Cross Tensor Approximation for Image and Video Completion

Salman Ahmadi-Asl*[a], Maame Gyamfua Asante-Mensah[a], Andrzej Cichocki[a], Anh Huy Phan[a], Ivan Oseledets[a], Jun Wang[a]

[a]*Skolkovo Institute of Science and Technology (SKOLTECH), CDISE, Moscow, Russia,* *corresponding author: s.asl@skoltech.ru*

Abstract

This paper proposes a general framework to use the cross tensor approximation or tensor CUR approximation for reconstructing incomplete images and videos. The new algorithms are simple and easy to be implemented with low computational complexity. For the case of data tensors with 1) structural missing components or 2) a high missing rate, we propose an efficient smooth tensor CUR algorithms which first make the sampled fibers smooth and then apply the proposed CUR algorithms. The main contribution of this paper is to develop/investigate improved multistage CUR algorithms with filtering (smoothing) preprocessing for tensor completion. The second contribution is a detailed comparison of the performance of image recovery for four different CUR strategies via extensive computer simulations. Our simulations clearly indicated that the proposed algorithms are much faster than most of the existing state-of-the-art algorithms developed for tensor completion, while performance is comparable and often even better. Furthermore, we will provide in GitHub developed software in MATLAB which can be used for various applications. Moreover, to our best knowledge, the CUR (cross approximation) algorithms have not been investigated nor compared till now for image and video completion.

Keywords: Cross tensor approximation, tensor CUR approximation, tensor completion, image/video reconstruction and enhancement.

1. Introduction

Tensors are efficient tools for multi-way data processing (analysis) as they can preserve the intrinsic structures of the multidimensional data tensor, e.g., images and video [1, 2, 3, 4]. In many real applications, the underlying data tensors contain missing components or are corrupted by noise/outliers due to inaccurate data acquisition processes

or corruption by artifacts. The process of recovering (reconstructing) an incomplete data tensor from its partially observed data tensor is called tensor completion. In the past few decades, many algorithms have been developed to solve this problem [5, 6, 7] due to its importance in several applications such as in recommender systems [8], computer vision [9], chemometrics [10], link prediction [11], etc. Tensor completion algorithms are generally categorized into 1) rank minimization and 2) tensor decomposition techniques in which the concepts of tensor rank and tensor decomposition play key roles in the analysis. It is worth mentioning that in contrast to the matrices, the notion of the rank for tensors is not unique, and different tensor ranks can be defined. There are several types of tensor decompositions such as CANDECOMP/ PARAFAC decomposition [12, 13], Tucker decomposition [14, 15, 16] and its special case, i.e., Higher Order SVD (HOSVD) [17], Block Term decomposition [18, 19, 20], Tensor Train/Tensor Ring(Chain) (TT-TR(TC)) decomposition [21, 22, 23], tubal SVD (t-SVD) [24, 25, 26], each of which generalizes the notion of the matrix rank to tensors in efficient ways. Each iteration of the completion algorithms usually needs a low-rank tensor approximation of the underlying data tensors. The deterministic algorithms are prohibitive for these computations, especially when many iterations are required for convergence or the underlying incompleted data tensor is very large. These drawbacks motivate developing fast algorithms for the low-rank tensor approximation [27, 28].

The CUR or cross-skeleton algorithms have been introduced in the numerical multi-linear/linear algebra community mainly for fast low-rank matrix/tensor approximation of large-scale data matrices/tensors. The main motivations for utilizing the CUR algorithms are 1- fast low tensor rank approximation, 2- higher compression ratio, and 3- data interpretation issues. For example, in the case of matrices (second-order tensors), the SVD of sparse matrices does not provide sparse factor matrices while the cross-skeleton approximation achieves this goal, and this leads to more compact data representation. This paper intends to benefit from the first property of the CUR approaches and develop fast tensor completion algorithms.

Although the CUR algorithms have been extensively utilized for the low-rank matrix/tensor approximation and compression purposes, here we use them for the data completion task. Similar algorithms to matrix completion and tensor completion using the CUR approximation techniques have been studied in [29] and [30]. Our work differs from them in several aspects. First, our proposed algorithms are simple and easy to be implemented. Moreover, our work proposes a general framework that covers a variety of tensor/matrix CUR approximation techniques e.g., Tucker decomposition, tubal decomposition as special cases.

We should highlight that the main advantage of our proposed algorithms is their

simplicity, where they can be implemented in a few lines of code. They also have low computational complexity than other completion algorithms because of utilizing CUR algorithms.

Our main contributions in this paper include

- Proposing a general framework of tensor CUR approximation, e.g., Tucker decomposition, tubal decomposition, and tensor CUR approximation, for the tensor completion task. Besides, we extensively compare the performance of the proposed tensor CUR models in the simulation part.
- Proposing smooth tensor CUR completion algorithms for reconstructing incomplete data tensors with structural missing patterns or a high missing ratio. The quality of different smoothing techniques are also compared in our experiments.

This paper is organized as follows. In Section 2, preliminary concepts and definitions are given. Section 3 is devoted to introducing different types of matrix and tensor CUR algorithms. In Section 4, the tensor completion problem is described, and the proposed algorithm is discussed in Section 5. The computational complexity of the algorithms is studied in Section 6. In Section 7, extensive simulations are conducted to verify that the proposed algorithms are applicable and fast. Finally, the conclusion is given in Section 8.

2. Preliminary concepts and definitions

In this section, we present basic 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 [1]. The generalizations of these concepts to higher order tensors are straightforward. For example, for an Nth-order tensor, N types of fibers can be defined and they are referred to n-mode fibers for $n = 1, 2, \ldots, N$. Note that 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 norm and the Hadamard

[1]The notion of MP and transpose is also defined for tensors based on the t-product, see Appendix I.

(element-wise) product of tensors/matrices are denoted by $\|.\|_F$ and $\circledast$, respectively. We use the notation $\mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{X}})$ to denote the projection operator which keeps fixed the components of the tensor $\underline{\mathbf{X}}$ for the indices $\underline{\mathbf{\Omega}}$ and zeros-out the others. $\underline{\mathbf{\Omega}}^{\perp}$ denotes the complement of the set $\underline{\mathbf{\Omega}}$.

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 tensor $\underline{\mathbf{X}}$ is denoted by $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$, and is constructed by stacking all n-mode fibers.

The *mode-n product* is a generalization of matrix-matrix multiplication, defined for $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathbf{B} \in \mathbb{R}^{J \times I_n}$, as

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

for $j = 1, 2, \ldots, J$. Here, we have

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

Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be a given tensor, then the Tucker decomposition model is defined as follows: [14, 15, 16]

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

where $\underline{\mathbf{S}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ is the core tensor and $\mathbf{A}_n \in \mathbb{R}^{I_n \times R_n},\ R_n \leq I_n,\ n = 1, 2, \ldots, N$ are the factor matrices. 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 $\mathbf{X}_{(n)}$, $n = 1, 2, \ldots, N$.

In the next section, we introduce the Cross Matrix Approximation (CMA) and Cross Tensor Approximation (CTA) algorithms.

3. Cross Matrix Approximation (CMA) and Cross Tensor Approximation (CTA) methods

Computing a low-rank approximation of a given matrix based on a part of its individual columns and rows was first developed in [31] and is known as *skeleton or cross approximation*. Since the matrix intersecting the columns and rows is used within the approximation procedure, this framework is called cross approximation. The column or row selection can be performed in either a randomized or deterministic manner. The Maxvol-based algorithm [32, 33], Cross2D algorithm [34, 35] and Discrete Empirical

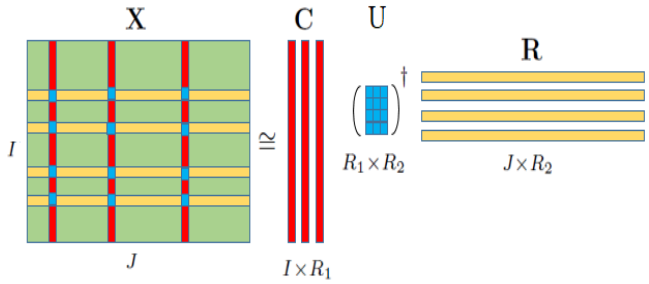


Figure 1: Illustration of cross decomposition for low-rank matrix approximation $\mathbf{X} \cong \mathbf{CUR}$ where $\mathbf{U} = \mathbf{W}^{+}$ [1].

Interpolatory Method (DEIM) [36, 37] are known algorithms for such selections. If the columns or rows are selected randomly, this framework is often known as randomized CUR approximation. Compared with conventional algorithms, e.g., SVD, this approach requires less memory usage and floating-point operations. It can also preserve the structure of the original data matrix, such as nonnegativity, sparsity, and smoothness.

The problem is formally formulated as follows. Let $\mathbf{X} \in \mathbb{R}^{I \times J}$ be a given matrix and $\mathbf{C} \in \mathbb{R}^{I \times R_1}$, $\mathbf{R} \in \mathbb{R}^{J \times R_2}$ are the selected columns and rows, respectively, and the intersection matrix is $\mathbf{W} \in \mathbb{R}^{R_1 \times R_2}$, see Figure 1. The low-rank cross approximation is computed as follows:

$$\mathbf{X} \cong \mathbf{CUR} = \mathbf{U} \times_1 \mathbf{C} \times_2 \mathbf{R}, \tag{2}$$

where $\mathbf{U} \in \mathbb{R}^{R_1 \times R_2}$ should be computed to yield the smallest error. The best middle matrix $\mathbf{U}$ in the least-squares sense is $\mathbf{U} = \mathbf{C}^{+}\mathbf{X}\mathbf{R}^{+}$ because

$$\mathbf{C}^{+}\mathbf{X}\mathbf{R}^{+} = \underset{\mathbf{U} \in \mathbb{R}^{R1 \times R2}}{\arg\min} \|\mathbf{X} - \mathbf{U} \times_1 \mathbf{C} \times_2 \mathbf{R}\|_F.$$

The approximation computed in the above formulation is exact if $\operatorname{rank}(\mathbf{X}) \leq \min\{R_1, R_2\}$ [31].

A faster way for computing the matrix $\mathbf{U}$ in (2) is using the Moore-Penrose of the intersection matrix $\mathbf{W}$ and computing the approximation $\mathbf{X} \cong \mathbf{C}\mathbf{W}^{+}\mathbf{R}$. It is known that the quality of the intersection matrix quite depends on the module of the determinant of the intersection matrix, which is called *matrix volume* and an intersection with as much volume as possible should be selected [32].

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 *CY decomposition*. This problem is known as column (feature) selection

in the field of machine learning and data analysis [38, 39, 40, 41, 42, 43, 44]. Let $\mathbf{X} \in \mathbb{R}^{I \times J}$ be a given data matrix and $\mathbf{C} \in \mathbb{R}^{I \times R}$ be the number of selected columns. Then the matrix column selection is formulated as follows

$$\mathbf{X} \cong \mathbf{C}\mathbf{Y}, \tag{3}$$

where $\mathbf{C} \in \mathbb{R}^{I \times R}$ is a matrix containing the selected columns and $\mathbf{Y} \in \mathbb{R}^{R \times J}$ should be computed in such a way that the approximate error should be as small as possible. The best solution to the problem (3) in the least-squares (LS) sense is $\mathbf{Y} = \mathbf{C}^{+}\mathbf{X}$, and if $\mathrm{rank}(\mathbf{X}) = R$, then the approximation is exact, i.e., $\mathbf{X} = \mathbf{C}\mathbf{C}^{+}\mathbf{X}$.

Cross Tensor Approximation (CTA) is a generalization of cross/skeleton matrix and CUR matrix approximation and is an efficient approach for fast low-rank tensor approximation. The *Skeleton* or *Cross approximation* or equivalently matrix CUR approximation (CMA) computes a low-rank approximation based on a part of individual columns and rows. It has found applications in deep learning [45], signal processing [46, 47], scientific computing [48, 49, 50] and machine learning [51, 52, 53]. In the next subsequent sections, we explain how the CMA can be generalized to tensors. In general, there are three main categories under which the CMA techniques are generalized to tensor as follows [54]:

- Fiber selection [55, 56, 57],
- Slice-tube selection [58],
- Slice selection [59].

The first category is concerned with the Tucker decomposition in which the factor matrices $\mathbf{A}^{(n)}$, $n = 1, 2, \ldots, N$ are computed by sampling the n-mode fibers of a given data tensor $\underline{\mathbf{X}}$ and the core tensor is computed in different ways, see Appendix II for the details. The essential differences between the algorithms proposed in [55, 56, 57] are 1) the number of fibers required for approximation and 2) the way that the core tensor is computed. For example, in [55], for a given Tucker rank $(R_1, R_2, \ldots, R_N)$, R_n numbers of mode-n fibers are selected to approximate the factor matrix $\mathbf{A}^{(n)}$ $n = 1, 2, \ldots, N$, and the core tensor is computed as follows

$$\underline{\mathbf{S}} = \underline{\mathbf{X}} \times_1 \mathbf{A}_1^{+} \times_2 \mathbf{A}_2^{+} \cdots \times_N \mathbf{A}_N^{+} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}, \tag{4}$$

while in the Fast Sampling Tucker Decomposition (FSTD) [57], the core tensor is constructed by sampling a part of the components of the tensor $\underline{\mathbf{X}}$.

In the second category, some slices are selected along with the fibers. Here, the following model is considered

$$\underline{\mathbf{X}} \cong \underline{\mathbf{C}} \times_3 (\mathbf{U}\mathbf{R})^T, \tag{5}$$

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 computed so that the approximation error obtained from model (5) be as small as possible.

The last category is associated with the tubal SVD. More precisely, the tubal cross approximation based on t-product is formulated as follows

$$\underline{\mathbf{X}} \cong \underline{\mathbf{C}} * \underline{\mathbf{U}} * \underline{\mathbf{R}}, \tag{6}$$

where $*$ stands for the t-product [24], $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times L_1 \times I_3}$ 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 (6) should be as small as possible. For the brevity of the presentation, we have provided the details of different tensor CUR approaches in Appendix II.

