Robust Tensor Tracking With Missing Data Under Tensor-Train Format

Le Trung Thanh[†,⋆], Karim Abed-Meraim[†,‡], Nguyen Linh Trung[⋆] and Adel Hafiane[†]

[†]University of Orléans, INSA-CVL, PRISME, EA 4229, 45067 Orléans, France

[⋆] VNU University of Engineering and Technology, 100000 Hanoi, Vietnam

[‡]Academic Institute of France (IUF), 75005 Paris, France

***Abstract*—Robust tensor tracking or robust adaptive tensor decomposition of streaming tensors is crucial when observations are corrupted by sparse outliers and missing data. In this paper, we introduce a novel tensor tracking algorithm for factorizing incomplete streaming tensors with sparse outliers under tensor-train (TT) format. The proposed algorithm consists of two main stages: online outlier rejection and tracking of TT-cores. In the former stage, outliers affecting the data streams are efficiently detected by an ADMM solver. In the latter stage, we propose an effective recursive least-squares solver to incrementally update TT-cores at each time t. Several numerical experiments on both simulated and real data are presented to verify the effectiveness of the proposed algorithm.**

***Index Terms*—Tensor-train decomposition, robust adaptive algorithms, streaming data, missing data, sparse outliers.**

I. Introduction

Over the last decade, data stream analysis has gained increasing attention in the signal processing and machine learning community as many modern streaming systems generate massive data streams over time [1]. There exist several inherent issues which are still challenging for mining and analysing data streams. For example, the data size is unbounded, while the underlying process that generates streaming data can be time-varying. Also, uncertainties (e.g., incomplete, noisy, and corrupted elements) can arise during data acquisition, and thus, they may lead to undesired results.

In parallel, tensor decomposition (TD) has become a powerful processing tool for analysing multidimensional data and found many applications in various areas [2], [3]. TD allows factorizing a tensor – a multiway array – into a set of basic components and factors (e.g., vectors, matrices, or "simpler" tensors). When factorizing tensors derived from data streams (aka streaming tensors), we may refer to such a decomposition as tensor tracking or adaptive (online) tensor decomposition. In addition, missing data and sparse outliers become more and more ubiquitous in streaming and online applications [4]. Therefore, it would be of great interest to develop a *robust* variant of tensor tracking, which is capable of handling data corruptions and inherent issues in streaming systems.

Corresponding author: Nguyen Linh Trung (linhtrung@vnu.edu.vn). This work has been done under the research project QG.22.62 on "Multidimensional data analysis and application to Alzheimer's disease diagnosis" of Vietnam Nation University, Hanoi.

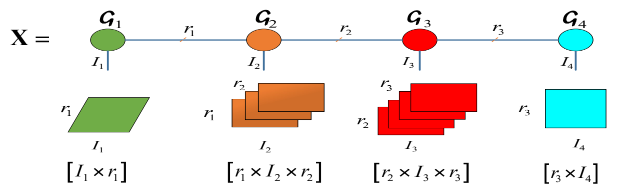

Fig. 1. TT-decomposition of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ with TT-rank $[r_1, r_2, r_3]$.

In this study, we consider the problem of robust tensor tracking in the presence of both missing data and outliers under the tensor-train (TT) format. Specifically, we can represent a N-order tensor under the TT format by a set of N tensors of 3-order via a multilinear product [5], see Fig. 1 for an illustration. Compared to the two standard CP and Tucker decompositions, TT decomposition offers several appealing features, such as: (i) we can factorize any high-order tensor under the TT format and its computation is stable; (ii) TT rank can be effectively determined, and (iii) it can break the curse of dimensionality. As a result, this decomposition has the potential to handle large-scale and high-order tensors. The readers are referred to [6] for a good review on (batch) tensor-train decomposition.

Related Works: In streaming (adaptive) settings, tensor-train decomposition has not got as much attention and popularity as CP and Tucker decompositions. Specifically, there exist only a few online tensor-train algorithms for tensor tracking in the literature so far.

Liu *et al.* in [7] introduced an incremental tensor-train algorithm for factorizing tensors whose one mode can increase with time, namely iTTD. By considering data streams as individual tensors, iTTD factorizes the incoming data into TT-cores and then concatenates them into old estimations. Wang *et al.* in [8] also proposed an incremental algorithm called AITT for streaming tensor-train decomposition. By utilizing a relation between the integration of unfolding matrices and the directly reshaped matrix, AITT can update the underlying TT-cores at a low cost. However, the optimization framework of iTTD and AITT is not an online streaming learning, but incremental batch learning. In parallel, we proposed two effective online methods called TT-FOA and ATT for adaptive tensor-train decomposition in [9], [10], respectively. Although TT-FOA and ATT can track the low-rank components of high-order tensors successfully with time, they have not been designed for handling data corruptions. It is worth noting that all the existing adaptive TT algorithms above are sensitive to either time variation, missing data, or sparse outliers.

Main Contribution: In this paper, we introduce a new tensor-train method for factorizing incomplete high-order streaming tensors possibly corrupted by sparse outliers. The proposed method is referred to as ROBOT which stands for ROBust Online Tensor-Train decomposition. ROBOT involves two well-known optimization methods: block-coordinate descent (BCD) and recursive least-squares (RLS). Thanks to the BCD framework, ROBOT decomposes the main optimization into two stages: (i) online outlier rejection and (ii) tracking of TT-cores in time. In the former stage, we apply an effective ADMM solver to estimate the last (temporal) TT-core and sparse outliers living in observations. In the latter stage, we present an efficient RLS solver to minimize an exponential weighted least-squares objective function accounting for missing entries and time variations of TT-cores. Technically, ROBOT is capable of estimating the low-rank components of the underlying tensor from imperfect streams (i.e., due to noise, outliers, and missing data) and tracking their time variation in dynamic environments. To the best of our knowledge, ROBOT is the first streaming TT decomposition robust to sparse outliers, missing data, and time variation. For reference, we summarize in Tab. I some frequently used notations in this paper.

