eOTD: An Efficient Online Tucker Decomposition for Higher Order Tensors

Houping Xiao
J. Mack Robinson College of Business
Georgia State University
hxiao@gsu.edu

Fei Wang
Weill Cornell Medical School
Cornell University
few2001@med.cornell.edu

Fenglong Ma, Jing Gao
Computer Science and Engineering
SUNY at Buffalo
{fenglong, jing}@buffalo.edu

***Abstract*—A tensor (i.e., an N-mode array) is a natural representation for multidimensional data. Tucker Decomposition (TD) is one of the most popular methods, and a series of batch TD algorithms have been extensively studied and widely applied in signal/image processing, bioinformatics, etc. However, in many applications, the large-scale tensor is dynamically evolving at all modes, which poses significant challenges for existing approaches to track the TD for such dynamic tensors. In this paper, we propose an efficient Online Tucker Decomposition (eOTD) approach to track the TD of dynamic tensors with an arbitrary number of modes. We first propose corollaries on the multiplication of block tensor matrix. Based on this corollary, eOTD allows us 1) to update the projection matrices using those projection matrices from the previous timestamp and the auxiliary matrices from the current timestamp, and 2) to update the core tensor by a sum of tensors that are obtained by multiplying smaller tensors with matrices. The auxiliary matrices are obtained by solving a series of least square regression tasks, not by performing Singular Value Decompositions (SVD). This overcomes the bottleneck in computation and storage caused by computing SVDs on large-scale data. A Modified Gram-Schmidt (MGS) process is further applied to orthonormalize the projection matrices. Theoretically, the output of the eOTD framework is guaranteed to be low-rank. We further prove that the MGS process will not increase Tucker decomposition error. Empirically, we demonstrate that the proposed eOTD achieves comparable accuracy with a significant speedup on both synthetic and real data, where the speedup can be more than 1,500 times on large-scale data.**

***Index Terms*—Tucker Decomposition, Low Rankness, Online Learning**

I. Introduction

In many applications, data can be naturally represented by a *tensor* (i.e., a multidimensional or N-mode array) [1]–[4], [6], [10], [18], [24], [25]. Decompositions of higher-order tensors (i.e., N-mode arrays with $N \geq 3$) are popular tools for analyses on multi-mode arrays, such as feature extraction, dimensionality reduction, and knowledge discovery. Two particular tensor decompositions, CANDECOMP/PARAFAC (CP) and Tucker Decomposition (TD), can be considered as higher-order extensions of matrix Singular Value Decomposition (SVD). CP decomposes a tensor as a sum of rank-1 tensors while the TD is a higher-order form of principal component analysis. Once the core tensor in TD is restricted to be diagonal, TD is degenerated to CP. Recently, TD has a broad range of applications in signal processing [3], [6], [18], anomaly detection [23]–[25], neuroscience [1], [2], [4], etc.

In the era of big data, *data*, represented by a *tensor*, is usually dynamically changing over time. Especially, in many applications all modes of a data tensor dynamically evolve. For example, in collaborative filtering for movie recommendation, we have the user-movie-date tensor where the (i, j, k)-th value represents the rating user i gave to movie j on the k-th day. TD can obtain the latent time-sensitive user and movie representations which can be used for rating prediction [9], [27]. Clearly, all three modes of this tensor evolve from time to time. Existing batch TD methods cannot handle data tensors that evolve on all modes due to the high computation and storage costs [5], [8], [11], [22], [25]. Therefore, an effective and efficient online Tucker Decomposition method is desired for decomposing real-time large-scale tensors.

Online tensor decomposition aims to dynamically update a tensor while preserving the low-rank structure. While online matrix decomposition has been intensively studied, online tensor decomposition remains largely under-explored. The problem is extremely challenging due to the inherent complexity of tensor analysis. For the low-rankness, although nuclear norm is widely used as the rank constraints and algorithms are developed to solve the problem, the solutions of these algorithms can easily get stuck in suboptimal ones. Moreover, solving an optimization problem with nuclear norm regularization is computationally expensive. It is difficult to apply nuclear norm regularization on large-scale applications in which TD is needed. There are also some online TD methods that works well in the scenario where data only evolves in one single mode [7], [20], [21]. However, in many real applications (e.g., movie recommendation), data continuously arrive at every mode. Existing online TD methods are not able to solve this problem where data evolve at all modes.

In this paper, we propose an efficient Online Tucker Decomposition (eOTD) approach to on-the-fly track the TD for dynamic large-scale tensors (i.e., tensors that have an arbitrary large order and evolve at all modes). We first introduce the *block tensor matrix multiplication* corollary. Based on this corollary, eOTD allows us (1) to update the projection matrices using the projection matrices from the previous timestamp and the auxiliary matrices from the current timestamp, and (2) to update the core tensor by a sum of tensors that are obtained by multiplying smaller tensors with matrices. During the update of projection matrices, the auxiliary matrices

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00180

are obtained by solving a series of least square regression tasks, instead of conducting singular value decomposition or eigenvalue decomposition are used by conventional methods. Consequently, we overcome the bottleneck in computation and storage caused by performing SVD on large-scale data. We further apply a Modified Gram-Schmidt (MGS) process for orthonormalization of the projection matrices.

Theoretically, we make the following contributions in this paper. First, we are the first to investigate block tensor matrix multiplication (i.e., Corollaries 3.1 and 3.2), which show the rule for conducting block multiplication between a tensor and a matrix and lay the foundation for the proposed eOTD. Second, the output of the proposed eOTD is guaranteed to be low-rank and is a proximal point. And finally, we prove that the MGS process, used for orthonormalization of projection matrices, does not increase the decomposition error.

Experimentally, on both synthetic and real data, we demonstrate that the proposed eOTD can achieve comparable performance with the most accurate method, i.e., Alternative Least Square method, while being computationally much more efficient. Specifically, on small and moderate datasets, eOTD is tens to hundreds of times faster than batch TD algorithm, while for large-scale datasets, the speedup can be more than 1,500 times. As a side outcome, eOTD can be a strategy to decompose a large-scale tensor in a batch way by a two-step procedure: (1) Conduct an SVD on a very small partition of the large-scale tensor, and (2) apply the proposed eOTD. In this way, the bottleneck of computation and memory caused by batch TD algorithms on large-scale tensors can be solved.