4. Tensor Completion

Tensor completion is the problem of completing (reconstructing) an incompleted data tensor from only a part of observed data components. It includes the matrix completion problem as a special case as a matrix is a second-order tensor. The completion algorithms are mainly categorized as 1) rank minimization and 2) tensor factorization techniques. The former is formally stated as follows:

$$\begin{aligned} \min_{\underline{\mathbf{X}}} \quad & \operatorname{Rank}(\underline{\mathbf{X}}) \\ \text{s.t.} \quad & \mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{X}}) = \mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{M}}), \end{aligned} \tag{7}$$

where $\underline{\mathbf{M}}$ is the incompleted data tensor with only observed components and their corresponding indices $\underline{\mathbf{\Omega}}$ and $\underline{\mathbf{X}}$ is the unknown data tensor that needs to be determined. The rank mentioned in formulations (7) can be tensor rank, TT (TC) rank, Tucker rank, etc, and the corresponding minimization problem is considered. For example in the case of Tucker rank which is the N-tuple $(R_1, R_2, \ldots, R_N)$, we have the following problem

$$\begin{aligned} \min_{\underline{\mathbf{X}}} \quad & \textstyle\sum_{n=1}^{N} R_n \\ \text{s.t.} \quad & \mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{X}}) = \mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{M}}), \end{aligned} \tag{8}$$

while for the TT model, the TT-rank [21] which is an $(N-1)$-tuple should be minimized. The rank minimization problem, in general, is an NP-hard problem because this problem for the matrices as a subset of tensors is NP hard [60]. So convex and tractable surrogates of the rank usually are replaced [6, 5]. The nuclear norm is the convex envelop of the rank and was extensively used in the literature but it is prohibitive especially for large-scale data due to the utilization of SVD [60]. The tensor decomposition formulation is a more efficient alternative to the nuclear norm minimization approach. The tensor decomposition variant of formulation (7) is presented as follows

$$\begin{aligned} \min_{\underline{\mathbf{X}}} \quad & \|\mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{X}}) - \mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{M}})\|_F^2, \\ \text{s.t.} \quad & \text{Rank}(\underline{\mathbf{X}}) = R, \end{aligned} \tag{9}$$

where the unknown tensor $\underline{\mathbf{X}}$ has low tensor rank representation. Here again, different kinds of tensor ranks and associated tensor decompositions can be considered. Using an auxiliary variable $\underline{\mathbf{C}}$, the optimization problem (9) can be solved more conveniently by the following reformulation

$$\begin{aligned} \min_{\underline{\mathbf{X}},\underline{\mathbf{C}}} \quad & \|\underline{\mathbf{X}} - \underline{\mathbf{C}}\|_F^2, \\ \text{s.t.} \quad & \text{Rank}(\underline{\mathbf{X}}) = R, \\ & \mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{C}}) = \mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{M}}) \end{aligned} \tag{10}$$

and we can alternatively solve optimization problem (9) over variables $\underline{\mathbf{X}}$ and $\underline{\mathbf{C}}$. Thus, the solution to the minimization problem (9) can be approximated by the following iterative procedure

$$\underline{\mathbf{X}}^{(n)} \leftarrow \mathcal{L}(\underline{\mathbf{C}}^{(n)}), \tag{11}$$

$$\underline{\mathbf{C}}^{(n+1)} \leftarrow \underline{\mathbf{\Omega}} \circledast \underline{\mathbf{M}} + (\underline{\mathbf{1}} - \underline{\mathbf{\Omega}}) \circledast \underline{\mathbf{X}}^{(n)}, \tag{12}$$

where $\mathcal{L}$ is an operator to compute a low-rank tensor approximation of the data tensor $\underline{\mathbf{X}}^{(n)}$ and $\underline{\mathbf{1}}$ is a tensor whose all components are equal to one. Note that Equation (11) solves the minimization problem (10) over $\underline{\mathbf{X}}$ for fixed variable $\underline{\mathbf{C}}$. Also Equation (12) solves the minimization problem (10) over $\underline{\mathbf{C}}$ for fixed variable $\underline{\mathbf{X}}$. The algorithm consists of two main steps, *low rank tensor approximation* (11) and *Masking computation* (12). It starts from the initial incomplete data tensor $\underline{\mathbf{X}}^{(0)}$ with the corresponding index set $\underline{\mathbf{\Omega}}$ and sequentially improves the approximate solution till some stopping criterion is satisfied or the maximum number of iterations is reached. Note that the term $\underline{\mathbf{\Omega}} \circledast \underline{\mathbf{M}}$ is not required to be computed at each iteration as it is equal to the initial data tensor $\underline{\mathbf{X}}^{(0)}$. In general, any type of tensor decomposition can be utilized for low-rank approximation

in (11). For example, in [61], the Tucker decomposition is exploited at each iteration while in [62] and [63] the CPD and the TR/TC decomposition are used. In the existing papers, usually the deterministic algorithms are utilized for computation of the tensor decomposition which may be expensive especially when a large number of iterations is required for convergence or the data tensor is quite large. In this paper, we propose to use the tensor CUR algorithms instead of the deterministic counterparts in which the actual fibers or slices are used for the low-rank approximation. A main feature of the paper is proposing a general framework to utilize the tensor CUR algorithm for the completion task. More precisely, all types of the tensor CUR algorithms including fiber sampling, slice-tube sampling or slice sampling can be exploited. It is experimentally shown that this can accelerate the basic completion algorithms significantly.

5. Proposed Approach

In this section, we describe our proposed algorithm for the tensor completion task. A key computation in our algorithm is the CUR approximation of the underlying data tensors. More precisely, at each iteration of our algorithm, a CUR approximation of the underlying data tensor is computed, after which the mask operator is applied. Indeed, the operator $\mathcal{L}$ in (11) is replaced by the tensor CUR algorithms in Section 3. The algorithm is initialized with a random data tensor with zero mean and unite variance and is sequentially updated to reconstruct the incompleted data tensor. We experimentally confirmed that the same results are achieved if the algorithm starts with the initial incomplete data tensor. Our proposed algorithm is summarized in Algorithm 1. This formulation is quite general and can incorporate different tensor/matrix factorization cases. For example, for the case of matrices, we have the matrix CUR algorithm which we only select some columns and rows ($\mathbf{C} \in \mathbb{R}^{I_1 \times R}$ and $\mathbf{R} \in \mathbb{R}^{R \times I_2}$) and compute the middle matrix as $\mathbf{U} = \mathbf{C}^{+}\mathbf{X}\mathbf{R}^{+} \in \mathbb{R}^{R \times R}$ and a low CUR approximation $\mathbf{X} \cong \mathbf{CUR}$ is computed.

Also, for the tensor case, different tensor decompositions can be utilized, such as Tucker decomposition [15, 16], tubal decomposition [24, 25], tensor CUR approximation [58]. To be more precise, let us consider the Tucker decomposition case. Here, in the first stage, the factor matrices $\mathbf{C}_n \in \mathbb{R}^{I_n \times R_n}$ are computed by sampling the columns of n-unfolding matrices (or n-mode fibers) after which the core tensor is computed as follows

$$\underline{\mathbf{S}} = \underline{\mathbf{X}} \times_1 \mathbf{C}_1^{+} \times_2 \mathbf{C}_2^{+} \cdots \times_N \mathbf{C}_N^{+} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}, \tag{13}$$

Then, a low CUR Tucker approximation is computed at each iteration as follows

$$\underline{\mathbf{X}} = \underline{\mathbf{S}} \times_1 \mathbf{C}_1 \times_2 \mathbf{C}_2 \cdots \times_N \mathbf{C}_N. \tag{14}$$

Of course, other kinds of CUR Tucker approximations such as FSTD/adaptive fiber sampling [57], Cross3D [56] or TT-Cross approximation [64] can be applied. In all our experiments, the sampling procedure is performed randomly, and as a result, the proposed algorithm can be considered a randomized tensor completion algorithm. It is also possible to exploit heuristic ones as it is a special case of our algorithm. However, they are slower than the randomized ones. In the case of images, we basically use the Tucker-2 model and only select columns/rows as the third mode is small and it is always considered as 3. In the case of videos, fibers in all modes are selected because the third mode which is the time (number of frames) is large.

The CY approximation can be incorporated in Algorithm 1. We have not considered it in our simulations as it is a special case of the CUR approximation, and there will not be a significant difference in the results. Moreover, our simulations show that for the Tucker CUR and the tubal CUR decompositions, the middle matrix should be computed very carefully and accurately otherwise, the approximation scheme will be unstable, and the results will be very poor. In all our experiments for the Tucker CUR and tubal CUR, we have utilized formulas (18) and (25), respectively.

5.1. Smoothing Techniques

The smoothing techniques, also known as *low-pass filtering* or *curve fitting* tools, are useful pre-processing approaches used in the machine learning community. In many applications, the elements of the underlying data tensors change smoothly (continuously), e.g., images, videos, time series. It is natural to consider the smoothness constraint when we are working on the optimization problems associated with the mentioned data tensors. There are various techniques and approaches to make the elements of a data tensor smooth such as moving average, locally estimated scatterplot smoothing (LOESS) and its weighed variant (LWOESS), the robust LOESS (RLOESS) and robust LWOESS (RLWOESS) etc., see for [65, 66, 67, 68], for more details. The moving average is the simplest type of smoothing procedure used widely in the signal community as finite impulse response (low-pass) filter. It smooths out the elements of given data, by replacing the elements of the data with a sequence of averages of different subsets of the given data points. To be more precise, assume that (x_1, y_1), $(x_2, y_2), \ldots, (x_N, y_N)$, are given data points, where x_i and y_i stand for independent and dependent variables, respectively. For the above data points, the so-called cumulative moving average gives a

new sequence $z_1, z_2, \ldots, z_N$ for the dependent variables y_i as follows

$$\begin{aligned} z_1 &= x_1, \\ z_1 &= \frac{x_1 + x_2}{2}, \\ &\vdots \\ z_N &= \frac{x_1 + x_2 + \ldots + x_N}{N}, \end{aligned} \tag{15}$$

while the simple moving average is defined as

$$z_n = \frac{1}{n} \sum_{i=N-n+1}^{N} y_i,$$

for $n = 1, 2, \ldots, N$. Other types of moving average include central, exponential and weighted moving average. The data points can also be made smooth by the regression strategy. In the regression approach, the goal is fitting the given data points by a function f, i.e., $y_i = f(x_i, \beta) + e_i$, $i = 1, 2, \ldots, N$, where e_i are the prediction error and β is the coefficients vector used in the data fitting. For example, the linear function $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$, $i = 1, 2, \ldots, N$ or the parabolic function $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$, $i = 1, 2, \ldots, N$ can be used in which the coefficient parameter β are $\beta = [\beta_0, \beta_1]$ and $[\beta_0, \beta_1, \beta_2]$, respectively. By minimizing the sum of the errors e_n, $n = 1, 2, \ldots, N$ and solving the following minimization problem

$$\min_{\beta} \sum_{n=1}^{N} e_i = \min_{\beta} \sum_{n=1}^{N} (y_n - f(x_n, \beta)),$$

the underlying coefficient vector β is computed. The above procedure fits the whole data points by a single line or a single polynomial. However it is more desirable to fit each part of data by a line or a polynomial separately to get more accurate approximations. This is called *local regression* and is a special case of the regression problem. The LOESS and the WLOESS are two examples of the local regression techniques. We will exploit a variety of smoothing techniques in our experiments and compare their performance for reconstructing data tensors.

As discussed in Section 3, the CMA methods can be generalized to tensors by sampling fibers, slices-tubes and slices. We proposed to first smooth out the selected fibers/slices and then compute the corresponding low CTA. We experimentally confirmed that this idea always provides better results than the algorithm without imposing smoothness. More importantly, in the situation that we have a high missing ratio of

elements or structural missing components, e.g., sequential columns and rows, the basic Algorithm 1 may not work properly while its smooth variant works perfectly.

In our extensive simulations, we found that for difficult incomplete data tensors, e.g., removing sequential columns and rows or a high missing ratio (95%), the tensor CUR algorithms may have a problem in reconstructing the incomplete data tensors. This is because the selected fibers of the incomplete data tensors have many zero components, and their components do not change continuously (smoothly) as expected in the case of images/videos. To overcome this difficulty, we proposed to first smooth out the selected fibers and then compute the CTA, see Figure 2 for the difference between the structure of a smooth signal and nonsmooth ones. We have tried different types of smoothing techniques in our computations, including moving average, LOESS, LWOESS, LOESS, RLOESS, RLWOESS, and Savitzky-Golay. In all our experiment we used the MATLAB command "smooth" can be utilized for the mentioned smoothing techniques. In fact, in the simulation part, we show that Algorithm 1 with smoothing the fibers provides better results than those without smoothing. As we will show in the simulations, this idea significantly improves the Tucker CUR algorithm while it requires almost the same running time. In the next section, we will explain the smoothness property and exiting approaches to perform it.

Algorithm 1: Tensor CUR algorithm for Nth-order tensor completion.

Input : An incomplete data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, Tensor Rank $\mathbf{R}$, the set of observed components $\underline{\mathbf{\Omega}}$, error bound ε and $\mathrm{MaxIter}$.

