A Contemporary and Comprehensive Survey on Streaming Tensor Decomposition

Le Trung Thanh [1,1], karim abed-meraim [2], Nguyen Linh-Trung [2], and adel hafiane [2]

[1]University of Orleans
[2]Affiliation not available

October 31, 2023

Abstract

Tensor decomposition has been demonstrated to be successful in a wide range of applications, from neuroscience and wireless communications to social networks. In an online setting, factorizing tensors derived from multidimensional data streams is however non-trivial due to several inherent problems of real-time stream processing. In recent years, many research efforts have been dedicated to developing online techniques for decomposing such tensors, resulting in significant advances in streaming tensor decomposition or tensor tracking. This topic is emerging and enriches the literature on tensor decomposition, particularly from the data stream analystics perspective. Thus, it is imperative to carry out an overview of tensor tracking to help researchers and practitioners understand its development and achievements, summarise the current trends and advances, and identify challenging problems. In this article, we provide a contemporary and comprehensive survey on different types of tensor tracking techniques. We particularly categorize the state-of-the-art methods into three main groups: streaming CP decompositions, streaming Tucker decompositions, and streaming decompositions under other tensor formats (i.e., tensor-train, t-SVD, and BTD). In each group, we further divide the existing algorithms into sub-categories based on their main optimization framework and model architectures. Finally, we present several research challenges, open problems, and potential directions of tensor tracking in the future.

A Contemporary and Comprehensive Survey on Streaming Tensor Decomposition

Le Trung Thanh, Karim Abed-Meraim, *Fellow, IEEE*, Nguyen Linh Trung, *Senior Member, IEEE,* and Adel Hafiane, *Member, IEEE*

Abstract—Tensor decomposition has been demonstrated to be successful in a wide range of applications, from neuroscience and wireless communications to social networks. In an online setting, factorizing tensors derived from multidimensional data streams is however nontrivial due to several inherent problems of real-time stream processing. In recent years, many research efforts have been dedicated to developing online techniques for decomposing such tensors, resulting in significant advances in streaming tensor decomposition or tensor tracking. This topic is emerging and enriches the literature on tensor decomposition, particularly from the data stream analytics perspective. Thus, it is imperative to carry out an overview of tensor tracking to help researchers and practitioners understand its developments and achievements, summarise the current trends and advances, and identify challenging problems. In this article, we provide a contemporary and comprehensive survey on different types of tensor tracking techniques. We particularly categorize the state-of-the-art methods into three main groups: streaming CP decompositions, streaming Tucker decompositions, and streaming decompositions under other tensor formats (i.e., tensor-train, t-SVD, and BTD). In each group, we further divide the existing algorithms into sub-categories based on their main optimization framework and model architectures. Finally, we present several applications, research challenges, open problems, and potential directions of tensor tracking in the future.

Index Terms—Tensor decomposition, CP, Tucker, tensor-train, t-SVD, BTD, data stream, online optimization, low-rank tensor approximation.

1 Introduction

Tensor decomposition (TD) has attracted much attention from the signal/image processing and machine learning communities [1]. As a tensor is a multiway array, it provides a natural representation for multidimensional data. Accordingly, TD which factorizes a tensor into a set of basis components (e.g., vectors, matrices, or simpler tensors) has become a popular tool for multivariate and high-dimensional data analysis. In particular, we have witnessed significant advances in TD and a rapid growth in its applications over the last two decades [2]. Several types of TD, such as CANDECOMP/PARAFAC (CP) [3], high-order SVD (HOSVD)/Tucker [4], tensor train/network [5], t-SVD [6], and block-term decomposition (BTD) [7], have been developed and successfully applied to various domains, from neuroscience [8], [9] and wireless communications [10], [11] to social networks [12], [13].

The demand for (near) real-time stream processing has been increasing over the years since many modern applications (e.g., Internet-of-Things) generate massive amounts of streaming data over time and analytical insights from such data can bring several benefits, e.g., for real-time decision making [14]. As its name implies, (near) real-time stream processing needs to immediately deliver and analyse data streams upon their arrival. Since streaming data grow bigger, faster, and become more complex by the time, there exist several inherent problems which are still challenging issues, such as (i) the unbounded size of streaming data, (ii) time-varying model, concept drift, or dataset shift, and (iii) uncertainty and imperfection, etc. We refer the readers to [14], [15] for good surveys on data stream analysis.

When using tensors to represent data streams, TD is generally referred to as dynamic tensor analysis, tensor tracking or adaptive/online/streaming tensor decomposition. Specifically, factorizing a streaming tensor is nontrivial due to several computational challenges. First, as tensor streams are continuously generated, their volume grows significantly over time and possibly to infinity. Applying the conventional batch TD methods to such tensors is not possible as they require data to be stored and processed offline. Second, properties of streaming tensors (e.g., the low-rank approximation model) can vary with time in unforeseen ways. Moreover, tensor streams often happen in real-time, so retransmission of a stream is difficult, even impossible. Accordingly, batch tensor estimation and decomposition become less accurate when time passes. Last but not least, some modern applications require high-speed data acquisition systems to rapidly acquire and process massive data streams. In such a case, very fast and (near) real-time processing is highly important. However, batch TDs are often of high complexity, and hence turn out to be inefficient. These characteristics make tensor tracking much different from batch tensor decomposition and lead to several distinguishing requirements for tensor trackers, such as low latency and memory storage, high scalability, adaptation to time variation, and robustness, to name a few.

As the literature of tensor tracking has dramatically expanded in recent years, it is imperative to conduct an extensive overview of the state-of-the-art tensor tracking algorithms to help researchers and practitioners to identify: (i) which topics in tensor tracking are significant and emerging, (ii) what kind of tracking models and related analysis techniques have already been deployed to date and how to apply them in specific tasks, and (iii) main research challenges, open problems, and potential directions of tensor tracking in the future.

1.1 State-of-the-art Tensor Surveys

The very first introduction to TD was provided by Rasmus in [16] two decades ago. This reference offered a tutorial on CP decomposition covering features, properties, methods, and applications in chemometrics. Since then, there have been many published survey papers which provided different points of view on tensor computation. We can broadly divide them into three classes, including (i) surveys on models, methods, and tools for factorizing tensors, (ii) surveys on general tensor problems, e.g., tensor operations, uniqueness, ranks, filtering, spectral analysis,

- *Le Trung Thanh is with the University of Orléans, INSA-CVL, PRISME, France and with the VNU University of Engineering and Technology, Hanoi, Vietnam. Email: trung-thanh.le@univ-orleans.fr.*
- *Karim Abed-Meraim and Adel Hafiane are with the University of Orléans, INSA-CVL, PRISME, France. Karim Abed-Meraim is also a member of IUF (Institut Universitaire de France). Emails: karim.abed-meraim@univ-orleans.fr, adel.hafiane@insa-cvl.fr.*
- *Nguyen Linh Trung (corresponding author) is with the VNU University of Engineering and Technology, Hanoi, Vietnam. Email: lintrung@vnu.edu.vn.*

TABLE 1
The State-of-the-art Surveys on Tensor Decompositions and Applications

Class	Review (Year)	Objects & Topics	Key Contributions
Surveys on tensor factorization models, methods, and tools	[16] (1997)	CP/PARAFAC decomposition	An overview of CP decomposition with respect to aspects: features, properties, methods, and applications in chemometrics.
	[17] (2008)	CP & Tucker decomposition	A literature survey on unsupervised multiway data analysis: multiway models (i.e., CP family and Tucker family), their workhorse algorithms and applications.
	[18] (2009)	CP & Tucker decomposition	An extensive survey on main algorithms, properties and applications of CP, Tucker decompositions and their variants + A list of software and toolboxes for tensor processing.
	[19] (2010)	Tucker/HOSVD decomposition	An overview of numerical methods for Tucker/HOSVD decomposition and its applications in signal processing.
	[20] (2013)	Low-rank tensor approximations	A literature survey on low-rank tensor approximation models and algorithms.
	[21] (2014)	Incomplete tensor decomposition	A survey on numerical methods for factorizing incomplete tensors and their connections to signal processing applications.
	[22] (2016)	Tensor network	An extensive tutorial on tensor networks, their operations and algorithms.
	[23] (2020)	Tucker/HOSVD decomposition	A survey on randomized algorithms for computing Tucker/HOSVD decomposition.
	[24] (2020)	Structured tensor decomposition	A unified nonconvex optimization perspective for computing large-scale matrix and tensor decompositions with structured factors.
Surveys on general tensor problems	[25] (2007)	Tensor filtering	A review of tensor signal algebraic filtering methods.
	[26] (2009)	CP & Tucker decompositions	A review of theoretical results on the existence, uniqueness, degeneracies, and numerical complexities of alternating least-squares and other tales.
	[27] (2013)	Complexity of tensor problems	An in-depth survey on theoretical and complexity results of some tensor problems: e.g., tensor ranks, tensor eigen/singular values, and the best rank-1 approximation.
	[28] (2014)	Tensor formats & tensor ranks	A brief introduction on different types of tensor formats and tensor ranks.
	[1] (2017)	Fundamentals & backgrounds	An comprehensive overview of tensor decompositions w.r.t. aspects: uniqueness, tensor ranks, algorithms, bounds, and applications + A list of software and toolboxes for tensor processing.
	[29] (2018)	Tensor PCA	An introduction to tensors and tensor decompositions in the lens of PCA.
	[30] (2022)	Reproducibility	A review of critical issues and solutions for guaranteeing the computational reproducibility in matrix and tensor factorizations.
Surveys on tensor applications	[31] (2011)	Data analysis	An overview of tensor applications for a wide variety of data and problem domains.
	[32] (2015)	Signal processing	A comprehensive survey on tensor decompositions for signal processing.
	[33] (2015)	EEG applications	A brief survey on tensor decompositions of EEG signals.
	[9] (2015)	Neuroscience	A survey on tensor analysis and fusion of multimodal brain images.
	[34] (2016)	Anomaly detection	An interdisciplinary survey on tensor-based anomaly detection.
	[35] (2016)	Biomedical science	A brief survey on matrix- and tensor-based component analysis methods for biomedical data.
	[36] (2017)	Data fusion	A review of tensor decompositions with emphasis on data fusion applications.
	[37] (2017)	Machine learning & data analysis	An tutorial on tensor network models for super-compressed representation of data and their applications in machine learning and data analytics.
	[38] (2019)	Machine learning	An overview of tensor techniques and applications in machine learning.
	[39] (2021)	Multisensor processing	A comprehensive survey on tensor methods for multisensor signal processing.
	[11] (2021)	Wireless communications	A comprehensive overview of tensor decompositions for wireless communications.
	[13] (2021)	Social networks	A survey on tensor decomposition for analysing time-evolving social networks.
	[40] (2021)	Computer vision & deep learning	A practical overview of tensor methods for computer vision and deep learning
	[41] (2022)	System identification	A tutorial on tensor methods for nonlinear system identification.
	[2] (2022)	Data analysis	A systematic and up-to-date overview of tensor decompositions from the engineer's view
	This paper	**Streaming tensor decomposition (Tensor tracking)**	**A contemporary and comprehensive survey on methods for factorizing tensors derived from data streams under CP, Tucker, TT, t-SVD, and BTD formats. Applications, research challenges, open problems, and future directions for tensor tracking.**

and complexity, and (iii) surveys on tensor applications. We refer the readers to Tab. 1 for the main contributions of the state-of-the-art surveys on tensors.

Among them, the most cited survey paper is the work of Kolda *et al.* in [18] that was published more than a decade ago. The survey presented basic multiway models (i.e., CP family and Tucker family) and workhorse algorithms for factorizing tensors under these models. Some applications and software for tensors were also mentioned. The second key survey in the literature is the work of Sidiropoulos *et al.* in [1] that appeared five years ago. To fill some gaps in the existing surveys on CP and Tucker decompositions of that time, the authors provided an in-depth overview of tensors with respect to the following aspects: uniqueness, ranks, bounds, algorithms, and applications. Moreover, an up-to-date list of software and toolboxes for tensor computation was provided therein. To extend beyond the two standard multiway models, Cichocki *et al.* conducted a comprehensive tutorial on tensor networks in [22], [37]. Particularly, its coverage includes tensor network models, the associated operations and algorithms, and their applications. Besides, it also highlighted connections of tensor networks to dimensionality reduction and large-scale optimization problems. Very recently, Liu *et al.* provided a general overview of tensors from the engineer's point of view in the book [2]. It covers various aspects of tensor computations and decompositions, from operations and well-known multiway representations to tensor-based data analysis techniques and practical applications.

However, to date, we are not aware of any survey paper

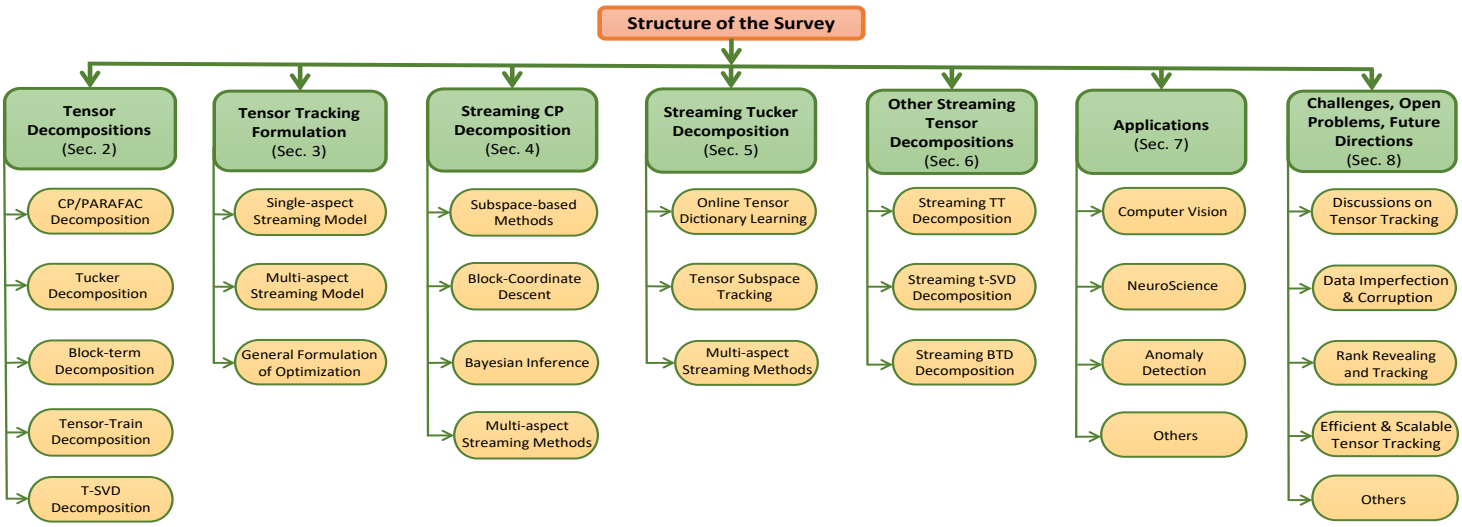


Fig. 1. Structure of this survey.

TABLE 2
Notational conventions.

Notations	Descriptions
$x, \mathbf{x}, \mathbf{X}, \boldsymbol{\mathcal{X}}$	scalar, vector, matrix, and tensor, respectively
$x_{i_1,\ldots,i_N}$ / $[\boldsymbol{\mathcal{X}}]_{i_1\ldots i_N}$	$(i_1,\ldots,i_N)$-th entry of $\boldsymbol{\mathcal{X}}$
$\mathbf{x} = \text{vec}(\mathbf{X})$	vectorization of $\mathbf{X}$
$\mathbf{X} = \text{diag}(\mathbf{x})$	diagonal matrix $\mathbf{X}$ with $\mathbf{x}$ on the main diagonal
$\mathbf{X}(i,:), \mathbf{X}(:,j)$	i-th row and j-th column of $\mathbf{X}$
$\mathbf{X}^\top, \mathbf{X}^{-1}, \mathbf{X}^\#$	transpose, inverse, and pseudo-inverse of $\mathbf{X}$
$\underline{\mathbf{X}}^{(n)} = \text{unfold}_n(\boldsymbol{\mathcal{X}})$	mode-n unfolding of $\boldsymbol{\mathcal{X}}$
$\boldsymbol{\mathcal{Y}} = \text{bcirc}(\boldsymbol{\mathcal{X}})$	block circulant tensor $\boldsymbol{\mathcal{Y}}$ specified by $\boldsymbol{\mathcal{X}}$
$\mathbf{U}^{(n)}$	n-th loading factor/matrix
$\circ, \odot, \circledast$	outer, Khatri-Rao, and Hadamard product
$\boldsymbol{\mathcal{X}} \times_n \mathbf{U}$	n-mode product of $\boldsymbol{\mathcal{X}}$ with $\mathbf{U}$,
$\boldsymbol{\mathcal{X}} \boxplus_n \boldsymbol{\mathcal{Y}}$	concatenation of $\boldsymbol{\mathcal{X}}$ with $\boldsymbol{\mathcal{Y}}$ along the n-th mode
$\boldsymbol{\mathcal{X}} \times_n^1 \boldsymbol{\mathcal{Y}}$	mode-$(n,1)$ contracted product of $\boldsymbol{\mathcal{X}}$ with $\boldsymbol{\mathcal{Y}}$
$\boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}}$	t-product of $\boldsymbol{\mathcal{X}}$ with $\boldsymbol{\mathcal{Y}}$ (defined in [6])
$\boldsymbol{\mathcal{X}} \subseteq \boldsymbol{\mathcal{Y}}$	$\boldsymbol{\mathcal{X}}$ is a sub-tensor of $\boldsymbol{\mathcal{Y}}$
$\boldsymbol{\mathcal{X}} \cup \boldsymbol{\mathcal{Y}}$	union of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$
$\boldsymbol{\mathcal{X}} \backslash \boldsymbol{\mathcal{Y}}$	relative difference between $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$
$[\![\{\mathbf{U}^{(n)}\}_{n=1}^N]\!]$	$\sum_{i=1}^r \mathbf{U}^{(1)}(:,i) \circ \mathbf{U}^{(2)}(:,i) \circ \cdots \circ \mathbf{U}^{(N)}(:,i)$
$[\![\boldsymbol{\mathcal{X}}; \{\mathbf{U}^{(n)}\}_{n=1}^N]\!]$	$\boldsymbol{\mathcal{X}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)}$
$\bigodot_{n=1}^N \mathbf{U}^{(n)}$	$\mathbf{U}^{(N)} \odot \mathbf{U}^{(N-1)} \odot \cdots \odot \mathbf{U}^{(1)}$
$\bigotimes_{n=1}^N \mathbf{U}^{(n)}$	$\mathbf{U}^{(N)} \otimes \mathbf{U}^{(N-1)} \otimes \cdots \otimes \mathbf{U}^{(1)}$
$\|.\|_F, \|.\|_p, \|.\|_*$	Euclidean norm, ℓ_p norm, and nuclear norm
FFT, iFFT	Fast Fourier transform and its inverse

specifically reviewing the problem of streaming tensor decomposition or tensor tracking. Therefore, it is of great interest to carry out an overview of this topic to enrich the tensor literature.

1.2 Main Contribution

In this paper, we present a contemporary and comprehensive survey on the state-of-the-art online techniques which are capable of factorizing tensors derived from data streams.

Our survey begins with basic coverage of five common tensor decompositions and their main features. They are CP/PARAFAC, HOSVD/Tucker, BTD, tensor-train, and t-SVD. Two kinds of streaming models are then introduced to represent streaming tensors, namely single-aspect and multi-aspect. Next, we review four main groups of streaming CP decomposition algorithms: (i) subspace-based, (ii) block-coordinate descent, (iii) Bayesian inference, and (iv) multi-aspect streaming CP decomposition. Under the Tucker format, we categorize currently available single-aspect tensor tracking algorithms into two main classes: online tensor dictionary learning and tensor subspace tracking. Multi-aspect streaming Tucker decomposition algorithms are then overviewed. In addition, we provide a short survey on other online techniques for tracking tensors under tensor-train, t-SVD, and BTD formats. Finally, we discuss a number of important challenges and open problems as well as highlight potential directions for the problem of tensor tracking in the future. To the best of our knowledge, our survey offers for the first time a thorough review of techniques for factorizing tensors in an online fashion. Fig. 1 depicts the organization of the paper. For ease of reference, we summarize in Tab. 2 notations which are frequently used in this paper.

2 Tensor Decompositions

In this section, we briefly describe the background of the five popular tensor decompositions which have already been deployed to factorize streaming tensors in an online fashion. They are CP, Tucker, BTD, tensor-train, and t-SVD.

2.1 CP Decomposition

Under the CP format [16], a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ can be decomposed into a set of N matrices $\{\mathbf{U}^{(n)}\}_{n=1}^N$ sharing the same number of columns as follows

$$\boldsymbol{\mathcal{X}} \triangleq [\![\{\mathbf{U}^{(n)}\}_{n=1}^N]\!] = \sum_{i=1}^r \mathbf{U}^{(1)}(:,i) \circ \cdots \circ \mathbf{U}^{(N)}(:,i), \quad (1)$$

where the so-called tensor factor $\mathbf{U}^{(n)}$ is of size $I_n \times r$ with $1 \leq n \leq N$. The smallest integer r satisfying (1) is referred to as the CP-rank of $\boldsymbol{\mathcal{X}}$. CP is among the best memory-saving format for representing high-order tensors, and hence, it can overcome the curse of dimensionality which particularly limits the order of tensors to be analysed. Under certain conditions, CP decomposition is essentially unique up to a permutation and scale. However, its main disadvantage is that finding the true rank r is known as an NP-hard problem [27]. Even though the CP-rank is given in advance, the best rank-r approximation of a tensor may not exist [42]. To compute the CP decomposition, one of the most widely-used approaches is based on the alternating least-squares (ALS) technique [18].

