Dynamic L1-norm Tucker Tensor Decomposition

Dimitris G. Chachlakis,[‡] Mayur Dhanaraj,[‡] Ashley Prater-Bennette,[†] and Panos P. Markopoulos^{‡,*}

[‡]Department of Electrical and Microelectronic Engineering

Rochester Institute of Technology

Rochester, NY 14623, USA

E-mail: dimitris@mail.rit.edu, mxd6023@rit.edu, and panos@rit.edu

[†]Air Force Research Laboratory, Information Directorate

Rome, NY 13441, USA

E-mail: ashley.prater-bennette@us.af.mil

Abstract

Tucker decomposition is a standard method for processing multi-way (tensor) measurements and finds many applications in machine learning and data mining, among other fields. When tensor measurements arrive in a streaming fashion or are too many to jointly decompose, incremental Tucker analysis is preferred. In addition, dynamic basis adaptation is desired when the nominal data subspaces change. At the same time, it has been documented that outliers in the data can significantly compromise the performance of existing methods for dynamic Tucker analysis. In this work, we present Dynamic L1-Tucker: an algorithm for dynamic and outlier-resistant Tucker analysis of tensor data. Our experimental studies on both real and synthetic datasets corroborate that the proposed method (i) attains high basis estimation performance, (ii) identifies/rejects outliers, and (iii) adapts to nominal subspace changes.

Index Terms – Data analysis, L1-norm, outliers, robust, tensors, Tucker decomposition.

I. INTRODUCTION

Data collections across diverse sensing modalities are naturally stored and processed in the form of N -way arrays, also known as *tensors* [1]. Tucker decomposition is a standard method for tensor analysis, with important applications in machine learning [2, 3], pattern recognition [4, 5], communications [6–8], and computer vision [9–11], among other fields. Common uses of Tucker include compression, denoising, and model identification. Canonical Polyadic

This research is supported in part by the U.S. National Science Foundation, under grant OAC-1808582, and by the U.S. Air Force Office of Scientific Research, under grant 18RICOR029 and the Young Investigator Program.

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

*Corresponding author.

Decomposition (CPD) [12, 13], also known as Parallel Factor Analysis (PARAFAC), is another successful tensor analysis scheme with many applications in data mining and machine learning.

Tucker can be viewed as a high-order extension of Principal-Component Analysis (PCA) [14]. Similar to PCA, which jointly analyzes a collection of vectors, Tucker analyzes a collection of $(N \geq 1)$ -way tensors to extract one orthonormal basis for each tensor mode. Instead of applying PCA on vectorized measurements, Tucker treats multi-way measurements in their tensor form, thus leveraging inherent data structure and allowing for superior inference. Standard solvers for Tucker are the Higher-Order Singular Value Decomposition (HOSVD) and the Higher-Order Orthogonal Iterations (HOOI) [15, 16].

The merits of Tucker analysis have been demonstrated in a wide range of applications. However, it is also well-documented that Tucker is very sensitive against faulty measurements (outliers). Such outliers appear often in modern datasets, due to sensor malfunctions, errors in data storage/transfer, and even deliberate dataset contamination in adversarial environments [17–19]. The outlier-sensitivity of Tucker is attributed to its L2-norm (Frobenius) formulation, which places quadratic emphasis to peripheral tensor entries. To remedy the impact of outliers, researchers have proposed robust reformulations of Tucker. For instance, Higher-Order Robust PCA (HoRPCA) [20] models and decomposes the processed tensor as the sum of a low multi-linear rank tensor (nominal data) and a sparse tensor (outliers). Another straightforward robust reformulation is L1-Tucker [21, 22], which derives by simple substitution of the L2-norm in the Tucker formulation by the more robust L1-norm (not to be confused with sparsity-inducing L1-norm regularization schemes). Algorithms for the (approximate) solution of L1-Tucker have been proposed in [21–27].

In many applications of interest, the tensor measurements arrive in a streaming way. Accordingly, the sought-after Tucker bases have to be computed incrementally. Incremental solvers are also preferred, from a computational standpoint, when there are too many collected measurements to efficiently process them as a batch. For such cases, researchers have proposed an array of algorithms for incremental Tucker decomposition, including Dynamic Tensor Analysis (DTA), Streaming Tensor Analysis (STA), Window-based Tensor Analysis (WTA) [28, 29], and Accelerated Online Low-Rank Tensor Learning (ALTO) [30], to name a few. Despite their computational merits, similar to batch Tucker analysis, most of the existing incremental methods are sensitive against outliers.

In this work, we present Dynamic L1-Tucker: a scalable method for incremental L1-Tucker analysis, with the ability to (i) provide quality estimates of the Tucker bases, (ii) detect and reject outliers, and (iii) adapt to nominal subspace changes.

The rest of this paper is organized as follows. In Section II, we introduce notation and provide an overview of the relevant technical background (tensors, Tucker decomposition, L1-Tucker, and existing methods for dynamic/incremental Tucker). In Section III, we formally state the problem of interest. In Section IV, we present the proposed Dynamic L1-Tucker (D-L1-Tucker) method. Section V holds extensive experimental studies on synthetic and real datasets. Concluding remarks are drawn in Section VI.

II. TECHNICAL BACKGROUND

A. Notation and Tensor Preliminaries

In this manuscript, vectors and matrices are denoted by lower- and upper-case bold letters, respectively –e.g., $\mathbf{x} \in \mathbb{R}^{D_1}$ and $\mathbf{X} \in \mathbb{R}^{D_1 \times D_2}$. N -way tensors are denoted by upper-case calligraphic bold letters –e.g., $\mathcal{X} \in \mathbb{R}^{D_1 \times \dots \times D_N}$. Collections/sets of tensors are denoted by upper-case calligraphic letters –e.g., $\mathcal{X} = \{\mathcal{X}, \mathcal{Y}\}$. The squared Frobenius/L2 norm, $\|\cdot\|_F^2$, returns the sum of squared entries of its tensor argument while the L1-norm, $\|\cdot\|_1$, returns the sum of the absolute entries of its tensor argument. $\mathbb{S}_{D \times d} = \{\mathbf{Q} \in \mathbb{R}^{D \times d} : \mathbf{Q}^\top \mathbf{Q} = \mathbf{I}_d\}$ is the Stiefel manifold of rank- d orthonormal matrices in $\mathbb{R}^D$. Each entry of $\mathcal{X}$ is identified by N indices $\{i_n\}_{n=1}^N$ such that $i_n \leq D_n$ for every $n \in [N] = \{1, 2, \dots, N\}$. For every $n \in [N]$, $\mathcal{X}$ can be seen as a collection of $P_n = \prod_{m \in [N] \setminus n} D_m$ length- D_n vectors known as mode- n fibers of $\mathcal{X}$. For instance, given a fixed set of indices $i_{m \in [N] \setminus n}$, $\mathcal{X}(i_1, \dots, i_{n-1}, :, i_{n+1}, \dots, i_N)$ is a mode- n fiber of $\mathcal{X}$. A matrix with columns of all the mode- n fibers of $\mathcal{X}$ is called the mode- n unfolding (or, flattening) of $\mathcal{X}$ and will henceforth be denoted as $\text{mat}(\mathcal{X}, n) \in \mathbb{R}^{D_n \times P_n}$. $\mathcal{X} \times_n \mathbf{A}$ is the mode- n product of tensor $\mathcal{X}$ with matrix $\mathbf{A}$ of conformable size, and $\mathcal{X} \times_{n \in [N]} \mathbf{Q}_n^\top$ compactly denotes the multi-way product $\mathcal{X} \times_1 \mathbf{Q}_1^\top \times_2 \mathbf{Q}_2^\top \dots \times_n \mathbf{Q}_N^\top$. In accordance with the common convention, the order in which the mode- n fibers of $\mathcal{X}$ appear in $\text{mat}(\mathcal{X}, n)$ is as specified in [15].

