PAPER • OPEN ACCESS

Matrix and tensor completion using tensor ring decomposition with sparse representation

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
4 September 2020

REVISED
19 October 2020

ACCEPTED FOR PUBLICATION
17 November 2020

PUBLISHED
17 May 2021

PAPER

Matrix and tensor completion using tensor ring decomposition with sparse representation

Maame G Asante-Mensah[1], Salman Ahmadi-Asl[1] and Andrzej Cichocki[1,2]

[1] Skolkovo Institute of Science and Technology (SKOLTECH), CDISE, Moscow, Russia
[2] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

E-mail: gyamfua.asantemensah@skoltech.ru

Keywords: tensor decomposition, tensor networks, tensor ring, tensor completion, Hankelization, sparse representation

Abstract
Completing a data tensor with structured missing components is a challenging task where the missing components are not distributed randomly but they admit some regular patterns, e.g. missing columns and rows or missing blocks/patches. Many of the existing tensor completion algorithms are not able to handle such scenarios. In this paper, we propose a novel and efficient approach for matrix/tensor completion by applying Hankelization and distributed tensor ring decomposition. Our main idea is first Hankelizing an incomplete data tensor in order to obtain high-order tensors and then completing the data tensor by imposing sparse representation on the core tensors in tensor ring format. We apply an efficient over-complete discrete cosine transform dictionary and sparse representation techniques to learn core tensors. Alternating direction methods of multiplier and accelerated proximal gradient approaches are used to solve the underlying optimization problems. Extensive simulations performed on image, video completions and time series forecasting show the validity and applicability of the method for different kinds of structured and random missing elements.

1. Introduction

In various applications of multi-dimensional data processing, the underlying data tensors or matrices are corrupted by outliers and/or incomplete due to imprecise data acquisition process, or corruption by artifacts [1, 2]. The problem of recovering an original data tensor from its partially observed entries is called tensor completion and has found many applications such as in computer vision, data mining, recommender systems, image inpainting and signal reconstruction. Due to its importance and many practical applications, many algorithms have been developed to solve this problem. Most of the methods are based on low-rank matrix and tensor decomposition. For a comprehensive overview of such algorithms, we refer to [3, 4]. The applicability and performance of these algorithms highly depend on the probability distribution upon which the data components are missing. Many of these algorithms work quite well only when the underlying missing components are distributed uniformly and fail to recover the data tensor when a whole part of a data tensor is missing, e.g. several sequential rows or columns of a frame are missing.

Motivated by such limitations of existing tensor completion techniques, we exploit tensor ring (TR) decomposition combined with Hankelization [5, 6]. The idea of Hankelization was first proposed in [7] and recently generalized to tensors in [5] but so far has not been exploited extensively for the TR representation with sparse constraints. A key pre-processing step is a prior Hankelization procedure (to be discussed in section 3) which allows us to apply the TR representation and develop an efficient tensor completion algorithm. The original data tensor is recovered through a so-called inverse Hankelization or De-Hankelization [5]. In this paper, we propose a new approach for the tensor completion problem. Our main idea is based on applying dictionary learning technique to the block Hankelized high-order data tensor in TR format to yield sparse representations of the core tensors. More precisely, in contrast to the existing Hankelized tensor completion methods [5, 8], we do not complete a Hankelized data tensor directly or using Tucker decomposition, instead we use constrained TR format. We first initialize the core tensors and then they are updated sequentially using an efficient dictionary learning procedure applied to each core tensor. The size of core tensors is smaller than the original data tensor which leads to lower-scale optimization problems.

The problem is formulated as a convex optimization problem which is solved efficiently by the alternating direction methods of multiplier (ADMM) [9]. Our main contributions in this paper are as follows:

- Applying Hankelization of incomplete data in order to obtain higher order tensors and represent it in the TR format.
- Exploiting the dictionary learning technique for sparse representations of the TR core tensors.
- Developing an optimised algorithm, which allows to reconstruct images corrupted by structured distortions, like rows, columns or blocks.

The rest of this paper is organized as follows. In section 2, basic notations and concepts used throughout the paper are introduced. In section 3, we describe the Hankelization technique and related concepts. The proposed method and methodology is presented in section 4. Extensive results are illustrated in section 6, and finally, a conclusion is given in section 7.

2. Notations and preliminaries

In this section, basic notations used throughout the paper are introduced and they are adapted from [10]. A scalar is denoted by standard lowercase letters, e.g. a, vectors or 1st-order tensors are denoted by boldface lower case letters, e.g. $\mathbf{X}$, matrices or second order tensors are denoted by boldface capital letters, e.g. $\mathbf{X}$ and higher order tensor are denoted by bold underlined capital letters, e.g. $\underline{\mathbf{X}}$. Also a tensor graphically represented as, see figure 1(a). An $(i_1, i_2, \ldots, i_N)$th element of an Nth-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $x_{i_1, i_2, \ldots, i_N} = \underline{\mathbf{X}}(i_1, i_2, \ldots, i_N)$. The trace and Moore–Penrose pseudoinverse of matrices are denoted by 'Tr' and †, respectively. The multi-index is defined as $\overline{i_1 i_2 \cdots i_N} = i_N + (i_{N-1} - 1)\, I_N + \cdots + (i_1 - 1)\, I_2 I_3 \cdots I_N$. Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be an Nth-order tensor, then the n-unfolding of the tensor $\underline{\mathbf{X}}$ is of size $\prod_{i=1}^{n} I_i \times \prod_{i=n+1}^{N} I_i$ and denoted by $\mathbf{X}_{\langle n \rangle}$ whose elements are defined as (see figure 1)

$$\mathbf{X}_{\langle n \rangle}\left(\overline{i_1 \cdots i_n}, \overline{i_{n+1} \cdots i_N}\right) = \mathbf{X}(i_1, i_2, \ldots, i_N).$$

Similarly, the mode-n unfolding matrix of size $I_n \times \prod_{i \neq n} I_i$ is denoted by $\mathbf{X}_{(n)}$ whose components are

$$\mathbf{X}_{(n)}\left(i_n, \overline{i_{n+1} \cdots i_N i_1 \cdots i_{n-1}}\right) = \underline{\mathbf{X}}(i_1, i_2, \ldots, i_N).$$

For a graphical illustration see figure 1(b). Slices are sub-tensors generated by fixing all indices except two of them and as a result, they are matrices. For example, for a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the slices $\underline{\mathbf{X}}(:, :, i)$, $i = 1, 2, \ldots, I_3$, are called frontal slices. The notation 'Vec' denotes the vectorization operator which stacks the columns of a matrix. The inner product of two tensors $\underline{\mathbf{X}}, \underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined as $\langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle = \sum_{i_1} \sum_{i_2} \cdots \sum_{i_N} a_{i_1, \ldots, i_N} b_{i_1, \ldots, i_N}$ and the induced Frobenius norm is $\|\underline{\mathbf{X}}\|_F = \sqrt{\langle \underline{\mathbf{X}}, \underline{\mathbf{X}} \rangle}$.

The n-mode (matrix) multiplication of a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\mathbf{Q} \in \mathbb{R}^{J \times I_n}$ is denoted by $\underline{\mathbf{X}} \times_n \mathbf{Q} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$ whose components are

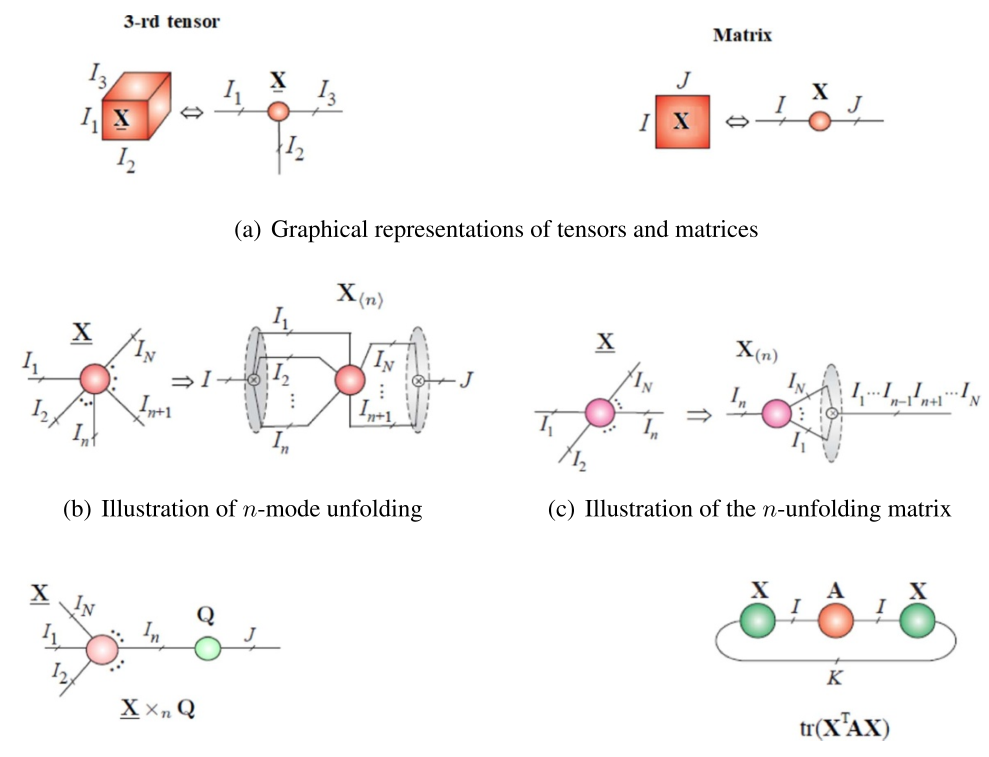
$$(\underline{\mathbf{X}} \times_n \mathbf{Q})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n = 1}^{I_N} x_{i_1 i_2 \cdots i_N} q_{j i_n}, \quad j = 1, 2, \ldots, J. \tag{1}$$

For a graphical illustration see figure 1(d).

2.1. Tensor train (TT) and TR decompositions

Tensor networks (TNs) decompose higher-order tensors into sparsely interconnected small-scale low-order core tensors [10]. TT [11] and TR[3] (TR) [12–15] are two powerful and relatively simple TNs representing a higher order tensor as a train and ring (or chain) of third order core tensors, for a comprehensive study on TNs and applications in physics and machine learning, we refer to [16–26]. The TT and the TR decompositions are also known as matrix product state (MPS) and matrix product state with periodic boundary condition (MPS-PBC) in quantum physics [16, 27]. They have found many practical applications

[3] It is also known as tensor chain (TC) decomposition.


