Adaptive Algorithms to Track the PARAFAC Decomposition of a Third-Order Tensor

Dimitri Nion, *Associate Member, IEEE*, and Nicholas D. Sidiropoulos, *Fellow, IEEE*

***Abstract*—The PARAFAC decomposition of a higher-order tensor is a powerful multilinear algebra tool that becomes more and more popular in a number of disciplines. Existing PARAFAC algorithms are computationally demanding and operate in batch mode—both serious drawbacks for on-line applications. When the data are serially acquired, or the underlying model changes with time, adaptive PARAFAC algorithms that can track the sought decomposition at low complexity would be highly desirable. This is a challenging task that has not been addressed in the literature, and the topic of this paper. Given an estimate of the PARAFAC decomposition of a tensor at instant t, we propose two adaptive algorithms to update the decomposition at instant $t+1$, the new tensor being obtained from the old one after appending a new slice in the 'time' dimension. The proposed algorithms can yield estimation performance that is very close to that obtained via repeated application of state-of-art batch algorithms, at orders of magnitude lower complexity. The effectiveness of the proposed algorithms is illustrated using a MIMO radar application (tracking of directions of arrival and directions of departure) as an example.**

***Index Terms*—Adaptive algorithms, DOA/DOD tracking, higher-order tensor, MIMO radar, PARAllel FACtor (PARAFAC).**

I. Introduction

IN signal processing, subspace-based methods are key to many applications such as direction finding (ESPRIT [1], MUSIC [2]), blind beamforming [3], blind channel identification [4], and speech dereverberation [5]. These methods exploit the orthogonality between the signal and noise subspaces of the observed matrix. If these subspaces are varying with time, they have to be tracked. However, the computation of a full singular value decomposition (SVD) at every sampling instant is not suitable in real-time applications, for complexity reasons. In this context, the development of SVD tracking algorithms has been—and still is—under intensive research (see, e.g., [6]–[9], and references therein).

In an increasing number of applications, the observed data have a multiway structure, i.e., the samples are indexed by three or more independent indices, giving rise to a higher-order tensor or multiway array, rather than a matrix (two-way array). Exploitation of this structure requires the use of signal processing tools based on multilinear algebra rather than standard linear algebra [10]. The Tucker decomposition/higher-order singular value decomposition (HOSVD) [11]–[13] is a possible generalization of the matrix SVD to higher-order tensors, which has recently found important applications in face recognition [14] and image texture analysis [15]. The PARAllel FACtor (PARAFAC) decomposition [16], [17] is another possible generalization of the matrix SVD to higher-order tensors. PARAFAC is tied to the concept of tensor rank and low-rank decomposition, and its distinguishing characteristic is its uniqueness properties. The decomposition of a tensor in a sum of rank-one tensors has a long history going back to Hitchock in 1927 [18], [19]; but uniqueness was touched upon by Cattell in 1944 [20] and fleshed out by Harshman in 1970 [16]. Unlike the matrix case, low-rank PARAFAC decomposition can be unique for rank higher than one, and this is a key strength of PARAFAC. Since 1970, PARAFAC has slowly found its way in various disciplines such as Chemometrics and food technology [21], exploratory data analysis [22], wireless communications and array processing [23], [24], blind source separation [25]–[27]. In these applications, the PARAFAC decomposition is always computed via off-line optimization algorithms [28], which are computationally demanding. In many cases, however, the data are serially acquired, or the underlying data-generating process varies with time, and thus cannot be globally modeled by a low-rank PARAFAC model. While in both cases one can run a batch algorithm repeatedly to account for new data or model variation, and in fact the previous estimated model can be used to initialize the next run of the batch algorithm, this is still computationally costly. The reason is that existing batch PARAFAC algorithms are relatively complex and iterative in nature, requiring many steps to convergence, even if well initialized. It is, therefore, of great practical interest to develop adaptive algorithms that track the PARAFAC decomposition. To our knowledge, this problem has not been addressed in the literature before.

In this paper, we consider the problem of tracking the PARAFAC decomposition of a third-order tensor of which one dimension is "time." The observed tensor at time $t+1$ is obtained from that at time t after appending a new slice in the time dimension. To solve this problem, we propose two different adaptive algorithms. The first algorithm, called Simultaneous Diagonalization Tracking (PARAFAC-SDT), is the adaptive version of the batch algorithm based on simultaneous diagonalization (SD) proposed in [29]. The second algorithm is based on the recursive minimization of a weighted least-squares

Manuscript received September 24, 2008; accepted January 30, 2009. First published March 10, 2009; current version published May 15, 2009. The associate editor coordinating the review of this paper and approving it for publication was Dr. Zhengyuan (Daniel) Xu. The work of D. Nion was supported by a Postdoctoral Grant from the Délégation Générale pour l'Armement (DGA) via ETIS Lab., UMR 8051 (ENSEA, CNRS, Univ. Cergy-Pontoise), France.

