# Fast Algorithm for Low-rank Tensor Completion in Delay-embedded Space

Ryuki Yamamoto*, Hidekata Hontani*, Akira Imakura†, and Tatsuya Yokota*,⋆
* Nagoya Institute of Technology, Aichi, Japan,
r.yamamoto.496@stn.nitech.ac.jp, {hontani, t.yokota}@nitech.ac.jp
⋆ RIKEN Center for Advanced Intelligence Project, Tokyo, Japan
† University of Tsukuba, Ibaraki, Japan, imakura@cs.tsukuba.ac.jp

## Abstract

*Tensor completion using multiway delay-embedding transform (MDT) (or Hankelization) suffers from the large memory requirement and high computational cost in spite of its high potentiality for the image modeling. Recent studies have shown high completion performance with a relatively small window size, but experiments with large window sizes require huge amount of memory and cannot be easily calculated. In this study, we address this serious computational issue, and propose its fast and efficient algorithm. Key techniques of the proposed method are based on two properties: (1) the signal after MDT can be diagonalized by Fourier transform, (2) an inverse MDT can be represented as a convolutional form. To use the properties, we modify MDT-Tucker [26], a method using Tucker decomposition with MDT, and introducing the fast and efficient algorithm. Our experiments show more than 100 times acceleration while maintaining high accuracy, and to realize the computation with large window size.*

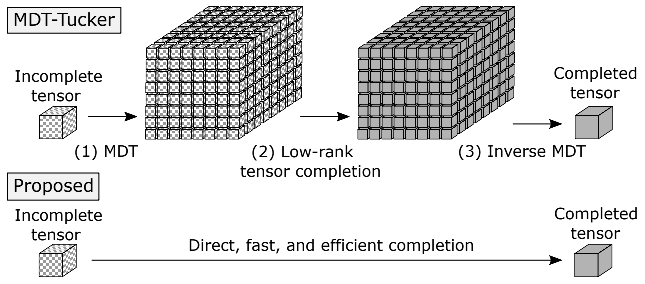

Figure 1. Conceptual illustrations of MDT-Tucker [26] and the proposed method. MDT-Tucker consists of three steps of MDT, low-rank tensor completion, and inverse MDT. By contrast, the proposed method directly obtains the completed tensor.

Table 1. Memory requirements (double precision) for a Hankel structured tensor obtained by MDT with delay window size $\tau$.

| Input tensor size | $\tau = 1$ | $\tau = 16$ | $\tau = 32$ |
|---|---|---|---|
| 256 | 2.04KB | 30.8KB | 57.6MB |
| $256 \times 256$ | 524KB | 118MB | 414MB |
| $256 \times 256 \times 256$ | 134MB | 458GB | 2.99TB |

## 1. Introduction

Tensor/matrix completion using a Multiway Delay-embedding Transform (MDT) (or multiway Hankelization) has become a very important framework in recent years [12, 16–19, 21, 22, 25, 26, 28]. The MDT constructs a Hankel (structured) tensor from an original tensor, and it is known that the Hankel tensor has low-rank structure in many cases such as images and videos [3, 9]. Then, various low-rank tensor completion methods [1, 6, 8, 10, 11, 20, 30] can be directly applied into the Hankel tensor, and return the completed tensor by inverse MDT of it (see Fig. 1).

Although tensor processing using MDT has been actively studied in recent years, there is a bottleneck of high memory requirements and expensive computations. For example, Tab. 1 shows the memory requirements for MDT, where $\tau$ is a delay window size. In a rough calculation, the number of entries will be $\tau^N$ times that of the original in case of an $N$-th order tensor. In a case of MDT-Tucker [26] (Tucker decomposition (TKD) with MDT), many iterations of singular value decomposition (SVD) in alternating least squares (ALS) algorithm are applied to such a huge tensor, and it is highly expensive for large $\tau$.

In this study, we address this issue, and propose a fast alternative method for MDT-Tucker. First, we focus on the redundant circulant structure of Hankel tensor. There are prior studies on fast processing of Hankel matrices/tensors [4, 13, 14, 24], and these results are very useful. The key ideas in these studies are the Fourier diagonalization of a circulant matrix and the fast convolutional operation in Fourier space, and we also use these results in our study.

This work was supported in part by Japan Science and Technology Agency (JST) ACT-I under Grant JPMJPR18UU, and in part by JSPS KAKENHI under Grant 20H04208.

However, these algorithms [4, 13, 14, 24] cannot be directly applied to MDT-Tucker [26]. To apply these acceleration techniques to tensor completion problem, we propose to modify MDT-Tucker as follow:

- MDT is replaced with circulant MDT.
- Three steps in MDT-Tucker are combined to one.
- Half of factor matrices are reduced from TKD.

Then, we formulate a new optimization problem and derive a solution algorithm. Finally, we show that the proposed algorithm can be efficiently computed by using Fast Fourier Transform (FFT) [5].

### 1.1. Mathematical notations

We follow basic mathematical notations in [26], except with $\odot$ for Hadamard product. In addition, let us consider $N$ matrices $\boldsymbol{U}_n \in \mathbb{R}^{I_n \times R_n}$ and an $N$-th order tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$, then *all-mode product* is defined as

$$\boldsymbol{\mathcal{G}} \times \{\boldsymbol{U}\} := \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}_1 \cdots \times_N \boldsymbol{U}_N. \tag{1}$$

All-mode product excluding the $n$-th mode is defined as

$$\begin{aligned}\boldsymbol{\mathcal{G}} \times_{-n} \{\boldsymbol{U}\} := \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}_1 \cdots \times_{n-1} \boldsymbol{U}_{n-1} \\ \times_{n+1} \boldsymbol{U}_{n+1} \cdots \times_N \boldsymbol{U}_N.\end{aligned} \tag{2}$$

In contrast that above product is for all-mode, next we consider *all-odd-mode*. Let us consider $N$ matrices $\boldsymbol{U}_n \in \mathbb{R}^{I_n \times R_n}$ and a $2N$-th order tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times J_1 \times \cdots \times R_N \times J_N}$, then *all-odd-mode product* is

$$\boldsymbol{\mathcal{G}} \times^{\text{odd}} \{\boldsymbol{U}\} := \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}_1 \times_3 \boldsymbol{U}_2 \cdots \times_{2N-1} \boldsymbol{U}_N. \tag{3}$$

Moreover, all-odd-mode product excluding the $(2n-1)$-th mode (*i.e.*, the $n$-th *odd* mode) is defined as

$$\begin{aligned}\boldsymbol{\mathcal{G}} \times^{\text{odd}}_{-n} \{\boldsymbol{U}\} := \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}_1 \cdots \times_{2n-3} \boldsymbol{U}_{n-1} \\ \times_{2n+1} \boldsymbol{U}_{n+1} \cdots \times_{2N-1} \boldsymbol{U}_N.\end{aligned} \tag{4}$$

## 2. Review of MDT-Tucker

First, the MDT-Tucker [26] consists of following three steps: (1) MDT is applied to an observed incomplete tensor and a mask tensor. (2) Low-rank tensor completion based on Tucker decomposition (TKD) is applied to the incomplete Hankel tensor. Finally (3) the inverse MDT is applied to the complete Hankel tensor.

**Step 1**: Let $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$, $\boldsymbol{\mathcal{Q}} \in \{0,1\}^{T_1 \times \cdots \times T_N}$ be an incomplete input tensor and its mask tensor respectively. The first step is given by

$$\boldsymbol{\mathcal{T}}_H = \widetilde{\mathcal{H}}(\boldsymbol{\mathcal{T}}) \in \mathbb{R}^{J_1 \times \cdots \times J_{2N}}, \tag{5}$$

$$\boldsymbol{\mathcal{Q}}_H = \widetilde{\mathcal{H}}(\boldsymbol{\mathcal{Q}}) \in \{0,1\}^{J_1 \times \cdots \times J_{2N}}, \tag{6}$$

where $\widetilde{\mathcal{H}}(\cdot)$ is an operator of MDT (see Sec. 2.1 for the definition). Note that $\boldsymbol{\mathcal{T}}_H$ is a $2N$-th order Hankel tensor in contrast that the original tensor $\boldsymbol{\mathcal{T}}$ is an $N$-th order tensor.