3-rd tensor
Matrix
$\mathbf{X}_{\langle n \rangle}$
$\mathbf{X}_{(n)}$
$\underline{\mathbf{X}} \times_n \mathbf{Q}$
$\mathrm{tr}(\mathbf{X}^T\mathbf{A}\mathbf{X})$

(a) Graphical representations of tensors and matrices

(b) Illustration of n-mode unfolding

(c) Illustration of the n-unfolding matrix

(d) Illustration of n-mode (matrix) and tensor multiplication

(e) Illustration of trace multiplication of three matrices

Figure 1. Graphical illustration of some tensor operations (for details see [10]).

such as machine learning [21, 28–30], compressing deep neural networks [31–33], tensor completion [34–37], and hyperspectral image super-resolution [38, 39]. The memory storage requirement of these two TNs scale linearly with the order of the tensor so they break the *curse of dimensionality* which is a main bottleneck in handling high order data tensors [40]. The TT decomposition can be considered as a special case of the TR decomposition and because of this, throughout the paper, we mainly focus on the TR decomposition. Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be an Nth-order data tensor. The TR decomposition represents the data tensor $\underline{\mathbf{X}}$ into a sequence of latent core tensors $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$, $n = 1, 2, \ldots, N$. The element-wise representation of the data tensor $\underline{\mathbf{X}}$ in the TR format can be expressed as

$$\underline{\mathbf{X}}(i_1, i_2, \ldots, i_N) \cong \mathrm{Tr}\left(\prod_{n=1}^{N} \mathbf{G}^{(n)}(i_n)\right), \tag{2}$$

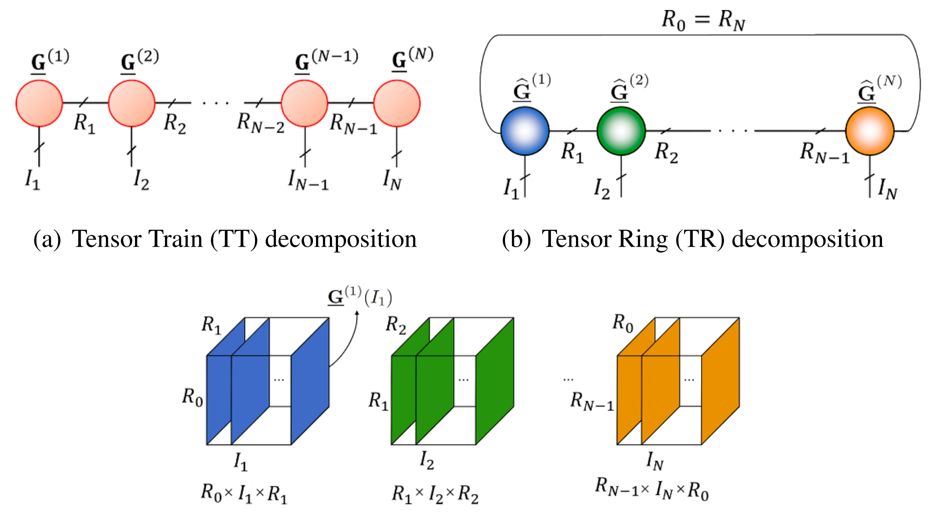
where $\mathbf{G}^{(n)}(i_n) \in \mathbb{R}^{R_{n-1} \times R_n}$ is the i_nth lateral slice matrix of the core tensor $\underline{\mathbf{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$. The expanded form of (2), is

$$\underline{\mathbf{X}}(i_1, i_2, \ldots, i_N) \cong \sum_{r_0=1}^{R_0} \cdots \sum_{r_{N-1}=1}^{R_{N-1}} \underline{\mathbf{G}}^{(1)}(r_0, i_1, r_1) \cdots \underline{\mathbf{G}}^{(N)}(r_{N-1}, i_N, r_0), \tag{3}$$

and the N-tuple $(R_0, R_1, \ldots, R_{N-1})$ is called TR-ranks. Note that in the TR decomposition, we have $R_0 = R_N$ and it is also shown in [15] that the TR-ranks satisfy $R_0 R_n \leqslant \mathrm{rank}\left(\mathbf{X}_{\langle n \rangle}\right)$ for $n = 1, 2, \ldots, N$. A shorthand notation for the TR decomposition is as follows [10]:

$$\underline{\mathbf{X}} \cong \ll \underline{\mathbf{G}}^{(1)}, \underline{\mathbf{G}}^{(2)}, \ldots, \underline{\mathbf{G}}^{(N)} \gg .$$

As mentioned earlier, the TT/MPS decomposition is a particular case of the TR/MPS-PBC decomposition, for $R_0 = R_N = 1$. This means that the first and last cores in the TT decomposition are matrices and rest of them are 3rd-order tensors. For graphical illustration of the TT and the TR decompositions, see figures 2(a)–(c). For a comprehensive overview on fast algorithms for the TR decomposition, see [41].


$\underline{\mathbf{G}}^{(1)}$
$\underline{\mathbf{G}}^{(2)}$
$\underline{\mathbf{G}}^{(N-1)}$
$\underline{\mathbf{G}}^{(N)}$
R_1
R_2
R_{N-2}
R_{N-1}
I_1
I_2
I_{N-1}
I_N
$R_0 = R_N$
$\widehat{\underline{\mathbf{G}}}^{(1)}$
$\widehat{\underline{\mathbf{G}}}^{(2)}$
$\widehat{\underline{\mathbf{G}}}^{(N)}$
$\underline{\mathbf{G}}^{(1)}(I_1)$
R_0
R_{N-1}
$R_0 \times I_1 \times R_1$
$R_1 \times I_2 \times R_2$
$R_{N-1} \times I_N \times R_0$

(a) Tensor Train (TT) decomposition (b) Tensor Ring (TR) decomposition

(c) Slice-wise representation of the tensor ring (TR) decomposition

Figure 2. Graphical illustration of the tensor train (TT) and tensor ring (TR) decompositions, reprinted from [41].

2.2. Low-rank TR completion

Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be an incomplete data tensor with observation index tensor $\underline{\mathbf{\Omega}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, defined as

$$\underline{\mathbf{\Omega}}(i_1, i_2, \ldots, i_N) = \begin{cases} 1 & \text{if} x_{i_1, i_2, \ldots, i_N} \text{ is known,} \\ 0 & \text{if} x_{i_1, i_2, \ldots, i_N} \text{ is unknown,} \end{cases}$$

and its complement as

$$\underline{\mathbf{\Omega}}^{\perp}(i_1, i_2, \ldots, i_N) = \begin{cases} 0 & \text{if} x_{i_1, i_2, \ldots, i_N} \text{ is known,} \\ 1 & \text{if} x_{i_1, i_2, \ldots, i_N} \text{ is unknown.} \end{cases}$$

The operator $\mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{X}})$ projecting the data tensor $\underline{\mathbf{X}}$ onto the observation index tensor $\underline{\mathbf{\Omega}}$ is defined as

$$\mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{X}}) = \begin{cases} x_{i_1, i_2, \ldots, i_N} & (i_1, i_2, \ldots, i_N) \in \underline{\mathbf{\Omega}}, \\ 0 & \text{Otherwise.} \end{cases}$$

The task of low rank TR completion, can be formulated as the following optimization problem

$$\underset{\widehat{\underline{\mathbf{X}}}}{\arg\min} \left\| \mathbf{P}_{\underline{\mathbf{\Omega}}}(\underline{\mathbf{X}}) - \mathbf{P}_{\underline{\mathbf{\Omega}}}\left(\widehat{\underline{\mathbf{X}}}\right) \right\|_F, \tag{4}$$

where

$$\widehat{\underline{\mathbf{X}}} = \ll \underline{\mathbf{G}}^{(1)}, \underline{\mathbf{G}}^{(2)}, \ldots, \underline{\mathbf{G}}^{(N)} \gg, \tag{5}$$

and the goal is finding a low rank TR approximation $\widehat{\underline{\mathbf{X}}}$. If the TR-ranks $(R_0, R_1, \ldots, R_{N-1})$ are known in advance, then the completion algorithm is called *non-adaptive* otherwise it is called *adaptive* where the TR ranks is obtained by the algorithm automatically. Our proposed algorithm is non-adaptive. For simplicity, we assume that $R_n = R, \forall n$.

3. Block Hankelization

A matrix is called a Hankel matrix if all its elements along the skew-diagonals are constant. A Hankel tensor is a generalization of Hankel matrices. A data tensor $\underline{\mathbf{X}}$ is called a Hankel tensor if all components $x_{i_1, i_2, \ldots, i_N}$ for which the quantity $i_1 + i_2 + \cdots + i_N$ is fixed, are the same. The procedure of generating a Hankel matrix/tensor from a given data vector/matrix/tensor is called Hankelization.

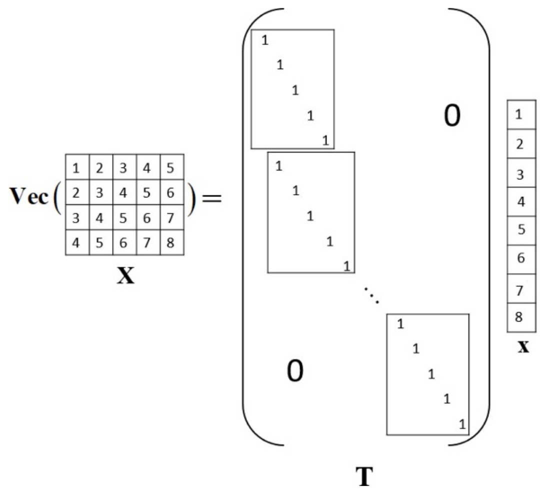


Figure 3. Duplication matrix for a vector of size 8 and window size $\tau = 5$.

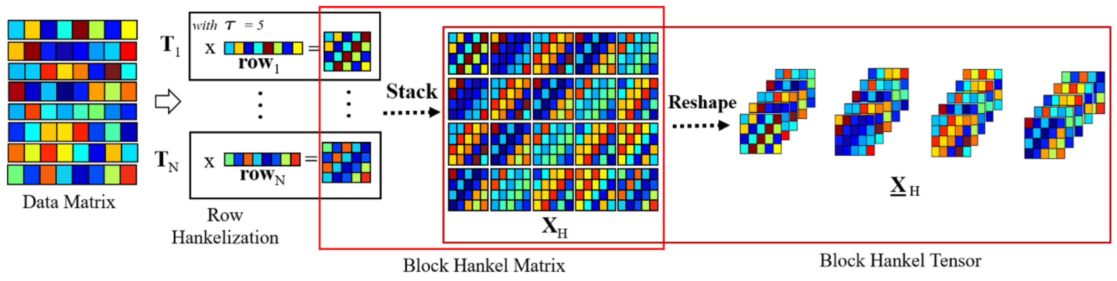


Figure 4. Illustration of block Hankelization of a matrix and construction of a 4th-order tensor using row scanning. Similar tensors can be obtained by column scanning.

