $\underline{\mathbf{W}} \in \mathbb{R}^{R_1 \times R_2 \times L_2}$, $R_1 R_2 = L_1$

6 Generate Matrix $\mathbf{W} = \text{reshape}(\underline{\mathbf{W}}, L_2, L_1)$

7 Define Matrix $\mathbf{U} = \mathbf{D}_1(\mathbf{D}_2\mathbf{W}\mathbf{D}_1)^{+}\mathbf{D}_2$

A. TUBAL SVD DECOMPOSITION

Tubal SVD (t-SVD) is a special kind of tensor decomposition representing a 3rd-order tensor as a product of three 3rd-order tensors where the middle tensor has nonzero tubes located only in the main diagonal [15]–[17], [64], see Figure 10. The t-SVD has found many applications in deep learning [107], [108], tensor completion [109], [110], numerical analysis [111]–[114], image reconstruction [115]. The generalization of the t-SVD to higher-order tensors is given in [116]. Throughout this paper for t-SVD, we only focus on 3rd-order tensors.

The number of nonzero tubes is called tubal rank. The truncated t-SVD gives the best approximation in the least-squares sense for any unitary invariant tensor norm, unlike other tensor decompositions. In order to introduce the t-SVD, we first need to present some basic definitions, such as the t-product operation and f-diagonal tensors.

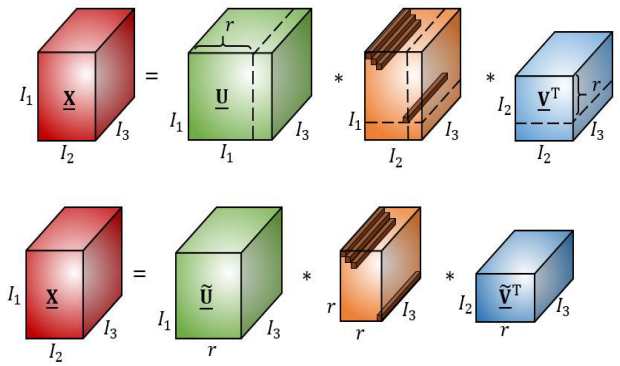

FIGURE 10. Illustration of (a) Tubal SVD (t-SVD) and (b) truncated t-SVD for a 3rd-order tensor.

Definition 2 (t-Product): Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{I_2 \times I_4 \times I_3}$, the t-product $\underline{\mathbf{X}} * \underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_4 \times I_3}$ is defined as follows

$$\underline{\mathbf{C}} = \underline{\mathbf{X}} * \underline{\mathbf{Y}} = fold\left(circ\left(\underline{\mathbf{X}}\right) unfold\left(\underline{\mathbf{Y}}\right)\right), \tag{29}$$

where

$$circ\left(\underline{\mathbf{X}}\right) = \begin{bmatrix} \underline{\mathbf{X}}(:,:,1) & \underline{\mathbf{X}}(:,:,I_3) & \cdots & \underline{\mathbf{X}}(:,:,2) \\ \underline{\mathbf{X}}(:,:,2) & \underline{\mathbf{X}}(:,:,1) & \cdots & \underline{\mathbf{X}}(:,:,3) \\ \vdots & \vdots & \ddots & \vdots \\ \underline{\mathbf{X}}(:,:,I_3) & \underline{\mathbf{X}}(:,:,I_3-1) & \cdots & \underline{\mathbf{X}}(:,:,1) \end{bmatrix},$$

and

$$unfold(\underline{\mathbf{Y}}) = \begin{bmatrix} \underline{\mathbf{Y}}(:,:,1) \\ \underline{\mathbf{Y}}(:,:,2) \\ \vdots \\ \underline{\mathbf{Y}}(:,:,I_3) \end{bmatrix}, \quad \underline{\mathbf{Y}} = fold\left(unfold\left(\underline{\mathbf{Y}}\right)\right).$$

In view of (29), it is seen that the t-product operation is the circular convolution operator, and because of this, it can be easily computed through Fast Fourier Transform (FFT) transform. More precisely, we first transform all tubes of two tensors $\underline{\mathbf{X}}$, $\underline{\mathbf{Y}}$, into the frequency domain, and construct two new tensors $\widehat{\underline{\mathbf{X}}}$ and $\widehat{\underline{\mathbf{Y}}}$ which are called spectral tensors. Then, the frontal slice of the spectral tensors $\widehat{\underline{\mathbf{X}}}$ and $\widehat{\underline{\mathbf{Y}}}$ are multiplied. Finally, we apply the Inverse FFT (IFFT) transform to all tubes of the last tensor. This procedure is summarized in Algorithm 8. Note that other types of tensor decompositions in the t-product format can be computed in a similar manner. For example, to compute the tubal QR computation of a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, i.e., $\underline{\mathbf{X}} = \underline{\mathbf{Q}} * \underline{\mathbf{R}}$, we first compute the FFT of the tensor $\underline{\mathbf{X}}$ as

$$\widehat{\underline{\mathbf{X}}} = fft(\underline{\mathbf{X}}, [], 3), \tag{30}$$

and then the QR decomposition of all frontal slices of the tensor $\widehat{\underline{\mathbf{X}}}$ is computed as follows

$$\widehat{\underline{\mathbf{X}}}(:,:,i) = \widehat{\underline{\mathbf{Q}}}(:,:,i)\, \widehat{\underline{\mathbf{R}}}(:,:,i).$$

In [117], the unitary transform matrices are used instead of discrete Fourier transform, and it is shown that it can provide

Algorithm 8: t-Product in the Fourier Domain [15]

Input : Two data tensors $\underline{\mathbf{Z}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\underline{\mathbf{X}} \in \mathbb{R}^{I_2 \times I_4 \times I_3}$
Output: t-product $\underline{\mathbf{C}} = \underline{\mathbf{Z}} * \underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_4 \times I_3}$

$\widehat{\underline{\mathbf{Z}}} = fft\left(\underline{\mathbf{Z}}, [], 3\right)$
$\widehat{\underline{\mathbf{X}}} = fft\left(\underline{\mathbf{X}}, [], 3\right)$
for $i = 1, 2, \ldots, I_3$ **do**
  $\widehat{\underline{\mathbf{C}}}(:,:,i) = \widehat{\underline{\mathbf{Z}}}(:,:,i)\, \widehat{\underline{\mathbf{X}}}(:,:,i)$
end
$\underline{\mathbf{C}} = ifft\left(\widehat{\underline{\mathbf{C}}}, [], 3\right)$

Algorithm 9: Truncated t-SVD [16]

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and target tubal rank R
Output: $\underline{\mathbf{U}}_R \in \mathbb{R}^{I_1 \times R \times I_3}$, $\underline{\mathbf{S}}_R \in \mathbb{R}^{R \times R \times I_3}$, $\underline{\mathbf{V}}_R \in \mathbb{R}^{I_2 \times R \times I_3}$

$\widehat{\underline{\mathbf{X}}} = fft\left(\underline{\mathbf{X}}, [], 3\right)$
for $i = 1, 2, \ldots, I_3$ **do**
  $[\mathbf{U}, \mathbf{S}, \mathbf{V}] = truncated\_svd\left(\widehat{\underline{\mathbf{X}}}(:,:,i), R\right)$
  $\widehat{\underline{\mathbf{U}}}(:,:,i) = \mathbf{U}$
  $\widehat{\underline{\mathbf{S}}}(:,:,i) = \mathbf{S}$
  $\widehat{\underline{\mathbf{V}}}(:,:,i) = \mathbf{V}$
end
$\underline{\mathbf{U}} = ifft\left(\widehat{\underline{\mathbf{U}}}, [], 3\right)$, $\underline{\mathbf{S}} = ifft\left(\widehat{\underline{\mathbf{S}}}, [], 3\right)$, $\underline{\mathbf{V}} = ifft\left(\widehat{\underline{\mathbf{V}}}, [], 3\right)$

decomposition with a lower tubal rank. Note that (30) is equivalent to computing the FFT of all tubes of the tensor $\underline{\mathbf{X}}$. Finally, the IFFT operator is applied to the tensors $\widehat{\underline{\mathbf{Q}}}$ and $\widehat{\underline{\mathbf{R}}}$, to compute the tensors $\underline{\mathbf{Q}}$ and $\underline{\mathbf{R}}$.

Definition 3 (Transpose): Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ be a given data tensor. Then the conjugate transpose of tensor $\underline{\mathbf{X}}$ is denoted by $\underline{\mathbf{X}}^T \in \mathbb{R}^{I_2 \times I_1 \times I_3}$, which is constructed by applying transpose to all its frontal slices and reversing the order of second till last transposed frontal slices.

Definition 4 (Identity Tensor): Identity tensor $\underline{\mathbf{I}} \in \mathbb{R}^{I_1 \times I_1 \times I_3}$ is a tensor whose first frontal slice is an identity matrix of size $I_1 \times I_1$ and all other frontal slices are zero.

Definition 5 (Orthogonal Tensor): A tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_1 \times I_3}$ is orthogonal (under t-product operator) if $\underline{\mathbf{X}}^T * \underline{\mathbf{X}} = \underline{\mathbf{X}} * \underline{\mathbf{X}}^T = \underline{\mathbf{I}}$.

Definition 6 (f-Diagonal Tensor): If all frontal slices of a tensor are diagonal then the tensor is called f-diagonal. Assume $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, then it can be decomposed as

$$\underline{\mathbf{X}} = \underline{\mathbf{U}} * \underline{\mathbf{S}} * \underline{\mathbf{V}}^T,$$

where $\underline{\mathbf{U}} \in \mathbb{R}^{I_1 \times I_1 \times I_3}$, $\underline{\mathbf{V}} \in \mathbb{R}^{I_2 \times I_2 \times I_3}$ are orthogonal tensors and tensor $\underline{\mathbf{S}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is f-diagonal.

The procedure of computation of the t-SVD for tensors in the Fourier domain is outlined in Algorithm 9. In the next section, we briefly introduce the matrix column selection and CMA approaches. They are used in our subsequent cross tensor analysis.

The MP of tensors can be defined based on the t-product as follows, which will be used in tubal CTA.

Algorithm 10: Computation of Moore-Penrose Pseudoinverse of Tensors

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$

Output: The Moore-Penrose pseudoinverse of tensor $\underline{\mathbf{X}}$

1 $\underline{\mathbf{Y}} = fft\left(\underline{\mathbf{X}}, [], 3\right)$
2 **for** $i = 1, 2, \ldots, I_3$ **do**
3 $\quad \underline{\mathbf{Z}}(:, :, i) = \underline{\mathbf{Y}}(:, :, i)^{+}$
4 **end**
5 $\underline{\mathbf{X}}^{+} = ifft\left(\underline{\mathbf{Z}}, [], 3\right)$

Definition 7: Assume $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, then the MP of the tensor $\underline{\mathbf{X}}$ is denoted by $\underline{\mathbf{X}}^{+} \in \mathbb{R}^{I_2 \times I_1 \times I_3}$ and defined as a unique tensor satisfying the next four relations

$$\underline{\mathbf{X}} * \underline{\mathbf{X}}^{+} * \underline{\mathbf{X}} = \underline{\mathbf{X}}, \quad \underline{\mathbf{X}}^{+} * \underline{\mathbf{X}} * \underline{\mathbf{X}}^{+} = \underline{\mathbf{X}}^{+},$$
$$\left(\underline{\mathbf{X}} * \underline{\mathbf{X}}^{+}\right)^{T} = \underline{\mathbf{X}} * \underline{\mathbf{X}}^{+}, \quad \left(\underline{\mathbf{X}}^{+} * \underline{\mathbf{X}}\right)^{T} = \underline{\mathbf{X}}^{+} * \underline{\mathbf{X}}.$$

Similar to other operations, the MP pseudoinverse of tensors can be computed through FFT and this is described in Algorithm 10.

B. TENSOR LATERAL SLICE SELECTION AND CTA BASED ON TUBAL PRODUCT (T-producT)

In the framework of tubal SVD, a 3rd order data tensor in $\mathbb{R}^{I_1 \times I_2 \times I_3}$ is viewed as a "matrix of tubes" also known as components of the ring $\mathbb{R}^{I_3}$ with the addition and multiplication defined as the vector addition and circular convolution. From this perspective, the lateral and horizontal slices are considered two-dimensional columns and rows of a 3rd-order tensor [62]. The "matrix of tubes" viewpoint leads to the tubal SVD and corresponding tubal rank concept. Here, the tensor variants of the CMA and matrix column selection are called the tubal CTA and lateral slice selection, respectively. To be precise, let $\underline{\mathbf{X}}$ be a given 3rd-order tensor. The tubal CTA is formulated based on t-product as follows

$$\underline{\mathbf{X}} \cong \underline{\mathbf{C}} * \underline{\mathbf{U}} * \underline{\mathbf{R}}, \tag{31}$$

where $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times L_1 \times I_3}$ and $\underline{\mathbf{R}} \in \mathbb{R}^{L_2 \times I_2 \times I_3}$ are some sampled lateral and horizontal slices of the original tensor $\underline{\mathbf{X}}$ respectively and the middle tensor $\underline{\mathbf{U}} \in \mathbb{R}^{L_1 \times L_2 \times I_3}$ is computed in such a way that the approximation (31) should be as small as possible, see Figures 11 and 12, for graphical illustration concerning lateral and horizontal slice selections and also tubal CTA, respectively.