B. Tucker Decomposition

Consider coherent tensor measurements $\mathcal{X}_t \in \mathbb{R}^{D_1 \times \dots \times D_N}$, $t = 1, 2, \dots, T$. Also, define their concatenation tensor $\mathcal{X} \in \mathbb{R}^{D_1 \times \dots \times D_N \times T}$, such that $\mathcal{X}(:, :, \dots, :, t) = \mathcal{X}_t$. Tucker analysis of the measurement batch $\{\mathcal{X}_t\}_{t=1}^T$ is formulated as

$$\max_{\{\mathbf{Q}_n \in \mathbb{S}_{D_n \times d_n}\}_{n \in [N]}} \left\| \mathcal{X} \times_{n \in [N]} \mathbf{Q}_n^\top \right\|_2^F, \quad (1)$$

where $\|\mathcal{X} \times_{n \in [N]} \mathbf{Q}_n^\top\|_F^2 = \sum_{t \in [T]} \|\mathcal{X}_t \times_{n \in [N]} \mathbf{Q}_n^\top\|_F^2$. For $d_n \leq D_n \forall n \in [N]$, (1) seeks N low-rank bases to compress the tensor measurements so that the aggregate preserved variance is maximized. Tucker is commonly implemented by means of the HOSVD or HOOI algorithms. HOSVD is a single-shot method that approximates the N bases in (1) disjointly by N parallel PCAs of the form

$$\max_{\mathbf{Q} \in \mathbb{S}_{D_n \times d_n}} \|\text{mat}(\mathcal{X}, n)^\top \mathbf{Q}\|_F^2. \quad (2)$$

On the other hand, HOOI is an iterative method that optimizes the N bases jointly. In general, initialized at bases $\{\mathbf{Q}_{n,0}\}_{n \in [N]}$, at iteration $i > 0$ and for $n = 1, 2, \dots, N$, HOOI returns $\mathbf{Q}_{n,i}$ as the solution to the PCA problem

$$\max_{\mathbf{Q} \in \mathbb{S}_{D_n \times d_n}} \|\mathbf{A}_{n,i}^\top \mathbf{Q}\|_F^2, \quad (3)$$

where

$$\mathbf{A}_{n,i} = \text{mat}(\mathcal{X} \times_{m \in [n-1]} \mathbf{Q}_{m,i}^\top \times_{k \in [N-n]+n} \mathbf{Q}_{k,i-1}^\top, n). \quad (4)$$

C. Outliers and L1-Tucker

Outliers appear often in datasets and can significantly compromise the performance of Tucker methods. Motivated by the success of L1-PCA in vector-data analysis [31], L1-Tucker decomposition has been proposed as an outlier-resistant Tucker reformulation. L1-Tucker derives by substituting the outlier-responsive L2 norm in (1) by the more robust L1-norm, as

$$\max_{\{\mathbf{Q}_n \in \mathbb{S}_{D_n \times d_n}\}_{n \in [N]}} \left\| \mathcal{X} \times_{n \in [N]} \mathbf{Q}_n^\top \right\|_1, \quad (5)$$

where $\|\mathcal{X} \times_{n \in [N]} \mathbf{Q}_n^\top\|_1 = \sum_{t \in [T]} \|\mathbf{x}_t \times_{n \in [N]} \mathbf{Q}_n^\top\|_1$. L1-HOSVD [22, 32, 33] approximates the solution to L1-Tucker by N parallel L1-PCA problems. That is, for every $n \in [N]$, it finds $\mathbf{Q}_n$ by solving (approximately or exactly) the L1-PCA

$$\max_{\mathbf{Q} \in \mathbb{S}_{D_n \times d_n}} \|\text{mat}(\mathcal{X}, n)^\top \mathbf{Q}\|_1. \quad (6)$$

On the other hand, L1-HOOI is an iterative process that provably attains a higher L1-Tucker metric when initialized at the solution of L1-HOSVD [22, 34]. Initialized at $\{\mathbf{Q}_{n,0}\}_{n \in [N]}$ (typically by means of L1-HOSVD), at every iteration $i \geq 1$, L1-HOOI updates $\mathbf{Q}_{n,i}$ by solving

$$\max_{\mathbf{Q} \in \mathbb{S}_{D_n \times d_n}} \|\mathbf{A}_{n,i}^\top \mathbf{Q}\|_1, \quad (7)$$

where $\mathbf{A}_{n,i}$ is defined in (4).

As seen above, L1-HOSVD and L1-HOOI are implemented through a series of L1-PCAs. L1-PCA admits an exact solution by combinatorial optimization with high cost [31]. However, there are multiple high-performing approximate L1-PCA solvers in the literature that can be used by L1-Tucker methods. In the algorithmic developments of this work, we consider the L1-norm Bit-Flipping (L1-BF) algorithm of [35]. For the sake of completeness, a brief description of L1-BF follows.

Consider matrix $\mathbf{X} \in \mathbb{R}^{Z \times Q}$, for $Q \geq Z$, and the L1-PCA

$$\max_{\mathbf{Q} \in \mathbb{S}_{Z \times z}} \left\| \mathbf{X}^\top \mathbf{Q} \right\|_1. \quad (8)$$

L1-BF is based on the following Theorem, presented in [31].

Theorem 1. [31] *Let $\mathbf{B}_{opt} \in \{\pm 1\}^{Q \times z}$ be a solution to $\max_{\mathbf{B} \in \{\pm 1\}^{Q \times z}} \|\mathbf{X}\mathbf{B}\|_*$. Then, $\text{Proc}(\mathbf{X}\mathbf{B}_{opt})$ is an exact solution to L1-PCA in (8).*

The nuclear norm $\|\cdot\|_*$ returns the sum of the singular values of its argument and, for any tall matrix $\mathbf{A} \in \mathbb{R}^{Z \times z}$ that admits SVD $\mathbf{A} = \mathbf{U}\Sigma_{z \times z}\mathbf{V}^\top$, $\text{Proc}(\mathbf{A}) = \mathbf{U}\mathbf{V}^\top$.

In view of Theorem 1, [35] proposed to initialize at arbitrary $\mathbf{B}_0 \in \{\pm 1\}^{Q \times z}$ and iteratively conduct optimal single-bit flips (negations). Let $\mathbf{e}_{q,Q}$ denote the q -th column of the size- Q identity matrix $\mathbf{I}_Q$. Then, at iteration $i \geq 1$, L1-BF solves

$$(k', l') = \underset{(k,l) \in [Q] \times [z]}{\text{argmax}} \left\| \mathbf{X} \left(\mathbf{B}_{i-1} - 2\mathbf{e}_{k,Q}\mathbf{e}_{l,z}^\top [\mathbf{B}_{i-1}]_{k,l} \right) \right\|_* \quad (9)$$

and updates $\mathbf{B}_i = \mathbf{B}_{i-1} - 2\mathbf{e}_{k',Q}\mathbf{e}_{l',z}[\mathbf{B}_{i-1}]_{k',l'}$. Among all possible single bit-flips, negation of the (k', l') -th entry of $\mathbf{B}_{i-1}$ offers the maximum possible value in $\|\mathbf{X}\mathbf{B}_i\|_*$. Importantly, L1-BF is guaranteed to monotonically increase the metric and converge in finite (in practice, few) iterations.

D. Existing Methods for Incremental and Dynamic Tucker

While HOSVD and HOOI are standard methods for batch processing, incremental alternatives are needed when the tensor measurements arrive in a streaming fashion or they are too many to efficiently process in a batch.

Dynamic Tensor Analysis (DTA) [28, 29] efficiently approximates the HOSVD solution by processing measurements incrementally, with a fixed computational cost per update. Moreover, DTA can track multi-linear subspace changes, weighing past measurements with a forgetting factor. Streaming Tensor Analysis (STA) [28, 29] is a fast alternative to DTA, particularly designed for time-critical applications. Window-based Tensor Analysis (WTA) is another DTA variant which, in contrast to DTA and STA, adapts to changes by only considering a sliding window of measurements. The Accelerated Online Low-Rank Tensor Learning (ALTO) method was presented in [30]. For each new measurement, ALTO updates the bases through a tensor regression model. In [36], authors presented another method for Low-Rank Updates to Tucker (LRUT). When a new measurement arrives, LRUT projects it on the current bases and few more randomly chosen orthogonal directions, forming an augmented core tensor. Then it updates the bases by standard Tucker (e.g., HOSVD) on this extended core. In [37], authors consider very large tensors and propose randomized algorithms for Tucker decomposition based on the TENSORSKETCH [38]. It is stated that these algorithms can also extend for processing streaming data. Randomized methods for Tucker of streaming tensor data were also proposed in [39]. These methods rely on dimension-reduction maps for sketching the Tucker decomposition and they are accompanied by probabilistic performance guarantees. More methods for incremental tensor processing were presented in [40–43], focusing on specific applications, such as foreground segmentation, visual tracking, and video foreground/background separation.