We first review basic concepts of Hankelization taken from [5]. From a vector $\mathbf{x} \in \mathbb{R}^N$, we can generate a Hankel matrix. For example, for a given vector $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T \in \mathbb{R}^N$, and a window size τ, the operator $\mathcal{H}_\tau(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}^{\tau \times (N-\tau+1)}$ generates a Hankel matrix as

$$\mathbf{X}_H = \mathcal{H}_\tau(\mathbf{X}) := \begin{pmatrix} x_1 & x_2 & \cdots & x_{N-\tau+1} \\ x_2 & x_3 & \cdots & x_{N-\tau+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_\tau & x_{\tau+1} & \cdots & x_N \end{pmatrix} \in \mathbb{R}^{\tau \times (N-\tau+1)}.$$

The operator $\mathcal{H}_\tau(\mathbf{X})$ is also called delay embedding transform of vector $\mathbf{X}$ with window size τ [5, 6]. Equivalently the Hankelization procedure can be expressed by a duplication matrix which provides a relationship between a Hankel matrix $\mathbf{X}_H$ and a corresponding given vector $\mathbf{X}$. Let $\mathbf{X}_H$, be a Hankel matrix, then $\mathbf{T} \in \{0,1\}^{\tau(N-\tau+1)\times N}$, is called a duplication matrix [5, 42] if

$$\text{vec}(\mathbf{X}_H) = \mathbf{T}\mathbf{x}. \tag{6}$$

The operator $\text{fold}_{(N,\tau)} : \mathbb{R}^{\tau(N-\tau+1)} \to \mathbb{R}^{\tau \times (N-\tau+1)}$ which returns back a vectorized form of a Hankel matrix to the original Hankel matrix is called folding operator, that is

$$\mathbf{X}_H = \text{fold}_{(N,\tau)}(\mathbf{T}\mathbf{x}). \tag{7}$$

A sample illustration of this definition for a vector of size 8 and window size $\tau = 5$ is presented in figure 3. It is seen that the duplication matrices includes block of shifted identity matrices.

A block Hankel matrix is defined similarly, where instead of the individual entries of a matrix, its blocks are repeated along the block skew-diagonals of the matrix. More precisely, from a set of matrices, we can generate a block Hankel matrix; see figure 4 for an illustration of the block Hankelization procedure.

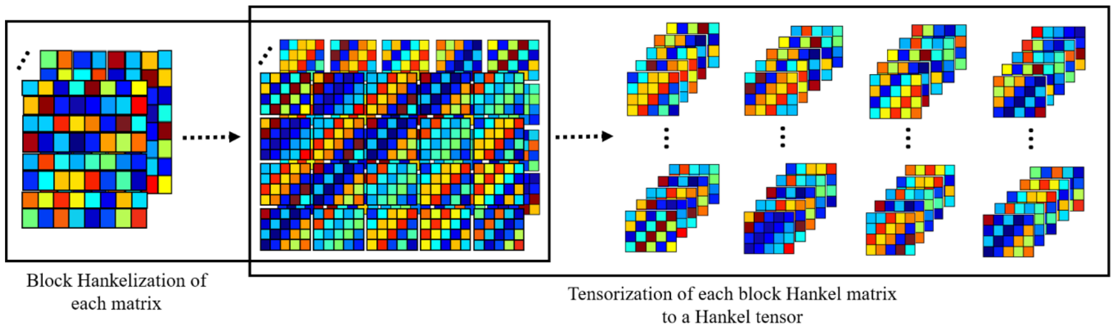

Block Hankelization of each matrix
Tensorization of each block Hankel matrix to a Hankel tensor

Figure 5. Illustration of Hankelization procedure for 3rd order tensor which leads to generation of a 5th order tensor.

A vector from which a Hankel matrix was constructed can be obtained through so called inverse Hankelization [5] as follows:

$$\mathbf{X} = \mathcal{H}_\tau^{-1}(\mathbf{X}_H) = \mathbf{T}^\dagger \text{vec}(\mathbf{X}_H),$$

where $\mathbf{T}$ is a duplication matrix. The concept of delay embedding can be generalized to higher order tensors via multi-way delay embedding [5, 6, 42]. The intuition behind this generalization is that the classical delay embedding can be considered as a tensor-vector multiplication i.e. $\mathbf{X}\times_1\mathbf{T} = \mathbf{TX}$ where for a vector as a first order tensor, we have only one duplication matrix $\mathbf{T}$. For a matrix $\mathbf{X}\in\mathbb{R}^{I_1\times I_2}$ (which is a second order tensor), two duplication matrices $\mathbf{T}_1 \in \{0,1\}^{\tau_1(I_1-\tau_1+1)\times I_1}$ and $\mathbf{T}_2 \in \{0,1\}^{\tau_2(I_2-\tau_2+1)\times I_2}$ are used and we have

$$\mathbf{Y} = \mathbf{X}\times_1\mathbf{T}_1\times_2\mathbf{T}_2 = \mathbf{T}_1\,\mathbf{X}\,\mathbf{T}_2^T \in \mathbb{R}^{\tau_1(I_1-\tau_1+1)\times\tau_1(I_2-\tau_2+1)}.$$

Then the operator $\text{fold}_{(N,\tau)}(\mathbf{X}\times_1\mathbf{T}_1\times_2\mathbf{T}_2)$ reshapes the data matrix $\mathbf{Y}$ to a fourth order Hankel tensor of size $\tau_1\times(I_1-\tau_1+1)\times\tau_1\times(I_2-\tau_2+1)$.

For an Nth-order tensor, we should consider N duplication matrices $\mathbf{T}_n\in\{0,1\}^{\tau_n(I_n-\tau_n+1)\times I_n}$, $n=1,2,\ldots,N$ and the multi-way embedded space is defined as [5]

$$\underline{\mathbf{X}}_H = \mathcal{H}_\tau(\underline{\mathbf{X}}) = \text{fold}_{(I,\tau)}(\underline{\mathbf{X}}\times_1\mathbf{T}_1\cdots\times_N\mathbf{T}_N),$$

where $\text{fold}_{(I,\tau)}:\mathbb{R}^{\tau_1(I_1-\tau_1+1)\times\cdots\times\tau_N(I_N-\tau_N+1)}\to\mathbb{R}^{\tau_1\times(I_1-\tau_1+1)\times\cdots\times\tau_N\times(I_N-\tau_N+1)}$ transforms an Nth-order tensor to an $2N$th-order tensor. The N-tuple $(\tau_1,\tau_2,\ldots,\tau_N)$ indicates the window size of different duplication matrices corresponding to different modes. Illustration for Hankelization of a matrix and transforming it to a 4th-order tensor is shown in figure 4. The reverse procedure where the task is retrieving the original data tensor from the tensor $\underline{\mathbf{X}}_H$ was constructed, can be expressed as

$$\mathcal{H}_\tau^{-1}(\underline{\mathbf{X}}_H) = \text{unfold}_{(I,\tau)}(\underline{\mathbf{X}}_H)\times_1\mathbf{T}_1^\dagger\cdots\times_N\mathbf{T}_N^\dagger,$$

where $\text{unfold}_{(I,\tau)} = \text{fold}_{(I,\tau)}^{-1}$, and it basically transforms the $2N$th-order block Hankel tensor $\underline{\mathbf{X}}_H$ of size $\tau_1\times(I_1-\tau_1+1)\times\cdots\times\tau_N\times(I_N-\tau_N+1)$ to an Nth-order tensor of size $\tau_1(I_1-\tau_1+1)\times\cdots\times\tau_N(I_N-\tau_N+1)$.

In the image processing community, the Hankelization procedure or equivalently delay/shift embedding is a technique that duplicates patches of an image with prescribed window sizes. It is known that a prior Hankelization procedure as a pre-processing step can capture the delay/shift-invariant features (e.g. non-local similarity) of signals/images [42]. This is an effective property to learn the hidden structure of the images which allow us to apply this technique in denoising and completion [5, 42]. In particular, it is experimentally shown that for incomplete data tensors with unstructured missing patterns, this technique is quite efficient. For example, for a video with removing slices instead of pixels, the superiority of Hankelization in recovering such data tensors is shown in [5, 42].

Although it is possible to use more sophisticated Hankelization, such as patch-based/block Hankelization [8], we however perform a relatively simple row Hankelization procedure due to its simplicity and efficiency. In our approach, each row of the frontal slices of the data tensor is Hankelized using a given window size and a corresponding duplication matrix. Then, from these matrices, block Hankel matrices are constructed. Having computed all the block Hankel matrices, they are reshaped to higher order tensors, as illustrated in figure 5. By column Hankelization, we can obtain similar or the same results.

The existing algorithms are based on applying a completion technique to the Hankelized data tensor. In [5] Tucker decomposition was used, while in [8] the TT is exploited. Our work is quite different from them in the sense that we learn the core tensors of the TR decomposition of the incomplete data tensor through a dictionary learning technique and sparse representations.

4. Proposed optimization procedure and learning algorithm

In this section, we discuss in detail our approach. Our main idea is first Hankelizing the incomplete data tensor $\underline{\mathbf{X}}$ into $\underline{\mathbf{X}}_H$ using row Hankelization approach and then recovering the incomplete Hankelized data tensor while imposing sparse representation constraint on the core tensors. We will experimentally show that such a formulation allows us to recover the structured missing patterns of an incomplete data tensor efficiently and achieve better performance compared to many other existing algorithms for this task. Unlike the methods in [5, 8], where the completion approaches employ Tucker or the TT representations is applied to $\underline{\mathbf{X}}_H$ and then the original data tensor is reconstructed through inverse Hankelization, we consider the optimization problem (4) with block Hankelized data tensor $\widehat{\underline{\mathbf{X}}}_H$ (described in section 3 and figures 4 and 5) and exploit a TR decomposition as

$$\widehat{\underline{\mathbf{X}}}_H = \ll \widehat{\underline{\mathbf{G}}}^{(1)}, \widehat{\underline{\mathbf{G}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{G}}}^{(N)} \gg,$$

where the core tensors $\widehat{\underline{\mathbf{G}}}^{(n)}$, $n = 1, 2, \ldots, N$, are estimated using dictionary learning. To this end, we formulate an optimization problem in which the constrained core tensors $\widehat{\underline{\mathbf{G}}},^{(n)}$ $n = 1, 2, \ldots, N$, are updated sequentially to minimize the cost function in (4). Moreover, inspired by [43, 44], we imposed implicit sparse representation constraint on the core tensors $\widehat{\underline{\mathbf{G}}}^{(n)}$, $(n = 1, 2, \ldots, N)$. So we formulated the following constrained optimization problem:

$$\begin{aligned} \min_{\left\{\widehat{\underline{\mathbf{G}}}^{(1)}, \widehat{\underline{\mathbf{G}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{G}}}^{(N)}\right\}} \quad & \left\| \mathbf{P}_{\Omega_H} \left(\widehat{\underline{\mathbf{X}}}_H - \ll \widehat{\underline{\mathbf{G}}}^{(1)}, \widehat{\underline{\mathbf{G}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{G}}}^{(N)} \gg \right) \right\|_F^2 + \eta \sum_{n=1}^{N} \left\| \mathbf{C}^{(n)} \right\|_1 \\ \text{s.t.} \quad & \widehat{\mathbf{G}}^{(n)}_{(2)} = \mathbf{D}^{(n)} \mathbf{C}^{(n)}, \quad n = 1, 2, \ldots, N, \end{aligned} \tag{8}$$

