Low-Rank Triple Decomposition of Streaming Tensors and Its Application to Video Completion

Thanh Trung Le*§, Nguyen Hong Thinh*, Luu Manh Ha*, Tran Thi Thuy Quynh*, Le Vu Ha*,
Vy-Thuy-Lynh Hoang†, Karim Abed-Meraim‡, Philippe Ravier‡, Olivier Buttelli‡

* VNU University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam
† BRGM French Geological Survey, France
‡ PRISME Laboratory, University of Orleans, France

***Abstract*—Triple tensor decomposition (TriTD) has recently emerged as a good alternative for the two classical CP/PARAFAC and Tucker tensor decompositions. However, current state-of-the-art TriTD methods are primarily designed for batch processing, requiring the storage and processing the entire tensor data at once. In this paper, we propose, for the first time, an effective and efficient incremental method for online tensor decomposition in the TriTD format. We validate the performance of our method with the task of online video completion. Experimental results demonstrate that our method outperforms state-of-the-art streaming tensor decomposition methods.**

***Index Terms*—Tensor decomposition, triple decomposition, online learning, missing data, video completion**

I. Introduction

Recently, Qi *et al.* [1] proposed a new tensor format for representing three-way tensors using three simpler tensor factors of the same order, called triple decomposition (TriTD). Given a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$, its TriTD, denoted as $\boldsymbol{\mathcal{X}} = [[\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}]]$, is given by

$$\boldsymbol{\mathcal{X}}_{i,j,k} = \sum_{r_1=1}^{R}\sum_{r_2=1}^{R}\sum_{r_3=1}^{R} \boldsymbol{\mathcal{A}}_{i,r_2,r_3}\boldsymbol{\mathcal{B}}_{r_1,j,r_3}\boldsymbol{\mathcal{C}}_{r_1,r_2,k}, \qquad (1)$$

where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I\times R\times R}$, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{R\times J\times R}$, and $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{R\times R\times K}$ are tensor factors of the third order. The smallest R satisfying (1) refers to the triple rank of $\boldsymbol{\mathcal{X}}$.

TriTD can be regarded as an extension of the most two widely-used tensor decompositions: CP/PARAFAC and Tucker/HOSVD (see [2] for a good overview). We refer readers to [1] for the relationship among these three decompositions. In practice, TriTD and its variants (e.g., non-negative and hypergraph-regularized versions) have shown promise in applications, such as image compression [3], network traffic recovery [4], and data clustering [5].

In recent years, there has been a significant increase in the demand for processing (big) data streams, particularly in the context of streaming tensor decomposition [6]. Many online, adaptive, and incremental methods have been proposed for factorizing and tracking streaming tensors over time. Most of these methods are based on the two most well-known and widely-used tensor formats: CP/PARAFAC and Tucker. These methods are based on stochastic techniques (e.g., [7]–[9]), recursive least-squares filtering (e.g., [10]–[12]), and Bayesian inference (e.g., [13]–[15]), to name a few. There have also been efforts to develop streaming tensor decomposition methods under other formats, such as BTD [16]–[18], tensor train [19]–[22], tensor ring [23]–[25], t-SVD [26], and UTV [27]. For a comprehensive overview on online methods for streaming tensor decomposition and tracking, we refer the reader to our recent survey [6].

The recent emergence of TriTD highlights a gap in the tensor literature, as no method has yet been specifically developed for online triple tensor decomposition of streaming tensors. TriTD offers a more balanced tensor representation compared to CP and Tucker formats. The triple rank is not greater than the CP rank or the middle value of the Tucker rank. This makes TriTD a viable and compact alternative whenever the tensor data can be effectively represented using CP or Tucker decompositions. In such cases, a low-rank tensor representation via TriTD can also be achieved [1]. Consequently, TriTD is applicable in scenarios where CP and Tucker decompositions have been successful, thereby extending their potential to new contexts. Moreover, the representation properties of tensor formats, including TriTD, are preserved in the streaming setting, similar to those in the batch setting. These observations suggest that developing online, incremental, adaptive TriTD methods for factorizing streaming tensors is a promising and important research direction. In this paper, we address this gap by proposing, for the first time, an efficient incremental online method for factorizing streaming tensors in the TriTD format. The proposed method, called OTD (which stands for Online Triple Decomposition), effectively tracks low-rank components of streaming tensors in the presence of noise, missing data, and abrupt changes. Moreover, OTD successfully addresses the task of video completion.

Notations: Lowercase letters represent scalars (e.g., x), while boldface lowercase letters indicate vectors (e.g., $\mathbf{x}$). Matrices are denoted using boldface capital letters (e.g., $\mathbf{X}$), and tensors are represented using bold calligraphic letters (e.g., $\boldsymbol{\mathcal{X}}$). The (i_1, i_2, i_3)-th element of $\boldsymbol{\mathcal{X}}$ is denoted as

§ This work was funded by the Postdoctoral Scholarship Programme of Vingroup Innovation Foundation (VINIF), code VINIF.2024.STS.40. Corresponding author: Thanh Trung Le (thanhletrung@vnu.edu.vn).

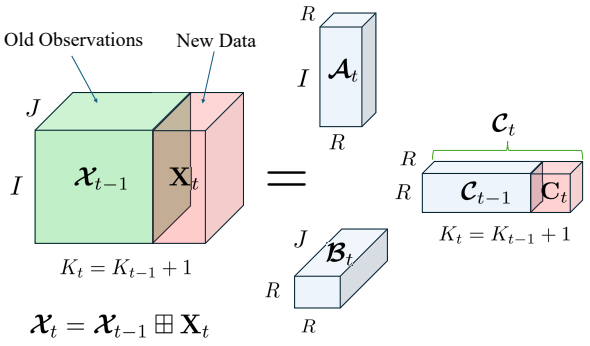


Fig. 1: Online Triple Tensor Decomposition

$\mathcal{X}_{i_1,i_2,i_3}$ or $\mathcal{X}(i_1,i_2,i_3)$. The k-th frontal slices of $\mathcal{X}$ is expressed as $\mathcal{X}_{:,:,k}$ or $\mathcal{X}(:,:,k)$. The mode-n unfolding matrix of $\mathcal{X}$ is denoted by $[\mathcal{X}]_{(n)}$. The transpose operation is indicated by $(\cdot)^\top$, and the Frobenius norm is denoted as $\|\cdot\|_F$. The symbol $\otimes$ represents the Kronecker product. $\mathbf{X} \succeq \mathbf{Y}$ (resp. $\succ$), then $\mathbf{X}-\mathbf{Y}$ is a positive semidefinite (resp. positive definite) matrix.