TABLE I
NOTATIONAL CONVENTIONS

Notations	Descriptions
$x, \mathbf{x}, \mathbf{X}, \boldsymbol{\mathcal{X}}$	scalar, vector, matrix, and tensor
$[\boldsymbol{\mathcal{X}}]_{i_1 i_2 \dots i_N}$	$(i_1, i_2, \dots i_N)$-th element of $\boldsymbol{\mathcal{X}}$
$\mathbf{X}(i,:), \mathbf{X}(:,j)$	i-th row and j-th column of $\mathbf{X}$
$\mathbf{X}^{-1}$, $\mathbf{X}^{\top}$, and $\mathbf{X}^{-\top}$	inverse, transpose of $\mathbf{X}$, and transpose of $\mathbf{X}^{-1}$
$\underline{\mathbf{X}}^{(n)}$	mode-n unfolding/matricization of $\boldsymbol{\mathcal{X}}$
$\texttt{reshape}\{\boldsymbol{\mathcal{X}}, [\mathbf{a}]\}$	reshape $\boldsymbol{\mathcal{X}}$ into a new tensor of size $a_1 \times \dots \times a_N$
$\boldsymbol{\mathcal{X}} \boxplus_n \boldsymbol{\mathcal{Y}}$	concatenation of $\boldsymbol{\mathcal{X}}$ with $\boldsymbol{\mathcal{Y}}$ along the n-th dimension
$\boldsymbol{\mathcal{X}} \times_n^1 \boldsymbol{\mathcal{Y}}$	mode-$(n,1)$ contracted product of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$
$\otimes$, $\circledast$	Kronecker and Hadamard product
$\|.\|_F$ and $\|.\|_p$	Frobenius norm and ℓ_p norm ($p = 1,2$)

II. Problem Formulation

In this paper, we study the robust adaptive tensor-train decomposition of a N-order streaming tensor $\boldsymbol{\mathcal{X}}_t$ in the presence of both sparse outliers and missing data. Without loss of generality, we suppose the last dimension of $\boldsymbol{\mathcal{X}}_t$ is temporal, while the others remain constant with time, i.e., $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_{N-1} \times I_N^t}$. Specifically, at time t, $\boldsymbol{\mathcal{X}}_t$ is obtained by concatenating the incoming data stream $\boldsymbol{\mathcal{Y}}_t \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_{N-1} \times W}$ (with $W \geq 1$) to the old observation $\boldsymbol{\mathcal{X}}_{t-1}$ along the temporal dimension I_N^t, i.e.,

$$\boldsymbol{\mathcal{X}}_t = \boldsymbol{\mathcal{X}}_{t-1} \boxplus_N \boldsymbol{\mathcal{Y}}_t \quad \text{and} \quad I_N^t = I_N^{t-1} + W. \tag{1}$$

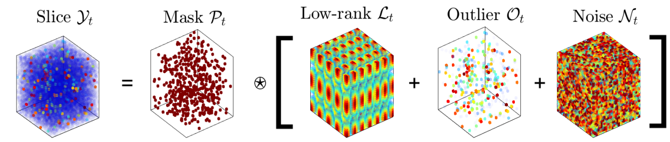

Fig. 2. Temporal slice $\boldsymbol{\mathcal{Y}}_t$ with missing data and outliers.

The temporal slice $\boldsymbol{\mathcal{Y}}_t$ is supposed to have the form

$$\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \big(\boldsymbol{\mathcal{L}}_t + \boldsymbol{\mathcal{O}}_t + \boldsymbol{\mathcal{N}}_t\big), \tag{2}$$

see Fig. 2 for an illustration. Particularly, $\boldsymbol{\mathcal{P}}_t$ is a binary mask tensor, $\boldsymbol{\mathcal{O}}_t$ is a sparse outlier tensor, $\boldsymbol{\mathcal{N}}_t$ is a Gaussian noise tensor, and they share the same size as $\boldsymbol{\mathcal{Y}}_t$. The low-rank component $\boldsymbol{\mathcal{L}}_t$ of $\boldsymbol{\mathcal{Y}}_t$ is expressed as

$$\boldsymbol{\mathcal{L}}_t = \boldsymbol{\mathcal{G}}_t^{(1)} \times_2^1 \boldsymbol{\mathcal{G}}_t^{(2)} \times_3^1 \cdots \times_N^1 \mathbf{G}_t^{(N)}, \tag{3}$$

where $\boldsymbol{\mathcal{G}}_t^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$ for $n = 1, 2, \dots, N$ with $r_0 = r_N = 1$ is the n-th TT-core; $[r_1, r_2, \dots, r_{N-1}]$ is called TT-rank; and $\mathbf{G}_t^{(N)} \in \mathbb{R}^{r_{N-1} \times W}$ contains the last W columns of $\boldsymbol{\mathcal{G}}_t^{(N)}$.