Methods for incremental CPD/PARAFAC tensor analysis were presented in [44, 45]. Robust incremental solvers for PARAFAC were also presented in [46, 47]. However, the problem of outlier-resistant dynamic/incremental Tucker analysis remains to date largely unexplored.

III. PROBLEM STATEMENT

Focusing on outlier-resistant tensor processing, we wish to estimate the L1-Tucker bases of a tensor-data model, as formulated in (5). We assume, however, that the measurements $\{\mathbf{x}_t\}_{t=1}^T$ are originally unavailable and collected in a streaming fashion, one at a time.

To set our algorithmic guidelines, we start by considering two simplistic antipodal approaches. On the one hand, an *instantaneous* approach would L1-Tucker-decompose each new measurement to return new bases, independently of any previously seen data. While this approach is memory-less and computationally simple, its basis estimation performance is bound to be limited, especially in low Signal-to-Noise Ratio (SNR). On the other hand, an

Fig. 1. Schematic illustration of the proposed algorithm for streaming L1-norm Tucker decomposition.

increasing-batch approach would append the new measurement to the already collected ones and re-solve the L1-Tucker problem from scratch. As the data collection increases, this method could attain superior basis estimation performance at the expense of increasingly high computational and storage overhead.

Both these extreme approaches exhibit an unfavorable performance/cost trade-off. In contrast, a preferred method would leverage each new measurement, together with previous ones, to efficiently update the existing bases. The development of such a method is the main contribution of this paper, as presented in detail in the following section.

IV. PROPOSED ALGORITHM

The proposed *Dynamic L1-Tucker Decomposition (D-L1-Tucker)* is a method for incremental estimation of the L1-Tucker bases. D-L1-Tucker is designed to (i) attain high basis estimation performance, (ii) suppress outliers, and (iii) adapt to nominal subspace changes. In this section, we present D-L1-Tucker in detail, addressing basis initialization, basis updates, parameter tuning, and modifications for long-term efficiency.

A. Batch Initialization

Considering the availability of an initial batch of $B \ll T$ measurements, $\mathcal{B} = \{\mathbf{x}_1, \dots, \mathbf{x}_B\}$, we run on it L1-HOSVD or L1-HOOI to obtain an initial set of L1-Tucker estimates $\mathcal{Q}_0 = \{\mathbf{Q}_1^{(0)}, \dots, \mathbf{Q}_N^{(0)}\}$. Apart from $\mathcal{Q}_0$, we also initialize a *memory set* $\mathcal{M}_0 = \Omega(\mathcal{B}, M)$, for some maximum memory size $M \geq 0$. For any ordered set $\mathcal{I}$ and integer $Z \geq 0$, we define

$$\Omega(\mathcal{I}, Z) = \begin{cases} \mathcal{I}, & \text{if } |\mathcal{I}| \leq Z, \\ [\mathcal{I}]_{|\mathcal{I}| - Z + 1: |\mathcal{I}|}, & \text{if } |\mathcal{I}| > Z. \end{cases} \quad (10)$$

That is, $\Omega(\mathcal{B}, M)$ returns the last $\min\{M, B\}$ elements in $\mathcal{B}$.

If an initialization batch $\mathcal{B}$ is not available, the bases in $\mathcal{Q}_0$ are chosen arbitrarily and the initial memory $\mathcal{M}_0$ is empty. In this case, D-L1-Tucker becomes purely streaming.

Algorithm 1. Proposed Dynamic L1-Tucker Decomposition**Input:** $\{\bar{\mathcal{X}}_t\}_{t \in [T]}$, B , M , W , τ , $\mathcal{Q} \leftarrow \cup_{n \in [N]} \mathbf{Q}_n$

- 1: $\mathcal{B} \leftarrow \bigcup_{t \in [B]} \bar{\mathcal{X}}_t$
- 2: $(\mathcal{Q}, \mathcal{M}, \mathcal{W}) \leftarrow \text{batch-init}(\mathcal{B}, \mathcal{Q}, M)$
- 3: For $t > B$
- 4: $(\mathcal{Q}, \mathcal{M}, \mathcal{W}) \leftarrow \text{online-updates}(\bar{\mathcal{X}}_t, \mathcal{M}, \mathcal{Q}, \mathcal{W}, \tau)$

Return: $\mathcal{Q} \rightarrow \{\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_N\}$ **Function:** $\text{batch-init}(\mathcal{B}, \mathcal{Q}, M)$

- 1: $\mathcal{Q} \leftarrow \text{L1-Tucker}(\mathcal{B}, \mathcal{Q})$
- 2: $\mathcal{M} \leftarrow \Omega(\mathcal{B}, M)$
- 3: $\mathcal{W} \leftarrow \emptyset$

Return: $\mathcal{Q}, \mathcal{M}, \mathcal{W}$ **Function:** $\text{online-updates}(\bar{\mathcal{X}}_t, \mathcal{M}, \mathcal{Q}, \mathcal{W}, \tau)$

- 1: $r_t \leftarrow \|\bar{\mathcal{X}}_t \times_{n \in [N]} \mathbf{Q}_n^\top\|_F^2 \|\bar{\mathcal{X}}_t\|_F^{-2}$
- 2: If $r_t > \tau$
- 3: $\mathcal{Y} \leftarrow \Phi(\mathcal{M}, \bar{\mathcal{X}}_t)$
- 4: $\mathcal{Q} \leftarrow \text{L1-Tucker}(\mathcal{Y}, \mathcal{Q})$
- 5: $\mathcal{M} \leftarrow \Omega(\mathcal{Y}, M)$
- 6: $\mathcal{W} \leftarrow \emptyset$
- 7: Else
- 8: $\mathcal{W} \leftarrow \mathcal{W} \cup \bar{\mathcal{X}}_t$
- 9: If $|\mathcal{W}| = W$
- 10: $(\mathcal{Q}, \mathcal{M}, \mathcal{W}) \leftarrow \text{batch-init}(\mathcal{W}, \mathcal{Q}, M)$

Return: $\mathcal{Q}, \mathcal{M}, \mathcal{W}$

Fig. 2. The proposed Dynamic L1-norm Tucker Decomposition algorithm.

B. Streaming Updates

When a new measurement $\bar{\mathcal{X}}_t$, $t \geq 1$ is collected¹ we perform a reliability check on it to assess its reliability based on the most recently updated set of bases $\mathcal{Q}_{t-1}$. Motivated by [27, 48], we define the reliability as

$$r_t = \left\| \bar{\mathcal{X}}_t \times_{n \in [N]} \mathbf{Q}_n^{(t-1)\top} \right\|_F^2 \|\bar{\mathcal{X}}_t\|_F^{-2} \in [0, 1]. \quad (11)$$

By definition, the value of r_t will be between 0 and 1. If $r_t = 1$, then the bases in $\mathcal{Q}_{t-1}$ perfectly describe $\bar{\mathcal{X}}_t$. In contrast, if $r_t = 0$, then the set $\mathcal{Q}_{t-1}$ does not capture any component of $\bar{\mathcal{X}}_t$. Then, we introduce a user-defined parameter τ and consider that $\bar{\mathcal{X}}_t$ is reliable for processing if $r_t \geq \tau$. Otherwise, $\bar{\mathcal{X}}_t$ is considered to be an outlier and it is rejected.

If $\bar{\mathcal{X}}_t$ passes the reliability check, we use it to update the bases and memory as follows. First, we append the new measurement to the most recent memory set $\mathcal{M}_{t-1}$ by computing the extended memory $\mathcal{Y} = \Phi(\mathcal{M}_{t-1}, \bar{\mathcal{X}}_t) = \mathcal{M}_{t-1} \cup \bar{\mathcal{X}}_t$. Then, we update the basis set to $\mathcal{Q}_t$ by running L1-HOOI on $\mathcal{Y}$, initialized to the bases in $\mathcal{Q}_{t-1}$.

¹A bar over a tensor denotes that it is streaming.