II. Online Triple Tensor Decomposition

Without loss of generality, we assume that the third dimension of the underlying streaming tensor varies with time. Accordingly, we define a streaming tensor $\mathcal{X}_t \in \mathbb{R}^{I\times J\times K_t}$ that is observed sequentially in its frontal slices. At each time t, $\mathcal{X}_t$ is formed by stacking new data $\mathbf{X}_t \in \mathbb{R}^{I\times J}$ with the previous data $\mathcal{X}_{t-1} \in \mathbb{R}^{I\times J\times K_{t-1}}$ along the third dimension (i.e., $\mathcal{X}_t(:,:,t) = \mathbf{X}_t$ and $K_t = K_{t-1}+1$). See Fig. 1 for an illustration.

Here, the new data $\mathbf{X}_t$ is assumed to consist of a main component $\mathbf{L}_t$ plus a Gaussian noise $\mathbf{N}_t$:[1]

$$\mathbf{X}_t = \mathbf{L}_t + \mathbf{N}_t = [\mathcal{A}_t]_{(1)}\big(\mathbf{I}_R \otimes \mathbf{C}_t^\top\big)[\mathcal{B}_t]_{(2)}^\top + \mathbf{N}_t. \quad (2)$$

Here, two factors $\mathcal{A}_t \in \mathbb{R}^{I\times R\times R}$ and $\mathcal{B}_t \in \mathbb{R}^{R\times J\times R}$ are of fixed size, and we refer them to as non-temporal factors. The matrix $\mathbf{C}_t \in \mathbb{R}^{R\times R}$ is the t-th frontal slice of the temporal factor $\mathcal{C}_t \in \mathbb{R}^{R\times R\times K_t}$.

While it is possible to apply batch TriTD methods (e.g., [1]) to decompose $\mathcal{X}_t$ at each time t, this approach becomes inefficient and ineffective regarding both computational complexity and memory storage in online settings, particularly when K_t is huge and/or when factors $\mathcal{A}_t$ and $\mathcal{B}_t$ vary slowly over time. Therefore, to enable online processing, we utilize the following objective function:

$$\{\mathcal{A}_t,\mathcal{B}_t,\mathcal{C}_t\} = \underset{\mathcal{A},\mathcal{B},\mathcal{C}}{\operatorname{argmin}} \left\{ f_t\big(\mathcal{A},\mathcal{B}\big) = \sum_{k=1}^{K_t} \lambda^{K_t-k}\ell\big(\mathcal{A},\mathcal{B}|\mathbf{X}_k\big)\right\}, \quad (3)$$

where the loss function $\ell(\cdot)$ is defined as follows

$$\ell\big(\mathcal{A},\mathcal{B} \mid \mathbf{X}_k\big) = \min_{\mathbf{C}_k} \tilde{\ell}\big(\mathbf{C}_k|\mathbf{X}_k,\mathcal{A},\mathcal{B}\big), \quad (4)$$

[1]Note that if $[[\mathcal{A},\mathcal{B},\mathcal{C}]] = \mathcal{X}$, the k-th frontal slice of $\mathcal{X}$ can be expressed as $\mathcal{X}_{:,:,k} = [\mathcal{A}]_{(1)}\big(\mathbf{I}_R \otimes \mathcal{C}_{:,:,k}^\top\big)[\mathcal{B}]_{(2)}^\top \ \forall k$, thanks to (1).

with

$$\tilde{\ell}\big(\mathbf{C}_k|\mathbf{X}_k,\mathcal{A},\mathcal{B}\big) = \big\|\mathbf{X}_k - [\mathcal{A}]_{(1)}\big(\mathbf{I}_R \otimes \mathbf{C}_k^\top\big)[\mathcal{B}]_{(2)}^\top\big\|_F^2, \quad (5)$$

and $0 < \lambda \le 1$ is a forgetting factor to reduce the effect of past observations. When $\lambda = 1$, the objective function $f_t(\mathcal{A},\mathcal{B})$ in (3) is a sum of loss functions over all observations (aka tensor slices) $\{\mathbf{X}_k\}_{k=1}^t$. This formulation is useful in stationary environments where the factors $\mathcal{A}$ and $\mathcal{B}$ do not change with time. When $0 < \lambda < 1$, the weight λ^{K_t-k} becomes small if $K_t - k$ is large, indicating that the contribution of $\ell(\mathcal{A},\mathcal{B} \mid \mathbf{X}_k)$ to the objective function $f_t(\mathcal{A},\mathcal{B})$ is minimal for values of k far from t. In nonstationary and dynamic environments where older observations may no longer reflect accurately the true underlying data model at time t, the use of $0 < \lambda < 1$ exponentially discounts the effect of these observations. This ensures that observations from the distant past have a significantly reduced influence in the objective function compared to more recent ones.