The authors are with the Department of Electronic and Computer Engineering, Technical University of Crete, 73100 Chania, Greece (e-mail: nion@telecom.tuc.gr; nikos@telecom.tuc.gr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2009.2016885

criterion and is called PARAFAC-Recursive Least Squares Tracking (RLST). These algorithms can be used either with a sliding or an exponential window. Moreover, their respective complexity is very low compared to their batch counterpart and in this sense, they can be considered as a first step towards real-time PARAFAC-based applications. Finally, the good tracking capabilities of these novel algorithms are illustrated through the application of tracking the direction of arrival (DOA) and direction of departure (DOD) of multiple targets in a multiple-input multiple-output (MIMO) radar system.

This paper is organized as follows. Some multilinear algebra prerequisites are introduced in Section II, and the core problem considered is set up in Section III. Section IV outlines the basic idea upon which both adaptive algorithms will be subsequently built. Section V explains how the observed data are windowed before being processed. Sections VI and VII flesh out the proposed PARAFAC-SDT and PARAFAC-RLST algorithms, respectively. Section VIII is a brief note on initialization of these algorithms, and Section IX provides simulation results for the MIMO radar tracking application. Finally, Section X summarizes conclusions.

Notation: A third-order tensor of size $I \times J \times K$ is denoted by a calligraphic letter $\mathcal{X}$, and its elements are denoted by x_{ijk}, $i = 1, \dots, I$, $j = 1, \dots, J$ and $k = 1, \dots, K$. A bold-face capital letter $\mathbf{X}$ denotes a matrix and a bold-face lower-case letter $\mathbf{x}$ a vector. The transpose, complex conjugate, complex conjugate transpose and pseudo-inverse are denoted by $\mathbf{X}^T$, $\mathbf{X}^*$, $\mathbf{X}^H$, and $\mathbf{X}^\dagger$, respectively. $\|\mathbf{X}\|$ is the Frobenius norm of $\mathbf{X}$. $\mathrm{vec}(\mathbf{X})$ is a vector built by stacking the columns of $\mathbf{X}$ one above each other. $\mathrm{diag}(\mathbf{x})$ is a diagonal matrix that holds $\mathbf{x}$ on its diagonal. The Kronecker product is denoted by $\otimes$. The Khatri-Rao product (or column-wise Kronecker product) is denoted by $\odot$, i.e., $[\mathbf{a}_1, \dots, \mathbf{a}_I] \odot [\mathbf{b}_1, \dots, \mathbf{b}_I] = [\mathbf{a}_1 \otimes \mathbf{b}_1, \dots, \mathbf{a}_I \otimes \mathbf{b}_I]$. The $P \times P$ identity matrix is denoted by $\mathbf{I}_P$. We will also use a Matlab-type notation for matrix subblocks, i.e., $[\mathbf{X}]_{l:m,n:p}$ represents the matrix built after selection of $m-l+1$ rows of $\mathbf{X}$, from the lth to the mth, and $p-n+1$ columns of $\mathbf{X}$, from the nth to the pth. $[\mathbf{X}]_{:,n:p}$ is used to denote selection of all rows and $[\mathbf{X}]_{l:m,:}$ to denote selection of all columns.

II. Multilinear Algebra Prerequisites

A. Definitions

A Nth order tensor is an N-way array, i.e., its elements are addressed by N indices. In this paper, third-order tensors only will be used. Before introducing PARAFAC, we need the following definitions.

Definition 1. (Outer Product): The outer product $(\mathbf{a} \circ \mathbf{b} \circ \mathbf{c})$ of three vectors $\mathbf{a} \in \mathbb{C}^{I \times 1}$, $\mathbf{b} \in \mathbb{C}^{J \times 1}$ and $\mathbf{c} \in \mathbb{C}^{K \times 1}$, is a third-order tensor $\mathcal{X} \in \mathbb{C}^{I \times J \times K}$ with elements defined by $x_{ijk} = a_i b_j c_k$, for all values of the indices.

Definition 2. (Rank-1 Tensor): A third-order tensor is rank-1 if it can be written as the outer-product of three vectors.

