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.

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.
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 ⊗ for Hadamard product. In addition, let us consider an incomplete input tensor and its mask tensor respectively.

\[ \mathcal{T} \]

Then, we define a delay embedding of a vector as

\[ x = (x_1, x_2, \ldots, x_T)^T \in \mathbb{R}^T \]

with delay window size \( \tau \)

\[ \mathcal{H}_\tau(x) := \begin{pmatrix} x_1 & x_2 & \cdots & x_{T-\tau+1} \\ x_2 & x_3 & \cdots & x_{T-\tau+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_\tau & x_{\tau+1} & \cdots & x_T \end{pmatrix} \in \mathbb{R}^{T \times (T-\tau+1)}. \]

where \( \mathcal{H}_\tau(\cdot) \) is an operator of 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 \( \mathcal{H}(\cdot) \), and the delay embedding for a vector \( x \) is defined as

\[ \mathcal{H}_\tau(x) := \begin{pmatrix} x_1 & x_2 & \cdots & x_{T-\tau+1} \\ x_2 & x_3 & \cdots & x_{T-\tau+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_\tau & x_{\tau+1} & \cdots & x_T \end{pmatrix} \in \mathbb{R}^{T \times (T-\tau+1)}. \]

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

\[ \tilde{\mathcal{H}}_\tau(x) := \text{fold}_{(\tau, T-\tau+1)}(Sx), \]

where \( \text{fold}_{(\tau, V)} : \mathbb{R}^{v \times V} \rightarrow \mathbb{R}^{v \times \tau \times V} \) 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 \( \tilde{X}_H \in \mathbb{R}^{\tau \times (T-\tau+1)} \), the inverse MDT of \( \tilde{X}_H \) is defined as

\[ \mathcal{H}_\tau^{-1}(\tilde{X}_H) := \tilde{S}^\dagger \text{vec}(\tilde{X}_H) \in \mathbb{R}^{\tau}, \]

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

2.1.2 Tensor extension (Step 1 and Step 3)

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

MDT-Tucker Proposed (circulant)

(a) Delay embedding

\[ x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 7 \end{pmatrix}, \quad \tau = 3 \Rightarrow \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}, \quad \overline{\mathbf{h}}_S(x) = \text{fold}_{\{1,3\}}(\mathbf{S}x) \]

(b) Inverse delay embedding

\[ \overline{\mathbf{h}}_S(\overline{\mathbf{X}}_H) = \text{fold}_{\{1,3\}}(\mathbf{S}x) \]

Proposed (circulant)

(a) Circular delay embedding

\[ x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{pmatrix}, \quad \tau = 3 \Rightarrow \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{pmatrix}, \quad \overline{\mathbf{h}}_S(x) = \text{fold}_{\{1,3\}}(\mathbf{S}x) \]

(b) Inverse circular delay embedding

\[ \overline{\mathbf{h}}_S(\overline{\mathbf{X}}_H) = \text{fold}_{\{1,3\}}(\mathbf{S}x) \]

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

\[ \overline{\mathbf{H}}_\tau(\mathbf{X}) := \text{fold}_{\{\tau-\tau_1\}}(\mathbf{X} \times \{\mathbf{S}\}), \quad (11) \]

where \( \text{fold}_{\{u,v\}} : \mathbb{R}^{v_1 \times \cdots \times v_N} \rightarrow \mathbb{R}^{v_1 \times \cdots \times v_N \times v_N} \) is a folding operator from an \( N \)-th order tensor to the \( 2N \)-th order tensor. For a \( 2N \)-th order tensor \( \overline{\mathbf{X}}_H \in \mathbb{R}^{\tau_1 \times (\tau_1-\tau_1+1) \times \cdots \times \tau_N \times (\tau_N-\tau_N+1)} \), the inverse MDT is defined as

\[ \overline{\mathbf{H}}_\tau^1(\overline{\mathbf{X}}_H) := \text{unfold}_{\{\tau-\tau+1\}}(\overline{\mathbf{X}}_H \times \{\mathbf{S}^\dagger\}) \quad (12) \]

where \( \text{unfold}_{\{u,v\}} \) is an inverse transformation of \( \text{fold}_{\{u,v\}} \).