II. Definitions and Preliminaries

Following [10], we denote tensors with calligraphic letters (e.g. $\mathcal{X}$), matrices with uppercase bold letters (e.g. $\mathbf{U}$), row vectors with lowercase bold letters (e.g. x), and scalars with lowercase normal font (e.g. n). A tensor is said to evolve over time, if there is at least one mode whose size increases over time. For example, $\mathcal{X}^{(t)} \in \mathbb{R}^{N_1^{(t)} \cdots N_K^{(t)}}$ evolving on all modes means that $N_k^{(t+1)} \geq N_k^{(t)}, \forall k \in [K]$. Here, $[K] \equiv \{1, \cdots, K\}$. For simplification, we say that $(r_1, \cdots, r_K) \leqslant r$ if $r_k \leqslant r, \forall k \in [K]$, and $\mathcal{X} \times_1 \mathbf{U}_1 \times \cdots \times_K \mathbf{U}_K \equiv \mathcal{X} \times \{\mathbf{U}_k\}$.

Definition 2.1 (Mode-k Multiplication): The k-mode multiplication between a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_k \times \cdots \times N_K}$ and a matrix $\mathbf{U} \in \mathbb{R}^{I_k \times N_k}$ is defined as $(\mathcal{X} \times_k \mathbf{U})_{n_1 \cdots i_k \cdots n_K} = \sum_{n_k=1}^{N_k} \mathcal{X}_{n_1 \cdots n_k \cdots n_K} \mathbf{U}_{i_k n_k}$.
For distinct modes in a series of multiplications, $\mathcal{X} \times_k \mathbf{U} \times_{k'} \mathbf{U}' = \mathcal{X} \times_{k'} \mathbf{U}' \times_k \mathbf{U} (k' \neq k)$. If the modes are the same, then $\mathcal{X} \times_k \mathbf{U} \times_k \mathbf{U}' = \mathcal{X} \times_k (\mathbf{U}'\mathbf{U})$.

Definition 2.2 (Mode-k Matricization): The mode-k matricization or unfolding of an tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_K}$, denoted as $\mathcal{X}_{(k)}$, is obtained by treating k as the first mode of the matrix and orderly concatenating other modes.

Definition 2.3 (Frobenius Norm): The Frobenius norm of a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_K}$ is defined as $\|\mathcal{X}\|_F^2 = \sum_{n1=1}^{N_1} \cdots \sum_{n_K=1}^{N_K} \mathcal{X}_{n_1 \cdots n_K}^2$.

Definition 2.4 (Tensor Rank): Given $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_K}$, its rank is defined as $\text{rank}(\mathcal{X}) = (R_1, \cdots, R_K)$ s.t. $\text{rank}(\mathcal{X}_{(k)}) = R_k$, where $\text{rank}(\mathcal{X}_{(k)})$ is the matrix rank of the mode-k matricization $\mathcal{X}_{(k)}$ which is also referred to the k-rank of $\mathcal{X}$.

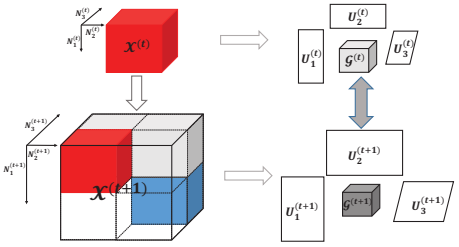


Fig. 1. Demo of online Tucker Decomposition for a third-order tensor.

III. Methodology

A. Tucker Decomposition

Given a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_K}$, its Low-Rank Tucker Decomposition is generally formulated as the following optimization problem with constraints:

$$\min_{\widehat{\mathcal{X}}} \|\mathcal{X} - \widehat{\mathcal{X}}\|_F^2, \text{s.t.} \widehat{\mathcal{X}} = \mathcal{G} \times \{\mathbf{U}_k\}, \text{and, rank}(\widehat{\mathcal{X}}) \leqslant r, \quad (1)$$

where $\mathcal{G}$ is a core tensor and $\{\mathbf{U}_k\}$ is a set of projection matrices. Usually, $\{\mathbf{U}_k\}$ are restricted to be unitary. However, there is some work [24], [25] which does not assume this condition. In this paper, we propose algorithms for both scenarios. For existing TD methods, such as HOSVD [22], ALS [5], also known as HOOI, TUCKALS3 [8], [11] etc, they suffer from the following drawbacks: 1) They are time-consuming when $\mathcal{X}$ is large-scale, as SVD operation is extremely expensive, let alone conducting SVD on every mode iteratively. 2) These algorithms need too much space in online settings, because they need to store all data at previous timestamps.

In the following, we will propose a novel framework for Tucker Decomposition which can overcome these challenges.

B. Online Tucker Decomposition

In real-world applications, data evolves over time at all modes. Denote a tensor stream until time t as $\{\mathcal{X}^{(t')} \in \mathbb{R}^{N_1^{t'} \times \cdots \times N_K^{t'}}\}_{t' \in [t]}$, and their TDs are $\{\mathcal{G}^{(t')}, \mathbf{U}_k^{(t')}\}$. $\mathcal{X}^{(t+1)}$ is the data at time $T+1$. The proposed eOTD framework can obtain the TD of $\mathcal{X}^{(t+1)}$ such that(1) it does not require to conduct SVD operation on all mode-k matricization matrices; and (2) instead of storing the data $\mathcal{X}^{(t')}$ in previous timestamp $t' \in [t]$, it only requires to store its TD result at previous time t, $\{\mathcal{G}^{(t)}, \mathbf{U}_k^{(t)}\}$, which needs much smaller space to store.

Problem Setting. Assume that there is a tensor stream, $\{\mathcal{X}_m^{(t)} \in \mathbb{R}^{N_1^t \times \cdots \times N_K^t} : m \in [M]\}$, along with its TD $\mathcal{X}^{(t)} = \mathcal{G}^{(t)} \times \{\mathbf{U}_k\}$. At time $t+1$, the tensor is $\mathcal{X}^{(t+1)} \in \mathbb{R}^{N_1^{t+1} \times \cdots \times N_K^{t+1}} = (\mathcal{X}_{i_1 \cdots i_K}^{(t+1)})_{i_k \in [2]}$, where $\mathcal{X}_{1 \cdots 1}^{(t+1)} = \mathcal{X}^{(t)}$ and the remaining subtensors newly arrive. Our goal is to find a new TD for $\mathcal{X}^{(t+1)}$ given $(\mathcal{X}_{i_1 \cdots i_k}^{(t+1)})_{(i_1, \cdots, i_K) \neq (1, \cdots, 1)}$, $\mathcal{G}^{(t)}$ and $\{\mathbf{U}_k^{(t)}\}$. The problem setting for a third-order tensor in a online setting is visually shown in Figure 1.