Finally, we update the memory by discarding the oldest measurement, as

$$\mathcal{M}_t = \Omega(\mathcal{M}', M). \quad (12)$$

In view of the above, the cost of the L1-HOOI algorithm remains low across updates because, at any given instance, the extended memory $\mathcal{Y}$ will always comprise at most $M + 1$ measurements.

If $\bar{\mathcal{X}}_t$ fails the reliability check, we discard it and update the bases and memory by setting $\mathcal{Q}_t = \mathcal{Q}_{t-1}$ and $\mathcal{M}_t = \mathcal{M}_{t-1}$, respectively. A schematic representation of the proposed algorithm is offered in Fig. 1.

C. Zero Centering

In specific applications (e.g., in image processing) we are interested to estimate subspaces of zero-centered data. To this end, we can modify the proposed algorithm so that, at every update instance $(t - 1)$, it computes and maintains the mean $\mathcal{C}_{t-1} = (1/M) \sum_{m=1}^M [\mathcal{M}_{t-1}]_m$, where $[\mathcal{M}_{t-1}]_m$ returns the m -th tensor measurement in $\mathcal{M}_{t-1}$. Then, when $\bar{\mathcal{X}}_t$ is collected, it will be first zero-centered as $\bar{\mathcal{X}}_t^c = \bar{\mathcal{X}}_t - \mathcal{C}_{t-1}$ and then, if it passes the reliability check, $\bar{\mathcal{X}}_t^c$ will be used to update the bases, as described above.

D. Adaptation to Subspace Changes

In many applications of interest, the underlying data subspaces change across time. In such cases, an ambiguity naturally rises on whether a rejected measurement was actually an outlier or the nominal data subspaces have changed and need to be tracked. To resolve this ambiguity and allow D-L1-Tucker to adapt, we work as follows.

First, we make the mild assumption that outlying measurements appear sporadically. Then, we introduce a buffer of ambiguous measurements, $\mathcal{W}$, with capacity $W > 0$. When a streaming measurement fails the reliability check, we insert it to $\mathcal{W}$. If a measurement passes the reliability check, then we empty $\mathcal{W}$. If at any update instance $|\mathcal{W}|$ reaches W (i.e., W consecutive streaming measurements were rejected as outliers), then we detect a nominal subspace change. In order to adapt to this change, we empty the memory, we set $\mathcal{B} = \mathcal{W}$, and re-initialize (reset) the bases and memory, as described in Section IV-A. Next, the updates proceed as described in Sections IV-B and IV-D. A pseudocode of the proposed D-L1-Tucker algorithm is presented in Fig. 2.

E. Long-Run Efficiency

As measurements are streaming, D-L1-Tucker keeps refining the basis estimates. Naturally, after a sufficiently large number of measurements have been processed, the enhancement rate of the bases estimates can be so low that does not justify the computational effort expended for the update.

In view of this observation, we can enhance the long-run efficiency of D-L1-Tucker by introducing an exponentially decreasing probability ρ_t to determine whether or not the t -th measurement will be processed. Intuitively, when a large number of reliable measurements have been processed, ρ_t should be low enough to limit the number of updates performed. For example, let us denote by α_{t-1} the number of consecutive measurements that have passed

the reliability check at update instance $t - 1$. Then, if $\bar{\mathcal{X}}_t$ passes the reliability check, it will be processed with probability $\rho_t = \rho^{\alpha_{t-1}+1}$, for some initial probability $\rho > 0$, close to 1. If $\bar{\mathcal{X}}_t$ fails the reliability check, then it is rejected and α_t is reset to 0.

F. Parameter Configuration

The performance of D-L1-Tucker largely depends on three parameters: the initialization batch size B , the memory size M , and the reliability threshold τ . Here, we discuss how to select these parameters.

Batch size B : B determines the quality of the initial set of bases. That is, higher values of B will generally offer better set of bases. Naturally, a very large B would contradict the streaming nature of the method.

Memory size M : M determines how many measurements L1-Tucker will process at each time instance. Similar to B , higher values of M can enable superior estimation performance. At the same time, high values of M increase the overhead of storage and computation (cost of L1-Tucker updates). Thus, a rule of thumb is to set M as high as the storage/computation limitations of the application permit.

Reliability threshold τ : For $\tau = 0$, all measurements will be processed (including outliers); for $\tau = 1$, all measurements will fail the reliability check and no basis updates will take place. Appropriate tuning of τ between 0 and 1 may ask for some prior knowledge on the SNR quality of the nominal data. Alternatively, in the sequel we present a data driven method for setting τ .

We start with the reasonable assumption that the initialization batch $\mathcal{B}$ is outlier-free. Then, we conduct on $\mathcal{B}$ a leave-one-out cross-validation to tune τ . For every $i \in [B]$, we first form $\mathcal{B}_i = \mathcal{B} \setminus \mathcal{X}_i$. Then, we obtain the basis set $\mathcal{Q}_i$ by running L1-HOOI on $\mathcal{B}_i$. Next, we capture in r_i the reliability of $\mathcal{X}_i$ evaluated on $\mathcal{Q}_i$ (notice that $\mathcal{X}_i$ did not participate in the computation of $\mathcal{Q}_i$). Finally, we set τ to the minimum, median, or maximum value of the cross-validated reliabilities $\{r_1, \dots, r_B\}$, depending on the noise-tolerance/outlier-robustness level that we want to enforce.

V. EXPERIMENTAL STUDIES

A. Testing Parameter Configurations

We first study the performance of the proposed D-L1-Tucker algorithm across varying parameter configurations. We consider T ($N = 3$)-way measurements $\bar{\mathcal{X}}_1, \dots, \bar{\mathcal{X}}_T$, where

$$\bar{\mathcal{X}}_t = \mathcal{G}_t \times_{n \in [N]} \mathbf{Q}_n^{\text{nom}} + \mathcal{N}_t + \mathcal{O}_t \in \mathbb{R}^{D_1 \times D_2 \times \dots \times D_N}, \quad (13)$$

for a nominal set of bases $\mathcal{Q}^{\text{nom}} = \{\mathbf{Q}_n^{\text{nom}} \in \mathbb{S}_{D_n \times d_n}\}_{n \in [N]}$. The core tensor $\mathcal{G}_t \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_N}$ draws entries independently from $\mathcal{N}(0, \sigma_s^2)$. $\mathcal{N}_t$ models Additive White Gaussian Noise (AWGN) and draws entries from $\mathcal{N}(0, \sigma_n^2)$. $\mathcal{O}_t$ models sporadic heavy outlier corruption and is non-zero with probability p_o . When non-zero, $\mathcal{O}_t$ draws entries from $\mathcal{N}(0, \sigma_o^2)$. In order to measure data quality, we define the SNR as

$$\text{SNR} = \frac{\mathbb{E}\{\|\bar{\mathcal{X}}_t\|_F^2\}}{\mathbb{E}\{\|\mathcal{N}_t\|_F^2\}} = \frac{\sigma_s^2}{\sigma_n^2} \left(\frac{d}{D}\right)^N \quad (14)$$

Fig. 3. $N = 3$, $D = 10$, $d = 5$, $B = 5$, $T = 30$, SNR= 0dB, ONR= 14dB, 3000 realizations.

and the Outlier-to-Noise Ratio (ONR)

$$\text{ONR} = \frac{\mathbb{E}\{\|\mathcal{O}_t\|_F^2\}}{\mathbb{E}\{\|\mathcal{N}_t\|_F^2\}} = \frac{\sigma_o^2}{\sigma_n^2}. \quad (15)$$

Our objective is to recover $\mathbf{Q}^{\text{nom}}$ by processing measurements $\tilde{\mathcal{X}}_{t \in [T]}$ in a streaming way. Denoting by $\hat{\mathbf{Q}}_n$ the estimate of $\mathbf{Q}_n^{\text{nom}}$, we quantify performance by means of the Mean Aggregate Normalized Subspace Squared Error (MANSSE)

$$\text{MANSSE} = \frac{1}{2Nd} \sum_{n \in [N]} \left\| \mathbf{Q}_n^{\text{nom}} \mathbf{Q}_n^{\text{nom}^\top} - \hat{\mathbf{Q}}_n \hat{\mathbf{Q}}_n^\top \right\|_F^2. \quad (16)$$

First, we set $N = 3$, $D_n = 10 \forall n$, $d_n = 5 \forall n$, $B = 5$, and $T = 30$. Moreover, we set σ_s^2 , σ_n^2 , σ_o^2 , such that SNR= 0dB and ONR= 14dB. In Fig. 3, we plot the MANSSE metric versus varying $M \in \{5, 10, 15, 20\}$ and fixed $(p_o, \tau) \in \{(0.1, 0), (0.06, 0.4), (0.1, 0.6), (0.06, 0.7)\}$. We observe that the curves corresponding to $\tau \geq 0.6$ are almost horizontal. This implies that these values of τ are too strict, rejecting almost all measurements. For $\tau = 0$, all measurements are processed (outliers and nominal ones); therefore, we see that the estimation performance improves as M increases, however, the estimation error is somewhat high because of the processed outliers. The curve corresponding to $\tau = 0.4$ exhibits the best performance across the board.