where $\widehat{\mathbf{G}}^{(n)}_{(2)} \in \mathbb{R}^{I_n \times R_{n-1} R_n}$ is mode-2 matricization of the core tensor $\widehat{\underline{\mathbf{G}}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$, $\mathbf{D}^{(n)} \in \mathbb{R}^{I_n \times R_{n-1} R_n}$ is a pre-defined dictionary, $\mathbf{C}^{(n)} \in \mathbb{R}^{R_n R_{n-1} \times R_n R_{n-1}}$ is a sparse matrix and $\eta > 0$ is a regularization parameter. The method works well for quite a range of η from 0.05 to 0.3. We chose $\eta = 0.082$ to achieve best performance. It is worth mentioning that the optimization problem (8) is convex with respect to each core tensor $\widetilde{\underline{\mathbf{G}}}^{(n)}$ but it is not convex with respect to the whole set of core tensors. As a result, there is not any guarantee to converge to a global minimum, however we have used a suboptimal approximation. Also, in contrast to [43], where the sparse representation of unfolding matrices are employed, we impose sparsity representation constraint on the core tensors of the TR network. This can reduce the computational complexity of the algorithm because of smaller size of the TR core tensors compared to the unfolding matrices and also improve the performance. Various dictionary learning algorithms can be used for computing the dictionary $\mathbf{D}^{(n)}$ such as over-complete discrete cosine transform (ODCT) dictionary[4] [45, 46], online dictionary learning [47] and group sparse dictionary learning technique [44]. We adopted for our purpose the ODCT method [46, 48] because it is quite versatile and works for a broad range of data.

In the algorithm proposed in [43], the sparse representation constraints are imposed on the mode-n unfolding matrices while the nuclear norm minimization is applied on the n-unfolding matrices[5]. As a result, our proposed formulation (8) can be considered as a non-trivial extension and combination of techniques considered in [43] and [35], where on the one hand, we exploit l_1-norm instead of nuclear norm minimization [35] and on the other hand, similar to [43], the dictionary learning technique is employed. It is worth mentioning that ignoring the sparsity constraints, we come up to the algorithm proposed in [43].

According to works by [15, 35, 49], the relation between the core tensor $\widehat{\underline{\mathbf{G}}}^{(n)}$ and the matricized tensor can be represented by

$$\mathbf{X}_{H\langle n\rangle} = \widehat{\mathbf{G}}^{(n)}_{(2)} \left(\widehat{\mathbf{G}}^{(\neq n)}_{\langle 2\rangle}\right)^T, \quad \forall n, \tag{9}$$

[4] The MATLAB implementation of this dictionary can be found in https://github.com/kvdeepak/ksvd-denoising.

[5] Note that mode-n unfolding and n-unfolding matrices are related to the Tucker and the TT decompositions, respectively.

where $\widehat{\mathbf{G}}^{(\neq n)}$ is a 3rd-order tensor of size $R_n \times \prod_{i=1,i\neq n}^{N} I_i \times R_{n-1}$ obtained by merging all other core tensors whose slice matrices are computed as

$$\underline{\widehat{\mathbf{G}}}^{(\neq n)}\left(\overline{i_{n+1}\cdots i_N i_1\cdots i_{n-1}}\right) = \prod_{j=n+1}^{N} \underline{\widehat{\mathbf{G}}}^{(j)}\left(i_j\right) \prod_{j=1}^{n-1} \underline{\widehat{\mathbf{G}}}^{(j)}\left(i_j\right).$$

For brevity of notation, in the rest of the paper we define $\widehat{\mathbf{G}}^{(n)} := \widehat{\mathbf{G}}^{(n)}_{(2)}$ and $\widehat{\mathbf{G}}^{(\neq n)} := \widehat{\mathbf{G}}^{(\neq n)}_{<2>}$.

By employing formula (9) and taking into account the fact that the core tensors are independent, we can optimize the core tensors in optimization problem (8) independently while keeping other core tensors fixed as follows:

$$\begin{aligned} \min_{\{\widehat{\mathbf{G}}^{(n)}\}} \quad & \mathcal{H}\left(\widehat{\mathbf{G}}^{(n)}\right) + \eta \left\|\mathbf{C}^{(n)}\right\|_1 \\ \text{s.t.} \quad & \widehat{\mathbf{G}}^{(n)} = \mathbf{D}^{(n)}\mathbf{C}^{(n)}, \end{aligned} \tag{10}$$

for $n = 1, 2, \ldots, N$, where

$$\mathcal{H}\left(\widehat{\mathbf{G}}^{(n)}\right) = \left\|\mathbf{P}_{\mathbf{\Omega}_{H\,\langle n\rangle}}\left(\widehat{\mathbf{X}}_{H\,\langle n\rangle} - \widehat{\mathbf{G}}^{(n)}\left(\widehat{\mathbf{G}}^{(\neq n)}\right)^T\right)\right\|_F^2.$$

The augmented Lagrangian function corresponding to the constrained optimization problem (10), can be constructed as

$$\begin{aligned} \mathcal{L}\left(\widehat{\mathbf{G}}^{(n)}, \mathbf{C}^{(n)}, \mathbf{B}\right) = \mathcal{H}\left(\widehat{\mathbf{G}}^{(n)}\right) + \eta\left\|\mathbf{C}^{(n)}\right\|_1 + \left\langle \mathbf{B}, \widehat{\mathbf{G}}^{(n)} - \mathbf{D}^{(n)}\mathbf{C}^{(n)}\right\rangle \\ + \frac{\lambda}{2}\left\|\widehat{\mathbf{G}}^{(n)} - \mathbf{D}^{(n)}\mathbf{C}^{(n)}\right\|_F^2, \end{aligned} \tag{11}$$

where $\mathbf{B}$ is a matrix representing the Lagrangian multipliers and λ is penalty parameter. While the method works well for λ between 0.1 and 1.1. We set $\lambda = 0.5$ in our simulations. All hyper-parameters were optimized to provide the best performance.

Based on Lagrangian function (11), the ADMM [9] update rules for solving (8) are straightforwardly converted to simpler optimization problems as follows:

$$\widehat{\mathbf{G}}^{(n)}_{k+1} = \min_{\widehat{\mathbf{G}}^{(n)}_k} \mathcal{H}\left(\widehat{\mathbf{G}}^{(n)}_k\right) + \frac{\lambda}{2}\left\|\widehat{\mathbf{G}}^{(n)}_k - \mathbf{D}^{(n)}\mathbf{C}^{(n)}_k + \mathbf{B}^{(k)}\right\|_F^2, \tag{12}$$

$$\mathbf{C}^{(n)}_{k+1} = \min_{\mathbf{C}^{(n)}_k} \left\|\mathbf{C}^{(n)}_k\right\|_1 + \frac{\lambda}{2}\left\|\widehat{\mathbf{G}}^{(n)}_k - \mathbf{D}^{(n)}\mathbf{C}^{(n)}_k + \mathbf{B}^{(k)}\right\|_F^2, \tag{13}$$

$$\mathbf{B}^{(k+1)} = \mathbf{B}^{(k)} + \left(\widehat{\mathbf{G}}^{(n)}_{k+1} - \mathbf{D}^{(n)}\mathbf{C}^{(n)}_{k+1}\right). \tag{14}$$

It is worth to note that the ADMM algorithm is a workhorse approach for solving a variety of optimization problems such as sparse inverse covariance selection, generalized and group lasso problems, etc. For a comprehensive study of this technique and related problems, see [9]. Moreover, this strategy has been utilized for solving many tensor completion problems: please see [43, 50–53], and for a comprehensive list of references see the review papers [3, 4] and the references therein.

4.1. Solving the sub optimization problem (12)

The first optimization sub-problem (12) can be solved via the standard gradient descent method. To be more precise, the gradient of its objective function of (12) in kth iteration is

$$\begin{aligned} \nabla_{\widehat{\mathbf{G}}^{(n)}_k} J = \mathbf{P}_{\mathbf{\Omega}_{H\,\langle n\rangle}}\left(\widehat{\mathbf{G}}^{(n)}_k\left(\widehat{\mathbf{G}}^{(\neq n)}_k\right)^T - \mathbf{X}_{H\,\langle n\rangle}\right)\widehat{\mathbf{G}}^{(\neq n)} \\ + \lambda\left(\widehat{\mathbf{G}}^{(n)}_k - \mathbf{D}^{(n)}\mathbf{C}^{(n)}_k + \mathbf{B}^{(k)}\right), \end{aligned} \tag{15}$$

where $J = \mathcal{L}\left(\widehat{\mathbf{G}}^{(n)}_k, \mathbf{C}^{(n)}, \mathbf{B}\right)$ and the first and second terms of (15) are the gradient of the first and second terms of the objective function of optimization problem (12), respectively. The gradient of the first term of

(15) has been derived in [35] and the gradient of the second term can be derived by the following straightforward computations:

$$\frac{\lambda}{2}\left\|\widehat{\mathbf{G}}_k^{(n)} - \mathbf{D}^{(n)}\mathbf{C}_k^{(n)} + \mathbf{B}^{(n)}\right\|_F^2 = \frac{\lambda}{2}\left(\left\|\widehat{\mathbf{G}}_k^{(n)}\right\|_F^2 + \left\|\mathbf{B}^{(n)} - \mathbf{D}^{(n)}\mathbf{C}_k^{(n)}\right\|_F^2 + 2\mathrm{Tr}\left(\widehat{\mathbf{G}}_k^{(n)\,T}\left(\mathbf{B}^{(n)} - \mathbf{D}^{(n)}\mathbf{C}_k^{(n)}\right)\right)\right),$$

and taking the gradient with respect to $\mathbf{G}_k^{(n)}$.