**Step 2**: Low rank tensor completion based on TKD is performed for $\boldsymbol{\mathcal{T}}_H$. This TKD consists of $2N$ factor matrices $\{\boldsymbol{U}_n \in \mathbb{R}^{J_n \times R_n}\}_{n=1}^{2N}$ and a $2N$-th order core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_{2N}}$. With $R_n = 1$ for all $n$ as the initial values, the optimum $R_n$ are obtained by gradually increasing each $R_n$ until the error becomes smaller than the threshold value. As the results, the TKD ($\widehat{\boldsymbol{U}}_1$, ..., $\widehat{\boldsymbol{U}}_{2N}$, $\widehat{\boldsymbol{\mathcal{G}}}$) are obtained.

**Step 3**: Finally, the resultant tensor is obtained by inverse MDT as follow:

$$\widehat{\boldsymbol{\mathcal{X}}} = \widetilde{\mathcal{H}}^{\dagger}(\widehat{\boldsymbol{\mathcal{G}}} \times \{\widehat{\boldsymbol{U}}\}), \tag{7}$$

where $\widetilde{\mathcal{H}}^{\dagger}(\cdot)$ is an operator of inverse MDT.

### 2.1. MDT

#### 2.1.1 Delay embedding for a vector

First, we define a delay embedding of a 1-dimensional signal (*i.e.*, Hankelization). The operator of the MDT is denoted by $\widetilde{\mathcal{H}}(\cdot)$, and the delay embedding for a vector $\boldsymbol{x} = \left(x_1, x_2, \ldots, x_T\right)^{\top} \in \mathbb{R}^T$ with delay window size $\tau$ is defined as

$$\widetilde{\mathcal{H}}_{\tau}(\boldsymbol{x}) := \begin{pmatrix} x_1 & x_2 & \ldots & x_{T-\tau+1} \\ x_2 & x_3 & \ldots & x_{T-\tau+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\tau} & x_{\tau+1} & \ldots & x_T \end{pmatrix} \in \mathbb{R}^{\tau \times (T-\tau+1)}. \tag{8}$$

Note that anti-diagonal entries are the same, and such a matrix is called as Hankel matrix. There exists a duplication matrix $\widetilde{\boldsymbol{S}} \in \{0,1\}^{\tau(T-\tau+1) \times T}$ that satisfies $\text{vec}(\widetilde{\mathcal{H}}_{\tau}(\boldsymbol{x})) = \widetilde{\boldsymbol{S}}\boldsymbol{x}$. Then, the delay embedding can also be expressed by

$$\widetilde{\mathcal{H}}_{\tau}(\boldsymbol{x}) = \text{fold}_{(\tau, T-\tau+1)}(\widetilde{\boldsymbol{S}}\boldsymbol{x}), \tag{9}$$

where $\text{fold}_{(v,V)} : \mathbb{R}^{vV} \to \mathbb{R}^{v \times V}$ is a folding operator from a vector to a matrix (see Fig. 2).

Since the delay embedding is essentially a duplication operation, its inverse is essentially a mean operation. Let be $\widetilde{\boldsymbol{X}}_H \in \mathbb{R}^{\tau \times (T-\tau+1)}$, the inverse MDT of $\widetilde{\boldsymbol{X}}_H$ is defined as

$$\widetilde{\mathcal{H}}^{\dagger}_{\tau}(\widetilde{\boldsymbol{X}}_H) := \widetilde{\boldsymbol{S}}^{\dagger}\text{vec}(\widetilde{\boldsymbol{X}}_H) \in \mathbb{R}^T, \tag{10}$$

where $\widetilde{\boldsymbol{S}}^{\dagger} = (\widetilde{\boldsymbol{S}}^{\top}\widetilde{\boldsymbol{S}})^{-1}\widetilde{\boldsymbol{S}}^{\top}$ is a Moore-Penrose pseudo-inverse of $\widetilde{\boldsymbol{S}}$. A matrix $\widetilde{\boldsymbol{S}}^{\top}\widetilde{\boldsymbol{S}}$ is diagonal, and its diagonal entries are the number of duplications of individual entries.

#### 2.1.2 Tensor extension (Step1 and Step3)

Delay embedding can be naturally extended for an $N$-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$. Let us consider $N$ duplication matrices $\widetilde{\boldsymbol{S}}_n \in \{0,1\}^{\tau_n(T_n-\tau_n+1) \times T_n}$ with window size $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_N) \in \mathbb{R}^N$, Multiway Delay-embedding

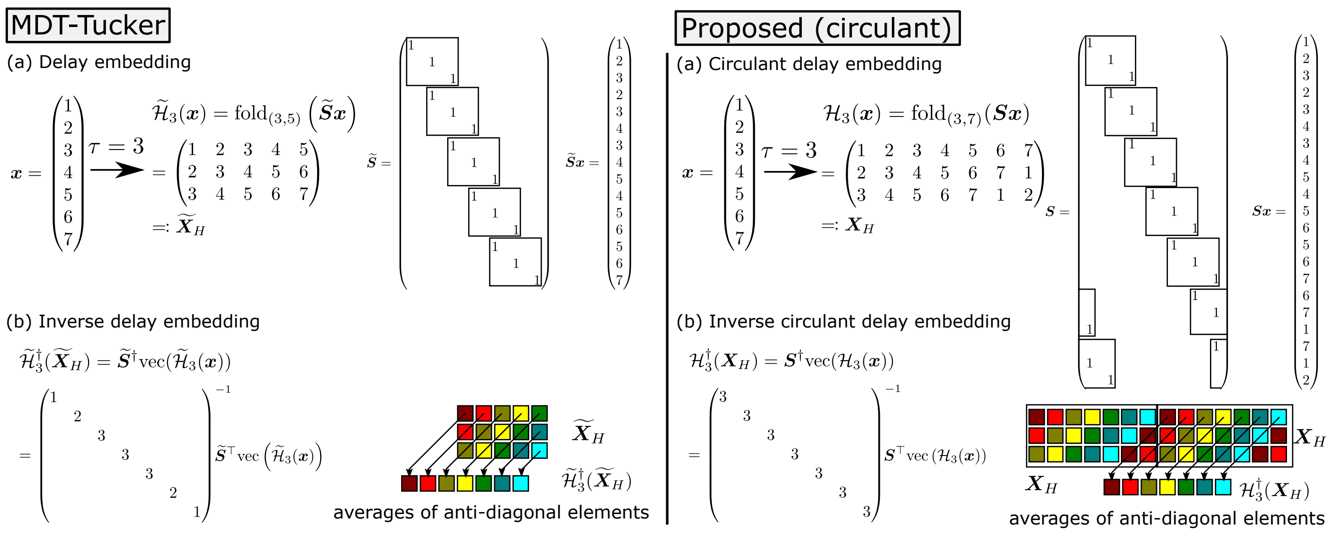
MDT-Tucker
(a) Delay embedding
(b) Inverse delay embedding
averages of anti-diagonal elements
Proposed (circulant)
(a) Circulant delay embedding
(b) Inverse circulant delay embedding
averages of anti-diagonal elements

Figure 2. A MDT used by MDT-Tucker and a circulant MDT used by the proposed method.

Transform (MDT) is defined using all-mode product and folding as

$$\widetilde{\mathcal{H}}_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{X}}) := \mathrm{fold}_{(\boldsymbol{\tau},\boldsymbol{T}-\boldsymbol{\tau}+\mathbf{1})}(\boldsymbol{\mathcal{X}} \times \{\widetilde{\boldsymbol{S}}\}), \tag{11}$$

where $\mathrm{fold}_{(\boldsymbol{v},\boldsymbol{V})} : \mathbb{R}^{v_1 V_1 \times \cdots \times v_N V_N} \to \mathbb{R}^{v_1 \times V_1 \times \cdots \times v_N \times V_N}$ is a folding operator from an $N$-th order tensor to the $2N$-th order tensor. For a $2N$-th order tensor $\widetilde{\boldsymbol{\mathcal{X}}}_H \in \mathbb{R}^{\tau_1 \times (T_1-\tau_1+1) \times \cdots \times \tau_N \times (T_N-\tau_N+1)}$, the inverse MDT is defined as

$$\widetilde{\mathcal{H}}^{\dagger}_{\boldsymbol{\tau}}(\widetilde{\boldsymbol{\mathcal{X}}}_H) := \mathrm{unfold}_{(\boldsymbol{\tau},\boldsymbol{T}-\boldsymbol{\tau}+\mathbf{1})}(\widetilde{\boldsymbol{\mathcal{X}}}_H) \times \{\widetilde{\boldsymbol{S}}^{\dagger}\} \tag{12}$$

where $\mathrm{unfold}_{(\boldsymbol{v},\boldsymbol{V})}$ is an inverse transformation of $\mathrm{fold}_{(\boldsymbol{v},\boldsymbol{V})}$.