Note that similar to other CTA algorithms discussed so far, the procedure of both lateral and horizontal slices sampling can be performed based on prior probability distributions.[5] The probability distributions used in [62] are uniform and nonuniform (length-squared and leverage scores) distributions. Let us first consider the tubal CTA. Similar to the CMA, the best solution for the middle tensor $\underline{\mathbf{U}}$ in the least-squares sense is

$$\underline{\mathbf{U}} = \underline{\mathbf{C}}^{+} * \underline{\mathbf{X}} * \underline{\mathbf{R}}^{+}, \tag{32}$$

[5]Here, various probability distributions (with/without replacement) can be used but we will not go through the theoretical details.

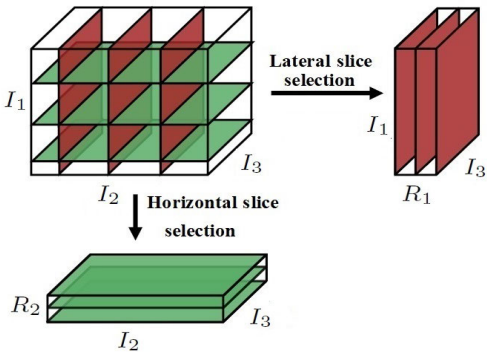

FIGURE 11. Illustration of lateral and horizontal slice selections for the tubal CTA.

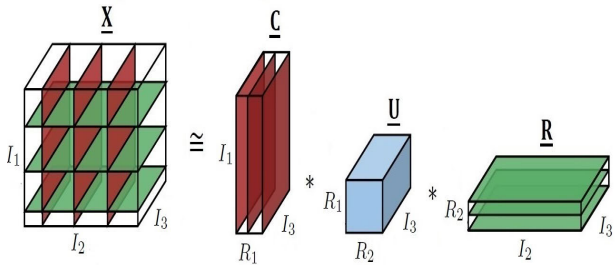

FIGURE 12. Illustration of the tubal CTA for a 3rd-order tensor.

which is a straightforward generalization of the CMA [67] to tensors. Formula (32) can be computed in the Fourier domain and these computations are summarized in Algorithm 11. However, it is clear that (32) needs to pass the data tensor $\underline{\mathbf{X}}$ once again, and this is of less practical interest for very large-scale data tensors, especially when the data tensors do not fit into the memory and communication between memory and disk is expensive [75]. To solve this problem, the MP pseudoinverse of the intersection subtensor $\underline{\mathbf{W}} \in \mathbb{R}^{L_2 \times L_1 \times I_3}$, which is obtained based on intersecting the sampled horizontal and lateral slices, should be approximated as

$$\underline{\mathbf{U}} = \underline{\mathbf{C}} * \underline{\mathbf{W}}^{+} * \underline{\mathbf{R}}.$$

It is not difficult to see that the tensor $\underline{\mathbf{W}}$ consists of some tubes of the original data tensor $\underline{\mathbf{X}}$. A similar algorithm for computation of the tensor $\underline{\mathbf{W}}$, is proposed in [62] which is a tensor generalization of that proposed for matrices in [26] but as mentioned before the algorithms are rather sophisticated. Algorithm 12 describes this procedure.

Remark 3: For data tensors with missing entries, an algorithm is proposed in [118] for the computation of the tubal CTA. It is a generalization of the algorithm developed in [119] to compute the CMA of data matrices with missing entries.

Remark 4: In Algorithm 9, in Line 3, the truncated SVD can be replaced by cross algorithms such as Cross2D, Maxvol and sampling techniques and the resulting algorithms can be considered as new generalizations of CTA. The idea of sampling applied on lateral slices was used in [118].

Algorithm 11: Two-Passes Tubal CTA (TP-TCTA) Algorithm

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, positive numbers $L_1,\ L_2$, probability distributions $p_i,\ i = 1, 2, \ldots, I_2$ and $p'_i,\ i = 1, 2, \ldots, I_1$

Output: Tubal CTA $\underline{\mathbf{X}} \cong \underline{\mathbf{Z}} = \underline{\mathbf{C}} * \underline{\mathbf{U}} * \underline{\mathbf{R}}$

Select L_1 lateral slices of the tensor $\underline{\mathbf{X}}$ based on probability distributions $p_i,\ i = 1, 2, \ldots, I_2$ and set tensor $\underline{\mathbf{C}}$

Select L_2 horizontal slices of the tensor $\underline{\mathbf{X}}$ based on probability distributions $p'_i,\ i = 1, 2, \ldots, I_2$ and set tensor $\underline{\mathbf{R}}$

$\widehat{\underline{\mathbf{X}}} = fft\left(\underline{\mathbf{X}}, [], 3\right)\ \widehat{\underline{\mathbf{C}}} = fft\left(\underline{\mathbf{C}}, [], 3\right),\ \widehat{\underline{\mathbf{R}}} = fft\left(\underline{\mathbf{R}}, [], 3\right)$

for $i = 1, 2, \ldots, I_3$ **do**

  $\widehat{\underline{\mathbf{U}}}(:, :, i) = \widehat{\underline{\mathbf{C}}}(:, :, i)^{+}\, \widehat{\underline{\mathbf{X}}}(:, :, i)\, \widehat{\underline{\mathbf{R}}}(:, :, i)^{+}$

end

for $i = 1, 2, \ldots, I_3$ **do**

  $\widehat{\underline{\mathbf{Z}}}(:, :, i) = \widehat{\underline{\mathbf{C}}}(:, :, i)\, \widehat{\underline{\mathbf{U}}}(:, :, i)\, \widehat{\underline{\mathbf{R}}}(:, :, i)$

end

$\underline{\mathbf{Z}} = ifft\left(\widehat{\underline{\mathbf{Z}}}, [], 3\right),\ \underline{\mathbf{U}} = ifft\left(\widehat{\underline{\mathbf{U}}}, [], 3\right)$

Also, the tensor lateral slice selection based on the t-product is formulated as follows

$$\underline{\mathbf{X}} \cong \underline{\mathbf{C}} * \widetilde{\underline{\mathbf{X}}}, \tag{33}$$

where $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times L \times I_3}$ is a part of lateral slices of the tensor $\underline{\mathbf{X}}$ and the tensor $\widetilde{\underline{\mathbf{X}}} \in \mathbb{R}^{L \times I_2 \times I_3}$ is computed in such a way that the reconstruction error (33) is as small as possible [62], (see Figure 13). The best option for the tensor $\widetilde{\underline{\mathbf{X}}}$ is

$$\widetilde{\underline{\mathbf{X}}} = \underline{\mathbf{C}}^{+} * \underline{\mathbf{X}},$$

which provides the best approximation in a least-squares sense, that is

$$\left\|\underline{\mathbf{X}} - \underline{\mathbf{C}} * \left(\underline{\mathbf{C}}^{+} * \underline{\mathbf{X}}\right)\right\|_F = \min_{\underline{\mathbf{Y}} \in \mathbb{R}^{L \times I_2 \times I_3}} \left\|\underline{\mathbf{X}} - \underline{\mathbf{C}} * \underline{\mathbf{Y}}\right\|_F$$

through a projection approximation as $\underline{\mathbf{X}} \cong \underline{\mathbf{C}} * \underline{\mathbf{C}}^{+} * \underline{\mathbf{X}}$.

This procedure is summarized in Algorithm 13. Please note that this algorithm needs to pass the original whole data tensor $\underline{\mathbf{X}}$ but a single-pass variant can be found in [120], which is a direct generalization of the matrix case developed in [83]. The span of the lateral slices can be captured via random projection [121], i.e., multiplication with random tensors, which is equal to the linear combination of the lateral slices. The span of the lateral slices can also be captured via the count-sketch idea [122]. To the best of our knowledge, this idea has not been investigated yet. Besides, all results and algorithms discussed so far for the tubal CTA are for 3rd order tensors and generalization of these results to Nth order tensors is possible using the theory presented in [116].

V. NUMBER OF PARAMETERS AND COMPUTATIONAL COMPLEXITY

In this section, we compare the number of parameters of the CTA algorithms and the associated computational complexity

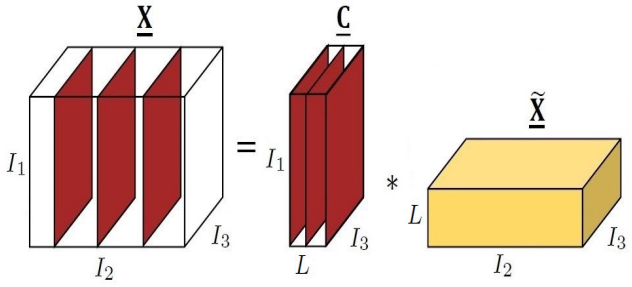

FIGURE 13. Illustration of the CX approximation based on the t-product for a 3rd-order tensor.

Algorithm 12: Fast Tubal CTA (F-TCTA) Algorithm

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, positive numbers $R_1,\ R_2$, probability distributions $p_i,\ i = 1, 2, \ldots, I_2$ and $p'_i,\ i = 1, 2, \ldots, I_1$

Output: Tubal CTA: $\underline{\mathbf{X}} \cong \underline{\mathbf{Z}} = \underline{\mathbf{C}} * \underline{\mathbf{U}} * \underline{\mathbf{R}}$

Select R_1 lateral slices of the tensor $\underline{\mathbf{X}}$ based on probability distributions $p_i,\ i = 1, 2, \ldots, I_2$ and set tensor $\underline{\mathbf{C}}$

Select R_2 horizontal slices of the tensor $\underline{\mathbf{X}}$ based on probability distributions $p'_i,\ i = 1, 2, \ldots, I_1$ and set tensor $\underline{\mathbf{R}}$

Compute the intersection subtensor $\underline{\mathbf{W}}$

$\widehat{\underline{\mathbf{C}}} = fft\left(\underline{\mathbf{C}}, [], 3\right),\ \widehat{\underline{\mathbf{R}}} = fft\left(\underline{\mathbf{R}}, [], 3\right),\ \widehat{\underline{\mathbf{W}}} = fft\left(\underline{\mathbf{W}}, [], 3\right)$

for $i = 1, 2, \ldots, I_3$ **do**

  $\widehat{\underline{\mathbf{U}}}(:, :, i) = \widehat{\underline{\mathbf{W}}}(:, :, i)^{+}$

end

for $i = 1, 2, \ldots, I_3$ **do**

  $\widehat{\underline{\mathbf{Z}}}(:, :, i) = \widehat{\underline{\mathbf{C}}}(:, :, i)\, \underline{\mathbf{U}}(:, :, i)\, \widehat{\underline{\mathbf{R}}}(:, :, i)$

end

$\underline{\mathbf{Z}} = ifft\left(\widehat{\underline{\mathbf{Z}}}, [], 3\right),\ \underline{\mathbf{U}} = ifft\left(\widehat{\underline{\mathbf{U}}}, [], 3\right)$

required for computing them. We evaluate the performance of the algorithms in Section IX, experimentally. For the simplicity of presentation, we consider a 3rd order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times I \times I}$ and depending on the CTA model of interest, the following ranks and parameters are considered

- (Tucker based model): $R_1 = R_2 = R_3$,
- (Tubal CTA model): $L_1 = L_2 = R$,
- (Model (27)): $L_1 = L_2 = R$.

In the Tucker based model, we assume that R rows, R columns and R tubes are selected. In the tubal CTA model, R lateral and R horizontal slices are selected, and for the case of lateral slice selection (model (33)) which only deals with the lateral slice selection, R lateral slices are sampled. In model (27), where both frontal slices and tubes should be selected, the number of sampled slices and tubes are the same and equal R. For the SP-STucker algorithm, the sketching size $S = (S_1, S_2, \ldots, S_N)$ is used where $S_n > 2R_n$ and R_n is the nth component of the multilinear rank.[6]

[6]In our experimental results, we have used $S_n = 2K_n + 1$ where $K_n > R_n$.

TABLE 2. The number of parameters and computational complexity of different CTA methods for low multilinear rank approximation (R, R, R) approximation, tubal rank R of a 3rd order tensor of size $I \times I \times I$. For the CTA-FS Algorithm, R frontal slices and R tubes are selected.