4.2. Solving the sub-optimization problem (13)

In order to solve optimization problem (13), we applied accelerated proximal gradient (APG) method [54, 55]. Assuming that the $E\left(\mathbf{C}_k^{(n)}\right) = \left\|\mathbf{C}_k^{(n)}\right\|_1 + F\left(\mathbf{C}_k^{(n)}\right)$, we have the following cost function:

$$F\left(\mathbf{C}_k^{(n)}\right) = \frac{\lambda}{2}\left\|\widehat{\mathbf{G}}_k^{(n)} + \mathbf{B}^{(k)} - \mathbf{D}^{(n)}\mathbf{C}_k^{(n)}\right\|_F^2.$$

The first step in the proximal gradient algorithms is considering an auxiliary proximal variable $\mathbf{Z}_k$, and computing a quadratic approximation of $E(\mathbf{C}_k^{(n)})$ around $\mathbf{Z}_k$ using the following equation:

$$H\left(\mathbf{C}_k^{(n)}, \mathbf{Z}_k\right) = \left(\left\|\mathbf{C}_k^{(n)}\right\|_1 + F(\mathbf{Z}_k)\right) + \left\langle \nabla_{\mathbf{C}_k} F(\mathbf{Z}_k), \mathbf{C}_k^{(n)} - \mathbf{Z}_k \right\rangle + \frac{c}{2}\left\|\mathbf{C}_k^{(n)} - \mathbf{Z}_k\right\|_F^2, \tag{16}$$

where the gradient $\nabla_{\mathbf{C}_k} F(\mathbf{Z}_k)$ is computed straightforwardly by expanding $F(\mathbf{Z}_k)$ and taking gradient with respect to $\mathbf{C}_k$ as

$$\nabla_{\mathbf{C}_k} F(\mathbf{Z}_k) = \lambda\left(\mathbf{D}^{(n)\,T}\mathbf{D}^{(n)}\mathbf{C}_k^{(n)} - \mathbf{D}^{(n)\,T}\left(\widehat{\mathbf{G}}_k^{(n)} + \mathbf{B}^{(k)}\right)\right). \tag{17}$$

The parameter $c > 0$ in (16) is a given parameter and in our simulation we set $c = \left\|\mathbf{D}^{(n)}\right\|_F^2$. Substituting $\mathbf{W}_k = \mathbf{Z}_k - \nabla_{\mathbf{C}_k} f(\mathbf{Z}_k)/c$ in (16), we obtained

$$H\left(\mathbf{C}_k^{(n)}, \mathbf{Z}_k\right) = T(\mathbf{C}_k^{(n)}, \mathbf{W}_k) - \frac{1}{2c}\|\nabla_{\mathbf{C}_k} f(\mathbf{Z}_k)\|_F^2, \tag{18}$$

where

$$T(\mathbf{C}_k^{(n)}, \mathbf{W}_k) = \left\|\mathbf{C}_k^{(n)}\right\|_1 + \frac{c}{2}\left\|\mathbf{C}_k^{(n)} - \mathbf{W}_k\right\|_F^2. \tag{19}$$

As a result, $T(\mathbf{C}_k^{(n)}, \mathbf{W}_k)$ is equivalent to $H\left(\mathbf{C}_k^{(n)}, \mathbf{Z}_k\right)$ up to a constant and it can be minimised instead of $H\left(\mathbf{C}_k^{(n)}, \mathbf{Z}_k\right)$. In the proximal gradient algorithms, instead of minimizing $E(\mathbf{C}_k^{(n)})$, which is a non-differentiable optimization problem, the objective function $T(\mathbf{C}_k^{(n)}, \mathbf{Z}_k)$ is minimized for a sequence of proximal variables $\mathbf{Z}_k^j$. This leads to the following minimization problem which can be solved iteratively:

$$\mathbf{C}_{k,j}^{(n+1)} = \underset{\mathbf{C}_k^{(n)}}{\arg\min} \left\|\mathbf{C}_k^{(n)}\right\|_1 + \frac{c}{2}\left\|\mathbf{C}_k^{(n)} - \mathbf{W}_k^{j+1}\right\|_F^2, \tag{20}$$

where $\mathbf{W}_k^{j+1} = \mathbf{Z}_k^j - \nabla_{\mathbf{C}_k} f\left(\mathbf{Z}_k^j\right)/c$. Finally, we obtained the following closed form solution:

$$\mathbf{C}_{k,j+1}^{(n+1)} = \mathrm{soft}\left(\mathbf{W}_k^{j+1}, \frac{1}{c}\right), \tag{21}$$

where the soft thresholding operator is defined as $\mathrm{soft}(g,\tau) = \mathrm{sign}(g)\max(|g| - \tau, 0)$. Note that in (20), the soft thresholding operator is applied in an element-wise manner to all components of the matrix $\mathbf{W}_k^{j+1}$.

Algorithm 1. Algorithm for tensor ring decomposition with sparse representation (TRDSR).

```
   Input: An incomplete data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, the observation index tensor $\underline{\mathbf{\Omega}}$, TR-ranks $(R_0, R_2, \ldots, R_{N-1})$,
          for simplicity, we assume that $R_n = R \ \forall \ n$, a window size $\tau$ and regularization parameter $\lambda > 0$.
   Output: Completed data tensor $\widehat{\underline{\mathbf{X}}}$
 1 Hankelize the data tensor $\underline{\mathbf{X}}$ and $\underline{\mathbf{\Omega}}$ as $\underline{\mathbf{X}}_H$ and $\underline{\mathbf{\Omega}}_H$
 2 Initialize the TR core tensors for the Hankelized tensor $\underline{\mathbf{X}}_H$ as $\widehat{\underline{\mathbf{G}}}^{(1)}, \widehat{\underline{\mathbf{G}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{G}}}^{(N)}$ with Gaussian random tensors
 3 while A stopping criterion is not satisfied do
 4     for $n = 1, 2, \ldots, N$ do
 5         Compute dictionary $\mathbf{D}^{(n)}$ for mode-2 matricization of the core tensor $\widehat{\underline{\mathbf{G}}}^{(n)}$
 6         Solve optimization Problem (10) for $\widehat{\mathbf{G}}^{(n)}$ and $\mathbf{C}^{(n)}$ using the update ADMM rules (12)–(14)
 7     end
 8     Compute $\widehat{\underline{\mathbf{X}}}_H = \ll \widehat{\underline{\mathbf{G}}}^{(1)}, \widehat{\underline{\mathbf{G}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{G}}}^{(N)} \gg$
 9     Compute $\widehat{\underline{\mathbf{X}}}_H = \mathbf{P}_{\mathbf{\Omega}_H}\left(\widehat{\underline{\mathbf{X}}}_H\right) + \mathbf{P}_{\mathbf{\Omega}_H^{\perp}}\left(\widehat{\underline{\mathbf{X}}}_H\right)$
10     De-Hankelize $\widehat{\underline{\mathbf{X}}}_H$
11 end
```

There are several possible options for updating the proximal variable $\mathbf{Z}_k^{j+1}$, but in this paper we used the accelerated one introduced in [54] as follows:

$$\begin{cases} t^{k+1} = \dfrac{1+\sqrt{4t^{2j}+1}}{2}, \\ \mathbf{Z}_k^{j+1} = \mathbf{C}_{k,j}^{(n+1)} + \dfrac{t^k - 1}{t^{k+1}}\left(\mathbf{C}_{k,j+1}^{(n+1)} - \mathbf{C}_{k,j}^{(n+1)}\right), \end{cases}$$

where $t^1 = 1$. The whole procedure is summarized in the pseudo-code of algorithm 1. The stopping criterion which we used was that the relative error be less than a predefined tolerance or the maximum number of iterations is reached.

5. Discussion on computational complexity

Most of state-of-the-art tensor completion algorithms exploit the nuclear norm minimization formulation in which the computation of the singular value decomposition (SVD) is required multiple times. This increases their computational complexity and also makes them relatively slow. On the contrary, we avoid SVD computation because we do not exploit the nuclear norm minimization formulation. For the sake of simplicity, let us assume $I_1 = I_2 = \cdots = I_N = I$ and $R_1 = R_2 = \cdots = R_N = R$, then the computational complexity of TR-ALS [34] and TRLRF [49, 51] are $\mathcal{O}(pNR^4I^N + NR^6)$ and $\mathcal{O}(NR^2I^N + NR^6)$, respectively, where p is constant between 0 and 1. The main operation in our algorithms is matrix–matrix multiplication in (15) where we need to multiply matrices $\widehat{\mathbf{G}}_k^{(n)} \in \mathbb{R}^{R^2\times I_n}$ and $\widehat{\mathrm{G}}_k^{(\neq n)} \in \mathbb{R}^{\prod_{i\neq n} I_i \times R^2}$ which costs $\mathcal{O}(R^2I^N)$. This indicates the lower complexity of our algorithm compared with others

6. Experiments

In this section, we evaluate the proposed algorithm referred to as TRDSR using real datasets including images, videos, and time series signals. All simulations were conducted on a laptop computer with 2.60 GHz Intel(R) Core(TM) i7-5500U processor and 16GB memory. The *relative squared error* (RSE) and *peak signal-to-noise ratio* (PSNR) are used to evaluate and compare the performance of various algorithms. The RSE is defined as

$$\mathrm{RSE} = \frac{\left\|\widehat{\underline{\mathbf{X}}} - \underline{\mathbf{X}}\right\|_F}{\|\underline{\mathbf{X}}\|_F},$$

where $\underline{\mathbf{X}}$ and $\widehat{\underline{\mathbf{X}}}$ are the original data tensor with all observations and the reconstructed data tensor, respectively, while the PSNR is defined as

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

where

$$\mathrm{MSE} = \left\|\widehat{\underline{\mathbf{X}}} - \underline{\mathbf{X}}\right\|_F^2 / \mathrm{num}\left(\underline{\mathbf{X}}\right),$$

Figure 6. Colored images used in numerical experiments.

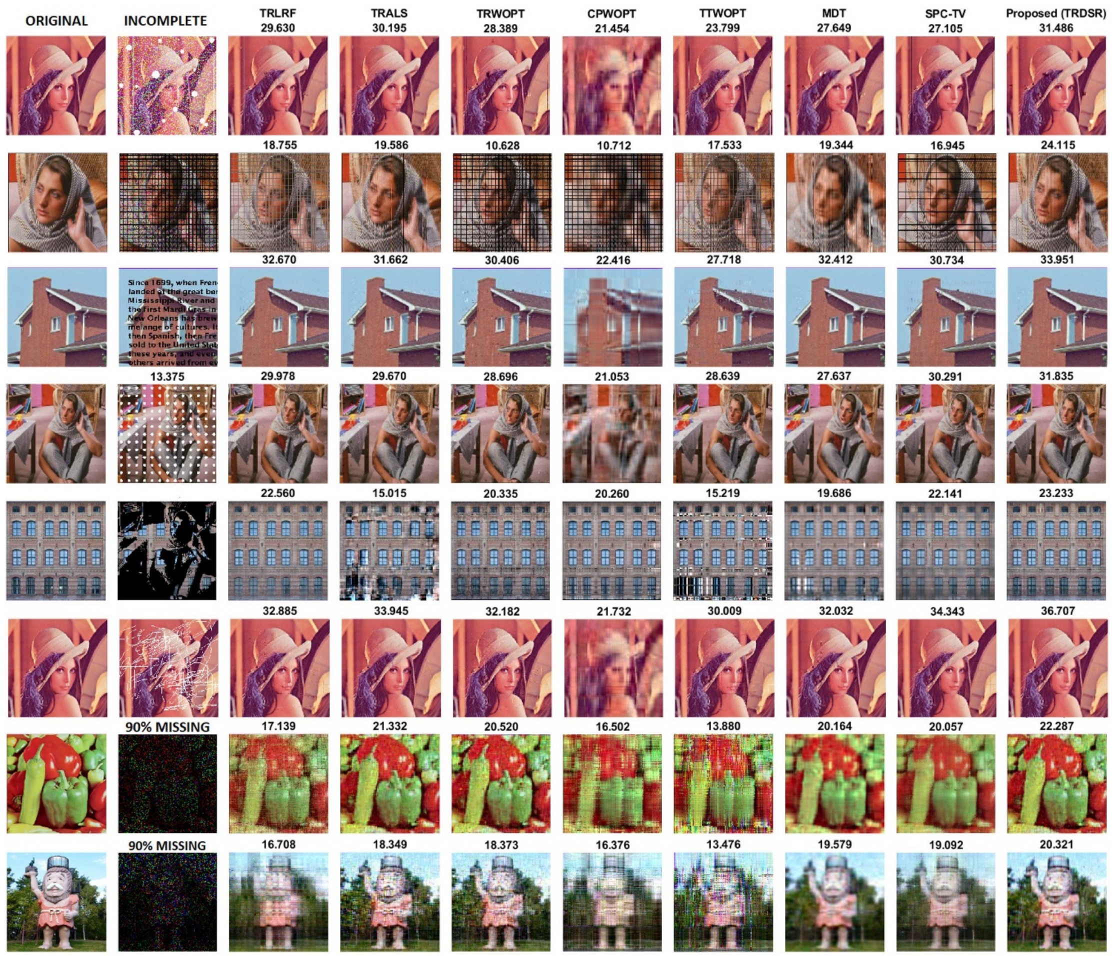


Figure 7. Visual and PSNR comparison of reconstructed images using different tensor completion algorithms. In most cases, our algorithm provides the best performance.