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,

\[
\begin{aligned}
\min_{\mathbf{Q}_H, \{\mathbf{U}_n\}_{n=1}^{2N}} & \| \mathbf{Q}_H \odot (\mathbf{T}_H - \mathbf{G} \times \{\mathbf{U}\}) \|_F^2, \\
\text{s.t.} & \mathbf{G} \in \mathbb{R}^{R_1 \times \cdots \times R_2 \times N}, \\
& \mathbf{U}_n \in \mathbb{R}^{J_n \times R_n}, \quad \mathbf{U}_n^\top \mathbf{U}_n = \mathbf{I}_{J_n}, \\
& R_n \leq J_n \quad (\forall n \in \{1, \ldots, 2N\}).
\end{aligned}
\]

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 \( \mathbf{L}_n \).

2.3. Hint for improvements

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

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 follows:

- 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 $U_{2n-1} \in \mathbb{R}^{T \times n}$, we do not consider even-mode factor matrices (i.e., even-mode factor matrices are assumed as identity matrices $U_{2n} = I_{R_n}$) 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 $x \in \mathbb{R}^T$ with delay window size $\tau$ is defined as

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

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 a circulant shift of $x$ with width $k-1$. In similar way to MDT (see Sec. 2.1), a duplication matrix $S$ 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 $\mathcal{X} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$ with delay window sizes $\tau = (\tau_1, \ldots, \tau_N) \in \mathbb{N}$ can be defined as

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

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

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

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

### 3.3. Reformulated optimization problem

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

$$\text{minimize}_{\mathcal{G}, \{F_n\}_{n=1}^N} \left\| \mathcal{Q} \circ (\mathcal{T} - \mathcal{H}_\tau^\dagger(\mathcal{G} \times \text{odd} \{F\})) \right\|_F^2,$$

subject to $\mathcal{G} \in \mathbb{R}^{R_1 \times T_1 \times \cdots \times R_N \times T_N}$,

$$F_n \in \mathbb{R}^{\tau_n \times R_n}, \quad F_n^\dagger F_n = I_{R_n}, \quad R_n \leq \tau_n \quad (\forall n \in \{1, \ldots, N\}),$$

where $\mathcal{T} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$ and $\mathcal{Q} \in \{0,1\}^{T_1 \times \cdots \times T_N}$ is an input incomplete tensor and a mask tensor, respectively. Note that these are not $\mathcal{T}_H$ and $\mathcal{Q}_H$ obtained by MDT. A resultant tensor is given by $\mathcal{X} = \mathcal{H}_\tau^\dagger \left( \mathcal{G} \times \text{odd} \{F\} \right)$, and it is efficiently calculated (see Sec. 3.5). Although the size of a core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times T_1 \times \cdots \times R_N \times T_N}$ is still large, it is not necessary to be explicitly calculated. In fact, $\mathcal{H}_\tau^\dagger \left( \mathcal{G} \times \text{odd} \{F\} \right)$ can be obtained from a tensor $\mathcal{Z} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$ with a much smaller size in optimization (see Algorithm 2 and Sec. 3.5). Since the circulant MDT $\mathcal{H}_\tau$ and its inverse $\mathcal{H}_\tau^\dagger$ 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 $\mathcal{G}$ 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 problem (17), we consider the minimization of two auxiliary functions: $h$ and $g$. The cost function $f$ and its auxiliary function $h$ are

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

$$h(\theta | \theta') := \| \mathcal{Q} \circ (\mathcal{T} - \mathcal{X}_\theta) \|_F^2 + \| \mathcal{Q} \circ (\mathcal{X}_\theta - \mathcal{X}_{\theta'}) \|_F^2.$$

Now we consider the minimization of $h$. The auxiliary

<table>
<thead>
<tr>
<th>MDT-Tucker</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay embedding</td>
<td>$O(T^N)$</td>
</tr>
<tr>
<td>Formulated steps</td>
<td>3</td>
</tr>
<tr>
<td>Factor matrices</td>
<td>all-modes</td>
</tr>
<tr>
<td>Memory size</td>
<td>$O(\tau^N T N)$</td>
</tr>
<tr>
<td>Computational cost</td>
<td>$O(\tau^N + 1 T N)$</td>
</tr>
</tbody>
</table>