Next, motivated by the above observations, we fix SNR= 0dB and let τ vary in $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$. In Fig. 4, we plot MANSSE versus τ for different values of outlier probability p_o . We notice that for any $\tau \in [0.3, 0.5]$, D-L1-Tucker exhibits high, almost identical MANSSE performance independently of p_o . This, in turn, suggests that the SNR plays an important role in determining the optimal value of τ , for which nominal measurements will be processed and outliers will be rejected with high probability. For the same study, we present the frequency of rejection versus τ in Fig. 5. Again, we notice that for very low values of τ most measurements are accepted for processing. In contrast, for very high values of τ , most measurements are rejected. Interestingly,

Fig. 4. $N = 3$, $D = 10$, $d = 5$, $B = 5$, $M = 10$, $T = 30$, SNR= 0dB, ONR= 14dB, 3000 realizations.

Fig. 5. $N = 3$, $D = 10$, $d = 5$, $B = 5$, $M = 10$, $T = 30$, SNR= 0dB, ONR= 14dB, 3000 realizations.

this figure suggests that for any given parameter configuration there will be an optimal value of τ for which the frequency of rejection will approach the probability of outliers p_o –which, in turn, implies that in general outliers will be rejected and nominal data will be processed.

Finally, we let p_o vary in $\{0, 0.02, 0.06, 0.08, 0.1\}$ and, in Fig. 6, we plot the frequency of rejection versus p_o . In accordance with previous observations, we see that high values of τ result in high rejection frequency, independently of the value of p_o . Interestingly, we see that values of τ within $[0.3, 0.4]$ appear to be near-optimal for this particular SNR and parameter configuration, as their performance almost coincides with the 45° slope, at all points of which the frequency of rejection equals the outlier probability p_o .

Fig. 6. $N = 3$, $D = 10$, $d = 5$, $B = 5$, $M = 10$, $T = 30$, SNR= 0dB, ONR= 14dB, 3000 realizations.

B. Dynamic Subspace Adaptation

We consider a total of $T = T_1 + T_2$ streaming measurements, in the form of (13). The first T_1 measurements are generated by nominal bases $\mathbf{Q}_n^{\text{nom},1}$. For $t > T_1$ and on, the measurements are generated by bases $\mathbf{Q}_n^{\text{nom},2}$. The angular proximity of $\mathbf{Q}_n^{\text{nom},1}$ to $\mathbf{Q}_n^{\text{nom},2}$, defined as $1 - \|\mathbf{Q}_n^{\text{nom},1} \mathbf{Q}_n^{\text{nom},1\top} - \mathbf{Q}_n^{\text{nom},2} \mathbf{Q}_n^{\text{nom},2\top}\|_F^2 (2d)^{-1}$, is set between 30% and 40 % for every $n \in [N]$. Moreover, we consider that the outlier is only active at instance $t = t_o = 45$. We set $N = 3$, $D_n = 10$, $d_n = 3$, $T_1 = 70$, and $T_2 = 30$. The SNR and ONR are set to -6dB and 18dB , respectively. We process all measurements by the proposed D-L1-Tucker algorithm for $B = 2$, $M = 12$, $W = 4$, and data driven τ (median of cross-validated batch reliabilities). We also process the streaming measurements with DTA ($\lambda = 0.2, 0.8$), LRUT (additional core dimensions $k = D - d - 2$), and instantaneous HOSVD² counterparts.

In Fig. 7, we plot the MANSSE versus update index t . All methods, except for the instantaneous HOSVD, start from a higher MANSSE value and refine their bases by processing streaming measurements until they reach a low plateau. At $t = 45$, when the outlier appears, we observe that all competing methods suffer a significant performance loss. In contrast, the proposed D-L1-Tucker algorithm discards the outlier and its performance remains unaffected. When subsequent measurements are streaming, the competing methods start recovering until they reach again a low plateau, which is largely determined by the SNR and the parameter configuration of each method. Interestingly, the instantaneous HOSVD recovers rapidly, after just one measurement, because it is memory-less. DTA ($\lambda = 0.2$) recovers faster than DTA ($\lambda = 0.8$) but its MANSSE plateau is higher. LRUT also recovers and reaches its plateau performance after it has seen about 10 measurements after the outlier. At time instance 71 the nominal data subspaces shift, affecting all methods expect for the memoryless/instantaneous HOSVD. D-L1-Tucker attains a high value of MANSSE for about W time instances while its ambiguity window is being filled. Right

²At update instance t , instantaneous HOSVD returns the HOSVD solution of $\tilde{\mathcal{X}}_t$, independently of any previous measurements.

Fig. 7. $N = 3$, $D = 10$, $d = 3$, $T_1 = 70$, $T_2 = 30$, $B = 2$, $M = 12$, $W = 4$, $\text{SNR} = -6\text{dB}$, $\text{ONR} = 18\text{dB}$, data driven τ , 3000 realizations.

after, it rapidly recovers to a low MANSSE value and keeps refining as more measurements are streaming. DTA and LRUT are also adapting to the new underlying structure after processing a few measurements. Another interesting observation is that the low plateau level for each method appears to be the same in the two distinct coherence windows.

In Fig. 8, we plot the reliability of the streaming measurements across updates in accordance with (11). At the same figure, we illustrate the frequency of rejection; that is, the frequency by which measurements fail the reliability check. We notice that the outlier at $t = 45$ and the W measurements following the subspace change are rejected with probability close to 1. In addition, we observe the instantaneous reliability drop when the outlier appears and when nominal subspaces change. For this value of $\text{SNR} = -6\text{dB}$, the reliability level for nominal measurements is about 0.2 and our data driven τ is accordingly low.

We conclude this study by comparing the run time of each method across updates.³ In Fig. 9 and Fig. 10, we plot the instantaneous and cumulative run times, respectively. We observe that the instantaneous HOSVD and DTA exhibit constant run time across updates independently of outliers or subspace changes. D-L1-Tucker also exhibits about constant runtime after its memory has been filled. Moreover, we notice an instantaneous drop in the runtime at index $t = 45$ which is because D-L1-Tucker discarded the outlier and did not process it. In contrast, when the outlier appears and when the subspaces change, LRUT attains an increase in runtime, as it tries to adapt.

C. Dynamic Video Foreground/Background Separation

Video foreground/background separation is a common task for object detection, security surveillance, and traffic monitoring. The omnipresent background in a static camera scene determines a nominal subspace, while any

³Reported computation times are measured in MATLAB R2019a, running on an Intel(R) core(TM) i7-8700 processor at 3.2 GHz and 32 GB RAM.

Fig. 8. $N = 3, D = 10, d = 3, T_1 = 70, T_2 = 30, B = 2, M = 12, W = 4, \text{SNR} = -6\text{dB}, \text{ONR} = 18\text{dB}$, data driven τ , 3000 realizations.

Fig. 9. $N = 3, D = 10, d = 3, T_1 = 70, T_2 = 30, B = 2, M = 12, W = 4, \text{SNR} = -6\text{dB}, \text{ONR} = 18\text{dB}$, data driven τ , 3000 realizations.

foreground movement such as vehicles or people represent intermittent outliers. In this experiment, we use D-L1-Tucker to estimate the background and foreground of incoming video frames and compare its performance with that of state-of-the-art alternatives.