C. Basic ideas for eOTD

Corollary 3.1 (Block Tensor Matrix Multiplication i): Suppose $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_k \cdots \times N_K}$ and $\mathcal{Y} \in \mathbb{R}^{N_1 \times \cdots \times N_k' \cdots \times N_K}$ are two tensors, and $\mathbf{U}_k^x \in \mathbb{R}^{R_k \times N_k}$ and $\mathbf{U}_k^y \in \mathbb{R}^{R_k \times N_k'}$ are two

matrices. Let $\mathcal{Z} \in \mathbb{R}^{N_1 \times \cdots \times (N_k + N'_k) \cdots \times N_K}$ be a tensor which is concatenated by $\mathcal{X}$ and $\mathcal{Y}$ along mode-k. Thus,
1) if $k' = k$ and $\mathbf{U}^z_k \in \mathbb{R}^{R_k \times (N_k + N'_k)} = [\mathbf{U}^x_k \quad \mathbf{U}^y_k]$, then

$$\mathcal{Z} \times_{k'} \mathbf{U}^z_{k'} = \mathcal{X} \times_k \mathbf{U}^x_k + \mathcal{Y} \times_k \mathbf{U}^y_k; \tag{2}$$

2) if $k' \neq k$ and $\mathbf{U}^z_{k'} \in \mathbb{R}^{R_{k'} \times N_{k'}}$, then $\mathcal{Z} \times_k \mathbf{U}^z_{k'}$ is concatenated by $\mathcal{X} \times_{k'} \mathbf{U}^z_{k'}$ and $\mathcal{Y} \times_{k'} \mathbf{U}^z_{k'}$ along the mode-k.

Corollary 3.2 (Block Tensor Matrix Multiplication ii): Suppose $\mathcal{X} \in \mathbb{R}^{N_1 \times \cdots \times N_K}$ is a tensor. We split $\mathcal{X}$ into 2^K tensors, $(\mathcal{X}_{i_1 i_2 \cdots i_K})_{i_k \in [2]}$, such that $\sum_{i_k \in [2]} N_{k,i_k} = N_k$, and $\mathbf{U}^\top_k = [\mathbf{U}^\top_{k,1} \mathbf{U}^\top_{k,2}] \in \mathbb{R}^{(N_{k,1} + N_{k,2}) \times R_k}$. Then, we have that

$$\mathcal{X} \times \{\mathbf{U}_k\} = \sum_{(i_1, \cdots, i_K) \in [2]^K} \mathcal{X}_{i_1 i_2 \cdots i_K} \times \{\mathbf{U}_{k, i_k}\}. \tag{3}$$

As illustrates below, these two corollaries are the footstones of the proposed eOTD framework.

IV. eOTD FRAMEWORK

Given the TD (i.e., Tucker decomposition) at t-th timestamp, that is, $\mathcal{X}^{(t)} = \mathcal{G}^{(t)} \times \{\mathbf{U}^{(t)}_k\}$, at time $t+1$ we are going to (a) update $\{\mathbf{U}^{(t+1)}_k\}$ using the new coming data $(\mathcal{X}^{(t+1)}_{i_1 \cdots i_K})_{(i_1, \cdots, i_K) \neq (1, \cdots, 1)}$ and (b) update the core tensor.

The full tensor at $(t+1)$-th timestamp is $\mathcal{X}^{(t+1)} = (\mathcal{X}^{(t+1)}_{i_1 \cdots i_K})$, with $\mathcal{X}^{(t+1)}_{1 \cdots 1} = \mathcal{X}^{(t)} = \mathcal{G}^{(t)} \times \{\mathbf{U}^{(t)}_k\}$. For any other sub-tensor $\mathcal{X}^{(t+1)}_{i_1 \cdots i_K}$, it should be used to update one or more from $\{\mathbf{U}'_k \in \mathbb{R}^{(N^{(t+1)}_k - N^{(t)}_k) \times I_k}\}$ which is defined as auxiliary matrices. After spliting the full tensor into 2^K sub-tensors (as shown in Figure 1), the subindex of every sub-tensor implies which auxiliary matrices will be updated. If a subindex of a sub-tensor equals to 2, it will update one specific auxiliary matrix. For example, in the third-order tensor setting, $\mathcal{X}^{(t+1)}_{212}$ is used for the update of both $\mathbf{U}'_1$ and $\mathbf{U}'_3$. Next, we first illustrate the eOTD framework on third-order tensors and then extend it to higher-order tensors.

A. Third-order Tensors

Now, we show the update rules for both the projection matrices $\{\mathbf{U}^{(t+1)}_k\}$ and the core tensor $\mathcal{G}^{(t+1)}$ at time $t+1$.

1) Update Projection Matrices $\{\mathbf{U}_k\}$: At time $t+1$, the full tensor $\mathcal{X}^{(t+1)}$ is split into 2^3 subtensors $\{\mathcal{X}^{(t+1)}_{i_1 i_2 i_3}\}_{i_k \in [2]}$ such that $\mathcal{X}^{(t+1)}_{111} = \mathcal{X}^{(t)}$. Note that $\mathcal{X}^{(t+1)}_{111}$ is irrelevant to the auxiliary matrices $\{\mathbf{U}'_k\}$. The remaining subtensors, however, can be partitioned into three categories according to the number of subindices that equal to 2s (i.e., indicating how many auxiliary matrices to update). For third-order tensors, we have that $\mathbb{C}_1 = \{\mathcal{X}^{(t+1)}_{211}, \mathcal{X}^{(t+1)}_{121}, \mathcal{X}^{(t+1)}_{112}\}$, $\mathbb{C}_2 = \{\mathcal{X}^{(t+1)}_{221}, \mathcal{X}^{(t+1)}_{212}, \mathcal{X}^{(t+1)}_{122}\}$, and $\mathbb{C}_3 = \{\mathcal{X}^{(t+1)}_{222}\}$.

For each subtensor $\mathcal{X}^{(t+1)}_{i_1 i_2 i_3}$, Corollary 3.2 gives us that $\mathcal{X}^{(t+1)}_{i_1 i_2 i_3} = \mathcal{G}^{(t)} \times \{\mathbf{U}_k\}$, where $\mathbf{U}^{(t)}_k$ if $i_k =$ and $\mathbf{U}'_k$ if $i_k = 2$. Let $\mathcal{G}_{i_1, i_2, i_3} = \mathcal{G}^{(t)} \times \{\mathbf{U}_k\}$, where if a subindex is missing, it represents an auxiliary tensor constructed by multiplying $\mathcal{G}^{(t)}$ with the remaining projection matrices. For instance, $\mathcal{G}_{\cdot, i_2, i_3} = \mathcal{G}^{(t)} \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$. Next, we show the update rule of the auxiliary matrices using every subtensor category by category.