Output : Completed data tensor $\underline{\mathbf{X}}^*$

$\underline{\mathbf{X}}^{(0)} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the original data tensor with missing pixels;

$\underline{\mathbf{Y}}^{(0)} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a zero tensor;

for $n = 0, 1, 2, \ldots$ **do**

  $\underline{\mathbf{Y}}^{(n+1)} \leftarrow$Compute CUR approximation of the data tensor $\underline{\mathbf{X}}^{(n)}$ using selected fibers/slices and smoothing them,

  $\underline{\mathbf{X}}^{(n+1)} \leftarrow \mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{X}}^{(n)}) + \mathbf{P}_{\underline{\mathbf{\Omega}}^{\perp}}(\underline{\mathbf{Y}}^{(n+1)})$,

  if $\frac{\left\|\underline{\mathbf{X}}^{(n+1)} - \underline{\mathbf{X}}^{(n)}\right\|_F}{\left\|\underline{\mathbf{X}}^{(n+1)}\right\|_F} < \varepsilon$ *or* $n > \mathrm{MaxIter}$ **then**

    $\underline{\mathbf{X}}^* = \underline{\mathbf{X}}^{(n+1)}$ and **break**,

  end

end

6. Computational Complexity

In this section, we make a brief comparison between the complexity of the completion algorithms. Let $\in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. For the simplicity, assume $I_1 = I_2 = \ldots =$

Figure 2: Comparison of the structure of smooth fibers and nonsmooth in CUR algorithms. The results are for the column fiber $\underline{\mathbf{X}}(:, 50, 1)$ using the Tucker CUR and the iterations 1, 10, 20, 30, 40, (from the top to the bottom). Vertical axis represents the number of pixels, while the vertical axis represents the corresponding pixel intensity.

$I_N = I$ and for the Tucker rank $(R_1, R_2, \ldots, R_N)$, let $R_1 = R_2 = \ldots = R_N = R$. The computational complexities of the TR-ALS, and TR-WOPT per iteration are $\mathcal{O}(PNR^4I^N + NR^6)$, and $\mathcal{O}(2NI^NR^2)$, respectively, where P is the total number of observations. The computational complexity of Tucker CUR, Tubal CUR, FSTD and Slice-tube CUR are $\mathcal{O}(I^NR)$, $\mathcal{O}(I^{N-1}\log(I)R)$, $\mathcal{O}(IR^N)$ and $\mathcal{O}(I^NR)$, respectively, all of them are lower than the other completion algorithms. Our experimental results also confirm this. In our experiments, the TR-WOPT required more iterations for convergence and this is why in Figure 7, and 16, the TR-WOPT has high running time.

7. Simulations

In this section, we examine the proposed CUR algorithms on image/video completion. All numerical simulations were performed on a laptop computer with 2.60 GHz Intel(R) Core(TM) i7-5600U processor and 8GB memory. In all our experiments, the sampling of the fibers/slices are performed without replacement. We use Pick to Signal of Noise Ration (PSNR) and Structural SIMilarity (SSIM) to compare the performance of different completion algorithms. It is worth mentioning that it has been recently shown that the Hankelization completion algorithms can achieve high performance for recovering incomplete images with structured missing components, however we have not compared our algorithms with them as they are much slower and have higher computational complexity because the size and order of the data tensor increased in a pre-processing step (Hankelization step). Besides, since the running time of the tensor CUR algorithms are less than the other completion algorithms, we only compare the Tucker CUR with them as the baseline and the running time of tensor CUR algorithms are compared with themselves. We mainly use tubal CUR, Tucker CUR, FSTD, and slice-tube CUR algorithms and compare them with TR-WOPT [69], TR-ALS [63], and SPC [70], which are among the state-of-the-art algorithms for the tensor completion task. The codes are accessible at `https://github.com/SalmanAhmadi-Asl/Cross_Tensor_Completion`.

Image completion We first consider the image completion case. The benchmark images used in our simulations are "Baboon", "House", "Peppers", "Lena", "Facade" shown in Figure 3. All of them have size $256 \times 256 \times 3$. We considered different kinds of missing components (randomly and structured) shown in the second column of Figure 6 and the reconstructed images using different completion algorithms along with their corresponding PSNR and SSIM and running times are reported in Figure 6. In Table 1, detailed information regarding each of the TR-WOPT, TR-ALS, and SPC are mentioned. Since each image has only three slices and all of them are necessary to be selected, it

is not possible to apply Algorithm 6 directly. To this end, we reshaped the images to a 3rd order tensor of size $64 \times 64 \times 48$ and then selected some slices and tubes. In our algorithm for the Tucker CUR case, we considered Tucker rank $(70, 70, 3)$ in which we select 70 columns, 70 rows, and 3 channels. For the tubal CUR we considered 40 lateral and 40 horizontal slices, and for the slice-tube CUR, we considered 35 frontal slices and 2500 tubes.

In the first set of experiments we show the effect of smoothness for achieving better accuracy. To this end, we consider the "Lena" image and its degraded form as an image with incomplete pixels. We apply different CUR algorithms and their smooth variants on the degraded Lena image. The reconstructed images using different CUR completion algorithms are displayed in Figure 4. The corresponding PSNR/SSIM and execution time required to perform the completion is also reported. The results show that the smooth CUR algorithms always work better than ordinary CUR completion algorithms. Besides, we experimentally found that the smooth Tucker CUR algorithm was the most promising algorithm for reconstructing the Lena image. We conducted new experiments with more incomplete images and again compared the performance of different CUR completion algorithms. The results are reported in Figure 5. As can be seen we have used images with different missing patterns. All of them have structured missing expect the face for which we have removed 90% pixels. The results again show the superiority of the Tucker CUR algorithm over other CUR completion algorithms.

The smooth CUR Tucker completion algorithm which was the most efficient algorithm in our previous experiments is compared with the TR-WOPT, TR-ALS, and SPC algorithms. The results are reported in Figure 6. From Figure 6, it is seen the smooth Tucker CUR requires less time to complete the images with missing components while achieving roughly the same or even better performance compared to other completion algorithms. The running time of each completion algorithm for different incomplete images was almost them and because of this, we only report it for house image in Figure 7. As seen, the PSNR versus the CPU time of different completion algorithms are compared for the house image. From Figure 7, it is visible that the proposed CUR algorithms are more scalable for achieving higher performance that other completion algorithms.

In another simulation, we considered images with more difficult structural missing components where several sequential columns/rows are removed and seen in Figure 8 (Second column). The reconstructed images using the Tucker CUR and the smooth Tucker algorithms with the Tucker rank $(37, 37, 3)$ are shown in Figure 8 (c-d). The superiority of the smooth Tucker CUR over the Tucker CUR algorithm is visible. Recovering images with a high missing ratio is also a difficult and more challenging

Table 1: The selected parameters for different completion algorithms used in our experiments for images/videos completion.

Images		
TR-WOPT	TR-ALS	SPC
Size=$(256, 256, 3)$ TR-Rank=$(5, 5, 5)$ Max # Iteration= 100	Size=$(4, 4, 16, 4, 4, 16, 3)$ TR-Rank=$(10, 10, \ldots, 10)$, Max # Iteration= 10	Size=$(256, 256, 3)$ Type=Quadratic Max # Iteration=150
Videos		
TR-WOPT	TR-ALS	SPC
Size=$(176, 144, 30)$ TR-Rank=$(5, 5, 5)$ Max # Iteration= 100	Size=$(16, 4, 4, 6, 33, 15)$ TR-Rank=$(10, 10, \ldots, 10)$, Max # Iteration= 10	Size=$(176, 144, 30)$ Type=Quadratic Max # Iteration=150

Figure 3: Benchmark images used in our experiments.

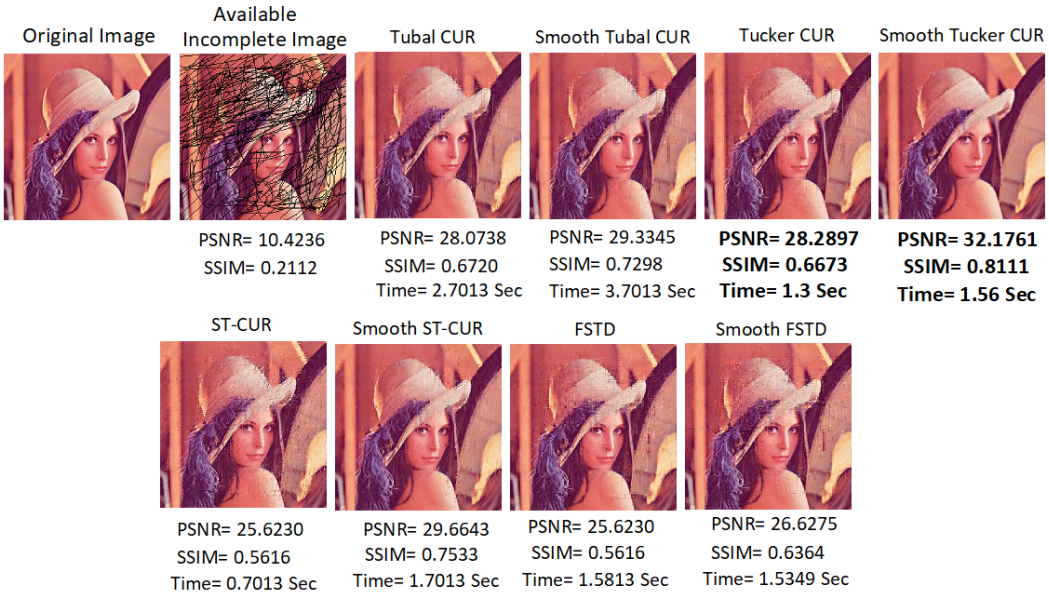


Figure 4: The reconstructed images using different tensor CUR completion algorithms.

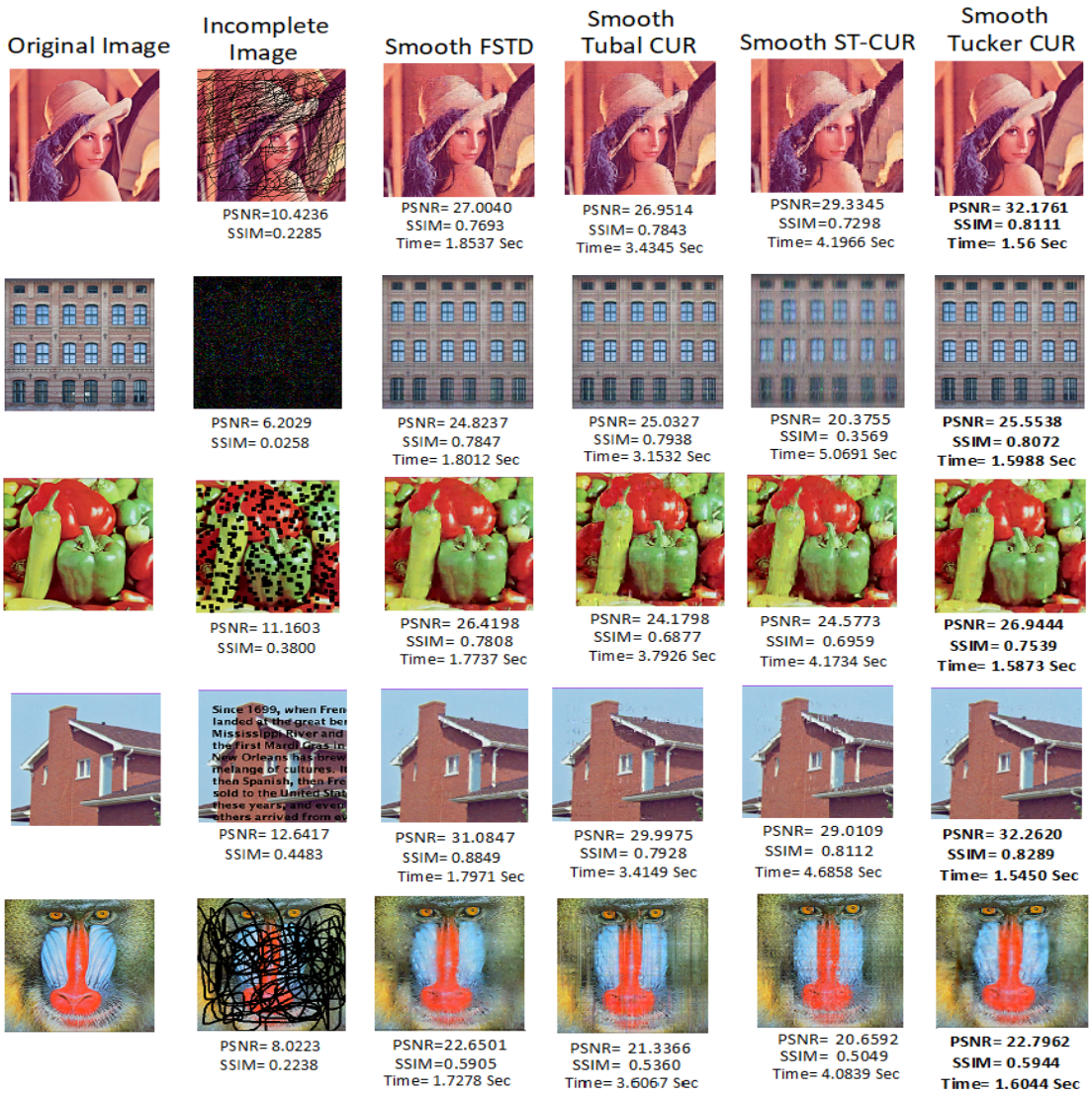


Figure 5: The reconstructed images using different tensor CUR completion algorithms.