Definition 3. (Tensor Rank): The rank of $\mathcal{X}$ is defined as the minimum number of rank-1 tensors whose sum yields $\mathcal{X}$.

Definition 4. (Matrix Representations of a Tensor): The three standard matrix representations of a third-order tensor $\mathcal{X} \in \mathbb{C}^{I \times J \times K}$, denoted by $\mathbf{X}^{(1)} \in \mathbb{C}^{IK \times J}$, $\mathbf{X}^{(2)} \in \mathbb{C}^{JI \times K}$ and $\mathbf{X}^{(3)} \in \mathbb{C}^{KJ \times I}$ are defined by $[\mathbf{X}^{(1)}]_{(i-1)K+k,j} = x_{ijk}$, $[\mathbf{X}^{(2)}]_{(j-1)I+i,k} = x_{ijk}$ and $[\mathbf{X}^{(3)}]_{(k-1)J+j,i} = x_{ijk}$,, respectively.

B. PARAFAC Decomposition

We now have enough material to define the PARAFAC decomposition.

Definition 5. (PARAFAC in Tensor Format): The Parallel Factor Decomposition [16] of a tensor $\mathcal{X} \in \mathbb{C}^{I \times J \times K}$ is a decomposition of $\mathcal{X}$ as a sum of a minimal number of rank-1 tensors

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{1}$$

where $\mathbf{a}_r$, $\mathbf{b}_r$, $\mathbf{c}_r$ are the rth columns of the so-called "loading matrices" $\mathbf{A} \in \mathbb{C}^{I \times R}$, $\mathbf{B} \in \mathbb{C}^{J \times R}$ and $\mathbf{C} \in \mathbb{C}^{K \times R}$, respectively.

The PARAFAC decomposition can also be written in matrix format.

Definition 6. (PARAFAC in Matrix Format): The three matrix representations of a tensor $\mathcal{X} \in \mathbb{C}^{I \times J \times K}$, that follows the PARAFAC decomposition (1), are linked to the loading matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ as follows:

$$\begin{cases} \mathbf{X}^{(1)} = (\mathbf{A} \odot \mathbf{C})\mathbf{B}^T \\ \mathbf{X}^{(2)} = (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T \\ \mathbf{X}^{(3)} = (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T. \end{cases}$$

A key feature of PARAFAC is its uniqueness property under mild conditions. The PARAFAC decomposition of $\mathcal{X}$ is said to be *essentially unique* if any matrix triplet $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}})$ that also fits the model is related to $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ via

$$\mathbf{A} = \tilde{\mathbf{A}}\mathbf{\Pi}\mathbf{\Lambda}_1 \quad \mathbf{B} = \tilde{\mathbf{B}}\mathbf{\Pi}\mathbf{\Lambda}_2 \quad \mathbf{C} = \tilde{\mathbf{C}}\mathbf{\Pi}\mathbf{\Lambda}_3 \tag{2}$$

with $\mathbf{\Lambda}_1$, $\mathbf{\Lambda}_2$, $\mathbf{\Lambda}_3$ arbitrary diagonal matrices satisfying $\mathbf{\Lambda}_1\mathbf{\Lambda}_2\mathbf{\Lambda}_3 = \mathbf{I}_R$, and $\mathbf{\Pi}$ an arbitrary permutation matrix. PARAFAC uniqueness is studied in [29], [30]–[32]. A specific case is where two loading matrices are full column rank and the third does not contain colinear columns. In this case, PARAFAC is unique up to its trivial indeterminacies [33]. In [29], a uniqueness bound was derived for the case where one matrix is full column rank and the others are full rank. This result is summarized in the following theorem.

Theorem 1: Assume $\mathbf{A} \in \mathbb{C}^{I \times R}$ and $\mathbf{C} \in \mathbb{C}^{K \times R}$ are drawn from a jointly continuous distribution with respect to the Lebesgue measure in $\mathbb{C}^{I \times R + K \times R}$, and $\mathbf{B} \in \mathbb{C}^{J \times R}$ is full column rank. If

$$R \geq 2 \quad \text{and} \quad I(I-1)K(K-1) \geq 2R(R-1) \tag{3}$$

then the PARAFAC decomposition of $\mathcal{X}$ is essentially unique, almost surely.

Note that for $R = 1$, the PARAFAC decomposition is (trivially) always unique.

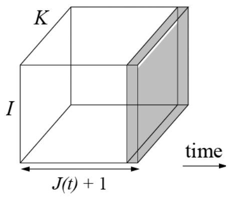


Fig. 1. Acquisition of a new slice in the tensor of observations.