Update $\{\mathbf{U}'_k\}$ with $\mathbb{C}_1$. In $\mathbb{C}_1$, each subtensor is used once for updating a corresponding auxiliary matrix. We show the procedure of updating $\mathbf{U}'_1$ using $\mathcal{X}^{(t+1)}_{211}$; update rules for other auxiliary matrices are similar. We first construct an auxiliary tensor $\mathcal{G}_{\cdot,1,1} = \mathcal{G}^{(t)} \times_2 \mathbf{U}^{(t)}_2 \times_3 \mathbf{U}^{(t)}_3$. Then, we update $\mathbf{U}'_1$:

$$\mathbf{U}'_1 \leftarrow (\mathcal{X}^{(t+1)}_{211})_{(1)} (\mathcal{G}_{\cdot,1,1\,(1)})^\dagger, \tag{4}$$

where † denotes matrix pseudo-inverse. Similarly,

$$\mathbf{U}'_2 \leftarrow (\mathcal{X}^{(t+1)}_{121})_{(2)} (\mathcal{G}_{1,\cdot,1\,(2)})^\dagger, \text{ and } \mathbf{U}'_3 \leftarrow (\mathcal{X}^{(t+1)}_{112})_{(3)} (\mathcal{G}_{1,1,\cdot\,(3)})^\dagger;$$

where $\mathcal{G}_{1,\cdot,1} = \mathcal{G}^{(t)} \times_1 \mathbf{U}^{(t)}_1 \times_3 \mathbf{U}^{(t)}_3$ and $\mathcal{G}_{1,1,\cdot} = \mathcal{G}^{(t)} \times_1 \mathbf{U}^{(t)}_1 \times_2 \mathbf{U}^{(t)}_2$. Although all auxiliary matrices have been updated using sub-tensors within $\mathbb{C}_1$, we need to update them again using sub-tensors in $\mathbb{C}_2$ and $\mathbb{C}_3$, because all these sub-tensors contribute to one or more auxiliary matrices.

Update $\{\mathbf{U}'_k\}$ with $\mathbb{C}_2$. Every subtensor in $\mathbb{C}_2$ is used twice for updating two corresponding auxiliary matrices. For example, $\mathcal{X}^{(t+1)}_{221}$ is used for updating both $\mathbf{U}'_1$ and $\mathbf{U}'_2$. To update $\mathbf{U}'_1$, we use the auxiliary matrix $\mathbf{U}'_2$ obtained from $\mathbb{C}_1$. Namely, $\mathcal{G}_{\cdot,2,1} = \mathcal{G}^{(t)} \times_2 \mathbf{U}'_2 \times_3 \mathbf{U}^{(t)}_3$. Then,

$$\mathbf{U}'^{\text{new}}_1 \leftarrow \alpha \mathbf{U}'^{\text{old}}_1 + (1-\alpha)(\mathcal{X}^{(t+1)}_{211})_{(1)} (\mathcal{G}_{\cdot,2,1\,(1)})^\dagger. \tag{5}$$

Here, α is a memory rate deciding how much information is inherited from previous step. For $\mathbf{U}'_2$, we have that

$$\mathbf{U}'^{\text{new}}_2 \leftarrow \alpha \mathbf{U}'^{\text{old}}_2 + (1-\alpha)(\mathcal{X}^{(t+1)}_{211})_{(2)} (\mathcal{G}_{2,\cdot,1\,(2)})^\dagger, \tag{6}$$

where $\mathcal{G}_{2,\cdot,1} = \mathcal{G}^{(t)} \times_1 \mathbf{U}'_1 \times_3 \mathbf{U}^{(t)}_3$. Similar procedure is conducted for $\mathcal{X}^{(t+1)}_{212}$ and $\mathcal{X}^{(t+1)}_{122}$. Namely,

$$\mathbf{U}'^{\text{new}}_1 \leftarrow \alpha \mathbf{U}'^{\text{old}}_1 + (1-\alpha)(\mathcal{X}^{(t+1)}_{212})_{(1)} (\mathcal{G}_{\cdot,1,2\,(1)})^\dagger, \tag{7}$$

$$\mathbf{U}'^{\text{new}}_3 \leftarrow \alpha \mathbf{U}'^{\text{old}}_2 + (1-\alpha)(\mathcal{X}^{(t+1)}_{212})_{(3)} (\mathcal{G}_{2,1,\cdot\,(3)})^\dagger, \tag{8}$$

where $\mathcal{G}_{\cdot,1,2} = \mathcal{G}^{(t)} \times_2 \mathbf{U}^{(t)}_2 \times_3 \mathbf{U}'_3$; $\mathcal{G}_{2,1,\cdot} = \mathcal{G}^{(t)} \times_1 \mathbf{U}'_1 \times_2 \mathbf{U}^{(t)}_2$.

$$\mathbf{U}'^{\text{new}}_2 \leftarrow \alpha \mathbf{U}'^{\text{old}}_2 + (1-\alpha)(\mathcal{X}^{(t+1)}_{122})_{(2)} (\mathcal{G}_{1,\cdot,2\,(2)})^\dagger, \tag{9}$$

$$\mathbf{U}'^{\text{new}}_3 \leftarrow \alpha \mathbf{U}'^{\text{old}}_2 + (1-\alpha)(\mathcal{X}^{(t+1)}_{122})_{(3)} (\mathcal{G}_{1,2,\cdot\,(3)})^\dagger, \tag{10}$$

where $\mathcal{G}_{1,\cdot,2} = \mathcal{G}^{(t)} \times_1 \mathbf{U}^{(t)}_1 \times_3 \mathbf{U}'_3$; $\mathcal{G}_{1,2,\cdot} = \mathcal{G}^{(t)} \times_1 \mathbf{U}^{(t)}_1 \times_2 \mathbf{U}'_2$.

Update $\{\mathbf{U}'_k\}$ with $\mathbb{C}_3$. Note that $\mathcal{X}^{(t+1)}_{222} = \mathcal{G}^{(t)} \times \{\mathbf{U}'_k\}$ is related to all auxiliary matrices. Following the same procedure, we iteratively update all auxiliary matrices as follows:

$$\mathbf{U}'^{\text{new}}_1 \leftarrow \alpha \mathbf{U}'^{\text{old}}_1 + (1-\alpha)(\mathcal{X}^{(t+1)}_{222})_{(1)} (\mathcal{G}_{\cdot,2,2\,(1)})^\dagger; \tag{11}$$

$$\mathbf{U}'^{\text{new}}_2 \leftarrow \alpha \mathbf{U}'^{\text{old}}_2 + (1-\alpha)(\mathcal{X}^{(t+1)}_{222})_{(2)} (\mathcal{G}_{2,\cdot,2\,(2)})^\dagger; \tag{12}$$

$$\mathbf{U}'^{\text{new}}_3 \leftarrow \alpha \mathbf{U}'^{\text{old}}_3 + (1-\alpha)(\mathcal{X}^{(t+1)}_{222})_{(3)} (\mathcal{G}_{2,2,\cdot\,(3)})^\dagger; \tag{13}$$

where $\mathcal{G}_{\cdot,2,2} = \mathcal{G}^{(t)} \times_2 \mathbf{U}'_2 \times_3 \mathbf{U}'_3$, $\mathcal{G}_{2,\cdot,2} = \mathcal{G}^{(t)} \times_1 \mathbf{U}'_1 \times_3 \mathbf{U}'_3$, and $\mathcal{G}_{2,2,\cdot} = \mathcal{G}^{(t)} \times_1 \mathbf{U}'_1 \times_2 \mathbf{U}'_3$.