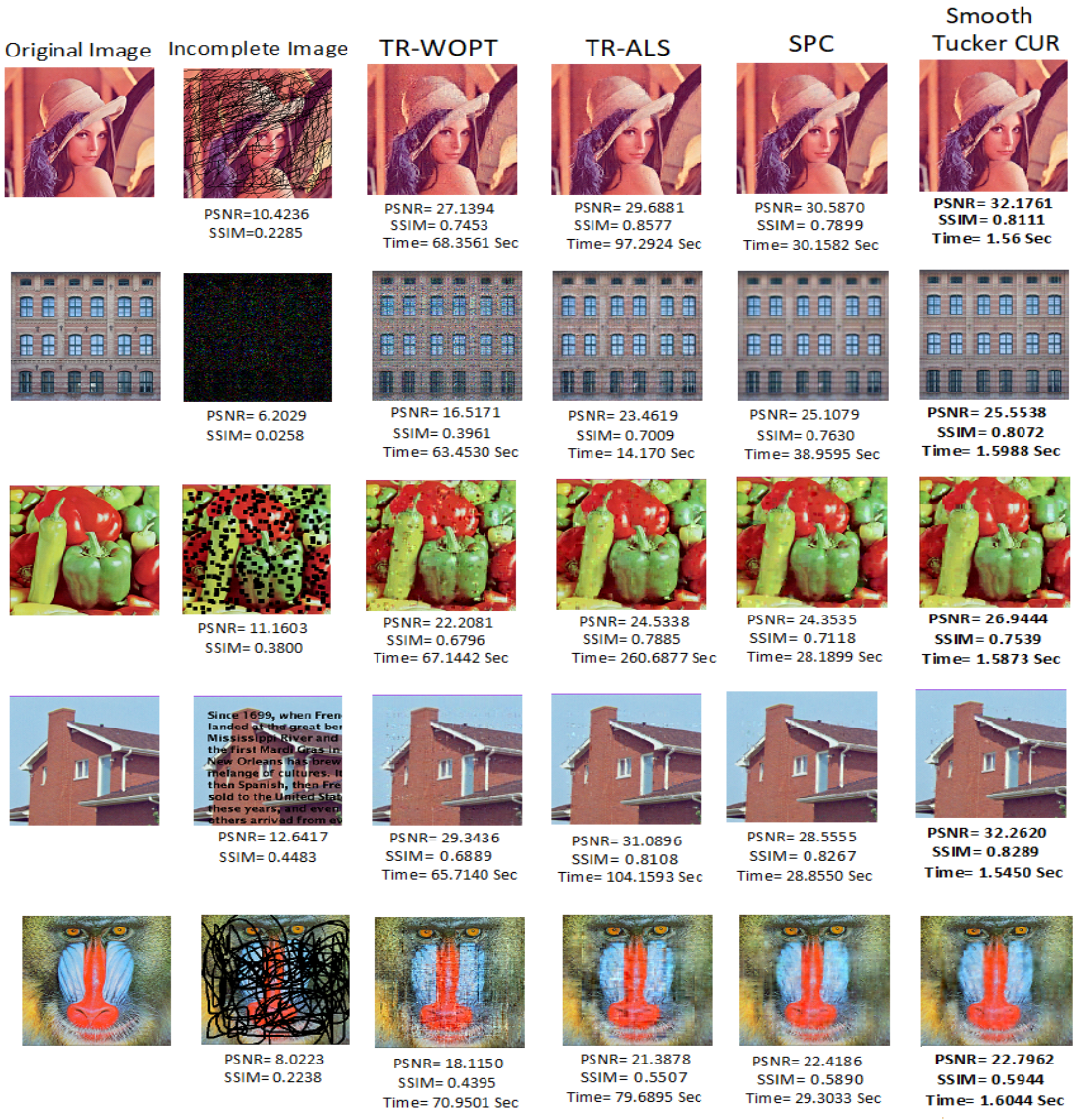


Figure 6: The reconstructed images using different tensor completion algorithms.

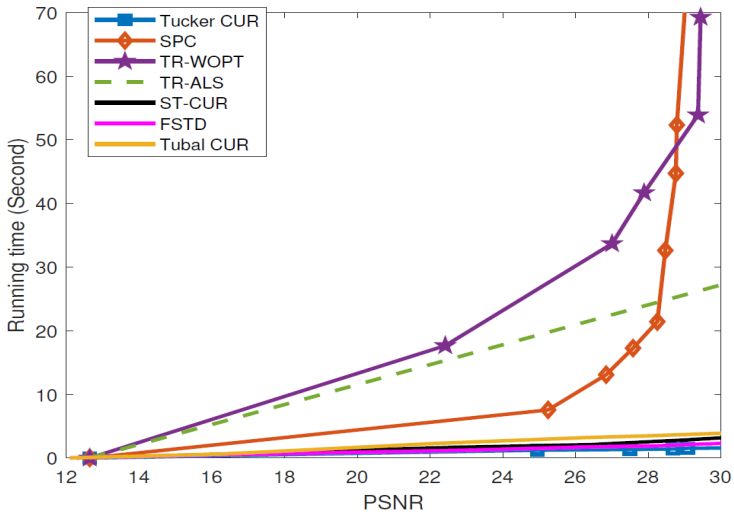


Figure 7: Comparing the performance and running times of different completion algorithms for the picture house.

Table 2: The PSNR and SSIM of the initial incomplete images and those achieved by the smooth Tucker CUR, smooth tubal CUR, and the smooth FSTD with corresponding running time (in second) for images with 95% missing components. The Tucker rank $(37, 37, 3)$ and tubal rank $R_1 = 25,\ R_2 = 25$ were used.

Image	Incomplete image [PSNR, SSIM]	Smooth FSTD [PSNR, SSIM, Time]	Smooth Tubal CUR [PSNR, SSIM, Time]	Smooth Tucker CUR [PSNR, SSIM, Time]
Lena	[5.3439, 0.0077]	[19.0626, 0.8614, 3.22]	[21.4871, 0.8919, 5.97]	[**22.1557**, **0.9011**, 4.11]
House	[4.8375, 0.0042]	[19.6538, 0.7140, 2.08]	[21.9302 0.7923, 5.26]	[**22.6334**, **0.8168**, 3.15]
Facade	[5.9693, 0.0045]	[20.0580, 0.4271, 3.22]	[22.3742, 0.6061, 4.23]	[**22.5591**, **0.6445**, 3.57]
Peppers	[6.1378, 0.0088]	[15.3699, 0.6983,2.76]	[20.0312, 0.8580, 4.620]	[**20.6701**, **0.8751**, 2.32]
Baboon	[5.5815, 0.0064]	[18.1769, 0.4988, 2.16]	[19.5935, 0.5624, 5.82]	[**19.8792**, **0.5763**, 3.38]

problem. We evaluated the performance of the Tucker CUR, the FSTD, the tubal CUR, for 95% missing components, we compared the reconstructed images of the CUR algorithms and their smooth variants using the Tucker rank $(37, 37, 3)$ and tubal rank $(25, 25)$ are reported in Figure 9 and Table 2. The results show that the proposed smooth Tucker CUR algorithm is able to recover images 95% with missing components within a few seconds. To the best of our knowledge our algorithms is the fastest algorithm to recover such difficult images.

We also conducted simulations with 99% missing pixels and applied Tucker, tubal CUR approximations and their smooth variants. In the case of Tucker CUR, we have used Tucker rank $(25, 25, 3)$ and for the tubal CUR, we have used tubal rank $(22, 22)$.

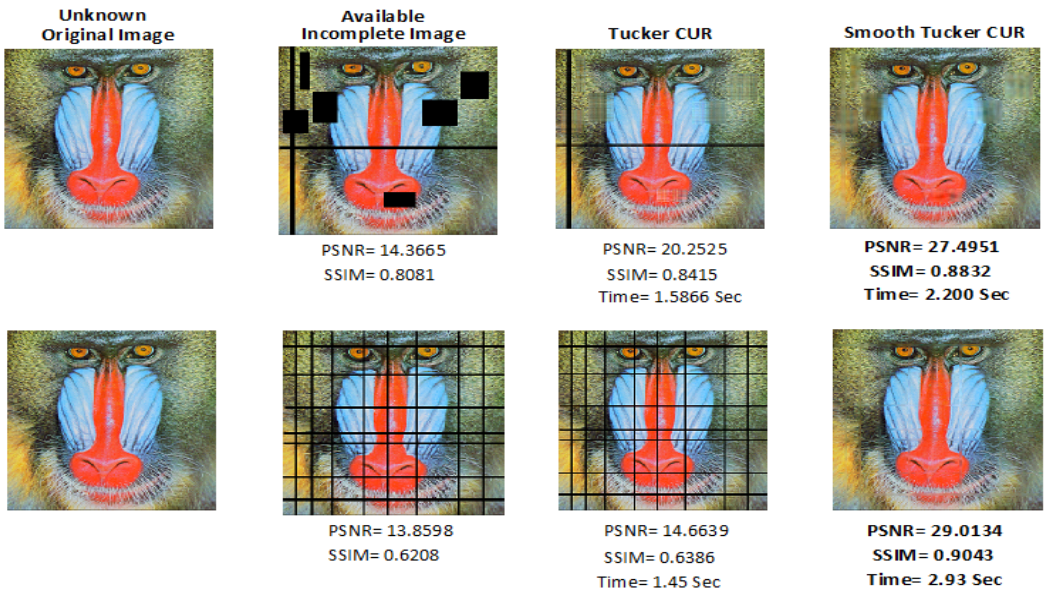


Figure 8: The reconstructed images using the Tucker CUR and the smooth Tucker CUR algorithm. The Tucker rank $(37, 37, 3)$ was used.

The reconstructed images and corresponding PSNR and SSIM are reported in Figure 10. The results again show the effectiveness of the idea of smoothness for recovering incomplete images with a high missing rate.

In Figures 11-12, for the "pepper" image, we consider different missing rates and compare the number of iterations against the achieved PSNR and SSIM using the Tucker and tubal decompositions, respectively.

The quality of different smoothing strategies is compared in Figures 13 and 14, for the Tucker CUR and the tubal CUR decompositions, respectively. The two images were used, the incompleted pepper in the third row of Figure 8 and the incompleted baboon in the last row of Figure 8. For the Tucker CUR, Tucker rank $(70, 70, 3)$ and for the tubal CUR, 40 lateral/horizontal slices were sampled.

For the Tucker decomposition and both images, the RLOWESS smoothing provided the best results while and the tubal decomposition, the moving average was the most promising smoother.

Video completion In the second experiment, we consider "Akiyo" and "News"

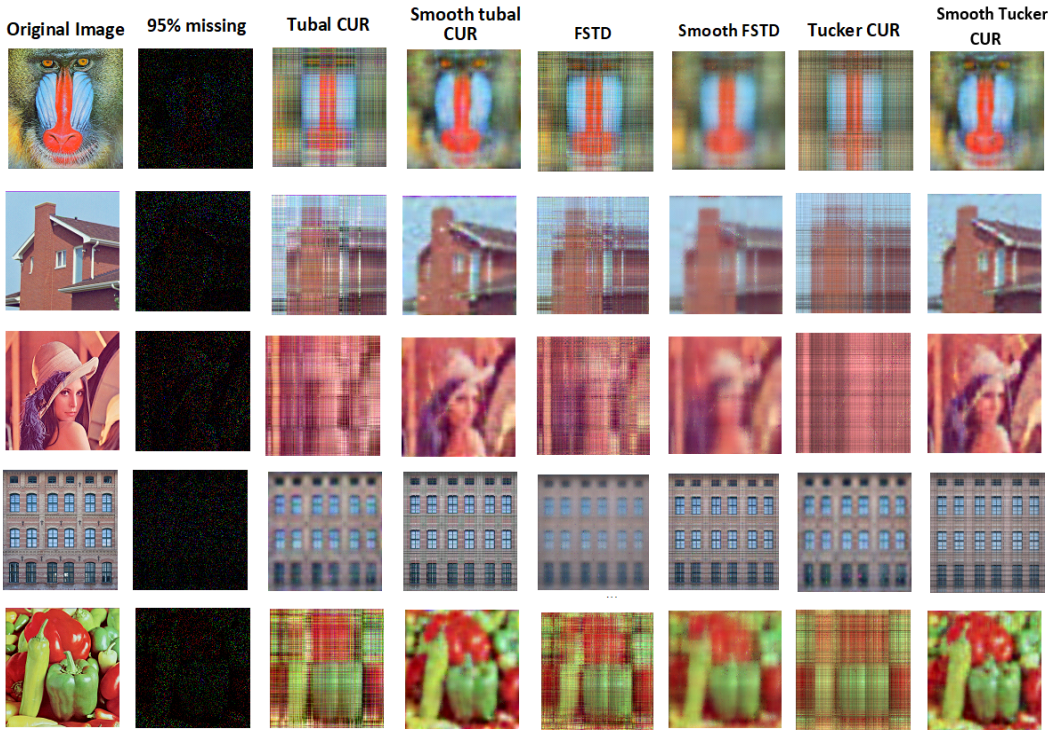


Figure 9: The reconstructed images using the Tucker CUR, the tubal CUR and their smooth variants for images with 95% missing pixels. The Tucker rank $(37, 37, 3)$ and tubal rank $R_1 = 25,\ R_2 = 25$ were used.