III. Problem Statement

Let us consider an estimate of the PARAFAC decomposition of a third-order tensor $\mathcal{X}(t) \in \mathbb{C}^{I\times J(t)\times K}$, at time t

$$\mathbf{X}^{(1)}(t) \simeq \mathbf{H}(t)\mathbf{B}^T(t) \tag{4}$$

where $\mathbf{H}(t) \overset{\text{def}}{=} \mathbf{A}(t) \odot \mathbf{C}(t)$ has dimensions $IK \times R$, $\mathbf{B}(t) \in \mathbb{C}^{J(t)\times R}$ has a dimension growing with time and $\mathbf{X}^{(1)}(t)$ is the $IK \times J(t)$ matrix representation of $\mathcal{X}(t)$.

In practice, one possible interpretation of this model is the following: $\mathbf{B}(t)$ holds $J(t)$ successive data-vectors of interest (e.g., samples of R transmitted signals), $\mathbf{H}(t)$ is a time-varying unknown transformation of these vectors (e.g., a time-varying channel) and $\mathbf{X}^{(1)}(t)$ holds the successive observed vectors.

Let $\mathcal{X}(t+1) \in \mathbb{C}^{I\times J(t+1)\times K}$ be obtained from $\mathcal{X}(t)$ after appending a new observed slice in the second dimension, as illustrated by Fig. 1, such that $J(t+1) = J(t)+1$. An estimate of the PARAFAC decomposition of $\mathcal{X}(t+1)$ is

$$\mathbf{X}^{(1)}(t+1) \simeq \mathbf{H}(t+1)\mathbf{B}^T(t+1) \tag{5}$$

where $\mathbf{H}(t+1) = \mathbf{A}(t+1) \odot \mathbf{C}(t+1)$. In order to estimate the new loading matrices $\mathbf{A}(t+1)$, $\mathbf{B}(t+1)$ and $\mathbf{C}(t+1)$, one can optimally fit a PARAFAC model on $\mathcal{X}(t+1)$ by resorting to batch algorithms [28]. However, though initialization of such algorithms with the old estimates $\mathbf{A}(t)$, $\mathbf{B}(t)$ and $\mathbf{C}(t)$ may speed up convergence, the computation of a whole new PARAFAC decomposition at each sampling instant is not suitable for online applications. For instance, one cycle of the batch Alternating Least Squares (ALS) algorithm [17], [23] requires the computation of three pseudo-inverses, and convergence often requires many cycles, even if the algorithm is well-initialized. The development of adaptive algorithms to track the PARAFAC decomposition is thus a key step towards real-time PARAFAC-based applications. Following these considerations, the core problem we propose to solve in this paper can be summarized as follows.

Problem: Given estimates $\mathbf{A}(t)$, $\mathbf{B}(t)$ and $\mathbf{C}(t)$ for the PARAFAC decomposition in R terms of the $I \times J(t) \times K$ tensor $\mathcal{X}(t)$, find recursive updates for $\mathbf{A}(t+1)$, $\mathbf{B}(t+1)$ and $\mathbf{C}(t+1)$, which stand for estimates of the PARAFAC decomposition in R terms of the $I \times J(t+1) \times K$ tensor $\mathcal{X}(t+1)$, the latter being obtained from $\mathcal{X}(t)$ after appending a new slice in the second dimension.

We will work under the following assumptions:

A1) All entries of $\mathbf{A}$ and $\mathbf{C}$ may change between t and $t+1$, according to an *unknown*, but *slowly* time-varying model.

A2) The conditions in Theorem 1 are satisfied $\forall t$, i.e., uniqueness can be assumed at each sampling instant.

A3) R is known or has been estimated and does not vary with time. Under the conditions of Theorem 1, R can be determined from the rank of matrix $\mathbf{X}^{(1)}(t)$, but the tracking problem is seriously compounded if rank changes with time, even in the matrix SVD case. We therefore leave this issue for future work.

IV. Sketch of the Basic Idea

In this section, we propose a first approach towards adaptive computation of the PARAFAC decomposition of $\mathcal{X}(t+1)$, given that of $\mathcal{X}(t)$. Though formulated in terms of time-consuming operations (mainly pseudoinverse and SVD), the following steps allow to draw the skeleton of the fully adaptive algorithms to be derived next.