2.2 Tucker Decomposition

Under the Tucker format [4], we can factorize $\boldsymbol{\mathcal{X}}$ into a core tensor $\boldsymbol{\mathcal{G}}$ of a smaller size and N factors $\{\mathbf{U}^{(n)}\}_{n=1}^N$ as

$$\boldsymbol{\mathcal{X}} \triangleq [\![\boldsymbol{\mathcal{G}}; \{\mathbf{U}^{(n)}\}_{n=1}^N]\!] = \boldsymbol{\mathcal{G}} \times_1 \mathbf{U}^{(1)} \times_2 \cdots \times_N \mathbf{U}^{(N)}, \quad (2)$$

where $\boldsymbol{\mathcal{G}}$ is of size $r_1 \times r_2 \times \cdots \times r_N$ ($r_n \leq I_n$) and $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}$ is an orthogonal matrix. The vector $\mathbf{r} = [r_1, r_2, \ldots, r_N]$ is called the multilinear rank or rank-$(r_1, r_2, \ldots, r_N)$ of $\boldsymbol{\mathcal{X}}$. Tucker

decomposition is more flexible than CP in the sense that we can write any $\boldsymbol{\mathcal{X}}$ in the form (2) and its computation can be done effectively. The two most popular algorithms for computing (2) are HOSVD and Higher-order Orthogonal Iteration (HOOI) [43]. Both HOSVD and HOOI offer a good rank-$(r_1, r_2, \ldots, r_N)$ tensor approximation for $\boldsymbol{\mathcal{X}}$ and they can be efficiently implemented in practice. In general, the Tucker representation is not unique but the subspace covering $\mathbf{U}^{(n)}$ is physically unique. Therefore, the main interest in Tucker decomposition is for finding subspaces of the tensor factors, and hence, for approximation, dimensionality reduction, and feature extraction [1].

2.3 Block-Term Decomposition

Block-term decomposition (BTD) allows to represent $\boldsymbol{\mathcal{X}}$ as a sum of low multilinear-rank tensors [7]:

$$\boldsymbol{\mathcal{X}} = \sum_{l=1}^{L} [\![\boldsymbol{\mathcal{G}}_l; \{\mathbf{U}_l^{(n)}\}_{n=1}^{N}]\!], \tag{3}$$

where $\{\boldsymbol{\mathcal{G}}_l\}_{l=1}^{L}$ with $\boldsymbol{\mathcal{G}}_l \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$ is the set of core tensors, $\mathbf{U}^{(n)} = \left[\mathbf{U}_1^{(n)}, \ldots, \mathbf{U}_L^{(n)}\right]$ with $\mathbf{U}_l^{(n)} \in \mathbb{R}^{I_n \times r_n}$ is the n-th tensor factor, and $r_n \leq I_n \; \forall \, l, n$. The BTD format (3) can be considered as a combination of CP and Tucker. As its name reveals, the basic components in BTD are rank-$(r_1, r_2, \ldots, r_N)$ blocks while they are rank-1 terms in CP. When these blocks are rank-1 tensors (i.e., $r_n = 1 \; \forall n$), it boils down to CP. When it has only one block (i.e., $L = 1$), BTD becomes the standard Tucker decomposition. It is worth noting that the number of blocks L relies on the block's size. Like CP, BTD is essentially unique [7]. The common approach to find (3) is also based on the ALS technique [44].

2.4 Tensor-train Decomposition

Tensor-train (TT) decomposition expresses $\boldsymbol{\mathcal{X}}$ as a multilinear product of third-order tensors $\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N}$ according to

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}}^{(1)} \times_1^1 \boldsymbol{\mathcal{G}}^{(2)} \times_2^1 \cdots \times_N^1 \boldsymbol{\mathcal{G}}^{(N)}, \tag{4}$$

where $\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$ is the n-th TT-core (aka tensor carriage) with $n = 1, 2, \ldots, N$. Here, $r_0 = r_N = 1$ and the quantities $\{r_n\}_{n=1}^{N-1}$ are called TT-ranks [5]. This type of TD offers several appealing benefits for representing tensors, especially high-order tensors. Given an arbitrary tensor $\boldsymbol{\mathcal{X}}$, we always find a set of TT-cores $\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N}$ satisfying (4) with suitable TT ranks. Besides, its TT-ranks can be effectively estimated in a stable way in contrast to the CP-rank determination [27]. TT also offers a memory-saving representation for tensors and can break the curse of dimensionality like CP. With respect to the implementation, the workhorse algorithm to compute TT is TT-SVD [5].

2.5 T-SVD Decomposition

Tensor SVD (t-SVD) is another multiway extension of SVD for decomposing tensors in which $\boldsymbol{\mathcal{X}}$ is factorized into three tensors $\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{G}}$, and $\boldsymbol{\mathcal{V}}$ of the same order:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{G}} * \boldsymbol{\mathcal{V}}^H, \tag{5}$$

where $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ are unitary tensors, and $\boldsymbol{\mathcal{G}}$ is a rectangle f-diagonal tensor whose frontal slices are diagonal matrices [6]. To define the low-rank tensor approximation under the t-SVD format, the so-called tubal-rank r_{t} is determined as the number of non-zero tubes of $\boldsymbol{\mathcal{G}}$ (e.g., when the tensor $\boldsymbol{\mathcal{X}}$ is of order 3, $r_{\mathrm{t}}(\boldsymbol{\mathcal{X}}) = \sum_i \mathbf{1}[\boldsymbol{\mathcal{G}}(i,i,:) \neq \mathbf{0}]$ where $\mathbf{1}$ is an indicator function). The t-SVD algebraic framework is quite different from the classical multilinear algebra in other types of TD. Thanks to the t-product and Fourier transform, several linear, multilinear operators and other transformations are successfully extended from matrices to tensors, such as transpose, orthogonality, and inverse. In particular, t-SVD can be obtained by computing SVDs in Fourier domain and its performance (i.e., exact recovery with high probability) can be guaranteed under mild conditions [6].

3 Tensor Tracking Formulation

In this section, the problem of tensor tracking is formulated. Specifically, we first divide streaming tensor tracking models into two classes and then construct some terminologies to support the problem statement. They are single-aspect and multi-aspect streaming models. After that, we formulate the general tensor tracking problem which is suitable for many applications.

3.1 Single-aspect Streaming Model

In the classical online setting, we are interested in the decomposition of an N-order streaming tensor $\boldsymbol{\mathcal{X}}_t$ fixing all but one dimension. Without loss of generality, we suppose the last dimension is temporal, and hence, we can write $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times I_N^t}$ where I_N^t is increasing with time.

The following definition of temporal slices is useful to formulate the problem of single-aspect tensor tracking.

Definition 1 (Temporal slice). Given a streaming tensor $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N^t}$, we say $\boldsymbol{\mathcal{Y}}_\tau = \boldsymbol{\mathcal{X}}_t(:, \ldots, :, \tau) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1}}$ is the τ-th *temporal slice* of $\boldsymbol{\mathcal{X}}_t$ for $1 \leq \tau \leq I_N^t$.

Without loss of generality, we assume that $I_N^t = t$ meaning that at each time instant one new slice of the tensor is observed. Accordingly, the streaming tensor $\boldsymbol{\mathcal{X}}_t$ can be viewed as a set of temporal slices $\{\boldsymbol{\mathcal{Y}}_\tau\}_{\tau=1}^{t}$. In other words, $\boldsymbol{\mathcal{X}}_t$ is derived from appending the new comming temporal slice $\boldsymbol{\mathcal{Y}}_t$ to the previous observations $\boldsymbol{\mathcal{X}}_{t-1}$ along the time dimension, i.e.,

$$\boldsymbol{\mathcal{X}}_t = \boldsymbol{\mathcal{X}}_{t-1} \boxplus_N \boldsymbol{\mathcal{Y}}_t \quad \text{and} \quad I_N^t = I_N^{t-1} + 1 = t. \tag{6}$$

Generally, $\boldsymbol{\mathcal{Y}}_t$ has the form

$$\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast (\boldsymbol{\mathcal{L}}_t + \boldsymbol{\mathcal{N}}_t + \boldsymbol{\mathcal{O}}_t), \tag{7}$$

where $\boldsymbol{\mathcal{L}}_t$ is a low-rank component, $\boldsymbol{\mathcal{P}}_t$ is a binary tensor, $\boldsymbol{\mathcal{N}}_t$ is a noise tensor, and $\boldsymbol{\mathcal{O}}_t$ is a sparse tensor. The data model (7) is a general form which is suitable for many scenarios. For example, $\boldsymbol{\mathcal{P}}_t$ represents missing and observed entries of $\boldsymbol{\mathcal{Y}}_t$; $\boldsymbol{\mathcal{N}}_t$ is an additive white Gaussian noise; and $\boldsymbol{\mathcal{O}}_t$ denotes the sparse outliers. Meanwhile, the low-rank $\boldsymbol{\mathcal{L}}_t$, which can be formulated by any tensor formats, can be static or time-varying.

Based on these terminologies, the problem of single-aspect tensor tracking can be formally stated as follows:

> **Single-aspect Tensor Tracking**: At time t, given a temporal slice $\boldsymbol{\mathcal{Y}}_t$ and old estimates of $\boldsymbol{\mathcal{X}}_{t-1}$ (e.g., core tensors and tensor factors), we want to track the new estimates of $\boldsymbol{\mathcal{X}}_t = \boldsymbol{\mathcal{X}}_{t-1} \boxplus_N \boldsymbol{\mathcal{Y}}_t$ in time.

3.2 Multi-aspect Streaming Model

In some modern online applications, tensor data may evolve in multiple dimensions/modes over time, and thus, the single-aspect streaming model is not useful for modelling such streaming data. In [45], Fanaee-T *et al.* for the first time introduced the concept of multi-aspect streaming tensors to represent streaming data having more than one dimension increasing with time. Since then, some online algorithms have been developed to deal with the problem of multi-aspect streaming tensor decomposition.

For convenience, we first introduce the definitions of multi-aspect streaming tensors and temporal tubes.

Definition 2 (Multi-aspect streaming tensor). A set of N-order tensors $\{\boldsymbol{\mathcal{X}}_t\}_{t \geq 1}$ is called a multi-aspect streaming tensor sequence denoted as $\{\boldsymbol{\mathcal{X}}\}$ when $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I_1^t \times I_2^t \times \cdots \times I_N^t}$,

$I_n^t = I_n^{t-1} + W_n^t$ where $W_n^t \geq 0, 1 \leq n \leq N$, and $\boldsymbol{\mathcal{X}}_{t-1}$ is a sub-tensor of $\boldsymbol{\mathcal{X}}_t$. If $\boldsymbol{\mathcal{X}}_t$ belongs to such a sequence $\{\boldsymbol{\mathcal{X}}\}$, we say that $\boldsymbol{\mathcal{X}}_t$ is a *multi-aspect streaming tensor*.

Definition 3 (Temporal tube). Given two successive tensors $\boldsymbol{\mathcal{X}}_{t-1}$ and $\boldsymbol{\mathcal{X}}_t$ derived from the same multi-aspect streaming tensor sequence $\{\boldsymbol{\mathcal{X}}\}$, the coming data at time t can be represented by $\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{X}}_t \backslash \boldsymbol{\mathcal{X}}_{t-1}$ of the same size as $\boldsymbol{\mathcal{X}}_t$ with entries

$$\left[\boldsymbol{\mathcal{Y}}_t\right]_{i_1,\ldots,i_N} = \begin{cases} \left[\boldsymbol{\mathcal{X}}_t\right]_{i_1,\ldots,i_N} & \text{if } I_n^{t-1} < i_n \leq I_n^t, \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

for $1 \leq n \leq N$. We say that the non-zero entries in $\boldsymbol{\mathcal{Y}}_t$ are *temporal tubes*.

Now, we can state the problem of multi-aspect tensor tracking as follows:

> **Multi-aspect Tensor Tracking**: At time t, given temporal tubes in $\boldsymbol{\mathcal{Y}}_t$, and old estimates of $\boldsymbol{\mathcal{X}}_{t-1}$ (e.g., core tensors and tensor factors), we want to track the new estimates of $\boldsymbol{\mathcal{X}}_t = \boldsymbol{\mathcal{X}}_{t-1} \cup \boldsymbol{\mathcal{Y}}_t$ in time.

It is worth noting that the single-aspect tensor tracking problem also belongs to the class of multi-aspect tensor tracking where most of the tensor dimensions I_n are constant by setting $W_n^t = 0$, except the last one I_N^t. Besides, temporal slices may be regarded as frontal slices of the tensor $\boldsymbol{\mathcal{Y}}_t$ defined as in (8).

3.3 General Formulation of Optimization

We here provide a general formulation of tensor tracking which can be used in many applications. In particular, the optimization problem can be written as

$$\begin{aligned} \min_{\{\boldsymbol{\mathcal{G}}\},\{\boldsymbol{\mathcal{U}}\},\boldsymbol{\mathcal{O}}} \Bigg[& \underbrace{\sum_{\tau=1}^{t} \beta_\tau \ell\Big(\boldsymbol{\mathcal{Y}}_\tau, \boldsymbol{\mathcal{P}}_\tau, \{\boldsymbol{\mathcal{G}}\}, \{\boldsymbol{\mathcal{U}}\}, \boldsymbol{\mathcal{O}}\Big)}_{\text{Minimize residual errors}} \\ & + \underbrace{\rho_{\text{G}} \mathcal{R}_{\text{G}}\big(\{\boldsymbol{\mathcal{G}}\}\big)}_{\text{Regularize cores}} + \underbrace{\rho_{\text{U}} \mathcal{R}_{\text{U}}\big(\{\boldsymbol{\mathcal{U}}\}\big)}_{\text{Regularize factors}} + \underbrace{\rho_{\text{O}} \mathcal{R}_{\text{O}}\big(\boldsymbol{\mathcal{O}}\big)}_{\text{Promote sparsity}} \\ & + \underbrace{\lambda_{\text{G}} \mathcal{L}_{\text{G}}\big(\{\boldsymbol{\mathcal{G}}\}\big) + \lambda_{\text{U}} \mathcal{L}_{\text{U}}\big(\{\boldsymbol{\mathcal{U}}\}\big)}_{\text{Application dependent constraints}} \Bigg]. \end{aligned} \tag{9}$$

Here, $\{\boldsymbol{\mathcal{G}}\}$ and $\{\boldsymbol{\mathcal{U}}\}$ denote the set of core tensors and tensor factors respectively, while $\boldsymbol{\mathcal{O}}$ is to represent data corruptions by impulsive noise or outliers. Specifically, the three terms in the second line of (9) are used to present regularizations or penalty terms imposed on parameters of interest. The last two penalty terms of (9) are for the application orientation. The main loss function $\ell(.)$ is defined to minimize the residual errors between the estimations and observations.

4 Streaming CP Decomposition

The primary objective of this section is to provide technical descriptions of the-state-of-the-art online techniques for factorizing streaming tensors under the CP format. In the literature, there are many streaming CP algorithms and they can be categorized into the following groups: (i) subspace-based methods, (ii) block-coordinate descent methods, (iii) Bayesian inference, and (iv) multi-aspect streaming CP decompositions. The three former groups are particularly developed for single-aspect streaming models, while the latter is dedicated to the factorization of tensors having more than one temporally varying mode. The readers are referred to Tabs. 3 and 4 for quick comparisons of the existing streaming CP decomposition algorithms. In what follows, we take each group into consideration.

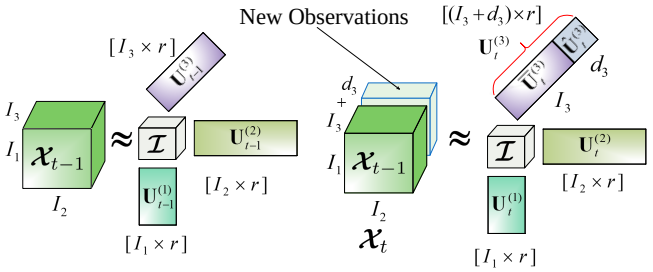


Fig. 2. Single-aspect streaming CP decomposition of a 3rd-order tensor. Here, the core $\boldsymbol{\mathcal{I}}$ is an identity tensor in which the main diagonal entries (i.e., $[\boldsymbol{\mathcal{I}}]_{j,\ldots,j}$) are equal to 1 and the remaining ones are zeros.

4.1 Subspace-based Methods

The very first study addressing the problem of streaming CP decomposition is of Nion and Sidiropoulos in [46]. Specifically, the authors introduced the two novel adaptive CP algorithms called PARAFAC-SDT and PARAFAC-RLS capable of tracking third-order streaming tensors having one temporal dimension. Both algorithms are based on the subspace-based approach in which we first track a low-dimensional tensor subspace, and then recover the loading matrices from exploiting its Khatri-Rao structure. Following the same line, some other adaptive CP algorithms were proposed for tensor tracking such as CP-PETRELS [53], 3D-OPAST [47], and SOAP [50]. In the following, we describe their subspace-based framework for factorizing streaming tensors with time.

First, we recall that the low-rank $\boldsymbol{\mathcal{L}}_t$ of $\boldsymbol{\mathcal{Y}}_t$ has the form $\boldsymbol{\mathcal{L}}_t = [\![\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_t^{(N)}]\!]$, where $\mathbf{u}_t^{(N)}$ is the last row of $\mathbf{U}_t^{(N)}$. Thus, $\boldsymbol{\mathcal{L}}_t$ can be recast into the form:

$$\boldsymbol{\ell}_t \triangleq \text{vec}(\boldsymbol{\mathcal{L}}_t) = \left[\bigodot_{n=1}^{N-1} \mathbf{U}_t^{(n)}\right] \left(\mathbf{u}_t^{(N)}\right)^\top = \mathbf{H}_t \left(\mathbf{u}_t^{(N)}\right)^\top, \tag{10}$$

where $\mathbf{H}_t \in \mathbb{R}^{I_1 \ldots I_{N-1} \times r}$ plays a role as a mixing matrix while $\mathbf{u}_t^{(N)}$ can be viewed as a coefficient vector in subspace tracking problems. Accordingly, streaming CP decomposition can boil down to a constrained problem of subspace tracking where the basis matrix has a Khatri-Rao structure. Particularly for $N = 3$, the authors in [46], [47], [50], [53] proposed to solve the following objective function:

$$\min_{\{\mathbf{U}^{(n)}\}_{n=1}^3} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \mathbf{p}_\tau \circledast \left(\mathbf{y}_\tau - \mathbf{H}(\mathbf{u}_\tau^{(3)})^\top\right) \right\|_2^2 \tag{11}$$

where $\mathbf{H} = \mathbf{U}^{(1)} \odot \mathbf{U}^{(2)}$, $\mathbf{y}_\tau = \text{vec}(\boldsymbol{\mathcal{Y}}_\tau)$, $\mathbf{p}_\tau = \text{vec}(\boldsymbol{\mathcal{P}}_\tau)$, and $\mathbf{u}_\tau$ is the τ-th row of the temporal factor $\mathbf{U}_t^{(3)}$, and β is a forgetting factor aimed at discounting the impact of distant observations. Specifically, (11) can be effectively solved by applying the following procedure:

- **Stage 1** : Estimate $\mathbf{H}_t$ and $\mathbf{u}_t^{(3)}$, given $\mathbf{U}_{t-1}^{(1)}$ and $\mathbf{U}_{t-1}^{(2)}$;
- **Stage 2** : Find $\mathbf{U}_t^{(1)}$, $\mathbf{U}_t^{(2)}$ satisfying $\mathbf{H}_t \simeq \mathbf{U}_t^{(1)} \odot \mathbf{U}_t^{(2)}$, and then re-update $\mathbf{H}_t \leftarrow \mathbf{U}_t^{(1)} \odot \mathbf{U}_t^{(2)}$;
- **Stage 3** : Update $\mathbf{U}_t^{(3)} = \left[(\mathbf{U}_{t-1}^{(3)})^\top (\mathbf{u}_t^{(3)})^\top\right]^\top$ where $\mathbf{u}_t^{(3)}$ can be re-estimated as in Step 1 (optional).

In stage 1, the authors in [46] proposed two solvers for estimating $\mathbf{H}_t$ and $\mathbf{u}_t$, namely recursive least-squares (RLS) and simultaneous diagonalization tracking (SDT). Chinh *et al.* in [53] adopted a well-known subspace tracking algorithm called PETRELS. Dung *et al.* in [47] applied another subspace tracking algorithm for this task, namely OPAST. In [50], the same authors introduced another tracker to estimate $\mathbf{H}_t$ with rank-1 updates.

In stage 2, all the existing subspace-based methods used the bi-SVD procedure introduced in [69] to recover $\mathbf{U}_t^{(1)}$ and $\mathbf{U}_t^{(2)}$ from $\mathbf{H}_t$. Particularly, we can represent each column of $\mathbf{H}_t$ as $\mathbf{H}_t(:,i) = \text{vec}(\mathbf{U}_t^{(1)}(:,i)(\mathbf{U}_t^{(2)}(:,i))^\top)$. Thus, the right and left

TABLE 3
Main Features of the State-of-the-art Single-aspect Streaming CP Decomposition Algorithms.