Update $\mathbf{U}^{(t+1)}_k$. At time $t+1$, the new projection matrices $\{\mathbf{U}^{(t+1)}_k\}$ are updated based on $\{\mathbf{U}^{(t)}_k\}$ from the previous step and the auxiliary matrices $\{\mathbf{U}'_k\}$ from the current step. One strategy to update $\mathbf{U}^{(t+1)}_k$ $(\forall k \in [K])$ is to concatenate $\mathbf{U}^{(t)}_k$ and $\mathbf{U}'_k$ along the second mode. Namely, $(\mathbf{V}^{(t+1)}_k)^\top = [(\mathbf{U}^{(t)}_k)^\top \ (\mathbf{U}'_k)^\top] \in \mathbb{R}^{N^{(t+1)}_k \times I_k} (\forall k \in [K])$. However, $\{\mathbf{V}^{(t+1)}_k\}$ are not guaranteed to be unitary. Modified Gram-Schmidt (MGS) process is further conducted on $\mathbf{V}^{(t+1)}_k$ to create orthonormal projection matrices $\{\mathbf{U}^{(t+1)}_k\}$.

Throughout the update of the projection matrices, we only apply cheap tensor matrix multiplication and matrix pseudo-inverse instead of conducting the expensive SVD operations on large matrices. This makes it easy for the proposed framework to apply to large-scale applications.

TABLE I
COMPUTATION AND STORAGE COMPARISON

Method	Computation	Storage
eOTD	$\mathcal{O}(rd^{2(m-1)}N^{2(K-m)})$	$\mathcal{O}(KdN^{K-1}+d^K+r^K)$
HOSVD	$\mathcal{O}(K(N+d)^{2k-1})$	$\mathcal{O}((N+d)^K$

2) Update $\mathcal{G}$: Given the new coming data at $(t+1)$-th timestamp $(\mathcal{X}^{(t+1)}_{ijk})_{(i,j,k)\neq(1,1,1)}$, associated with $\mathcal{G}^{(t)}$ and $\{\mathbf{U}^{(t+1)}_k\}$, to update $\mathcal{G}^{(t+1)}$, we first split $\mathbf{U}^{(t+1)}_k$ into two matrices: $\mathbf{U}^{(t+1)}_{k,1} \in \mathbb{R}^{N^{(t)}\times I_k}$ and $\mathbf{U}^{(t+1)}_{k,2} \in \mathbb{R}^{(N^{(t+1)}-N^{(t)})\times I_k}$. As $\{\mathbf{U}^{(t+1)}_k\}$ are unitary matrices and according to Corollary 3.2, $\mathcal{X}^{(t+1)} = \mathcal{G}^{(t+1)} \times \{\mathbf{U}^{(t+1)}_k\}$ implies that

$$\begin{aligned}\mathcal{G}^{(t+1)} &= \mathcal{X}^{(t+1)} \times \{\mathbf{U}^{(t+1)^\top}_k\} \\ &= \mathcal{G}^{(t)} \times \{\mathbf{U}^{(t+1)^\top}_{k,1}\mathbf{U}^{(t)}_k\} + \sum_{(i_1,i_2,i_3)\neq(1,1,1)} \mathcal{X}^{(t+1)}_{i_1i_2i_3} \times \{\mathbf{U}^{(t+1)^\top}_{k,i_k}\}.\end{aligned}$$

It shows that the core tensor is updated by taking the sum of tensors which are obtained by multiplying the new coming subtensors with the submatrices of projection matrices. These subtensors is much smaller compared with that of the original tensor. It is proven to be faster than multiplying the original tensors with the transport of the updated projection matrices.

B. Extension to Higher-order Tensors

Let $\mathcal{X}^{(t)} \in \mathbb{R}^{N_{1,1}\times\cdots\times N_{K,1}}$ and its TD $\mathcal{X}^{(t)} = \mathcal{G}^{(t)}\times\{\mathbf{U}^{(t)}_k\}$. $\mathcal{X}^{(t+1)} \in \mathbb{R}^{(N_{1,1}+N_{1,2})\times\cdots\times(N_{K,1}+N_{K,2})}$ is split into 2^K subtensors $(\mathcal{X}^{(t+1)}_{i_1\cdots i_K})_{i_k\in[2]}$ such that $\mathcal{X}^{(t+1)}_{1\cdots1} = \mathcal{X}^{(t)}$.

1) Update Projection Matrices $\{\mathbf{U}_k\}$: The procedure is similar to that in the third-order tensor scenario. However, in the higher-order tensor setting, there are K categories $\{\mathbb{C}_m\}_{m\in[K]}$. Every subtensor within $\mathbb{C}_m$ will be used m times for auxiliary matrix update. For $\mathcal{X}^{(t+1)}_{i_1\cdots i_K}$ in $\mathbb{C}_m$, Corollary 3.2 shows that $\mathcal{X}^{(t+1)}_{i_1\cdots i_K} = \mathcal{G}^{(t)} \times \{\mathbf{U}_k\}$, where $\mathbf{U}_k = \mathbf{U}^{(t)}_k$, if $i_k = 1$ or $\mathbf{U}_k = \mathbf{U}'_k$ if $i_k = 2$. Denote the to-be-updated subindex set as $\mathbb{S} = \{i_{k_1}, \cdots, i_{k_m}\}$. For each subindex $i_{k_j} \in \mathbb{S}$, the update rules are as follows:

$$\mathbf{U}'^{\text{new}}_{i_{k_j}} \leftarrow \alpha\mathbf{U}'^{\text{old}}_{i_{k_j}} + (1-\alpha)(\mathcal{X}^{(t+1)}_{i_1\cdots i_K})_{(i_{k_j})}(\mathcal{G}_{i_{k_j}})^\dagger_{(i_{k_j})}, \tag{14}$$

where $\mathcal{G}_{i_{k_j}} \triangleq \mathcal{G}_{i_1,\cdots,i_{k_j},\cdots,i_K} = \mathcal{G}^{(t)} \times_{-i_{k_j}} \{\mathbf{U}_k\}$. After obtaining $\{\mathbf{U}'_k\}$, we augment them to $\{\mathbf{U}^{(t)}_k\}$ and then apply the MGS to orthogonalize and normalize the constructed matrices for $\{\mathbf{U}^{(t+1)}_k\}$.