Assuming that the non-temporal factors are either fixed or change slowly with time ($\mathcal{A}_t \cong \mathcal{A}_{t-1}$ and $\mathcal{B}_t \cong \mathcal{B}_{t-1}$) and the triple rank R is fixed, we can estimate the t-th frontal slice of the temporal factor $\mathcal{C}_t$ upon the arrival of new data $\mathbf{X}_t$ from $\mathbf{C}_t = \operatorname{argmin}_{\mathbf{C}} \tilde{\ell}\big(\mathbf{C} \mid \mathbf{X}_t, \mathcal{A}_{t-1},\mathcal{B}_{t-1}\big)$ in (4). Now, at each time t, given a set of estimates $\{\mathbf{C}_k\}_{k=1}^{K_t}$, we propose to minimize the following surrogate function $\tilde{f}_t(\cdot)$ of $f_t(\cdot)$:

$$\big\{\mathcal{A}_t,\mathcal{B}_t\big\} = \underset{\mathcal{A},\mathcal{B}}{\operatorname{argmin}}\, \tilde{f}_t\big(\mathcal{A},\mathcal{B}\big), \quad \text{where} \quad (6)$$

$$\tilde{f}_t\big(\mathcal{A},\mathcal{B}\big) = \sum_{k=1}^{K_t} \lambda^{t-k}\Big\|\mathbf{X}_k - [\mathcal{A}]_{(1)}\big(\mathbf{I}_R \otimes \mathbf{C}_k^\top\big)[\mathcal{B}]_{(2)}^\top\Big\|_F^2. \quad (7)$$

It stems from the three observations that: (i) $\tilde{f}_t(\cdot)$ is an upper bound of $f_t(\cdot)$ $\forall t$; (ii) solving (6) is "easier" than (3); and (iii) in our convergence analysis, we show that $\tilde{f}_t(\mathcal{A}_t,\mathcal{B}_t)$ and $f_t(\mathcal{A}_t,\mathcal{B}_t)$ converge almost surely to the same limit. Consequently, the solution $\{\mathcal{A}_t,\mathcal{B}_t\}$ obtained by minimizing $\tilde{f}_t(\mathcal{A},\mathcal{B})$ coincides with the solution of $f_t(\mathcal{A},\mathcal{B})$ when t goes to infinity.

III. Proposed Method

On the arrival of new data $\mathbf{X}_t$ at each time t, our optimization procedure follows two steps: (i) First, we estimate $\mathbf{C}_t$ using previous estimates $\mathcal{A}_{t-1}$ and $\mathcal{B}_{t-1}$, which then serve as the t-th frontal slice of $\mathcal{C}_t$. (ii) Next, we recursively update $\mathcal{A}_t$ and $\mathcal{B}_t$ based on the newly estimated $\mathbf{C}_t$ along with $\mathcal{A}_{t-1}$ and $\mathcal{B}_{t-1}$.

1) Estimation of temporal tensor factor $\mathcal{C}_t$: Interestingly, we exploit that minimizing the loss function $\tilde{\ell}\big(\mathbf{C} \mid \mathbf{X}_t,\mathcal{A}_{t-1},\mathcal{B}_{t-1}\big)$ with regularization at time t is equivalent to

$$\mathbf{C}_t = \underset{\mathbf{C}\in\mathbb{R}^{R\times R}}{\operatorname{argmin}} \ \big\|\operatorname{vec}(\mathbf{X}_t) - \mathbf{H}_t\operatorname{vec}(\mathbf{C})\big\|_F^2 + \rho_{\mathcal{C}}\big\|\operatorname{vec}(\mathbf{C})\big\|_2^2,$$

$$\text{where } \mathbf{H}_t = \sum_{r_3}^{R}\begin{bmatrix} \mathcal{B}_{t-1}(:,:,r_3)\otimes\mathcal{A}_{t-1}(:,1,r_3)^\top \\ \mathcal{B}_{t-1}(:,:,r_3)\otimes\mathcal{A}_{t-1}(:,2,r_3)^\top \\ \cdots \\ \mathcal{B}_{t-1}(:,:,r_3)\otimes\mathcal{A}_{t-1}(:,R,r_3)^\top \end{bmatrix}^\top. \quad (8)$$

Here, we introduce the regularization term $\rho_{\mathcal{C}}\|\operatorname{vec}(\mathbf{C})\|_2^2$ to enhance the conditioning of the optimization, particularly in situations where $\mathbf{H}_t$ may be singular or nearly singular. Interestingly, the problem (8) is a convex regularized least-squares (LS) minimization, and the closed-form solution for $\mathbf{C}_t$ is given by

$$\operatorname{vec}(\mathbf{C}_t) = \left(\mathbf{H}_t^\top \mathbf{H}_t + \rho_{\mathcal{C}} \mathbf{I}_R\right)^{-1} \mathbf{H}_t^\top \operatorname{vec}(\mathbf{X}_t). \tag{9}$$

It is also worth noting that the matrix $\mathbf{H}_t$ in (8) is of a Kronecker-structure. We can speed up the LS estimation by utilizing randomized/sketching LS methods [28] to solve (8) more efficiently.

2) Estimation of non-temporal factors $\boldsymbol{\mathcal{A}}_t$ and $\boldsymbol{\mathcal{B}}_t$: We employ the alternating minimization framework to solve (6). In what follows, we present the update rule of $\boldsymbol{\mathcal{A}}_t$ only while $\boldsymbol{\mathcal{B}}_t$ is updated in the same way as $\boldsymbol{\mathcal{A}}_t$.

The factor $\boldsymbol{\mathcal{A}}_t$ is obtained from sub-problems of (6):

$$\begin{aligned}\boldsymbol{\mathcal{A}}_t = \underset{\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I \times R \times R}}{\operatorname{argmin}} \sum_{k=1}^{K_t} \lambda^{K_t-k} \left\|\mathbf{X}_k - [\boldsymbol{\mathcal{A}}]_{(1)} \mathbf{W}_{k,\boldsymbol{\mathcal{A}}}\right\|_F^2 \\ + \eta \|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{A}}_{t-1}\|_F^2,\end{aligned} \tag{10}$$

where $\mathbf{W}_{k,\boldsymbol{\mathcal{A}}} = (\mathbf{I}_R \otimes \mathbf{C}_k^\top)[\boldsymbol{\mathcal{B}}_{t-1}]_{(2)}^\top$. The penalty term $\eta\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{A}}_{t-1}\|_F^2$ is included to regulate the time variation of $\boldsymbol{\mathcal{A}}$ as well as to establish momentum for their estimate at time t.