Algorithm	Missing Data?	Sparse Outliers?	High-order ($N \geq 4$)?	Convergence Guarantee?	Warm Start?	Computational Complexity	Additional Information (approaches + supports)
PARAFAC-RLST/SDT [46]	✗	✗	✗	✗	✓	$\mathcal{O}(r^2I^2)$	- Subspace-based - Tracking using RLST/SDT
3D-OPAST [47]	✗	✗	✗	✗	✓	$\mathcal{O}(rI^2)$	- Subspace-based - Tracking using OPAST
TeCPSGD [48]	✓	✗	✗	✗	random	$\mathcal{O}(r^2\|\Omega\|)$	- BCD + SGD
OLCP [49]	✗	✗	✓	✗	✓	$\mathcal{O}(r^2I^{N-1})$	- BCD + SGD
SOAP [50]	✗	✗	✗	✗	✓	$\mathcal{O}(rI^2)$	- Subspace-based + Second-order estimation - Supports nonnegativity
CP-NLS [51]	✗	✗	✗	✗	✓	$\mathcal{O}(r^2I^2)$	- Nonlinear least-squares
BRST [52]	✓	✓	✓	✗	✓	unavailable	- Variational Bayesian
CP-PETRELS [53]	✓	✗	✗	✗	✓	$\mathcal{O}(r^2\|\Omega\|)$	- Subspace-based - Tracking using PETRELS
CP-stream [54]	✗	✗	✓	✗	random	$\mathcal{O}(r^2I^{N-1})$	- ADMM + tuning-free - Supports sparsity
POST [55]	✓	✗	✓	✗	✓	$\mathcal{O}(r^3NI^{N-1})$	- Variational Bayesian
OLSTEC [56]	✓	✗	✗	✓	random	$\mathcal{O}(r^2I^2)$	- BCD + RLS
iPARAFAC [57]	✗	✗	✗	✗	✓	$\mathcal{O}(r^2\|\mathcal{S}\|)$ $\|\mathcal{S}\|$: size of the selected set	- Apache Spark[a] - Randomized MTTKRP
TensorNOODL [58]	✗	✗	✗	✓	✓	$\mathcal{O}(r^2I^2)$	- Online dictionary learning - Supports sparsity
SPADE [59]	✗	✗	✓	✗	✓	$\mathcal{O}(r^3I^{N-1})$	- Streaming PARAFAC2[b]
SliceNStitch [60]	✗	✗	✓	✗	random	$\mathcal{O}(rN\|\mathcal{S}\| + (rN)^2 + Nr^3)$ $\|\mathcal{S}\|$: no. of non-zero elements	- Sparse decomposition
SOFIA [61]	✓	✓	✓	✗	✓	$\mathcal{O}(r^3I^{N-1})$	- Holt-Winters fitting[c] - Supports seasonality
STF [62]	✓	✗	✓	✗	✓	$\mathcal{O}((N+r)Nr\|\Omega\| + NIr^3)$	- BCD + SGD
ACP [63], [64]	✓	✗	✓	✓	random	$\mathcal{O}(r^2\|\mathcal{S}\|)$ $\|\mathcal{S}\|$: size of the selected set	- Random sampling - BCD + RLS
RACP [65]	✓	✓	✓	✓	random	$\mathcal{O}(r^2I^{N-1})$	- ADMM + RLS - ℓ_1-norm penalty
Online CPDL [66]	✗	✗	✓	✓	✓	$\mathcal{O}(r^2I^{N-1})$	- Nonnegative decomposition - Markovian data - Online dictionary learning

– Suppose that $I_1 = I_2 = \cdots = I$, $r_{\text{CP}} = r$, and $|\Omega|$ is the number of observed elements.
– Abbreviations: RLS (recursive least-squares), SDT (simultaneous diagonalization tracking), BCD (block-coordinate descent), PETRELS (parallel estimation and tracking by recursive least squares), ADMM (alternating direction method of multipliers), SGD (stochastic gradient descent), and MTTKRP (matricized-tensor times Khatri-Rao product).
[a] Apache Spark is a unified data analytics framework that supports distributed storage and large-scale processing: https://spark.apache.org/.
[b] PARAFAC2 is a flexible variant of CP [67]. While the classical CP model requires the tensor factors to be the same for all tensor slices, PARAFAC2 only requires their cross product to be the same and these factors can be different in size slice by slice.
[c] Holt-Winters is an effective time series forecasting procedure [68].

singular vector of the reshaped matrix from $\mathbf{H}_t(:,i)$ can provide a good estimate of $\mathbf{U}_t^{(1)}(:,i)$ and $\mathbf{U}_t^{(2)}(:,i)$, respectively, i.e.,

- $\left[\mathbf{b}_i, \lambda_i, \mathbf{a}_i\right] \leftarrow \text{SVD}\left(\text{reshape}(\mathbf{H}_t(:,i), [I_2\ I_1])\right)$
- $\mathbf{U}_t^{(1)}(:,i) \leftarrow \mathbf{a}_i^*$ and $\mathbf{U}_t^{(2)}(:,i) \leftarrow \lambda_i \mathbf{b}_i$

Computation of SVD may be expensive when dealing with large-scale streaming tensors, we can use the alternative update based on power iteration as follows

- $\mathbf{H}_t^{(i)} \leftarrow \text{reshape}\left(\mathbf{H}_t(:,i), [J \times I]\right)$
- $\mathbf{U}_t^{(1)}(:,i) \leftarrow (\mathbf{H}_t^{(i)})^\top \mathbf{U}_{t-1}^{(2)}(:,i)$
- $\mathbf{U}_t^{(2)}(:,i) \leftarrow \dfrac{\mathbf{H}_t^{(i)}\mathbf{U}_t^{(1)}(:,i)}{\left\|\mathbf{H}_t^{(i)}\mathbf{U}_t^{(1)}(:,i)\right\|}$.

As these algorithms are only designed for tracking third-order streaming tensors, there are still rooms to develop subspace-based methods capable of handling $N \geq 4$.

4.2 Block-Coordinate Descent

The second approach is based on the block-coordinate descent (BCD) framework in which we decompose the main optimization into two main stages at each time t: (i) estimate the temporal factor $\mathbf{U}_t^{(N)}$ given $\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}$, and (ii) update the non-temporal factor $\mathbf{U}_t^{(n)}$ with $1 \leq n \leq N-1$ in sequential or parallel given $\mathbf{U}_t^{(N)}$ and the remaining factors. Many tracking algorithms adopt this approach for estimating the low-rank approximation of streaming tensors over time in the literature. We can list here some: TeCPSGD [48], OLCP [49], OLSTEC [56], CP-stream [54], SPADE [59], SOFIA [61], iCP-AM [70], ACP [64], and RACP [65]. In what follows, we review their strategy in each stage.

In stage 1, the general formulation of the optimization to estimate the last row $\mathbf{u}_t^{(N)}$ of $\mathbf{U}_t^{(N)}$ can be given by

$$\min_{\mathbf{u}^{(N)},\boldsymbol{\mathcal{O}}} \left\|\boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{O}} - [\![\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}^{(N)}]\!]\right)\right\|_F^2 + \rho_{\text{u}}\|\mathbf{u}^{(N)}\|_2^2 + \rho_{\text{o}}\|\boldsymbol{\mathcal{O}}\|_1, \tag{12}$$

where $\rho_{\text{u}}\|\mathbf{u}\|_2^2$ is for avoiding the ill-posed computation and $\rho_{\text{o}}\|\boldsymbol{\mathcal{O}}\|_1$ promotes the sparsity in $\boldsymbol{\mathcal{O}}$. Then, the temporal factor $\mathbf{U}_t^{(N)}$ is obtained by appending the recent updated $\mathbf{u}_t^{(N)}$ to the

old estimate $\mathbf{U}_{t-1}^{(N)}$. Most of the existing BCD-based tracking algorithms suppose that observations are outlier-free (i.e., without $\boldsymbol{\mathcal{O}}$), and hence, they apply the regularized/randomized least-squares methods for solving (12). In the presence of sparse outliers, (12) can be effectively minimized by ADMM or shrinkage-thresholding solvers, as presented in SOFIA [61] and RACP [65].

In stage 2, the non-temporal factors $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$ can be derived from solving the following optimization

$$\min_{\mathbf{U}^{(n)}} \sum_{\tau=1}^{t} \beta^{t-\tau} \Big\| \underline{\mathbf{P}}_{\tau}^{(n)} \circledast \Big(\mathbf{U}^{(n)} \big(\mathbf{W}_{\tau}^{(n)}\big)^{\top} + \underline{\mathbf{O}}_{\tau}^{(n)} - \underline{\mathbf{Y}}_{\tau}^{(n)} \Big) \Big\|_F^2 + \rho_{\text{U}} \mathcal{R}_{\text{U}}\big(\mathbf{U}^{(n)}\big), \tag{13}$$

where $\rho_{\text{U}}\mathcal{R}_{\text{U}}(.)$ is a regularization term on $\mathbf{U}^{(n)}$ and

$$\mathbf{W}_{\tau}^{(n)} = \begin{cases} \Big(\bigodot_{i=1, i\neq n}^{N-1} \mathbf{U}_{t-1}^{(i)} \Big) \odot \mathbf{u}_{\tau}^{\top} & \text{[Jacobi]}, \\ \Big(\bigodot_{i=1}^{n-1} \mathbf{U}_{t}^{(i)} \Big) \odot \Big(\bigodot_{i=n+1}^{N-1} \mathbf{U}_{t-1}^{(i)} \Big) \odot \mathbf{u}_{\tau}^{\top} & \text{[Gauss-Seidel]}. \end{cases}$$

Here, we can apply one of the two iterative schemes to update $\mathbf{U}_t^{(n)}$: the Jacobi scheme supports the parallel and/or distributed processing while the Gauss-Seidel scheme is useful for a sequential (serial) one. The regularization can be $\|\mathbf{U}^{(n)}\|_F^2$ for smoothness, $\|\mathbf{U}^{(n)} - \mathbf{U}_{t-1}^{(n)}\|_F^2$ for slow time-variation, or $\mathbf{U}^{(n)} \succeq \mathbf{0}$ for non-negativity constraints. Next, we review two common types of solver for optimizing (13): adaptive least-squares filters and stochastic gradient solvers.

a) Adaptive Least-Squares (LS) Filters. We can see that the first term of (13) is of a weighted LS form very common in adaptive filtering while the second one is to regularize the estimators. Accordingly, (13) can be effectively minimized by adaptive LS filters in general and recursive least-squares (RLS) filters in particular.

In [56], Kasai proposed an exponential RLS algorithm called OLSTEC to minimize (13) when the observations are outlier-free. OLSTEC is, however, designed for third-order streaming tensors only and its complexities are relatively high compared to other algorithms. Thanh *et al.* in [64] proposed another RLS algorithm called ACP which is capable of dealing with big streaming tensors of higher order ($N \geq 4$). ACP is fast and requires much lower complexity than OLSTEC. Very recently, the same authors in [65] proposed a sliding-window version of ACP robust to both sparse outliers and missing data, namely RACP. Interestingly, three algorithms belong to the class of provable online CP algorithms in which their convergence is guaranteed under certain conditions. In [51], Vandecappelle *et al.* introduced a nonlinear least-squares (NLS) algorithm for computing the streaming CP decomposition of third-order tensors. In particular, the authors recast the objective function of (13) into a truncated exponential window one by incorporating a weighting matrix $\mathbf{L} = \text{diag}\big([0, \ldots, 0, \beta^{L-1}, \beta^{L-2}, \ldots, \beta, 1]\big)$ and applied a NLS solver to track the tensor factors with time. Following the same line, Smith *et al.* in [54] proposed another online CP algorithm called CP-stream. This algorithm has the potential to factorize higher-order streaming tensors as well as support constraints on the CP tracking such as smoothness and nonnegativity.

b) Stochastic Gradient Solvers. Instead of optimizing (13) directly, we can minimize its t-th summand:

$$\min_{\mathbf{U}^{(n)}} \Big\| \underline{\mathbf{P}}_{t}^{(n)} \circledast \Big(\underline{\mathbf{Y}}_{t}^{(n)} - \underline{\mathbf{O}}_{t}^{(n)} - \mathbf{U}^{(n)} \big(\mathbf{W}_{t}^{(n)}\big)^{\top} \Big) \Big\|_F^2 + \rho_{\text{U}} \mathcal{R}_{\text{U}}\big(\mathbf{U}^{(n)}\big). \tag{14}$$

Three algorithms TeCPSGD [48], OLCP [49], and SOFIA [61] adopt this replacement for tracking tensor factors with time. The main difference among them is the type of $\mathcal{R}_{\text{U}}(.)$. Besides, they obtain different forms of update:

$$\text{[SOFIA]}: \ \mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \gamma_t \boldsymbol{\Delta}\mathbf{U}_t^{(n)}, \tag{15}$$

$$\text{[TeCPSGD]}: \ \mathbf{U}_t^{(n)} = \Big(1 - \frac{\beta_t}{t\eta_t}\Big)\mathbf{U}_{t-1}^{(n)} + \frac{1}{\eta_t}\Delta\mathbf{U}_t^{(n)}, \tag{16}$$

$$\text{[OLCP]}: \ \mathbf{U}_t^{(n)} = \mathbf{P}_t^{(n)} \big(\mathbf{Q}_t^{(n)}\big)^{-1} \text{ with } \mathbf{P}_t^{(n)} = \mathbf{P}_{t-1}^{(n)} + \boldsymbol{\Delta}\mathbf{P}_t^{(n)} \text{ and } \mathbf{Q}_t^{(n)} = \mathbf{Q}_{t-1}^{(n)} + \boldsymbol{\Delta}\mathbf{Q}_t^{(n)}. \tag{17}$$

Here, γ_t, η_t, $\boldsymbol{\Delta}\mathbf{U}_t^{(n)}$, $\boldsymbol{\Delta}\mathbf{P}_t^{(n)}$, and $\boldsymbol{\Delta}\mathbf{Q}_t^{(n)}$ can be obtained from $\{\mathbf{U}_{t-1}^{(m)}\}_{m=1}^{N-1}$ and the error tensor $\boldsymbol{\Delta}\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast (\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{O}}_t - [\![\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_t^{(N)}]\!])$. It is worth noting that SOFIA is capable of dealing with sparse corruptions. TeCPSGD has the ability to track tensors from missing observations, while OLCP can handle streaming tensors of order greater than 3.

In [70], Zeng *et al.* proposed an incremental ALS algorithm called iCP-AM to minimize a reinforced version of (14) which is defined as

$$\min_{\mathbf{U}^{(n)}} \Bigg\| \begin{bmatrix} \underline{\mathbf{Y}}_t^{(n)} & \mathbf{U}_{t-1}^{(n)} \Big(\mathbf{U}_{t-1}^{(N)} \odot \mathbf{V}_{t-1}^{(n)} \Big)^{\top} \end{bmatrix} - \mathbf{U}^{(n)} \Bigg(\begin{bmatrix} \mathbf{u}_t^{(N)} \\ \bar{\mathbf{U}}_t^{(n)} \end{bmatrix} \odot \mathbf{V}_t^{(n)} \Bigg)^{\top} \Bigg\|_F^2, \tag{18}$$

where $\mathbf{V}_{\tau}^{(n)} = \big(\bigodot_{i=1}^{n-1} \mathbf{U}_{\tau}^{(i)}\big) \odot \big(\bigodot_{i=n+1}^{N-1} \mathbf{U}_{\tau}^{(i)}\big)$. An appealing feature of iCP-AM against other online CP algorithms is that it has a strategy to deal with the variation of the CP rank over time, i.e., to change the number of low-rank components throughout the tracking process.

In parallel, Gujral *et al.* in [59] proposed an online algorithm called SPADE for tracking tensors under the PARAFAC2 format. Specifically, SPADE tracks a fixed (non-temporal) factor along one mode and allows the other tensor factors (modes) to vary with time. Thanks to its stochastic design, SPADE is fast and memory-efficient. However, the stationary assumption that time variation or concept drift is not allowed limits its applicability.

4.3 Bayesian Inference

Besides, another good approach for dealing with the problem of streaming CP decomposition is Bayesian inference. The state-of-the-art Bayesian-based streaming CP decomposition algorithms are POST [55], BRST [52], and SBDT [71]. In general, three algorithms start with a prior distribution of unknown parameters and then infer a posterior that best approximates the joint distribution of these parameters on the arrival of new streaming data. The estimated posterior is then used as the prior for the next update. In this subsection, we briefly describe the two online Bayesian inference frameworks which were already used for tensor tracking: (i) streaming variational Bayes (SVB) and (ii) assumed-density filtering (ADF). Also, prior distributions of parameters of interest are reviewed.

a) Streaming Variational Bayes. The two former algorithms POST and BRST adopted the SVB framework [72] which is based on the following Bayes' rule:

$$p\Big(\boldsymbol{\Theta} | \boldsymbol{\mathcal{X}}_{t-1} \boxplus_N \boldsymbol{\mathcal{Y}}_t\Big) \propto p(\boldsymbol{\mathcal{Y}}_t | \boldsymbol{\Theta}) p(\boldsymbol{\Theta} | \boldsymbol{\mathcal{X}}_{t-1}), \tag{19}$$

where $\boldsymbol{\Theta}$ denotes the parameters of interest, e.g., tensor factors, CP rank, and other parameters. On the arrival of $\boldsymbol{\mathcal{Y}}_t$, SVB first uses the current posterior $q_{t-1}(\boldsymbol{\Theta}) := p(\boldsymbol{\Theta} | \boldsymbol{\mathcal{X}}_{t-1})$ as the prior of $\boldsymbol{\Theta}$, and then integrates with the likelihood of $\boldsymbol{\mathcal{Y}}_t$ to obtain

$$\tilde{p}_t(\boldsymbol{\Theta}) = p(\boldsymbol{\mathcal{Y}}_t | \boldsymbol{\Theta}) q_{t-1}(\boldsymbol{\Theta}), \tag{20}$$

which can be served as an approximation of the joint distribution $p(\boldsymbol{\Theta}, \boldsymbol{\mathcal{Y}}_t)$ up to a scale factor. The variational posterior $q_t(\boldsymbol{\Theta})$ is derived from maximizing the variational model evidence lower bound (ELBO) $\mathcal{L}(q(\boldsymbol{\Theta})) = \mathbb{E}_q\big[\log\big(\tilde{p}_t(\boldsymbol{\Theta})/q(\boldsymbol{\Theta})\big)\big]$ which is equivalent to minimizing the Kullback-Leibler (KL) divergence:

$$\min_q \left[\mathrm{KL}\left(q(\boldsymbol{\Theta})\|\tilde{p}_t(\boldsymbol{\Theta})\right) = \int q(\boldsymbol{\Theta})\log\left\{\frac{q(\boldsymbol{\Theta})}{\tilde{p}_t(\boldsymbol{\Theta})}\right\}d\boldsymbol{\Theta}\right]. \quad (21)$$

The optimized form of $q_t(\boldsymbol{\Theta}_i)$ of (21) can be given by

$$\log q_t(\boldsymbol{\Theta}_i) = \mathbb{E}_{q(\boldsymbol{\Theta}/\boldsymbol{\Theta}_i)}\big[\log \tilde{p}_t(\boldsymbol{\Theta})\big] + \text{const}, \quad (22)$$

where $\mathbb{E}_{q(\boldsymbol{\Theta}/\boldsymbol{\Theta}_i)}[.]$ is an expectation w.r.t. q over all but $\boldsymbol{\Theta}_i$.

b) Assumed-Density Filtering. The latter algorithm, SBDT, applied the ADF framework to infer the posterior distribution $q_t(\boldsymbol{\Theta})$ over time. Particularly, ADF is an incremental learning framework that allows for computing the approximate posteriors in Bayesian inference for stochastic processes [73]. The ADF framework is also grounded on the Bayes' rule (19) but utilizes a distribution from the exponential family (e.g., Gaussian distribution) to approximate the current posterior. Instead of minimizing the KL divergence or maximizing the variational ELBO like SVB, ADF projects $\tilde{p}_t(\boldsymbol{\Theta})$ into the selected distribution through moment matching to obtain $q_t(\boldsymbol{\Theta})$.

c) Prior Distributions over $\boldsymbol{\Theta}$. We list common prior distributions over $\boldsymbol{\Theta}$ which were used by POST, BRST, and SBDT.

Prior distribution of tensor factors: All three algorithms assume that the prior over tensor factors is derived from the following Gaussian distribution which is controlled by the hyperparameter $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_r]$:

$$p(\mathbf{U}^{(n)}|\boldsymbol{\lambda}) = \prod_{i=1}^{I_n}\mathcal{N}(\mathbf{u}_i^{(n)}|\mathbf{0}, \boldsymbol{\Lambda}^{-1}), \forall n \in [1, N], \quad (23)$$

where $\mathbf{u}_i^{(n)}$ is the i-th row of $\mathbf{U}^{(n)}$ and $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$ denotes the inverse covariance matrix. Here, $\boldsymbol{\lambda}$ is supposed to follow a Gamma distribution:

$$p(\boldsymbol{\lambda}) = \prod_{j=1}^{r}\mathrm{Gam}\left(\lambda_j|c_j, d_j\right), \quad (24)$$

where $\mathrm{Gam}\left(\lambda_j|c_j, d_j\right) = \frac{d_j{}^{c_j}}{\Gamma(c_j)}\lambda_j{}^{c_j-1}e^{-d_j\lambda_j}$ with $\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}dx$. Specifically, the mean and variance of $\mathrm{Gam}(\lambda_j|c_j, d_j)$ are, respectively, c_j/d_j and c_j/d_j^2 which aim to control the magnitude of $\boldsymbol{\lambda}$.

Prior distribution of noises: The noise is often assumed to be Gaussian, i.e., $\boldsymbol{\mathcal{N}}_t \sim \prod_{i_1\ldots i_N}\mathcal{N}(0, \tau^{-1})$ with a noise precision $\tau > 0$. The parameter τ is further assigned to another Gamma distribution $p(\tau|a, b) = \mathrm{Gam}\left(\tau|a, b\right)$ in the same way as for $\boldsymbol{\lambda}$.

Prior distribution of sparse components: Only BRST in [52] has the ability to handle sparse outliers. Here, BRST places a Gaussian prior distribution over the sparse $\boldsymbol{\mathcal{O}}_t$ as

$$p(\boldsymbol{\mathcal{O}}_t|\boldsymbol{\gamma}) = \prod_{i_1}^{I_1}\cdots\prod_{i_N}^{I_N}\mathcal{N}\Big(\left[\boldsymbol{\mathcal{O}}_t\right]_{i_1\ldots i_N}\Big|0, \gamma_{i_1\ldots i_N}^{-1}\Big)^{[\boldsymbol{\mathcal{P}}_t]_{i_1\ldots i_N}}. \quad (25)$$

where $\boldsymbol{\gamma}$ is the sparsity precision parameter. If the value of $\gamma_{i_1\ldots i_N}$ is large, the corresponding entry in $\boldsymbol{\mathcal{O}}_t$ is likely to have a small magnitude. By controlling the value of $\gamma_{i_1\ldots i_N}$, we can control the sparsity of $\boldsymbol{\mathcal{O}}_t$.

Prior distribution of NN's weights: SBDT in [71] incorporates neural networks (NN) into tensor factorization. SBDT assigns a spike-and-slab prior distribution over NN weights to sparsify the network. Each weight $\omega_{mjt} = [\mathbf{W}_m]_{jt}$ of NN is particularly sampled from

$$p(\omega_{mjt}|s_{mjt}) = s_{mjt}\mathcal{N}(\omega_{mjt}|0, \sigma_0^2) + (1 - s_{mjt})\delta(\omega_{mjt}), \quad (26)$$

where $\delta(.)$ denotes the delta function and the binary selection indicator s_{mjt} is derived from $p(s_{mjt}) = \mathrm{Bern}(s_{mjt}|\rho_0) = \rho_0^{s_{mjt}}(1-\rho_0)^{1-s_{mjt}}$.

TABLE 4
Main Features of Multi-aspect Streaming CP Decomposition Algorithms.