CTA Model	Number of parameters		Space complexity	Time complexity
STucker Algorithm	$\mathcal{O}(IR + R^3)$		$\mathcal{O}(I^3 + 3IR)$	$\mathcal{O}(I^3R + I^2R + IR^2 + IR^3)$
SP-STucker Algorithm	$\mathcal{O}(IR + R^3)$		$\mathcal{O}(I^3 + 3IR)$	$\mathcal{O}(I^3S + S^3R + S^2R^2 + SR^3)$
Cross3D Algorithm	$\mathcal{O}(IR + R^3)$		$\mathcal{O}(IR)$	$\mathcal{O}(I^2R^2)$
FSTD Algorithm	$\mathcal{O}(IR + R^3)$		$\mathcal{O}(IR^2 + R^3)$	$\mathcal{O}(IR^3 + R^4)$
AFS Algorithm	$\mathcal{O}(IR + R^3)$		$\mathcal{O}(3IR)$	$\mathcal{O}(IR^4 + R^5)$
HOID Algorithm	$\mathcal{O}(IR + R^3)$		$\mathcal{O}(I^3 + 3IR)$	$\mathcal{O}(I^3R + I^2R + IR^2)$
CTA-FS Algorithm	Uniform Sampling	$\mathcal{O}(I^2R + IR + R^2)$	$\mathcal{O}(I^2R + IR)$	$\mathcal{O}(I^2R^2)$
	Length-Squared	$\mathcal{O}(I^2R + IR + R^2)$	$\mathcal{O}(I^3 + 2I^2R)$	$\mathcal{O}(I^3 + I^2R^2)$
TP-TCTA Algorithm	$\mathcal{O}(I^2R + IR^2)$		$\mathcal{O}(I^3 + 2I^2R)$	$\mathcal{O}(I^3 \log(I) + I^3R + I^2R^2)$
F-TCTA Algorithm	$\mathcal{O}(I^2R + IR^2)$		$\mathcal{O}(2I^2R + R^2I)$	$\mathcal{O}(I^2 \log(I)R + I^2R^2)$
LSS Algorithm	$\mathcal{O}(I^2R)$		$\mathcal{O}(I^3 + 2I^2R)$	$\mathcal{O}(I^3 \log(I) + I^3R + I^2 \log(I)R + I^2R^2)$

Algorithm 13: Lateral Slice Selection (LSS) Algorithm

Input : A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, a positive numbers L, probability distribution $p_i,\ i = 1, 2, \ldots, I_2$

Output: Low tubal-rank approximation $\underline{\mathbf{X}} \cong \underline{\mathbf{Z}} = \underline{\mathbf{C}} * \underline{\mathbf{C}}^{+} * \underline{\mathbf{X}}$

Select L lateral slices of the tensor $\underline{\mathbf{X}}$ based on probability distributions $p_i,\ i = 1, 2, \ldots, I_2$ and set tensor $\underline{\mathbf{C}}$

$\widehat{\underline{\mathbf{C}}} = fft\left(\underline{\mathbf{C}}, [], 3\right),\ \widehat{\underline{\mathbf{X}}} = fft\left(\underline{\mathbf{X}}, [], 3\right),$

for $i = 1, 2, \ldots, I_3$ **do**

  $\widehat{\underline{\mathbf{Z}}}(:, :, i) = \widehat{\underline{\mathbf{C}}}(:, :, i)\, \widehat{\underline{\mathbf{C}}}(:, :, i)^{+}\, \widehat{\underline{\mathbf{X}}}(:, :, i)$

end

$\underline{\mathbf{Z}} = ifft\left(\widehat{\underline{\mathbf{Z}}}, [], 3\right)$

The number of parameters and also computational complexity of the CTA algorithms are outlined in Table 2. Note that in order to achieve the same level of accuracy for different CTA algorithms, the number of fibers, slices, and tubes of the algorithms may not be the same, so the comparison made in Table 2 is true only for the above-mentioned special case. The complexity of SP-STucker algorithm can be higher than the STucker algorithm but as discussed in the paper, the former is pass-efficient while the latter needs to pass the original data tensor for every multilinear rank. Here the cost of communication may exceed our main computations.

VI. PASS-EFFICIENT ALGORITHMS

In the randomized framework for low-rank tensor/matrix approximation, the whole dataset should be passed as few times as possible and according to this, the randomized algorithms are categorized to the single-pass and the multi-pass randomized algorithms. The pass-efficient algorithms refer to algorithms that need to access the data only a few times. The elements of the data tensors are used in the sketching procedure where a summary of the original large dataset is produced to be used in the subsequent computations. More precisely, the elements of the dataset are used to capture the range of unfolding matrices [55], computing the probability distribution under which the slices/fibers are selected [54], [58]. The question is that is it possible to compute a low-rank tensor approximation without even passing the data tensor one time? Clearly based on the algorithms we discussed so far, the answer is yes, and actually, in Table 3, we have categorized the algorithms in the sense of the number of passes they need to compute a low-rank approximation. Note that some CTA algorithms require at least one time to access the whole dataset, but some only need a part of the data. The latter category is of more interest, especially when the whole data tensor is extremely large and cannot be handled in a single machine. It is worth mentioning that if the uniform sampling is used for slice/tube selection within the CTA-FS algorithm, it does not need to pass the whole data tensor, while if the length squared probability distribution in (28) is used, then it becomes a single-pass algorithm.

In Table 3, we have not explicitly mentioned the number of slices and tubes to be selected in the CTA-FS algorithm because there is not a specific definition of rank for this decomposition.

VII. APPLICATIONS OF CTA MODELS

CTA can be potentially utilized in several applications in which the low-rank tensor approximation is required. Such applications include but not limited to

- Tensor completion
- Tensor Robust PCA
- Denoising
- Compressed Sensing (CS)

The idea of replacing the deterministic low-rank tensor decomposition with the CMA counterparts reduces the computational complexity of the algorithms and leads to faster algorithms. In this section, we discuss the possibility of using CTA methods for the three above-mentioned problems.

A. TENSOR COMPLETION

Tensor completion is the problem of recovering an incomplete data tensor from partially observed components and has found several applications in recommendation systems [123], traffic data prediction [124], [125], image processing [126], etc. The most useful and widely used technique for matrix completion is through the nuclear norm minimization [127], [128], which is the tightest convex surrogate of the matrix

TABLE 3. The number of passes required in different CTA methods for low multilinear rank approximation (R, R, R) approximation, tubal rank R of a 3rd order tensor of size $I \times I \times I$.

CTA Model	Number of passes		Remark
STucker Algorithm	1		–
SP-STucker Algorithm	1		–
Cross3D Algorithm	Does not pass the whole data		It needs only IR numbers of components.
FSTD Algorithm	Does not pass the whole data		It needs only $3IR^2 + R^3$ numbers of components
AFS Algorithm	Does not pass the whole data		It needs only IR numbers of components of the original tensor
HOID Algorithm	1		–
CTA-FS Algorithm	Uniform Sampling	Does not pass the whole data	It needs only some slices and fibers
	Length-Squared	1	-
TP-TCTA Algorithm	2		–
F-TCTA Algorithm	Does not pass the whole data		It needs only $2I^2R + IR^2$ numbers of components
LSS Algorithm	1		–

rank. As mentioned before, the notion of rank is not unique for tensors. In fact, in tensor completion problems, we use different types of tensor decomposition and the corresponding tensor ranks [129]. However, in almost all of these formulations, the key operation is the computation of the SVD of some matrices which are computationally prohibitive. In [130]–[133], the SVD computation is replaced by the randomized SVD, while in [119], the CMA decomposition is used. For the tensor case, the same idea can be exploited. For example, tubal CTA is used in [62] and [118] for tensor completion in two different ways. In [118], the CMA was used for low-rank approximation of the underlying matrices, while in [62] some lateral and horizontal slices are selected. The idea of incorporating the CMA within the framework of different tensor completion models such as Tucker, TT/TR, and CP decompositions is a potential topic that needs to be further investigated.

B. ROBUST PCA

As there exist different kinds of tensor decompositions, several different tensor RPCA (TRPCA) problems can be formulated naturally. The formulation of TRPCA based on the Tucker and the tubal tensor decomposition are introduced in [62] and [134], [135] respectively. The main computationally expensive operation involved in these formulations is either computing the SVD or tensor decomposition of some underlying matrices or tensors. Motivated by this bottleneck, for the matrix RPCA, in [136], [137], the randomized SVD for low-rank approximation is used while in [31] the CMA is utilized where the computation of the SVD is replaced by the CMA.

In [62], the CMA is used to find a low tubal rank approximation of the underlying data tensors.

The idea of exploiting the CMA approaches for low-rank approximation can be analogously used in robust TT/TR decompositions [138], and faster algorithms can be developed.

C. DENOISING

The truncated tensor decomposition algorithms can be naturally used for denoising the data tensors in a similar way as the SVD is used for denoising the data matrices. The deterministic algorithms for tensor decompositions can be replaced by the randomized tensor decomposition variants, e.g., CTA algorithms. For example, in [139], the randomized CPD decomposition is used, while in [62], the tubal CTA algorithm is exploited. The application of the CTA algorithms for tensor denoising is an interesting research topic that needs to be further investigated.

D. COMPRESSED SENSING (CS)

In recent years, Compressed Sensing (CS) technology has accelerated signal acquisition by restricting the number of measurements in, for example, the Fourier domain [93], [94], [140]. This is the case of Magnetic Resonance Imaging (MRI) systems for medical applications where the measurements are taken explicitly in the Fourier domain, also known as the k-space. Here, the goal of CS is to reconstruct an image from incomplete measurements taken in the Fourier domain. It is interesting to note that in the MLProj reconstruction formula (eq. (12)), when the projection matrices Φ_n are constructed as collections of selected columns in the Fourier transform matrix, then taking measurements $\underline{\mathbf{Z}}^{(n)}$ correspond to subsampling fibers in the Fourier domain. In Section IX, we illustrate how to apply the MLProj method to recover cardiac cine MRI data from incomplete measurements in the Fourier domain.

VIII. DISCUSSION AND FUTURE CHALLENGES

Recently many approaches and results, i,g., perturbation analysis, have been investigated in [68], [141], for CMA and it is of intense interest to generalize these results to tensors. For example, some of these results are generalized to tensors in [63], [142]. The current tubal CTA algorithms are randomized where the lateral and horizontal slices are selected randomly. The generalization of the max-volume and adaptive cross concepts to this type of decomposition is a challenging topic. The CMA is used in [29] for compressing fully connected layers of deep neural networks (DNNs). Closely related works can be found in [143], where instead of the CMA, the randomization technique is used. As the intrinsic structure of the layers in the DNNs are in the tensor forms, many tensor decomposition models have been used

in which the tensor decomposition methods are utilized to compress either fully connected or convolutional layers, for example, see [107], [144]–[146]. Exploiting the randomized algorithms, particularly the CTA algorithms, for compressing layers in DNNs seems to be a perspective research direction.

In the next section, we discuss the challenges and methods for tensor rank selection which plays a key role in the tensor analysis.

A. TENSOR RANK SELECTION METHODS

The concept of rank in the context of tensor decomposition is different to the one used for matrices and can take different forms depending on the type of the used tensor decomposition. For example, when using the CPD, the tensor rank is defined as the minimum number of rank-1 tensor terms needed to exactly represent a given data tensor. Tensor rank selection is challenging for CPD; in fact, it is known that its determination is NP-hard [147]. Moreover, in contrast to the matrix case, given a CPD approximation of a data tensor, it is not guaranteed that a reduction of the approximation error can be achieved by increasing the number of rank-1 tensors in the CPD. However, some pragmatic procedures have been proposed in the past, such as the core consistency diagnostic (CORCONDIA) algorithm [21] and other alternatives [148]–[150]. Fortunately, the previous difficulties are not present when working with a Tucker approximation. In this case, the Tucker-rank or multilinear rank is the tuple $(R_1, R_2, \ldots, R_N)$ that determines the size of the core tensor. It is essential to highlight that the Tucker-rank can be, in theory, computed by finding the matrix-rank of each unfolding matrix $\mathbf{X}_{(n)}$ obtained from the data tensor $\underline{\mathbf{X}}$. More importantly, in this case, it is guaranteed that, given a data tensor for which we don't know in advance its multilinear-rank, we can select some initial Tucker-rank and reduce the approximation error by successively increasing the multilinear rank [151]. Although the multilinear rank of a given tensor can be computed exactly or approximated, it would require accessing the full data tensor and involve high computational cost. In practice, our methods use a greedy approach; in other words, they usually start at some initial multilinear rank and successively increase it until the approximation error is less than some threshold ϵ. The tubal SVD is another decomposition defined based on the concept of tubal tensor-tensor multiplication. Here, the tubal rank is defined. Similar to the Tucker model, computing the tubal rank is possible. The interesting property of the truncated t-SVD is that it provides the best rank approximation under any unitary invariant tensor norm. In this paper, we studied the CTA algorithms mostly based on the Tucker model and the tubal SVD decomposition.

IX. SIMULATIONS

In this section, we experimentally evaluate the validity and performance of the CTA algorithms discussed in the papers. The MATLAB implementations of the algorithms are accessible at https://github.com/SalmanAhmadi-Asl/Cross_Tensor_Approximation.

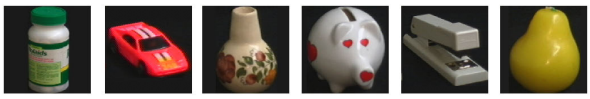

FIGURE 14. Some random sample images from the COIL-100 dataset (Object number 5, 23, 30, 48, 68, 83 under the 250 degree rotation).

All numerical simulations were performed on a cluster computer 120Gb RAM and 48 CPUs 1.2GHz. The efficiency and performance of algorithms were compared in terms of *compression ratio*, *sampling rate*, *relative error* and *running time* on real datasets.

The compression ratio is defined as

$$Compression\ ratio = \frac{C_1}{C_2},$$

where C_1 and C_2 are, respectively, the number of components of the original data tensor and the number of the parameters to represent a data tensor in the decomposition format. The sampling ratio is defined as

$$Sampling\ ratio = \frac{C_3}{C_1},$$

where C_3 is the number of observed (sampled) entries in the original data tensor. The sampling ratio is used to compare the CTA algorithms which do not need to pass the whole data tensor, e.g., FSTD Algorithm.