In online settings, we propose to minimize the following objective function:

$$\underset{\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^N, \boldsymbol{\mathcal{O}}}{\operatorname{argmin}} \sum_{k=1}^{t} \beta^{t-k} \bigg(\Big\| \boldsymbol{\mathcal{P}}_k \circledast \Big(\boldsymbol{\mathcal{G}}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \boldsymbol{\mathcal{G}}^{(N-1)} \times_N^1 \mathbf{G}_k^{(N)} + \boldsymbol{\mathcal{O}}_k - \boldsymbol{\mathcal{Y}}_k \Big) \Big\|_F^2 + \rho_1 \big\| \boldsymbol{\mathcal{O}}_k \big\|_1 \bigg) + \rho_2 \sum_{n=1}^{N-1} \Big\| \boldsymbol{\mathcal{G}}^{(n)} - \boldsymbol{\mathcal{G}}_{t-1}^{(n)} \Big\|_F^2. \tag{4}$$

Here, $\beta \in (0,1]$ plays the role of a forgetting factor in adaptive filter theory which aims to reduce the impact of distant observations as well as deal with nonstationary environments [11]. The ℓ_1-norm enforces the sparsity on $\boldsymbol{\mathcal{O}}$ (the outliers), while the last regularization term of (4) is to control the time variation of TT-cores between two consecutive instances. In addition, we make two mild assumptions on the data model to support our algorithm development in Section III: TT-cores $\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N-1}$ may either be static or vary slowly with time, i.e., $\boldsymbol{\mathcal{G}}_t^{(n)} \simeq \boldsymbol{\mathcal{G}}_{t-1}^{(n)}$; and the TT-rank is supposed to be known.

III. Proposed Method

In this section, we propose an adaptive method called ROBOT (which stands for ROBust Online Tensor-Train) for factorizing tensors derived from data streams in the presence of sparse outliers and missing data. Particularly, we decompose the main problem (4) into two stages:

- **Stage 1**: update $\boldsymbol{\mathcal{G}}_t^{(N)}$ and $\boldsymbol{\mathcal{O}}_t$ given $\big\{\boldsymbol{\mathcal{G}}_{t-1}^{(n)}\big\}_{n=1}^{N-1}$;
- **Stage 2**: estimate $\boldsymbol{\mathcal{G}}_t^{(n)}$ given $\boldsymbol{\mathcal{G}}_t^{(N)}$, $\boldsymbol{\mathcal{O}}_t$, and the remaining TT-cores, for $n = 1, 2, \dots, N-1$.

A. Estimation of the last TT-core $\boldsymbol{\mathcal{G}}_t^{(N)}$ and Outlier $\boldsymbol{\mathcal{O}}_t$

At each time t, we estimate $\mathbf{G}_t^{(N)}$ and $\boldsymbol{\mathcal{O}}_t$ by solving

$$\Big\{\mathbf{G}_t^{(N)}, \boldsymbol{\mathcal{O}}_t\Big\} = \underset{\mathbf{G}^{(N)}, \boldsymbol{\mathcal{O}}}{\operatorname{argmin}} \Big[\Big\| \boldsymbol{\mathcal{P}}_t \circledast \Big(\boldsymbol{\mathcal{H}}_{t-1} \times_N^1 \mathbf{G}^{(N)} + \boldsymbol{\mathcal{O}} - \boldsymbol{\mathcal{Y}}_t \Big) \Big\|_F^2 + \rho_1 \big\| \boldsymbol{\mathcal{O}} \big\|_1 + \rho_2 \big\| \mathbf{G}^{(N)} \big\|_F^2 \Big], \tag{5}$$

where $\mathcal{H}_{t-1} = \mathcal{G}_{t-1}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \mathcal{G}_{t-1}^{(N-1)}$ and the term $\rho_2\|\mathbf{G}^{(N)}\|_F^2$ is to mitigate ill matrix conditions. Interestingly, we exploit the fact that (5) can be decomposed into W sub-problems w.r.t. W columns of $\mathbf{G}_t^{(N)}$, as follows:

$$\underset{\mathbf{g}_i,\mathbf{o}_i}{\operatorname{argmin}} \Big\|\underline{\mathbf{P}}_{t,i}\Big(\mathbf{H}_{t-1}\mathbf{g}_i + \mathbf{o}_i - \mathbf{y}_{t,i}\Big)\Big\|_2^2 + \rho_1\big\|\mathbf{o}_i\big\|_1 + \rho_2\big\|\mathbf{g}_i\big\|_2^2. \quad (6)$$

Here, $\mathbf{g}_i, \mathbf{o}_i$, and $\mathbf{y}_{t,i}$ are, respectively, the i-th column of $\mathbf{G}^{(N)}$, the two unfolding matrices of $\mathcal{O}$ and $\mathcal{Y}_t$; the mask $\underline{\mathbf{P}}_{t,i} = \operatorname{diag}\{\underline{\mathbf{P}}_t^{(N)}(i,:)\}$; while the matrix $\mathbf{H}_{t-1} \in \mathbb{R}^{I_1 \ldots I_{N-1} \times r_{N-1}}$ is a matricization of $\mathcal{H}_{t-1}$.