Algorithm	MAST (2017 [74])	OR-MSTC (2019 [75])	InParTen2 (2020 [76])	DisMASTD (2021 [77])
Missing?	✓	✓	✗	✗
Outliers?	✗	✓	✗	✗
High-order? ($N \geq 4$)	✓	✓	✗	✓
Distributed?	✗	✗	✓	✓

4.4 Multi-aspect Streaming CP Decomposition

In the literature, there are some algorithms capable of tracking multi-aspect streaming tensors under the CP format, such as MAST [74], OR-MSTC [75], InParTen2 [76], and DisMASTD [77]. We refer the readers to Tab. 4 for their key features. In what follows, we first describe the main dynamic tensor decomposition (DTD) framework shared by most of these algorithms and then highlight their characteristics in the following text.

For ease of reference, we denote by $\boldsymbol{\mathcal{X}}_{t-1} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ and $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{(I_1+d_1)\times\cdots\times(I_N+d_N)}$ the two successive snapshots at $t-1$ and t, see Fig. 5 for an illustration (when $\boldsymbol{\mathcal{G}}_t = \boldsymbol{\mathcal{I}}$). At time t, given $\boldsymbol{\mathcal{X}}_t$ and the old estimates $\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^N$ of $\boldsymbol{\mathcal{X}}_{t-1}$, we wish to update $\{\mathbf{U}_t^{(n)}\}_{n=1}^N$ such that $\boldsymbol{\mathcal{X}}_t \approx [\![\{\mathbf{U}_t^{(n)}\}_{n=1}^N]\!]$.

The DTD introduced in [74] offers an online framework for the problem of multi-aspect streaming CP decomposition. Particularly, DTD relaxes the CP representation of $\boldsymbol{\mathcal{X}}_t$ in the sense that if $\boldsymbol{\mathcal{X}}_t$ is expressed by $[\![\{\mathbf{U}_t^{(n)}\}_{n=1}^N]\!]$, then its sub-tensor $\boldsymbol{\mathcal{X}}_{t-1}$ can be approximated by $[\![\{\bar{\mathbf{U}}_t^{(n)}\}_{n=1}^N]\!]$ where $\bar{\mathbf{U}}_t^{(n)} \in \mathbb{R}^{I_n\times r}$ is the sub-matrix of $\mathbf{U}_t^{(n)} \in \mathbb{R}^{(I_n+d)\times r}$. Accordingly, DTD enables us to divide $\boldsymbol{\mathcal{X}}_t$ into two parts $\boldsymbol{\mathcal{X}}_{t-1}$ and $\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{X}}_t\backslash\boldsymbol{\mathcal{X}}_{t-1}$ in order to take advantages of old estimates. We can first update $\bar{\mathbf{U}}_t^{(n)}$ incrementally from $\mathbf{U}_{t-1}^{(n)}$ with a low cost and then estimate the remaining part $\hat{\mathbf{U}}_t^{(n)} \in \mathbb{R}^{d\times r}$ of $\mathbf{U}_t^{(n)}$. The tensor factors are particularly derived from

$$\min_{\{\mathbf{U}^{(n)}\}_{n=1}^N}\left[\ell\Big(\boldsymbol{\mathcal{Y}}_t, \{\mathbf{U}^{(n)}\}_{n=1}^N\Big) + \rho\Big(\sum_{n=1}^N\|\mathbf{U}^{(n)}\|_*\Big)\right], \quad (27)$$

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

$$\ell\Big(\boldsymbol{\mathcal{Y}}_t, \{\mathbf{U}^{(n)}\}_{n=1}^N\Big) = \mu\Big\|[\![\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^N]\!] - [\![\{\bar{\mathbf{U}}^{(n)}\}_{n=1}^N]\!]\Big\|_F^2 + \Big\|\mathcal{P}_{\Omega_t}(\boldsymbol{\mathcal{Y}}_t) - \mathcal{P}_{\Omega_t}\Big([\![\{\mathbf{U}^{(n)}\}_{n=1}^N]\!]\Big)\Big\|_F^2. \quad (28)$$

Here, Ω_t denotes the set of observed entries and $\mu, \rho > 0$ are two regularized parameters. Depending on the type of constraints, additional information imposed and the method of optimization, we can obtain several types of estimators for tracking multi-aspect streaming tensors with time under the DTD framework.

In [74], Song *et al.* developed the so-called MAST algorithm for tracking multi-aspect streaming tensors. The authors recast (27) into a constrained minimization and then formed the following Lagrangian function

$$\mathcal{L}\Big(\boldsymbol{\mathcal{Y}}_t, \boldsymbol{\Theta}\Big) = \ell\Big(\boldsymbol{\mathcal{Y}}_t, \{\mathbf{U}^{(n)}\}_{n=1}^N\Big) + \sum_{n=1}^N\Big(\rho\|\mathbf{Z}^{(n)}\|_* + \langle\boldsymbol{\Lambda}^{(n)}, \mathbf{Z}^{(n)} - \mathbf{U}^{(n)}\rangle + \frac{\eta}{2}\|\mathbf{Z}^{(n)} - \mathbf{U}^{(n)}\|_F^2\Big), \quad (29)$$

where $\boldsymbol{\Theta} = \{\mathbf{U}^{(n)}, \mathbf{Z}^{(n)}, \boldsymbol{\Lambda}^{(n)}\}_{n=1}^N$ with auxiliary matrices $\{\mathbf{Z}^{(n)}\}_{n=1}^N$ and Lagrange multiplier matrices $\{\boldsymbol{\Lambda}^{(n)}\}_{n=1}^N$, and

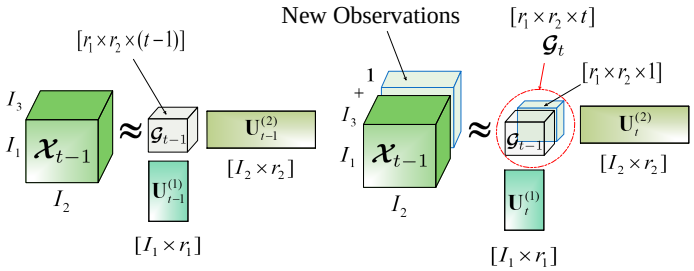


Fig. 3. Online tensor dictionary learning.

$\eta > 0$ is a regularization parameter. Since terms of (29) are all convex, it can be effectively minimized by several methods. In particular, MAST applies an ADMM solver to minimize (29) in order to balance the trade-off between effectiveness and efficiency in tracking process.

Since MAST is not designed for handling sparse outliers, Najafi *et al.* in [75] introduced a robust version of it called OR-MSTC. In the presence of sparse outliers, they proposed to regularize the objective function of (27) by adding an ℓ_1-norm regularization term $\lambda\|\mathcal{O}\|_1$ and replacing $\mathcal{Y}_t$ with $\mathcal{Y}_t - \mathcal{O}$ in the first term of $\ell(.)$ in (29). Because the term $\lambda\|\mathcal{O}\|_1$ is convex, OR-MSTC also adopts the well-known ADMM method in a similar way to MAST.

In [76], Yang *et al.* proposed a distributed version of MAST called InParTen2. Thanks to Apache Spark[1], it can handle large-scale streaming tensors efficiently with a limited memory. However, the use of InParTen2 is limited for 3-order streaming tensors only. In [77], Yang *et al.* introduced another distributed method called DisMASTD capable of dealing with tensors of higher order. One of appealing feature of DisMASTD is that it can avoid repetitive computation and reduce network communication cost.

5 Streaming Tucker Decomposition

We can broadly categorize the streaming Tucker decomposition algorithms into three main classes: (i) online tensor dictionary learning, (ii) tensor subspace tracking, and (iii) multi-aspect streaming Tucker decomposition. Specifically, the first two classes are designed for two specific cases of single-aspect streaming Tucker decompositions, while the latter class is for multi-aspect streaming tensors.

5.1 Online Tensor Dictionary Learning

In the class of online tensor dictionary learning methods, we are particularly interested in a specific case of single-aspect streaming Tucker decomposition where the underlying tensor $\mathcal{X}_T \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times T}$ – which represents a set of T data streams $\{\mathcal{Y}_t\}_{t=1}^T$ of the same size $I_1 \times I_2 \times \cdots \times I_{N-1}$ – is supposed to be modelled by

$$\mathcal{X}_T = \left[\!\left[\mathcal{G}_T; \{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{I}_T\right]\!\right], \tag{30}$$

where the core $\mathcal{G}_T$ is of size $r_1 \times \cdots \times r_{N-1} \times T$ (i.e., $r_N = T$), the tensor factors $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}$ are of fixed size, and the last factor $\mathbf{U}^{(N)}$ is an identify matrix. Specifically, the t-th temporal slice $\mathcal{Y}_t$ of $\mathcal{X}_T$ is expressed as

$$\mathcal{Y}_t = \left[\!\left[\mathcal{G}_t; \{\mathbf{U}^{(n)}\}_{n=1}^{N-1}\right]\!\right], t = 1, 2, \ldots, T, \tag{31}$$

where $\mathcal{G}_t \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_{N-1}}$ is the t-th slice of the core tensor $\mathcal{G}_T$. The primary objective here is to estimate $\mathcal{G}_t$ and incrementally update $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ on the arrival of $\mathcal{Y}_t$ at each time t. In what follows, we review two main approaches to handle this problem.

1. Apache Spark: https://spark.apache.org/

a) Incremental Subspace Learning on Tensor Unfolding Matrices. A natural and very first approach for streaming Tucker decomposition is to incrementally update the subspaces covering unfolding matrices of the underlying tensor. The central idea of this approach stems from the fact that the n-th tensor factor $\mathbf{U}_t^{(n)}$ which is derived from the standard HOSVD is given by

$$\mathbf{U}_t^{(n)} = \text{EVD}\left(\left[\underline{\mathbf{X}}_{t-1}^{(n)}, \underline{\mathbf{Y}}_t^{(n)}\right]\left[\underline{\mathbf{X}}_{t-1}^{(n)}, \underline{\mathbf{Y}}_t^{(n)}\right]^\top\right), \tag{32}$$

where $\underline{\mathbf{X}}_{t-1}^{(n)} = \left[\underline{\mathbf{Y}}_1^{(n)}, \ldots, \underline{\mathbf{Y}}_{t-1}^{(n)}\right]$ with $\underline{\mathbf{Y}}_\tau^{(n)}$ is the mode-n unfolding matrix of $\mathcal{Y}_\tau$. Accordingly at time t, we can apply the following dynamic tensor analysis (DTA) framework introduced in [78], [79] to estimate $\mathcal{G}_t$ and update $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$:

$$\mathbf{C}_t^{(n)} \leftarrow \beta\mathbf{C}_{t-1}^{(n)} + \left(\underline{\mathbf{Y}}_t^{(n)}\right)^\top \underline{\mathbf{Y}}_t^{(n)}, \tag{33a}$$

$$\mathbf{U}_t^{(n)} \leftarrow \text{eig}\left(\mathbf{C}_t^{(n)}, r\right), \tag{33b}$$

$$\mathcal{G}_t \leftarrow \left[\!\left[\mathcal{Y}_t, \{(\mathbf{U}_t^{(n)})^\top\}_{n=1}^{N-1}\right]\!\right], \tag{33c}$$

where $0 < \beta \leq 1$ is a forgetting factor and $\text{eig}(\mathbf{C}_t^{(n)}, r)$ computes the top r principal eigenvectors of $\mathbf{C}_t^{(n)}$. Since the two steps (33a) and (33b) are generally expensive, there have been some studies offering good modifications or fast alternatives for (33).

In [78], [79], Sun *et al.* proposed a streaming tensor analysis (STA) algorithm for tracking $\mathbf{U}_t^{(n)}$ with time, instead of taking the orthonormal step (33b) directly. Particularly on the arrival of $\mathcal{Y}_t$, STA first divides its unfolding matrix $\underline{\mathbf{Y}}_t^{(n)}$ into column vectors $\{\mathbf{y}_{m,t}^{(n)}\}$ and then performs the following steps on each vector $\mathbf{y}_{m,t}^{(n)}$: (i) projects it onto the subspace $\mathbf{U}_{t-1}^{(n)}$, (ii) evaluates the corresponding residual error and the energy for each entry of $\mathbf{y}_{m,t}^{(n)}$, and (iii) updates the matrix $\mathbf{U}_t^{(n)}$. Intuitively, the larger the residual error is, the more $\mathbf{U}_t^{(n)}$ is updated. The computational complexity of STA is moderate while its effectiveness was demonstrated with the problem of anomaly detection and multi-way latent semantic indexing.

In [80], [81], Hu *et al.* introduced the so-called IRTSA algorithm to track the dominant subspaces $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$. Specifically, instead of computing (33a), IRTSA applies a fast incremental SVD (ISVD) proposed by Ross *et al.* in [100] on the mode-n unfolding matrix $\underline{\mathbf{X}}_t^{(n)} = \left[\underline{\mathbf{X}}_{t-1}^{(n)}, \underline{\mathbf{Y}}_t^{(n)}\right]$ in (32). Thanks to ISVD, IRTSA shares the same order of computational complexity with STA while offers a better estimation than STA for the problem of background modelling and object tracking. Although the current version of IRTSA is designed for factorizing three-order streaming tensors, it is not difficult to extend IRTSA for dealing with higher-order tensors. Besides, a modified version of IRTSA was introduced by Zang *et al.* in [82] for the problem of web service recommendation.

In [83], Kuang *et al.* also proposed an incremental SVD-based streaming Tucker decomposition, namely IHOSVD. In particular, this algorithm performs the following three processes in a serial manner: (i) applies a recursive SVD method to compute singular values and singular vectors of unfolding matrices of the new tensor, (ii) merges the new results with the old estimations from past observations, and (iii) obtains the core tensor with n-mode products. Theoretical analyses and experimental results on transportation applications demonstrate the use of IHOSVD.

In [101], Li *et al.* modified slightly the recursive update of the covariance matrix $\mathbf{C}_t^{(n)}$ in (33a) as follows

$$\mathbf{C}_t^{(n)} = (1-\alpha)\mathbf{C}_{t-1}^{(n)} + \alpha\left(\underline{\mathbf{Y}}_t^{(n)}\right)^\top \underline{\mathbf{Y}}_t^{(n)}, \tag{34}$$

with a weight $0 < \alpha \leq 1$ and then introduced a robust incremental algorithm called RTSL which has the potential to model

TABLE 5
Main Features of the State-of-the-art Streaming Tucker Decomposition Algorithms.

Algorithm	Missing Data?	Sparse Outliers?	High-order ($N \geq 4$)?	Convergence Guarantee?	Computational Complexity	Additional Information (approaches + supports)
STA [78], [79]	✗	✗	✓	✗	$\mathcal{O}\big((N-1)rI^{N-1}\big)$	- Subspace tracking + deflation
IRTSA [80], [81]	✗	✗	✗	✗	$\mathcal{O}\big(3rI^3\big)$ (with $N=3$)	- ISVD-based tracking
ITF [82]	✗	✗	✗	✗	$\mathcal{O}\big(3rI^3\big)$ (with $N=3$)	- ISVD-based tracking
IHOSVD [83]	✗	✗	✓	✗	$\mathcal{O}\big(Nr^2I^N\big)$	- Adopts recursive matrix SVD
ALTO [84]	✗	✗	✗	✓	$\mathcal{O}\big(3(r+k)^6I^3\big)$ k: no. of randomly selected columns	- Adds noise perturbation - Uses tensor sequential mapping
LRUT [85]	✗	✗	✓	✗	$\mathcal{O}\big(N(r+k)^{2N}I^N\big)$ k: no. of randomly selected columns	- Adds noise perturbation - Supports parallel computing
Riemannian-based Tucker [86]	✓	✗	✗	✗	unavailable	- Computes SGD on Riemannian manifold
HO-RLSL [87]	✗	✓	✓	✗	$3I^2\mathcal{O}\big(I^3\big)$	- For $N=4$ only
IHOSVD [88]	✗	✗	✓	✗	$\mathcal{O}\big(N(I/d)^{2(N-1)}\big)$ d: number of cores	- Supports distributed computing - Adopts RoundRobin process + columnwise Jacobi-rotation
MIHOSVD [89]	✗	✗	✓	✗	$\mathcal{O}\big(N(I/d)^{2(N-1)}\big)$ d: number of cores	- Supports distributed computing - Adopts tree-based integration + columnwise Jacobi-rotation
SIITA [90]	✓	✗	✓	✗	$\mathcal{O}\big(K(r^N\|\Omega\| + NIMr)\big)$ K, M: no. of iterations and columns of side information matrices	- Multi-aspect streaming method - Supports side information + nonnegativity + sparsity
eOTD [91]	✗	✗	✓	✗	$\mathcal{O}\big(rd^{2(m-1)}I^{2(N-m)}\big)$ d: no. of coming temporal slices m: no. of temporal modes	- Multi-aspect streaming method - Adopts SGD + MGS + block tensor matrix multiplications
OTL [92]	✗	✗	✓	✓	$\mathcal{O}\big(d(N-1)(Ir^2)^{N-1}\big)$ d: no. of coming temporal slices	- Promotes sparse coding - Supports nonnegativity + orthogonality
Singleshot [93]	✗	✗	✓	✓	$\mathcal{O}\big(pNr^NI^{N-1} + Nr^{2N}\big)$ p: dimensionality of new coming tensor	- Uses tensor sketching - Supports multiple coming temporal slices + nonnegativity
TTMTS [94]	✗	✗	✓	✓	$\mathcal{O}\big((Nk+d)I^N\big)$ $d = (s(1-(s/I)^N)/(1-s/I)$ k, s: parameters of random projection	- Uses tensor random projection - Supports one/two-pass approximations
SNBTD [95]	✗	✗	✓	✓	$\mathcal{O}(I^{N-1}(NIr + MR + 4M^2))$ M: no. of pseudo inputs[a] R: size of the pseudo input	- Nonlinear decomposition with Fourier features - Uses Bayesian inference + ADF
D-L1-Tucker [96]	✗	✓	✓	✗	$\mathcal{O}\big(K(rI^{N-1} + I^2r^{N-1})\big)$ K: no. of iterations	- Applies threshold-based outlier detection + L1-HOOI
BASS-Tucker [97]	✗	✗	✓	✗	$\mathcal{O}\big(r^{3(N-1)} + (Ir)^{N-1} + Nr^3I^{N-1}\big)$	- Sparse decomposition - Uses Bayesian inference + ADF
SBDT [71]	✗	✗	✓	✗	$\mathcal{O}\big(NIr + KI^{N-1}\big)$ K: no. of weights in NNs	- Uses Bayesian inference + ADF - Incorporates NNs
Zoom-Tucker [98]	✗	✗	✓	✗	$\mathcal{O}\big(KBNrI^{N-1} + KN^2(r^{N+1} + r^2I)\big)$ K, B: no. of iterations and blocks	- Supports multiple coming temporal slices - Requires a preprocessing phase
RI/BK-NTD [99]	✗	✗	✓	✗	$\mathcal{O}\big(KN(Ir)^N\big)$ K: no. of iterations	- Nonnegative decomposition - Uses NNLS + BCD
ATD [64]	✓	✗	✓	✓	$\mathcal{O}\big(r\|\Omega\| + r^2(I^{N-2} + \|\mathcal{S}_1\|) + r^{2N}\|\mathcal{S}_2\|\big)$ $\|\mathcal{S}_1\|, \|\mathcal{S}_2\|$: size of the sampling sets	- Uses BCD + Randomized sampling - Supports parallel computing

[−] Suppose that $I_1 = I_2 = \cdots = I_N = I$, $r_1 = r_2 = \cdots = r_N = r$, and $|\Omega|$ is the number of observed elements.
[−] Abbreviations: ISVD, (incremental SVD), SGD (stochastic gradient descent), MGS (modified Gram-Schmidt process), BCD (block-coordinate descent), ADF (assumed-density filtering), NN (neural network), and NNLS (nonnegative constrained least-squares solver).
[a] Pseudo inputs: a small active pseudo set, which is not necessarily required to be a subset of the real data, is introduced to break the dependencies between outputs and hence avoid the explicit computation of the full covariance matrix.

background and detect anomalies in applications of computer vision. Since RTSL still applies directly the DTA framework, its complexity is relatively high. Thus, it may become inefficient for handling large-scale and high dimensional streaming data.

Some other algorithms for streaming Tucker decomposition belonging to this group were presented in [87]–[89], [102], focusing on specific applications such as dynamic brain analysis, smart city services, cyber-physical-social networks and systems.

b) Online Multimodal Dictionary Learning. Another good strategy for the problem of single-aspect tensor tracking is to apply online multimodal dictionary learning (OMDL) techniques. As OMDL is a stochastic version of the multimodal dictionary (multilinear subspace) learning [103], it allows estimating dictionaries (i.e., tensor factors) with one-pass processing. In the literature, there exist some algorithms applying OMDL for tracking the low multilinear-rank component of streaming tensors with time, such as OTDL [92], ODL [104], ORLTM [105], OLRTR [106], D-L1-Tucker [96], and ROLTD [107].

The two former algorithms OTDL and ODL adopt the typical two-step learning procedure to track the tensor factors over time,

namely (i) tensor coding or inference of coefficients in the core tensor and (ii) dictionary update per each tensor mode.

Step 1: Tensor Coding. When $\boldsymbol{\mathcal{Y}}_t$ is observed, the general formulation of optimization for this step is given by:

$$\min_{\boldsymbol{\mathcal{G}}} \left\|\boldsymbol{\mathcal{Y}}_t - [\![\boldsymbol{\mathcal{G}}; \{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}]\!]\right\|_F^2 + \rho_{\mathrm{G}}\mathcal{R}_{\mathrm{G}}(\boldsymbol{\mathcal{G}}), \tag{35}$$

where $\rho_{\mathrm{G}}\mathcal{R}_{\mathrm{G}}(.)$ is a regularization term on the core tensor $\boldsymbol{\mathcal{G}}$ to promote sparsity or nonnegativity for instance. Since the first term of (35) is differentiable while the second term may admit a proximal operator (e.g., ℓ_p-norm), OTDL, ODL, and ROLTD applied proximal methods to minimize it.