2) Update $\mathcal{G}$: Based on the split of the tensor at current time $t+1$, $\mathbf{U}^{(t+1)}_k$ can also be split into two matrices: $\mathbf{U}^{(t+1)}_{k,1} \in \mathbb{R}^{N^{(t)}\times I_k}$ and $\mathbf{U}^{(t+1)}_{k,2} \in \mathbb{R}^{(N^{(t+1)}-N^{(t)})\times I_k}$. Corollary 3.2 and the fact that $\{\mathbf{U}^{(t+1)}_k\}$ are unitary matrices yield that

$$\begin{aligned}\mathcal{G}^{(t+1)} &= \mathcal{X}^{(t+1)} \times \{\mathbf{U}^{(t+1)^\top}_k\} \\ &= \mathcal{G}^{(t)} \times \{\mathbf{U}^{(t+1)^\top}_{k,1}\mathbf{U}^{(t)}_k\} + \sum_{(i_1,\cdots,i_K)\neq(1,\cdots,1)} \mathcal{X}^{(t+1)}_{i_1\cdots i_K} \times \{\mathbf{U}^{(t+1)^\top}_{k,i_k}\}\end{aligned}.$$

C. Complexity Analysis

We conduct the complexity comparison between eOTD and HOSVD in terms of computation and storage. For ALS and Tuck-ALS3, as they usually adopt the results from HOSVD as initialization, we do not include them in this comparison. We show the advantages of the proposed eOTD over HOSVD in Table I in both computation and storage.

V. THEORETICAL ANALYSIS OF EOTD

Now, we provide theoretical analysis of eOTD in terms of low-rankness, the behavior of the Modified Gram-Schmidt process as well as approximation accuracy.

Lemma 5.1 (Low-Rankness): Suppose $\mathcal{X}^{(t)} = \mathcal{G}^{(t)}\times\{\mathbf{U}^{(t)}_k\}$. Let the new full tensor be $\mathcal{X}^{(t+1)} = (\mathcal{X}^{(t+1)}_{i_1\cdots i_K})$ s.t. $\mathcal{X}^{(t+1)}_{1\cdots1} = \mathcal{X}^{(t)}$. Thus, for the reconstructed tensor $\widetilde{\mathcal{X}}^{(t+1)}$ returned by the eOTD, its k-rank is no greater than r for all modes.

Lemma 5.2: Suppose $\{\mathbf{V}^{(t+1)}_k\}$ is concatenated by $\{\mathbf{U}^{(t)}_k\}$ and $\{\mathbf{U}'_k\}$ along the second mode, then we have unique $\{\mathbf{U}^{(t+1)}_k\}$ s.t. (1) $\{\mathbf{U}^{(t+1)}_k\}$ are orthogonormal matrices obtained via conducting Modified Gram-Schmidt process on $\{\mathbf{V}^{(t+1)}_k\}$, and (2) the reconstructed tensors on $\{\mathbf{U}^{(t+1)}_k\}$ and $\{\mathbf{V}^{(t+1)}_k\}$ are $\widetilde{\mathcal{X}}^{(t+1)} = \widetilde{\mathcal{G}}^{(t+1)} \times \{\mathbf{U}^{(t+1)}_k\}$ and $\widehat{\mathcal{X}}^{(t+1)} = \widehat{\mathcal{G}}^{(t+1)} \times \{\mathbf{V}^{(t+1)}_k\}$, respectively, where $\widetilde{\mathcal{G}}^{(t+1)} = \mathcal{X}^{(t+1)} \times \{\mathbf{U}^{(t+1)^\top}_k\}$ and $\widehat{\mathcal{G}}^{(t+1)} = \mathcal{X}^{(t+1)} \times \{(\mathbf{V}^{(t+1)}_k)^{-1}_{\text{left}}\}$. Also, we have that $\|\widetilde{\mathcal{X}}^{(t+1)} - \mathcal{X}^{(t+1)}\|^2_F = \|\widehat{\mathcal{X}}^{(t+1)} - \mathcal{X}^{(t+1)}\|^2_F$.
Lemma 5.2 shows that the Modified Grem-Schmidt process does not introduce extra error for the reconstruction of a tensor.

Lemma 5.3: Given a tensor $\mathcal{X}^{(t+1)}$, the proposed eOTD computes a proximal point $\widetilde{\mathcal{X}}^{(t+1)}$ such that

$$\widetilde{\mathcal{X}}^{(t+1)} = \arg\min_{\widehat{\mathcal{X}}} \|\widehat{\mathcal{X}}^{(t+1)} - \mathcal{X}^{(t+1)}\|^2_F, \text{ s.t. rank}(\widehat{\mathcal{X}}^{(t+1)}) \leq R.$$

VI. EXPERIMENTS

A. Experiment Setup

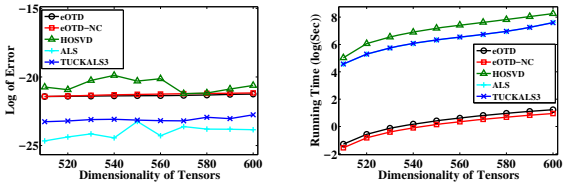
Methods. We compare our algorithms with the following baselines: HOSVD [22], ALS (i.e., HOOI) [5], and TUCK-ALS3 [8], [11]. The details of the baselines are deferred to the related work (Section VII). We propose two online Tucker decomposition approaches: eOTD and eOTD-NC. For eOTD, the projection matrices are orthonormal via the modified Grem-Schmidt process. For eOTD-NC, however, the projection matrices are not restricted to be orthonormal.

Evaluation Measure. The reconstruction error is defined as $\text{error}(\widehat{\mathcal{X}}, \mathcal{X}) = \frac{\|\mathcal{X}-\widehat{\mathcal{X}}\|^2_F}{\|\mathcal{X}\|^2_F}$. As the scale of the measure is small, we report the logarithm value of the error in following experiments. Threfore, the error and Logarithm value of error is interchangeably used in the experiments. The lower the measure, the closer the reconstructed tensor to the original one, the better the TD method.