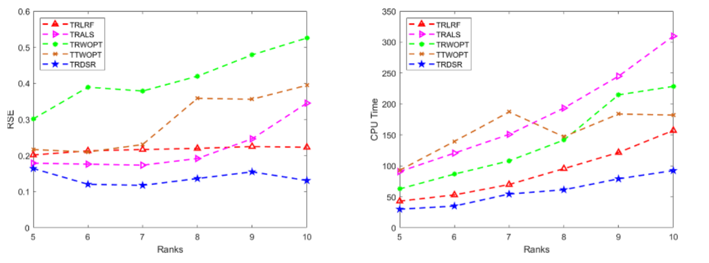
and 'num(.)' denotes the number of components of the tensor $\underline{\mathbf{X}}$. We used the Matlab toolbox poblano toolbox 1.1 [56] for solving the underlying non-linear optimization problems.

Example 1. (Image recovery) In these experiments, we used the benchmark images 'Lena', 'Barbara', 'House', 'Facade', 'Peppers' and 'Giant' as shown in figure 6. All color images are of size $256 \times 256 \times 3$. We have tried different types of missing patterns. For example, in the case of 'Lena' image, we removed randomly 20% of pixels grouped in form of white big spots with some holes in the picture (first image) or continuous scratching the image (sixth image) both of them are considered as structured missing. For the 'Barbara' image (second image), we removed more than 40% of whole columns and rows of pixels randomly. The last two 'Pepper' and 'Giant' images have 90% of missing pixels. The missing patterns of the 'House', 'Facade' images were also structured distortions. Using the window size $\tau = (252, 252, 3)$, the incomplete images were first Hankelized into a 5th-order tensor of size $252 \times 5 \times 252 \times 5 \times 3$ and represented in TR format before applying the completion algorithms. We compared the performance of our algorithm with some recent tensor completion algorithms under various conditions and for different missing rates. The algorithms which we used were TRLRF [57], TRALS [34], TR-WOPT [35], TT-WOPT [55], CP-WOPT [58], SPC-TV [59] and MDT [5]. In order to provide a fair comparison with other algorithms, we use the same TT/TR

Table 1. Performance comparison using PSNR and SSIM for benchmark images.

IMAGE	Results	TRLRF	MDT	TRALS	TRWOPT	TTWOPT	CPWOPT	SPC-TV	Proposed (TRDSR)
Lena	PSNR	29.63	27.65	30.19	28.39	23.80	21.45	27.11	**31.49**
	SSIM	0.9156	0.8233	0.9024	0.9048	0.8130	0.6057	0.9049	**0.9422**
Barbara	PSNR	18.76	19.34	19.59	10.63	17.53	10.71	16.96	**24.12**
Lines	SSIM	0.5251	0.4700	0.7300	0.2525	0.4619	0.1242	0.5199	**0.7702**
House	PSNR	32.67	32.41	31.66	30.49	27.78	22.73	30.73	**33.95**
	SSIM	0.9186	0.8696	0.9050	0.8411	0.8601	0.6287	0.9075	**0.9353**
Barbara	PSNR	29.98	26.77	29.67	28.70	28.64	21.05	30.29	**31.84**
White Circles	SSIM	0.9200	0.7813	0.9177	0.8999	0.9126	0.5713	0.9279	**0.9456**
Occluded	PSNR	22.56	19.91	16.18	20.699	15.22	22.14	20.06	**23.24**
Windows	SSIM	0.7300	0.6258	0.5721	0.6560	0.5732	0.6704	0.7041	**0.7425**
Lena	PSNR	32.89	32.03	33.95	32.18	30.00	21.73	34.34	**36.71**
Scratched	SSIM	0.9159	0.9236	0.9390	0.9011	0.8932	0.6090	0.9513	**0.9677**
Peppers	PSNR	17.85	20.11	21.33	20.52	13.88	16.50	20.06	**22.19**
90% Missing	SSIM	0.2772	0.5619	0.5056	0.4689	0.1569	0.2617	**0.6370**	0.5506
Giant	PSNR	16.71	19.58	18.35	18.37	13.48	16.38	19.09	**20.32**
90% Missing	SSIM	0.3110	0.4631	0.3986	0.3756	0.1550	0.2531	0.4797	**0.5272**

Note: The bold values mean that the results of our proposed algorithm outperform other algorithms.



(a) Comparison of RSE vs TR Ranks R (b) Comparison of CPU Time vs TR Ranks (in seconds)

Figure 8. Comparison of RSE and CPU time with TR Ranks of some tensor completion algorithms for 'Lena' image with 90% missing components.

ranks for TRLRF, TRALS, TR-WOPT, TT-WOPT, CP-WOPT algorithms and tune the hyper-parameters of each algorithm to make performance as best as possible. The MDT and the SPC-TV algorithms were able to tune their ranks automatically. The reconstructed images are displayed in figure 7 and detailed PSNR and SSIM comparison are reported in table 1. From the results, it is seen that our proposed algorithm, in general, outperforms the other algorithms in most of the cases for various scenarios and missing patterns. Also, we examined our proposed algorithms in terms of robustness to TR rank selection. To this end, we considered the 'Lena' image with 90% random missing pixels and compared the RSE. The results are reported in figure 8(a) which show that our TRDSR algorithm achieves the lowest RSE compared to the other completion methods and also it is relatively insensitive to selected TR ranks. We should mention that one of the main challenging task in tensor completion problem is a good selection of rank and a larger rank does not always necessarily provides better results. This is mainly because of the over-fitting problem where the model is not selected appropriately. Because of this issue, it is of interest to have a tensor completion algorithm which is less sensitive to rank selection. Actually, in this experiment we wanted to show that different from other algorithms, ours is less sensitive to the choice of TR-ranks and it is more stable to various TR-ranks. We also compared the computational time of our method with other existing methods. The results are reported in figure 8(b). It is seen that our algorithm is the fastest algorithm while at the same time, it provides better performance. Finally, we applied our proposed algorithm to an MRI images. Here, we mainly exploited four different kinds of structural missing patterns (see figure 9). In the first experiment, we scratched the data image (first image in figure 9), in the second one we removed 10% of pixels and 10% of columns and rows randomly (second image in figure 9), in the third experiment, we removed 40% of pixels and putting some spots in the image (third image in figure 9), and in the fourth experiment, we removed 40% columns and rows (fourth image in figure 9). The recovered images together with indicated corresponding PSNR index are reported in figure 9. Our computer experiments indicate that the proposed algorithm also provides good results for recovering real-life medical images. Moreover, we compared the performance of our algorithm for the MRI image with some other tensor completion algorithm. These results are reported in figure 10. Here again, our algorithms provided better performance.

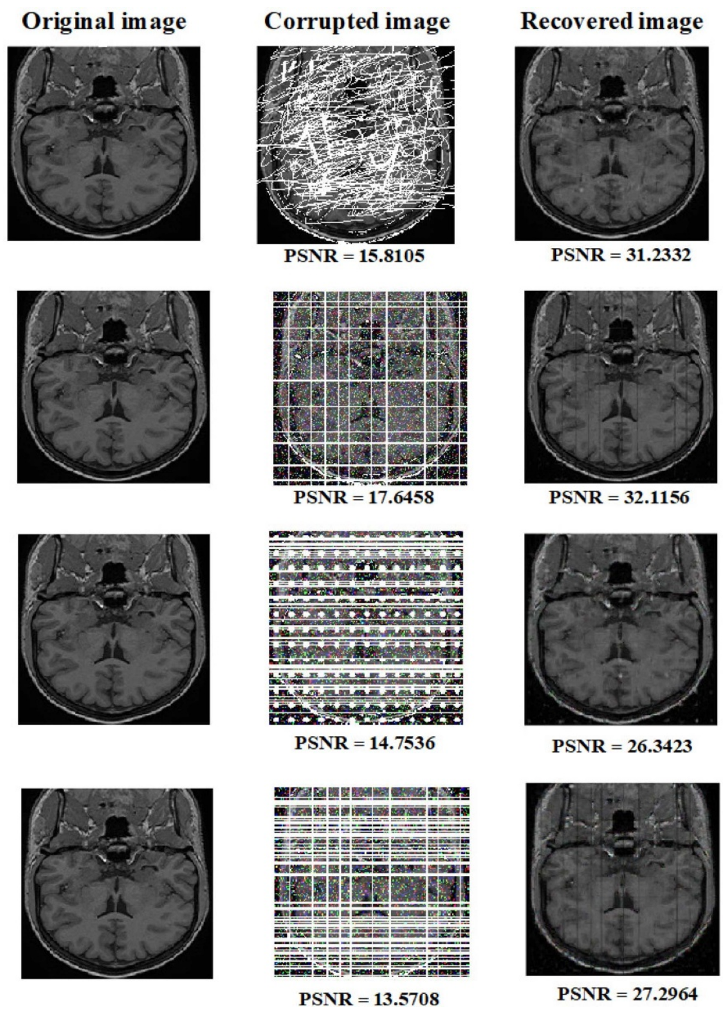


Figure 9. Performance of the proposed TRDSR algorithm in recovering an MRI image with different kind of missing pixels.

Example 2. In this experiment, we consider the video recovery task. The dataset which we used was the GunShot video [34, 60] of the size[6] $100 \times 260 \times 3 \times 85$. We removed 80% of voxels randomly. We Hankelized video frames into an 8th-order tensor of the size $252 \times 9 \times 96 \times 5 \times 3 \times 1 \times 83 \times 3$ for reconstruction using the window sizes $\tau = (252, 96, 3, 83)$. The reconstructed results of TR-ALS, MDT, SPC and our proposed TRDSR algorithm are displayed in figure 11. From the results, the superiority of our technique is visible.