The relative error is defined as

$$Relative\ error = \frac{\left\|\widetilde{\underline{\mathbf{X}}} - \underline{\mathbf{X}}\right\|_F}{\left\|\underline{\mathbf{X}}\right\|_F},$$

where $\widetilde{\underline{\mathbf{X}}}$ and $\underline{\mathbf{X}}$ are approximate and exact data tensors, respectively. Note that in all our sampling algorithms, we use sampling without replacement.

The PSNRs is used to assess the quality of the images reconstructed by the algorithms, defined as

$$PSNR = 10\log_{10}\left(255^2/\mathrm{MSE}\right),$$

where $MSE = \left\|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\right\|_F^2 / \mathrm{num}\left(\underline{\mathbf{X}}\right)$, and "num" denotes the number of parameters of a given data tensor.

Example 4.1 (Real Data Compression): In this experiment, we applied the CTA algorithms to compress COIL-100 dataset [152]. This data tensor consists of 7200 color images (100 objects and 72 rotations per object, see Figure 14 for some samples of this data tensor). The size of each image is $128 \times 128 \times 3$, and the whole data is represented as a fifth-order tensor of size $128 \times 128 \times 3 \times 100 \times 72$. Since some of the CTA algorithms are applicable only for 3rd order tensors, we first reshaped the tensor to a 3rd order tensor of size $768 \times 768 \times 600$, then applied the CTA algorithms to compress it. Finally, the approximate tensors were reshaped to the original size of the dataset. We first considered the algorithms which only need to access a part of the data tensor, i.e., the FSTD algorithm, the Cross3D algorithm, the AFS

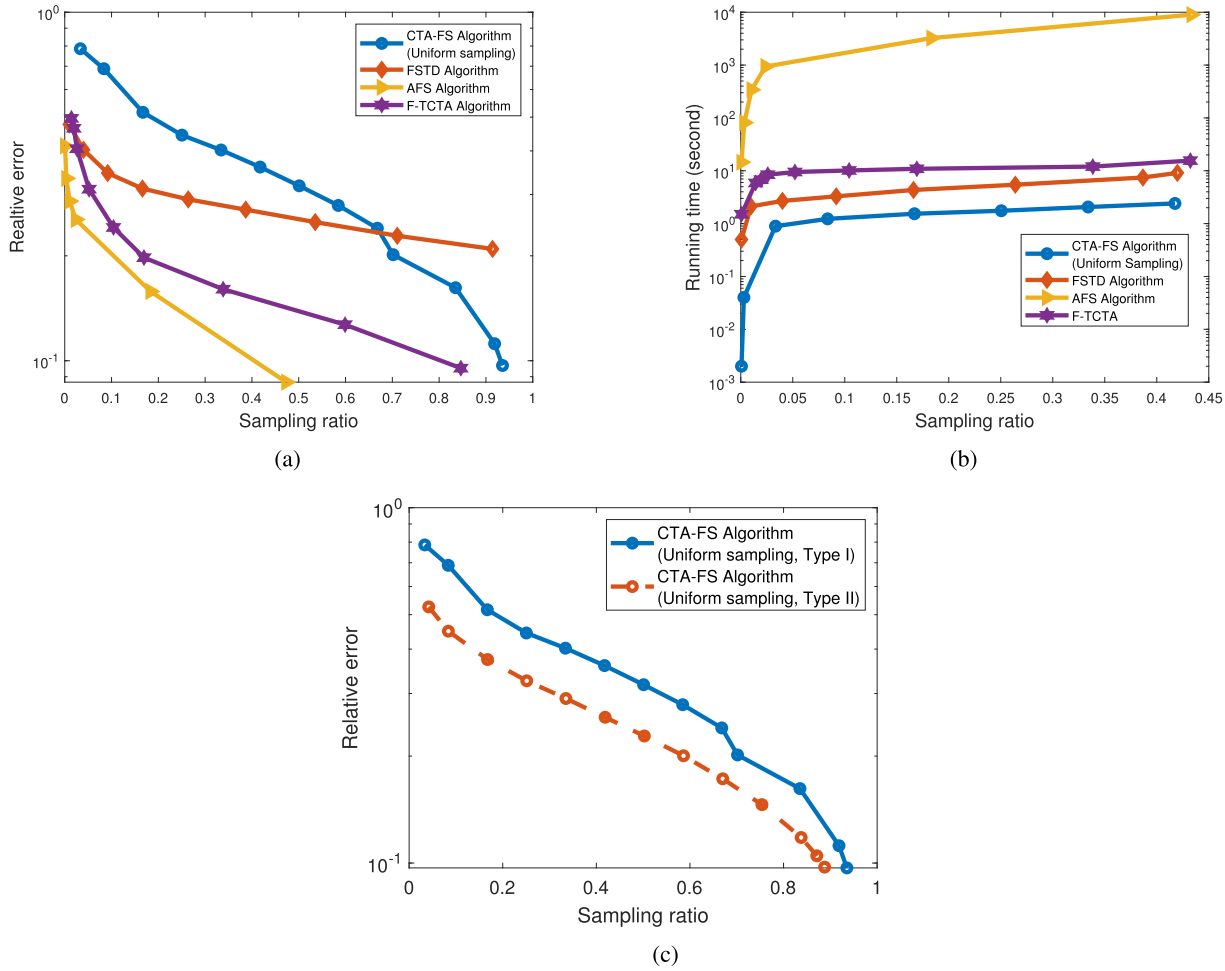

FIGURE 15. **(a) Relative error comparison versus sampling ratio for the sampling CTA algorithms, (b) Running time comparison versus sampling ratio for the sampling CTA algorithms, (c) Relative error comparison versus sampling ratio for the CTA-FS algorithm. (Type I. The number of slices is twice the number of frontal fibers. Type II. The number of slices is five times larger than the tubes.)**

algorithm, the CTA-FS algorithm, and the F-TCTA algorithm. The results achieved by the naive Cross3D algorithm[7] and the AFS algorithm were almost the same, and because of this we only report the AFS algorithm.

For the FSTD algorithm, we assumed that the number of indices that are selected in each mode are the same, and were chosen in the range, 40, 80, 120, . . . , 360. For the AFS algorithm, we assumed that the number of fibers to be selected in each mode are the same and they were selected in the range, 50, 100, 150, . . . , 550. For the CTA-FS algorithm, we considered the uniform sampling without replacement and the number of slices to be selected were 50, 100, 150, . . . , 500 and also, we assumed that the number of tubes is twice the number of slices, i.e., 100, 200, 300, . . . , 1000. We found that sampling without replacement is more efficient than sampling with replacement for the slice selection case while we did not see any significant difference for the various fiber selection scenarios.

For the F-TCTA, we found that the algorithm is sensitive to the number of selected lateral/horizontal slices. Our experiments show that when the number of lateral/horizontal are the same or close to each other, the approximation is relatively poor. However, as they are different (far from each other), the results are considerably improved. In our experiments, we used (3, 8), (5, 10), (5, 15), (10, 30), (60, 20), (40, 90), (160, 100), (260, 200), (290, 360) where the first/second component denotes the number of selected lateral/horizontal slices. For each of the above assumptions, we computed the sampling ratio, running time, and relative errors. The relative error and running time comparisons against sampling ratio are reported in Figures 15 (a)-(b), respectively. From the results, we found that the AFS /Cross3D algorithms are the most promising and accurate CTA algorithms for computing low tensor approximation but only when the multilinear rank is relatively small. Nonetheless, when the multilinear rank is large, their computational

[7]Applying the Cross2D directly on the unfolding without taking into account the second cross approximation of the reshaped form of the long rows.

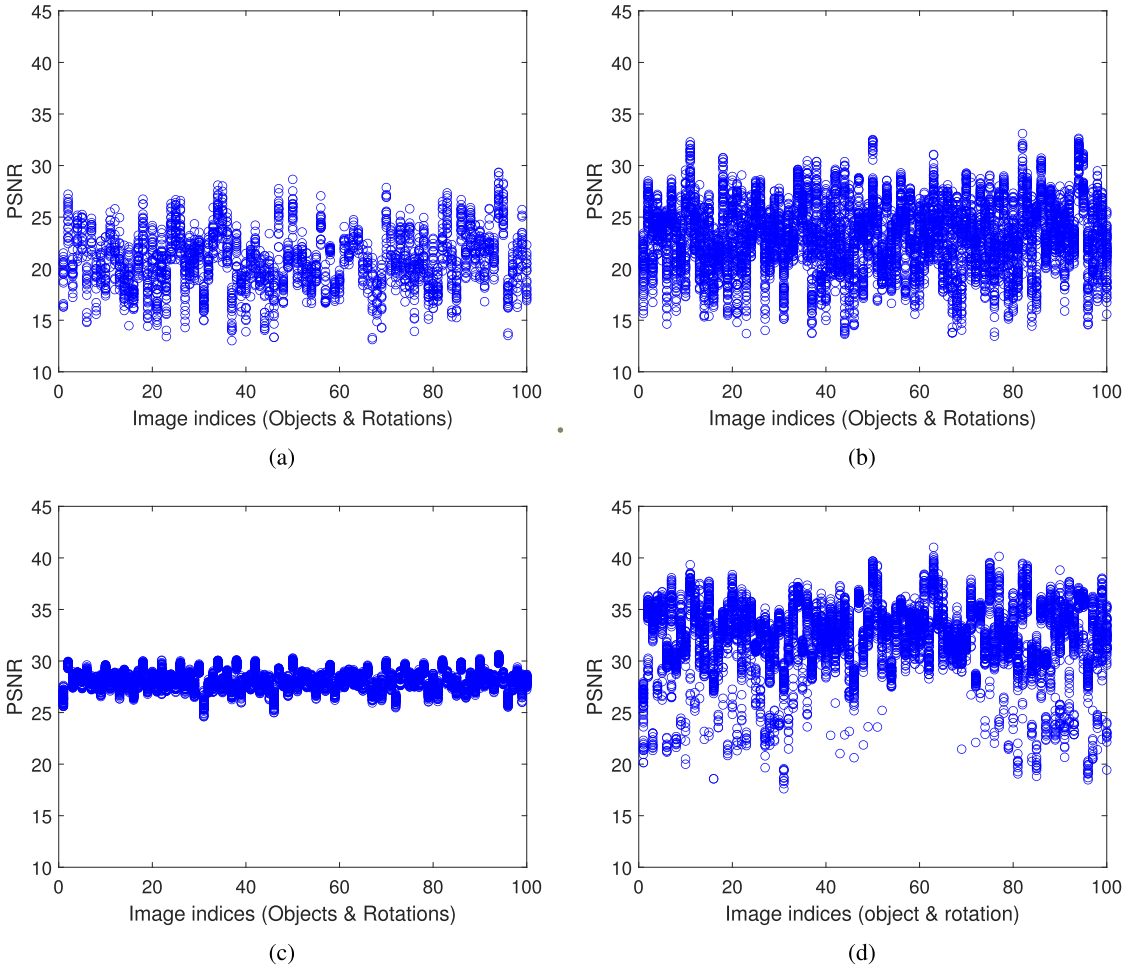

FIGURE 16. The quality of images reconstructed by the sampling CTA algorithms (a) CTA-FS algorithm (Uniform sampling) (sampling ratio 0.6701 with relative error 0.1683), (b) FSTD algorithm (sampling ratio 0.7106 with relative error 0.2283), (c) F-TCTA (sampling ratio 0.6120 with relative error 0.1168), (d) AFS algorithm (sampling ratio 0.4734 with relative error 0.0868).

complexity is exponentially increased. Note that in the case of the CTA-FS algorithm, there is no any criterion guiding how many slices and fibers should be selected and how their relationship should be to achieve the optimal compression. In our experiments, we first considered the case that the number of fibers is twice the number of slices, which might not always be an optimal criterion. Due to this fact, we performed a new experiment where the number of tubes to be selected was five times larger than the number of slices. The results of this experiment are reported in Figure 15 (c), where Type I/II refers to the case that the number of tubes is two/five times larger than the frontal slices. From Figure 15, it is seen that the CTA-FS algorithm using Type II parameters achieves a lower approximation error than Type I.

To compare the reconstruction properties and performances of the algorithms, we first consider the CTA-FS algorithm and select 400 frontal slices and 2000 tubes. The sampling ratio, in this case, is 0.6701, with the corresponding relative error 0.1683. We found that the CTA-FS algorithm has a different feature than other CTA algorithms. More precisely, it reconstructs some images of the data set perfectly[8] while the other images might be good or bad approximated. In our experiment, a total of 4800 images were recovered by inf PSNR. The PSNRs of the rest 2400 images reconstructed by CTA-FS algorithm can be seen in Figure 16 (a).

For the FSTD algorithm, we assumed that the same number of indices in each mode (equal to 320) is selected. For this case, the sampling ratio is 0.7106 with the corresponding relative error 0.2283. The quality of images reconstructed by this algorithm is displayed in Figure 16 (b).

For the F-TCTA algorithm, we assumed that 270 horizontal slices and 200 lateral slices were selected. Here, we have a sampling ratio 0.6120 and a relative error 0.1168. The quality of reconstructed images by this algorithm are displayed in Figure 16 (c).

For the AFS algorithm, we assumed that 550 fibers are selected in each mode. Here, we have a sampling ratio 0.4734

[8]With inf PSNR.