B. Experiments on Synthetic Data

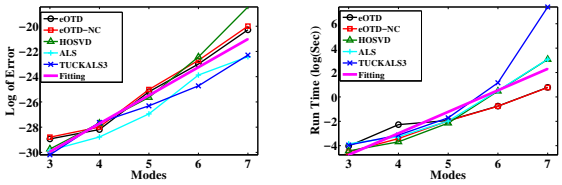
In this part, we report the accuracy and efficiency of the proposed eOTD(-NC) approaches, compared with baselines in two scenarios: 1) the dimensionality of the new coming tensor is much less than that of the previous one (i.e., $N \gg d$), and 2) the dimensionality of new coming tensor is much larger than that of the previous one (i.e., $N \ll d$).

Data Generation. For experiments when $N \gg d$, we first generate a three-mode tensor $\mathcal{X} \in \mathbb{R}^{600\times600\times600}$ whose rank is $(3,3,3)$. $\mathcal{X}_0$ is set to be $(500,500,500)$ in size. We randomly generate a tensor stream with 10 timestamps by a stepsize of 10 on all modes from $\mathcal{X}$. When $N \ll d$, we generate a three-mode tensor $\mathcal{X} \in \mathbb{R}^{620\times620\times620}$ with the rank of $(3,3,3)$. $\mathcal{X}_0$ is set to be $(20,20,20)$ in size and there are 10 stream



(a) Reconstruction Error (b) Run Time

Fig. 2. Synthetic data where $\boldsymbol{N \gg d}$ over 10 timestamps.



(a) Reconstruction Error (b) Running Time

Fig. 3. Experiments w.r.t. # of tensor modes.

tensors at different timestamps by a stepsize of 60 on all modes. Assuming the rank is known, we run all TD methods.

Result Analysis. Figure 2 presents the results for the proposed eOTD(-NC) approaches and baselines on synthetic data when $N \gg d$. eOTD(-NC) can achieve comparable accuracy compared with baselines, although eOTD(-NC) access less original data than batch baselines. In Figure 2(a), the negative value of y-axis actually indicates that the reconstruction error is very small because it is in log scale. eOTD(-NC) can even achieve higher accuracy than HOSVD. The reason could be the following: At each timestamp, we add additional rows into the projection matrices. It prevents the algorithm from being stuck in local optimal which is common in HOSVD. Figure 2(b) shows the running time for all TD methods. We can see that the running time of the eOTD(-NC) algorithms is much less than that of baselines. Actually, our methods are tens to hundreds of times faster than baselines. Especially, as the tensor continues to increase, the speedup can be more than a thousand times. We can conduct similar conclusions on the scenario when $N \ll d$ (the details of experiments are omitted due to page limit). Experimental results show that the proposed eOTD(-NC) can achieve comparable accuracy with a significant speedup in either $N \gg d$ or $N \ll d$ scenarios.

Performance w.r.t. # of Modes K. We conduct experiments to show the performance of the proposed algorithms on higher-order tensors. A tensor stream with 2 timestamps is generated, that is, $\mathcal{X}^0 \in \mathbb{R}^{8\times\cdots\times 8}$and $\mathcal{X}^1 \in \mathbb{R}^{10\times\cdots\times 10}$ where the number of modes K increases from 3 to 7. The logarithm value of reconstruction error and running time are respectively presented in Figures 3(a) and 3(b), along with a linear fitting line. For both the reconstruction error and running time, they almost exponentially increase as the number of modes of a tensor increases. As the figures show the logarithm values, the results imply that they are exponential to the number of modes. Actually, eOTD(-NC) can achieve comparable accuracy when compared with baselines. As the dimensionality on each mode is small (i.e., 10 in total), the running time of baselines is comparable or even lower than the eOTD when $K \leq 5$. However, when $K \geq 6$, baselines take much more time.

C. Experiments on Real Data

We conduct experiments on three real world applications and their data are listed in Table II. The details for all datasets are omitted due to the page limit, but they can be obtained on requests or be found in the corresponding references. For each real application, we split the data into tensor streams with T timestamps where each tensor is randomly chosen from the original input data. The goal is to evaluate the performance of the proposed algorithms and baselines at the final timestamp.

Result Analysis. For all applications, the proposed eOTD(-NC) approaches show very promising results in accuracy as well as efficiency. All baselines have almost the same accuracy which is slightly better than that of the proposed eOTD(-NC) algorithms. However, eOTD(-NC) algorithms have shown a significant speedup in running time. It is 662 times (on HSC-HDA), and even 1,628 times (on COIL-100) faster than the baselines. The TUCKALS3 algorithm is the worst in terms of efficiency in all applications. The underlying reason is that it needs to compute the Kronecker product over $K-1$ projections on each mode. Although both proposed algorithms can achieve comparable results with baselines, eOTD algorithm outperforms eOTD(-NC). This implies that the modified Gram-Schmidt process can help to increase the accuracy.

VII. Related Work

Batch Tucker Decomposition (BTD) has witnessed great success in real applications. In [22], "Tucker1" method, known as the higher-order singular value decomposition (HOSVD) algorithm, aims to find the components that best capture the variation in mode k, independent of the other modes. Later in [8], [11], based on the alternative least square techniques, Kroonenberg and De Leeuwcalled propose TUCKALS3. In [5], De Lathauwer et al. propose more efficient techniques to calculate the factor matrices, called higher-order orthogonal iteration (HOOI). HOOI computes only the dominant singular vectors of $\mathcal{X}_{(k)}$ using a SVD rather than an eigenvalue decomposition or even just computing an orthonormal basis of the dominant subspace. However, these BTD methods have issues in terms of computation and storage. In contrast, the proposed eOTD(-NC) is efficient in both computation and storage.

[20], [21], propose both offline tensor analysis (OTA) for a tensor sequence and incremental tensor analysis (ITA) for a tensor stream. In OTA, projection matrices are updated by diagonalizing the variance matrix. To overcome the issues of computation and space, the approach conducts ITA. There are several variants of ITA in [21], such as dynamic tensor analysis, streaming tensor analysis, and window-based tensor analysis. They use different methods to incrementally update the variance matrix. As all methods need to diagonalize the variance on each mode, they are inefficient in computation. Additionally, another trend for improving the efficiency of BTD in online setting is to replace SVD by incremental SVD methods. Several applications of this idea can be found in computer vision and anomaly detection [7], [12], [15], [19]. However, all these methods can only handle the case where data only evolve at one single mode. In contrast, the

TABLE II
REAL DATASETS DESCRIPTION AND COMPARISON BETWEEN eOTD(-NC) WITH BASELINES. THE RATIOS OF THEIR RUNNING TIME BETWEEN BASELINES AND THE PROPOSED eOTD ALGORITHM ARE SHOWN IN PARENTHESIS.