Since both ℓ_1-norm and ℓ_2-norm are convex, (6) can be effectively minimized by several methods, e.g., block coordinate descent (BCD) [12] and alternating direction method of multipliers (ADMM) [13]. In this work, we adopt the ADMM solver introduced in our companion work on subspace tracking [14]. Due to the presence of the regularization term $\rho_2\|\mathbf{g}_i\|_2^2$, the update rule at the j-th iteration of the ADMM solver in [14] is specifically modified as follows

$$\begin{aligned}
\mathbf{g}^j &= \Big(\mathbf{H}_{t-1}^\top \underline{\mathbf{P}}_{t,i}\mathbf{H}_{t-1} + \rho_2\mathbf{I}_{r_{N-1}}\Big)^{-1}\mathbf{H}_{t-1}^\top\underline{\mathbf{P}}_{t,i}\Big(\mathbf{y}_{t,i} - \mathbf{o}^{j-1} + \mathbf{e}^{j-1}\Big),\\
\mathbf{z}^j &= \underline{\mathbf{P}}_{t,i}\Big(\mathbf{H}_{t-1}\mathbf{g}^j + \mathbf{s}^{j-1} - \mathbf{y}_{t,i}\Big),\\
\mathbf{e}^j &= \frac{\lambda_1}{1+\lambda_1}\mathbf{z}^j + \frac{1}{1+\lambda_1}\mathcal{S}_{1+\frac{1}{\lambda_1}}\big(\mathbf{z}^j\big),\\
\mathbf{u}^j &= \frac{1}{1+\lambda_2}\Big(\underline{\mathbf{P}}_{t,i}\big(\mathbf{y}_{t,i} - \mathbf{H}_{t-1}\mathbf{g}^j\big)\Big) - \lambda_2(\mathbf{o}^{j-1} - \mathbf{r}^{j-1}),\\
\mathbf{o}^j &= \mathcal{S}_{\rho_2/\lambda_2}\big(\mathbf{u}^j + \mathbf{r}^{j-1}\big),\\
\mathbf{r}^j &= \mathbf{r}^{j-1} + \mathbf{u}^j - \mathbf{s}^j.
\end{aligned}$$

Here, $\{\mathbf{z}^j, \mathbf{e}^j, \mathbf{u}^j, \mathbf{r}^j\}$ are auxiliary variables aiming to accelerate the update initialized as zeros; the augmented Lagrangian parameters λ_1 and λ_2 can be chosen in the range $[1, 1.8]$; and $\mathcal{S}_\alpha(.)$ is the soft-thresholding operator defined as $\mathcal{S}_\alpha(x) = \max(0, x-\alpha) - \max(0, -x-\alpha)$. We refer the readers to [14] for further details. Note that since (6) is a biconvex minimization problem, and thus, we can apply any other existing proved algorithm to obtain its optimal solution [15].

The temporal TT-core $\mathcal{G}_t^{(N)}$ is simply obtained by $\mathcal{G}_t^{(N)} = [\mathcal{G}_{t-1}^{(N)} \;\; \mathbf{G}_t^{(N)}]$. In addition, we can re-update $\mathbf{G}_t^{(N)}$ in the same way as above when others TT-cores $\{\mathcal{G}_t^{(n)}\}_{n=1}^{N-1}$ are updated. Furthermore, after obtaining the outlier $\mathcal{O}_t$, we can accelerate the tracking ability of ROBOT by re-updating the observation mask $\mathcal{P}_t$ as follows

$$\big[\widetilde{\mathcal{P}}_t\big]_{i_1 i_2 \ldots i_N} = \begin{cases} 0, & \text{if } \big[\mathcal{O}_t\big]_{i_1 i_2 \ldots i_N} \neq 0, \\ \big[\mathcal{P}_t\big]_{i_1 i_2 \ldots i_N}, & \text{otherwise.} \end{cases} \quad (7)$$

It is motivated by the following observation: In the literature of robust subspace tracking (RST), the outlier rejection step can facilitate the tracking ability of RST estimators because only "clean" data are involved in the tracking process [14]. Our stage 2 for tracking the TT-cores can be viewed as an extended version of RST for high-order streaming tensors, so the outlier rejection mechanism of (7) can improve its performance.

B. Estimation of TT-cores $\{\mathcal{G}_t^{(n)}\}_{n=1}^{N-1}$

We estimate $\{\mathcal{G}^{(n)}\}_{n=1}^{N-1}$ by minimizing

$$\mathcal{G}_t^{(n)} = \underset{\mathcal{G}^{(n)}}{\operatorname{argmin}} \Bigg[\sum_{k=1}^t \beta^{t-k}\Big\|\widetilde{\mathcal{P}}_k \circledast \Big(\mathcal{A}_{t-1}^{(n)} \times_n^1 \mathcal{G}^{(n)} \times_{n+1}^1 \mathcal{B}_k^{(n)} - \mathcal{Y}_k\Big)\Big\|_F^2 + \rho_2\Big\|\mathcal{G}^{(n)} - \mathcal{G}_{t-1}^{(n)}\Big\|_F^2\Bigg], \quad (8)$$

where $\mathcal{A}_{t-1}^{(n)} = \mathcal{G}_{t-1}^{(1)} \times_2^1 \cdots \times_{n-1}^1 \mathcal{G}_{t-1}^{(n-1)}$ and $\mathcal{B}_k^{(n)} = \mathcal{G}_{t-1}^{(n+1)} \times_{n+2}^1 \cdots \times_{N-1}^1 \mathcal{G}_{t-1}^{(N-1)} \times_N^1 \mathbf{G}_k^{(N)}$ while the term $\mathcal{O}_k$ is discarded due to outlier rejection mechanism (7), i.e., $\widetilde{\mathcal{P}}_t \circledast (\mathcal{Y}_t - \mathcal{O}_t) = \widetilde{\mathcal{P}}_t \circledast \mathcal{Y}_t$. Particularly, (8) can be regarded as the optimization problem of adaptive TT decomposition from incomplete observations $\{\mathcal{Y}_k\}_{k=1}^t$ with new binary masks $\{\widetilde{\mathcal{P}}_k\}_{k=1}^t$. Accordingly, we can apply the effective recursive least-squares (RLS) method as proposed in our work [10] for minimizing (8). For the sake of completeness, we describe here the main steps of the RLS solver and refer the readers to [10] for further details.