where $\mathcal{T} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$ and $\mathcal{Q} \in \{0,1\}^{T_1 \times \cdots \times T_N}$ are an input incomplete tensor and a mask tensor, respectively. Note that these are not $\mathcal{T}_H$ and $\mathcal{Q}_H$ obtained by MDT. A resultant tensor is given by $\mathcal{X} = \mathcal{H}_\tau^\dagger \left( \mathcal{G} \times \text{odd} \{F\} \right)$, and it is efficiently calculated (see Sec. 3.5). Although the size of a core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times T_1 \times \cdots \times R_N \times T_N}$ is still large, it is not necessary to be explicitly calculated. In fact, $\mathcal{H}_\tau^\dagger \left( \mathcal{G} \times \text{odd} \{F\} \right)$ can be obtained from a tensor $\mathcal{Z} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$ with a much smaller size in optimization (see Algorithm 2 and Sec. 3.5). Since the circulant MDT $\mathcal{H}_\tau$ and its inverse $\mathcal{H}_\tau^\dagger$ 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 $\mathcal{G}$ 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 problem (17), we consider the minimization of two auxiliary functions: $h$ and $g$. The cost function $f$ and its auxiliary function $h$ are

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

$$h(\theta | \theta') := \| \mathcal{Q} \circ (\mathcal{T} - \mathcal{X}_\theta) \|_F^2 + \| \mathcal{Q} \circ (\mathcal{X}_\theta - \mathcal{X}_{\theta'}) \|_F^2,$$

where $\theta = \{\mathcal{G}, F_1, \ldots, F_N\}$ is a set of parameters, $\mathcal{X}_\theta = \mathcal{H}_\tau^\dagger \left( \mathcal{G} \times \text{odd} \{F\} \right)$ is an inverse MDT of Tucker decomposition, and $\mathcal{Q} := 1 - \mathcal{Q}$. Based on the theory of MM algorithm [7, 15], an update rule of $\theta^{k+1} = \text{argmin}_{\theta'} h(\theta | \theta')$ has monotonically non-increasing property for the cost function $f(\theta^{k+1}) \leq f(\theta^k)$.

Next, we consider the minimization of $h$. The auxiliary
Algorithm 1 Tucker-based tensor completion with rank increment in MDT-Tucker [26]

1: input: $\mathcal{T} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$, $\mathcal{Q} \in \{0, 1\}^{T_1 \times \cdots \times T_N}$, $\tau = (\tau_1, \ldots, \tau_N)$, $\{L_1, \ldots, L_{2N}\}$, tol, $\epsilon$
2: initialize: $k_n \leftarrow 1$, $R_n \leftarrow L_n(k_n)$ (\forall n), and $\mathcal{X}_H \in \mathbb{R}^{J_1 \times \cdots \times J_{2N}}$, $\{U_n \in \mathbb{R}^{J_n \times R_n}\}_{n=1}^{2N}$ randomly
3: $T_H \leftarrow \mathcal{H}_r(T)$, and $\mathcal{Q}_H \leftarrow \mathcal{H}_r(Q)$ (Step 1)
4: $f_1 \leftarrow \|Q_H \odot (T_H - X_H)\|_F^2$
5: repeat // (Step 2)
6: $Z_H \leftarrow Q_H \odot T_H + Q_H \odot X_H$
7: for $n = 1, \ldots, 2N$ do
8: $U_n \leftarrow R_n$ leading singular vectors of $[Z_{H \times -n} \{U^T\}]_{(n)}$
9: end for
10: $X_H \leftarrow Z_H \times \{U U^T\}$
11: $f_2 \leftarrow \|Q_H \odot (T_H - X_H)\|_F^2$
12: if $|f_2 - f_1| \leq \text{tol}$ then
13: $X'_{\mathcal{T}} \leftarrow Q_H \odot (T_H - X_H)$
14: $n' \leftarrow \text{argmax}_{n} \|X'_{\mathcal{T} \times -n} \{U^T\}\|_F^2$
15: $k_n' \leftarrow k_n' + 1$, and $R_n' \leftarrow L_n(k_n')$
16: else
17: $f_1 \leftarrow f_2$
18: end if
19: until $f_2 \leq \epsilon$
20: $\tilde{\mathcal{X}} \leftarrow \mathcal{H}_{\tau}^T(X_H)$ (Step 3)
21: output: $\tilde{\mathcal{X}}, U_1, \ldots, U_{2N}$

function $h$ can be transformed as follows:

$$h(\theta) = \|Q \odot (T - X_{\theta})\|_F^2 + \|Q \odot (X_{\theta} - X_{\theta})\|_F^2$$

$$= \|Z_{\theta h} - \mathcal{H}_{\tau}^T(Y_{\theta})\|_F^2,$$  \hspace{20pt} (20)

where $Z_{\theta h} = Q \odot T + Q \odot X_{\theta h}$ and $Y_{\theta} = Q \odot \{F\}$. It is difficult to minimize $h$ directly using the ALS.