As a second experiment on videos, we considered the smoke video[7] [5] which is composed of 64 frames of size $90 \times 160 \times 3$ with tensor size of $90 \times 160 \times 3 \times 64$. We removed 95% of voxels randomly to be used in our simulations. The incompleted data tensor was Hankelized into an 8th-order tensor of size $85 \times 6 \times 155 \times 6 \times 3 \times 1 \times 60 \times 5$ using the window sizes $\tau = (85, 155, 3, 60)$. The reconstructed results of the MDT [5] and our proposed TRDSR algorithm are displayed in figure 12.

Example 3. In this simulation, we evaluate our proposed algorithm for reconstruction of time series generated by the Lorentz system which is illustrated in figure 13. The size of the Lorentz system was 1×4600, which we reshaped to 3rd-order tensors of size $46 \times 10 \times 10$. We considered two kinds of missing patterns as follows:

- Removing some parts of the signal in the first and middle parts along with a Gaussian noise (see figure 13).
- Removing 20% of samples and also removing 300 samples in the middle and last parts of the signal (see figure 14).

In both above cases, we used window size $\tau = (5, 44, 5)$ and Hankelized the incomplete 3rd-order tensor into a 6th-order tensor of size $5 \times 6 \times 44 \times 3 \times 5 \times 6$. The constructed signals using our proposed algorithm

[6] Video includes 80 frames of RGB frames of size $100 \times 260 \times 3$.
[7] The video can be downloaded from www.nhk.or.jp/archives/creative/material/category-list.html?i = 22.

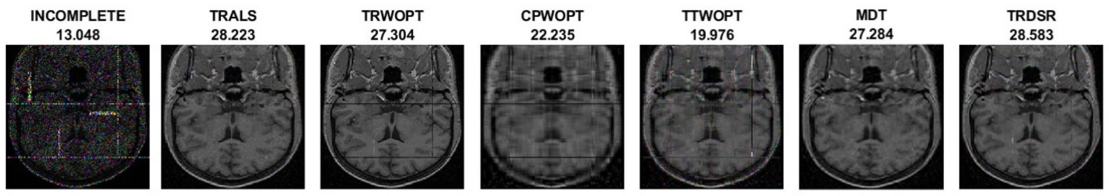


Figure 10. Visual and PSNR comparison of MRI image reconstructed using different tensor completion algorithms.

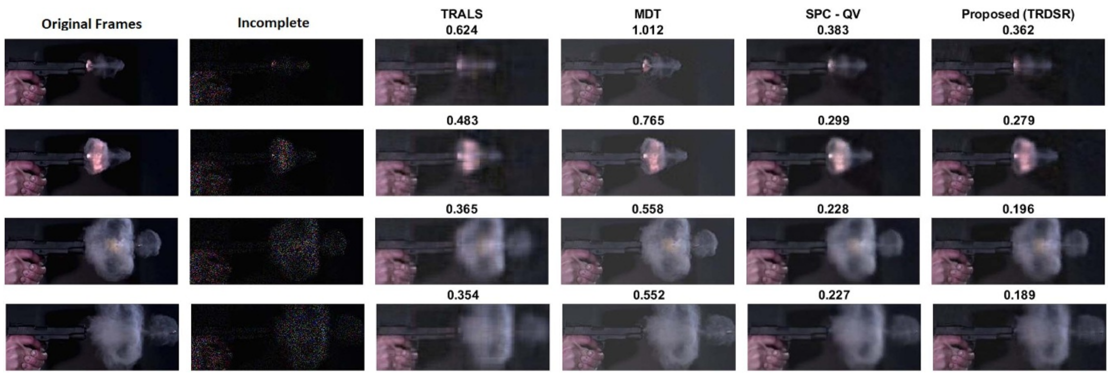


Figure 11. Comparison of RSE on the GunShot Video Reconstruction with 80% random missing pixels.

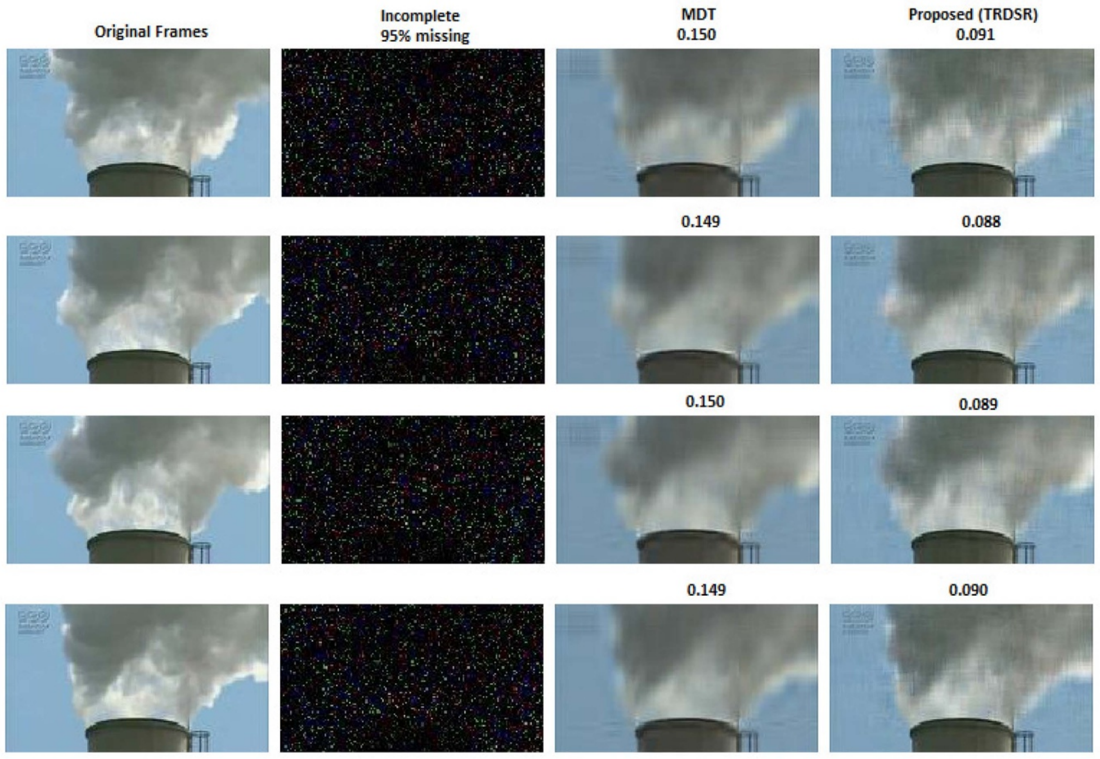


Figure 12. Comparison of RSE on the Smoke Video Reconstruction with 95% random missing pixels.

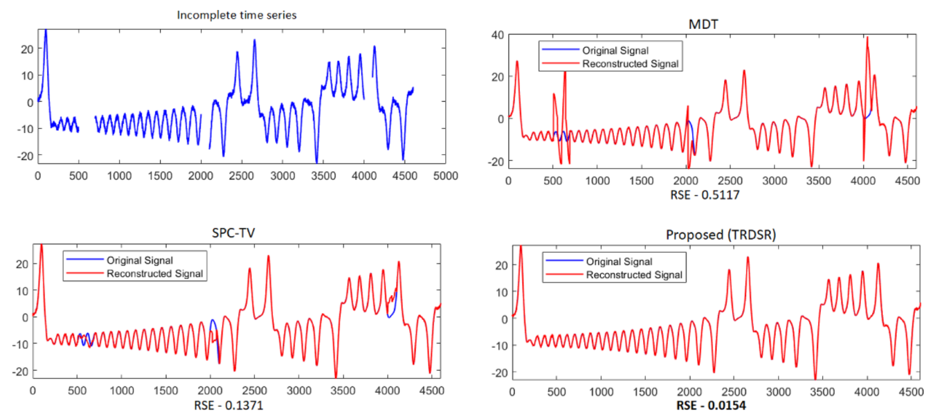


Figure 13. Signal reconstruction (generated by Lorentz system) using different algorithms.

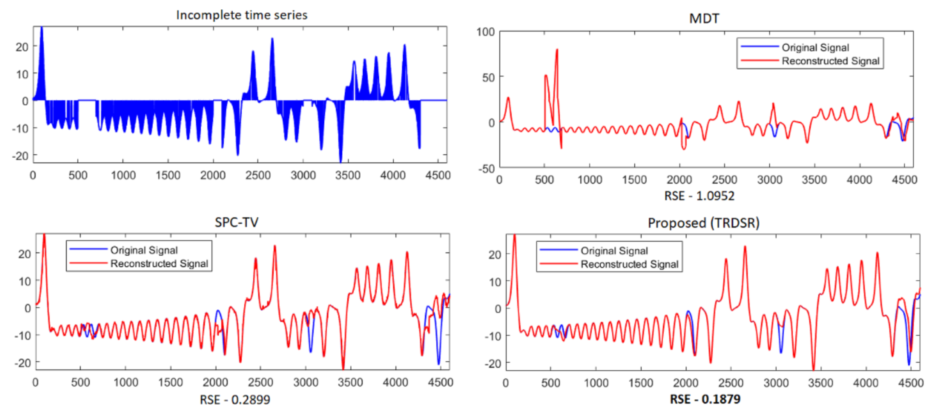


Figure 14. Comparison of performance of signal denoising and prediction (generated by Lorentz system corrupted by additive noise, where middle and last samples are missing) using different algorithms.

compared to the MDT and the SPC-TV algorithms are displayed in figures 13 and 14, respectively. Our algorithm achieved the lowest RSE and this confirms the effectiveness and applicability of our method.

7. Conclusion

In this paper, a new and efficient tensor completion algorithm was proposed for recovering data tensors with random and/or structured missing entries. The main idea is a prior Hankelization of the incomplete data tensor after which the data tensor is completed through learning the core tensors of the TR representation with sparse constraints. The ADMM algorithm and APG approaches were adopted to solve the underlying optimization problems. Extensive simulations show that our proposed method is capable of recovering incomplete data tensors with different types of structured and random missing elements. Our algorithm exhibits higher performance in comparison to many existing tensor decomposition methods, while providing lower computational cost in comparison to other tensor completion approaches.

Data availability statement

Data sharing is not applicable to this article as no new data were created or analysed in this study. The analyzed data are available online.

The data of images and videos are available at https://youtu.be/7y9apnbI6GA, https://www.nhk.or.jp/archives/creative/material/category-list.html?i = 22 and https://drive.google.com/file/d/15xk67wYZ9GI2Kn93g_aaA4CsmgnqGlHE/view.

Acknowledgment

The authors would like to thank two anonymous reviewers for their constructive comments which have greatly improved the quality of the paper. This research was also partially supported by the Ministry of Education and Science of the Russian Federation (Grant No. 14.756.31.0001).