For a better interpretation, we first recast (8) as

$$\mathbf{G}_t^{(n)} = \underset{\mathbf{G}^{(n)}}{\operatorname{argmin}} \Bigg[\sum_{m=1}^{I_n}\Bigg(\sum_{k=1}^t \beta^{t-k}\Big\|\overline{\mathbf{P}}_{k,m}^{(n)}\Big(\mathbf{g}_m^{(n)}(\mathbf{B}_k^{(n)} \otimes \mathbf{A}_{t-1}^{(n)}) - \mathbf{y}_{k,m}^{(n)}\Big)\Big\|_2^2 + \rho_2\Big\|\mathbf{g}_m^{(n)} - \mathbf{g}_{t-1,m}^{(n)}\Big\|_2^2\Bigg)\Bigg], \quad (9)$$

where $\mathbf{g}_m^{(n)}$ is the m-th row of $\mathbf{G}^{(n)} \in \mathbb{R}^{I_n \times r_{n-1}r_n}$ which is the transpose of the mode-2 unfolding matrix of $\mathcal{G}^{(n)}$, $\overline{\mathbf{P}}_{k,m} = \operatorname{diag}\{\underline{\tilde{\mathbf{P}}}_k^{(n)}(m,:)\}$, $\mathbf{A}_{t-1}^{(n)} = \operatorname{reshape}\{\mathcal{A}_{t-1}^{(n)}, [r_{n-1}, I_1 I_2 \ldots I_{n-1}]\}$, and $\mathbf{B}_k^{(n)} = \operatorname{reshape}\{\mathcal{B}_t^{(n)}, [r_n, I_{n+1}I_{n+2}\ldots I_{N-1}]\}$.

Let us denote $\mathbf{W}_k^{(n)} = \mathbf{B}_k^{(n)} \otimes \mathbf{A}_{t-1}^{(n)}$, $\mathbf{S}_{k,m}^{(n)} = \sum_{k=1}^t \beta^{t-k}\mathbf{W}_t^{(n)}\overline{\mathbf{P}}_{t,m}^{(n)}\big(\mathbf{W}_t^{(n)}\big)^\top$, and $\mathbf{d}_{t,m}^{(n)} = \sum_{k=1}^t \beta^{t-k}\mathbf{W}_k^{(n)}\overline{\mathbf{P}}_{k,m}^{(n)}\big(\mathbf{y}_{k,m}^{(n)}\big)^\top$. At time t, we then have

$$\mathbf{S}_{t,m}^{(n)} = \beta\mathbf{S}_{t-1,m}^{(n)} + \mathbf{W}_t^{(n)}\overline{\mathbf{P}}_{t,m}^{(n)}\big(\mathbf{W}_t^{(n)}\big)^\top \quad (10)$$

$$\mathbf{d}_{t,m}^{(n)}, = \beta\mathbf{d}_{t-1,m}^{(n)} + \mathbf{W}_t^{(n)}\overline{\mathbf{P}}_{t,m}^{(n)}\big(\mathbf{y}_{t,m}^{(n)}\big)^\top. \quad (11)$$

Setting the gradient of (9) to zero results in:

$$\sum_{m=1}^{I_n}\big(\mathbf{S}_{t,m}^{(n)} + \rho_2\mathbf{I}_{r_{n-1}r_n}\big)\big(\mathbf{g}_m^{(n)}\big)^\top = \sum_{m=1}^{I_n}\Big(\mathbf{d}_{t,m}^{(n)} + \rho_2\big(\mathbf{g}_{t-1,m}^{(n)}\big)^\top\Big). \quad (12)$$

Therefore, we can express each row $\mathbf{g}_{t,m}^{(n)}$ of $\mathbf{G}_t^{(n)}$ separately as

$$\big(\mathbf{S}_{t,m}^{(n)} + \rho_2\mathbf{I}_{r_{n-1}r_n}\big)\big(\mathbf{g}_{t,m}^{(n)}\big)^\top = \mathbf{d}_{t,m}^{(n)} + \rho_2\big(\mathbf{g}_{t-1,m}^{(n)}\big)^\top. \quad (13)$$

Thanks to (10) and (11), we further recast (13) as

$$\mathbf{g}_{t,m}^{(n)} = \mathbf{g}_{t-1,m}^{(n)} + \Big(\boldsymbol{\delta}\mathbf{y}_{t,m}^{(n)}\overline{\mathbf{P}}_{t,m}^{(n)}\big(\mathbf{W}_t^{(n)}\big)^\top + \beta\rho_2\boldsymbol{\delta}\mathbf{g}_{t-1,m}^{(n)}\Big) \times \times\Big(\mathbf{S}_{t,m}^{(n)} + \rho_2\mathbf{I}_{r_{n-1}r_n}\Big)^{-\top}, \quad (14)$$