## 2.2. Tucker-based tensor completion (Step 2)

Here, we briefly explain how to obtain Tucker-decomposition at the Step 2 in MDT-Tucker. In this step, the following optimization problem is considered,

$$\begin{aligned} \underset{\boldsymbol{\mathcal{G}},\{\boldsymbol{U}_n\}_{n=1}^{2N}}{\text{minimize}} \quad & \|\boldsymbol{\mathcal{Q}}_H \odot (\boldsymbol{\mathcal{T}}_H - \boldsymbol{\mathcal{G}} \times \{\boldsymbol{U}\})\|_F^2, \\ \text{s.t.} \quad & \boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_{2N}}, \\ & \boldsymbol{U}_n \in \mathbb{R}^{J_n \times R_n},\ \boldsymbol{U}_n^{\top}\boldsymbol{U}_n = \boldsymbol{I}_{J_n}, \\ & R_n \leq J_n \ \ (\forall n \in \{1,\dots,2N\}). \end{aligned} \tag{13}$$

It can be solved by a combination of alternating least squares (ALS) algorithm [2] and majorization-minimization (MM) algorithm [7, 15]. In addition, we solve (13) iteratively while increasing $R_n$ until the cost function becomes sufficiently small. All processes are summarized in Algorithm 1. How to increase the rank of each mode (*e.g.*, $R_n \leftarrow R_n + 1$) can be set by using a vector $\boldsymbol{L}_n$.

## 2.3. Hint for improvements

In MDT, an $N$-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{T \times \cdots \times T}$ is transformed to a Hankel tensor and it can also be represented as a multilevel Hankel matrix $\boldsymbol{H}_{\boldsymbol{\mathcal{X}}}$ by reshaping. A fast SVD for (multilevel) Hankel matrix is studied to exploit the Hankel structure [13, 24], and a fast product between of the Hankel tensor and the vector is studied in [4]. The key idea of these studies is from a fact that any multilevel anti-circulant matrix $\boldsymbol{C}_{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{T^N \times T^N}$ can be diagonalized by $N$-dimensional Fourier basis $\boldsymbol{W} \in \mathbb{C}^{T^N \times T^N}$. Then, a matrix $\boldsymbol{W}\boldsymbol{C}_{\boldsymbol{\mathcal{X}}}\boldsymbol{W}$ is diagonal, and its diagonal entries can be obtained by $N$-dimensional Fourier transform of the original tensor $\boldsymbol{\mathcal{X}}$. In addition, the multilevel Hankel matrix $\boldsymbol{H}_{\boldsymbol{\mathcal{X}}}$ is completely included as a part of the multilevel anti-circulant matrix $\boldsymbol{C}_{\boldsymbol{\mathcal{X}}}$.

The above results show that the Hankel tensor can be represented by the Fourier transform of the original tensor without explicit calculation. It also shows that the multiplication of a Hankel tensor and a vector (or matrix) can be efficiently calculated by using the Fourier transform of the original tensor without using the Hankel tensor explicitly.

## 3. Proposed method

### 3.1. Overview of fast MDT-Tucker

MDT-Tucker [26] is computationally expensive due to the explicit calculation of the Hankel tensor by MDT. We propose to apply the results shown in Sec. 2.3 to the MDT-Tucker for fast and efficient implementation. However, it cannot be directly applied as is. Then, we propose to reformulate the MDT-Tucker in this study.

The reformulation of MDT-Tucker is as follow:

- To exploit the property of a circulant matrix, we define a circulant MDT and replace it with a normal MDT.

- To avoid the explicit calculation of the Hankel tensor, we skip Step 1 in MDT-Tucker, and combine all three steps into one optimization problem.
- For the efficient update of (odd-number) factor matrices $\boldsymbol{U}_{2n-1} \in \mathbb{R}^{\tau_n \times R_n}$, we do not consider even-mode factor matrices (*i.e.*, even-mode factor matrices are assumed as identity matrices $\boldsymbol{U}_{2n} = \boldsymbol{I}_{T_n}$) in TKD.

In the following sections, we explain details of the proposed method step by step. First, we define circulant MDT in Sec. 3.2. Second, we show the reformulated optimization problem in Sec. 3.3, and derive its solution algorithm in Sec. 3.4. Note that solution algorithm in Sec. 3.4 is not fast and efficient as is, but fast and efficient by using specific implementation with FFT. Finally, we show the techniques of implementation for fast and efficient computation in Sec. 3.5.

### 3.2. Circulant MDT

First, a circulant delay embedding of $\boldsymbol{x} \in \mathbb{R}^T$ with delay window size $\tau$ is defined as

$$\mathcal{H}_\tau(\boldsymbol{x}) := \begin{pmatrix} x_1 & x_2 & \dots & x_T \\ x_2 & x_3 & \dots & x_1 \\ \vdots & \vdots & \ddots & \vdots \\ x_\tau & x_{\tau+1} & \dots & x_{\tau-1} \end{pmatrix} \in \mathbb{R}^{\tau \times T}, \quad (14)$$

where $\mathcal{H}_\tau(\cdot)$ is a circulant delay-embedding operator. Note that the first row is the same as the input vector and the $k$-th row is constructed by circulant shift of $\boldsymbol{x}$ with width $k-1$. In similar way to MDT (see Sec. 2.1), a duplication matrix $\boldsymbol{S}$ can be considered. Fig. 2 shows an example of a circulant delay embedding and its inverse with $T = 7$ and $\tau = 3$.

A circulant MDT for an $N$-th order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$ with delay window sizes $\boldsymbol{\tau} = (\tau_1, \dots, \tau_N) \in \mathbb{R}^N$ can be defined as

$$\mathcal{H}_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{X}}) := \text{fold}_{(\boldsymbol{\tau},\boldsymbol{T})}(\boldsymbol{\mathcal{X}} \times \{\boldsymbol{S}\}), \quad (15)$$

where $\boldsymbol{S}_n \in \{0,1\}^{\tau_n T_n \times T_n}$ are duplication matrices.

For a $2N$-th order tensor $\boldsymbol{\mathcal{X}}_H \in \mathbb{R}^{\tau_1 \times T_1 \times \cdots \times \tau_N \times T_N}$, the inverse circulant MDT is defined as

$$\mathcal{H}^\dagger_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{X}}_H) := \text{unfold}_{(\boldsymbol{\tau},\boldsymbol{T})}(\boldsymbol{\mathcal{X}}_H) \times \{\boldsymbol{S}^\dagger\}. \quad (16)$$

### 3.3. Reformulated optimization problem

In this section, we reformulate (13) by using the circulant MDT while avoiding the explicit use of $\boldsymbol{\mathcal{T}}_H$. Then, the proposed low rank tensor completion using a circulant MDT is given by

$$\begin{aligned} \underset{\boldsymbol{\mathcal{G}},\{\boldsymbol{F}_n\}_{n=1}^N}{\text{minimize}} \quad & \left\| \boldsymbol{\mathcal{Q}} \odot \left(\boldsymbol{\mathcal{T}} - \mathcal{H}^\dagger_{\boldsymbol{\tau}}\left(\boldsymbol{\mathcal{G}} \times^{\text{odd}} \{\boldsymbol{F}\}\right)\right) \right\|_F^2, \quad (17) \\ \text{s.t.} \quad & \boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times T_1 \times \cdots \times R_N \times T_N}, \\ & \boldsymbol{F}_n \in \mathbb{R}^{\tau_n \times R_n}, \ \boldsymbol{F}_n^\top \boldsymbol{F}_n = \boldsymbol{I}_{R_n}, \\ & R_n \le \tau_n \ (\forall n \in \{1, \dots, N\}), \end{aligned}$$