FIGURE 17. Reconstructed images by the CTA algorithms for some randomly selected images, a) CTA-FS algorithm (Length-Squared sampling) (compression ratio 1.4874 and relative error 0.1651), b) CTA-FS algorithm (Uniform sampling) (sampling ratio 0.6701 with relative error 0.1683), c) F-TCTA algorithm (sampling ratio 0.6120 with relative error 0.1168), d) FSTD algorithm (sampling ratio 0.7106 with relative error 0.2283), e) AFS algorithm (sampling ratio 0.4734 with relative error 0.0868), f) STucker algorithm (compression ratio 1.6287 and relative error 0.0740), g) LSS algorithm (compression ratio 1.6134 and relative error 0.0740), h) TP-TCTA algorithm (compression ratio 1.6526 and relative error 0.0799).

and a relative error 0.0868. The PSNRs of the reconstructed images using this algorithm are displayed in Figure 16 (d).

The recovered images of some samples corresponding to the above-mentioned cases are visualized in Figure 17.

In the next step, we compared the efficiency of the CTA algorithms, which need at least one pass to all parts of the data tensor to compute a low tensor approximation, i.e., the STucker algorithm, the SP-STucker algorithm, the HOID algorithm, the TP-TCTA algorithm, and the LSS algorithm. For the STucker and HOID algorithms, we assumed that the number of fibers to be selected in each mode was the same and in the range, $R = 50, 150, \ldots, 600$. For the LSS algorithm, we assumed that the number of lateral slices are uniformly sampled and are in the range, $R = 10, 40, 70, \ldots, 240$, and for the TP-TCTA algorithm, we assumed that the number of lateral/horizontal slices to be selected are the same and in the range, $R = 10, 40, 70, \ldots, 240$. We also used the CTA-FS algorithm with length-squared probability distribution in (28) as a single-pass algorithm in our experiments. We assumed that the number of slices to be selected were in the range, $R = 50, 100, 150, \ldots, 500$, and the number of selected tubes were five times larger than the number of frontal slices. Figures 18 (a) and (b), respectively, show the probability distribution under which the frontal slices and tubes were selected. Note that since all the mentioned algorithms pass the whole data tensor at least once, we compared the algorithms in terms of running time and compression ratio, not

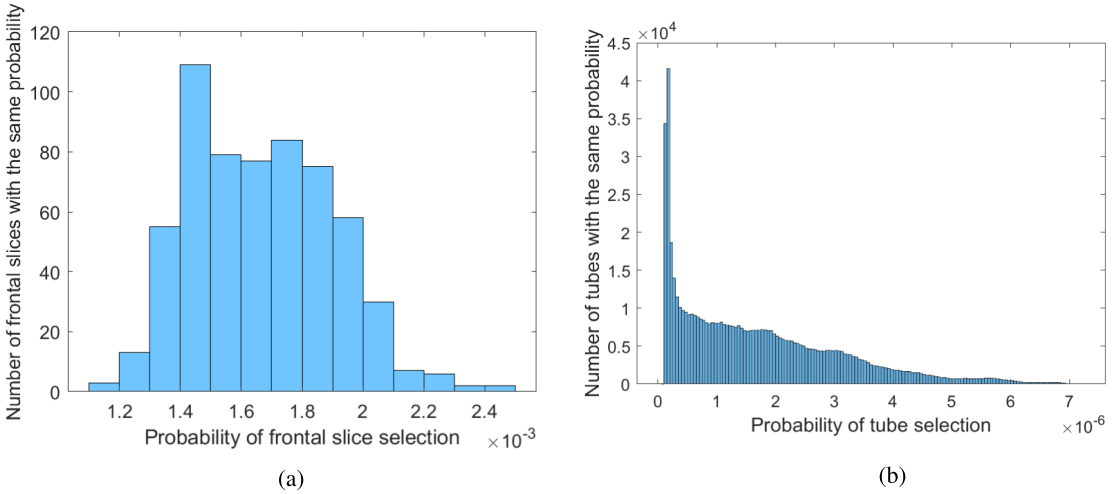

FIGURE 18. (a) The probability distribution for frontal slice selection, (b) The probability distribution for tube selection.

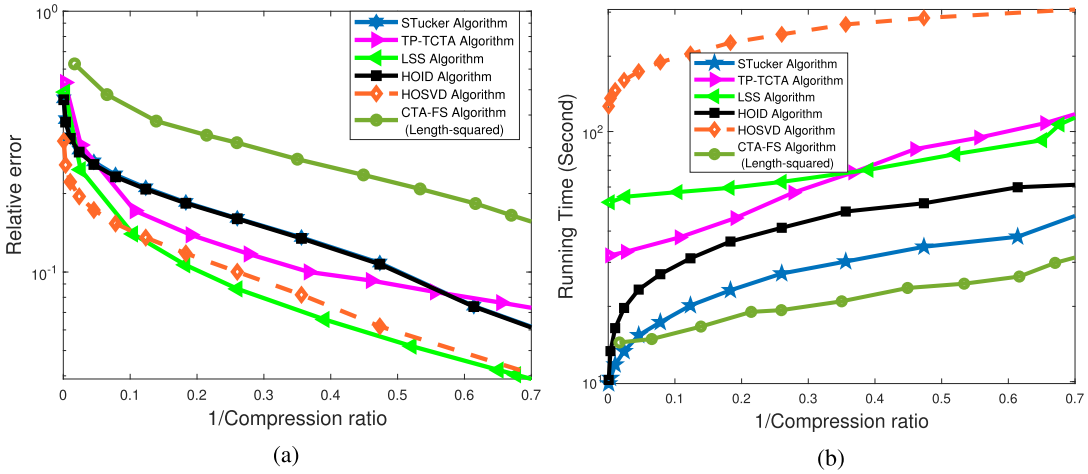

FIGURE 19. (a) Relative error comparison versus sampling ratio for the pass-efficient algorithms, (b) Running time comparison versus sampling ratio for the pass-efficient algorithms.

sampling ratio. The results are reported in Figures 19 (a)-(b). For this dataset, the LSS algorithms achieved the best results in terms of relative error and compression ratio, while its computational complexity was higher. We remark that for this dataset, the FCD-FS algorithm with length-squared probability distribution provides almost the same performance as the FCD-FS algorithm with the uniform sampling because from Figures 18 (a)-(b), it is seen that the probability distribution of slice/tube selection is approximately uniform. The potentials of uniform sampling for image compression/approximation was also confirmed in [153] although we confirmed that uniform sampling without replacement works better than the one with replacement. In general, uniform sampling works quite well for data with low coherence [54]. Our computer experiments also confirm this. Nonetheless, the length-squared requires a high pass cost over the data and has higher computational complexity for computing the length-probability distributions in (28).

Here, we compare the quality of the reconstructed images by the pass-efficient algorithms. For the STucker algorithm, we assumed that the number of fibers to be selected is the same and equal to 600. The compression ratio was 1.6287 with the corresponding relative error 0.0740. In Figure 20 (a), the reconstructed images (object and rotations) are displayed. For the LSS algorithm, we sample 238 lateral slices for which the compression ratio 1.6134 and the relative error 0.0443 were achieved. The quality of the images reconstructed by this algorithm are displayed in Figure 20 (b). Also for the TP-TCTA algorithm, we sampled 205 lateral slices and 205 horizontal slices for which the compression ratio 1.6526 and relative error 0.0799 were archived. The reconstructed images obtained by this algorithm are displayed in Figure 20(c). For the CTA-FS algorithm, we selected 400 slices and 2000 tubes using length-squared probability distribution (without replacement) for which the compression ratio 1.4874 and relative error 0.1651 were achieved.

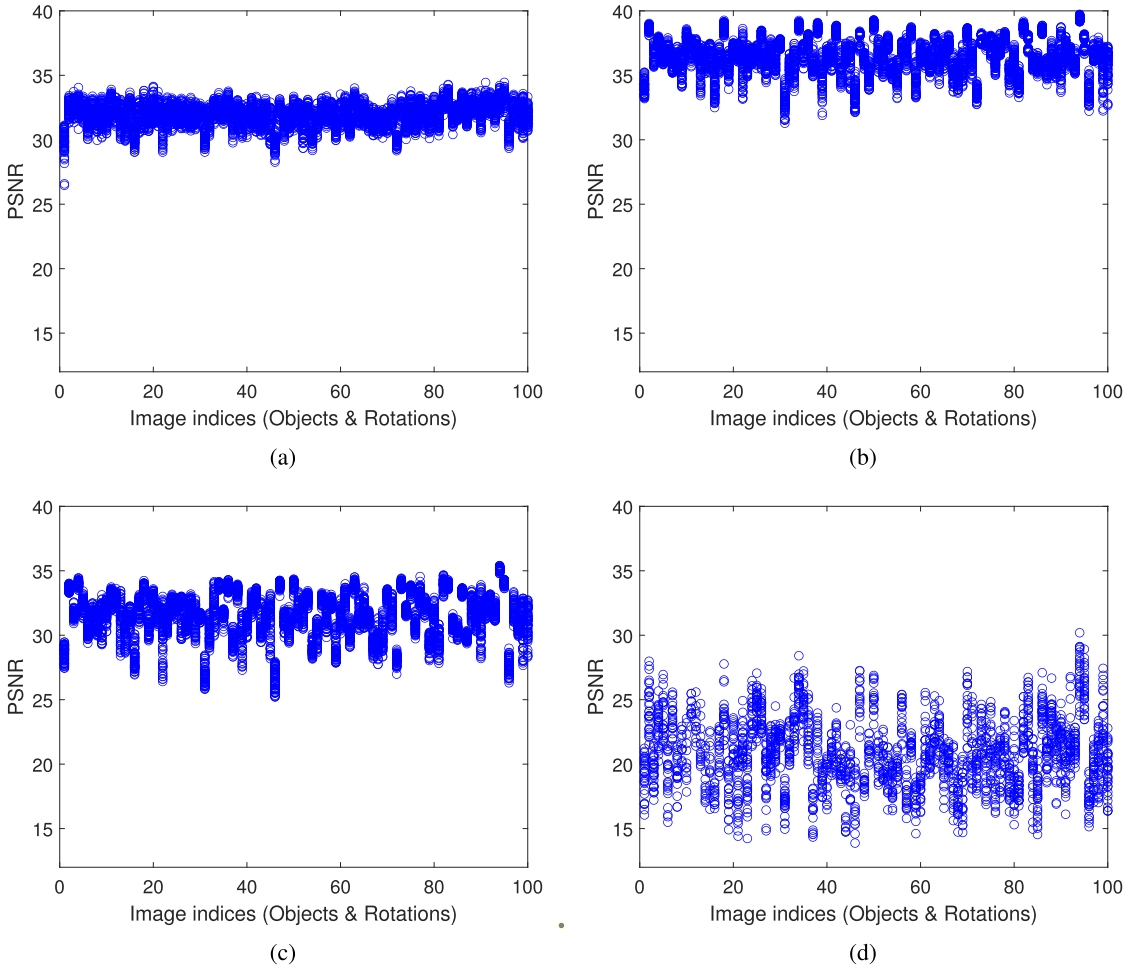

FIGURE 20. The quality of images reconstructed by the pass-efficient CTA algorithms, (a) STucker algorithm (compression ratio 1.6287 and relative error 0.0740), (b) LSS algorithm (compression ratio 1.6134 and relative error 0.0443) (c) TP-TCTA algorithm (compression ratio 1.6526 and relative error 0.0799). (d) CTA-FS algorithm (Length Squared) (compression ratio 1.4874 and relative error 0.1651).

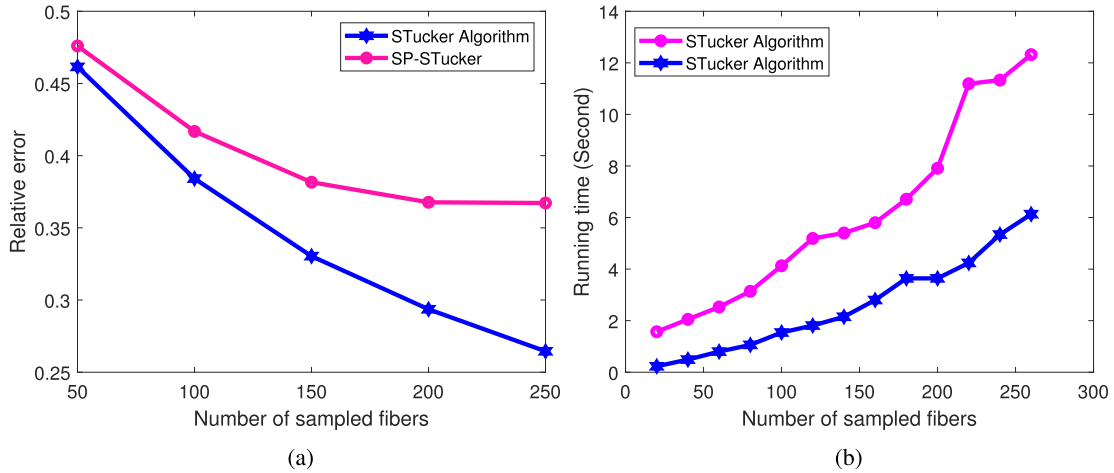

FIGURE 21. (a) Relative error comparison versus the number of sampled fibers for the SP-STucker and the STucker algorithms, (b) Running time comparison versus the number of sampled fibers for the SP-STucker and the STucker algorithms.

The reconstructed images of this experiment are displayed in Figure 20 (d). The recovered images of some samples corresponding to the above-mentioned cases are visualized in Figure 17.