where $\boldsymbol{\delta}\mathbf{y}_{t,m}^{(n)} = \overline{\mathbf{P}}_{t,m}^{(n)}\big(\mathbf{y}_{t,m}^{(n)} - \mathbf{g}_{t-1,m}^{(n)}\mathbf{W}_t^{(n)}\big)^\top$ and $\boldsymbol{\delta}\mathbf{g}_{t-1,m}^{(n)} = \mathbf{g}_{t-1,m}^{(n)} - \mathbf{g}_{t-2,m}^{(n)}$. Collecting all rows $\mathbf{g}_{t,m}^{(n)}$ together (for $m = 1, 2, \ldots, I_n$), we obtain a simpler recursive rule as

$$\mathbf{G}_t^{(n)} = \mathbf{G}_{t-1}^{(n)} + \Big(\big(\underline{\mathbf{P}}_t^{(n)} \circledast \boldsymbol{\Delta}\underline{\mathbf{Y}}_t^{(n)}\big)\big(\mathbf{W}_t^{(n)}\big)^\top + \beta\rho_2\boldsymbol{\Delta}\mathbf{G}_{t-1}^{(n)}\Big) \times \times\Big(\mathbf{S}_t^{(n)} + \rho_2\mathbf{I}_{r_{n-1}r_n}\Big)^{-\top}, \quad (15)$$

where $\Delta\underline{\mathbf{Y}}_{t,m}^{(n)} = \underline{\mathbf{Y}}_t^{(n)} - \mathbf{G}_{t-1}^{(n)}\mathbf{W}_t^{(n)}$ and $\Delta\mathbf{G}_{t-1}^{(n)} = \mathbf{G}_{t-1}^{(n)} - \mathbf{G}_{t-2}^{(n)}$, and $\mathbf{S}_t^{(n)} = \beta\mathbf{S}_{t-1}^{(n)} + \mathbf{W}_t^{(n)}\big(\mathbf{W}_t^{(n)}\big)^\top$. To enable the recursive update (15), we set $\Delta\mathbf{G}_0^{(n)} = \mathbf{0}$ and $\mathbf{S}_0^{(n)} = \delta^{(n)}\mathbf{I}_{r_{n-1}r_n}$ with $\delta^{(n)} > 0$.

C. Computational Complexity and Memory Storage

For short, we suppose $I_n = I$ and $r_n = r$ for all $n = 1,\ldots,N-1$. In Stage 1, ROBOT requires a cost of $\mathcal{O}(W|\Omega_t|r^2)$ flops for estimating $\mathbf{G}_t^{(N)}$ and $\mathcal{O}_t$ where $|\Omega_t|$ denotes the number of observed data in $\mathcal{Y}_t$. In Stage 2, ROBOT needs a cost of $\mathcal{O}\big((N-1)I^{N-1}r^4\big)$ flops for tracking $N-1$ TT-cores $\{\mathcal{G}_t^{(n)}\}_{n=1}^{N-1}$. Therefore, the overall complexity of ROBOT is $\mathcal{O}\big(r^2\max\big\{(N-1)I^{N-1}r^2, W|\Omega_t|\big\}\big)$ flops. With respect to memory storage, ROBOT requires $\mathcal{O}\big((N-1)(2Ir^2+r^4)\big)$ for storing $\{\mathcal{G}_t^{(n)}\}_{n=1}^{N-1}$, $\{\Delta\mathcal{G}_t^{(n)}\}_{n=1}^{N-1}$, and $\{\mathbf{S}_t^{(n)}\}_{n=1}^{N-1}$.

IV. Simulations

In this section, we evaluate the performance of ROBOT in terms of the following aspects: (i) impact of noise, (ii) its tracking ability in nonstationary environments, (iii) impact of missing observations, (iv) impact of outliers, and (v) its use for the problem of video background and foreground separation.

Experiment Setup: We follow the problem formulation in Section II to simulate temporal slices $\{\mathcal{Y}_t\}_{t\geq 1}$. In particular, $\mathcal{Y}_t$ is randomly generated under the model

$$\mathcal{Y}_t = \mathcal{P}_t \circledast \Big(\mathcal{L}_t + \mathcal{O}_t + \mathcal{N}_t\Big), \tag{16}$$

$$\text{where } \mathcal{L}_t = \mathcal{G}_t^{(1)} \times_2^1 \mathcal{G}_t^{(2)} \times_3^1 \mathcal{G}_t^{(3)} \times_4^1 \mathbf{g}_t^{(4)}. \tag{17}$$

Here, $\mathcal{P}_t \in \mathbb{R}^{I_1\times I_2\times I_3\times 1}$ is a binary mask tensor whose entries are obtained by a Bernoulli model with probability $1-\omega_{miss}$ (i.e., ω_{miss} represents the missing density).[1] $\mathcal{N}_t$ is a Gaussian noise tensor whose entries are i.i.d. from $\mathcal{N}(0,\sigma_n^2)$. $\mathcal{O}_t$ is a sparse tensor containing outliers whose amplitude is uniformly chosen in the interval $[0, \texttt{fac-outlier}]$ while their indices (locations) follow another Bernoulli model with probability $\omega_{outlier}$. $\mathcal{L}_t$ is the low-rank component of $\mathcal{Y}_t$ in which $\mathbf{g}_t^{(4)} \in \mathbb{R}^{r_3\times 1}$ is a standard normal random vector. At time t, TT-cores are varied under the model $\mathcal{G}_t^{(n)} = \mathcal{G}_{t-1}^{(n)} + \varepsilon\mathcal{V}_t^{(n)}$, where ε denotes the time-varying factor, $\mathcal{V}_t^{(n)}$ shares the same size as $\mathcal{G}_t^{(n)} \in \mathbb{R}^{r_{n-1}\times I_n\times r_n}$ and its entries are derived from $\mathcal{N}(0,1)$. At $t=0$, $\mathcal{G}_0^{(n)}$ is initialized by a Gaussian distribution with zero mean and unit variance.