For this experiment, we use videos from the benchmark CAVIAR database [49]. Specifically, we use two videos, each containing 100 frames of size $(D_1 = 173) \times (D_2 = 231)$ and capturing a different scene. Each video is viewed as collection of frames (i.e., 2-way tensor measurements). We collate the two videos, one after the other across the frame mode, and form tensor $\mathcal{X} \in \mathbb{R}^{D_1 \times D_2 \times (T=200)}$. Henceforth, $\bar{\mathcal{X}}_t$ will denote the t -th frame of $\mathcal{X}$.

We apply the proposed algorithm on $\mathcal{X}$ by setting $d_n = d = 3, B = 5, M = 10, W = 20$, and τ by the proposed batch reliability cross-validation. For every $t \in [T - B] + B$, we obtain bases $\mathbf{Q}_1^{(t)}$ and $\mathbf{Q}_2^{(t)}$ and the mean frame $\mathcal{C}_t$. Accordingly, we estimate the background as $\mathcal{X}_t^{\text{BG}} = \mathbf{Q}_1^{(t)} \mathbf{Q}_2^{(t)\top} (\bar{\mathcal{X}}_t - \mathcal{C}_t) \mathbf{Q}_2^{(t)} \mathbf{Q}_2^{(t)\top} + \mathcal{C}_t$ and the foreground

Fig. 10. $N = 3$, $D = 10$, $d = 3$, $T_1 = 70$, $T_2 = 30$, $B = 2$, $M = 12$, $W = 4$, $\text{SNR} = -6\text{dB}$, $\text{ONR} = 18\text{dB}$, data driven τ , 3000 realizations.

as $\mathbf{x}_t^{\text{FG}} = \bar{\mathbf{x}}_t - \mathbf{x}_t^{\text{BG}}$.

We compare the performance of the proposed algorithm with that of DTA, LRUT, OSTD, HOOI (increasing batch), and L1-HOOI (increasing batch). For the last two benchmark approaches, at any frame index t we run HOOI/L1-HOOI on the measurements $\{\bar{\mathbf{x}}_j\}_{j \in [t]}$, starting from arbitrary initialization. We notice that DTA is capable of tracking scene changes using a forgetting factor λ . Since the background estimation involves mean subtraction, for a fair comparison with the proposed method, we enable mean tracking for DTA by computing $\mathbf{C}_t^{\text{DTA}} = \lambda \mathbf{C}_{t-1}^{\text{DTA}} + (1 - \lambda) \bar{\mathbf{x}}_t$, for $\mathbf{C}_1^{\text{DTA}} = \bar{\mathbf{x}}_1$. For all other methods, we compute the mean incrementally at any t as $\mathbf{C}_t = ((t-1)\mathbf{C}_{t-1} + \bar{\mathbf{x}}_t) / t$. For DTA, we use two values of forgetting factor, $\lambda = 0.95, 0.7$ and for LRUT we set the number of additional core dimensions to $k_n = D_n - d - 3$.

In Figs. 11 and 12, we present the backgrounds and foregrounds obtained by the proposed method, along with other methods under comparison at 75-th frame (scene 1) and 150-th frame (scene 2), respectively. We observe from Fig. 11 that HOOI (increasing batch), LRUT, and OSTD perform similarly with a trail of ghostly appearance behind the person in their respective foreground frames. We notice that OSTD and L1-HOOI (increasing batch) perform better with a smoother trail behind the person in their foreground frames. DTA with $\lambda = 0.7$ captures the person in its background, leading to an undesirably smudged foreground estimate. DTA with $\lambda = 0.95$ demonstrates a cleaner foreground estimate, similar to that of adaptive mean (background estimated by the same adaptive mean that we use for DTA), however their backgrounds contain a ghostly appearance of the person. The proposed method extracts a cleaner background and foreground owing to its outlier rejection capability.

We demonstrate the performance after the scene changes at $t = 100$, by presenting the estimated backgrounds and foregrounds at frame index $t = 150$. From Fig. 12, we observe that HOOI, L1-HOOI, OSTD, and LRUT perform poorly because they are not deigned to track changes in the scene. DTA with $\lambda = 0.95$ demonstrates better

Fig. 11. Dynamic video foreground/background separation experiment. (a) Original 75-th frame (scene 1). Background extracted by (b) Adaptive Mean ($\lambda = 0.95$), (c) DTA ($\lambda = 0.95$), (d) DTA ($\lambda = 0.7$), (e) LRUT, (f) OSTD, (g) HOOI (increasing memory), (h) L1-HOOI (increasing memory), and (i) D-L1-TUCKER (proposed). Foreground extracted by (j) Adaptive Mean ($\lambda = 0.95$), (k) DTA ($\lambda = 0.95$), (l) DTA ($\lambda = 0.7$), (m) LRUT, (n) OSTD, (o) HOOI (increasing memory), (p) L1-HOOI (increasing memory), and (q) D-L1-TUCKER (proposed).

performance compared to that of $\lambda = 0.7$ on frame 75, however, on frame 150, we observe that DTA with $\lambda = 0.95$ captures some of the background from scene 1, while DTA with $\lambda = 0.7$ obtains a clean background and hence a smooth foreground, wherein the person appears slightly blurry. The proposed method is capable of tracking scene changes and we observe that it obtains a good estimate of the background and a clear foreground.

To quantify the background/foreground estimation performance, we compute, for every frame, the Peak Signal-to-Noise Ratio (PSNR) defined as $\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\text{MSE}} \right)$, where MSE is the mean squared error of the estimated background from the ground truth (clean) background. In Fig. 13, we plot PSNR versus frame index and observe that all methods begin with high PSNR and, as they process frames with foreground movement, the PSNR drops. We observe that the PSNR of the proposed method is the highest after approximately frame 25. When the scene changes, the PSNR of all methods drops instantaneously. The PSNR values of HOOI, L1-HOOI, LRUT, and OSTD increase

Fig. 12. Dynamic video foreground/background separation experiment. (a) Original 150-th frame (scene 2). Background extracted by (b) Adaptive Mean ($\lambda = 0.95$), (c) DTA ($\lambda = 0.95$), (d) DTA ($\lambda = 0.7$), (e) LRUT, (f) OSTD, (g) HOOI (increasing memory), (h) L1-HOOI (increasing memory), and (i) D-L1-TUCKER (proposed). Foreground extracted by (j) Adaptive Mean ($\lambda = 0.95$), (k) DTA ($\lambda = 0.95$), (l) DTA ($\lambda = 0.7$), (m) LRUT, (n) OSTD, (o) HOOI (increasing memory), (p) L1-HOOI (increasing memory), and (q) D-L1-TUCKER (proposed).

at a low rate as they process frames from the new scene. Adaptive mean and DTA with $\lambda = 0.95$ demonstrate better performance with faster PSNR increase. DTA with $\lambda = 0.7$ adapts to the new scene very quickly, but it is affected by foreground movement (depicted by oscillations in its PSNR values). The proposed method adapts to the new scene after it processes $W = 20$ measurements and attains the highest PSNR values across all frame indices thereafter. The PSNR values of the proposed method drop and stay constant for $W = 20$ frames after scene change because the first $W = 20$ points fail the reliability check and do not participate in the subspace update. After $W = 20$ frames, the ambiguity memory fills up and the proposed method rapidly adapts to the new scene. Right after adaptation, the proposed method is straightforwardly capable of extracting the background and foreground of all ambiguous frames in $\mathcal{W}$ in a retroactive fashion, leading to a better performance during scene change as shown in Fig. 13.

Fig. 13. Dynamic video foreground/background separation experiment. PSNR (dB) versus frame index.

Fig. 14. Dynamic video foreground/background separation experiment. Cumulative time (sec) versus frame index.

In Fig. 14, we plot the cumulative run-time versus frame index for the compared methods. For clarity in presentation, we exclude OSTD from this figure, as it is significantly slower and plotting its performance would need a change of scale. For instance, the cumulative run-time of OSTD at frame indices 50, 100, 150, and 200 is (approximately) 390, 762, 969, and 1173 seconds, respectively. Among the other methods, we observe that the increasing memory implementations of HOOI and L1-HOOI consume the highest amount of time, as expected, because they process an increasing number of frames. DTA is the fastest among the tested methods, followed by D-L1-Tucker and LRUT.