Step 2: Dictionary Update. When $\boldsymbol{\mathcal{G}}_t$ is estimated, the BCD framework can be used to update $\mathbf{U}_t^{(n)}$. Specifically, both algorithms optimize the following minimization:

$$\min_{\mathbf{U}^{(n)}} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\|\boldsymbol{\mathcal{Y}}_\tau - [\![\boldsymbol{\mathcal{G}}_\tau; \{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}]\!]\right\|_F^2 + \rho_{\mathrm{U}}\mathcal{R}_{\mathrm{U}}(\mathbf{U}^{(n)}), \tag{36}$$

with a penalty term $\rho_{\mathrm{U}}\mathcal{R}_{\mathrm{U}}(.)$ on $\mathbf{U}^{(n)}$. Interestingly, (36) can be recast into the standard least-squares cost function which is very common in adaptive filtering theory. Accordingly, OTDL introduced an effective recursive least-squares (RLS) solver to optimize it. Meanwhile, ODL used the stochastic gradient descent method to estimate $\mathbf{U}_t^{(n)}$ with a low cost.

The next two algorithms ORLTM and OLRTR, on the other hand, estimated the tensor factors without the need of tensor coding. In particular, the tensor factor $\mathbf{U}^{(n)}$ is directly derived from the following optimization

$$\min_{\mathbf{U}^{(n)}} \sum_{\tau=1}^{t} \beta^{t-\tau} \ell(\boldsymbol{\mathcal{Y}}_\tau, \mathbf{U}^{(n)}) + \rho_{\mathrm{U}}\mathcal{R}_{\mathrm{U}}(\mathbf{U}^{(n)}), \tag{37}$$

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

$$\ell(\boldsymbol{\mathcal{Y}}_\tau, \mathbf{U}^{(n)}) = \min_{\mathbf{R}^{(n)}, \mathbf{O}^{(n)}} \left\|\underline{\mathbf{Y}}_\tau^{(n)} - \mathbf{U}^{(n)}\mathbf{R}^{(n)} - \mathbf{O}^{(n)}\right\|_F^2 + \lambda_1\left\|\mathbf{O}^{(n)}\right\|_1 + \lambda_2\mathcal{R}_{\mathrm{R}}(\mathbf{R}^{(n)}). \tag{38}$$

Here, $\mathbf{R}^{(n)}$ and $\mathbf{O}^{(n)}$ play the role of the coefficient and the error, respectively. The main difference between ORLTM and OLRTR is the type of $\mathcal{R}_{\mathrm{R}}(.)$ used. Specifically, OLRTR uses the simple Frobenius norm regularization $\mathcal{R}_{\mathrm{R}}(\mathbf{R}^{(n)}) = \|\mathbf{R}^{(n)}\|_F^2$, while ORLTM reinforces $\mathbf{R}^{(n)} = \mathbf{W}^{(n)}\mathbf{Z}^{(n)}$ and then forms $\mathcal{R}_{\mathrm{R}}(\mathbf{R}^{(n)}) = \|\mathbf{W}^{(n)}\|_F^2 + \|\mathbf{Z}^{(n)}\|_F^2$. Intuitively, the minimization (37) may be regarded as a robust version of (36) which aims to deal with sparse corruptions. Also, the minimization (38) is not difficult to solve since its terms are all convex. Hence, both OLRTR and ORLTM applied the RLS method to update $\mathbf{U}_t^{(n)}$ over time.

In [96], Chachlakis *et al.* proposed a streaming Tucker decomposition called D-L1-Tucker for dealing with streaming tensors. D-L1-Tucker shares the same objective function with ORLTM and OLRTR, but adopts a different approach to handle data corruptions. Particularly on the arrival of $\boldsymbol{\mathcal{Y}}_t$, D-L1-Tucker first identifies whether $\boldsymbol{\mathcal{Y}}_t$ is an anomaly or not based on its reliability which is defined as

$$r_t = \left\|[\![\boldsymbol{\mathcal{Y}}_t; \{(\mathbf{U}_{t-1}^{(n)})^\top\}_{n=1}^{N-1}]\!]\right\|_F^2 \left\|\boldsymbol{\mathcal{Y}}_t\right\|_F^{-2}. \tag{39}$$

If $r_t \leq \tau$ where $\tau \in [0, 1]$ is a predefined threshold, $\boldsymbol{\mathcal{Y}}_t$ is labelled as an outlier slice and then it is disregarded. Otherwise, $\boldsymbol{\mathcal{Y}}_t$ is considered as reliable and useful for tracking process. In such a case, D-L1-Tucker appends $\boldsymbol{\mathcal{Y}}_t$ to the memory set $\boldsymbol{\mathcal{Z}}_t = \boldsymbol{\mathcal{Z}}_{t-1} \cup \boldsymbol{\mathcal{Y}}_t$ and then applies the batch L1-HOOI algorithm proposed in [108] for factorizing $\boldsymbol{\mathcal{Z}}_t$ in order to obtain tensor factors. After that, $\boldsymbol{\mathcal{Z}}_t$ is re-updated by removing the oldest measurement for the next processing. D-L1-Tucker requires a good batch initialization and its tracking ability is dependent on the threshold τ and the memory size M to store $\boldsymbol{\mathcal{Z}}_t$.

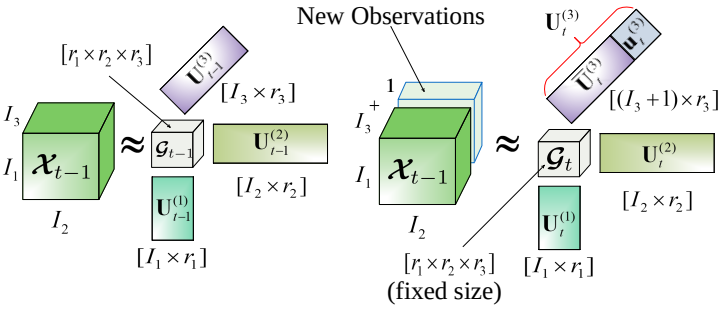


Fig. 4. Online tensor subspace learning.

5.2 Tensor Subspace Tracking

Apart from the model (30), $\boldsymbol{\mathcal{X}}_T \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times T}$ and its t-th temporal slice $\boldsymbol{\mathcal{Y}}_t$ with $1 \leq t \leq T$ can be modelled as follows

$$\boldsymbol{\mathcal{X}}_T = [\![\boldsymbol{\mathcal{G}}; \{\mathbf{U}^{(n)}\}_{n=1}^{N}]\!], \quad \boldsymbol{\mathcal{Y}}_t = [\![\boldsymbol{\mathcal{G}}; \{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_t^{(N)}]\!], \tag{40}$$

where the core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$ and $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ with $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}$ are of fixed size except the last factor $\mathbf{U}^{(N)} \in \mathbb{R}^{T \times r_N}$, and $\mathbf{u}_t^{(N)} \in \mathbb{R}^{1 \times r_N}$ is the t-th row of $\mathbf{U}^{(N)}$, see Fig. 4 for an illustration (when $N = 3$). At time t, given old estimations $\boldsymbol{\mathcal{G}}_{t-1}$ and $\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}$, we are interested in tracking $\boldsymbol{\mathcal{G}}_t$, $\mathbf{u}_t^{(N)}$, and $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$ which can compactly represent the temporal slice $\boldsymbol{\mathcal{Y}}_t$. We refer this problem to as tensor subspace tracking.[2]

It is worth mentioning that single-aspect streaming CP methods also belong to this class as the core tensor $\boldsymbol{\mathcal{G}}$ is constrained to be identity. In the literature, there exist some tensor subspace tracking methods which have the potential to deal with a general case of $\boldsymbol{\mathcal{G}}$. Each method adopts a different strategy to factorize streaming tensors. In what follows, we briefly describe their main features in chronological order.

a) Augmented Projection. In [85], Baskaran *et al.* introduced the so-called LRUT algorithm (which stands for Low-Rank Updates to Tucker decomposition) using a randomized projection technique for tracking the low multilinear-rank approximation of streaming tensors over time. When a data stream arrives, LRUT first projects it onto an extended tensor subspace and then forms an augmented core tensor. Specifically, LRUT adds a few more random dimensions to the current tensor subspace defined by old estimations of the tensor factors. The inclusion of some random vectors here plays a role of noise perturbation aimed to prevent the main optimization from getting stuck in local optima. Next, LRUT performs the standard Tucker decomposition (e.g., batch HOSVD or HOOI) on the resulting augmented core tensor to update tensor factors. In this way, we can avoid the computation of SVD on unfolding matrices of the full tensor which is highly expensive in an online setting. However, its computational complexity is still relatively high since LRUT uses several orthogonalization operations on augmented tensor factors and unfolding matrices of the projected tensor slice.

b) Riemannian Optimization. In [86], Kasai *et al.* developed a Riemannian manifold preconditioning approach for tensor completion. Specifically, its stochastic version can be adapted for

2. This name stems from the following observation: we can recast (40) into the standard vector-matrix form $\mathbf{y}_t = \mathbf{D}\mathbf{u}_t$, where $\mathbf{y}_t = \mathrm{vec}(\boldsymbol{\mathcal{Y}}_t)$, $\mathbf{u}_t = (\mathbf{u}_t^{(N)})^\top$ and $\mathbf{D}$ is the transpose of the mode-N unfolding matrix of $[\![\boldsymbol{\mathcal{G}}; \{\mathbf{U}^{(n)}\}_{n=1}^{N-1}]\!]$. Intuitively, it may be regarded as the data model which is very common and widely used in the problem of subspace tracking where we wish to incrementally update $\mathbf{D}$ on the arrival of $\mathbf{y}_t$ at each time t. Since the subspace matrix $\mathbf{D}$ has a tensor structure, we label this problem as "tensor subspace tracking" without hesitation.

factorizing incomplete streaming tensors in an online fashion. Since the Tucker format provides an effective representation for tensors in the manifold $\mathcal{M}_{\mathbf{r}} = \{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} | \operatorname{rank}(\boldsymbol{\mathcal{X}}) := \mathbf{r} = [r_1, r_2, \ldots, r_N]\}$, Riemannian optimization can offer a good approach for tensor decomposition and completion [109]. Accordingly, the authors proposed an efficient Riemannian gradient based method to estimate the low multilinear-rank component of tensors. The proposed method consists of a rank-one Riemannian gradient computation and a retraction step. Specifically, a novel Riemannian metric on the tangent space of $\mathcal{M}_{\mathbf{r}}$ and its quotient manifold was introduced to enable the Riemannian optimization framework. Furthermore, a map that combines all retractions on the individual manifolds of tensor factors was used to transform the estimations to the tensor manifold.

c) Bayesian Inference. In [97], Fang *et al.* introduced a Bayesian streaming Tucker decomposition method called BASS-Tucker for handling streaming sparse tensors. Similar to Bayesian methods for streaming CP decomposition, BASS-Tucker adopts the streaming variational Bayes (SVB) framework to infer the posterior of parameters of interest (e.g., tensor core, tensor factors, and nuisance parameters) over time. In addition, BASS-Tucker also utilizes the same priors for the tensor factors and noise variance except that of the core tensor. Here, the following spike-and-slab prior is used to model the core tensor:

$$p(\boldsymbol{\mathcal{S}}|\rho_0) = \prod_{j_1=1}^{r_1} \cdots \prod_{j_N=1}^{r_N} \text{Bern}\left(s_{j_1 \ldots j_N}|\rho_0\right), \tag{41}$$

$$p(\boldsymbol{\mathcal{G}}|\boldsymbol{\mathcal{S}}) = \prod_{j_1=1}^{r_1} \cdots \prod_{j_N=1}^{r_N} s_{j_1 \ldots j_N} \mathcal{N}(g_{j_1 \ldots j_N}|0, \sigma_0^2) + (1 - s_{j_1 \ldots j_N})\delta(g_{j_1 \ldots j_N}), \tag{42}$$

where $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$ is a binary tensor, $\text{Bern}(.|\rho_0)$ is the Bernoulli distribution with probability ρ_0, and $\delta(.)$ is the Delta function. We refer the readers to subsection 4.3 for details on prior distributions of $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ and other model parameters as well as how the SVB framework works.

d) Block-Coordinate Descent. There are three online Tucker algorithms using the BCD framework, including ATD [64], RT-NTD [99] and BK-NTD [99]. In general, they go through the following stages when $\boldsymbol{\mathcal{Y}}_t$ arrives:

Stage 1: Estimate the vector $\mathbf{u}_t^{(N)}$ given old estimations $\boldsymbol{\mathcal{G}}_{t-1}$ and $\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}$. Generally, $\mathbf{u}_t^{(N)}$ can be derived from

$$\min_{\mathbf{u}^{(N)}} \left\| \boldsymbol{\mathcal{Y}}_t - [\![\boldsymbol{\mathcal{G}}_{t-1}; \{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}^{(N)}]\!] \right\|_F^2 + \rho_{\text{u}} \mathcal{R}_{\text{u}}(\mathbf{u}^{(N)}). \tag{43}$$

Stage 2: Estimate the tensor factor $\mathbf{U}_t^{(n)}$ given $\mathbf{u}_t^{(N)}$, old estimation of $\mathbf{U}_{t-1}^{(n)}$ and the remaining factors, $1 \leq n \leq N-1$. The main optimization can be given by

$$\min_{\mathbf{U}^{(n)}} \sum_{\tau=1}^{t} \beta^{t-\tau} \ell(\boldsymbol{\mathcal{Y}}_\tau, \mathbf{U}^{(n)}) + \rho_{\text{U}} \mathcal{R}_{\text{U}}(\mathbf{U}^{(n)}), \tag{44}$$

where $\ell(\boldsymbol{\mathcal{Y}}_\tau, \mathbf{U}^{(n)}) = \|\underline{\mathbf{Y}}_\tau^{(n)} - \mathbf{U}^{(n)} \mathbf{W}_\tau^{(n)}\|_F^2$, $\underline{\mathbf{Y}}_\tau^{(n)}$ and $\mathbf{W}_\tau^{(n)}$ are respectively the mode-n unfolding matrices of $\boldsymbol{\mathcal{Y}}_\tau$ and $\boldsymbol{\mathcal{W}}_\tau$. Here, $\boldsymbol{\mathcal{W}}_\tau$ is defined as $\boldsymbol{\mathcal{W}}_\tau = [\![\boldsymbol{\mathcal{G}}_{t-1}; \{\mathbf{U}_{t-1}^{(m)}\}_{m=1, m\neq n}^{N-1}, \mathbf{u}_\tau^{(N)}]\!]$.

Stage 3: Estimate the core tensor $\boldsymbol{\mathcal{G}}_t$ given $\boldsymbol{\mathcal{G}}_{t-1}$, $\mathbf{u}_t^{(N)}$, and $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$ particularly from

$$\min_{\boldsymbol{\mathcal{G}}} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \underline{\mathbf{Y}}_\tau^{(1)} - \mathbf{U}_t^{(1)} \underline{\mathbf{G}}^{(1)} \mathbf{Z}_\tau \right\|_F^2 + \rho_{\text{G}} \mathcal{R}_{\text{G}}(\boldsymbol{\mathcal{G}}), \tag{45}$$

where $\mathbf{Z}_\tau = \mathbf{u}_\tau \otimes \left(\bigotimes_{n=2}^{N} \mathbf{U}_t^{(n)}\right)$.

Here, $\mathcal{R}_{\text{U}}(.)$, $\mathcal{R}_{\text{U}}(.)$, and $\mathcal{R}_{\text{G}}(.)$ are regularization terms on the coefficient $\mathbf{u}_t^{(N)}$, the factor $\mathbf{U}_t^{(n)}$, and the core tensor $\boldsymbol{\mathcal{G}}_t$, respectively. These penalties can be nonnegativity, smoothness, or sparsity depending on the specific application.

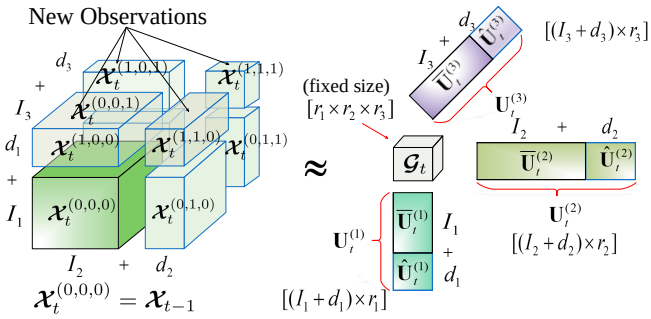


Fig. 5. Multi-aspect streaming Tucker decomposition of a 3rd-order $\boldsymbol{\mathcal{X}}_t$.

The former ATD algorithm was proposed by Thanh *et al.* in [64] which is capable of tracking the low multilinear-rank approximation of streaming tensors from highly incomplete observations. In stage 1, ATD particularly recasts (43) into a standard LS optimization and then applies a randomized LS technique to minimize it. In stage 2, ATD introduces a recursive LS solver to optimize (44) in an efficient way. Instead of solving (45) directly, ATD applies the stochastic gradient descent to obtain its solution.

The two latter RI-NTD and BK-NTD algorithms were proposed by Zdunek *et al.* in [99] for factorizing nonnegative tensors from streaming data. Both algorithms perform nonnegative least-square (NNLS) solvers to incrementally update the tensor factors and the core tensor. Particularly, RI-NTD utilizes a recursive strategy involving the nonnegatively constrained Gauss–Seidel method while BK-NTD adopts the block Kaczmarz method. Similar to ATD, both RI-NTD and BK-NTD estimate the core tensor using only the new coming data via a stochastic optimization.

5.3 Multi-aspect Streaming Tucker Decomposition

Besides single-aspect streaming Tucker decomposition methods, few online techniques are capable of tracking multi-aspect streaming tensors under the Tucker format over time, such as SITTA in [90] and eOTD in [91].

SIITA in [90] offers an online inductive framework for tracking the low-rank tensor approximation of multi-aspect streaming tensors as well as completing their missing data with side information. On the arrival of new data, SIITA particularly minimizes the following optimization

$$\min_{\boldsymbol{\mathcal{G}}, \{\mathbf{U}^{(n)}, \mathbf{A}^{(n)}\}_{n=1}^{N}} f_t\left(\boldsymbol{\mathcal{Y}}_t, \{\mathbf{S}_t^{(n)}\}_{n=1}^{N}, \boldsymbol{\mathcal{G}}, \{\mathbf{U}^{(n)}\}_{n=1}^{N}\right),$$

$$\text{with} \quad f_t\left(\boldsymbol{\mathcal{Y}}_t, \{\mathbf{S}^{(n)}\}_{n=1}^{N}, \boldsymbol{\mathcal{G}}, \{\mathbf{U}^{(n)}\}_{n=1}^{N}\right) = \left\| \mathcal{P}_{\Omega_t}(\boldsymbol{\mathcal{Y}}_t) - \mathcal{P}_{\Omega_t}\left([\![\boldsymbol{\mathcal{G}}; \{\mathbf{S}_t^{(n)} \mathbf{U}^{(n)}\}_{n=1}^{N}]\!]\right) \right\|_F^2 + \rho_{\text{G}} \|\boldsymbol{\mathcal{G}}\|_F^2 + \sum_{n=1}^{N} \rho_n \|\mathbf{U}^{(n)}\|_F^2, \tag{46}$$

where $\{\mathbf{S}_t^{(n)}\}_{n=1}^{N}$ with $\mathbf{S}_t \in \mathbb{R}^{M_n \times I_n}$ is the set of side information matrices and $\rho_{\text{G}}, \{\rho_i\}_{i=1}^{N}$ are regularization parameters. Here, SIITA incorporates the side information into the data model by using $\{\mathbf{S}_t^{(n)}\}_{n=1}^{N}$ as multiplicative terms. Accordingly, SIITA can accelerate the tracking process because the product $\mathbf{S}_t^{(n)} \mathbf{U}^{(n)}$ transforms the dimensionality of variables from I_n to M_n, and typically with $M_n \ll I_n$. As every term of (46) are convex, SITTA adopts the gradient descent to minimize it. Besides, a simple variant of SIITA namely NN-SITTA was also obtained for nonnegative tensor decomposition. NN-SITTA is specifically derived

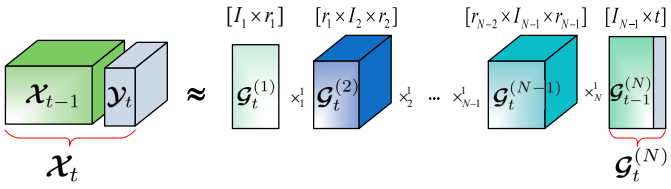


Fig. 6. Single-aspect streaming tensor-train decomposition.

from projecting the estimates of SIITA into their nonnegative orthant at each time t.

In [91], Xiao *et al.* proposed the so-called eOTD algorithm for the multi-aspect tensor tracking problem. Unlike SIITA, eOTD adopts the divide and conquer paradigm to deal with multi-aspect streaming tensors. In particular, it divides the underlying tensor $\boldsymbol{\mathcal{X}}_t$ into 2^N sub-tensors $\boldsymbol{\mathcal{X}}_t^{(i_1,\ldots,i_N)}$ with $i_n \in \{0,1\}$, $1 \leq n \leq N$, and $\boldsymbol{\mathcal{X}}_t^{(0,\ldots,0)} = \boldsymbol{\mathcal{X}}_{t-1}$, see Fig. 5 for an illustration. These sub-tensors are grouped into N classes $\{\mathcal{D}_n\}_{n=1}^N$ based on the sum of sub-indices. For example, for a third-order tensor, we have $\mathcal{D}_1 = \{\boldsymbol{\mathcal{X}}_t^{(1,0,0)}, \boldsymbol{\mathcal{X}}_t^{(0,1,0)}, \boldsymbol{\mathcal{X}}_t^{(0,0,1)}\}$, $\mathcal{D}_2 = \{\boldsymbol{\mathcal{X}}_t^{(1,1,0)}, \boldsymbol{\mathcal{X}}_t^{(1,0,1)}, \boldsymbol{\mathcal{X}}_t^{(0,1,1)}\}$, and $\mathcal{D}_3 = \{\boldsymbol{\mathcal{X}}_t^{(1,1,1)}\}$. If a sub-tensor $\boldsymbol{\mathcal{X}}_t^{(i_1,\ldots,i_n,\ldots,i_N)} \in \mathcal{X}_n$, factorizing it will results in $\boldsymbol{\mathcal{X}}_t^{(i_1,\ldots,i_n,\ldots,i_N)} = [\![\boldsymbol{\mathcal{G}}_t, \{\mathbf{V}_t^{(n)}\}_{n=1}^N]\!]$ where $\mathbf{V}_t^{(n)} = \hat{\mathbf{U}}_t^{(n)}$ if $i_n = 1$ and $\mathbf{V}_t^{(n)} = \mathbf{U}_t^{(n)}$ if $i_n = 0$. Here, the matrix $\hat{\mathbf{U}}_t^{(n)}$ is constantly updated as follows

$$\hat{\mathbf{U}}_{new}^{(n)} = \alpha \hat{\mathbf{U}}_{old}^{(n)} + (1-\alpha)\underline{\mathbf{X}}_{t_n}^{(i_1,\ldots,i_n,\ldots,i_N)} \big(\underline{\mathbf{G}}_{i_n}^{(n)}\big)^{\#}. \tag{47}$$

The factor $\mathbf{U}_t^{(n)}$ is derived from $\mathbf{U}_t^{(n)} = \text{orth}\left([\mathbf{U}_{t-1}^{(n)}; \hat{\mathbf{U}}_{new}^{(n)}]\right) = \left([\bar{\mathbf{U}}_1^{(n)}; \hat{\mathbf{U}}_t^{(n)}]\right)$ where the modified Gram-Schmidt process was applied to compute the $\text{orth}(.)$ operation. Finally, the tensor core $\boldsymbol{\mathcal{G}}_t$ of fixed size is estimated by

$$\begin{aligned}\boldsymbol{\mathcal{G}}_t = &[\![\boldsymbol{\mathcal{G}}_{t-1}; \{(\bar{\mathbf{U}}_t^{(n)})^\top \mathbf{U}_{t-1}^{(n)}\}_{n=1}^N]\!] \\ &+ \sum_{(i_1,\ldots,i_N)\neq(0,\ldots,0)} [\![\boldsymbol{\mathcal{X}}_t^{(i_1,\ldots,i_N)}; \{\hat{\mathbf{U}}_t^{(n)}\}_{n=1}^N]\!].\end{aligned} \tag{48}$$

An appealing feature of eOTD is that throughout the tracking process, eOTD only uses cheap tensor-matrix multiplications and pseudo-inverse operations instead of computing the expensive SVDs on big matrices. This makes eOTD easy for applying to large-scale applications.