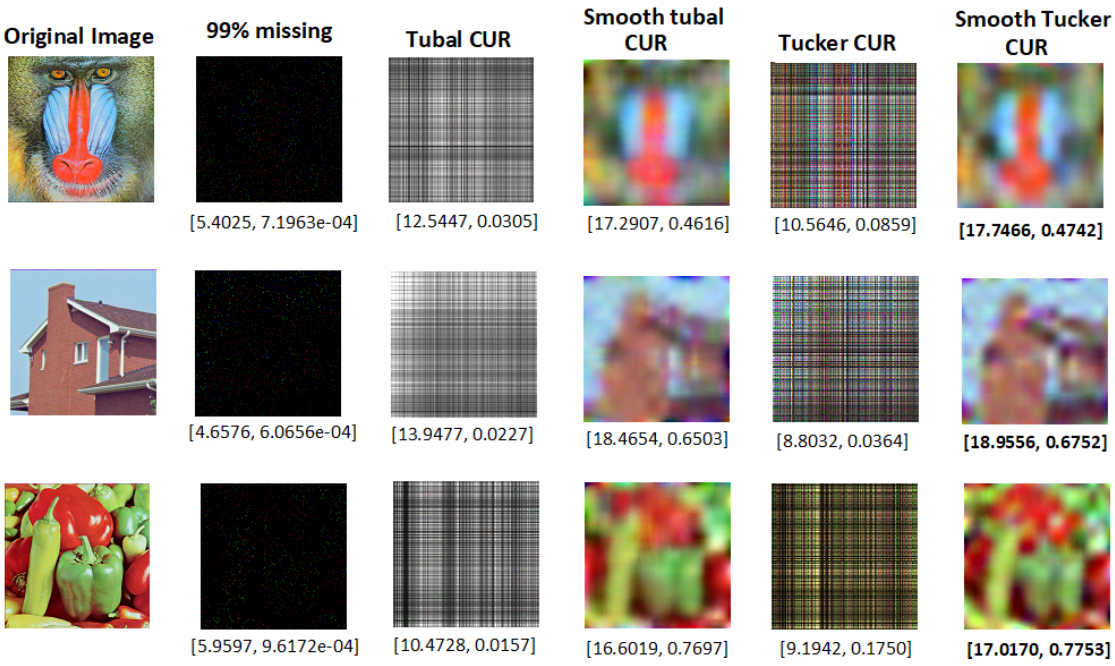
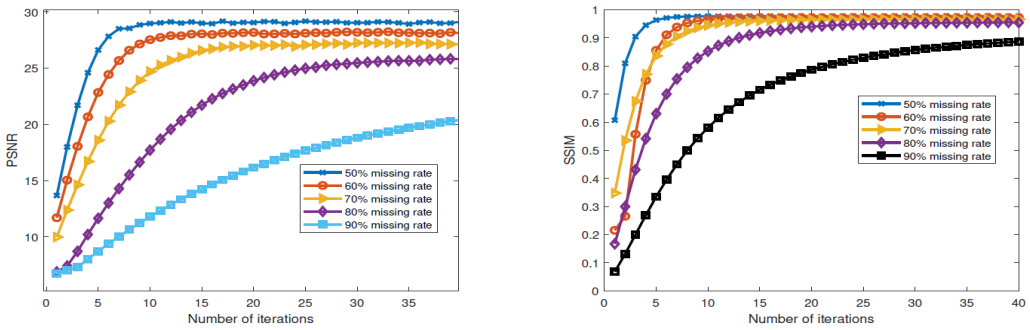
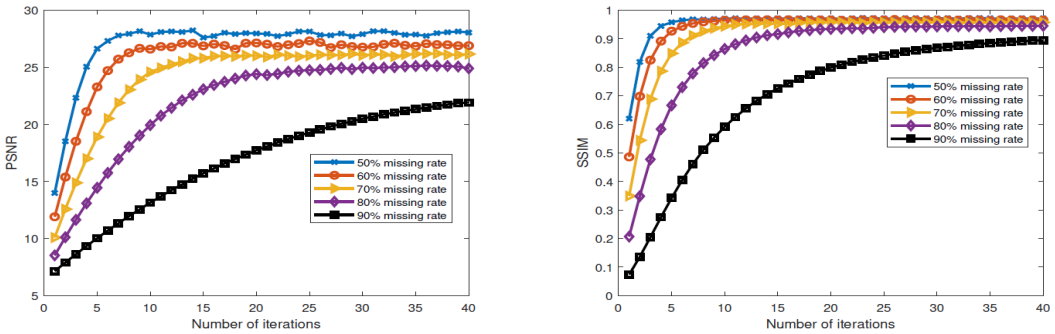


Figure 10: The reconstructed images using the Tucker CUR, the tubal CUR and their smooth variants for images with 99% missing pixels. The Tucker rank $(25, 25, 3)$ and tubal rank $R_1 = 22,\ R_2 = 22$ were used.



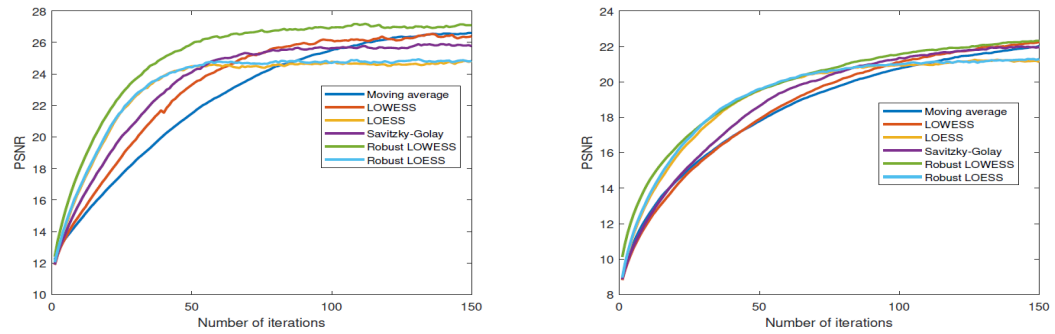
(a) (b)

Figure 11: The results of the smooth Tucker CUR for reconstructing the pepper image with different missing rates a) The PSNR comparison b) The SSIM comparison. The Tucker rank $(70, 70, 3)$ was used in the simulations.



(a) (b)

Figure 12: The results of the Smooth tubal CUR for the pepper image with different missing rates a) The PSNR comparison b) The SSIM comparison. The tubal rank $(50, 50)$ was used.



(a) (b)

Figure 13: Comparing the performance of different smoothing strategies using the Tucker CUR approximation for reconstructing images, a) incompleted pepper in third row of Figure 6 b) incompleted baboon in the last row of Figure 6. The Tucker rank $(70, 70, 3)$ was used.

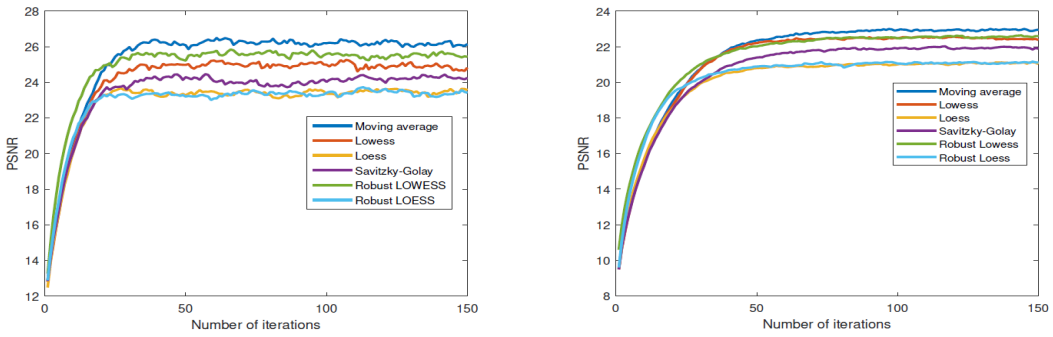


Figure 14: Comparing the performance of different smoothing strategies for reconstructing images using the tubal CUR approximation, a) incompleted pepper in Figure 6 b) incomplete baboon in Figure 6. 40 lateral/horizontal slices were used.

grayscale video datasets from[2] each of which is a 3rd order tensor of size $176 \times 144 \times 300$. In the next step, we remove randomly 70% pixels of the videos and apply the completion algorithms to complete them. Here similar to the images, the performance of algorithms are reported in two figures. For the Tucker CUR, we considered the Tucker rank $(90, 90, 120)$, for the tubal CUR we have selected 40 lateral and 40 horizontal slices and for the Slice-tube CUR algorithm, we have selected 80 frontal slices and 1500 tubes.

The PSNR achieved by different completion algorithms are reported in Figure 15 (a)-(b) for "Foreman" and "news" Videos, respectively. The ST-CUR algorithm outperformed other completion for both video datasets, while other CUR completion algorithms almost worked well. Besides, the running time of all completion algorithms for recovering the Akiyo video with 70% missing components are reported in Figure 16 (a)-(b). The results show that the Tucker CUR algorithms need less running time for recovering the mentioned video with almost the same or even better performance compared to the other completion algorithms. The results confirmed that the proposed framework is very efficient for reconstructing incomplete video datasets.

Yale and ORAL datasets. In the previous examples, the superiority of our CUR framework was visible. Now, in the rest of our experiments, we only consider the Tucker CUR algorithm, which almost was the cheapest and more efficient algorithm for the tensor completion task compared to the other tensor CUR algorithms. In the

[2] `http://trace.eas.asu.edu/yuv/`

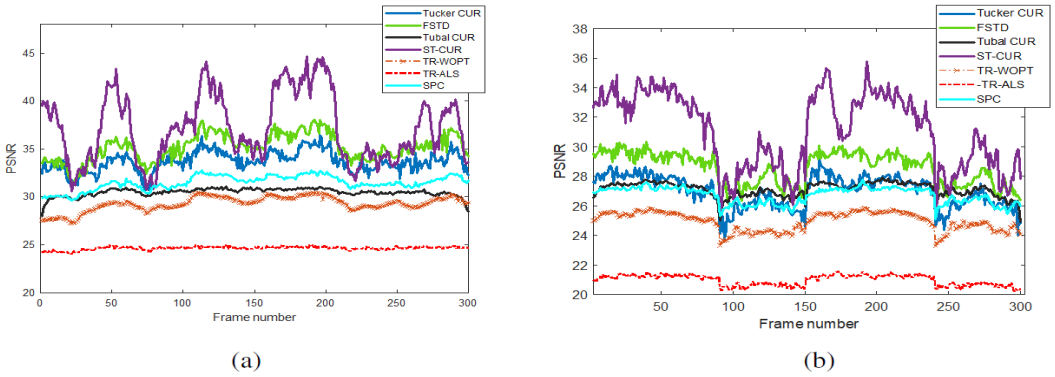


Figure 15: The PSNR comparison of different completion algorithms for a) Foreman video dataset and b) News video dataset.

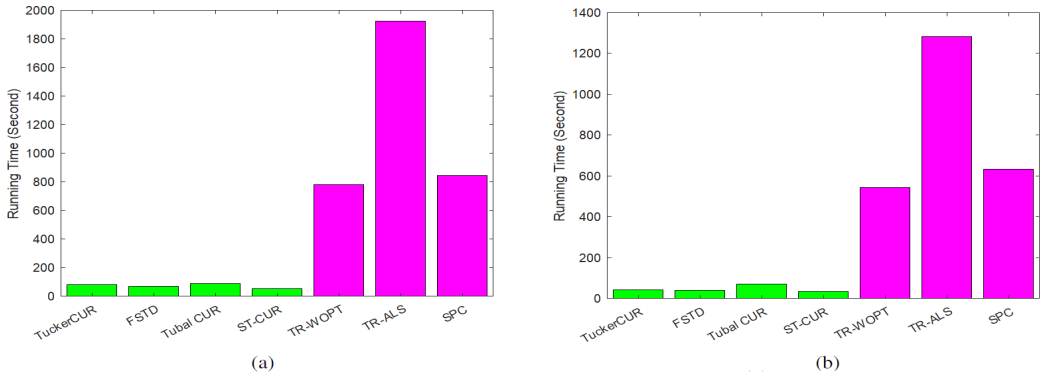


Figure 16: The running time comparison of different completion algorithms for a) Foreman video dataset and b) News video dataset.

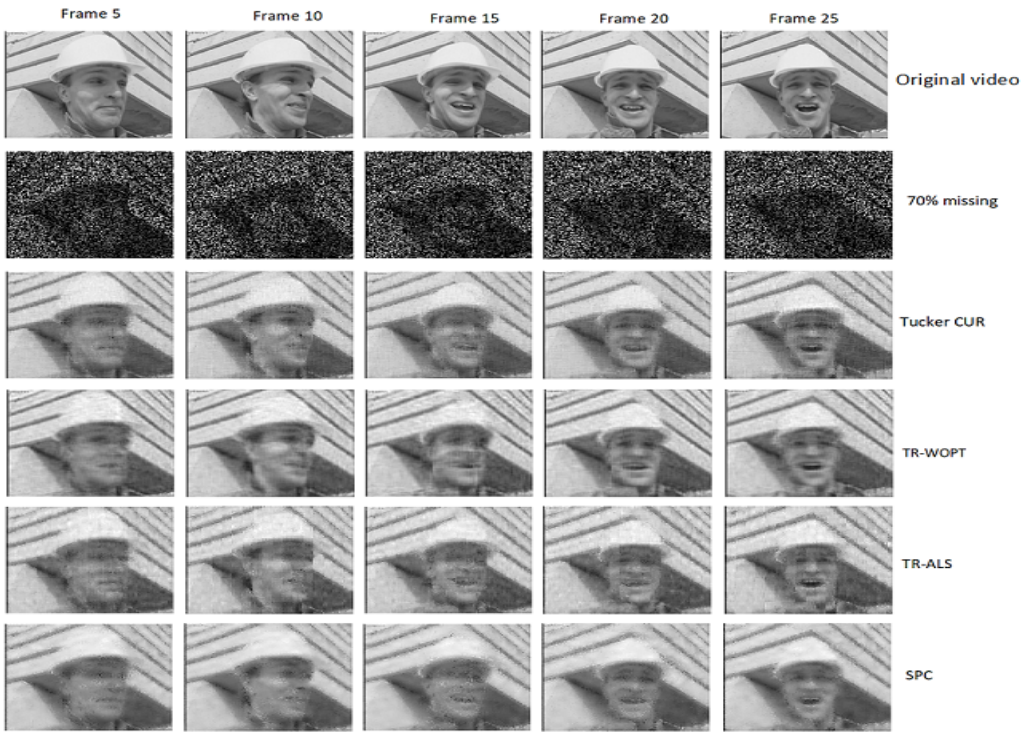


Figure 17: The reconstructed frames of the Foreman video dataset with 70% missing pixels using different completion algorithms.

Figure 18: The reconstructed frames of the News video dataset using different completion algorithms.

third experiment, we considered Yale and ORL datasets[3] which have been extensively utilized for the face recognition task [71, 72, 73, 74]. The Yale and ORL detests are a collection of images of size 64×64. For the Yale dataset, there are 15 persons under 11 different expressions e.g., sad, sleepy, surprised, and as a result a fourth order tensor of size $11 \times 15 \times 64 \times 64$ is produced while for the ORL dataset, 40 distinct subjects for each subject ten different images exist, and a fourth order dataset of size $10 \times 40 \times 64 \times 64$ is generated. We uniformly remove 70% pixels of both mentioned datasets and apply the Tucker CUR algorithms to reconstruct the whole datasets. For the Yale dataset we considered $R_1 = 11, R_2 = 15, R_3 = 50, R_4 = 50$ and for the ORL we used $R_1 = 10, R_2 = 40, R_3 = 45, R_4 = 45$. We first report the accuracy of some of the reconstructed images in Figure 19. Besides, in Figures 20 and 21, more reconstructed images are displayed. The results show the efficiency and applicability of the proposed algorithm for the case we want to reconstruct a set of images instead of only one single image.