To evaluate the performance of ROBOT, we use the following relative error:

$$\text{RE}\left(\mathcal{X}_{tr},\mathcal{X}_{es}\right) = \left\|\mathcal{X}_{tr} - \mathcal{X}_{es}\right\|_F / \left\|\mathcal{X}_{tr}\right\|_F, \tag{18}$$

where $\mathcal{X}_{tr}$ (resp. $\mathcal{X}_{es}$) refers to the true low-rank component (resp. estimation).

[1] Values in the tensor slices are here supposed to be missing completely at random. ROBOT still has the capability to deal with other forms of missingness (e.g., missing at random (MAR) and missing not at random (MNAR). Due to the space limitation, we omit experiments to demonstrate the performance of ROBOT in such settings.

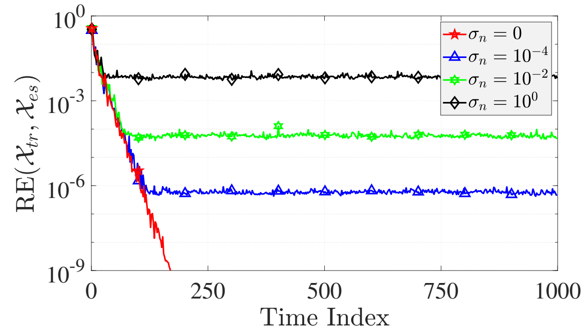

Fig. 3. Effect of the noise level σ_n on the performance of ROBOT.

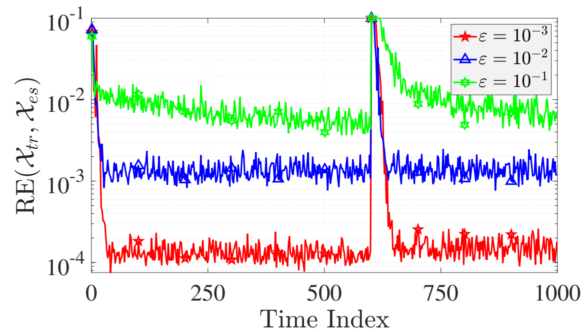

Fig. 4. Effect of the varying factor ε on the performance of ROBOT.

*1) **Effect of the noise level** σ_n:* We change the value of σ_n and measure the estimation accuracy of ROBOT. We used a streaming tensor of size $10\times 15\times 20\times 1000$ and rank $\mathbf{r}_{\text{TT}} = [5,5,5]$. Parameters of the data model were set as: time-varying factor $\varepsilon = 0$, missing density $\omega_{miss} = 0\%$, and outlier density $\omega_{outlier} = 0\%$ (i.e. outliers free observations). We fixed algorithmic parameters of ROBOT as follows: the forgetting factor $\beta = 0.5$ and two penalty parameters $\rho_1 = \rho_2 = 1$. The result is shown in Fig. 3. Clearly, the value of σ_n does not affect ROBOT's convergence rate but its relative error.

*2) **Effect of the time-varying factor** ε:* Next, we evaluate the performance of ROBOT in dynamic and nonstationary environments. We reused the streaming tensor above with 90% observations (i.e., $\omega_{miss} = 10\%$). The noise level σ_n was fixed at 10^{-3}. We set the outlier density and intensity to 10% and 1, respectively. The forgetting factor and two penalty parameters were kept as above. Also, an abrupt change was made at $t = 600$ to assess how fast ROBOT converges. Fig. 4 illustrates the effect of ε on the tracking ability of ROBOT. We can see that the performance of ROBOT increases when ε decreases and converges towards a steady-state error.

*3) **Effect of the missing density** ω_{miss}:* We then investigate the tracking ability of ROBOT in the presence of missing data. The value of ω_{miss} was chosen among $\{10\%, 50\%, 90\%\}$. We kept all experimental parameters as above, except the time-varying factor ε which was set to 10^{-3}. We can see from Fig. 5 that both convergence rate and estimation accuracy of ROBOT are affected by the value of ω_{miss}. The lower $\omega_{\texttt{miss}}$ is, the better performance ROBOT achieves.

*4) **Effect of outliers**:* Here, we measure the robustness of ROBOT against sparse outliers. Most of experimental parame-

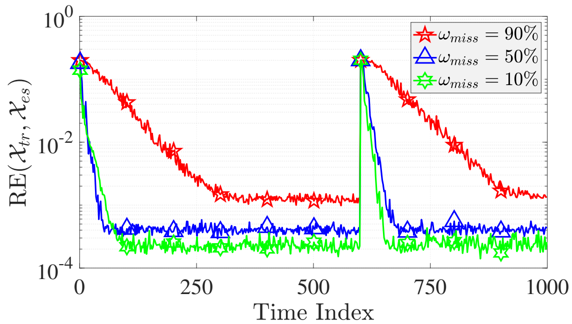

Fig. 5. Effect of the missing density ω_{miss} on the tracking ability of ROBOT.

ters were kept as in the previous tasks: $\omega_{miss} = 10\%$, $\beta = 0.5$, $\sigma_n = \varepsilon = 10^{-3}$, and $\rho_1 = \rho_2 = 1$. We investigated the case when 30% entries were corrupted by outliers. Three levels of the outlier intensity `fac-outlier` were considered, including 0.1, 1, and 10 (resp. low, moderate, and strong effect). Fig. 6 indicates that ROBOT is capable of tensor tracking from incomplete observations corrupted by sparse outliers.