ORCID iDs

Maame G Asante-Mensah https://orcid.org/0000-0003-0400-2866
Salman Ahmadi-Asl https://orcid.org/0000-0002-2614-0146
Andrzej Cichocki https://orcid.org/0000-0002-8364-7226

References

[1] Liu J, Musialski P, Wonka P and Ye J 2012 Tensor completion for estimating missing values in visual data *IEEE Trans. Pattern Anal. Mach. Intell.* **35** 208–20
[2] Asif M T, Mitrovic N, Dauwels J and Jaillet P 2016 Matrix and tensor based methods for missing data estimation in large traffic networks *IEEE Trans. Intell. Transp. Syst.* **17** 1816–25
[3] Song Q, Ge H, Caverlee J and Hu X 2019 Tensor completion algorithms in big data analytics *ACM Trans. Knowl. Discovery Data* **13** 1–48
[4] Long Z, Liu Y, Chen L and Zhu C 2019 Low rank tensor completion for multiway visual data *Signal Process.* **155** 301–16
[5] Yokota T, Erem B, Guler S, Warfield S K and Hontani H 2018 Missing slice recovery for tensors using a low-rank model in embedded space *Proc. Conf. on Computer Vision and Pattern Recognition* pp 8251–9
[6] Yokota T and Hontani H 2017 Simultaneous visual data completion and denoising based on tensor rank and total variation minimization and its primal-dual splitting algorithm *Proc. Conf. on Computer Vision and Pattern Recognition* pp 3732–40
[7] Chen Y and Chi Y 2013 Spectral compressed sensing via structured matrix completion (arXiv:1304.4610)
[8] Sedighin F, Cichocki A, Yokota T and Shi Q 2020 Matrix and tensor completion in multiway delay embedded space using tensor train, with application to signal reconstruction *IEEE Signal Process. Lett.* **27** 810–14
[9] Boyd S *et al* 2011 Distributed optimization and statistical learning via the alternating direction method of multipliers *Found. Trends Mach. Learn.* **3** 1–122
[10] Cichocki A, Lee N, Oseledets I V, Phan A-H, Zhao Q and Mandic D P 2016 Tensor networks for dimensionality reduction and large-scale optimization: part 1 perspectives and challenges *Found. Trends Mach. Learn.* **9** 249–429
[11] Oseledets I V 2011 Tensor-train decomposition *SIAM J. Sci. Comput.* **33** 2295–2317
[12] Khoromskij B N 2011 $O(d\log N)$-quantics approximation of n-d tensors in high-dimensional numerical modeling *Constr. Approx.* **34** 257–80
[13] Espig M, Hackbusch W, Handschuh S and Schneider R 2011 Optimization problems in contracted tensor networks *Comput. Vis. Sci.* **14** 271–85
[14] Espig M, Naraparaju K K and Schneider J 2012 A note on tensor chain approximation *Comput. Vis. Sci.* **15** 331–44
[15] Zhao Q, Zhou G, Xie S, Zhang L and Cichocki A 2016 Tensor ring decomposition (arXiv:1606.05535)
[16] White S R 1992 Density matrix formulation for quantum renormalization groups *Phys. Rev. Lett.* **69** 2863
[17] White S R 1993 Density-matrix algorithms for quantum renormalization groups *Phys. Rev.* B **48** 10345
[18] Schollwöck U 2011 The density-matrix renormalization group in the age of matrix product states *Ann. Phys.* **326** 96–192
[19] Orús R 2014 A practical introduction to tensor networks: matrix product states and projected entangled pair states *Ann. Phys.* **349** 117–58
[20] Bridgeman J C and Chubb C T 2017 Hand-waving and interpretive dance: an introductory course on tensor networks *J. Phys.* A **50** 223001
[21] Torlai G, Wood C J, Acharya A, Carleo G, Carrasquilla J and Aolita L 2020 Quantum process tomography with unsupervised learning and tensor networks (arXiv:2006.02424)
[22] Biamonte J and Bergholm V 2017 Tensor networks in a nutshell (arXiv:1708.00006)
[23] Biamonte J, Kardashin A and Uvarov A 2018 Quantum machine learning tensor network states (arXiv:1804.02398)
[24] Cichocki A, Lee N, Oseledets I, Phan A-H, Zhao Q and Mandic D P 2017 Tensor networks for dimensionality reduction and large-scale optimizations: Part 2 applications and future perspectives *Found. Trends Mach. Learn.* **9** 431–673
[25] Cichocki A 2014 Era of big data processing: a new approach via tensor networks and tensor decompositions (arXiv:1403.2048)
[26] Reyes J and Stoudenmire M 2020 A multi-scale tensor network architecture for classification and regression (arXiv:2001.08286)
[27] Pippan P, White S R and Evertz H G 2010 Efficient matrix-product state method for periodic boundary conditions *Phys. Rev.* B **81** 081103
[28] Yang Y, Krompass D and Tresp V 2017 Tensor-train recurrent neural networks for video classification (arXiv:1707.01786)
[29] Kuznetsov M, Polykovskiy D, Vetrov D P and Zhebrak A 2019 A prior of a googol Gaussians: a tensor ring induced prior for generative models *Advances in Neural Information Processing Systems* pp 4102–12
[30] Stoudenmire E and Schwab D J 2016 Supervised learning with tensor networks *Advances in Neural Information Processing Systems* pp 4799–807
[31] Novikov A, Podoprikhin D, Osokin A and Vetrov D P 2015 Tensorizing neural networks *Advances in Neural Information Processing Systems* pp 442–50
[32] Tjandra A, Sakti S and Nakamura S 2017 Compressing recurrent neural network with tensor train *2017 Int. Conf. on Neural Networks (IJCNN)* (IEEE) pp 4451–8
[33] Pan Y, Xu J, Wang M, Ye J, Wang F, Bai K and Xu Z 2019 Compressing recurrent neural networks with tensor ring for action recognition *Proc. Conf. on Artificial Intelligence* vol 33 pp 4683–90
[34] Wang W, Aggarwal V and Aeron S 2017 Efficient low rank tensor ring completion *Proc. IEEE Int. Conf. on Computer Vision* pp 5697–705

[35] Yuan L, Cao J, Zhao X, Wu Q and Zhao Q 2018 Higher-dimension tensor completion via low-rank tensor ring decomposition *2018 Asia-Pacific Signal and Information Processing Association Annual Summit Conf. (APSIPA ASC)* (IEEE) pp 1071–6

[36] Bengua J A, Phien H N, Tuan H D and Do M N 2017 Efficient tensor completion for color image and video recovery: low-rank tensor train *IEEE Trans. Image Process.* **26** 2466–79

[37] Zhao Q, Sugiyama M, Yuan L and Cichocki A 2019 Learning efficient tensor representations with ring-structured networks *ICASSP 2019-2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE) pp 8608–12

[38] He W, Chen Y, Yokoya N, Li C and Zhao Q 2020 Hyperspectral super-resolution via coupled tensor ring factorization (arXiv:2001.01547)

[39] Dian R, Li S and Fang L 2019 Learning a low tensor-train rank representation for hyperspectral image super-resolution *IEEE Trans. Neural Netw. Learn. Syst.* **30** 2672–83

[40] Bellman R E 2015 *Adaptive Control Processes: A Guided Tour* vol 2045 (Princeton, NJ: Princeton University Press)

[41] Ahmadi-Asl S, Cichocki A, Phan A H, Asante-Mensah M G, Mousavi F, Oseledets I and Tanaka T 2020 Randomized algorithms for fast computation of low-rank tensor ring model *Mach. Learn.: Sci. Technol.* **2** 011001

[42] Yokota T, Hontani H, Zhao Q and Cichocki A 2020 Manifold modeling in embedded space: an interpretable alternative to deep image prior *IEEE Trans. Neural Networks Learning Systems* (https://doi.org/10.1109/TNNLS.2020.3037923)

[43] Yang J, Zhu Y, Li K, Yang J and Hou C 2018 Tensor completion from structurally-missing entries by low-TT-rankness and fiber-wise sparsity *IEEE J. Sel. Top. Signal Process.* **12** 1420–34

[44] Zhang J, Zhao D and Gao W 2014 Group-based sparse representation for image restoration *IEEE Trans. Image Process.* **23** 3336–51

[45] Elad M and Aharon M 2006 Image denoising via sparse and redundant representations over learned dictionaries *IEEE Trans. Image Process.* **15** 3736–45

[46] Aharon M, Elad M and Bruckstein A 2006 K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation *IEEE Trans. Signal Process.* **54** 4311–22

[47] Mairal J, Bach F, Ponce J and Sapiro G 2009 Online dictionary learning for sparse coding *Proc. 26th Annual Int. Conf. on Machine Learning* pp 689–96

[48] Cai S, Kang Z, Yang M, Xiong X, Peng C and Xiao M 2018 Image denoising via improved dictionary learning with global structure and local similarity preservations *Symmetry* **10** 167

[49] Yuan L, Li C, Mandic D, Cao J and Zhao Q 2019 Tensor ring decomposition with rank minimization on latent space: an efficient approach for tensor completion *Proc. Conf. on Artificial Intelligence* vol 33 pp 9151–8

[50] Huang H, Liu Y, Long Z and Zhu C 2020 Robust low-rank tensor ring completion *IEEE Trans. Comput. Imaging* **6** 1117–26

[51] Yuan L, Li C, Cao J and Zhao Q 2020 Rank minimization on tensor ring: an efficient approach for tensor decomposition and completion *Mach. Learn.* **109** 603–22

[52] Zhao X-L, Nie X, Zheng Y-B, Ji T-Y and Huang T-Z 2019 Low-rank tensor completion via tensor nuclear norm with hybrid smooth regularization *IEEE Access* **7** 131888–901

[53] Ding M, Huang T-Z, Zhao X-L and Ma T-H 2020 Tensor completion via nonconvex tensor ring rank minimization with guaranteed convergence (arXiv:2005.09674)

[54] Toh K-C and Yun S 2010 An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems *Pac. J. Optim.* **6** 15

[55] Yuan L, Zhao Q, Gui L and Cao J 2019 High-order tensor completion via gradient-based optimization under tensor train format *Signal Process.: Image Commun.* **73** 53–61

[56] Dunlavy D M, Kolda T G and Acar E 2010 Poblano v1. 0: a matlab toolbox for gradient-based optimization *Technical Report* SAND2010-1422 (Sandia National Laboratories)

[57] Yuan L, Li C, Mandic D, Cao J and Zhao Q 2018 Tensor ring decomposition with rank minimization on latent space: an efficient approach for tensor completion (arXiv:1809.02288)

[58] Acar E, Dunlavy D M, Kolda T G and Mørup M 2011 Scalable tensor factorizations for incomplete data *Chemometr. Intell. Lab. Syst.* **106** 41–56

[59] Yokota T, Zhao Q and Cichocki A 2016 Smooth PARAFAC decomposition for tensor completion *IEEE Trans. Signal Process.* **64** 5423–36

[60] Discovery 2015 Pistol shot recorded at 73 000 frames per second (https://youtu.be/7y9apnbI6GA)