Update rules: We propose the following incremental recursive update rule to address (10):

$$\begin{aligned}[\boldsymbol{\mathcal{A}}_t]_{(1)} = [\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} + \left(\mathbf{X}_t - [\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}\right) \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top \mathbf{Q}_{t,\boldsymbol{\mathcal{A}}}^{-1} \\ + \lambda\eta \left([\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} - [\boldsymbol{\mathcal{A}}_{t-2}]_{(1)}\right) \mathbf{Q}_{t,\boldsymbol{\mathcal{A}}}^{-1},\end{aligned} \tag{11}$$

where $\mathbf{Q}_{t,\boldsymbol{\mathcal{A}}}$ of size $R^2 \times R^2$ is recursively updated over time as follows

$$\mathbf{Q}_{t,\boldsymbol{\mathcal{A}}} = \lambda \mathbf{Q}_{t-1,\boldsymbol{\mathcal{A}}} + (1-\lambda)\eta \mathbf{I}_{R^2} + \mathbf{W}_{t,\boldsymbol{\mathcal{A}}} \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top. \tag{12}$$

To enable the recursive procedures (11), at $t = 0$, we set $\mathbf{Q}_{0,\boldsymbol{\mathcal{A}}} = \mathbf{I}_{R^2}$, initialize $\boldsymbol{\mathcal{A}}_0$ at random, and define $\boldsymbol{\mathcal{A}}_{-1} = \mathbf{0}_{I \times R \times R}$. Here, we observe that the second term in (11) comes from the first-order optimality condition of the original problem (6). The third term, on the other hand, represents momentum of the solutions, which aids in accelerating the optimization process [29].

Derivations of (11): Taking the first derivative of (10) to zeros leads to

$$\begin{aligned}[\boldsymbol{\mathcal{A}}_t]_{(1)} \sum_{k=1}^{K_t} \lambda^{K_t-k} \left(\mathbf{W}_{k,\boldsymbol{\mathcal{A}}} \mathbf{W}_{k,\boldsymbol{\mathcal{A}}}^\top + \eta \mathbf{I}_{R^2}\right) \\ = \sum_{k=1}^{K_t} \lambda^{K_t-k} \mathbf{X}_k \mathbf{W}_{k,\boldsymbol{\mathcal{A}}}^\top + \eta [\boldsymbol{\mathcal{A}}_{t-1}]_{(1)}.\end{aligned} \tag{13}$$

Let us denote the right-hand side of (13) as $\mathbf{R}_{t,\boldsymbol{\mathcal{A}}}$ and obtain $\mathbf{R}_{t,\boldsymbol{\mathcal{A}}} = \lambda \mathbf{R}_{t-1,\boldsymbol{\mathcal{A}}} + \mathbf{X}_t \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top + \eta\left([\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} - \lambda[\boldsymbol{\mathcal{A}}_{t-2}]_{(1)}\right)$. We then express (13) as

$$\begin{aligned}
&[\boldsymbol{\mathcal{A}}_t]_{(1)} \mathbf{Q}_{t,\boldsymbol{\mathcal{A}}} = \mathbf{R}_{t,\boldsymbol{\mathcal{A}}} \\
&= \lambda \mathbf{R}_{t-1,\boldsymbol{\mathcal{A}}} + \mathbf{X}_t \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top + \eta\left([\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} - \lambda[\boldsymbol{\mathcal{A}}_{t-2}]_{(1)}\right) \\
&= \lambda [\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} \mathbf{Q}_{t-1,\boldsymbol{\mathcal{A}}} + \mathbf{X}_t \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top + \eta\left([\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} - \lambda[\boldsymbol{\mathcal{A}}_{t-2}]_{(1)}\right) \\
&= [\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} \left(\lambda \mathbf{Q}_{t-1,\boldsymbol{\mathcal{A}}} + (1-\lambda)\eta \mathbf{I}_{R^2} + \mathbf{W}_{t,\boldsymbol{\mathcal{A}}} \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top\right) \\
&\quad + \mathbf{X}_t \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top - [\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} \left(\mathbf{W}_{t,\boldsymbol{\mathcal{A}}} \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top + (1-\lambda)\eta \mathbf{I}_{R^2}\right) \\
&\quad + \eta\left([\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} - \lambda[\boldsymbol{\mathcal{A}}_{t-2}]_{(1)}\right) \\
&= [\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} \mathbf{Q}_{t,\boldsymbol{\mathcal{A}}} + \left(\mathbf{X}_t - [\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}\right) \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top \\
&\quad + \lambda\eta\left([\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} - [\boldsymbol{\mathcal{A}}_{t-2}]_{(1)}\right).
\end{aligned} \tag{14}$$

Due to the terms $(1-\lambda)\eta \mathbf{I}_{R^2} \succ \mathbf{0}$, $\mathbf{W}_{t,\boldsymbol{\mathcal{A}}} \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top \succeq \mathbf{0} \quad \forall t$, and $\mathbf{Q}_{0,\boldsymbol{\mathcal{A}}} = \mathbf{I}_{R^2} \succ \mathbf{0}$ initialized at $t = 0$, we can ensure that $\mathbf{Q}_{t,\boldsymbol{\mathcal{A}}}$ is always positive-definite and, therefore, invertible. Accordingly, multiplying both sides of (14) with $\mathbf{Q}_{t,\boldsymbol{\mathcal{A}}}^{-1}$ results in the update rule (11).

Parameter selection: In practice, the regularization parameters η and $\rho_{\mathcal{C}}$ can be selected within the range $[10^{-2}, 10^{-4}]$ to achieve reasonable performance across various settings. The forgetting factor λ is typically set around 0.5 in most scenarios. For fast time-varying environments, we can reduce the value of λ to facilitate the tracking ability of OTD. While for slowly time-varying or stationary cases, λ should be close to 1.

3) Dealing with missing data: Assume that missing data of $\mathbf{X}_t$ can be represented by a binary mask Ω_t, where $\Omega_t(i,j) = 0$ indicates that $\mathbf{X}_t(i,j)$ is missing and $\Omega_t(i,j) = 1$ if it is observed. We denote the slice $\mathbf{X}_t$ with its observation mask Ω_t as $(\mathbf{X}_t)_{\Omega_t}$ and use $|\Omega_t|$ to represent the number of observed entries in $\mathbf{X}_t$.