Table 2. Difference between MDT-Tucker and the proposed method ($T_n = T, \tau_n = \tau \le T/2$ ($\forall n$) are assumed)

| | MDT-Tucker | Proposed |
|---|---|---|
| Delay embedding | MDT | circulant MDT |
| Formulated steps | 3 | 1 |
| Factor matrices | all-modes | only odd-modes |
| Memory size | $\mathcal{O}(\tau^N T^N)$ | $\mathcal{O}(T^N)$ |
| Computational cost | $\mathcal{O}(\tau^{N+1} T^N)$ | $\mathcal{O}(NT^N \log T)$ |

where $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$ and $\boldsymbol{\mathcal{Q}} \in \{0,1\}^{T_1 \times \cdots \times T_N}$ are an input incomplete tensor and a mask tensor, respectively. Note that these are not $\boldsymbol{\mathcal{T}}_H$ and $\boldsymbol{\mathcal{Q}}_H$ obtained by MDT. A resultant tensor is given by $\widehat{\boldsymbol{\mathcal{X}}} = \mathcal{H}^\dagger_{\boldsymbol{\tau}}\left(\widehat{\boldsymbol{\mathcal{G}}} \times^{\text{odd}} \{\widehat{\boldsymbol{F}}\}\right)$, and it is efficiently calculated (see Sec. 3.5). Although the size of a core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times T_1 \times \cdots \times R_N \times T_N}$ is still large, it is not necessary to be explicitly calculated. In fact, $\mathcal{H}^\dagger_{\boldsymbol{\tau}}\left(\boldsymbol{\mathcal{G}} \times^{\text{odd}} \{\boldsymbol{F}\}\right) \in \mathbb{R}^{T_1 \times \cdots \times T_N}$ can be obtained from a tensor $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$ with a much smaller size in optimization (see Algorithm 2 and Sec. 3.5). Since the circulant MDT $\mathcal{H}_{\boldsymbol{\tau}}$ and its inverse $\mathcal{H}^\dagger_{\boldsymbol{\tau}}$ used in our algorithm can be implemented using FFT, we do not need to perform them explicitly.

### 3.4. Derivation of optimization algorithm

In this section, we explain the proposed algorithm for the reformulated problem (17). The basic strategy is the same as Tucker-based tensor completion in Sec. 2.2. It is summarized in Algorithm 2 and combining the following three techniques: rank-increment, MM-algorithm, and ALS-algorithm. Input and output in Algorithm 1 and Algorithm 2 are almost same. Note that we consider MDT, inverse MDT, and a core tensor $\boldsymbol{\mathcal{G}}$ in algorithm derivation, and a naive implementation of the derived algorithm is not fast and efficient. However, expensive process at the 8th, 10th, and 14th lines in Algorithm 2 can be implemented in fast and efficient way with FFT.

To solve the reformulated optimization (17), we consider the minimization of two auxiliary functions: $h$ and $g$. The cost function $f$ and its auxiliary function $h$ are

$$f(\theta) := \| \boldsymbol{\mathcal{Q}} \odot (\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{X}}_\theta) \|_F^2, \quad (18)$$

$$h(\theta|\theta') := \| \boldsymbol{\mathcal{Q}} \odot (\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{X}}_\theta) \|_F^2 + \| \overline{\boldsymbol{\mathcal{Q}}} \odot (\boldsymbol{\mathcal{X}}_{\theta'} - \boldsymbol{\mathcal{X}}_\theta) \|_F^2, \quad (19)$$

where $\theta = \{\boldsymbol{\mathcal{G}}, \boldsymbol{F}_1, \dots, \boldsymbol{F}_N\}$ is a set of parameters, $\boldsymbol{\mathcal{X}}_\theta = \mathcal{H}^\dagger_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{G}} \times^{\text{odd}} \{\boldsymbol{F}\})$ is an inverse MDT of Tucker decomposition, and $\overline{\boldsymbol{\mathcal{Q}}} := 1 - \boldsymbol{\mathcal{Q}}$. Based on the theory of MM algorithm [7, 15], an update rule of $\theta^{k+1} = \text{argmin}_\theta h(\theta|\theta^k)$ has monotonically non-increasing property for the cost function $f(\theta^{k+1}) \le f(\theta^k)$.

Next, we consider the minimization of $h$. The auxiliary

**Algorithm 1** Tucker-based tensor completion with rank increment in MDT-Tucker [26]

```
1: input: $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{T_1\times\cdots\times T_N}$, $\boldsymbol{\mathcal{Q}} \in \{0,1\}^{T_1\times\cdots\times T_N}$, $\boldsymbol{\tau} = (\tau_1,\ldots,\tau_N)$, $\{\boldsymbol{L}_1,\ldots,\boldsymbol{L}_{2N}\}$, tol, $\epsilon$
2: initialize: $k_n \leftarrow 1$, $R_n \leftarrow \boldsymbol{L}_n(k_n)$ $(\forall n)$, and $\boldsymbol{\mathcal{X}}_H \in \mathbb{R}^{J_1\times\cdots\times J_{2N}}$, $\{\boldsymbol{U}_n \in \mathbb{R}^{J_n\times R_n}\}_{n=1}^{2N}$, randomly
3: $\boldsymbol{\mathcal{T}}_H \leftarrow \widetilde{\mathcal{H}}_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{T}})$, and $\boldsymbol{\mathcal{Q}}_H \leftarrow \widetilde{\mathcal{H}}_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{Q}})$ // (Step 1)
4: $f_1 \leftarrow \|\boldsymbol{\mathcal{Q}}_H \odot (\boldsymbol{\mathcal{T}}_H - \boldsymbol{\mathcal{X}}_H)\|_F^2$
5: repeat // (Step 2)
6:     $\boldsymbol{\mathcal{Z}}_H \leftarrow \boldsymbol{\mathcal{Q}}_H \odot \boldsymbol{\mathcal{T}}_H + \overline{\boldsymbol{\mathcal{Q}}}_H \odot \boldsymbol{\mathcal{X}}_H$
7:     for $n = 1,\ldots,2N$ do
8:         $\boldsymbol{U}_n \leftarrow R_n$ leading singular vectors of $[\boldsymbol{\mathcal{Z}}_H \times_{-n} \{\boldsymbol{U}^\top\}]_{(n)}$
9:     end for
10:    $\boldsymbol{\mathcal{X}}_H \leftarrow \boldsymbol{\mathcal{Z}}_H \times \{\boldsymbol{U}\boldsymbol{U}^\top\}$
11:    $f_2 \leftarrow \|\boldsymbol{\mathcal{Q}}_H \odot (\boldsymbol{\mathcal{T}}_H - \boldsymbol{\mathcal{X}}_H)\|_F^2$
12:    if $|f_2 - f_1| \leq$ tol then
13:        $\boldsymbol{\mathcal{X}}'_H \leftarrow \boldsymbol{\mathcal{Q}}_H \odot (\boldsymbol{\mathcal{T}}_H - \boldsymbol{\mathcal{X}}_H)$
14:        $n' \leftarrow \mathrm{argmax}_n \|\boldsymbol{\mathcal{X}}'_H \times_{-n} \{\boldsymbol{U}^\top\}\|_F^2$
15:        $k_{n'} \leftarrow k_{n'} + 1$, and $R_{n'} \leftarrow \boldsymbol{L}_n(k_{n'})$
16:    else
17:        $f_1 \leftarrow f_2$
18:    end if
19: until $f_2 \leq \epsilon$
20: $\widehat{\boldsymbol{\mathcal{X}}} \leftarrow \widetilde{\mathcal{H}}_{\boldsymbol{\tau}}^\dagger(\boldsymbol{\mathcal{X}}_H)$ // (Step 3)
21: output: $\widehat{\boldsymbol{\mathcal{X}}}, \boldsymbol{U}_1,\ldots,\boldsymbol{U}_{2N}$
```

function $h$ can be transformed as follows:

$$\begin{aligned} h(\theta|\theta^k) &= \|\boldsymbol{\mathcal{Q}} \odot (\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{X}}_\theta)\|_F^2 + \|\overline{\boldsymbol{\mathcal{Q}}} \odot (\boldsymbol{\mathcal{X}}_{\theta^k} - \boldsymbol{\mathcal{X}}_\theta)\|_F^2 \\ &= \|\boldsymbol{\mathcal{Z}}_{\theta^k} - \mathcal{H}_{\boldsymbol{\tau}}^\dagger(\boldsymbol{\mathcal{Y}}_\theta)\|_F^2, \end{aligned} \tag{20}$$

where $\boldsymbol{\mathcal{Z}}_{\theta^k} = \boldsymbol{\mathcal{Q}} \odot \boldsymbol{\mathcal{T}} + \overline{\boldsymbol{\mathcal{Q}}} \odot \boldsymbol{\mathcal{X}}_{\theta^k}$ and $\boldsymbol{\mathcal{Y}}_\theta = \boldsymbol{\mathcal{G}} \times^{\mathrm{odd}} \{\boldsymbol{F}\}$. It is difficult to minimize $h$ directly using the ALS.