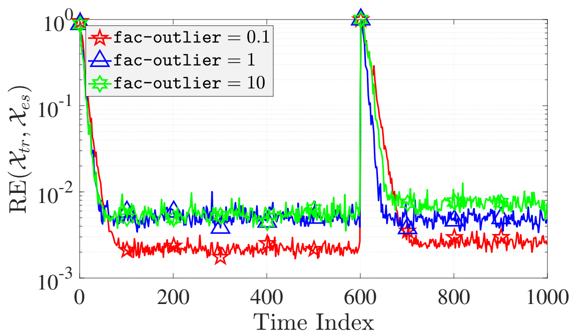

Fig. 6. Effect of the outliers on the tracking ability of ROBOT.

*5) **Video background/foreground separation:*** In this task,[2] we used three video datasets, including "Lobby", "Highway", and "Hall". The dataset "Lobby" includes 1700 frames of size 144×176. There are 1700 frames of size 240×320 in the data "Highway", while "Hall" consists of 3584 frames whose size is 174×144. The performance of ROBOT was evaluated in comparison with two online background/foreground separation algorithms, including PETRELS-ADMM [14] and GRASTA [16]. The subspace rank and TT-rank were set to 10 and $[10, 10]$, respectively. The result from Fig. 7 indicates that ROBOT is able to detect moving objects in real surveillance video sequences with reasonable performance.

V. Conclusions

In this paper, we have considered the problem of streaming tensor-train decomposition in the presence of both sparse outliers and missing data. A robust adaptive tensor-train algorithm has been introduced, namely ROBOT. The proposed algorithm is fully capable of tracking the underlying low-rank component of incomplete streaming tensors corrupted by sparse outliers even in nonstationary environments. The use of ROBOT for real data was illustrated with the problem of video background and foreground separation.

[2]Here, the foreground plays the role of outliers and its separation from the background is based on the proposed detection procedure.

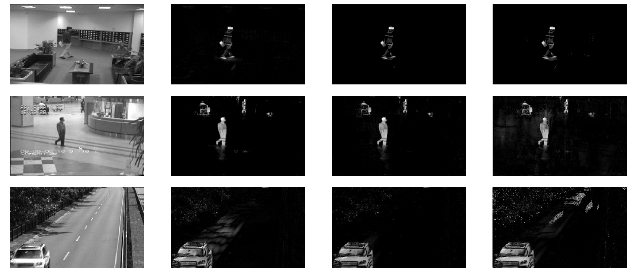

Fig. 7. Background and foreground separation. From bottom to top row: Highway, Hall, and Lobby. From left to right column: Original video frame, PETRELS-ADMM, GRASTA, and ROBOT.

References

[1] T. Kolajo, O. Daramola, and A. Adebiyi, "Big data stream analysis: A systematic literature review," *J. Big Data*, vol. 6, no. 1, pp. 1–30, 2019.

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

[3] N. D. Sidiropoulos, L. D. Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, 2017.

[4] T. Akidau, S. Chernyak, and R. Lax, *Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing*, 2018.

[5] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, 2011.

[6] A. Cichocki, N. Lee, I. V. Oseledets, A.-H. Phan, Q. Zhao, and D. P. Mandic, "Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions," *Found. Trends Mach. Learn.*, vol. 9, no. 4-5, pp. 249–429, 2016.

[7] H. Liu, L. T. Yang, Y. Guo, X. Xie, and J. Ma, "An incremental tensor-train decomposition for cyber-physical-social big data," *IEEE Trans. Big Data*, vol. 7, no. 2, pp. 341–354, 2021.

[8] X. Wang, L. T. Yang, Y. Wang, L. Ren, and M. J. Deen, "ADTT: A highly efficient distributed tensor-train decomposition method for IIoT big data," *IEEE Trans Ind. Inf.*, vol. 17, no. 3, pp. 1573–1582, 2021.

[9] L. T. Thanh, K. Abed-Meraim, N. Linh-Trung, and R. Boyer, "Adaptive algorithms for tracking tensor-train decomposition of streaming tensors," in *Eur. Signal Process. Conf.*, 2020, pp. 995–999.

[10] L. T. Thanh, K. Abed Meraim, N. Linh Trung, and A. Hafiane, "Scalable Adaptive Tensor-Train Decomposition with Incomplete Data," *IEEE Trans. Circuits Syst. II Express Briefs*, 2022 (under review).

[11] S. S. Haykin, *Adaptive Filter Theory*, 2008.

[12] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. Imaging Sci.*, vol. 6, no. 3, pp. 1758–1789, 2013.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[14] L. T. Thanh, N. V. Dung, N. L. Trung, and K. Abed-Meraim, "Robust subspace tracking with missing data and outliers: Novel algorithm with convergence guarantee," *IEEE Trans. Signal Process.*, vol. 69, pp. 2070–2085, 2021.

[15] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: A survey and extensions," *Math. Methods Oper. Res.*, vol. 66, no. 3, pp. 373–407, 2007.

[16] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video," in *IEEE Conf. Comput. Vis. Pattern Recogn.*, 2012, pp. 1568–1575.