Estimation of $\mathbf{C}_t$: Let $\mathbf{x}_{\Omega_t} = (\operatorname{vec}(\mathbf{X}_t))_{\Omega_t} \in \mathbb{R}^{|\Omega_t|}$, and $\mathbf{H}_{\Omega_t} \in \mathbb{R}^{|\Omega_t| \times R^2}$ is a sub-matrix of $\mathbf{H}_t$ by selecting rows corresponding to observed entries in $\mathbf{x}_{\Omega_t}$. In the presence of missing data, the solution of problem (8) now becomes

$$\operatorname{vec}(\mathbf{C}_t) = \left(\mathbf{H}_{\Omega_t}^\top \mathbf{H}_{\Omega_t} + \rho_{\mathcal{C}} \mathbf{I}_{R^2}\right)^{-1} \mathbf{H}_{\Omega_t}^\top \mathbf{x}_{\Omega_t}. \tag{15}$$

To prevent the underdetermined problem and guarantee the unique optimal solution for $\mathbf{C}_t$ as defined in (15), the number of observed entries, $|\Omega_t|$, must exceed R^2. If $|\Omega_t| < R^2$, additional information about $\mathbf{C}_t$ is required, such as its sparsity. In such cases, other regularized LS methods can be applied, depending on the available side information about $\mathbf{C}_t$. These methods may include LASSO, ℓ_0 regularization, elastic nets, or total variation regularization.

Estimation of $\boldsymbol{\mathcal{A}}_t$ and $\boldsymbol{\mathcal{B}}_t$: We exploit that the second term in (11) involves the error between the new data $\mathbf{X}_t$ and its recovered version based on the previous estimation $\boldsymbol{\mathcal{A}}_{t-1}$. While the remaining terms are independent of new data. Consequently, we only modify the second terms by disregarding the missing locations, leading to the new update rules as:

$$\begin{aligned}[\boldsymbol{\mathcal{A}}_t]_{(1)} = [\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} + \left(\mathbf{X}_t - [\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}\right)_{\Omega_t} \mathbf{W}_{t,\boldsymbol{\mathcal{A}}}^\top \mathbf{Q}_{t,\boldsymbol{\mathcal{A}}}^{-1} \\ + \lambda\eta \left([\boldsymbol{\mathcal{A}}_{t-1}]_{(1)} - [\boldsymbol{\mathcal{A}}_{t-2}]_{(1)}\right) \mathbf{Q}_{t,\boldsymbol{\mathcal{A}}}^{-1}.\end{aligned} \tag{16}$$

Here, $\mathbf{Q}_{t,\boldsymbol{\mathcal{A}}}$ is updated recursively as in (12).

IV. Performance Analysis

A. Computational Complexity

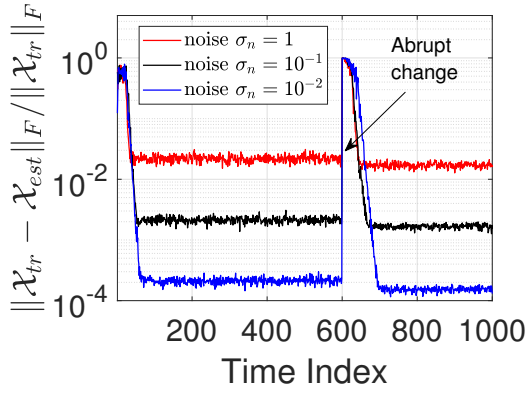
The computational complexity of OTD comes from two stages: (i) the estimation of $\mathbf{C}_t$ and (ii) the estimation of $\boldsymbol{\mathcal{A}}_t$ and $\boldsymbol{\mathcal{B}}_t$. In stage (i), forming $\mathbf{H}_t$ from $\boldsymbol{\mathcal{A}}_{t-1}$ and $\boldsymbol{\mathcal{B}}_{t-1}$ requires $\mathcal{O}\big(IJR^3 + IJR^2\big)$ flops, while computing the least-squares solution $\mathbf{C}_t$ needs $\mathcal{O}\big(IJR^4\big)$ flops. Consequently, the total complexity for this stage is $\mathcal{O}\big(IJR^4\big)$ which reduces to $\mathcal{O}\big(|\Omega|R^4\big)$ in the case of missing data. In stage (ii), the computation of $\mathbf{W}_{t,\boldsymbol{\mathcal{A}}}$ and $\mathbf{W}_{t,\boldsymbol{\mathcal{B}}}$ incurs a total cost of $\mathcal{O}\big((I+J)R^4\big)$ flops. Computing $\mathbf{Q}_{t,\boldsymbol{\mathcal{A}}}$, $\mathbf{Q}_{t,\boldsymbol{\mathcal{B}}}$ and their inverses requires $\mathcal{O}\big((I+J)R^4+R^6\big)$ flops in total. Finally, the recursive updates of two non-temporal factors $\boldsymbol{\mathcal{A}}_t$ and $\boldsymbol{\mathcal{B}}_t$ demand $\mathcal{O}\big((IJ+I+J)(R^4+R^2)\big)$ flops. To sum up, OTD requires a total computational complexity of $\mathcal{O}\big(R^4\max\{IJ,R^2\}\big)$ flops. Since the tensor dimension is often greater than the rank ($IJ > R^2$), the overall complexity of OTD is $\mathcal{O}\big(IJR^4\big)$ flops.