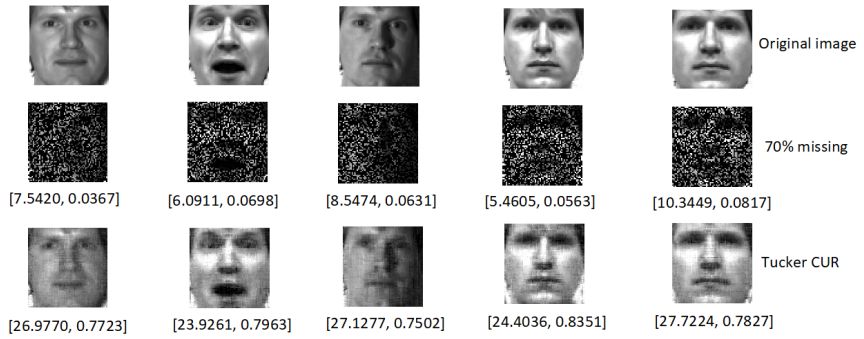
MRI data set. In the fourth experiment, we show that our proposed Tucker CUR algorithm is also applicable for medical images. To this end, we applied the Tucker CUR completion algorithm developed by us for the MRI dataset. The original dataset is a 3rd order tensor of size $256 \times 256 \times 21$, see Figure 22 for some sample slices of the data tensor. We uniformly removed 40% of the pixels of the original data tensor and We applied the Tucker CUR algorithm with $R_1 = R_2 = 160,\ R_3 = 21$. The reconstructed images (also images with missing pixels) and their corresponding PSNR and SSIM are reported in Figure 22. We also apply smoothing the fibers in the first and second modes and the PSNR and SSIM comparisons are reported in Figure 23.

The results show that the proposed algorithm, especially the algorithm equipped with smoothing idea is a promising approach for completing the medical images.

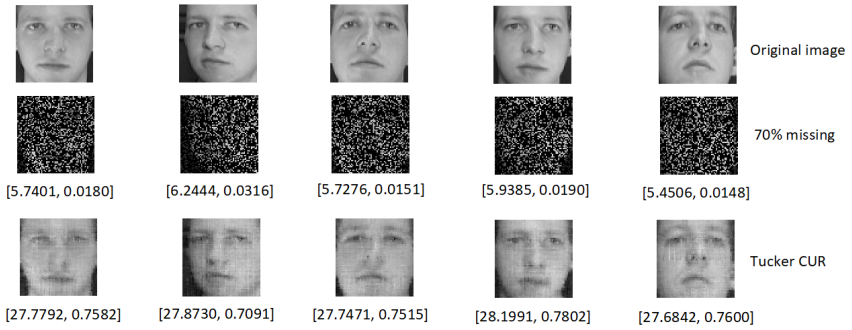
8. Conclusion

In this paper we proposed a general framework of tensor CUR algorithms for the tensor completion task. The smooth variants of the suggested CUR completion algorithms were also proposed to improve the results. We experimentally showed that the smooth algorithms tackle the problem of reconstructing data tensor with a high missing ratio. The proposed algorithms are simple and easy to be implemented only in a few lines of code. The algorithms have low computational complexity because of using CUR approximation within them instead of more expensive ones. The efficiency

[3] http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html



(a)



(b)

Figure 19: The reconstructed images using Tucker CUR algorithm for the a) Yale dataset b) ORL dataset with 70% missing components.

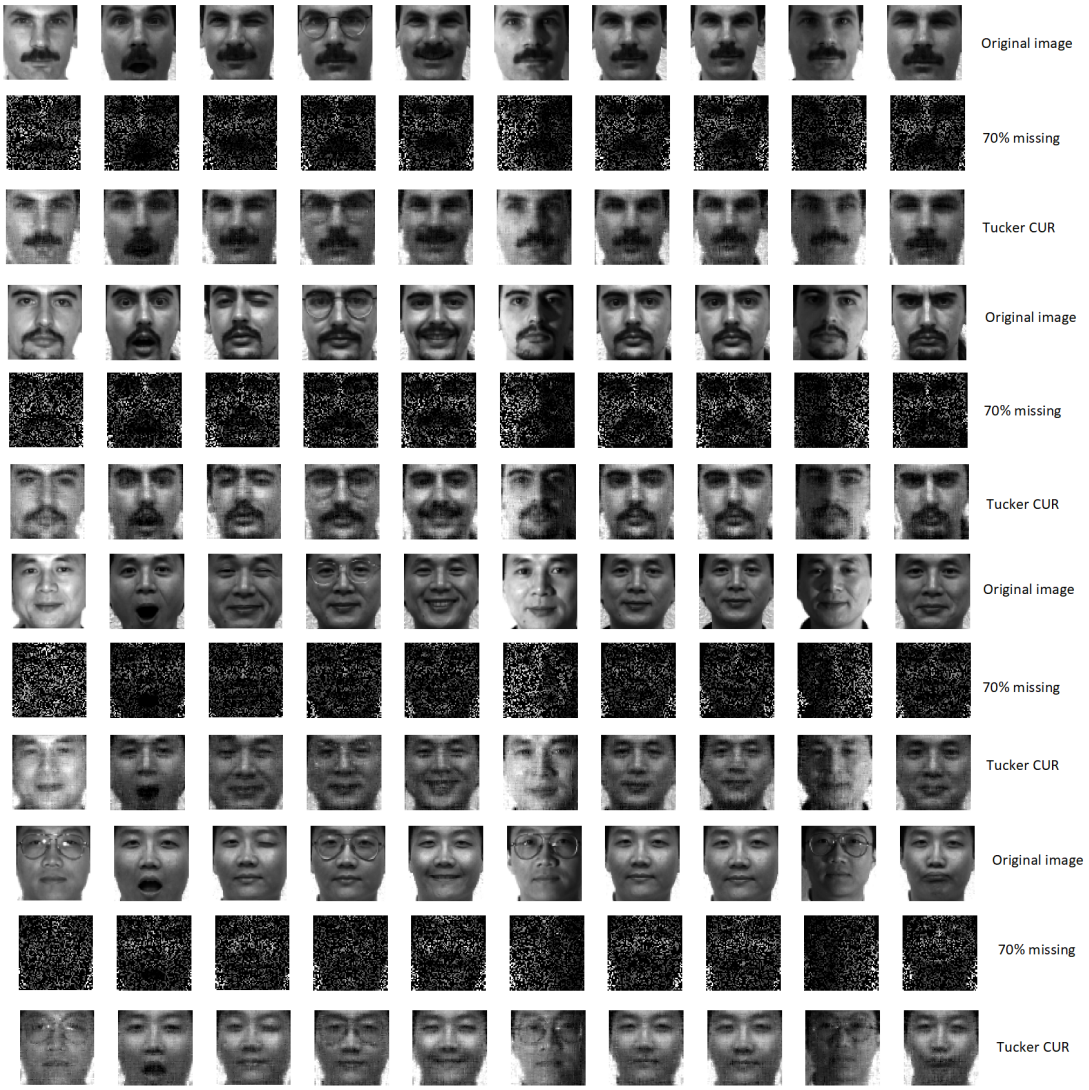


Figure 20: The reconstructed images of some samples of the Yale dataset under different expressions using the Tucker CUR algorithm.

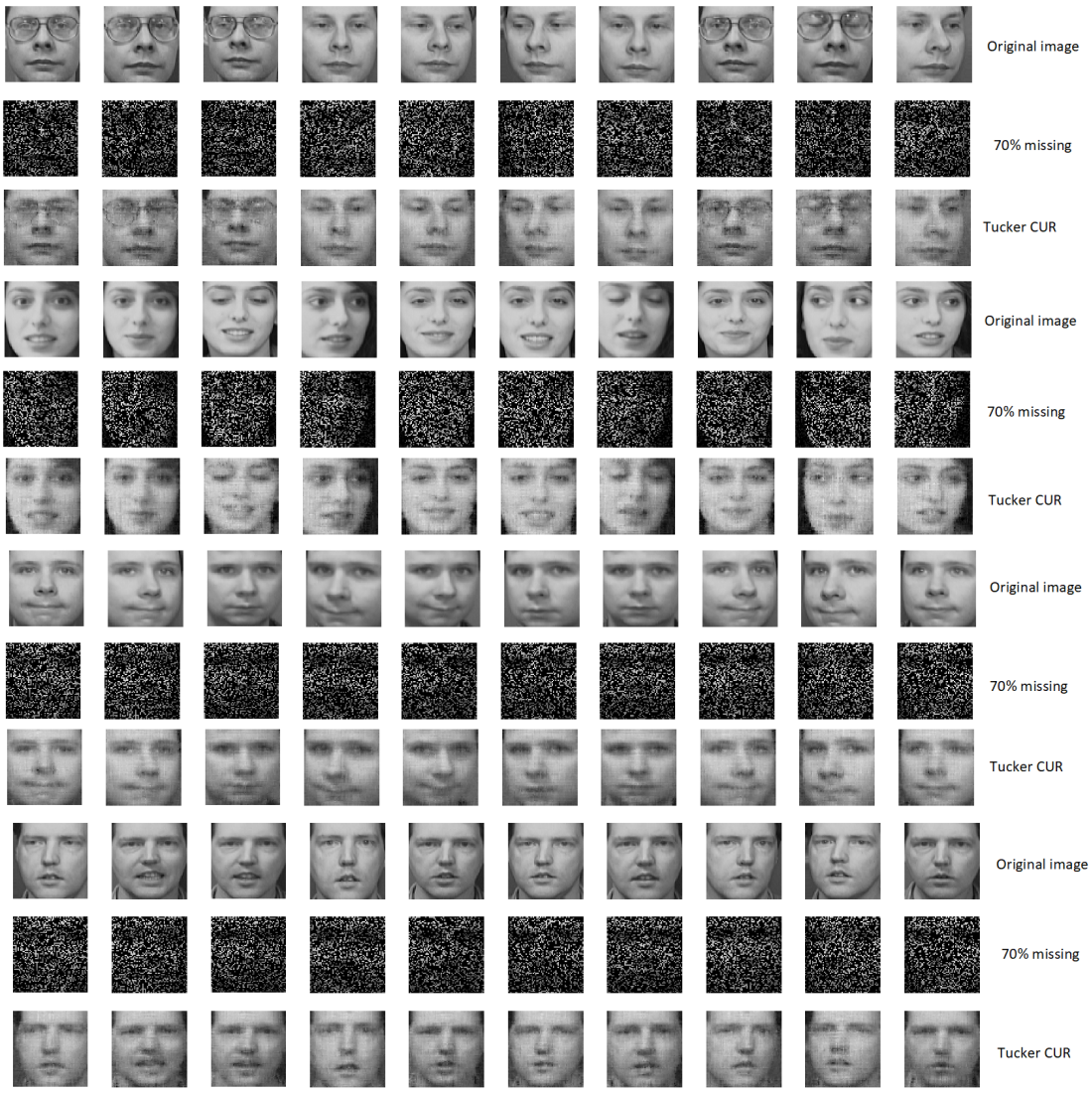


Figure 21: The reconstructed images of some samples of the ORAL dataset under different expressions using the Tucker CUR algorithm.

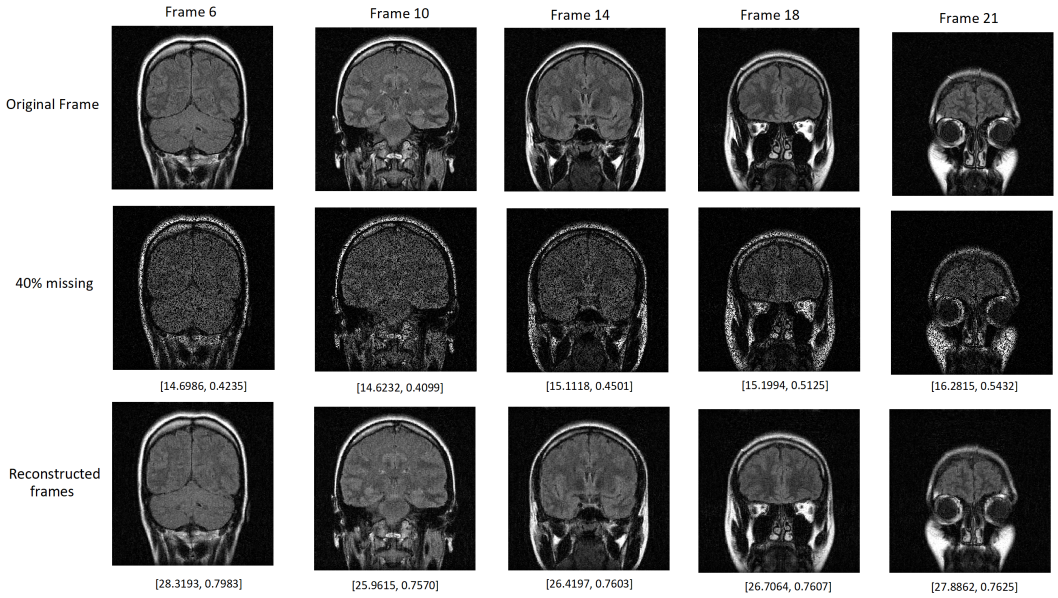
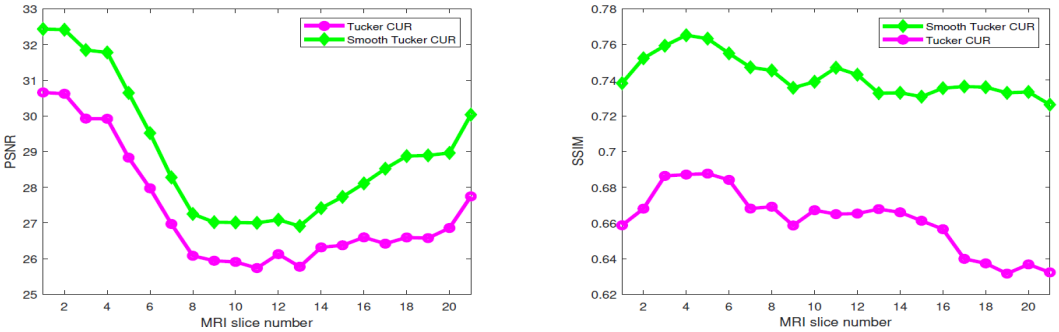


Figure 22: The reconstructed frames (number 6-10-14-18-21) using the Tucker CUR algorithm for the MRI brain dataset.