Moreover, we consider another auxiliary function $g$ to minimize $h$, which is defined as

$$g(\theta|\theta^k) := \|\mathcal{H}_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{Z}}_{\theta^k}) - \boldsymbol{\mathcal{Y}}_\theta\|_F^2. \tag{21}$$

When we regard $\mathcal{H}_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{Z}}_{\theta^k})$ as an input tensor, the auxiliary function $g$ is the same form of Tucker decomposition which can be directly minimized by ALS [2]. The necessary optimality conditions of $h$ and $g$ with respect to $\boldsymbol{\mathcal{Y}}_\theta$ are

$$\mathcal{H}_{\boldsymbol{\tau}}(\mathcal{H}_{\boldsymbol{\tau}}^\dagger(\boldsymbol{\mathcal{Y}}_\theta)) = \mathcal{H}_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{Z}}_{\theta^k}), \tag{22}$$

$$\boldsymbol{\mathcal{Y}}_\theta = \mathcal{H}_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{Z}}_{\theta^k}). \tag{23}$$

If the condition of $g$ (23) satisfies, then the condition of $h$ (22) satisfies. Note that $\mathcal{H}_{\boldsymbol{\tau}}$ is essentially duplication, and $\mathcal{H}_{\boldsymbol{\tau}}^\dagger$ is essentially mean. The function $g$ enforce the Tucker decomposition $\boldsymbol{\mathcal{Y}}_\theta$ to be Hankel structured, but the function $h$ does not. Therefore, it is expected that the use of $g$ improves the uniqueness of the solution than $h$.

**Algorithm 2** Proposed Tucker-based tensor completion with rank increment

```
1: input: $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{T_1\times\cdots\times T_N}$, $\boldsymbol{\mathcal{Q}} \in \{0,1\}^{T_1\times\cdots\times T_N}$, $\boldsymbol{\tau} = (\tau_1,\ldots,\tau_N)$, $\{\boldsymbol{L}_1,\ldots,\boldsymbol{L}_N\}$, tol, $\epsilon$
2: initialize: $k_n \leftarrow 1$, $R_n \leftarrow \boldsymbol{L}_n(k_n)$ $(\forall n)$, and $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{T_1\times\cdots\times T_N}$, $\{\boldsymbol{F}_n \in \mathbb{R}^{\tau_n\times R_n}\}_{n=1}^{N}$ randomly
3:
4: $f_1 \leftarrow \|\boldsymbol{\mathcal{Q}} \odot (\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{X}})\|_F^2$
5: repeat
6:     $\boldsymbol{\mathcal{Z}} \leftarrow \boldsymbol{\mathcal{Q}} \odot \boldsymbol{\mathcal{T}} + \overline{\boldsymbol{\mathcal{Q}}} \odot \boldsymbol{\mathcal{X}}$
7:     for $n = 1,\ldots,N$ do
8:         $\boldsymbol{F}_n \leftarrow R_n$ leading singular vectors of $[\mathcal{H}_\tau(\boldsymbol{\mathcal{Z}}) \times_{-n}^{\mathrm{odd}} \{\boldsymbol{F}^\top\}]_{(2n-1)}$ // (Sec. 3.5.1)
9:     end for
10:    $\boldsymbol{\mathcal{X}} \leftarrow \mathcal{H}_\tau^\dagger(\mathcal{H}_\tau(\boldsymbol{\mathcal{Z}}) \times^{\mathrm{odd}} \{\boldsymbol{F}\boldsymbol{F}^\top\})$ // (Sec. 3.5.2)
11:    $f_2 \leftarrow \|\boldsymbol{\mathcal{Q}} \odot (\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{X}})\|_F^2$
12:    if $|f_2 - f_1| \leq$ tol then
13:        $\boldsymbol{\mathcal{X}}' \leftarrow \boldsymbol{\mathcal{Q}} \odot (\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{X}})$
14:        $n' \leftarrow$ Eq. (24) // (Sec. 3.5.3)
15:        $k_{n'} \leftarrow k_{n'} + 1$, and $R_{n'} \leftarrow \boldsymbol{L}_n(k_{n'})$
16:    else
17:        $f_1 \leftarrow f_2$
18:    end if
19: until $f_2 \leq \epsilon$
20:
21: output: $\boldsymbol{\mathcal{X}}, \boldsymbol{F}_1,\ldots,\boldsymbol{F}_N$
```

Finally, we explain how to increment rank of each mode. We also employ the strategy "rank increment" used in [26] to optimize appropriate rank. First, we select the mode $n'$ whose mode residual is the maximum of all modes corresponding to the odd number. The $n$-th mode residual is defined as a residual on the multi-linear subspace spanned by all the factor matrices excluding the $n$-th mode factor matrix. Thus, the selected mode is given by:

$$n' = \underset{n}{\mathrm{argmax}} \|\mathcal{H}_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{X}}') \times_{-n}^{\mathrm{odd}} \{\boldsymbol{F}^\top\}\|_F^2, \tag{24}$$

where $\boldsymbol{\mathcal{X}}' = \boldsymbol{\mathcal{Q}} \odot (\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{X}}_\theta)$. It is interpreted as meaning that the selected $n'$-th mode expects to reduce the cost function $f$ highly when $R_{n'}$ increases while the other-mode ranks remain fix. How to increment the rank of each mode can be freely designed by $\boldsymbol{L}_n$. For example, the setting like $\boldsymbol{L}_n = (1, 2, 4, 8, \ldots, \tau_n)$ would be efficient than one by one increment $\boldsymbol{L}_n = (1, 2, 3, 4, \ldots, \tau_n)$.

## 3.5. Fast and efficient computations

The naive implementation of this algorithm is still slow and inefficient because it calculates the circulant MDT explicitly. In Algorithm 2, the parts where the (inverse) circulant MDT is calculated are lines 8, 10, and 14, and these can be efficiently computed by using FFT. As a result, the

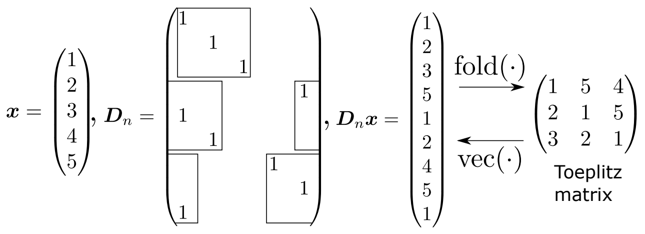

Figure 3. An example of $\boldsymbol{D}_n$ for making a Toeplitz matrix.

proposed process does not require the circulant MDT, explicitly, but equivalent results can be obtained in theory. In the remaining section, we explain the proposed implementation at each bottleneck in Algorithm 2. The MATLAB code is available via our GitHub repository[1].