6 Other Streaming Tensor Decompositions

Apart from the two most popular streaming CP and Tucker decompositions, some methods are capable of tracking tensors under other multiway models. This section focuses on tracking algorithms that exploit TT, BTD, and t-SVD formats to construct the low-rank tensor approximation in the streaming model.

6.1 Streaming Tensor-Train Decomposition

Despite success in the batch setting, TT decomposition has not gained in popularity as CP and Tucker for tensor tracking. In the literature, there exist few tracking algorithms developed for the problem of single-aspect tensor tracking under the TT format, see Fig. 6 for an illustration.

In [110]–[112], Thanh *et al.* proposed three adaptive TT algorithms called TT-FOA, ATT, and ROBOT for factorizing tensors in an online fashion. Particularly, TT-FOA in [110] is, to the best of our knowledge, the very first of its kind in the literature. However, its practical use is limited due to the lack of robustness to data corruption. To overcome the drawback, ATT in [111] and ROBOT in [112] were developed to deal with missing data and sparse outliers, respectively.

All three algorithms share the same optimization framework where BCD and RLS methods are utilized to minimize the cost function. In particular, a general formulation of the optimization problems can be written as

$$\begin{aligned}\min_{\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^N, \boldsymbol{\mathcal{O}}} \sum_{\tau=1}^{t} \beta^{t-\tau} \Bigg(\Big\| \boldsymbol{\mathcal{P}}_\tau \circledast \Big(\boldsymbol{\mathcal{G}}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \boldsymbol{\mathcal{G}}^{(N-1)} \\ \times_N^1 \mathbf{G}_\tau^{(N)} + \boldsymbol{\mathcal{O}}_\tau - \boldsymbol{\mathcal{Y}}_\tau \Big) \Big\|_F^2 + \rho_{\text{O}} \mathcal{R}_{\text{O}}(\boldsymbol{\mathcal{O}}_\tau) \Bigg) \\ + \rho_{\text{G}} \mathcal{R}_{\text{G}}\Big(\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N-1} \Big),\end{aligned} \tag{49}$$

where $\beta \in (0,1]$ is a forgetting factor to reduce the impact of old observations; $\rho_{\text{O}}\mathcal{R}_{\text{O}}(\boldsymbol{\mathcal{O}}_\tau)$ and $\rho_{\text{G}}\mathcal{R}_{\text{G}}(\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N-1})$ are two regularization terms. Specifically, TT-FOA does not impose the two penalties; ATT adopts $\mathcal{R}_{\text{G}}(\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N-1}) = \sum_{n=1}^{N-1} \|\boldsymbol{\mathcal{G}}^{(n)} - \boldsymbol{\mathcal{G}}_{t-1}^{(n)}\|_F^2$ to control the smoothness of TT-cores over time; and ROBOT applies the ℓ_1-norm regularization $\mathcal{R}_{\text{O}}(\boldsymbol{\mathcal{O}}_\tau) = \|\boldsymbol{\mathcal{O}}_\tau\|_1$ to promote the sparsity on $\boldsymbol{\mathcal{O}}_\tau$.

Thanks to the BCD framework, (49) can be effectively decomposed into two main stages: (i) estimate the temporal TT-core $\mathbf{G}_t^{(N)}$ and outlier $\boldsymbol{\mathcal{O}}_t$, and (ii) update non-temporal TT-cores $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$. In stage 1, TT-FOA and ATT apply the regularized least-squares method to estimate $\mathbf{G}_t^{(N)}$ under the assumption that $\boldsymbol{\mathcal{Y}}_t$ is outlier-free. Meanwhile ROBOT adopts an effective ADMM solver to account for the sparse outlier $\boldsymbol{\mathcal{O}}_t$. In stage 2, an effective RLS solver was introduced to estimate $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$ when $\mathbf{G}_t^{(N)}$ and $\boldsymbol{\mathcal{O}}_t$ (if any) are given in stage 1.

In parallel, Liu *et al.* in [113] proposed an incremental TT method called iTTD to factorize tensors having one temporal mode. Specifically, iTTD considers coming data streams as individual tensors and then factorizes them into TT-cores. The results are appended to old estimates derived from past observations. In [114], Wang *et al.* also developed an incremental TT method called AITT to decompose tensors from industrial IoT data streams. By exploiting a relationship between the directly reshaped matrix and integration of tensor unfolding matrices, AITT can estimate effectively the underlying TT-cores. However, the two frameworks of iTTD and AITT are not really online streaming learning ones but incremental batch learning. Therefore, they are not useful for data streams from dynamical observations in time-varying environments.

6.2 Streaming Block-Term Decomposition

The block-term decomposition (BTD) unifies the two well-known CP and Tucker decompositions, and thus, the tracking algorithms under the CP and Tucker formats principally belong to the class of the streaming BTD with one block. When the number of blocks is greater than 2, there are only two BTD methods able to deal with streaming tensors, including OnlineBTD [115] and O-BTD-RLS [116].

The former method was proposed by Gujral *et al.* in [115] for tensor tracking under the generalized BTD format of L blocks and a multilinear rank-$(r_1, r_2, \ldots, r_N)$. On the arrival of the temporal slice $\boldsymbol{\mathcal{Y}}_t$, OnlineBTD performs the following minimization:

$$\min_{\{\boldsymbol{\mathcal{G}}_l\}_{l=1}^L, \{\mathbf{U}^{(n)}\}_{n=1}^N} \Big\| \boldsymbol{\mathcal{Y}}_t - \sum_{l=1}^{L} [\![\boldsymbol{\mathcal{G}}_l; \{\mathbf{U}_l^{(n)}\}_{n=1}^N]\!] \Big\|_F^2, \tag{50}$$

where $\mathbf{U}^{(n)} = [\mathbf{U}_1^{(n)}, \mathbf{U}_2^{(n)}, \ldots, \mathbf{U}_L^{(n)}]$ with $\mathbf{U}_l^{(n)} \in \mathbb{R}^{I_n \times r_n}$ and $\boldsymbol{\mathcal{G}}_l \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$ $\forall\ l, n$. Here, $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ are supposed to remain unchanged with time except the last tensor factor $\mathbf{U}^{(N)}$. Prior information of r and rank-$(r_1, r_2, \ldots, r_N)$ are known in advance. Old estimates of the core tensors and tensor factors of

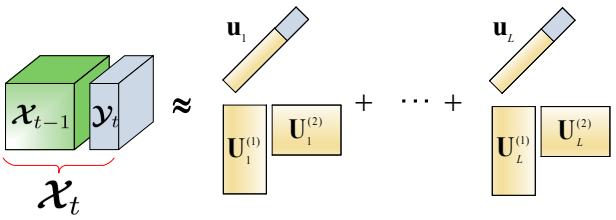


Fig. 7. Tracking the rank-$(r, r, 1)$ BTD of the 3-rd order streaming $\boldsymbol{\mathcal{X}}_t$.

$\boldsymbol{\mathcal{X}}_{t-1}$ are used as a "warm start" for OnlineBTD at each time t. To speed up the tracking process, OnlineBTD utilizes (i) an accelerated matricized tensor times Kronecker product, (ii) the pseudo-inverse operator using LU decomposition, and (iii) a dynamic programming strategy introduced by Zhou *et al.* in [49] to avoid the re-computation of duplicated Kronecker products.

The second method was introduced by Rontogiannis *et al.* in [116]. Specifically, O-BTD-RLS is designed for tracking the low rank-$(r, r, 1)$ terms of three-order streaming tensors (i.e., $r_1 = r_2 = r$ and $r_3 = 1$), see Fig. 7 for an illustration. In particular, the tensor factors of the underlying tensor are incrementally updated by minimizing the following objective function:

$$\min_{\{\mathbf{U}^{(n)}\}_{n=1}^{3}} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \mathbf{Y}_\tau - \mathbf{U}^{(1)} \mathbf{W}_\tau \left[\mathbf{U}^{(2)}\right]^\top \right\|_F^2 \tag{51}$$
$$+ \rho_1 \sum_{l=1}^{L} \sum_{k=1}^{r} \sqrt{\left\|\mathbf{u}_{l,k}^{(1)}\right\|_2^2 + \left\|\mathbf{u}_{l,k}^{(2)}\right\|_2^2 + \eta^2} + \rho_2 \sqrt{\left\|\boldsymbol{\Xi}\mathbf{u}_l\right\|_2^2 + \eta^2},$$

Here, $\mathbf{U}^{(n)} = \left[\mathbf{U}_1^{(n)}, \mathbf{U}_2^{(n)}, \ldots, \mathbf{U}_L^{(n)}\right]$ with $\mathbf{U}_l^{(n)} \in \mathbb{R}^{I_n \times r}$ is the n-th tensor factor of interest and $\mathbf{u}_{l,k}^{(n)}$ is the k-th column of $\mathbf{U}_l^{(n)}$, $n = 1, 2$; $\mathbf{u}^\tau$ and $\mathbf{u}_l$ are the τ-th row and l-th column of the temporal factor $\mathbf{U}^{(3)} \in \mathbb{R}^{t \times L}$, respectively; $\mathbf{W}_\tau = \text{diag}(\mathbf{u}^\tau) \otimes \mathbf{I}_r$ and $\boldsymbol{\Xi} = \text{diag}(\beta^{t-1}, \ldots, \beta, 1)$; ρ_1 and ρ_2 are two regularization parameters; and η^2 is a small positive number to promote smoothness at zero. Specifically, the former term of (51) has the form of weighted least-squares while the two latter terms are regularizations. Accordingly, an efficient recursive least-squares solver was introduced to minimize (51) effectively. An appealing feature of O-BTD-RLS is that it has the abiltity to reveal the BTD ranks (i.e., r and L) over time by specifying the number of columns of the tensor factors which are non-negligible in magnitude at each time t.

6.3 Streaming t-SVD Decomposition

Similar to TT and BTD, streaming t-SVD is still in its early stage. In the literature, there exists only two works of Zhang *et al.* in [117] and Gilman *et al.* in [118] addressing the problem of tensor tracking under the t-SVD format.

In [117], Zhang *et al.* introduced an online tensor PCA for sequential 2D data based on the t-SVD structure. When $\boldsymbol{\mathcal{Y}}_t$ arrives, the proposed algorithm updates:

Step 1: The coefficient matrix $\boldsymbol{\mathcal{W}}_t$ and the sparse outlier $\boldsymbol{\mathcal{O}}_t$ from solving the following minimization

$$\min_{\boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{O}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{U}}_{t-1} * \boldsymbol{\mathcal{W}} - \boldsymbol{\mathcal{O}}\|_F^2 + \frac{\lambda_1}{2}\|\boldsymbol{\mathcal{W}}\|_F^2 + \lambda_2\|\boldsymbol{\mathcal{O}}\|_1. \tag{52}$$

Step 2: The low tubal-rank tensor $\boldsymbol{\mathcal{U}}_t$ (a.k.a. basis dictionary) from taking iFFT of the tensor $\hat{\boldsymbol{\mathcal{U}}}_t$ along the third dimension where $\hat{\boldsymbol{\mathcal{U}}}_t$ is specifically derived from

$$\min_{\hat{\mathbf{U}}} \frac{1}{2} \text{tr}\left[\hat{\mathbf{U}}^\top (\hat{\mathbf{A}}_t + I_3\lambda_1\mathbf{I})\hat{\mathbf{U}}\right] - \text{tr}\left[\hat{\mathbf{U}}^\top \hat{\mathbf{B}}_t\right]. \tag{53}$$

Here, $\hat{\mathbf{A}}_t = \text{diag}(\text{FFT}(\boldsymbol{\mathcal{A}}_t))$ with $\boldsymbol{\mathcal{A}}_t = \boldsymbol{\mathcal{A}}_{t-1} + \boldsymbol{\mathcal{W}}_t * \boldsymbol{\mathcal{W}}_t^\top$, $\hat{\mathbf{B}}_t = \text{diag}(\text{FFT}(\boldsymbol{\mathcal{B}}_t))$ with $\boldsymbol{\mathcal{B}}_t = \boldsymbol{\mathcal{B}}_{t-1} + (\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{O}}_t) * \boldsymbol{\mathcal{W}}_t^\top$, and the solution $\hat{\mathbf{U}}$ is a matricization of $\hat{\boldsymbol{\mathcal{U}}}_t$.

As the online tensor PCA above is not designed for handling missing data, Gilman *et al.* in [118] proposed another algorithm called TOUCAN which is capable of tracking tensors from missing observations. Specifically, the authors proposed to solve the constrained minimization

$$\min_{\mathbf{U}, \mathbf{w}} \sum_{\tau=1}^{t} \left\| \mathcal{F}_{\Omega_\tau}\left(\mathbf{y}_\tau - \mathbf{U}\mathbf{w}_\tau\right) \right\|_2^2 \quad \text{s.t. } \mathbf{U}^\top\mathbf{U} = \mathbf{I}_{rI_3}, \tag{54}$$

where $\mathbf{y}_\tau = \text{unfold}(\boldsymbol{\mathcal{Y}}_\tau) \in \mathbb{C}^{I_1 I_3 \times 1}$, $\mathbf{w}_\tau = \text{unfold}(\boldsymbol{\mathcal{W}}_\tau) \in \mathbb{C}^{rI_3 \times 1}$, $\mathcal{F}_{\Omega_\tau} = \mathcal{P}_{\Omega_\tau}\left(\mathbf{F}_{I_3}^{-1} \otimes \mathbf{I}_{I_1}\right) \in \mathbb{C}^{|\Omega_\tau| \times I_1 I_3}$ is the subsampled inverse Fourier transform, $\mathbf{F}_n \in \mathbb{C}^{n \times n}$ denotes the Discrete Fourier Transform matrix, the mixing matrix $\mathbf{U} \in \mathbb{R}^{I_1 I_3 \times rI_3}$ is defined as $\mathbf{U} = \left(\mathbf{F}_{I_3} \otimes \mathbf{I}_{I_1}\right) \text{bcirc}(\boldsymbol{\mathcal{U}})\, \mathbf{F}_{I_3}^{-1}$.

Motivated by the so-called GROUSE algorithm for subspace tracking in [119], TOUCAN applies the incremental gradient descent on the tensor Grassman manifold to track $\boldsymbol{\mathcal{U}}_t$ with time. It is worth noting that the objective function (54) is very common in subspace tracking problems. Therefore, we can apply any subspace tracking algorithms which are capable of dealing with missing data to minimize (54) effectively.

7 Applications

Tensor tracking or dynamic tensor analysis has already been found several online applications and this section provides some typical examples in different research fields, from computer vision and neuroscience to anomaly detection.

7.1 Computer Vision

We begin this section with one of the earliest and most popular applications of tensor tracking: visual tracking which is an important task in computer vision [120]. Naturally, video datasets can be represented as 4-th order streaming tensors of dimensionality, width $\times$ height $\times$ channel $\times$ time. Accordingly, there are several studies devoted to developing tensor-based visual trackers for better modeling the appearance of target objects, such as [80], [121]–[123], to name a few. For example, Hu *et al.* in [80] proposed the so-called IRTSA tracker using incremental tensor subspace learning to capture the appearance of objects. Zhang *et al.* in [121] introduced another visual tracker called DTAMU which stands for dynamic tensor analysis with mean update. Weiming *et al.* in [122] developed a semi-supervised tensor-based visual tracker using graph embedding. Khan *et al.* in [123] built an online spatio-temporal tensor learning model for visual tracking using Bayesian inference. It is worth noting that most of the existing tensor-based visual trackers correspond to the streaming Tucker decomposition and its variants.

Another notable application of tensor tracking in computer vision is video background and foreground separation which is quite related to visual tracking, but with a different aim of modeling the scene background and detecting the information of changes in the scene. Similar to visual tracking, many tensor-based separators were proposed, such as [65], [105], [112], [124], [125]. Particularly in [65], Thanh *et al.* proposed a robust adaptive CP method called RACP which is capable of modeling video background and detecting moving objects. Li *et al.* in [105] introduced an online robust low-rank tensor modeling (ORLTM) method and found its success in video background subtraction. Andrews *et al.* in [124] developed an online stochastic tensor decomposition for background subtraction in multispectral video sequences. A robust streaming tensor-train algorithm was developed in [112] which also has the potential to detect foreground in video. Salut *et al.* in [125] proposed an online tensor robust principal component analysis and validated its effectiveness with the problem of background and foreground separation.

In parallel, there are other interesting computer vision applications of dynamic tensor analysis, such as visual data recovery [56], [126], online video denoising [127], [128], and segmentation/classification [94], [129].

7.2 Neuroscience

The brain can be viewed as a complex system with various interacting regions that can produce large multivariate data over time [130]. Many types of brain data can be represented by tensors, such as electroencephalography (EEG), magnetoencephalography (MEG), functional magnetic resonance imaging (fMRI), and near-infrared spectroscopy (NIRS) [131]. Apart from three intrinsic modes (i.e., frequency, channel, and time), brain data can have higher-order modes, such as, subjects, conditions, and trials [131]. Together with the fact that brain activities can change over time, dynamic tensor analysis has become an useful tool to study the structure and function of brain from such data.

In what follows, we list some appealing brain-computer interface applications to demonstrate the use of dynamic tensor analysis in neuroscience. First, for the problem of detecting dynamic functional connectivity networks (DFCNs), Ozdemir *et al.* in [87] introduced a recursive tensor-based framework capable of tracking DFCNs over time. The proposed framework was then applied for studying error-related negativity – a brain potential response when patients make errors during cognitive tasks [132]. Mahyari *et al.* in [133] developed a two-step approach using incremental tensor subspace analysis for detecting DFCNs. Particularly, they first detect change points at which the functional connectivity across subjects presents abrupt changes and then summarize DFCNs between successive change points. Recently, Acar *et al.* in [134] proposed to use the Parafac2 model for tracking the evolution of connectivity networks and compared its performance with ICA and IVA. For the problem of localizing dynamic brain sources over time, Ardeshir *et al.* in [135] utilized the boundary element method (BEM) [136] and the adaptive PARAFAC-RLST tracker [46] with two operational windowing schemes. A variant using augmented complex statistics in [137] also has the ability to track moving EEG sources with time. For the problem of online EEG completion, Trung *et al.* in [138] proposed an adaptive CP algorithm called NL-PETRELS capable of tracking and imputing incomplete EEG data. Thanh *et al.* in [64], [65] also demonstrated the use of ACP and RACP with real data by applying them for online EEG completion. Other neuroscience applications of tensor analysis were reviewed in [9], [33], [35].

7.3 Anomaly Detection

Anomaly detection, which corresponds to identifying patterns and data points that do not conform to normal behavior, plays an essential role in many applications, such as cyber security, statistics, and finance, to name a few [139]. Here, we provide some notable tensor-based anomaly detectors which are customized to specific online applications.

Shi *et al.* in [140] developed the so-called STenSr algorithm for anomaly detection and pattern discovery in spatio-temporal tensor streams from sensor networks. STenSr utilizes an incremental HOSVD and a metric based on Euclidean distance to detect abrupt changes when new data comes. Kasai *et al.* in [141] introduced an online time-structured traffic tensor tracking framework to detect network-level anomalies from link indirect measurements over time. In particular, it is based on a robust adaptive CP decomposition that uses RLS for tensor tracking and ADMM for detecting abnormal flows. Cao *et al.* in [142] designed an interactive system called Voila for detecting and monitoring visual anomalies. Voila is a tensor-based anomaly detector with an interaction design that can ranks anomalous patterns based on user input. Lin *et al.* in [143] proposed a novel method called TBAD to localize anomalous events. TBAD employs a spatial-feature-temporal tensor model and analyses latent patterns through unsupervised learning. Xu *et al.* in [144] introduced a tensor-based framework, namely SWTF, capable of detecting multiple types of anomalies in road networks. We refer the readers to [34] for a broader interdisciplinary survey of tensors for anomaly detection.

7.4 Others

Apart from online applications in the domains above, tensor tracking also found success in some other research fields, namely wireless communications (e.g., channel tracking [145], DOA tracking [146], and time delay estimation [147]), network analysis (e.g., link prediction [12], internet scale monitoring [148], and bot activities and network intrusions [149]), data analytics of chemical and biological manufacturing processes and components [150], [151], performance monitoring [152], [153], and transportation [154], [155].