(a) (b)

Figure 23: The comparison of Tucker CUR and Smooth Tucker CUR for the MRI dataset a) The PSNR comparison b) The SSIM comparison.

and performance of the proposed algorithms are substantiated by extensive simulations on images/videos. Our future work is using our proposed CUR framework combined with a Hankelization step as a pre-processing to further improve the performance of the results.

Acknowledgement

The work was partially supported by the Ministry of Education and Science of the Russian Federation (grant 075.10.2021.068).

Funding and/or conflicts of interests/competing interests

The authors have no relevant financial or non-financial interests to disclose.

References

[1] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D. P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Foundations and Trends® in Machine Learning 9 (4-5) (2016) 249–429.

[2] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, D. P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives, Foundations and Trends® in Machine Learning 9 (6) (2017) 431–673.

[3] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, H. A. Phan, Tensor decompositions for signal processing applications: From two-way to multiway component analysis, IEEE signal processing magazine 32 (2) (2015) 145–163.

[4] A. Cichocki, R. Zdunek, A. H. Phan, S.-i. Amari, Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation, John Wiley & Sons, 2009.

[5] Z. Long, Y. Liu, L. Chen, C. Zhu, Low rank tensor completion for multiway visual data, Signal Processing 155 (2019) 301–316.

[6] Q. Song, H. Ge, J. Caverlee, X. Hu, Tensor completion algorithms in big data analytics, ACM Transactions on Knowledge Discovery from Data (TKDD) 13 (1) (2019) 1–48.

[7] M. G. Asante-Mensah, S. Ahmadi-Asl, A. Cichocki, Matrix and tensor completion using tensor ring decomposition with sparse representation, Machine Learning: Science and Technology 2 (3) (2021) 035008.

[8] E. Frolov, I. Oseledets, Tensor methods and recommender systems, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 7 (3) (2017) e1201.

[9] J. Liu, P. Musialski, P. Wonka, J. Ye, Tensor completion for estimating missing values in visual data, IEEE transactions on pattern analysis and machine intelligence 35 (1) (2012) 208–220.

[10] G. Tomasi, R. Bro, Parafac and missing values, Chemometrics and Intelligent Laboratory Systems 75 (2) (2005) 163–180.

[11] E. Acar, D. M. Dunlavy, T. G. Kolda, Link prediction on evolving data using matrix and tensor factorizations, in: 2009 IEEE International conference on data mining workshops, IEEE, 2009, pp. 262–269.

[12] F. L. Hitchcock, Multiple invariants and generalized rank of a p-way matrix or tensor, Journal of Mathematics and Physics 7 (1-4) (1928) 39–79.

[13] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, Journal of Mathematics and Physics 6 (1-4) (1927) 164–189.

[14] L. R. Tucker, Implications of factor analysis of three-way matrices for measurement of change, Problems in measuring change 15 (1963) 122–137.

[15] L. R. Tucker, et al., The extension of factor analysis to three-dimensional matrices, Contributions to mathematical psychology 110119 (1964).

[16] L. R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (3) (1966) 279–311.

[17] L. De Lathauwer, B. De Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM journal on Matrix Analysis and Applications 21 (4) (2000) 1253–1278.

[18] L. De Lathauwer, Decompositions of a higher-order tensor in block terms—Part II: Definitions and uniqueness, SIAM Journal on Matrix Analysis and Applications 30 (3) (2008) 1033–1066.

[19] L. De Lathauwer, D. Nion, Decompositions of a higher-order tensor in block terms—Part III: Alternating least squares algorithms, SIAM journal on Matrix Analysis and Applications 30 (3) (2008) 1067–1083.

[20] L. De Lathauwer, Decompositions of a higher-order tensor in block terms—Part I: Lemmas for partitioned matrices, SIAM Journal on Matrix Analysis and Applications 30 (3) (2008) 1022–1032.

[21] I. V. Oseledets, Tensor-train decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317.

[22] Q. Zhao, G. Zhou, S. Xie, L. Zhang, A. Cichocki, Tensor ring decomposition, arXiv preprint arXiv:1606.05535 (2016).

[23] M. Espig, K. K. Naraparaju, J. Schneider, A note on tensor chain approximation, Computing and Visualization in Science 15 (6) (2012) 331–344.

[24] M. E. Kilmer, C. D. Martin, Factorization strategies for third-order tensors, Linear Algebra and its Applications 435 (3) (2011) 641–658.

[25] M. E. Kilmer, K. Braman, N. Hao, R. C. Hoover, Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging, SIAM Journal on Matrix Analysis and Applications 34 (1) (2013) 148–172.

[26] K. Braman, Third-order tensors as linear operators on a space of matrices, Linear Algebra and its Applications 433 (7) (2010) 1241–1253.

[27] S. Ahmadi-Asl, S. Abukhovich, M. G. Asante-Mensah, A. Cichocki, A. H. Phan, T. Tanaka, I. Oseledets, Randomized algorithms for computation of Tucker decomposition and higher order svd (HOSVD), IEEE Access 9 (2021) 28684–28706.

[28] S. Ahmadi-Asl, A. Cichocki, A. H. Phan, M. G. Asante-Mensah, M. M. Ghazani, T. Tanaka, I. Oseledets, Randomized algorithms for fast computation of low rank tensor ring model, Machine Learning: Science and Technology 2 (1) (2020) 011001.

[29] M. Xu, R. Jin, Z.-H. Zhou, CUR algorithm for partially observed matrices, in: International Conference on Machine Learning, 2015, pp. 1412–1421.

[30] L. Wang, K. Xie, T. Semong, H. Zhou, Missing data recovery based on tensor-CUR decomposition, IEEE Access 6 (2017) 532–544.

[31] S. A. Goreinov, E. E. Tyrtyshnikov, N. L. Zamarashkin, A theory of pseudoskeleton approximations, Linear algebra and its applications 261 (1-3) (1997) 1–21.

[32] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, N. L. Zamarashkin, How to find a good submatrix, in: Matrix Methods: Theory, Algorithms And Applications: Dedicated to the Memory of Gene Golub, World Scientific, 2010, pp. 247–256.

[33] S. A. Goreinov, E. E. Tyrtyshnikov, The maximal-volume concept in approximation by low-rank matrices, Contemporary Mathematics 280 (2001) 47–52.

[34] D. Savostyanov, Polilinear approximation of matrices and integral equations, Ph.D. thesis, PhD thesis, INM RAS, Moscow, 2006.(in Russian) (2006).

[35] E. Tyrtyshnikov, Incomplete cross approximation in the mosaic-skeleton method, Computing 64 (4) (2000) 367–380.

[36] S. Chaturantabut, D. C. Sorensen, Discrete empirical interpolation for nonlinear model reduction, in: Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, IEEE, 2009, pp. 4316–4321.

[37] D. C. Sorensen, M. Embree, A DEIM induced CUR factorization, SIAM Journal on Scientific Computing 38 (3) (2016) A1454–A1482.

[38] A. Frieze, R. Kannan, S. Vempala, Fast Monte-Carlo algorithms for finding low-rank approximations, Journal of the ACM (JACM) 51 (6) (2004) 1025–1041.

[39] C. Boutsidis, M. W. Mahoney, P. Drineas, An improved approximation algorithm for the column subset selection problem, in: Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms, SIAM, 2009, pp. 968–977.

[40] C. Boutsidis, P. Drineas, M. Magdon-Ismail, Near-optimal column-based matrix reconstruction, SIAM Journal on Computing 43 (2) (2014) 687–717.

[41] A. Deshpande, S. Vempala, Adaptive sampling and fast low-rank matrix approximation, in: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Springer, 2006, pp. 292–303.

[42] A. Deshpande, L. Rademacher, S. Vempala, G. Wang, Matrix approximation and projective clustering via volume sampling, Theory of Computing 2 (1) (2006) 225–247.

[43] A. Deshpande, L. Rademacher, Efficient volume sampling for row/column subset selection, in: 2010 ieee 51st annual symposium on foundations of computer science, IEEE, 2010, pp. 329–338.

[44] V. Guruswami, A. K. Sinop, Optimal column-based low-rank matrix reconstruction, in: Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms, SIAM, 2012, pp. 1207–1214.

[45] A. Mai, L. Tran, L. Tran, N. Trinh, Vgg deep neural network compression via SVD and CUR decomposition techniques, in: 2020 7th NAFOSTED Conference on Information and Computer Science (NICS), IEEE, 2020, pp. 118–123.

[46] E. P. Hendryx, B. M. Rivière, D. C. Sorensen, C. G. Rusin, Finding representative electrocardiogram beat morphologies with CUR, Journal of biomedical informatics 77 (2018) 97–110.

[47] H. Cai, K. Hamm, L. Huang, J. Li, T. Wang, Rapid robust principal component analysis: CUR accelerated inexact low rank estimation, IEEE Signal Processing Letters (2020).

[48] P. Drineas, R. Kannan, M. W. Mahoney, Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication, SIAM Journal on Computing 36 (1) (2006) 132–157.

[49] P. Drineas, R. Kannan, M. W. Mahoney, Fast Monte Carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix, SIAM Journal on computing 36 (1) (2006) 158–183.

[50] P. Drineas, R. Kannan, M. W. Mahoney, Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition, SIAM Journal on Computing 36 (1) (2006) 184–206.

[51] M. W. Mahoney, P. Drineas, CUR matrix decompositions for improved data analysis, Proceedings of the National Academy of Sciences 106 (3) (2009) 697–702.

[52] C. Li, X. Wang, W. Dong, J. Yan, Q. Liu, H. Zha, Joint active learning with feature selection via CUR matrix decomposition, IEEE transactions on pattern analysis and machine intelligence 41 (6) (2018) 1382–1396.

[53] A. Aldroubi, K. Hamm, A. B. Koku, A. Sekmen, CUR decompositions, similarity matrices, and subspace clustering, Frontiers in Applied Mathematics and Statistics 4 (2019) 65.

[54] S. Ahmadi-Asl, C. F. Caiafa, A. Cichocki, A. H. Phan, T. Tanaka, I. Oseledets, J. Wang, Cross tensor approximation methods for compression and dimensionality reduction, IEEE Access 9 (2021) 150809–150838.

[55] E. Drinea, P. Drineas, P. Huggins, A randomized singular value decomposition algorithm for image processing applications, in: Proceedings of the 8th panhellenic conference on informatics, Citeseer, 2001, pp. 278–288.

[56] I. V. Oseledets, D. Savostianov, E. E. Tyrtyshnikov, Tucker dimensionality reduction of three-dimensional arrays in linear time, SIAM Journal on Matrix Analysis and Applications 30 (3) (2008) 939–956.

[57] C. F. Caiafa, A. Cichocki, Generalizing the column–row matrix decomposition to multi-way arrays, Linear Algebra and its Applications 433 (3) (2010) 557–573.

[58] M. W. Mahoney, M. Maggioni, P. Drineas, Tensor-CUR decompositions for tensor-based data, SIAM Journal on Matrix Analysis and Applications 30 (3) (2008) 957–987.

[59] D. A. Tarzanagh, G. Michailidis, Fast randomized algorithms for t-product based tensor operations and decompositions with applications to imaging data, SIAM Journal on Imaging Sciences 11 (4) (2018) 2629–2664.

[60] M. Fazel, Matrix rank minimization with applications, Ph.D. thesis, PhD thesis, Stanford University (2002).

[61] Z. Fang, X. Yang, L. Han, X. Liu, A sequentially truncated higher order singular value decomposition-based algorithm for tensor completion, IEEE transactions on cybernetics 49 (5) (2018) 1956–1967.

[62] E. Acar, D. M. Dunlavy, T. G. Kolda, A scalable optimization approach for fitting canonical tensor decompositions, Journal of Chemometrics 25 (2) (2011) 67–86.

[63] W. Wang, V. Aggarwal, S. Aeron, Efficient low rank tensor ring completion, Rn 1 (r1) (2017) 1.

[64] I. Oseledets, E. Tyrtyshnikov, TT-cross approximation for multidimensional arrays, Linear Algebra and its Applications 432 (1) (2010) 70–88.

[65] A. Savitzky, M. J. Golay, Smoothing and differentiation of data by simplified least squares procedures., Analytical chemistry 36 (8) (1964) 1627–1639.

[66] W. S. Cleveland, Lowess: A program for smoothing scatterplots by robust locally weighted regression, American Statistician 35 (1) (1981) 54.

[67] W. S. Cleveland, Robust locally weighted regression and smoothing scatterplots, Journal of the American statistical association 74 (368) (1979) 829–836.

[68] R. V. Garimella, A simple introduction to moving least squares and local regression estimation, Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2017).

[69] L. Yuan, J. Cao, X. Zhao, Q. Wu, Q. Zhao, Higher-dimension tensor completion via low-rank tensor ring decomposition, in: 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE, 2018, pp. 1071–1076.

[70] T. Yokota, Q. Zhao, A. Cichocki, Smooth parafac decomposition for tensor completion, IEEE Transactions on Signal Processing 64 (20) (2016) 5423–5436.

[71] D. Cai, X. He, Y. Hu, J. Han, T. Huang, Learning a spatially smooth subspace for face recognition, in: Proc. IEEE Conf. Computer Vision and Pattern Recognition Machine Learning (CVPR'07), 2007.

[72] D. Cai, X. He, J. Han, Spectral regression for efficient regularized subspace learning, in: Proc. Int. Conf. Computer Vision (ICCV'07), 2007.