D. Online Tucker and Classification

In this experiment, we perform joint Tucker feature extraction and online classification. We use the Extended Yale Face Database B [50], consisting of face images of many individuals, captured at varying illuminations. For this experiment, we use the face images of subject 02 and subject 23 to perform binary classification. Each class has 64 images in total, out of which, at every realization, we choose 45 for training and the remaining 19 for testing. The size of each image is originally 192×168 and we down-sample it to 96×84 . Therefore, at every realization, we have a tensor with training data $\mathcal{X} \in \mathbb{R}^{96 \times 84 \times 90}$ containing 90 measurements in total (45 from each class) and a tensor with testing data $\mathcal{Y} \in \mathbb{R}^{96 \times 84 \times 38}$ containing 38 measurements in total (19 from each class). At every realization, we arbitrarily shuffle the training data and follow a parallel online feature extraction and classification approach as follows.

At update index t , we process the t training sample in $\mathcal{X}$, $\bar{\mathcal{X}}_t$, and update $\mathbf{Q}_1^{(t)}$ and $\mathbf{Q}_2^{(t)}$ using the proposed method ($B = 5$, $M = 10$, $d_1 = 15$, $d_2 = 6$, and cross-validated τ). Next, we use the updated bases to compress all previously seen training data $\{\bar{\mathcal{X}}_i\}_{i \in [t]}$ as $\mathcal{Z}_i = \mathbf{Q}_1^{(t)\top} \bar{\mathcal{X}}_i \mathbf{Q}_2^{(t)}$. We vectorize the compressed training measurements and give them as input to the the Online Support Vector Machine (OSVM) classifier of [51]⁴. We test the performance of the classifier on testing data, compressed using the same bases, and record the classification accuracy for every update index t . We repeat the experiment 300 times and plot the average classification accuracy versus update index in Fig. 15. Along with the proposed algorithm, we also plot the performance of the plain OSVM classifier, i.e., OSVM classifier run on vectorized (uncompressed) data, DTA with $\lambda = 0.33$, and OSTD. In Fig. 15, we observe that all compared methods attain almost identical performance. The classification accuracy starts low and as the update index increases it tends to 1.

Next, we repeat the experiment with the same setting, this time corrupting each training measurement outside the initial memory batch $\mathcal{B}$, with noise from $\mathcal{N}(2, 5)$, cropping pixel intensities outside $[0, 255]$. We compute the average classification accuracy over 300 realizations and plot it versus update index in Fig. 15. In this case, we notice that plain OSVM is significantly affected by the noise. DTA, OSTD, and D-L1-Tucker demonstrate resistance to noise corruption, especially for early update indices.

E. Online Anomaly Detection

We consider the ‘‘Uber Pickups’’ tensor of the Formidable Repository of Open Sparse Tensors and Tools (FROSTT) [52] which is a $(N = 4)$ -way tensor of size D_1 -by- D_2 -by- D_3 -by- D_4 where $D_1 = 1140$ latitudes, $D_2 = 1717$ longitudes, $D_3 = 24$ hours, and $D_4 = 183$ days. Each entry of the tensor models number of Uber pickups in New York City over a period of about 6 months.

Pre-processing: To make the tensor more manageable in size, we first take a summation across the day mode and obtain a size D_1 -by- D_2 -by- D_3 tensor where $D_3 = 183$ days; i.e., a collection of 183 size 1140-by-1717 matrix

⁴Matlab code available at <https://www.cpdiehl.org/code.html>.

Fig. 15. Online tensor compression and classification experiment. Average classification accuracy versus update index.

Fig. 16. $B = 5$, $M = 5$, $d = 10$, 300 realizations, data driven τ .

measurements. We further reduce the size of the matrix measurements by retaining a 250-by-250 area centered at Manhattan wherein most of the activity—in terms of Uber Pickups—occurs. We consider the resulting tensor $\mathcal{X}_{\text{uber}} \in \mathbb{R}^{250 \times 250 \times 183}$ to be a collection of 183 streaming measurements, one for each day.

Streaming processing: $\mathcal{X}_{\text{uber}}$ can be seen as a data stream of matrix measurements each of which corresponds to a day. Accordingly, 7 successive measurements across the day index must correspond to a week, which, in turn, is separated into weekdays and Saturdays. We assume that traffic during the weekdays is not the same as traffic on Saturdays and conjecture that weekdays belong in a coherent class/distribution while Saturdays belong to another.

We assume that we are given $B = 5$ measurements that correspond to weekdays and use those measurements to

initialize D-L1-Tucker with memory size $M = 5$ and $d = 10$ components per mode. Moreover, we leverage these B measurements to tune τ using the leave-one-out cross-validation approach that we presented above. We set τ to the median value of the B collected reliability values. Then, we update the decomposition of D-L1-Tucker by processing the rest of the measurements one by one. In Fig. 16, we plot the reliability of streaming measurements versus day (update) index. Moreover, we add the data driven value of τ as a horizontal line. Each reliability value above this curve is deemed reliable for processing, while each reliability value below that curve is considered to be an anomaly (outlier). For better understanding, we include vertical dotted lines on the day indices which correspond to Saturdays. The reported curves are averaged over 300 random initializations of bases and runs of D-L1-Tucker on $\mathcal{X}_{\text{uber}}$. Quite consistently, days that correspond to Saturdays exhibit reliability values that are clearly below the τ threshold and are considered anomalies. In contrast, almost all weekdays exhibit reliability values above the τ threshold. A different selection of the threshold τ may slightly improve the results, however, even with the data driven tuning of τ , the reliability check feature of D-L1-Tucker offers high accuracy in identifying anomalies/outliers.

VI. CONCLUSIONS

When tensor measurements arrive in a streaming fashion or are too many to jointly decompose, incremental Tucker analysis is preferred. In addition, dynamic basis adaptation is meaningful when the nominal data subspaces change. At the same time, outliers in the data can significantly compromise the performance of existing methods for dynamic Tucker analysis. In this work, we presented D-L1-Tucker: an algorithm for dynamic and outlier-resistant Tucker analysis of tensor data. Our experimental studies on real and synthetic datasets corroborate that the proposed method (i) attains high basis estimation performance, (ii) suppresses outliers, and (iii) adapts to nominal subspace changes.

REFERENCES

- [1] 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 Process. Mag.*, vol. 32, no. 2, pp. 145–163, 2015.
- [2] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jul. 2017.
- [3] D. Tao, X. Li, W. Hu, S. Maybank, and X. Wu, "Supervised tensor learning," in *IEEE Int. Conf. Data Mining*, Houston, TX, Nov. 2005, pp. 8–pp.
- [4] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.
- [5] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.
- [6] H. Q. Ngo and E. G. Larsson, "EVD-based channel estimation in multicell multiuser MIMO systems with very large antenna arrays," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, Kyoto, Japan, Mar. 2012, pp. 3249–3252.
- [7] M. B. Amin, W. Zirwas, and M. Haardt, "HOSVD-based denoising for improved channel prediction of weak massive MIMO channels," in *IEEE Vehicular Technol. Conf.*, Sydney, Australia, Jun. 2017, pp. 1–5.