8 Research Challenges, Open Problems, and Future Directions

In this section, we first discuss some advantages and disadvantages of the state-of-the-art tensor tracking models. Then, we present several research challenges and open problems that should be considered for the development of tensor tracking in the future. They are data imperfection and corruption; rank revealing and tracking; efficient and scalable tensor tracking; and other aspects such as theoretical analysis, symbolic data, and tracking under some less common tensor formats. Possible solutions for these challenges are also discussed.

8.1 Discussions on State-of-the-art Tensor Tracking Models

As reviewed in previous sections, most of the state-of-the-art tensor tracking methods were developed for streaming CP and Tucker decompositions. Indeed, the CP tracking model can be seen as a special case of streaming Tucker decomposition when the core tensor is constrained to be an identity tensor. Therefore, the complexity (w.r.t. computation and memory storage) of streaming CP decomposition is generally much lower than that of Tucker. In other words, CP trackers are often faster than Tucker ones in practice. Another interesting feature of the CP tracking model over Tucker's online variants is the uniqueness in which tensor factors of CP are unique up to a permutation and scale under certain conditions, similar to batch CP decomposition. This feature is a useful property in several applications, e.g., for source separation or to recover exact components and individuals hidden in the underlying data. However, CP is not really a practical model because finding the true CP rank of streaming tensors is nontrivial even when we have enough resources to process multiple snapshots at a time. The problem of tensor rank tracking will be discussed later in subsection 8.3.

Streaming Tucker decomposition is, however, more flexible than CP. It stems from the fact that we here only need to track the column spaces of tensor factors over time, instead of a particular basis. In addition, the Tucker rank determination may be "easier" than that of CP as we can take advances of rank estimation in the problem of subspace tracking (e.g., [156], [157]). However, when we deal with large-scale and high-dimensional streaming tensors, the size of the core tensor can be large which makes Tucker trackers less efficient for stream processing. The BTD tracking model lies between CP and Tucker in the sense that

the core tensor is block diagonal. Thus, it has the advantages of both CP and Tucker tracking models. Despite having that, tracking the underlying BTD approximation of streaming tensors may be harder than CP and Tucker as it takes several types of rank into account, i.e., the number of blocks and their size. The tensor-train tracking model shares the same drawback with BTD as it must estimate several TT-cores and their rank. An appealing advantage of streaming TT decomposition is that it can deal with tensors of a very high order, thanks to its memory-saving representation which is linear to the order of tensor [5]. The t-SVD tracking model has its own advantage and disadvantage. As its algebraic framework is in the Fourier domain, t-SVD can utilize fast operations (e.g., FFT and its inverse) to speed up the tracking process. Generalizing the t-SVD to tensors of order greater than three can be done via recursion [158] or hot-HOSVD [159] which are quite expensive. It becomes a bottleneck that can make tracking higher-order streaming tensors under the t-SVD model inefficient.

8.2 Data Imperfection and Corruption

Dealing with data imperfection and corruption has been a critical issue in many applications and tracking problems in particular [160]. We here present two main types of imperfect data that either remain unsolved or are still challenging for tensor tracking: (i) non-Gaussian and colored noises; (ii) outliers and missing data.

a) Non-Gaussian and Colored Noises. Most of the existing tensor tracking algorithms were proposed under the additive white Gaussian noise assumption. This assumption however does not always hold in practice. For example, impulsive noises (e.g., burst, alpha-stable, and spherically invariant random variable noise), which are introduced by human activities and natural sources, are one of the most common non-Gaussian noises that often appear in tracking applications such as direction of arrivals [161], OFDM systems [162] and adaptive system identification [163]. This type of noise can significantly impact the tracking ability of estimators and it requires specific treatments [164]. In parallel, colored noises that indicate types of noise that are correlated in space and/or time may reduce the performance of tracking algorithms [165]. Accordingly, standard tracking algorithms may be less effective in estimation accuracy in the presence of these noises. They need to be readapted or redesigned for more robustness.

To the best of our knowledge, we are not aware of any tensor tracking algorithm capable of handling such noises in the literature. Some potential approaches have been successfully demonstrated in subspace tracking problems (i.e., tracking tensors of order 2), see [164] for a brief survey. In particular, adaptive Kalman filtering and weighted RLS approaches can be adopted for dealing with impulsive noises. Oblique projection and instrumental variable-based techniques can handle colored noises. Therefore, it is desirable to extend these approaches from subspace tracking to tensor tracking.

b) Outliers and Missing Data. They are now becoming more and more ubiquitous in modern datasets. Outliers are data points that appear to be inconsistent with or exhibit abnormal behaviour different from others. Missing observations are often encountered during the data acquisition and collection. Both outliers and missing data can cause several issues (e.g., they introduce bias in estimation) for knowledge discovery from data in general and data streams in particular [166]. Accordingly, dealing with them is an essential task in the analysis of corrupted datasets which has been still a hot topic in data mining for decades. In general, handling such corruptions involves removing/ignoring them after detection or replacing them with alternative values.

There exist few tensor tracking algorithms robust to sparse outliers in the literature. Under the CP format, SOFIA [61] applies the robust Holt-Winters forecasting model using a pre-cleaning mechanism to identify and down-weight outliers. RACP [65] introduces a ℓ_1-norm penalty to promote the sparsity on outliers and then uses an ADMM solver to estimate them. Under the Tucker format, ORLTM [105], OLRTR [106], and D-L1-Tucker [96] are able to deal with sparse outliers. Both ORLTM and OLRTR propose to regularize the main objective function with a ℓ_1-norm regularization. Meanwhile, D-L1-Tucker adopts a threshold-based method to detect outliers. Except for RACP, most of the mentioned algorithms above are not designed for dealing with missing data. In parallel, most of the existing online tensor completion and tracking are sensitive to outliers, such as TeCPSGD [48], OLSTEC [56], and ACP [64]. Accordingly, there are plenty of opportunities to develop robust tensor tracking from incomplete observations as it is still in its early stage.

8.3 Rank Revealing and Tracking

Most of the state-of-the-art tensor tracking algorithms suppose that the tensor rank (e.g., CP, Tucker, BTD, TT, or tubal rank) is given as prior information. In practice, it is however a difficult assumption due to the facts that: (i) the tensor rank may change over time and (ii) a good rank determination at the initialization stage is not always guaranteed when the number of training samples is limited and (iii) the exact rank determination may be intractable (e.g., CP rank is NP-hard [27]). Therefore, it is essential to develop tracking algorithms that are capable of revealing the rank over time.

In the literature, there have been many heuristic methods developed for the problem of tensor rank estimation. Most of them adopt the Bayesian approach to infer the tensor rank from data, such as [167]–[169]. Theoretically, Bayesian inference offers a good recipe for the tensor rank estimation as we can integrate the low-rank promoting prior as well as the tensor rank into the learning framework. Another possible approach to determine the tensor rank is to use neural networks (NNs), such as [170]–[172]. Since the rank can be considered as one type of data feature, NNs which can extract hidden features within data can be used to solve the tensor rank determination. Although these methods often require the tensor data to be fully observed, it is possible to readapt or modify them such that their variant are able to handle tensors in an online fashion. For example, we can adopt online Bayesian inference or online learning algorithms for training NNs. Finally, robustness against rank overestimation errors is an issue that, to the best of our knowledge, has not been considered yet in practice.

8.4 Efficient and Scalable Tensor Tracking

Tabs. 3 and 5 indicate that most of the existing tensor tracking algorithms are of high complexity. When we deal with large-scale and high-multidimensional streams, they may become less efficient. Thus, it is necessary to develop efficient and scalable tracking techniques of low cost w.r.t. both computational complexity and memory storage. In what follows, we present three potential approaches which are theoretically capable of accelerating the tracking process, namely (a) randomized sketching, (b) parallel and distributed computing, and (c) neural networks-based methods.

a) Randomized Sketching. It is very well-known that randomized methods can reduce the computational cost of their counterparts while still achieving reasonable estimation [173].

Accordingly, many attempts have been made to take their advantages in computation for tensor decomposition in the literature, we refer the readers to [23] for a good overview. Among them, there are a few online algorithms utilizing successfully randomized techniques to speed up the tracking process, such as [57], [63], [64], [174]. Particularly, these algorithms involve solving several overdetermined least-squares (LS) problems. Thanks to the CP and Tucker structures, they use random sampling to build the sampled Khatri-Rao and Kronecker products, and then, recast the original LS problems into randomized ones. Solving the new LS problems can save a lot of computational complexity. Other randomized techniques (e.g., random projections and count sketch) with other tensor formats have not yet been investigated for tensor tracking and they deserve next investigations in the future.

b) Parallel and Distributed Computing. The second approach is to develop parallel and distributed computing frameworks for streaming tensor decomposition. It stems from the fact that we can leverage several computational resources to facilitate the tracking process. Moreover, computing systems in a parallel and distributed environment can offer more reliability than their counterparts in a central one as they can avoid the single point of failure which is a fundamental mistake from flaws in the implementation or design of a system. Besides, another appealing advantage of this computing is the scaling up-and-out process in which we can add and/or replace computational resources to the system. We refer the readers to [175] for a good reference.

In the tensor literature, there are several parallel and distributed systems for processing large-scale tensors. We can list here some efficient tools for: (a) distributed CP decomposition (e.g., DFacTo [176], SPLATT [177]), (b) distributed Tucker decomposition (e.g., DHOSVD [88], SGD-Tucker [178]), and (c) distributed TT decomposition (e.g., ADTT [114], ATTAC [179]), etc. These tools mainly distribute the unfolding matrices or sub-tensors among several clusters and integrate their low-rank tensor approximations to find the overall low-rank approximation of the underlying tensor. However, most of the existing distributed tensor decompositions are not suitable for handling streaming data. Therefore, it is of great interest to develop practical distributed systems for tracking tensors from data streams.

c) Neural Networks-based Methods. Another potential approach is to incorporate neural networks (NNs) into tensor factorization to benefit from their significant advances in computational power. On the one hand, the connection between TDs and NNs has been established in some studies, such as [180], [181]. For example, Cohen *et al.* in [180] showed that the convolutional NNs with ReLU activation and max/average pooling can be represented by tensor decomposition models. Wang *et al.* in [181] introduced two NN models for finding the low-tubal-rank approximation of three-order tensors. Accordingly, NN tools can be used to model and learn high-order interactions for tensors, and hence, for tensor factorization and tracking. On the other hand, NNs can directly map data streams (temporal slices) as input to the approximation result as output by applying some online learning techniques. In the literature of machine learning, there exist several kinds of learning capable of dealing with data streams, such as incremental learning, lifelong learning, and online continual learning, to name a few. They can be specifically adapted for tensor tracking.

8.5 Others

Next, we present some other issues and problems which also deserve future investigations.

a) Provable Tensor Tracking. Although the existing tensor tracking methods can provide competitive performance w.r.t. estimation accuracy and/or convergence rate in practice, most of them lack performance guarantees. The gap between practical uses/implementations and theoretical results in tensor tracking may be caused by the fact that most tensor problems are NP-hard [27], e.g., the best rank-1 tensor approximation is NP-hard even when all observations (temporal slices) are fully observed. Despite several difficulties, there are still attempts to bridge the gap in the literature. Under certain conditions (e.g., the underlying low-rank model remains unchanged over time), some studies established successfully theoretical results to analyse the convergence behavior of their methods, such as [56], [58], [64], [65], [93]. These initial results encourage us to investigate deeper theoretical aspects in tensor tracking, such as time variation, asymptotic convergence, and non-asymptotic convergence in low-sample-size settings.

b) Symbolic Tensor Tracking. In some applications, data may no longer be represented by single (certain) values, but need to be formatted or grouped within sets, intervals, histograms, etc. It leads to the so-called symbolic data analysis (SDA) paradigm in data mining and statistics to deal with such data [182]. In SDA, several new variables types and processing tools have been introduced to represent and analyse symbolic data, such as interval-valued, histogram-valued, and categorical modal variables, to name a few. The readers are referred to [182] for a good survey on SDA. In the tensor literature, Mauro *et al.* in [183] proposed for the first time a symbolic tensor decomposition for factorizing interval-valued tensors under the tensor-train format. Specifically, the authors extended a set of tools aiming to handle interval-valued matrices for high-order tensors and introduced efficient decomposition and reconstruction strategies. As the symbolic tensor decomposition is in its very early stage of development in both batch and online settings, there are a lot of aspects that need to be investigated in the future.

c) Tensor tracking under BTD, t-SVD, tensor network formats, and other variants. Despite great success in the batch setting, BTD, t-SVD, and tensor networks (e.g., tensor-train, tensor chain, and tensor ring) have not attracted much attention in real-time stream processing until recently. Thus, developing online methods for tracking tensors under these tensor formats and their variants is essential to take advantage from their advantage in representing large-scale tensors as well as fulfil the gap between the two most common formats and others.

9 Conclusions

Tensor tracking has recently gained increasing attention as a powerful tool for multidimensional data stream analysis. In this survey, we have provided a technical overview of online techniques for tracking streaming tensors over time. We highlighted the two most popular streaming CP and Tucker decompositions. Specifically, four main groups of streaming CP decomposition algorithms were emphasized, which are subspace-based, block-coordinate descent, Bayesian inference, and multi-aspect streaming decompositions. We categorized the current streaming Tucker decomposition methods into three major classes based on their model architecture. They are online tensor dictionary learning, tensor subspace tracking, and multi-aspect streaming decompositions. Recent years have also witnessed significant advances in other types of tensor decomposition such as tensor-train, BTD, and t-SVD. A brief survey on the existing methods which are capable of tracking tensors under these formats was presented. Finally, we discussed several research challenges, open problems, and future directions for tensor tracking.

REFERENCES

[1] N. D. Sidiropoulos, L. D. Lathauwer *et al.*, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, 2017.

[2] Y. Liu, J. Liu *et al.*, *Tensor Computation for Data Analysis*, 2022.

[3] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis," *UCLA Work. Pap. Phon.*, vol. 16, no. 1-84, 1970.

[4] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

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

[6] M. Kilmer and C. Martin, "Factorization strategies for third-order tensors," *Linear Algebra Appl.*, vol. 435, no. 3, pp. 641–658, 2011.

[7] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms – Part II: Definitions and uniqueness," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1033–1066, 2008.

[8] L. T. Thanh, N. T. A. Dao *et al.*, "Multi-channel EEG epileptic spike detection by a new method of tensor decomposition," *J. Neural Eng.*, vol. 17, no. 1, p. 016023, 2020.

[9] E. Karahan *et al.*, "Tensor analysis and fusion of multimodal brain images," *Proc. IEEE*, vol. 103, no. 9, pp. 1531–1559, 2015.

[10] D. Nion and N. D. Sidiropoulos, "Tensor algebra and multidimensional harmonic retrieval in signal processing for MIMO radar," *IEEE Trans. Signal Process.*, vol. 58, pp. 5693–5705, 2010.

[11] H. Chen, F. Ahmad *et al.*, "Tensor decompositions in wireless communications and MIMO radar," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 3, pp. 438–453, 2021.

[12] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Trans. Knowl. Discov. Data*, vol. 5, no. 2, pp. 1–27, 2011.

[13] S. Fernandes, H. Fanaee-T, and J. Gama, "Tensor decomposition for analysing time-evolving social networks: An overview," *Artif. Intell. Rev.*, vol. 54, no. 4, pp. 2891–2916, 2021.

[14] 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.

[15] M. Bahri *et al.*, "Data stream analysis: Foundations, major tasks and tools," *WIREs Data Min. Knowl. Discov.*, vol. 11, no. 3, p. 1405, 2021.

[16] R. Bro, "PARAFAC. Tutorial and applications," *Chemometr. Intell. Lab. Syst.*, vol. 38, no. 2, pp. 149–172, 1997.

[17] E. Acar and B. Yener, "Unsupervised multiway data analysis: A literature survey," *IEEE Trans Knowl. Data Eng.*, vol. 21, pp. 6–20, 2008.

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

[19] G. Bergqvist and E. G. Larsson, "The higher-order singular value decomposition: Theory and an application," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 151–154, 2010.

[20] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, 2013.

[21] N. Vervliet *et al.*, "Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis," *IEEE Signal Process. Mag.*, vol. 31, pp. 71–79, 2014.

[22] A. Cichocki *et al.*, "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.

[23] S. A. Asl *et al.*, "Randomized algorithms for computation of Tucker decomposition and higher-order SVD (HOSVD)," *IEEE Access*, vol. 9, pp. 28 684–28 706, 2021.

[24] X. Fu *et al.*, "Computing large-scale matrix and tensor decomposition with structured factors: A unified nonconvex optimization perspective," *IEEE Signal Process. Mag.*, vol. 37, no. 5, pp. 78–94, 2020.

[25] D. Muti and S. Bourennane, "Survey on tensor signal algebraic filtering," *Signal Process.*, vol. 87, no. 2, pp. 237–249, 2007.

[26] P. Comon, X. Luciani, and A. L. De Almeida, "Tensor decompositions, alternating least squares and other tales," *J. Chemom.*, vol. 23, no. 7-8, pp. 393–405, 2009.

[27] C. J. Hillar and L.-H. Lim, "Most tensor problems are NP-hard," *J. ACM*, vol. 60, no. 6, p. 45, 2013.

[28] P. Comon, "Tensors : A brief introduction," *IEEE Signal Process. Mag.*, vol. 31, no. 3, pp. 44–53, 2014.

[29] A. Zare, A. Ozdemir *et al.*, "Extension of PCA to higher order data structures: An introduction to tensors, tensor decompositions, and tensor PCA," *Proc. IEEE*, vol. 106, no. 8, pp. 1341–1358, 2018.

[30] T. Adali *et al.*, "Reproducibility in matrix and tensor decompositions: Focus on model match, interpretability, and uniqueness," *IEEE Signal Process. Mag.*, vol. 39, no. 4, pp. 8–24, 2022.

[31] M. Morup, "Applications of tensor (multiway array) factorizations and decompositions in data mining," *Data Min. Knowl. Discov.*, vol. 1, no. 1, pp. 24–40, 2011.

[32] A. Cichocki *et al.*, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, 2015.

[33] F. Cong *et al.*, "Tensor decomposition of EEG signals: A brief review," *J. Neurosci. Methods*, vol. 248, pp. 59–69, 2015.

[34] H. Fanaee-T and J. Gama, "Tensor-based anomaly detection: An interdisciplinary survey," *Knowl. Based Syst.*, vol. 98, pp. 130–147, 2016.

[35] G. Zhou, G. Zhao *et al.*, "Linked component analysis from matrices to high-order tensors: Applications to biomedical data," *Proc. IEEE*, vol. 104, no. 2, pp. 310–331, 2016.

[36] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 2, p. 16, 2017.

[37] A. Cichocki *et al.*, "Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives," *Found. Trends Mach. Learn.*, vol. 9, no. 6, pp. 431–673, 2017.

[38] Y. Ji *et al.*, "A survey on tensor techniques and applications in machine learning," *IEEE Access*, vol. 7, pp. 162 950–162 990, 2019.

[39] S. Miron *et al.*, "Tensor methods for multisensor signal processing," *IET Signal Process.*, vol. 14, no. 10, pp. 693–709, 2021.

[40] Y. Panagakis *et al.*, "Tensor methods in computer vision and deep learning," *Proc. IEEE*, vol. 109, no. 5, pp. 863–890, 2021.

[41] K. Batselier, "Low-rank tensor decompositions for nonlinear system identification: A tutorial with examples," *IEEE Control Syst. Mag.*, vol. 42, no. 1, pp. 54–74, 2022.

[42] V. De Silva and L.-H. Lim, "Tensor rank and the ill-posedness of the best low-rank approximation problem," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1084–1127, 2008.

[43] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-(r_1, r_2, ..., r_n) approximation of higher-order tensors," *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 1324–1342, 2000.

[44] L. De Lathauwer and D. Nion, "Decompositions of a higher-order tensor in block terms – Part III: Alternating least squares algorithms," *SIAM J. Matrix Anal. Appl.*, vol. 30, pp. 1067–1083, 2008.

[45] H. Fanaee-T and J. Gama, "Multi-aspect-streaming tensor analysis," *Knowl. Based Syst.*, vol. 89, pp. 332–345, 2015.

[46] D. Nion and N. D. Sidiropoulos, "Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2299–2310, 2009.

[47] V. D. Nguyen *et al.*, "Fast adaptive PARAFAC decomposition algorithm with linear complexity," in *IEEE ICASSP*, 2016, pp. 6235–6239.

[48] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Trans. Signal Process.*, vol. 63, no. 10, pp. 2663–2677, 2015.

[49] S. Zhou, N. X. Vinh *et al.*, "Accelerating online CP decompositions for higher order tensors," in *ACM KDD*, 2016, pp. 1375–1384.

[50] N. V. Dung, K. Abed-Meraim, and N. L. Trung, "Second-order optimization based adaptive PARAFAC decomposition of three-way tensors," *Digit. Signal Process.*, vol. 63, pp. 100–111, 2017.

[51] M. Vandecappelle *et al.*, "Nonlinear least squares updating of the canonical polyadic decomposition," in *EUSIPCO*, 2017, pp. 663–667.

[52] Z. Zhang and C. Hawkins, "Variational bayesian inference for robust streaming tensor factorization and completion," in *IEEE ICDM*, 2018, pp. 1446–1451.

[53] T. M. Chinh, N. V. Dung *et al.*, "Adaptive PARAFAC decomposition for third-order tensor completion," in *IEEE ICCE*, 2016, pp. 297–301.

[54] S. Smith, K. Huang *et al.*, "Streaming tensor factorization for infinite data sources," in *SIAM ICDM*, 2018, pp. 81–89.

[55] Y. Du, Y. Zheng *et al.*, "Probabilistic streaming tensor decomposition," in *IEEE ICDM*, 2018, pp. 99–108.

[56] 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.

[57] H.-K. Yang *et al.*, "Incremental PARAFAC decomposition for three-dimensional tensors using Apache Spark," in *ICWE*, 2019, pp. 63–71.

[58] S. Rambhatla *et al.*, "Provable online CP/PARAFAC decomposition of a structured tensor via dictionary learning," in *NeurIPS*, 2020, pp. 1–12.

[59] E. Gujral *et al.*, "SPADE: Streaming PARAFAC2 decomposition for large datasets," in *SIAM ICDM*, 2020, pp. 577–585.

[60] T. Kwon *et al.*, "SliceNStitch: Continuous CP decomposition of sparse tensor streams," in *IEEE ICDE*, 2021, pp. 816–827.