#### 3.5.1 The 8-th line: Updating factor matrices

First, we propose processes for the 8-th line in Algorithm 2. The naive implementation at the 8-th line comprises as following processes:

1. $\boldsymbol{H}_n \leftarrow \left[\mathcal{H}_{\boldsymbol{\tau}}(\boldsymbol{\mathcal{Z}}) \times_{-n}^{\text{odd}} \{\boldsymbol{F}^\top\}\right]_{(2n-1)}$;
2. $\boldsymbol{A}_n \leftarrow \boldsymbol{H}_n \boldsymbol{H}_n^\top \in \mathbb{R}^{\tau_n \times \tau_n}$;
3. $\boldsymbol{F}_n \leftarrow R_n$ leading left singular vectors of $\boldsymbol{A}_n$;

The size of $\boldsymbol{H}_n$ is $\tau_n \prod_{i \neq n} R_i$ times the size of the original input signal. On the other hand, the size of the resultant autocorrelation matrix $\boldsymbol{A}_n$ is $\tau_n \times \tau_n$, which is very smaller than the $\boldsymbol{H}_n$. Exploiting that the signal after a circulant MDT can be diagonalized, $\boldsymbol{A}_n$ can be computed without calculating $\boldsymbol{H}_n$ explicitly. The fast implementation for $\boldsymbol{A}_n$ is the following processes:

**Proc. 1** : $\boldsymbol{\mathcal{Z}}_F \leftarrow \text{FFT}_N\left(\text{IFFT}_N(\boldsymbol{\mathcal{Z}}) \odot \overline{\text{IFFT}_N(\boldsymbol{\mathcal{Z}})}\right)$;

**Proc. 2** : $\boldsymbol{\mathcal{Z}}_P \leftarrow \boldsymbol{\mathcal{Z}}_F \times \{\boldsymbol{P}\} \in \mathbb{R}^{(2\tau_1-1)\times\cdots\times(2\tau_N-1)}$;

**Proc. 3** : $\boldsymbol{f}_k \leftarrow \text{vec}(\boldsymbol{F}_k \boldsymbol{F}_k^\top) \in \mathbb{R}^{\tau_k^2}$ for all $k \neq n$;

**Proc. 4** : $\boldsymbol{a}_n \leftarrow \boldsymbol{\mathcal{Z}}_P \times_{-n} \{\boldsymbol{f}^\top \boldsymbol{D}\} \in \mathbb{R}^{2\tau_n - 1}$;

**Proc. 5** : $\boldsymbol{A}_n \leftarrow \text{unfold}(\boldsymbol{D}_n \boldsymbol{a}_n) \in \mathbb{R}^{\tau_n \times \tau_n}$;

where $\text{FFT}_N(\cdot)$ and $\text{IFFT}_N(\cdot)$ are operators of $N$-dimensional FFT and $N$-dimensional inverse FFT, $\boldsymbol{P}_n \in \{0,1\}^{(2\tau_n-1)\times T_n}$ is a cropping matrix, and $\boldsymbol{D}_n \in \{0,1\}^{\tau_n^2 \times (2\tau_n - 1)}$ is a duplication matrix for Toeplitz structure. A matrix $\boldsymbol{P}_n$ can be constructed as:

$$\boldsymbol{P}_n = \begin{pmatrix} \boldsymbol{I}_{\tau_n} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{\tau_n - 1} \end{pmatrix} \in \mathbb{R}^{(2\tau_n-1)\times T_n}. \tag{25}$$

Fig. 3 shows an example of a duplication matrix $\boldsymbol{D}_n$.

[1] https://github.com/yama30120/fast-MDT-Tucker

#### 3.5.2 The 10-th line: Inverse circulant MDT

We show the 10-th line in Algorithm 2. An inverse circulant MDT can be represented as a convolution form and has a linearity. Therefore, an inverse circulant MDT can be computed in Fourier space.

For example, let us consider two vectors $\boldsymbol{a} \in \mathbb{R}^\tau$ and $\boldsymbol{b} \in \mathbb{R}^T$ ($\tau \leq T$), the elements of $\mathcal{H}_\tau^\dagger(\boldsymbol{a}\boldsymbol{b}^\top) \in \mathbb{R}^T$ is

$$\left(\mathcal{H}_\tau^\dagger(\boldsymbol{a}\boldsymbol{b}^\top)\right)(t) = \frac{1}{\tau}\sum_{m=1}^{\tau} \boldsymbol{a}(m)\boldsymbol{b}(t-m+1), \tag{26}$$

and it is convolution. Note that the inverse circulant delay embedding is the same as averages of the anti-diagonal elements (see Fig. 2). We next consider a linearity of an inverse circulant delay embedding. For two matrices $\boldsymbol{A} = (\boldsymbol{a}_1, ..., \boldsymbol{a}_R) \in \mathbb{R}^{\tau\times R}$ and $\boldsymbol{B} = (\boldsymbol{b}_1, ..., \boldsymbol{b}_R) \in \mathbb{R}^{T\times R}$, the elements of $\mathcal{H}_\tau^\dagger(\boldsymbol{A}\boldsymbol{B}^\top) \in \mathbb{R}^T$ is

$$\left(\mathcal{H}_\tau^\dagger(\boldsymbol{A}\boldsymbol{B}^\top)\right)(t) = \sum_{r=1}^{R} \mathcal{H}_\tau^\dagger(\boldsymbol{a}_r \boldsymbol{b}_r^\top)(t). \tag{27}$$

It can be extended to the inverse circulant MDT because the Hankel tensor after a circulant MDT is represented as a multilevel Hankel matrix by unfolding. Exploiting a convolution form, the computation of an inverse circulant MDT can be efficiently computed by using FFT. The processes can be summarized as follows:

**Proc. 1** : $\boldsymbol{F}_n' \leftarrow (\boldsymbol{F}_n^\top, \boldsymbol{0})^\top \in \mathbb{R}^{T_n \times R_n}$;

**Proc. 2** : $\boldsymbol{f}_{\text{power}}^{(n)} \leftarrow \left(\text{FFT}_1(\boldsymbol{F}_n') \odot \overline{\text{FFT}_1(\boldsymbol{F}_n')}\right)\boldsymbol{1}$ $(\forall n)$;

**Proc. 3** : $\boldsymbol{\mathcal{F}} \leftarrow 1 \times \{\boldsymbol{f}_{\text{power}}\} \in \mathbb{R}^{T_1\times\cdots\times T_N}$;

**Proc. 4** : $\boldsymbol{\mathcal{X}} \leftarrow \frac{1}{\prod_{n=1}^N \tau_n}\text{FFT}_N(\boldsymbol{\mathcal{F}} \odot \text{IFFT}_N(\boldsymbol{\mathcal{Z}}))$;

#### 3.5.3 The 14-th line: Selecting the mode whose rank to be increased

Finally, we show fast and efficient implementation of the 14th line in Algorithm 2. The bottleneck can be accelerated by using previous technique explained in Sec. 3.5.1. Using $\boldsymbol{H}_n = \left[\mathcal{H}_\tau(\boldsymbol{\mathcal{X}}') \times_{-n}^{\text{odd}} \{\boldsymbol{F}^\top\}\right]_{-(2n-1)}$, the $n$-th mode residual is transformed as

$$\|\mathcal{H}_\tau(\boldsymbol{\mathcal{X}}') \times_{-n}^{\text{odd}} \{\boldsymbol{F}^\top\}\|_F^2 = \text{tr}(\boldsymbol{H}_n \boldsymbol{H}_n^\top), \tag{28}$$

where $\text{tr}(\cdot)$ is an operator calculating a trace of a matrix. In other words, the $n$-th mode residual can be computed by the autocorrelation matrix of $\boldsymbol{H}_n$ and its size is $\tau_n \times \tau_n$. Therefore, fast and efficient computation of $\boldsymbol{H}_n \boldsymbol{H}_n^\top$ can be done by inputting $\boldsymbol{\mathcal{X}}'$ instead of $\boldsymbol{\mathcal{Z}}$ in Sec. 3.5.1.

## 4. Experiments

The experiment in Sec. 4.2.1 was conducted in the following environments: CPU: Intel(R) Core(TM) i7-6900K CPU @ 3.20GHz, 8 cores /16 threads, Memory: 128GByte,

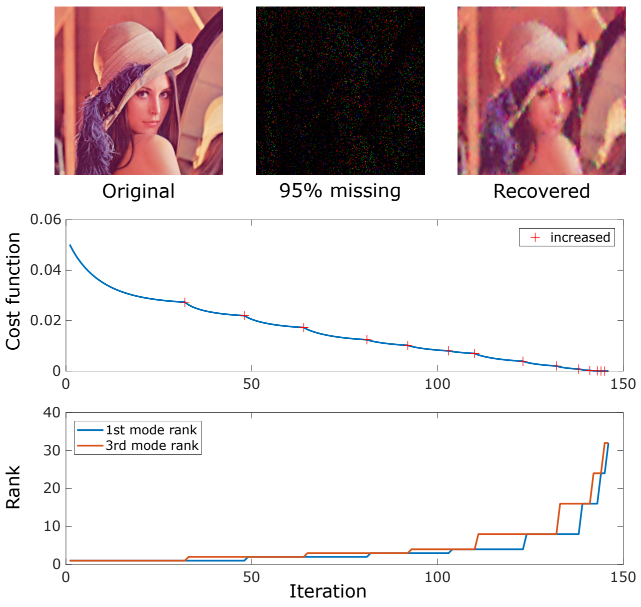

Figure 4. Cost function and ranks when recovering the 95% missing Lena image using the proposed method.