B. Convergence Analysis

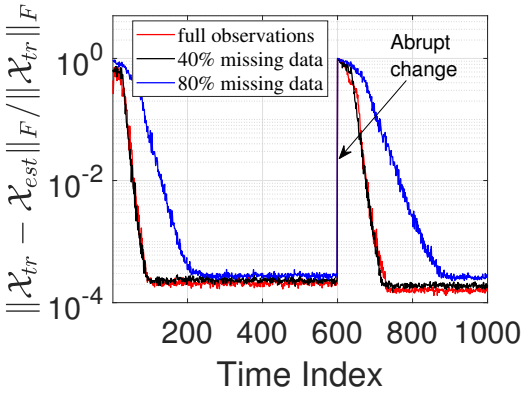
To support our theoretical results, we make the following three assumptions: (A1) Tensor slices $\{\mathbf{X}_t\}_{t=1}^{\infty}$ are norm-bounded matrices, i.e., $\|\mathbf{X}_t\|_F < M_x < \infty$. The main components $\{\mathbf{L}_t\}_{t=1}^{\infty}$ are deterministic, while the entries of noise components $\{\mathbf{N}_t\}_{t=1}^{\infty}$ are i.i.d. from a distribution with a compact support. (A2) The triple rank R is fixed, tensor factors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ are norm-bounded and remain unchanged over time. (A3) The surrogate function $\tilde{f}(\cdot)$ of $f(\cdot)$ is m-strongly multi-block convex, i.e., $\nabla^2\tilde{f}([\boldsymbol{\mathcal{A}}]_{(1)},\cdot) \succeq m_{\boldsymbol{\mathcal{A}}}\mathbf{I} \succ \mathbf{0}$ and $\nabla^2\tilde{f}([\boldsymbol{\mathcal{B}}]_{(2)},\cdot) \succeq m_{\boldsymbol{\mathcal{B}}}\mathbf{I} \succ \mathbf{0}$ with $m_{\boldsymbol{\mathcal{A}}}, m_{\boldsymbol{\mathcal{B}}} > 0$. Indeed, (A1) is a common assumption in online contexts and it applies to many real-life applications, as datasets are typically bounded. (A2) represents the stationary condition which is also commonly used to analyze the convergence and asymptotic behavior of online methods. Additionally, the regularization term in (10) helps ensure that (A3) is satisfied. Now, given (A1-A3), the following theorem demonstrates the convergence of OTD.

Theorem 1. *Assuming that assumptions (A1-A3) are satisfied and $\lambda = 1$, let $\mathcal{D}_t = \{\boldsymbol{\mathcal{A}}_t, \boldsymbol{\mathcal{B}}_t\}$ the solution generated by OTD at time t. As $t \to \infty$, $\mathcal{D}_t$ converges almost surely to a stationary point of $f_t(\cdot)$.*

Proof Sketch: To prove Theorem 1, we utilize the asymptotic convergence framework established for our robust CP decomposition of streaming tensors in [30]. It contains three main stages. (S1) The solutions $\{\mathcal{D}_t\}_{t=1}^{\infty}$ are bounded, and their time variation satisfies $\|\mathcal{D}_t - \mathcal{D}_{t-1}\|_F \to \mathcal{O}(1/t)$ a.s.; (S2) The non-negative sequence $\{\tilde{f}_t(\mathcal{D}_t)\}_{t=1}^{\infty}$ forms a quasi-martingale that converges almost surely, and we have $\tilde{f}_t(\mathcal{D}_t) - f_t(\mathcal{D}_t) \to 0$ a.s.; (S3) The objective function $f_t(\cdot)$ is continuously differentiable and Lipschitz, and as $t \to \infty$, $\nabla f_t(\mathcal{D}_t) \to 0$. Due to the space limit, we omit the details on this proof.



(a) Noisy + full observations.



(b) Missing + noise $\sigma_n = 10^{-2}$.

Fig. 2: Effect of noisy and missing data on performance of the proposed method. The streaming tensor has size $100 \times 100 \times 1000$ with triple-rank $R = 5$. An abrupt change (data drift) occurs at $t = 600$.

V. Numerical Results

In this section, we conduct several numerical experiments to demonstrate the effectiveness of the proposed method with both synthetic and real-world datasets.

A. Synthetic Data

We generate a streaming tensor $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I\times J\times t}$, where the t-th frontal slice $\mathbf{X}_t \in \mathbb{R}^{I\times J}$ is produced under the following model

$$\big(\mathbf{X}_t\big)_{\Omega_t} = \Big([\boldsymbol{\mathcal{A}}_t]_{(1)}\big(\mathbf{I}_R \otimes \mathbf{C}_t^{\top}\big)[\boldsymbol{\mathcal{B}}_t]_{(2)}^{\top} + \sigma_n^2\mathbf{N}_t\Big)_{\Omega_t}. \tag{17}$$

Here, $\Omega_t \in \mathbb{R}^{I\times J}$ is a binary matrix indicating that the (i,j)-th element of $\mathbf{X}_t$ is observed if $\Omega_t(i,j) = 1$ and missing if $\Omega_t(i,j) = 0$. The frontal slice $\mathbf{C}_t \in \mathbb{R}^{I\times J}$ and the noise $\mathbf{N}_t \in \mathbb{R}^{I\times J}$ are Gaussian matrices with entries that are i.i.d. from $\mathcal{N}(0,1)$, and $\sigma_n > 0$ controls the noise level. Two non-temporal tensor factors $\boldsymbol{\mathcal{A}}_t$ and $\boldsymbol{\mathcal{B}}_t$ are modeled as follows

$$\boldsymbol{\mathcal{A}}_t = (1-\epsilon)\boldsymbol{\mathcal{A}}_{t-1} + \epsilon\boldsymbol{\mathcal{N}}_{\boldsymbol{\mathcal{A}}_t}, \quad \boldsymbol{\mathcal{B}}_t = (1-\epsilon)\boldsymbol{\mathcal{B}}_{t-1} + \epsilon\boldsymbol{\mathcal{N}}_{\boldsymbol{\mathcal{B}}_t},$$

where $\boldsymbol{\mathcal{N}}_{\boldsymbol{\mathcal{A}}_t} \in \mathbb{R}^{I\times R\times R}$ and $\boldsymbol{\mathcal{N}}_{\boldsymbol{\mathcal{B}}_t} \in \mathbb{R}^{R\times J\times R}$ represent Gaussian noise with zero mean and unit variance; $0 \le \epsilon \le 1$ controls the time variation. At $t = 0$, $\boldsymbol{\mathcal{A}}_0$ and $\boldsymbol{\mathcal{B}}_0$ are initialized as Gaussian