[61] L. Dongjin and S. Kijung, "Robust factorization of real-world tensor streams with patterns, missing values, and outliers," in *IEEE ICDE*, 2021, pp. 840–851.

[62] D. Ahn, S. Kim, and U. Kang, "Accurate online tensor factorization for temporal tensor streams with missing values," in *ACM CIKM*, 2021, pp. 2822–2826.

[63] L. T. Thanh *et al.*, "A fast randomized adaptive CP decomposition for streaming tensors," in *IEEE ICASSP*, 2021, pp. 2910–2914.

[64] ——, "Tracking dynamic low-rank approximations of incomplete high-order streaming tensors," *Patterns*, 2022.

[65] ——, "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.

[66] H. Lyu *et al.*, "Online nonnegative CP-dictionary learning for Markovian data," *J. Mach. Learn. Res.*, no. 23, pp. 1–50, 2022.

[67] R. A. Harshman *et al.*, "Parafac2: Mathematical and technical notes," *UCLA Work. Pap. Phon.*, vol. 22, no. 3044, p. 122215, 1972.

[68] C. Chatfield, "The holt-winters forecasting procedure," *J. R. Stat. Soc. Ser. C Appl. Stat.*, vol. 27, no. 3, pp. 264–279, 1978.

[69] P. Strobach, "Bi-iteration SVD subspace tracking algorithms," *IEEE Trans. Signal Process.*, vol. 45, no. 5, pp. 1222–1240, 1997.

[70] 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.

[71] S. Fang *et al.*, "Streaming Bayesian deep tensor factorization," in *ICML*, 2021, pp. 3133–3142.

[72] T. Broderick *et al.*, "Streaming variational Bayes," in *NeurIPS*, 2013.

[73] X. Boyen and D. Koller, "Tractable inference for complex stochastic processes," in *UAI*, 1998, pp. 33–42.

[74] Q. Song *et al.*, "Multi-aspect streaming tensor completion," in *ACM KDD*, 2017, pp. 435–443.

[75] M. Najafi, L. He, and S. Y. Philip, "Outlier-robust multi-aspect streaming tensor completion and factorization." in *IJCAI*, 2019, pp. 3187–3194.

[76] H.-K. Yang and H.-S. Yong, "Multi-aspect incremental tensor decomposition based on distributed in-memory big data systems," *J. Data Inf. Sci.*, vol. 5, no. 2, pp. 13–32, 2020.

[77] K. Yang *et al.*, "DisMASTD: An efficient distributed multi-aspect streaming tensor decomposition," in *IEEE ICDE*, 2021, pp. 1080–1091.

[78] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: Dynamic tensor analysis," in *ACM KDD*, 2006, pp. 374–383.

[79] J. Sun *et al.*, "Incremental tensor analysis: Theory and applications," *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 3, pp. 1–37, 2008.

[80] X. Li, W. Hu *et al.*, "Robust visual tracking based on incremental tensor subspace learning," in *IEEE ICCV*, 2007, pp. 1–8.

[81] W. Hu, X. Li *et al.*, "Incremental tensor subspace learning and its applications to foreground segmentation and tracking," *Int. J. Comput. Vis.*, vol. 91, no. 3, pp. 303–327, 2011.

[82] W. Zhang *et al.*, "An incremental tensor factorization approach for web service recommendation," in *IEEE ICDM*, 2014, pp. 346–351.

[83] L. Kuang, F. Hao *et al.*, "A tensor-based approach for big data representation and dimensionality reduction," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 280–291, 2014.

[84] R. Yu, D. Cheng *et al.*, "Accelerated online low rank tensor learning for multivariate spatiotemporal streams," in *ICML*, 2015, pp. 238–247.

[85] M. Baskaran, M. H. Langston *et al.*, "Accelerated low-rank updates to tensor decompositions," in *IEEE HPEC*, 2016, pp. 1–7.

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

[87] A. Ozdemir, E. M. Bernat, and S. Aviyente, "Recursive tensor subspace tracking for dynamic brain network analysis," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 4, pp. 669–682, 2017.

[88] X. Wang, W. Wang *et al.*, "A distributed HOSVD method with its incremental computation for big data in cyber-physical-social systems," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 2, pp. 481–492, 2018.

[89] L. T. Yang, X. Wang *et al.*, "A multi-order distributed HOSVD with its incremental computing for big services in cyber-physical-social systems," *IEEE Trans. Big Data*, vol. 6, no. 4, pp. 666–678, 2020.

[90] N. Madhav, B. Mishra *et al.*, "Inductive framework for multi-aspect streaming tensor completion with side information," in *ACM CIKM*, 2018, pp. 307–316.

[91] H. Xiao, F. Wang *et al.*, "eOTD: An efficient online tucker decomposition for higher order tensors," in *IEEE ICDM*, 2018, pp. 1326–1331.

[92] A. Traore, M. Berar, and A. Rakotomamonjy, "Online multimodal dictionary learning," *Neurocomput.*, vol. 368, pp. 163–179, 2019.

[93] ——, "Singleshot: A scalable Tucker tensor decomposition," in *NeurIPS*, 2019, pp. 1–12.

[94] Y. Sun *et al.*, "Low-rank tucker approximation of a tensor from streaming data," *SIAM J. Math. Data Sci.*, vol. 2, no. 4, pp. 1123–1150, 2020.

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

[96] D. G. Chachlakis *et al.*, "Dynamic L1-norm Tucker tensor decomposition," *IEEE J. Sel. Topics Signal Process.*, vol. 15, pp. 587–602, 2021.

[97] S. Fang, R. M. Kirby, and S. Zhe, "Bayesian streaming sparse Tucker decomposition," in *UAI*, 2021, pp. 558–567.

[98] J.-G. Jang and U. Kang, "Fast and memory-efficient tucker decomposition for answering diverse time range queries," in *ACM KDD*, 2021, pp. 725–735.

[99] R. Zdunek and K. Fonal, "Incremental nonnegative Tucker decomposition with block-coordinate descent and recursive approaches," *Symmetry*, vol. 14, no. 1, p. 113, 2022.

[100] D. A. Ross *et al.*, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, no. 1, pp. 125–141, 2008.

[101] J. Li, G. Han *et al.*, "Robust tensor subspace learning for anomaly detection," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 89–98, 2011.

[102] X. Wang, L. T. Yang *et al.*, "Improved multi-order distributed HOSVD with its incremental computing for smart city services," *IEEE Trans. Sust. Comput.*, vol. 6, no. 3, pp. 456–468, 2021.

[103] H. Lu *et al.*, "A survey of multilinear subspace learning for tensor data," *Pattern Recognit.*, vol. 44, no. 7, pp. 1540–1551, 2011.

[104] R. Zhao and Q. Wang, "Learning separable dictionaries for sparse tensor representation: An online approach," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 66, no. 3, pp. 502–506, 2019.

[105] P. Li, J. Feng, X. Jin, L. Zhang, X. Xu, and S. Yan, "Online robust low-rank tensor modeling for streaming data analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1061–1075, 2019.

[106] Y. Hu, A. Qu *et al.*, "Streaming data preprocessing via online tensor recovery for large environmental sensor networks," *ACM Trans. Knowl. Disc. Data*, vol. 16, no. 6, pp. 1–24, 2022.

[107] L. T. Thanh *et al.*, "Robust online Tucker dictionary learning from multidimensional data streams," in *APSIPA ASC*, 2022, pp. 1812–1817.

[108] D. G. Chachlakis *et al.*, "L1-norm Tucker tensor decomposition," *IEEE Access*, vol. 7, pp. 178 454–178 465, 2019.

[109] D. Kressner *et al.*, "Low-rank tensor completion by Riemannian optimization," *BIT Numer. Math.*, vol. 54, no. 2, pp. 447–468, 2014.

[110] L. T. Thanh *et al.*, "Adaptive algorithms for tracking tensor-train decomposition of streaming tensors," in *EUSIPCO*, 2020, pp. 995–999.

[111] ——, "Streaming tensor-train decomposition with missing data," *TechRxiv*, 2022.

[112] ——, "Robust tensor tracking with missing data under tensor-train format," in *EUSIPCO*, 2022, pp. 832–836.

[113] H. Liu, L. T. Yang *et al.*, "An incremental tensor-train decomposition for cyber-physical-social big data," *IEEE Trans. Big Data*, vol. 7, no. 2, pp. 341–354, 2021.

[114] X. Wang, L. T. Yang *et al.*, "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.

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

[116] A. A. Rontogiannis *et al.*, "Online rank-revealing block-term tensor decomposition," in *ASILOMAR*, 2021, pp. 1678–1682.

[117] Z. Zhang *et al.*, "An online tensor robust PCA algorithm for sequential 2D data," in *IEEE ICASSP*, 2016, pp. 2434–2438.

[118] 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.

[119] L. Balzano *et al.*, "Online identification and tracking of subspaces from highly incomplete information," in *ALLERTON*, 2010, pp. 704–711.

[120] A. W. Smeulders *et al.*, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, 2013.

[121] X. Zhang *et al.*, "Visual tracking via dynamic tensor analysis with mean update," *Neurocomput.*, vol. 74, no. 17, pp. 3277–3285, 2011.

[122] W. Hu, J. Gao *et al.*, "Semi-supervised tensor-based graph embedding learning and its application to visual discriminant tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 1, pp. 172–188, 2016.

[123] S. Khan, G. Xu, R. Chan, and H. Yan, "An online spatio-temporal tensor learning model for visual tracking and its applications to facial expression recognition," *Expert Syst. Appl.*, vol. 90, pp. 427–438, 2017.

[124] A. Sobral, S. Javed, K. J. Soon *et al.*, "Online stochastic tensor decomposition for background subtraction in multispectral video sequences," in *IEEE ICCV*, 2015, pp. 946–953.

[125] M. M. Salut and D. V. Anderson, "Online tensor robust principal component analysis," *IEEE Access*, vol. 10, pp. 69 354–69 363, 2022.

[126] Y. He and G. K. Atia, "Patch tracking-based streaming tensor ring completion for visual data recovery," *IEEE Trans. Circuits Syst. Video Techn.*, 2022 (early access).

[127] B. Wen, Y. Li, L. Pfister, and Y. Bresler, "Joint adaptive sparsity and low-rankness on the fly: an online tensor reconstruction scheme for video denoising," in *IEEE Int. Conf. Comput. Vis.*, 2017, pp. 241–250.

[128] B. Wen, S. Ravishankar, and Y. Bresler, "VIDOSAT: High-dimensional sparsifying transform learning for online video denoising," *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 1691–1704, 2018.

[129] C. Min and G. Medioni, "Inferring segmented dense motion layers using 5D tensor voting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 9, pp. 1589–1602, 2008.

[130] D. S. Bassett and M. S. Gazzaniga, "Understanding complexity in the human brain," *Trends Cogn. Sci.*, vol. 15, no. 5, pp. 200–209, 2011.

[131] A. Cichocki *et al.*, "Noninvasive BCIs: Multiway signal-processing array decompositions," *Computer*, vol. 41, no. 10, pp. 34–42, 2008.

[132] N. Yeung, M. M. Botvinick, and J. D. Cohen, "The neural basis of error detection: Conflict monitoring and the error-related negativity," *Psychol. Rev.*, vol. 111, no. 4, p. 931, 2004.

[133] A. G. Mahyari, D. M. Zoltowski *et al.*, "A tensor decomposition-based approach for detecting dynamic network states from EEG," *IEEE Trans. Biomed. Eng.*, vol. 64, no. 1, pp. 225–237, 2016.

[134] E. Acar *et al.*, "Tracing evolving networks using tensor factorizations vs. ICA-based approaches," *Front. Neurosci.*, vol. 16, p. 861402, 2022.

[135] A. Fotouhi, E. Eqlimi, and B. Makkiabadi, "Evaluation of adaptive PARAFAC alogorithms for tracking of simulated moving brain sources," in *IEEE EMBC*, 2015, pp. 3819–3822.

[136] J. W. Meijs, O. W. Weier, M. J. Peters, and A. Van Oosterom, "On the numerical accuracy of the boundary element method (EEG application)," *IEEE Trans. Biomed. Eng.*, vol. 36, no. 10, pp. 1038–1049, 1989.

[137] A. Fotouhi, E. Eqlimi, and B. Makkiabadi, "Adaptive localization of moving EEG sources using augmented complex tensor factorization," in *IEEE TSP*, 2017, pp. 439–443.

[138] N. L. Trung *et al.*, "A non-linear tensor tracking algorithm for analysis of incomplete multi-channel EEG data," in *ISMICT*, 2018, pp. 1–6.

[139] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[140] L. Shi, A. Gangopadhyay, and V. P. Janeja, "STenSr: Spatio-temporal tensor streams for anomaly detection and pattern discovery," *Knowl. Inf. Syst.*, vol. 43, no. 2, pp. 333–353, 2015.

[141] H. Kasai, W. Kellerer, and M. Kleinsteuber, "Network volume anomaly detection and identification in large-scale networks based on online time-structured traffic tensor tracking," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 636–650, 2016.

[142] N. Cao, C. Lin *et al.*, "Voila: Visual anomaly detection and monitoring with streaming spatiotemporal data," *IEEE Trans. Vis. Comput. Graph*, vol. 24, no. 1, pp. 23–33, 2017.

[143] C. Lin, Q. Zhu *et al.*, "Anomaly detection in spatiotemporal data via regularized non-negative tensor analysis," *Data Min. Knowl. Discov.*, vol. 32, no. 4, pp. 1056–1073, 2018.

[144] M. Xu, J. Wu, H. Wang, and M. Cao, "Anomaly detection in road networks using sliding-window tensor factorization," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 12, pp. 4704–4713, 2019.

[145] J. Yuan, G. C. Alexandropoulos *et al.*, "Channel tracking for RIS-enabled multi-user SIMO systems in time-varying wireless channels," in *IEEE ICC*, 2022, pp. 145–150.

[146] K. Luo, X. Zhou *et al.*, "Sparse Bayes tensor and DOA tracking inspired channel estimation for V2X millimeter wave massive MIMO system," *Sensors*, vol. 21, no. 12, p. 4021, 2021.

[147] C. C. Garcez, D. V. de Lima *et al.*, "Tensor-based subspace tracking for time-delay estimation in GNSS multi-antenna receivers," *Sensors*, vol. 19, no. 23, p. 5076, 2019.

[148] Y.-R. Lin, K. S. Candan *et al.*, "SCENT: Scalable compressed monitoring of evolving multirelational social networks," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 7, no. 1, pp. 1–22, 2011.

[149] K. Shin, B. Hooi *et al.*, "Densealert: Incremental dense-subtensor detection in tensor streams," in *ACM KDD*, 2017, pp. 1057–1066.

[150] W. Sun and R. D. Braatz, "Opportunities in tensorial data analytics for chemical and biological manufacturing processes," *Comput. Chem. Eng.*, vol. 143, p. 107099, 2020.

[151] I. W. Sanou, R. Redon, X. Luciani, and S. Mounier, "Online nonnegative and sparse canonical polyadic decomposition of fluorescence tensors," *Chemometr. Intell. Lab. Syst.*, vol. 225, p. 104550, 2022.

[152] X. Meng *et al.*, "On-line monitoring of batch processes using a PARAFAC representation," *J. Chemometr.*, vol. 17, no. 1, pp. 65–81, 2003.

[153] S. Gourvenec, I. Stanimirova *et al.*, "Monitoring batch processes with the STATIS approach," *J. Chemometr.*, vol. 19, no. 5-7, pp. 288–300, 2005.

[154] H. Tan *et al.*, "Short-term traffic prediction based on dynamic tensor completion," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2123–2133, 2016.

[155] J. Wang *et al.*, "Understanding urban dynamics via context-aware tensor factorization with neighboring regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 11, pp. 2269–2283, 2019.

[156] B. Yang, "An extension of the PASTd algorithm to both rank and subspace tracking," *IEEE Signal Process. Lett.*, vol. 2, pp. 179–182, 1995.

[157] P. O. Perry *et al.*, "Minimax rank estimation for subspace tracking," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 3, pp. 504–513, 2010.

[158] C. D. Martin *et al.*, "An order-p tensor factorization with applications in imaging," *SIAM J. Sci. Comput.*, vol. 35, no. 1, pp. 474–490, 2013.

[159] Y. Wang and Y. Yang, "Hot-SVD: Higher-order t-singular value decomposition for tensors based on tensor-tensor product," *arXiv preprint arXiv:2204.10229*, 2022.

[160] A. Bifet, *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, 2010.

[161] A.-A. Saucan, T. Chonavel *et al.*, "CPHD-DOA tracking of multiple extended sonar targets in impulsive environments," *IEEE Trans. Signal Process.*, vol. 64, no. 5, pp. 1147–1160, 2016.

[162] S. Wang *et al.*, "New results on joint channel and impulsive noise estimation and tracking in underwater acoustic OFDM systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2601–2612, 2020.

[163] R. L. Das and M. Narwaria, "Lorentzian based adaptive filters for impulsive noise environments," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 64, no. 6, pp. 1529–1539, 2017.

[164] L. T. Thanh *et al.*, "Robust subspace tracking algorithms in signal processing: A brief survey," *REV J. Elect. Commun.*, vol. 11, no. 1-2, pp. 16–25, 2021.

[165] P. Hanggi and P. Jung, "Colored noise in dynamical systems," *Adv. Chem. Phys.*, vol. 89, pp. 239–326, 2007.

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

[167] M. Morup and L. K. Hansen, "Automatic relevance determination for multi-way models," *J. Chemom.*, vol. 23, no. 7-8, pp. 352–363, 2009.

[168] J. A. Bazerque, G. Mateos, and G. B. Giannakis, "Rank regularization and Bayesian inference for tensor completion and extrapolation," *IEEE Trans. Signal Process.*, vol. 61, no. 22, pp. 5689–5703, 2013.

[169] Q. Zhao, L. Zhang, and A. Cichocki, "Bayesian CP factorization of incomplete tensors with automatic rank determination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1751–1763, 2015.

[170] M. Che, A. Cichocki, and Y. Wei, "Neural networks for computing best rank-one approximations of tensors and its applications," *Neurocomput.*, vol. 267, pp. 114–133, 2017.

[171] M. Zhou, Y. Liu *et al.*, "Tensor rank learning in CP decomposition via convolutional neural network," *Signal Process. Image Commun.*, vol. 73, pp. 12–21, 2019.

[172] C. Hawkins *et al.*, "Towards compact neural networks via end-to-end training: A Bayesian tensor approach with automatic rank determination," *SIAM J. Math. Data Sci.*, vol. 4, no. 1, pp. 46–71, 2022.

[173] M. W. Mahoney, "Randomized algorithms for matrices and data," *Found. Trends Mach. Learn.*, vol. 3, no. 2, pp. 123–224, 2011.

[174] C. Ma, X. Yang, and H. Wang, "Randomized online CP decomposition," in *ICACI*, 2018, pp. 414–419.

[175] I. Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, 2020.

[176] J. H. Choi and S. Vishwanathan, "DFacTo: Distributed factorization of tensors," in *NeurIPS*, 2014, pp. 1–9.

[177] S. Smith, N. Ravindran *et al.*, "SPLATT: Efficient and parallel sparse tensor-matrix multiplication," in *IEEE IPDPS*, 2015, pp. 61–70.

[178] H. Li, Z. Li *et al.*, "SGD-Tucker: A novel stochastic optimization strategy for parallel sparse Tucker decomposition," *IEEE Trans. Parallel Distr. Syst.*, vol. 32, no. 7, pp. 1828–1841, 2021.

[179] H. Al Daas, G. Ballard *et al.*, "Parallel algorithms for tensor train arithmetic," *SIAM J. Sci. Comput.*, vol. 44, no. 1, pp. 25–53, 2022.

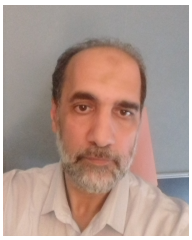
[180] N. Cohen and A. Shashua, "Convolutional rectifier networks as generalized tensor decompositions," in *ICML*, 2016, pp. 955–963.

[181] X. Wang, M. Che, and Y. Wei, "Tensor neural network models for tensor singular value decompositions," *Comput. Optim. Appl.*, vol. 75, no. 3, pp. 753–777, 2020.

[182] P. Brito, "Symbolic data analysis: Another look at the interaction of data mining and statistics," *Data Min. Knowl. Discov.*, vol. 4, no. 4, pp. 281–295, 2014.

[183] F. Di Mauro *et al.*, "Tensor-train decomposition in presence of interval-valued data," *IEEE Trans. Knowl. Data Eng.*, 2021 (early access).

Le Trung Thanh received the B.Sc. and M.Sc. degrees in Electronics and Communications from the VNU University of Engineering and Technology, Hanoi, Vietnam in 2016 and 2018 respectively, and the Ph.D. degree in Signal Processing from the University of Orleans, INSA CVL, PRISME, France in 2022. He is currently a postdoctoral reseacher at the University of Orleans, France. His research interests include signal processing, subspace tracking, tensor analysis, and system identification.



Karim Abed-Meraim (Fellow, IEEE) received the Engineering Degree from École Polytechnique, France, in 1990, the Engineering Degree from École Nationale Supérieure des Télécommunications (ENST), France, in 1992, the M.Sc. degree from Paris XI University, France, in 1992, and the Ph.D. degree in the field of signal processing and communications from ENST, in 1995. Currently he is a full professor at the University of Orléans, France (PRISME Laboratory). His research interests include signal processing for communications, adaptive filtering and tracking, array processing, and statistical performance analysis.

Nguyen Linh Trung obtained his B.Eng. and Ph.D. degrees, both in Electrical Engineering, from Queensland University of Technology, Brisbane, Australia, in 1998 and 2005. He joined VNU University of Engineering and Technology, Vietnam National University, Hanoi (VNU), in 2006. His broad technical interests include theory and methods of signal processing, including time-frequency signal analysis, blind source separation, compressive sampling, tensor-based signal analysis, graph signal processing, and application of signal processing in wireless communication, networking, medicine.

Adel Hafiane received an M.S. degree in embedded systems and information processing and a PhD from the University of Paris-Sud, in 2002 and 2005, respectively. He was a Postdoctoral Fellow in the Department of Computer Science at the University of Missouri, from 2007 to 2008. Since September 2008, he is an Associate Professor in the Electrical Engineering and Computer Science Departments at INSA CVL. His research interests include image processing theory and methods, computer vision and machine learning for several domains: medical, agriculture and robotics.