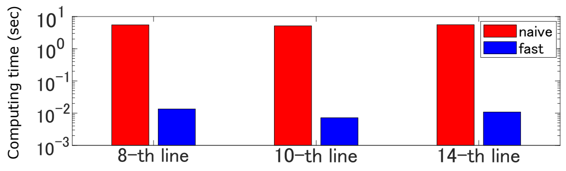

Figure 5. Comparison computing time (sec) of the naive implementation and fast one of the Algorithm 2. The red bars or left ones represent the naive implementation and the blue bars or right ones represent the accelerated implementation.

Software: Matlab R2019a. The other experiments were conducted in the following environments: 2CPUs: Intel(R) Xeon(R) Gold 6238 CPU @ 2.10GHz, 22 cores/44 threads, Memory: 755GByte, Software: Matlab R2017b.

### 4.1. Verificatation of the proposed method

#### 4.1.1 Decreasing a cost function monotonically

We verified that the cost function $f$ decreases monotonically in various data (*e.g.*, images, videos, and MRI). In this section, we show the experiment that the 95% random voxel missing of Lena image $(256 \times 256 \times 3)$ is recovered by the proposed Algorithm 2. We set the parameters: $\boldsymbol{\tau} = (32, 32, 1)$, $\boldsymbol{L}_1 = \boldsymbol{L}_2 = (1, 2, 3, 4, 8, 16, 24, 32)$, and $\boldsymbol{L}_3 = 1$. Fig. 4 shows the recovered image and the behaviors of the cost function $f$ and the ranks $(R_1, R_2)$. Although our algorithm derivated in Sec. 3.4 is not guaranteed to monotonically decrease the cost function exactly, our experiment shows decreasing monotonically the cost function.

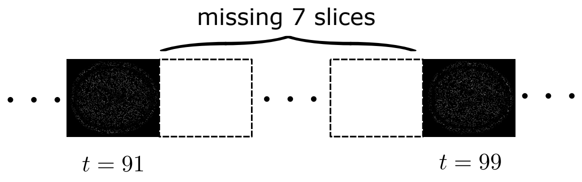

Figure 6. Missing MRI: 7 consecutive slices ($t = 92, 93, \dots, 98$) missing in MRI images and the other slices is 90% random voxel missing. Missing voxels are represented by black.

#### 4.1.2 Comparison of computing time

We compared the computing time of the naive algorithm and the fast one of the proposed method at 8-th, 10-th and 14-th lines in Algorithm 2 where the circulant MDT is calculated explicitly. We set the parameters: $\boldsymbol{T} = (256, 256, 3)$, $\boldsymbol{\tau} = (32, 32, 1)$, and $\boldsymbol{R} = \boldsymbol{\tau}$. Fig. 5 shows the results of the computing time of the individual lines. Note that the vertical axis of this graph is the logarithmic axis. The accelerated implementation is more than 100 times faster than the naive one in all the lines.

### 4.2. Comparison of recovery performance

#### 4.2.1 Videos recovery using various methods

We compared the performance of the proposed method and the other tensor completion algorithms[2]: nuclear-norm and TV regularization (LR&TV) [27], SPCQV (constrained PARAFAC tensor decomposition) [29], MDT-Tucker [26]. We prepared three video data[3] whose size are $(112 \times 160 \times 3 \times 32)$, $(90 \times 160 \times 3 \times 64)$, and $(90 \times 160 \times 3 \times 100)$ respectively. We created missing videos in each of three ways: slice missing (several frames, horizontal and vertical slices) and random voxel missing (70% and 95%). We applied the proposed method with $\boldsymbol{\tau} = (8, 8, 1, 8)$, $\boldsymbol{L}_1 = \boldsymbol{L}_2 = \boldsymbol{L}_4 = (1, 2, 3, 4, 5, 6, 7, 8)$, and $\boldsymbol{L}_3 = 1$.

Tab. 3 shows the peak signal-to-noise ratio (PSNR), the frame average of structural similarity (SSIM) [23], and computing times for these comparisons, where the best PSNR, SSIM, and computing time values are emphasized in bold font. SPCQV and MDT-Tucker are almost the best PSNR and SSIM, and the proposed method is the next or third best. However, computing time of the proposed one is hundreds of times faster than the others.

#### 4.2.2 MRI images recovery using MDT-Tucker and the proposed method

We recovered the MRI images using MDT-Tucker and the proposed method with various delay window sizes. We prepared the MRI images whose size is $(217 \times 181 \times 181)$ and

[2]The source codes of LR&TV, SPCQV, and MDT-Tucker are obtained from https://sites.google.com/site/yokotatsuya/home/software.

[3]smoke1 and smoke2 are obtained from NHK CREATIVE LIBRARY https://www.nhk.or.jp/archives/creative/, and these were down-sampled for the experiments.

Table 3. Comparison of the peak signal-to-noise ratio (PSNR), the frame average of structural similarity (SSIM) and the computing time (sec) of recovery videos. The entries represent (PSNR, SSIM, computing time).

| | LR&TV | SPCQV | MDT-Tucker | Proposed |
|---|---|---|---|---|
| ocean, slice missing | (30.1, 0.931, 529) | (25.9, 0.929, 457) | (**30.8**, **0.953**, 3160) | (30.1, 0.946, **4.27**) |
| ocean, 70% voxel missing | (26.2, 0.820, 528) | (**33.3**, **0.961**, 684) | (30.4, 0.930, 3789) | (26.0, 0.846, **6.52**) |
| ocean, 95% voxel missing | (20.6, 0.469, 973) | (**24.1**, 0.702, 588) | (24.0, **0.721**, 4346) | (22.0, 0.605, **9.48**) |
| smoke1, slice missing | (40.6, 0.990, 1040) | (37.6, 0.977, 197) | (**41.7**, **0.992**, 964) | (40.1, 0.990, **5.39**) |
| smoke1, 70% voxel missing | (35.1, 0.967, 1193) | (34.7, 0.941, 380) | (**38.6**, **0.975**, 1038) | (34.8, 0.962, **7.71**) |
| smoke1, 95% voxel missing | (27.7, 0.877, 1810) | (**31.7**, **0.914**, 515) | (28.3, 0.875, 1197) | (28.7, 0.885, **12.87**) |
| smoke2, slice missing | (32.9, 0.979, 1389) | (33.2, 0.977, 399) | (**40.6**, 0.992, 1851) | (39.2, **0.993**, **11.19**) |
| smoke2, 70% voxel missing | (26.2, 0.925, 1504) | (35.1, 0.964, 801) | (**41.3**, **0.992**, 2007) | (31.6, 0.966, **17.21**) |
| smoke2, 95% voxel missing | (19.6, 0.741, 2088) | (**31.6**, **0.935**, 1055) | (26.3, 0.877, 2255) | (23.7, 0.831, **33.40**) |

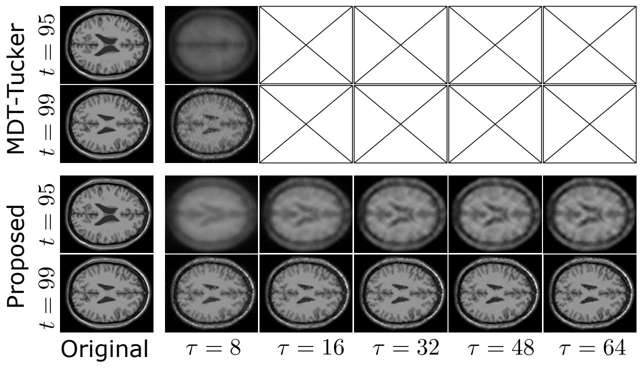

Figure 7. Results recovered the missing MRI using MDT-tucker and the proposed method with various $\tau$ values. The X in the box is indicated that the missing MRI cannot be recovered due to memory limitations.