Dataset	Dimensionality	Source	Measure	Method				
				HOSVD	ALS	TUCKALS3	eOTD-NC	eOTD
Climate Data	$19 \times 125 \times 12 \times 12$	[14] [13]	Error/log(Error) Running Time/s (Ratio)	-6.702 4.767(8.21)	-6.713 5.242(9.02)	-6.713 7.641(13.2)	-4.788 .5365(.923)	-6.784 .5810
PCOIL-20	$20 \times 72 \times 128 \times 128$	[17] [16]	Error/log(Error) Running Time/s (Ratio)	-8.229 108.1(40.1)	-8.241 110.7(41.2)	-8.241 130.3(48.5)	-8.032 2.689(1.00)	-8.032 2.684
UCOIL-20	$5 \times 72 \times 416 \times 448$	[17] [16]	Error/log(Error) Running Time/s (Ratio)	-8.257 27.52(9.51)	-8.263 29.25(10.1)	-8.263 298.1(103)	-7.406 2.863 (.990)	-7.966 2.894
USC-HAD	$12 \times 5 \times 14 \times 500 \times 6$	[26]	Error/log(Error) Running Time/s (Ratio)	-12.34 4635(891)	-12.35 4524(870)	-12.35 8464(1628)	-12.25 4.431(.852)	-13.39 5.198

proposed frameworks only involve tensor-matrix multiplication and pseudo-inverse of small matrices which make them computationally efficient. Moreover, the proposed frameworks can easily handle tensors with arbitrarily large modes.

VIII. CONCLUSIONS

In this paper, we propose a simple and efficient framework, named eOTD, to track Tucker decomposition for a tensor stream. We introduce the block tensor matrix multiplication to build rules for updating the projection matrices as well as the core tensor. More specifically, the auxiliary matrices are obtained by solving a series of optimization problems, where close-form solutions are available. The projection matrices are then obtained by augment, orthorgonalization, and normalization via a Modified Gram-Schmidt process. Theoretically, we prove that the reconstructed tensor obtained by the proposed eOTD framework is guaranteed to be low-rank as well as a proximal point. Moreover, we show that the Modified Gram-Schmidt process will not introduce extra error. We demonstrate that the proposed algorithms can produce comparable accuracy and significantly reduce the computational and storage costs.

IX. ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and suggestions. This work is supported in part by the ONR N00014-18-1-2585, NSF IIS-1716432, IIS-1750326 and IIS-1747614. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ONR or NSF.

REFERENCES

[1] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener. Multiway analysis of epilepsy tensors. *Bioinformatics*, 23(13):i10–i18, 2007.

[2] C. F. Beckmann and S. M. Smith. Tensorial extensions of independent component analysis for multisubject fmri analysis. *Neuroimage*, 25(1):294–311, 2005.

[3] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, 2015.

[4] F. Cong, Q.-H. Lin, L.-D. Kuang, X.-F. Gong, P. Astikainen, and T. Ristaniemi. Tensor decomposition of eeg signals: a brief review. *Neuroscience Methods*, 248:59–69, 2015.

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

[6] L. De Lathauwer and J. Vandewalle. Dimensionality reduction in higher-order signal processing and rank-(r1, r2,..., rn) reduction in multilinear algebra. *Linear Algebra and its Applications*, 391:31–55, 2004.

[7] W. Hu, X. Li, X. Zhang, X. Shi, S. Maybank, and Z. Zhang. Incremental tensor subspace learning and its applications to foreground segmentation and tracking. *IJCV*, 91(3):303–327, 2011.

[8] A. Kapteyn, H. Neudecker, and T. Wansbeek. An approach ton-mode components analysis. *Psychometrika*, 51(2):269–275, 1986.

[9] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proc. of RecSys*, pages 79–86, 2010.

[10] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[11] P. M. Kroonenberg and J. De Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, 1980.

[12] X. Li, W. Hu, Z. Zhang, X. Zhang, and G. Luo. Robust visual tracking based on incremental tensor subspace learning. In *Proc. of ICCV*, pages 1–8, 2007.

[13] Y. Liu, A. Niculescu-Mizil, A. C. Lozano, and Y. Lu. Learning temporal causal graphs for relational time-series analysis. In *Proc. of ICML*, pages 687–694, 2010.

[14] A. C. Lozano, H. Li, A. Niculescu-Mizil, Y. Liu, C. Perlich, J. Hosking, and N. Abe. Spatial-temporal causal modeling for climate change attribution. In *Proc. of KDD*, pages 587–596, 2009.

[15] X. Ma, D. Schonfeld, and A. Khokhar. Dynamic updating and downdating matrix svd and tensor hosvd for adaptive indexing and retrieval of motion trajectories. In *Proc. of ICASSP*, pages 1129–1132, 2009.

[16] S. Nayar, S. Nene, and H. Murase. Columbia object image library (coil 100). *Technical report CUCS-006-96*, 1996.

[17] S. A. Nene, S. K. Nayar, H. Murase, et al. Columbia object image library (coil-20). *Technical report CUCS-005-96*, 1996.

[18] D. Nion and N. D. Sidiropoulos. Tensor algebra and multidimensional harmonic retrieval in signal processing for mimo radar. *IEEE Transactions on Signal Processing*, 58(11):5693–5705, 2010.

[19] A. Sobral, C. G. Baker, T. Bouwmans, and E.-h. Zahzah. Incremental and multi-feature tensor subspace learning applied for background modeling and subtraction. In *Proc. of ICIAR*, pages 94–103, 2014.

[20] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proc. of KDD*, pages 374–383, 2006.

[21] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Incremental tensor analysis: Theory and applications. *TKDD*, 2(3):11, 2008.

[22] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[23] H. Xiao. *Multi-sourced Information Trustworthiness Analysis: Applications and Theory*. PhD thesis, SUNY at Buffalo, 2018.

[24] H. Xiao, J. Gao, D. S. Turaga, L. H. Vu, and A. Biem. Temporal multi-view inconsistency detection for network traffic analysis. In *Proc. of WWW*, pages 455–465, 2015.

[25] H. Xiao, Y. Li, J. Gao, F. Wang, L. Ge, W. Fan, L. H. Vu, and D. S. Turaga. Believe it today or tomorrow? detecting untrustworthy information from dynamic multi-source data. In *Proc. of SDM*, 2015.

[26] M. Zhang and A. A. Sawchuk. Usc-had: a daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proc. of Ubicomp*, pages 1036–1043, 2012.

[27] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *Proc. of AAAI*, volume 10, pages 236–241, 2010.