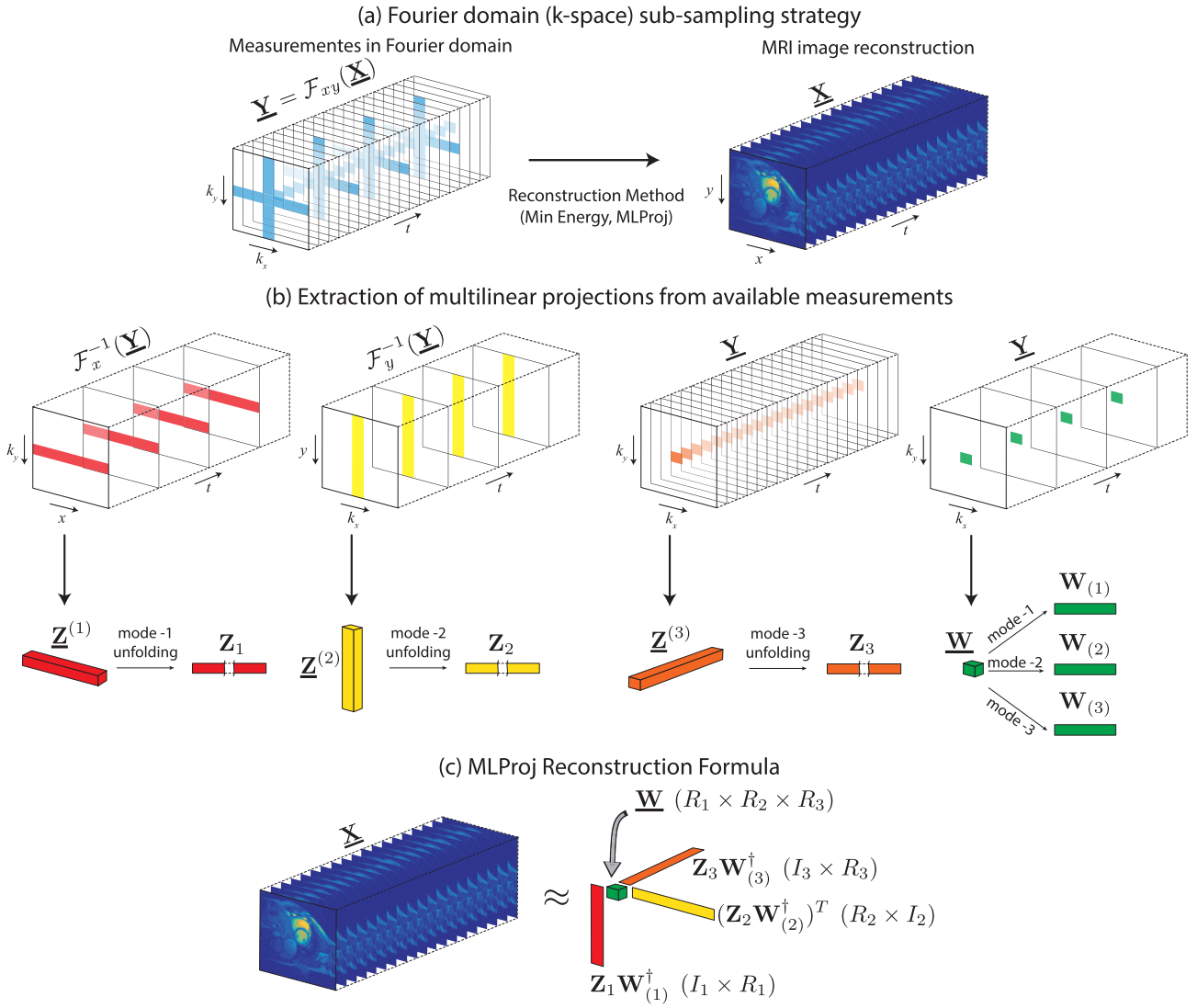

FIGURE 22. **Reconstruction of cine cardiac MRI from subsampling the Fourier domain throug the MultiLinar Projection (MLProj) method.**

Algorithm 2 (SP-STucker) is applicable only for relatively small multilinear rank because as we mentioned earlier, for theoretical reasons, the sketching parameters are required to satisfy $S_n > 2R_n$ which means that the Tucker rank $(R_1, R_2, \ldots, R_n)$ should be small enough. We assumed that the number of sampled fibers in each mode were the same and in the range, $R = 50, 100, \ldots, 250$, and the sketching parameters were $S_1 = S_2 = S_3 = 550$. We compared the relative error of the SP-STucker and the STucker algorithms for different numbers of sampled fibers in Figure 21 (a). From this figure, it is visible that as soon as the number of sampled fibers is increased, the quality of obtained results is significantly decreased. For completeness of our simulations, we compare the running time of SP-STucker and STucker algorithms. To this end, note that the algorithm SP-STucker is more efficient than the STucker algorithm in the following two main scenarios

- The data tensor is extremely large which burdens high communication costs.
- The low multilinear rank approximation of a tensor for several multilinear rank is required.

Since we were able to store the data in a single machine, we only considered the second above-mentioned scenario and made a comparison between the SP-STucker algorithm 2 and the STucker algorithm. To this end, we considered again the sketching parameters $S_1 = S_2 = S_3 = 500$ and the uniform multilinear rank (R, R, R) was used where $R = 40, 60, 80, \ldots, 260$. We utilized Gaussian random matrices with zero mean and variance 1 for computing the sketching tensor $\underline{\mathbf{W}}$ while they are not optimized for this computation. This operation was the most expensive part of the SP-STucker algorithm and took approximately 29 seconds. However, we do not compute it again through all of our computations for different multilinear ranks. The running time comparison

of these algorithms are reported in Figure 21 (b). Also the running times which are required to compute the multilinear rank approximation of the data tensor for 13 experiments with $R = 20, 40, 60, \ldots, 260$, based on the STucker and the SP-STucker algorithms were 79.34 and 62.61 seconds, respectively. The results clearly indicate that when multiple low multilinear rank approximation of a data is required, the SP-STucker algorithm 2 can be considerably faster than the STucker algorithm.

Example 4.2 Reconstruction of Cardiac MRI Images by Sub-Sampling in the Fourier Domain): Cardiac cine Magnetic Resonance Imaging (MRI) allows to obtain a temporal sequence of images within a single heart slice. At a given time, an MRI equipment provides measurements in the 2D-Fourier domain (k-space) of that single heart slice so the images are reconstructed by applying the inverse Fourier transform for each time. Thus, time resolution is constrained by the scanning of the full k-space. To improve time resolution, it is necessary to reduce the number of measurements and develop advanced processing techniques that could allow reconstructing images from incomplete measurements in the Fourier domain, which is the main motivation of the theory of Compressed Sensing (CS) developed in recent years [93], [94], [140]. CS techniques specialized in cine cardiac imaging were then developed by designing subsampling schemes using random Cartesian or radial trajectories in the (k_x, k_y)-space [154], [155].

Here, we show how to reconstruct cine cardiac MRI images by applying fixed masks to frontal slices in the data tensor as shown in Figure 22(a)-left. As a baseline method, we consider the Minimum Energy reconstruction, which consists in filling all unavailable measurements with zeros in the (k_x, k_y)-space and apply the inverse Fourier transform to each frontal slice. We can also apply the MLProj in eq. (12) by choosing projection matrices Φ_1 and Φ_2 as composed of the columns corresponding to low-frequencies in the Fourier transform matrix; and Φ_3 as composed by a subset of columns in the identity matrix, e.g. equally spaced. By doing so, it is easy to see that the multilinear projection tensors $\underline{\mathbf{Z}}^{(n)}, n = 1, 2, 3$ and $\underline{\mathbf{W}}$ of eqs. (11) and (8), respectively, can be computed from the available measurements as described in Fig. 22(b). Once the projection measurements are obtained, we can approximate the cine cardiac MRI by applying the reconstruction formula of eq. (12) as illustrated in Fig. 22(c). Since, the obtained reconstructed tensor has Tucker-rank (R, R, R_3), we also use a reference the best low Tucker-rank approximation which is obtained by applying the classical High Order Orthogonal Iteration (HOOI) algorithm [25] to the original tensor data.

In our experiments, we used a cardiac MRI data tensor $(256 \times 256 \times 25)$ as provided within the kt-FOCUSS demo.[9] In order to have a larger tensor, we have concatenated four replicates of this dataset along its 3rd mode thus, producing a $(256 \times 256 \times 100)$ data tensor. We computed the relative error with sampling rates in the range, $[5\%, 45\%]$,

[9]http://bispl.weebly.com/k-t-focuss.html

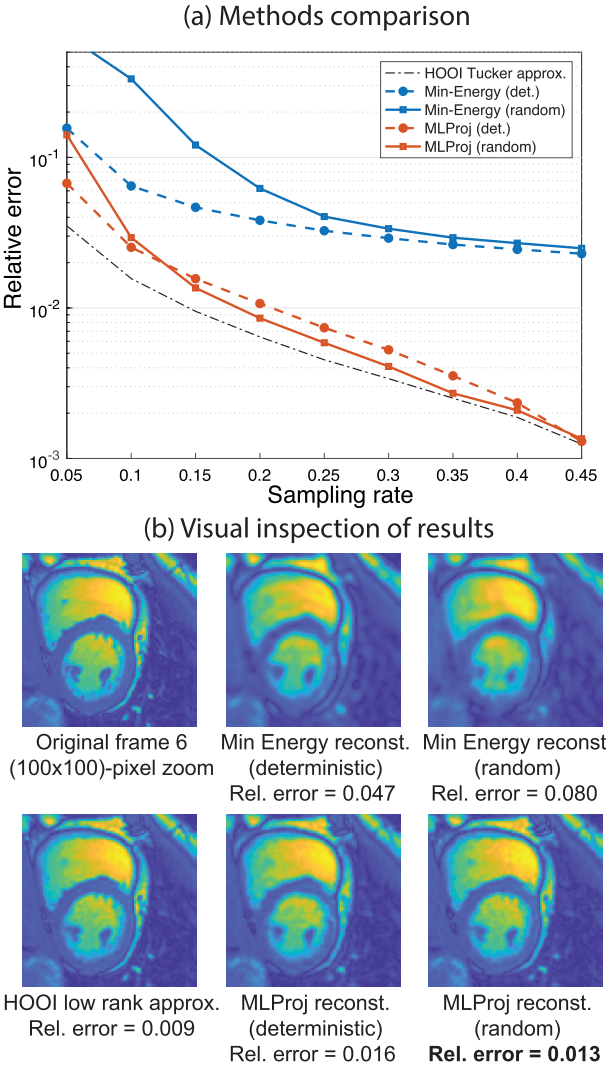

FIGURE 23. Simulation results on a real cine cardiac MRI data set.

for the Min. Energy and the MLProj reconstructions using deterministic and random masks. Deterministic masks were generated as shown in Fig. 22(a)-left, i.e., having a cross-type pattern every $D = 4$ frontal slices and a simple square covering center frequencies for the rest of the slices. We also randomly sampled indices around center frequencies using a double-side exponential law in order to evaluate random masks. The Tucker-rank of the obtained reconstruction is (R, R, R_3), where $R_3 = 25$ and R was computed according to the desired sampling rate. In Fig. 23(a) the obtained relative errors versus the sampling rate are shown for all the methods and the relative error of a Tucker approximation computed through the HOOI algorithm on the full original data tensor is also shown as a lower bound. It is remarkable that the MLProj method provides very excellent results which are close to the optimal low Tucker-rank decomposition. In Fig. 23(b), a visual comparison of the results obtained for a sampling rate equal to 15%, is shown. The best result was obtained with the MLPRoj (random) reconstruction method (relative

error = 1.3%), which provides images almost visually identical to the original ones. A remarkable characteristic of this method is that it is very fast as it is obtained by a simple closed formula (see Fig. 22(c))), which took only 0.3s to compute. We also applied a *state-of-the-art* CS method, namely the FOCUSS algorithm [155], using random Cartesian masks at a sampling rate also equal to 15% but using a different random mask for each slice in each data tensor. The obtained reconstruction was slightly better than MLProj (relative error = 0.6%) but visually almost indistinguishable. Such a good result is because in FOCUSS it is assumed that we are provided with random samples of every frontal slice of the tensor, which could be technologically more sophisticated. Moreover, as an iterative algorithm, FOCUSS has a considerably increased computational cost. It took about 124s to compute the reconstruction, which is about 3 orders of magnitude slower than the MLProj method.

X. CONCLUSION

In this paper, various deterministic and randomized algorithms for computation of Cross Tensor Approximations (CTAs) were reviewed and extended. Three possible generalizations of the matrix cross decomposition to tensors including: CTA based on fiber selection, slice-tube selection, and lateral-horizontal slices selection were discussed and reviewed. Interesting graphical illustrations are exploited to make the presentation much easier. Extensive simulations on image compression datasets were conducted to support and verify the validity and performance of some selected algorithms.

ACKNOWLEDGMENT

The authors thank three anonymous reviewers for their careful reading and constructive comments, which have significantly improved the quality of the paper.