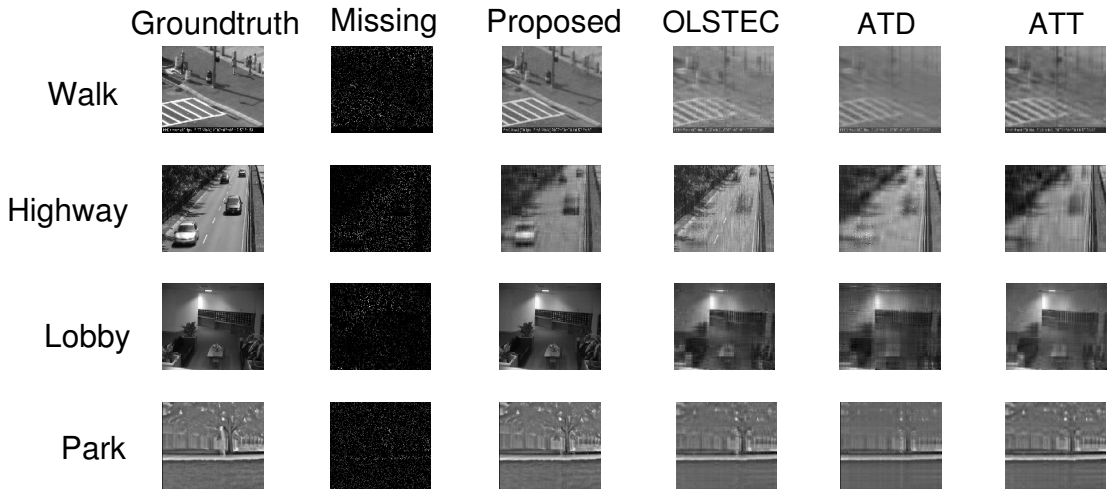


Fig. 3: Performance of tensor-based methods for online video completion with 90% missing data.

tensors with zero mean and unit variance entries. To evaluate the algorithm's performance, we measure the following error

$$\mathrm{ER}(\boldsymbol{\mathcal{X}}_{tr}, \boldsymbol{\mathcal{X}}_{est}) = \|\boldsymbol{\mathcal{X}}_{tr} - \boldsymbol{\mathcal{X}}_{est}\|_F \big/ \|\boldsymbol{\mathcal{X}}_{tr}\|_F, \tag{18}$$

where $\boldsymbol{\mathcal{X}}_{tr}$ and $\boldsymbol{\mathcal{X}}_{est}$ are the ground truth and the reconstructed tensor.

In this experiment, we used a streaming tensor $\boldsymbol{\mathcal{X}}_t$ of size $100 \times 100 \times 1000$ with a rank $R = 5$, testing various settings of noisy and missing data. We considered three noise levels: $\sigma_n = 1$, $\sigma_n = 10^{-1}$, and $\sigma_n = 10^{-2}$. Additionally, we investigated two missing ratio of 40% and 80%. The factor ϵ was fixed at 10^{-3} to create a slowly time-varying environment. An abrupt change was introduced at $t = 600$ (i.e., $\epsilon = 1$) to evaluate how quickly OTD can converge in nonstationary conditions. The forgetting factor of OTD was set to 0.5, while its regularized parameters are fixed at 10^{-2}. Fig. 2(a) illustrates the performance of OTD in relation to the noise level σ_n. At each noise level, OTD always converges to a steady-state error floor. As expected, lower values of σ_n correspond to better estimation accuracy for OTD. Fig. 2(b) demonstrates the impact of missing data on OTD's performance. We observe that OTD effectively handles missing data. When the missing ratio is not too high, its performance is comparable to that with complete observations. However, when the missing ratio is high (e.g., 80%), OTD converges more slowly, yet its error floor remains close to that achieved with full data.

B. Real-world Video Datasets

In this task, we demonstrate its effectiveness using real video datasets, comparing it to state-of-the-art tensor-based online video completion methods. These methods include the CP-based method OLSTEC [10], the Tucker-based method ATD [12], and the tensor-train-based method ATT [22].

We use four publicly video datasets, including "Walk", "Highway", "Lobby", and "Park". [2] In our experiment, we considered three case studies of the missing ratio: 10%, 50%, and 90%. To have a fair comparison, we select tensor ranks such that all methods share the similar space computational complexity. Specifically, we set the CP rank to 16, the Tucker rank to $[12, 12, 12]$, the tensor-train rank to $[6, 6]$, and the triple rank to 6. The algorithmic parameters for the compared methods were kept at their default settings. We used the same relative metric $\mathrm{ER}(\boldsymbol{\mathcal{X}}_{tr}, \boldsymbol{\mathcal{X}}_{est})$ in (18) to measure the effectiveness of the algorithms. The performance of the algorithms is illustrated graphically in Fig. 3 and statistically summarized in Tab. 1. The results demonstrate that our method achieves competitive performance compared to other tensor-based approaches for online video completion, particularly under high missing ratios.

[2] http://jacarini.dinf.usherbrooke.ca/dataset2014

VI. Conclusions

In this paper, we developed an online triple tensor decomposition method for factorizing streaming tensors over time, called OTD. Experiments indicate that our method outperforms other state-of-the-art tensor-based techniques for the task of online video completion with real datasets. The convergence of OTD is theoretically guaranteed in stationary environments under mild conditions. Future works will explore its convergence behavior in nonstationary environments and develop a variant capable of tracking the triple rank over time.

References

[1] L. Qi, Y. Chen, M. Bakshi, and X. Zhang, "Triple decomposition and tensor recovery of third order tensors," *SIAM J. Matrix Anal. Appl.*, vol. 42, no. 1, pp. 299–329, 2021.

[2] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

[3] F. Wu, C. Li, and Y. Li, "Manifold regularization nonnegative triple decomposition of tensor sets for image compression and representation," *J. Optim. Theory Appl.*, vol. 192, no. 3, pp. 979–1000, 2022.

[4] Z. Ming, Z. Qin, L. Zhang, Y. Xu, and L. Qi, "Network traffic recovery from link-load measurements using tensor triple decomposition strategy for third-order traffic tensors," *J. Comput. Appl. Math.*, vol. 447, p. 115901, 2024.

TABLE I: Averaged relative error of adaptive tensor decompositions on incomplete video sequences.