[73] D. Cai, X. He, J. Han, H.-J. Zhang, Orthogonal laplacianfaces for face recognition, IEEE Transactions on Image Processing 15 (11) (2006) 3608–3614.

[74] X. He, S. Yan, Y. Hu, P. Niyogi, H.-J. Zhang, Face recognition using laplacianfaces, IEEE Trans. Pattern Anal. Mach. Intelligence 27 (3) (2005) 328–340.

[75] P. Drineas, M. W. Mahoney, A randomized algorithm for a tensor-based generalization of the singular value decomposition, Linear algebra and its applications 420 (2-3) (2007) 553–571.

[76] S. Friedland, V. Mehrmann, A. Miedlar, M. Nkengla, Fast low rank approximations of matrices and tensors, The Electronic Journal of Linear Algebra 22 (2011).

[77] N. Halko, P.-G. Martinsson, J. A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM review 53 (2) (2011) 217–288.

Appendix I.

The tensor CUR approximation can be defined based on t-product [24, 25], and due to this, we introduce some concepts and definitions related to this concept.

Definition 1. [25] (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

$$\underline{\mathbf{C}} = \underline{\mathbf{X}} * \underline{\mathbf{Y}} = \text{fold}\left(\text{circ}\left(\underline{\mathbf{X}}\right) \text{unfold}\left(\underline{\mathbf{Y}}\right)\right), \tag{16}$$

where

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

and

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

In view of (16), 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). More precisely, we first transform all tubes of two tensors $\underline{\mathbf{X}}$, $\underline{\mathbf{Y}}$, into the frequency domain, then multiply frontal slices of the spectral tensors. Finally, we apply the Inverse FFT (IFFT) to all tubes of the last tensor. This procedure is summarized in Algorithm 2. Note that other types of tensor decompositions in the t-product format can be computed in a similar manner. For example, for computing 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 [25]

$$\widehat{\underline{\mathbf{X}}} = \text{fft}(\underline{\mathbf{X}}, [], 3), \tag{17}$$

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

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

Note that (17) 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 $\underline{\mathbf{R}}$.

Algorithm 2: t-product in the Fourier domain for 3rd-order tensors [24]

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 $\underline{\mathbf{C}} = \underline{\mathbf{Z}} * \underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_4 \times I_3}$

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

Definition 2. [25] (Transpose) Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ be a given 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 3. 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

$$\begin{aligned} \underline{\mathbf{X}} * \underline{\mathbf{X}}^+ * \underline{\mathbf{X}} &= \underline{\mathbf{X}}, & \underline{\mathbf{X}}^+ * \underline{\mathbf{X}} * \underline{\mathbf{X}}^+ &= \underline{\mathbf{X}}^+, \\ \left(\underline{\mathbf{X}} * \underline{\mathbf{X}}^+\right)^T &= \underline{\mathbf{X}} * \underline{\mathbf{X}}^+, & \left(\underline{\mathbf{X}}^+ * \underline{\mathbf{X}}\right)^T &= \underline{\mathbf{X}}^+ * \underline{\mathbf{X}}. \end{aligned}$$

Similar to other tensor operations, the MP pseudoinverse of tensors can be computed through FFT and this is described in Algorithm 3.

Algorithm 3: Computation of Moore-Penrose pseudoinverse for a 3rd-order tensor

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 $\underline{\mathbf{X}}$

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

So, for computing the MP of tensor, it is first converted to the frequency domain, and then the MP of all frontal slices is computed. Finally, the resulting tensor is converted to the original space using inverse FFT.

Appendix II.

In this part, we provide a comprehensive materials about different cross tensor approximation methods and algorithms. For more details we refer to our recent publication [54].

CTA Based on Fiber Selection

The CMA can be extended to tensors based on the Tucker-2 model, see Figure 25. Here, some columns and rows $\mathbf{C}$, $\mathbf{R}$ are selected. To be more precise, matrices are chosen and the middle core tensor $\underline{\mathbf{U}}$ should be computed to yield the smallest error. The optimal option for the middle core tensor is

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

The Tucker-2 model in suitable in situation that the 3rd mode is relatively small and reduction in that mode not required. For example, in the case of color images, the third mode is related to three colors (green-blue-red) and only columns/rows are selected. In the experimental results we show this procedure.

Natural and straightforward generalizations of the CMA to tensors are proposed in [55, 56], and [57]. Similar to the CMA in which a part of columns and rows of a given matrix is sampled, a set of fibers (along different modes) are selected, and the goal is to compute 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 [75] is randomized, while those proposed in [56, 57] 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 obvious. 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

$$\underline{\mathbf{S}} = \underline{\mathbf{X}} \times_1 \mathbf{A}_1^+ \times_2 \mathbf{A}_2^+ \cdots \times_N \mathbf{A}_N^+ \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}, \tag{18}$$

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 are taken from the original data tensor is computed. For noisy tensors, similar to the CMA, the accuracy

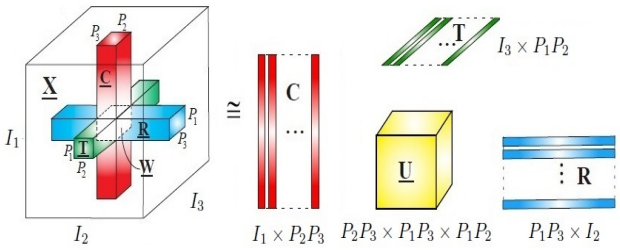


Figure 24: Illustration of the CTA (fiber selection version) for a 3rd-order low-rank tensor. For simplicity of presentation, we assume that all fibers build up to block sub-tensors, [1].

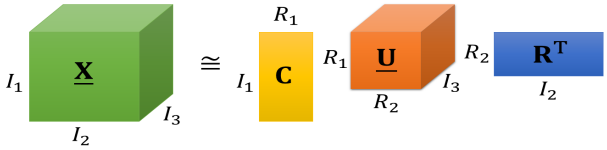


Figure 25: llustration of the Tucker-2 CUR approximation.

of approximation quite depends on the number of selected fibers and also the list of sampled fibers. The idea of sampling fibers and considering them as the factor matrices, first was proposed in [75]. In the first step, the factor matrices are generated, after which the core tensor is computed through (18) as described above. This is summarized in Algorithm 4. In Algorithm 4. a shorthand notation is used for the Tucker decomposition as

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

Algorithm 4: Tucker CUR Algorithm for Nth-order tensors [75]

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 $\underline{\mathbf{X}} \cong [\![\underline{\mathbf{S}}; \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]$

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

2 Sample R_n fibers in n-th mode and generate approximate factor matrix $\mathbf{A}_n \in \mathbb{R}^{I_n \times R_n}$

3 **end**

4 Compute the core tensor $\underline{\mathbf{S}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ as in (18)

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

$$\mathbf{U} = \mathbf{W} \times_1 \mathbf{W}_{(1)}^{+} \times_2 \mathbf{W}_{(2)}^{+} = \mathbf{W}^{+}\mathbf{W}\mathbf{W}^{+} = \mathbf{W}^{+}, \tag{19}$$

which is used as the middle matrix in the CMA, it is suggested [57] to compute the approximate core tensor in the Tucker decomposition as

$$\begin{aligned} \underline{\mathbf{U}} &= \underline{\mathbf{W}} \times_1 \mathbf{W}_{(1)}^{+} \times_2 \mathbf{W}_{(2)}^{+} \times_3 \mathbf{W}_{(3)}^{+} \\ &\equiv \left[\!\left[\underline{\mathbf{W}}; \mathbf{W}_{(1)}^{+}, \mathbf{W}_{(2)}^{+}, \mathbf{W}_{(3)}^{+} \right]\!\right]. \end{aligned} \tag{20}$$

This is a direct generalization of (19) to 3rd-order tensors. In view of (20), 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 24.

They should be selected in an appropriate way. It is shown in [57] 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 approximation can be found as

$$\begin{aligned} \underline{\mathbf{X}} &\cong [\![\underline{\mathbf{U}}; \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3]\!] \\ &\equiv \left[\!\left[\underline{\mathbf{W}}; \underbrace{\mathbf{A}_1\mathbf{W}_{(1)}^{+}}_{\widetilde{\mathbf{C}}_1}, \underbrace{\mathbf{A}_2\mathbf{W}_{(2)}^{+}}_{\widetilde{\mathbf{C}}_2}, \underbrace{\mathbf{A}_3\mathbf{W}_{(3)}^{+}}_{\widetilde{\mathbf{C}}_3} \right]\!\right]. \end{aligned} \tag{21}$$

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 (21).

We refer to this algorithm as Fast Sampling Tucker Decomposition (FSTD) and summarize it in Algorithm 5 [57]. A similar approach is proposed in [76] in the sense of the number of selected fibers in each mode.

CTA Based on Slice-tube 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]. We first briefly describe this idea

Algorithm 5: Fast Sampling Tucker Decomposition (FSTD) algorithm for 3rd-order tensors [57]

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 $\underline{\mathbf{X}} \cong [\![\underline{\mathbf{U}}; \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3]\!]$

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 $\underline{\mathbf{X}} \cong \left[\!\left[\underline{\mathbf{W}}, \mathbf{A}_1\mathbf{W}_1^+, \mathbf{A}_2\mathbf{W}_2^+, \mathbf{A}_3\mathbf{W}_3^+\right]\!\right]$

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, 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 26 (a)). The CTA is then defined as (see Figure 26 (b))

$$\underline{\mathbf{X}} \cong \underline{\mathbf{C}} \times_3 (\mathbf{U}\mathbf{R})^T, \tag{22}$$

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

$$\mathbf{U} = \mathbf{D}_1(\mathbf{D}_2\mathbf{W}\mathbf{D}_1)^+\mathbf{D}_2 \in \mathbb{R}^{L_1 \times L_2},$$

$$\mathbf{W} = \text{reshape}(\underline{\mathbf{W}}, [L_2, L_1]),$$

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

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

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 6. The length-

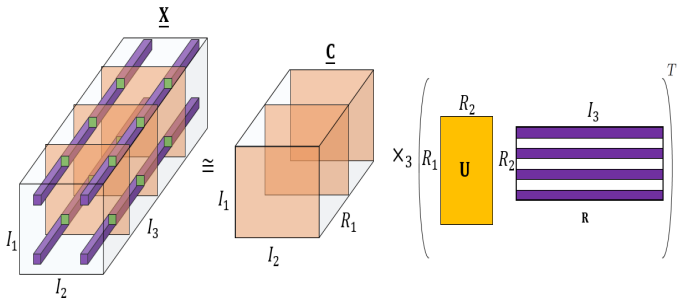


Figure 26: llustration of the CTA based on frontal slice and tube selection.

squared probability distributions are defined as follows

$$\begin{aligned} p_i &= \frac{\|\underline{\mathbf{X}}(:,:,i_3)\|_F^2}{\|\underline{\mathbf{X}}\|_F^2}, \ i_3 = 1, 2, \ldots, I_3, \\ q_j &= \frac{\underline{\mathbf{X}}(j_1, j_2, :)}{\|\underline{\mathbf{X}}\|_F^2}, \ \ j_1, j_2 \in J_1, J_2 \end{aligned} \tag{23}$$

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.

Algorithm 6: Slice-Tube CUR (ST-CUR) algorithm 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 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 $\mathbf{U} = \mathbf{D}_1 (\mathbf{D}_2 \mathbf{W} \mathbf{D}_1)^{+} \mathbf{D}_2$

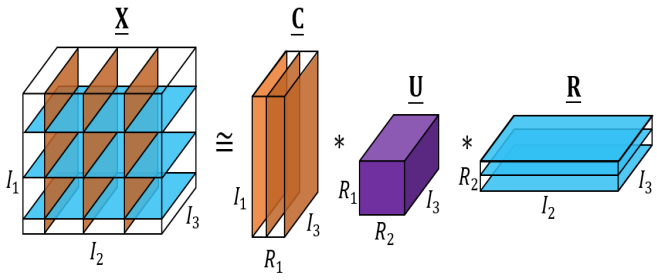


Figure 27: Illustration of the tubal CTA for a 3rd-order tensor.

CTA Based on Tubal Product (t-Product)

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 cross approximation based on t-product is formulated as follows

$$\underline{\mathbf{X}} \cong \underline{\mathbf{C}} * \underline{\mathbf{U}} * \underline{\mathbf{R}}, \tag{24}$$

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 (24) should be as small as possible, see Figure 27, for graphical illustration concerning the tubal CTA.

Note that similar to other CTA algorithms discussed so far, the lateral and horizontal slices sampling procedure can be performed based on prior probability distributions[4]. The probability distributions used in [59] 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

$$\underline{\mathbf{U}} = \underline{\mathbf{C}}^{+} * \underline{\mathbf{X}} * \underline{\mathbf{R}}^{+}, \tag{25}$$

because

$$\underline{\mathbf{C}}^{+} * \underline{\mathbf{X}} * \underline{\mathbf{R}}^{+} = \underset{\underline{\mathbf{U}} \in \mathbb{R}^{L \times I_2 \times I_3}}{\operatorname{argmin}} \|\underline{\mathbf{X}} - \underline{\mathbf{C}} * \underline{\mathbf{U}} * \underline{\mathbf{U}}\|_F,$$

This is a straightforward generalization of the CMA [31] to tensors. Formula (25) can be computed in the Fourier domain and these computations are summarized in

[4]Here different various probability distributions (with/without replacement) can be used but we will not go through the theoretical details [59].

Algorithm 7. However, it is clear that (25) 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 [77]. 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 $\underline{\mathbf{X}}$.

Algorithm 7: Tubal CUR algorithm for 3rd-order tensors

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 $\underline{\mathbf{R}}$

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

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

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

end

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

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

end

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

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

$$\underline{\mathbf{X}} \cong \underline{\mathbf{C}} * \underline{\mathbf{Y}}. \tag{26}$$

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 $\underline{\mathbf{Y}} \in \mathbb{R}^{L \times I_2 \times I_3}$ is computed in such a way that the reconstruction error (26) is as small as possible [59]. The best solution for the tensor $\underline{\mathbf{Y}}$ is

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

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

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

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