REFERENCES

[1] F. L. Hitchcock, "Multiple invariants and generalized rank of a P-way matrix or tensor," *J. Math. Phys.*, vol. 7, nos. 1–4, pp. 39–79, Apr. 1928.
[2] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Stud. Appl. Math.*, vol. 6, nos. 1–4, pp. 164–189, 1927.
[3] L. R. Tucker, "Implications of factor analysis of three-way matrices for measurement of change," *Problems Measuring Change*, vol. 15, pp. 122–137, Jan. 1963.
[4] L. R. Tucker, "The extension of factor analysis to three-dimensional matrices," in *Contributions to Mathematical Psychology*, H. Gulliksen and N. Frederiksen, Eds. New York, NY, USA: Holt, Rinehardt, & Winston, 1964, pp. 110–127.
[5] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.
[6] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000.
[7] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms—Part II: Definitions and uniqueness," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1033–1066, Jan. 2008.
[8] L. De Lathauwer and D. Nion, "Decompositions of a higher-order tensor in block terms—Part III: Alternating least squares algorithms," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1067–1083, Jan. 2008.
[9] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms—Part I: Lemmas for partitioned matrices," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1022–1032, Jan. 2008.
[10] W. Hackbusch and S. Kühn, "A new scheme for the tensor representation," *J. Fourier Anal. Appl.*, vol. 15, no. 5, pp. 706–722, Oct. 2009.
[11] L. Grasedyck, "Hierarchical singular value decomposition of tensors," *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 4, pp. 2029–2054, 2010.
[12] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, Jan. 2011.
[13] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, "Tensor ring decomposition," 2016, *arXiv:1606.05535*.
[14] M. Espig, K. K. Naraparaju, and J. Schneider, "A note on tensor chain approximation," *Comput. Vis. Sci.*, vol. 15, no. 6, pp. 331–344, Dec. 2012.
[15] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra Appl.*, vol. 435, no. 3, pp. 641–658, 2011.
[16] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover, "Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 148–172, 2013.
[17] K. Braman, "Third-order tensors as linear operators on a space of matrices," *Linear Algebra Appl.*, vol. 433, no. 7, pp. 1241–1253, 2010.
[18] A. Cichocki, D. Mandic, H. A. Phan, C. Caiafa, G. Zhou, Q. Zhao, and L. De Lathauwer, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, Mar. 2015.
[19] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jan. 2017.
[20] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent Component Analysis and Applications*. New York, NY, USA: Academic, 2010.
[21] R. Leardi, "Multi-way analysis with applications in the chemical sciences, age Smilde, Rasmus bro and Paul Geladi, Wiley, Chichester, 2004, ISBN 0-471-98691-7, 381 pp," *J. Chemometrics*, vol. 19, no. 2, pp. 119–120, 2005.
[22] A. H. Phan and A. Cichocki, "Tensor decompositions for feature extraction and classification of high dimensional datasets," *IEICE Nonlinear Theory Appl.*, vol. 1, no. 1, pp. 37–68, 2010.
[23] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, and D. P. Mandic, "Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions," *Found. Trends Mach. Learn.*, vol. 9, nos. 4–5, pp. 249–429, 2016.
[24] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, and D. P. Mandic, "Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives," *Found. Trends Mach. Learn.*, vol. 9, no. 6, pp. 431–673, 2017.
[25] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.
[26] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, "Relative-error CUR matrix decompositions," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 844–881, 2008.
[27] F. G. Kuruvilla, P. J. Park, and S. L. Schreiber, "Vector algebra in the analysis of genome-wide expression data," *Genome Biol.*, vol. 3, no. 3, pp. 1–11, 2002.
[28] T. Yokota and A. Cichocki, "Multilinear tensor rank estimation via sparse tucker decomposition," in *Proc. Joint 7th Int. Conf. Soft Comput. Intell. Syst. (SCIS) 15th Int. Symp. Adv. Intell. Syst. (ISIS)*, Dec. 2014, pp. 478–483.
[29] A. Mai, L. Tran, L. Tran, and N. Trinh, "VGG deep neural network compression via SVD and CUR decomposition techniques," in *Proc. 7th NAFOSTED Conf. Inf. Comput. Sci. (NICS)*, Nov. 2020, pp. 118–123.
[30] E. P. Hendryx, B. M. Rivière, D. C. Sorensen, and C. G. Rusin, "Finding representative electrocardiogram beat morphologies with CUR," *J. Biomed. Informat.*, vol. 77, pp. 97–110, Jan. 2018.
[31] H. Cai, K. Hamm, L. Huang, J. Li, and T. Wang, "Rapid robust principal component analysis: CUR accelerated inexact low rank estimation," *IEEE Signal Process. Lett.*, vol. 28, pp. 116–120, Dec. 2021.
[32] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication," *SIAM J. Comput.*, vol. 36, no. 1, pp. 132–157, Jan. 2006.
[33] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix," *SIAM J. Comput.*, vol. 36, no. 1, pp. 158–183, 2006.
[34] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition," *SIAM J. Comput.*, vol. 36, no. 1, pp. 184–206, Jan. 2006.

[35] M. W. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *Proc. Nat. Acad. Sci. USA*, vol. 106, no. 3, pp. 697–702, 2009.
[36] C. Li, X. Wang, W. Dong, J. Yan, Q. Liu, and H. Zha, "Joint active learning with feature selection via CUR matrix decomposition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 6, pp. 1382–1396, Jun. 2019.
[37] A. Aldroubi, K. Hamm, A. B. Koku, and A. Sekmen, "CUR decompositions, similarity matrices, and subspace clustering," *Frontiers Appl. Math. Statist.*, vol. 4, p. 65, Jan. 2019.
[38] A. Frieze, R. Kannan, and S. Vempala, "Fast Monte-Carlo algorithms for finding low-rank approximations," *J. ACM*, vol. 51, no. 6, pp. 1025–1041, Nov. 2004.
[39] C. Boutsidis, M. W. Mahoney, and P. Drineas, "An improved approximation algorithm for the column subset selection problem," in *Proc. 20th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2009, pp. 968–977.
[40] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, "Near-optimal column-based matrix reconstruction," *SIAM J. Comput.*, vol. 43, no. 2, pp. 687–717, 2014.
[41] A. Deshpande and S. Vempala, "Adaptive sampling and fast low-rank matrix approximation," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Berlin, Germany: Springer, 2006, pp. 292–303.
[42] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang, "Matrix approximation and projective clustering via volume sampling," *Theory Comput.*, vol. 2, no. 1, pp. 225–247, 2006.
[43] A. Deshpande and L. Rademacher, "Efficient volume sampling for row/column subset selection," in *Proc. IEEE 51st Annu. Symp. Found. Comput. Sci.*, Oct. 2010, pp. 329–338.
[44] V. Guruswami and A. K. Sinop, "Optimal column-based low-rank matrix reconstruction," in *Proc. 23rd Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2012, pp. 1207–1214.
[45] M. Bebendorf, "Approximation of boundary element matrices," *Numerische Math.*, vol. 86, no. 4, pp. 565–589, 2000.
[46] B. Hashemi and L. N. Trefethen, "Chebfun in three dimensions," *SIAM J. Sci. Comput.*, vol. 39, no. 5, pp. C341–C363, Jan. 2017.
[47] S. Dolgov, D. Kressner, and C. Strössner, "Functional tucker approximation using Chebyshev interpolation," *SIAM J. Sci. Comput.*, vol. 43, no. 3, pp. A2190–A2210, Jan. 2021.
[48] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, "How to find a good submatrix," in *Matrix Methods: Theory, Algorithms and Applications: Dedicated to the Memory of Gene Golub*. Singapore: World Scientific, 2010, pp. 247–256.
[49] S. A. Goreinov and E. E. Tyrtyshnikov, "The maximal-volume concept in approximation by low-rank matrices," *Contemp. Math.*, vol. 280, pp. 47–52, 2001.
[50] D. Savostyanov, "Polilinear approximation of matrices and integral equations," (in Russian), Ph.D. dissertation, Dept. Math., INM RAS, Moscow, Russia, 2006.
[51] E. Tyrtyshnikov, "Incomplete cross approximation in the mosaic-skeleton method," *Computing*, vol. 64, no. 4, pp. 367–380, Jun. 2000.
[52] S. Chaturantabut and D. C. Sorensen, "Discrete empirical interpolation for nonlinear model reduction," in *Proc. 48h IEEE Conf. Decis. Control (CDC) Held Jointly With 28th Chin. Control Conf.*, Dec. 2009, pp. 4316–4321.
[53] D. C. Sorensen and M. Embree, "A DEIM induced CUR factorization," *SIAM J. Sci. Comput.*, vol. 38, no. 3, pp. A1454–A1482, Jan. 2016.
[54] M. W. Mahoney, "Randomized algorithms for matrices and data," *Found. Trends Mach. Learn.*, vol. 3, no. 2, pp. 123–224, 2011.
[55] S. Ahmadi-Asl, S. Abukhovich, M. G. Asante-Mensah, A. Cichocki, A. H. Phan, T. Tanaka, and I. Oseledets, "Randomized algorithms for computation of tucker decomposition and higher order SVD (HOSVD)," *IEEE Access*, vol. 9, pp. 28684–28706, 2021.
[56] S. Ahmadi-Asl, A. Cichocki, A. H. Phan, M. G. Asante-Mensah, M. M. Ghazani, T. Tanaka, and I. Oseledets, "Randomized algorithms for fast computation of low rank tensor ring model," *Mach. Learn., Sci. Technol.*, vol. 2, no. 1, Dec. 2020, Art. no. 011001.
[57] L. Ma and E. Solomonik, "Fast and accurate randomized algorithms for low-rank tensor decompositions," 2021, *arXiv:2104.01101*.
[58] M. W. Mahoney, M. Maggioni, and P. Drineas, "Tensor-CUR decompositions for tensor-based data," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 957–987, 2008.
[59] I. V. Oseledets, D. V. Savostianov, and E. E. Tyrtyshnikov, "Tucker dimensionality reduction of three-dimensional arrays in linear time," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 939–956, 2008.
[60] I. Oseledets and E. Tyrtyshnikov, "TT-cross approximation for multidimensional arrays," *Linear Algebra Appl.*, vol. 432, no. 1, pp. 70–88, 2010.
[61] C. F. Caiafa and A. Cichocki, "Generalizing the column–row matrix decomposition to multi-way arrays," *Linear Algebra Appl.*, vol. 433, no. 3, pp. 557–573, 2010.
[62] D. A. Tarzanagh and G. Michailidis, "Fast randomized algorithms for t-product based tensor operations and decompositions with applications to imaging data," *SIAM J. Imag. Sci.*, vol. 11, no. 4, pp. 2629–2664, Jan. 2018.
[63] H. Cai, K. Hamm, L. Huang, and D. Needell, "Mode-wise tensor decompositions: Multi-dimensional generalizations of CUR decompositions," 2021, *arXiv:2103.11037*.
[64] D. F. Gleich, C. Greif, and J. M. Varah, "The power and Arnoldi methods in an algebra of circulants," *Numer. Linear Algebra Appl.*, vol. 20, no. 5, pp. 809–831, 2013.
[65] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. (2016). *Tensorlab User Guide*. [Online]. Available: www.tensorlab.net
[66] T. G. Kolda and B. W. Bader. *MATLAB Tensor Toolbox*. Accessed: 2012. [Online]. Available: http://www.tensortoolbox.org/
[67] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, "A theory of pseudoskeleton approximations," *Linear Algebra Appl.*, vol. 261, nos. 1–3, pp. 1–21, Aug. 1997.
[68] K. Hamm and L. Huang, "Perspectives on CUR decompositions," *Appl. Comput. Harmon. Anal.*, vol. 48, no. 3, pp. 1088–1099, May 2020.
[69] D. Anderson, S. Du, M. Mahoney, C. Melgaard, K. Wu, and M. Gu, "Spectral gap error bounds for improving CUR matrix decomposition and the Nyström method," in *Proc. Mach. Learn. Res. (PMLR), Artif. Intell. Statist.*, San Diego, CA, USA, vol. 38, 2015, pp. 19–27. [Online]. Available: https://proceedings.mlr.press/v38/
[70] A. Y. Mikhalev and I. V. Oseledets, "Iterative representing set selection for nested cross approximation," *Numer. Linear Algebra With Appl.*, vol. 23, no. 2, pp. 230–248, Mar. 2016.
[71] M. V. Rakhuba and I. V. Oseledets, "Fast multidimensional convolution in low-rank tensor formats via cross approximation," *SIAM J. Sci. Comput.*, vol. 37, no. 2, pp. A565–A582, Jan. 2015.
[72] S. Wang and Z. Zhang, "Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling," *J. Mach. Learn. Res.*, vol. 14, pp. 2729–2769, Sep. 2013.
[73] R. Kannan and S. Vempala, "Randomized algorithms in numerical linear algebra," *Acta Numerica*, vol. 26, pp. 95–135, May 2017.
[74] D. P. Woodruff, "Sketching as a tool for numerical linear algebra," *Found. Trend Theor. Comput. Sci.*, vol. 10, no. 2, pp. 1–157, 2014.
[75] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, 2011.
[76] S. Chaturantabut and D. C. Sorensen, "Nonlinear model reduction via discrete empirical interpolation," *SIAM J. Sci. Comput.*, vol. 32, no. 5, pp. 2737–2764, 2010.
[77] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling techniques for the Nyström method method," in *Proc. Artif. Intell. Statist.*, 2009, pp. 304–311.
[78] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the Nyström method," *J. Mach. Learn. Res.*, vol. 13, pp. 981–1006, Apr. 2012.
[79] V. D. Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 721–728.
[80] C. K. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 682–688.
[81] P. Drineas and M. W. Mahoney, "A randomized algorithm for a tensor-based generalization of the singular value decomposition," *Linear Algebra Appl.*, vol. 420, nos. 2–3, pp. 553–571, Jan. 2007.
[82] E. Begovic, "Hybrid CUR-type decomposition of tensors in the tucker format," 2020, *arXiv:2002.01992*.
[83] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, "Practical sketching algorithms for low-rank matrix approximation," *SIAM J. Matrix Anal. Appl.*, vol. 38, no. 4, pp. 1454–1485, Jan. 2017.
[84] Y. Sun, Y. Guo, C. Luo, J. Tropp, and M. Udell, "Low-rank tucker approximation of a tensor from streaming data," 2019, *arXiv:1904.10951*.
[85] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 671–687, 2003.