Table 4. Comparison of PSNR, the average of SSIM of all the slices and computing time (sec) of recovery the MRI images. The entries represent (PSNR, SSIM, computing time). "N.A." means not applicable due to memory limitations.

| $\tau$ | MDT-Tucker | Proposed |
|---|---|---|
| 8 | (26.39, 0.859, 10542) | (27.87, 0.896, 49.4) |
| 16 | N.A. | (28.62, 0.909, 77.2) |
| 32 | N.A. | (28.97, 0.914, 120.6) |
| 48 | N.A. | (29.04, 0.913, 157.4) |
| 64 | N.A. | (29.00, 0.911, 219.5) |

created the missing MRI in a way of 7 consecutive slices $(t = 92, 93, \ldots, 98)$ missing and 90% random voxel missing (see Fig. 6).

Fig. 7 shows the recovery results by MDT-Tucker and the proposed method with delay window size $\boldsymbol{\tau} = (\tau, \tau, \tau)$ $(\forall \tau \in \{8, 16, 32, 48, 64\})$. In all the $\tau$, the ranks increment one by one: $\boldsymbol{L}_n = (1, 2, 3, 4, \ldots, \tau)$ $(\forall n)$. The recovery using MDT-Tucker with $\tau \geq 16$ cannot be computed because of the huge memory requirements and we cannot obtain the results. Tab. 4 shows the PSNR, the average of SSIM of all the slices, and the computing time of recovery MRI by the two methods. When $\tau = 8$, the proposed method is better recovery than MDT-Tucker. As it $\tau$ is increased, the recovery accuracy also improves. However, the accuracy does not improve with $\tau \geq 48$.

## 5. Limitations and Conclusions

In this study, we proposed a fast and efficient algorithm for MDT-Tucker. For this purpose, we modify MDT-Tucker by introducing a circulant version of MDT, one step optimization with inverse MDT, and reduction of half of factor matrices. As the results, optimization algorithm can be efficiently computed by using FFT, and it is almost 100 times faster than original MDT-Tucker while maintaining high accuracy. Furthermore, the proposed method allows us to compute MDT-Tucker with larger $\tau$ which is not applicable in the original algorithm.

The differences of results come from the modifications of MDT-Tucker in this study.

The first modification is normal MDT to circulant MDT. The introduction of circulant MDT means assuming the target signal as a periodic signal. In other words, this is a method of image modeling for some image such that the left and right edges are connected. This change directly contributes to the speedup using the Fourier transform, but it will produce errors for the image restoration of signals that are not inherently periodic. In contrast, it has little effect on images of which the background is zero, such as MRI.

The second important modification is the reduction of half of factor matrices. This modification makes a difference of the image model. In contrast that the MDT-Tucker captures low-rank structure for all-modes, the proposed method captures low-rank structure for only odd-modes. Odd and even modes in a Hankel tensor are respectively corresponding to local and global patterns in an original tensor. Thus, the proposed method well captures the similarity of only local patterns in an image, but not for global patterns. From experimental results, the effect of the lack of a global low-rank structure was significant under extremely ill-posed setting (*e.g.*, 95% missing).

Our experiments show the proposed method performs good image reconstruction with appropriate $\tau$. However, the value of $\tau$ should be manually tuned depending on the input signal. Automatic selection of $\tau$ is very important for real world applications, and it is included in future works.

## References

[1] Evrim Acar, Daniel M Dunlavy, Tamara G Kolda, and Morten Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011. 1

[2] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000. 3, 5

[3] Tao Ding, Mario Sznaier, and Octavia I Camps. A rank minimization approach to video inpainting. In *Proceedings of ICCV*, pages 1–8. IEEE, 2007. 1

[4] Weiyang Ding, Liqun Qi, and Yimin Wei. Fast Hankel tensor–vector product and its application to exponential data fitting. *Numerical Linear Algebra with Applications*, 22(5):814–832, 2015. 1, 2, 3

[5] Matteo Frigo and Steven G Johnson. FFTW: An adaptive software architecture for the FFT. In *Proceedings of ICASSP*, volume 3, pages 1381–1384. IEEE, 1998. 2

[6] Silvia Gandy, Benjamin Recht, and Isao Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2), 2011. 1

[7] David R Hunter and Kenneth Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004. 3, 4

[8] Daniel Kressner, Michael Steinlechner, and Bart Vandereycken. Low-rank tensor completion by Riemannian optimization. *BIT Numerical Mathematics*, 54(2):447–468, 2014. 1

[9] Ye Li, KJ Ray Liu, and Javad Razavilar. A parameter estimation scheme for damped sinusoidal signals based on low-rank Hankel approximation. *IEEE Transactions on Signal Processing*, 45(2):481–486, 1997. 1

[10] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. In *Proceedings of ICCV*, pages 2114–2121. IEEE, 2009. 1

[11] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2013. 1

[12] Zhen Long, Yipeng Liu, Longxi Chen, and Ce Zhu. Low rank tensor completion for multiway visual data. *Signal Processing*, 155:301–316, 2019. 1

[13] Ling Lu, Wei Xu, and Sanzheng Qiao. A fast SVD for multilevel block Hankel matrices with minimal memory storage. *Numerical Algorithms*, 69(4):875–891, 2015. 1, 2, 3

[14] Ivan Markovsky. Structured low-rank approximation and its applications. *Automatica*, 44(4):891–909, 2008. 1, 2

[15] JM Ortega and WC Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*, volume 30. SIAM, 1970. 3, 4

[16] Farnaz Sedighin and Andrzej Cichocki. Image completion in embedded space using multistage tensor ring decomposition. *Frontiers in Artificial Intelligence*, 4, 2021. 1

[17] Farnaz Sedighin, Andrzej Cichocki, Tatsuya Yokota, and Qiquan Shi. Matrix and tensor completion in multiway delay embedded space using tensor train, with application to signal reconstruction. *IEEE Signal Processing Letters*, 27:810–814, 2020. 1

[18] Qiquan Shi, Jiaming Yin, Jiajun Cai, Andrzej Cichocki, Tatsuya Yokota, Lei Chen, Mingxuan Yuan, and Jia Zeng. Block Hankel tensor ARIMA for multiple short time series forecasting. In *Proceedings of AAAI*, pages 5758–5766, 2020. 1

[19] Qingquan Song, Hancheng Ge, James Caverlee, and Xia Hu. Tensor completion algorithms in big data analytics. *ACM Transactions on Knowledge Discovery from Data*, 13(1):1–48, 2019. 1

[20] Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. Efficient low rank tensor ring completion. In *Proceedings of ICCV*, pages 5697–5705, 2017. 1

[21] Xudong Wang, Luis Miranda-Moreno, and Lijun Sun. Hankel-structured tensor robust PCA for multivariate traffic time series anomaly detection. *arXiv preprint arXiv:2110.04352*, 2021. 1

[22] Xudong Wang, Yuankai Wu, Dingyi Zhuang, and Lijun Sun. Low-rank Hankel tensor completion for traffic speed estimation. *arXiv preprint arXiv:2105.11335*, 2021. 1

[23] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 7

[24] Wei Xu and Sanzheng Qiao. A fast symmetric SVD algorithm for square Hankel matrices. *Linear Algebra and its Applications*, 428(2-3):550–563, 2008. 1, 2, 3

[25] Tatsuya Yokota, Cesar F. Caiafa, and Qibin Zhao. Tensor methods for low-level vision. In Yipeng Liu, editor, *Tensors for Data Processing: Theory, Methods, and Applications*, chapter 11, pages 371–425. Academic Press Inc Elsevier Science, 2021. 1

[26] Tatsuya Yokota, Burak Erem, Seyhmus Guler, Simon K Warfield, and Hidekata Hontani. Missing slice recovery for tensors using a low-rank model in embedded space. In *Proceedings of CVPR*, pages 8251–8259, 2018. 1, 2, 3, 5, 7

[27] Tatsuya Yokota and Hidekata Hontani. Simultaneous visual data completion and denoising based on tensor rank and total variation minimization and its primal-dual splitting algorithm. In *Proceedings of CVPR*, pages 3732–3740, 2017. 7

[28] Tatsuya Yokota and Hidekata Hontani. Tensor completion with shift-invariant cosine bases. In *Proceedings of Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1325–1333. IEEE, 2018. 1

[29] Tatsuya Yokota, Qibin Zhao, and Andrzej Cichocki. Smooth PARAFAC decomposition for tensor completion. *IEEE Transactions on Signal Processing*, 64(20):5423–5436, 2016. 7

[30] Longhao Yuan, Chao Li, Jianting Cao, and Qibin Zhao. Rank minimization on tensor ring: an efficient approach for tensor decomposition and completion. *Machine Learning*, 109(3):603–622, 2020. 1