- [8] D. C. Araújo, A. L. De Almeida, J. P. Da Costa, and R. T. De Sousa, “Tensor-based channel estimation for massive MIMO-OFDM systems,” *IEEE Access*, vol. 7, pp. 42 133–42 147, 2019.
- [9] J. Yang, D. Zhang, A. F. Frangi, and J.-y. Yang, “Two-dimensional PCA: a new approach to appearance-based face representation and recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 131–137, 2004.
- [10] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, “TTHRESH: Tensor compression for multidimensional visual data,” *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 9, pp. 2891–2903, 2020.
- [11] H. Ben-Younes, R. Cadene, M. Cord, and N. Thome, “Mutan: Multimodal tucker fusion for visual question answering,” in *IEEE Int. Conf. Comput. Vision*, Venice, Italy, Oct. 2017, pp. 2612–2620.
- [12] A. Koochakzadeh and P. Pal, “On canonical polyadic decomposition of overcomplete tensors of arbitrary even order,” in *IEEE Int. Workshop Comput. Advances Multi-Sensor Adaptive Process.*, Curaçao, Dutch Antilles, Dec. 2017, pp. 1–5.
- [13] 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, pp. 16:1–16:44, Jan. 2017.
- [14] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.
- [15] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.
- [16] L. De Lathauwer, B. De Moor, and J. Vandewalle, “A multilinear singular value decomposition,” *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 1253–1278, 2000.
- [17] V. Barnett and T. Lewis, *Outliers in statistical data.*, 2nd ed. John Wiley & Sons Ltd., 1978.
- [18] J. W. Tukey, “The future of data analysis,” *The annals of mathematical statistics*, vol. 33, no. 1, pp. 1–67, 1962.
- [19] X. Fu, K. Huang, W. K. Ma, N. D. Sidiropoulos, and R. Bro, “Joint tensor factorization and outlying slab suppression with applications,” *IEEE Trans. Signal Process.*, vol. 63, pp. 6315–6328, Dec. 2015.
- [20] D. Goldfarb and Z. Qin, “Robust low-rank tensor recovery: Models and algorithms,” *SIAM J. Matrix Anal. Appl.*, vol. 35, no. 1, pp. 225–253, 2014.
- [21] P. P. Markopoulos, D. G. Chachlakis, and E. E. Papalexakis, “The Exact Solution to Rank-1 L1-norm TUCKER2 Decomposition,” *IEEE Signal Process. Lett.*, vol. 25, no. 4, pp. 511–515, Jan. 2018.
- [22] D. G. Chachlakis, A. Prater-Bennette, and P. P. Markopoulos, “L1-norm Tucker tensor decomposition,” *IEEE Access*, vol. 7, pp. 178 454–178 465, 2019.
- [23] Y. Pang, X. Li, and Y. Yuan, “Robust tensor analysis with L1-norm,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, pp. 172–178, Feb. 2010.
- [24] D. G. Chachlakis and P. P. Markopoulos, “Novel algorithms for exact and efficient L1-norm-based TUCKER2 decomposition,” in *IEEE Int. Conf. Acoust., Speech, Signal Process.*, Calgary, Canada, Apr. 2018, pp. 6294–6298.
- [25] D. G. Chachlakis and P. P. Markopoulos, “Robust decomposition of 3-way tensors based on L1-norm,” in *SPIE Defense and Commercial Sens.*, Orlando, FL, Apr. 2018, pp. 1 065 807:1–1 065 807:15.
- [26] X. Cao, X. Wei, Y. Han, and D. Lin, “Robust face clustering via tensor decomposition,” *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2546–2557, 2015.
- [27] K. Tountas, D. G. Chachlakis, P. P. Markopoulos, and D. A. Pados, “Iteratively re-weighted L1-PCA of tensor data,” in *IEEE Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, 2019, pp. 1658–1661.
- [28] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, “Incremental tensor analysis: Theory and applications,” *ACM Trans. Knowl. Discovery Data*, vol. 2, no. 3, pp. 1–37, 2008.

- [29] J. Sun, D. Tao, and C. Faloutsos, “Beyond streams and graphs: Dynamic tensor analysis,” in *ACM Int. Conf. Knowl. Discovery Data mining*, Philadelphia, PA, Aug. 2006, pp. 374–383.
- [30] R. Yu, D. Cheng, and Y. Liu, “Accelerated online low rank tensor learning for multivariate spatiotemporal streams,” in *Proc. Int. Conf. Machine Learn.*, Lille, France, Jul. 2015, pp. 238–247.
- [31] P. P. Markopoulos, G. N. Karystinos, and D. A. Pados, “Optimal algorithms for L1-subspace signal processing,” *IEEE Trans. Signal Process.*, vol. 62, pp. 5046–5058, Oct. 2014.
- [32] P. P. Markopoulos, D. G. Chachlakis, and A. Prater-Bennette, “L1-norm Higher-Order Singular-value Decomposition,” in *IEEE Global Conf. Signal Inf. Process.*, Anaheim, CA, Nov. 2018, pp. 1353–1357.
- [33] D. G. Chachlakis, M. Dhanaraj, A. Prater-Bennette, and P. P. Markopoulos, “Options for multimodal classification based on L1-tucker decomposition,” in *SPIE Defense Commercial Sens.*, Apr. 2019, pp. 109 890O:1–109 890O:13.
- [34] D. G. Chachlakis, A. Prater-Bennette, and P. P. Markopoulos, “L1-norm higher-order orthogonal iterations for robust tensor analysis,” in *IEEE Int. Conf. Acoust. Speech Signal Process.*, Barcelona, Spain (online), May 2020, pp. 4826–4830.
- [35] P. P. Markopoulos, S. Kundu, S. Chamadia, and D. A. Pados, “Efficient L1-norm Principal-Component Analysis via bit flipping,” *IEEE Trans. Signal Process.*, vol. 65, pp. 4252–4264, Aug. 2017.
- [36] M. Baskaran, M. H. Langston, T. Ramanandran, D. Bruns-Smith, T. Henretty, J. Ezick, and R. Lethin, “Accelerated low-rank updates to tensor decompositions,” in *IEEE High Perf. Extreme Comput. Conf.*, Waltham, MA, Sep. 2016, pp. 1–7.
- [37] O. A. Malik and S. Becker, “Low-rank Tucker decomposition of large tensors using tensorsketch,” in *Adv. Neural Inf. Process. Syst.*, 2018, pp. 10 096–10 106.
- [38] R. Pagh, “Compressed matrix multiplication,” *ACM Trans. Comput. Theory*, vol. 5, no. 3, pp. 1–17, 2013.
- [39] Y. Sun, Y. Guo, C. Luo, J. Tropp, and M. Udell, “Low-rank Tucker approximation of a tensor from streaming data,” *arXiv preprint arXiv:1904.10951*, 2019.
- [40] 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,” *Int. J. Comput. Vision*, vol. 91, no. 3, pp. 303–327, 2011.
- [41] X. Li, W. Hu, Z. Zhang, X. Zhang, and G. Luo, “Robust visual tracking based on incremental tensor subspace learning,” in *IEEE Int. Conf. Comput. Vision*, Rio de Janeiro, Brazil, Oct. 2007, pp. 1–8.
- [42] 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 *Int. Conf. Image Anal. Recogn.*, Algarve, Portugal, Oct. 2014, pp. 94–103.
- [43] A. Sobral, S. Javed, S. Ki Jung, T. Bouwmans, and E.-h. Zahzah, “Online tensor decomposition for background subtraction in multispectral video sequences,” in *IEEE Int. Conf. Comput. Vision*, Santiago, Chile, Dec. 2015, pp. 106–113. [Online]. Available: <https://github.com/andrewssobral/ostd>
- [44] S. Smith, K. Huang, N. D. Sidiropoulos, and G. Karypis, “Streaming tensor factorization for infinite data sources,” in *Proc. SIAM Int. Conf. Data Mining*, San Diego, CA, May 2018, pp. 81–89.
- [45] Y. Du, Y. Zheng, K.-c. Lee, and S. Zhe, “Probabilistic streaming tensor decomposition,” in *Proc. IEEE Int. Conf. Data Mining*, Singapore, Nov. 2018, pp. 99–108.
- [46] M. Najafi, L. He, and S. Y. Philip, “Outlier-robust multi-aspect streaming tensor completion and factorization,” in *Int. Joint Cond. Artificial Intell.*, Macao, China, Aug. 2019, pp. 3187–3194.
- [47] 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 Networks Learn. Syst.*, vol. 30, no. 4, pp. 1061–1075, 2018.
- [48] P. P. Markopoulos, M. Dhanaraj, and A. Savakis, “Adaptive L1-norm principal-component analysis with online outlier rejection,” *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 6, pp. 1131–1143, 2018.
- [49] Context aware vision using image-based active recognition (CAVIAR). [Online]. Available: <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/>. Accessed on: Jul. 2020.
- [50] The Extended Yale Face Database B. [Online]. Available: <http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html>. Accessed on: Jul. 2020.
- [51] C. P. Diehl and G. Cauwenberghs, “SVM incremental learning, adaptation and optimization,” in *Proc. IEEE Int. Joint Conf. Neural Net.*, Portland, OR, Jul. 2003, pp. 2685–2690.
- [52] S. Smith, J. W. Choi, J. Li, R. Vuduc, J. Park, X. Liu, and G. Karypis. (2017) FROSTT: The formidable repository of open sparse tensors and tools. [Online]. Available: <http://frostdt.io/>