[86] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2006, pp. 287–296.
[87] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, "A fast randomized algorithm for the approximation of matrices," *Appl. Comput. Harmon. Anal.*, vol. 25, no. 3, pp. 335–366, 2008.
[88] E. Liberty, "Accelerated dense random projections," Ph.D. dissertation, Dept. Comput. Sci., Yale Univ., New Haven, CT, USA, 2009.
[89] N. Ailon and E. Liberty, "Fast dimension reduction using Rademacher series on dual BCH codes," *Discrete Comput. Geometry*, vol. 42, no. 4, p. 615, Sep. 2008.
[90] G. W. Stewart, "Four algorithms for the the efficient computation of truncated pivoted QR approximations to a sparse matrix," *Numerische Math.*, vol. 83, no. 2, pp. 313–323, Aug. 1999.
[91] A. K. Saibaba, "HOID: Higher order interpolatory decomposition for tensors based on tucker representation," *SIAM J. Matrix Anal. Appl.*, vol. 37, no. 3, pp. 1223–1249, Jan. 2016.
[92] C. F. Caiafa and A. Cichocki, "Stable, robust, and super fast reconstruction of tensors using multi-way projections," *IEEE Trans. Signal Process.*, vol. 63, no. 3, pp. 780–793, Feb. 2015.
[93] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.
[94] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.
[95] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff, "Fast approximation of matrix coherence and statistical leverage," *J. Mach. Learn. Res.*, vol. 13, pp. 3475–3506, Dec. 2012.
[96] S. Goreinov, "On cross approximation of multi-index arrays," in *Doklady Mathematics*, vol. 77. Moscow, Russia: Springer, 2008, pp. 404–406.
[97] W. Hackbusch, D. Kressner, and A. Uschmajew, "Perturbation of higher-order singular values," *SIAM J. Appl. Algebra Geometry*, vol. 1, no. 1, pp. 374–387, Jan. 2017.
[98] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, "A new truncation strategy for the higher-order singular value decomposition," *SIAM J. Sci. Comput.*, vol. 34, no. 2, pp. A1027–A1052, Jan. 2012.
[99] H.-J. Flad, B. N. Khoromskij, D. V. Savostyanov, and E. E. Tyrtyshnikov, "Verification of the cross 3D algorithm on quantum chemistry data," *Russian J. Numer. Anal. Math. Model.*, vol. 23, no. 4, pp. 329–344, Jan. 2008.
[100] M. Espig, L. Grasedyck, and W. Hackbusch, "Black box low tensor-rank approximation using fiber-crosses," *Constructive Approximation*, vol. 30, no. 3, p. 557, 2009.
[101] J. Ballani, L. Grasedyck, and M. Kluge, "Black box approximation of tensors in hierarchical Tucker format," *Linear Algebra Appl.*, vol. 438, no. 2, pp. 639–657, Jan. 2013.
[102] D. J. Biagioni, D. Beylkin, and G. Beylkin, "Randomized interpolative decomposition of separated representations," *J. Comput. Phys.*, vol. 281, pp. 116–134, Jan. 2015.
[103] O. A. Malik and S. Becker, "Fast randomized matrix and tensor interpolative decomposition using CountSketch," *Adv. Comput. Math.*, vol. 46, no. 6, pp. 1–28, Dec. 2020.
[104] S. Friedland, V. Mehrmann, A. Miedlar, and M. Nkengla, "Fast low rank approximations of matrices and tensors," *Electron. J. Linear Algebra*, vol. 22, pp. 1031–1048, Jan. 2011.
[105] M. Gu and S. C. Eisenstat, "Efficient algorithms for computing a strong rank-revealing QR factorization," *SIAM J. Sci. Comput.*, vol. 17, no. 4, pp. 848–869, 1996.
[106] S. Voronin and P.-G. Martinsson, "Efficient algorithms for cur and interpolative matrix decompositions," *Adv. Comput. Math.*, vol. 43, no. 3, pp. 495–516, 2017.
[107] E. Newman, L. Horesh, H. Avron, and M. Kilmer, "Stable tensor neural networks for rapid deep learning," 2018, *arXiv:1811.06569*.
[108] E. Newman, "A step in the right dimension: Tensor algebra and applications," Ph.D. dissertation, Dept. Comput. Sci., Tufts Univ., Medford, MA, USA, 2019.
[109] Z. Zhang and S. Aeron, "Exact tensor completion using t-SVD," *IEEE Trans. Signal Process.*, vol. 65, no. 6, pp. 1511–1526, Mar. 2015.
[110] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer, "Novel methods for multilinear data completion and de-noising based on tensor-SVD," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3842–3849.
[111] K. Lund, "A new block Krylov subspace framework with applications to functions of matrices acting on multiple vectors," Ph.D. dissertation, Dept. Math., Universität Wuppertal, Fakultät Für Mathematik und Naturwissenschaften, Wuppertal, Germany, 2018.
[112] K. Lund, "The tensor t-function: A definition for functions of third-order tensors," *Numer. Linear Algebra with Appl.*, vol. 27, no. 3, May 2020, Art. no. e2288.
[113] Y. Miao, L. Qi, and Y. Wei, "Generalized tensor function via the tensor singular value decomposition based on the T-product," *Linear Algebra Appl.*, vol. 590, pp. 258–303, Apr. 2020.
[114] Y. Miao, L. Qi, and Y. Wei, "T-Jordan canonical form and t-Drazin inverse based on the t-product," *Commun. Appl. Math. Comput.*, vol. 3, pp. 201–220, Jan. 2020.
[115] S. Soltani, M. E. Kilmer, and P. C. Hansen, "A tensor-based dictionary learning approach to tomographic image reconstruction," *BIT Numer. Math.*, vol. 56, no. 4, pp. 1425–1454, 2016.
[116] C. D. Martin, R. Shafer, and B. LaRue, "An order-p tensor factorization with applications in imaging," *SIAM J. Sci. Comput.*, vol. 35, no. 1, pp. A474–A490, Jan. 2013.
[117] G. Song, M. K. Ng, and X. Zhang, "Robust tensor completion using transformed tensor singular value decomposition," *Numer. Linear Algebra with Appl.*, vol. 27, no. 3, May 2020, Art. no. e2299.
[118] L. Wang, K. Xie, T. Semong, and H. Zhou, "Missing data recovery based on tensor-CUR decomposition," *IEEE Access*, vol. 6, pp. 532–544, 2018.
[119] M. Xu, R. Jin, and Z.-H. Zhou, "CUR algorithm for partially observed matrices," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1412–1421.
[120] L. Qi and G. Yu, "T-singular values and T-sketching for third order tensors," 2021, *arXiv:2103.00976*.
[121] J. Zhang, A. K. Saibaba, M. E. Kilmer, and S. Aeron, "A randomized tensor singular value decomposition based on the t-product," *Numer. Linear Algebra With Appl.*, vol. 25, no. 5, Oct. 2018, Art. no. e2179.
[122] O. A. Malik and S. Becker, "Low-rank Tucker decomposition of large tensors using tensorsketch," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10117–10127.
[123] E. Frolov and I. Oseledets, "Tensor methods and recommender systems," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 7, no. 3, May 2017, Art. no. e1201.
[124] K. Xie, L. Wang, X. Wang, G. Xie, J. Wen, G. Zhang, J. Cao, and D. Zhang, "Accurate recovery of internet traffic data: A sequential tensor completion approach," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 793–806, Apr. 2018.
[125] Y. Li, K. Yu, and X. Wu, "Efficient tensor completion for Internet traffic data recovery," in *Proc. 2nd Int. Conf. Telecommun. Commun. Eng. (ICTCE)*, 2018, pp. 251–257.
[126] Z. Long, Y. Liu, L. Chen, and C. Zhu, "Low rank tensor completion for multiway visual data," *Signal Process.*, vol. 155, pp. 301–316, Feb. 2019.
[127] M. Fazel, "Matrix rank minimization with applications," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, 2002.
[128] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Rev.*, vol. 52, no. 3, pp. 471–501, 2010.
[129] Q. Song, H. Ge, J. Caverlee, and X. Hu, "Tensor completion algorithms in big data analytics," *ACM Trans. Knowl. Discovery From Data*, vol. 13, no. 1, pp. 1–48, Jan. 2019.
[130] T.-H. Oh, Y. Matsushita, Y.-W. Tai, and I. S. Kweon, "Fast randomized singular value thresholding for nuclear norm minimization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4484–4493.
[131] T.-H. Oh, Y. Matsushita, Y.-W. Tai, and I. S. Kweon, "Fast randomized singular value thresholding for low-rank optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 376–391, Feb. 2018.
[132] Y. Feng, G. Zhou, Y. Qiu, and W. Sun, "Orthogonal random projection based tensor completion for image recovery," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Nov. 2018, pp. 1350–1354.
[133] Y. Feng and G. Zhou, "Orthogonal random projection for tensor completion," *IET Comput. Vis.*, vol. 14, no. 5, pp. 233–240, Aug. 2020.
[134] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis with a new tensor nuclear norm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 925–938, Jan. 2020.

[135] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5249–5257.
[136] S. Liu and C. Zhang, "Randomized method for robust principal component analysis," in *Proc. 2nd Int. Conf. Comput. Sci. Appl. Eng. (CSAE)*, 2018, p. 19.
[137] Y. Li and W. Yu, "A fast implementation of singular value thresholding algorithm using recycling rank revealing randomized singular value decomposition," 2017, *arXiv:1704.05528*.
[138] H. Huang, Y. Liu, Z. Long, and C. Zhu, "Robust low-rank tensor ring completion," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 1117–1126, Jul. 2020.
[139] W. Gao and M. D. Sacchi, "Random noise attenuation via the randomized canonical polyadic decomposition," *Geophys. Prospecting*, vol. 68, no. 3, pp. 872–891, Mar. 2020.
[140] C. F. Caiafa and A. Cichocki, "Multidimensional compressed sensing and their applications," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 3, no. 6, pp. 355–380, Nov. 2013.
[141] H. Cai, K. Hamm, L. Huang, and D. Needell, "Robust CUR decomposition: Theory and imaging applications," 2021, *arXiv:2101.05231*.
[142] H. Cai, Z. Chao, L. Huang, and D. Needell, "Fast robust tensor principal component analysis via fiber cur decomposition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 189–197.
[143] J. Wen, L. Yang, and C. Shen, "Fast and robust compression of deep convolutional neural networks," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2020, pp. 52–63.
[144] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," 2014, *arXiv:1412.6553*.
[145] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," 2015, *arXiv:1511.06530*.
[146] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," 2015, *arXiv:1509.06569*.
[147] J. Håstad, "Tensor rank is NP-complete," *J. Algorithms*, vol. 11, no. 4, pp. 644–654, 1990.
[148] M. E. Timmerman and H. A. Kiers, "Three-mode principal components analysis: Choosing the numbers of components and sensitivity to local optima," *Brit. J. Math. Stat. Psychol.*, vol. 53, no. 1, pp. 1–16, 2000.
[149] E. Ceulemans and H. A. L. Kiers, "Selecting among three-mode principal component models of different types and complexities: A numerical convex hull based method," *Brit. J. Math. Stat. Psychol.*, vol. 59, no. 1, pp. 133–150, May 2006.
[150] M. Mørup and L. K. Hansen, "Automatic relevance determination for multi-way models," *J. Chemometrics*, vol. 23, nos. 7–8, pp. 352–363, 2009.
[151] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization," *SIAM J. Matrix Anal. A.*, vol. 28, no. 3, pp. 642–666, 2006.
[152] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-100)," Dept. Comput. Sci., Columbia Univ., New York, NY, USA, Tech. Rep. CUCS-006-96, 1996.
[153] E. Drinea, P. Drineas, and P. Huggins, "A randomized singular value decomposition algorithm for image processing applications," in *Proc. 8th Panhellenic Conf. Informat.* Princeton, NJ, USA: Citeseer, 2001, pp. 278–288.
[154] H. Jung, J. Park, J. Yoo, and J. C. Ye, "Radial k-t FOCUSS for high-resolution cardiac cine MRI," *Magn. Reson. Med.*, vol. 63, no. 1, pp. 68–78, Jan. 2010.
[155] H. Jung, K. Sung, K. S. Nayak, E. Y. Kim, and J. C. Ye, "K-t FOCUSS: A general compressed sensing framework for high resolution dynamic MRI," *Magn. Reson. Med.*, vol. 61, no. 1, pp. 103–116, Jan. 2009.