Dataset	Size	Missing	Adaptive Tensor-based Methods Methods for Video Completion							
			OLSTEC		ATD		ATT		OTD	
			ER	Time(s)	ER	Time(s)	ER	Time(s)	ER	Time(s)
Walk	$240 \times 352 \times 1200$	10%	0.0825	153.9	0.1102	50.86	0.1156	47.93	0.1021	51.72
		50%	0.2181	92.67	0.1638	48.90	0.2561	46.87	0.1353	49.24
		90%	0.2704	56.78	0.2723	40.62	0.2694	45.26	0.1735	44.31
Highway	$320 \times 240 \times 1700$	10%	0.0491	192.2	0.0603	68.91	0.1769	63.02	0.0714	69.17
		50%	0.1129	141.9	0.1478	64.16	0.1921	61.86	0.1105	64.78
		90%	0.1652	81.73	0.2418	55.34	0.2217	47.98	0.1961	50.21
Lobby	$128 \times 160 \times 1546$	10%	0.0312	53.88	0.0466	24.60	0.1103	15.54	0.0191	20.21
		50%	0.0798	44.92	0.1091	18.26	0.1208	14.52	0.0416	17.78
		90%	0.1592	27.28	0.2420	16.34	0.2326	14.48	0.1616	16.87
Park	$228 \times 352 \times 600$	10%	0.0269	105.3	0.0622	28.32	0.0945	34.83	0.0411	30.13
		50%	0.0527	68.02	0.1029	23.06	0.1048	31.54	0.0883	26.42
		90%	0.1031	38.24	0.1562	20.51	0.1337	27.20	0.1088	21.98

[5] Q. Liao, Q. Liu, and F. A. Razak, "Hypergraph regularized nonnegative triple decomposition for multiway data analysis," *Sci. Rep.*, vol. 14, no. 1, p. 9098, 2024.

[6] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "A contemporary and comprehensive survey on streaming tensor decomposition," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 11, pp. 10 897–10 921, 2023.

[7] S. Zhou, N. X. Vinh, J. Bailey, Y. Jia, and I. Davidson, "Accelerating online CP decompositions for higher order tensors," in *Proc. ACM KDD*, 2016, pp. 1375–1384.

[8] H. Kasai and B. Mishra, "Low-rank tensor completion: A Riemannian manifold preconditioning approach," in *Proc. ICML*, 2016, pp. 1012–1021.

[9] C. Zeng and M. K. Ng, "Incremental CP tensor decomposition by alternating minimization method," *SIAM J. Matrix Anal. Appl.*, vol. 42, no. 2, pp. 832–858, 2021.

[10] H. Kasai, "Fast online low-rank tensor subspace tracking by CP decomposition using recursive least squares from incomplete observations," *Neurocomput.*, vol. 347, pp. 177–190, 2019.

[11] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "A fast randomized adaptive CP decomposition for streaming tensors," in *Proc. IEEE ICASSP*, 2021, pp. 2910–2914.

[12] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "Tracking online low-rank approximations of higher-order incomplete streaming tensors," *Patterns*, vol. 4, no. 6, p. 100759, 2023.

[13] Y. Du, Y. Zheng, K.-c. Lee, and S. Zhe, "Probabilistic streaming tensor decomposition," in *Proc. IEEE ICDM*, 2018, pp. 99–108.

[14] Z. Pan, Z. Wang, and S. Zhe, "Streaming nonlinear Bayesian tensor decomposition," in *Proc. UAI*, 2020, pp. 490–499.

[15] S. Fang, A. Narayan, R. Kirby, and S. Zhe, "Bayesian continuous-time Tucker decomposition," in *Proc. ICML*, 2022, pp. 6235–6245.

[16] E. Gujral and E. E. Papalexakis, "OnlineBTD: Streaming algorithms to track the block term decomposition of large tensors," in *Proc. IEEE DSAA*, 2020, pp. 168–177.

[17] L. T. Thanh, K. Abed-Meraim, R. Philippe, and B. Olivier, "A novel tensor tracking algorithm for block-term decomposition of streaming tensors," in *Proc. IEEE SSP*, 2023, pp. 571–575.

[18] A. A. Rontogiannis, E. Kofidis, and P. V. Giampouras, "Online rank-revealing block-term tensor decomposition," *Signal Process.*, vol. 212, p. 109126, 2023.

[19] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and R. Boyer, "Adaptive algorithms for tracking tensor-train decomposition of streaming tensors," in *Proc. EUSIPCO*, 2020, pp. 995–999.

[20] L. T. Thanh, K. Abed-Meraim, N. Linh Trung, and A. Hafiane, "Robust tensor tracking with missing data under tensor-train format," in *Proc. EUSIPCO*, 2022, pp. 832–836.

[21] D. Kressner, B. Vandereycken, and R. Voorhaar, "Streaming tensor train approximation," *SIAM J. Sci. Comput.*, vol. 45, no. 5, pp. A2610–A2631, 2023.

[22] T. T. Le, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "A novel recursive least-squares adaptive method for streaming tensor-train decomposition with incomplete observations," *Signal Process.*, vol. 216, p. 109297, 2024.

[23] Z. Huang, Y. Qiu, J. Yu, and G. Zhou, "Multi-aspect streaming tensor ring completion for dynamic incremental data," *IEEE Signal Process. Lett.*, vol. 29, pp. 2657–2661, 2022.

[24] J. Yu, T. Zou, and G. Zhou, "Online subspace learning and imputation by tensor-ring decomposition," *Neural Netw.*, vol. 153, pp. 314–324, 2022.

[25] Y. Yu and H. Li, "Tracking tensor ring decompositions of streaming tensors," *Comput. Appl. Math.*, vol. 44, no. 1, pp. 1–30.

[26] K. Gilman, D. A. Tarzanagh, and L. Balzano, "Grassmannian optimization for online tensor completion and tracking with the t-SVD," *IEEE Trans. Signal Process.*, vol. 70, pp. 2152 – 2167, 2022.

[27] M. Vandecappelle and L. D. Lathauwer, "Updating the multilinear UTV decomposition," *IEEE Trans. Signal Process.*, vol. 70, pp. 3551–3565, 2022.

[28] D. P. Woodruff *et al.*, "Sketching as a tool for numerical linear algebra," *Found. Trends Theor. Comput. Sci.*, vol. 10, no. 1–2, pp. 1–157, 2014.

[29] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, 1999.

[30] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "Robust tensor tracking with missing data and outliers: Novel adaptive CP decomposition and convergence analysis," *IEEE Trans. Signal Process.*, vol. 70, pp. 4305 – 4320, 2022.