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

$$g(\theta) : = \|\mathcal{H}_r(Z_{\theta h}) - Y_{\theta}\|_F^2.$$  \hspace{20pt} (21)

When we regard $\mathcal{H}_r(Z_{\theta h})$ 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 $Y_{\theta}$ are

$$\mathcal{H}_r(\mathcal{H}_r^T(Y_{\theta})) = \mathcal{H}_r(Z_{\theta h}),$$

$$Y_{\theta} = \mathcal{H}_r(Z_{\theta h}).$$  \hspace{20pt} (22)

If the condition of $g$ (23) satisfies, then the condition of $h$ (22) satisfies. Note that $\mathcal{H}_r$ is essentially duplication, and $\mathcal{H}_r^T$ is essentially mean. The function $g$ enforce the Tucker decomposition $Y_{\theta}$ 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: $\mathcal{T} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$, $\mathcal{Q} \in \{0, 1\}^{T_1 \times \cdots \times T_N}$, $\tau = (\tau_1, \ldots, \tau_N)$, $\{L_1, \ldots, L_N\}$, tol, $\epsilon$
2: initialize: $k_n \leftarrow 1$, $R_n \leftarrow L_n(k_n)$ (\forall n), and $\mathcal{X} \in \mathbb{R}^{T_1 \times \cdots \times T_N}$, $\{F_n \in \mathbb{R}^{T_n \times R_n}\}_{n=1}^N$ randomly
3: $f_1 \leftarrow \|Q \odot (T - \mathcal{X})\|_F^2$
4: repeat
5: $Z \leftarrow Q \odot T + Q \odot \mathcal{X}$
6: for $n = 1, \ldots, N$ do
7: $F_n \leftarrow R_n$ leading singular vectors of $[\mathcal{H}_r(Z) \times_{-n} \{F^T\}]_{(2n-1)}$ (Sec. 3.5.1)
8: end for
9: $\mathcal{X} \leftarrow \mathcal{H}_{\tau}^T(\mathcal{X})$ (Step 3)
10: $f_2 \leftarrow \|Q \odot (T - \mathcal{X})\|_F^2$
11: if $|f_2 - f_1| \leq \text{tol}$ then
12: $\mathcal{X}' \leftarrow Q \odot (T - \mathcal{X})$
13: $n' \leftarrow \text{Eq. (24)}$ (Sec. 3.5.3)
14: $k_n' \leftarrow k_n' + 1$, and $R_n' \leftarrow L_n(k_n')$
15: else
16: $f_1 \leftarrow f_2$
17: end if
18: until $f_2 \leq \epsilon$
19: $\mathcal{X}, F_1, \ldots, F_N$
20: output: $\mathcal{X}, F_1, \ldots, F_N$

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:

$$n' = \text{argmax}_{n} \|H_{\tau}(\mathcal{X}') \times_{-n} \{F^T\}\|_F^2,$$  \hspace{20pt} (24)

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

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
The 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. \(H_n \leftarrow [\mathcal{H}_\tau(Z) \times_{-n} \{F^\top\}]_{(2n-1)}\);
2. \(A_n \leftarrow H_n H_n^\top \in \mathbb{R}^{\tau_n \times \tau_n}\);
3. \(F_n \leftarrow R_n\) leading left singular vectors of \(A_n\);

The size of \(H_n\) 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 \(A_n\) is \(\tau_n \times \tau_n\), which is very much smaller than the \(H_n\). Exploiting that the signal after a circulant MDT can be diagonalized, \(A_n\) can be computed without calculating \(H_n\) explicitly. The fast implementation for \(A_n\) is the following processes:

**Proc. 1:** \(Z_F \leftarrow \text{FFT}_N\left(\text{IFFT}_N(Z) \odot \text{IFFT}_N(Z)\right)\);

**Proc. 2:** \(Z_P \leftarrow Z_F \times \{P\} \in \mathbb{R}^{(2\tau_n-1)\times\cdots\times(2\tau_n-1)}\);

**Proc. 3:** \(f_k \leftarrow \text{vec}(F_k F_k^\top) \in \mathbb{R}^{2\tau^2}\) for all \(k \neq n\);

**Proc. 4:** \(a_n \leftarrow Z_P \times_{-n} \{f^\top D\} \in \mathbb{R}^{2\tau_n-1}\);

**Proc. 5:** \(A_n \leftarrow \text{unfold}(D_n, a_n) \in \mathbb{R}^{\tau_n \times \tau_n}\);

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

\[
P_n = \begin{pmatrix}
I_{\tau_n} & 0 & 0 \\
0 & 0 & I_{\tau_n-1}
\end{pmatrix} \in \mathbb{R}^{(2\tau_n-1)\times\tau_n}.
\]  

Fig. 3 shows an example of a duplication matrix \(D_n\).

### 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 \(a \in \mathbb{R}^n\) and \(b \in \mathbb{R}^n\) \((\tau \leq T)\), the elements of \(\mathcal{H}_n^\dagger(ab^\top) \in \mathbb{R}^T\) is

\[
(\mathcal{H}_n^\dagger(ab^\top))(t) = \frac{1}{\tau} \sum_{m=1}^\tau a(m)b(t - m + 1),
\]  

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 \(A = (a_1, ..., a_R) \in \mathbb{R}^{\tau \times R}\) and \(B = (b_1, ..., b_R) \in \mathbb{R}^{\tau \times R}\), the elements of \(\mathcal{H}_n^\dagger(AB^\top) \in \mathbb{R}^T\) is

\[
(\mathcal{H}_n^\dagger(AB^\top))(t) = \sum_{r=1}^R \mathcal{H}_n^\dagger(a_r b_r^\top)(t).
\]

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:** \(F'_n \leftarrow (F_n^\top, 0) \in \mathbb{R}^{\tau_n \times R}
\);

**Proc. 2:** \(f_{\psi}^{(n)} \leftarrow (\text{FFT}_1(F'_n) \odot \text{FFT}_1(F'_n)) 1 (\forall n)\);

**Proc. 3:** \(\mathcal{F} \leftarrow 1 \times \{f_{\psi}^{(n)}\} \in \mathbb{R}^{T_1 \times \cdots \times T_N}\);

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

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

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

\[
\|\mathcal{H}_n (\mathcal{X}') \times_{-n} \{F^\top\}\|_F^2 = \text{tr}(H_n H_n^\top),
\]  

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 \(H_n\) and its size is \(\tau_n \times \tau_n\). Therefore, fast and efficient computation of \(H_n H_n^\top\) can be done by inputting \(\mathcal{X}'\) instead of \(Z\) 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,

\[\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix}, D_n = \begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{pmatrix}
\]

\[D_n x = \begin{pmatrix}
2 & 5 & 1 \\
4 & 1 & 5 \\
2 & 3 & 2
\end{pmatrix} \text{fold}(\cdot) \begin{pmatrix}
1 & 5 & 4 \\
2 & 1 & 5 \\
3 & 2 & 1
\end{pmatrix} \text{vec}(\cdot)
\]

To be increased.

\[\text{Toeplitz matrix}
\]
4.1. Verification 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: $\boldsymbol{\tau} = (32, 32, 1)$, $L_1 = L_2 = (1, 2, 3, 4, 8, 16, 24, 32)$, and $L_3 = 1$. Fig. 4 shows the recovered image and the behaviors of the cost function $f$ and the ranks ($R_1, R_2$). Although our algorithm derived in Sec. 3.4 is not guaranteed to monotonically decrease the cost function exactly, our experiment shows decreasing monotonically the cost function.

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: nuclear-norm and TV regularization (LR&TV) [27], SPCQV (constrained PARAFAC tensor decomposition) [29], MDT-Tucker [26].

We prepared three video data whose size are $\left(112 \times 160 \times 3 \times 32\right)$, $\left(90 \times 160 \times 3 \times 64\right)$, and $\left(90 \times 160 \times 3 \times 100\right)$ 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 compared the 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 naive one in all the lines.

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 $\left(217 \times 181 \times 181\right)$ and

---

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

1 smoke1 and smoke2 are obtained from NHK CREATIVE LIBRARY https://www.nhk.or.jp/archives/creative/, and these were down-sampled for the experiments.
When \( \tau \) 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