Let $\mathbf{x}(t+1)$ be the $IK \times 1$ vectorized representation of the new slice appended to $\mathcal{X}(t)$, such that

$$\mathbf{X}^{(1)}(t+1) = [\mathbf{X}^{(1)}(t), \mathbf{x}(t+1)]. \tag{6}$$

Let us consider a smooth variation of $\mathbf{H}$ between t and $t+1$, i.e., $\mathbf{H}(t) \simeq \mathbf{H}(t+1)$. From (4), (5) and (6), we get

$$\mathbf{B}^T(t+1) \simeq [\mathbf{B}^T(t), \mathbf{b}^T(t+1)] \tag{7}$$

which means that $\mathbf{B}$ has approximately a time-shift structure. An initial estimate of $\mathbf{b}(t+1)$ is thus given in the least-squares sense by

$$\mathbf{b}^T(t+1) = \mathbf{H}^\dagger(t)\mathbf{x}(t+1) \tag{8}$$

after which the time shift-structure in (7) is exploited to build $\mathbf{B}(t+1)$. The least-squares update of $\mathbf{H}(t+1)$ is then given by

$$\mathbf{H}(t+1) = \mathbf{X}^{(1)}(t+1)(\mathbf{B}^T(t+1))^\dagger. \tag{9}$$

Given $\mathbf{H}(t+1)$, one can then re-update $\mathbf{b}(t+1)$ by substituting $\mathbf{H}(t)$ by $\mathbf{H}(t+1)$ in (8). Finally, since $\mathbf{H}(t+1)$ is an estimate for $\mathbf{A}(t+1) \odot \mathbf{C}(t+1)$, it provides estimates for $\mathbf{A}(t+1)$ and $\mathbf{C}(t+1)$, up to the trivial indeterminacies inherent to the PARAFAC model. In fact, the rth column of $\mathbf{A}(t+1)\odot\mathbf{C}(t+1)$ is $\mathbf{a}_r(t+1) \otimes \mathbf{c}_r(t+1) = \mathrm{vec}(\mathbf{c}_r(t+1)\mathbf{a}_r^T(t+1))$, so it corresponds to the vectorized representation of a rank-1 matrix. Consequently, $\mathbf{c}_r(t+1)$ can be estimated as the principal left singular vector multiplied by the corresponding singular value of $\mathrm{unvec}(\mathbf{a}_r(t+1)\otimes\mathbf{c}_r(t+1))$, while $\mathbf{a}_r(t+1)$ can be estimated as the conjugate principal right singular vector. This procedure is repeated for the R columns.

It is important to note that the least-squares update of $\mathbf{H}(t+1)$ in (9) ignores the Khatri-Rao product structure. As a result, the overall procedure (even if iterated) is not equivalent to ALS PARAFAC model fitting. This implies that estimation performance will be worse than that of the batch ALS algorithm, however this is to be expected in return for a far simpler adaptive algorithm, and we will see in our simulations in Section IX that

TABLE I
FIRST APPROACH TO TRACK THE PARAFAC DECOMPOSITION

```
STEP 1
Suppose $\mathbf{H}(t) \simeq \mathbf{H}(t+1)$ and get a first estimate of $\mathbf{b}(t+1)$:
$\mathbf{b}^T(t+1) = \mathbf{H}^\dagger(t)\mathbf{x}(t+1)$.

STEP 2
Get a first estimate of $\mathbf{B}(t+1)$: $\mathbf{B}^T(t+1) = [\mathbf{B}^T(t), \mathbf{b}^T(t+1)]$.

STEP 3
Estimate $\mathbf{H}(t+1)$: $\mathbf{H}(t+1) = \mathbf{X}^{(1)}(t+1)(\mathbf{B}^T(t+1))^\dagger$.

STEP 4
Estimate $\mathbf{A}(t+1)$ and $\mathbf{C}(t+1)$ from $\mathbf{H}(t+1)$:
For $r = 1 \ldots R$, Do
    $\mathbf{H}_r(t+1) = \mathrm{unvec}([\mathbf{H}(t+1)]_{:,r})$
    $[\mathbf{c}_r, \sigma_r, \mathbf{a}_r] = \mathrm{svd}(\mathbf{H}_r(t+1))$,   — = principal singular vectors —
    $[\mathbf{C}(t+1)]_{:,r} = \sigma_r \mathbf{c}_r$ and $[\mathbf{A}(t+1)]_{:,r} = \mathbf{a}_r^*$.
End

STEP 5
Re-estimate $\mathbf{B}(t+1)$ with a time-shift structure:
$\begin{cases} \mathbf{b}^T(t+1) = \mathbf{H}^\dagger(t+1)\mathbf{x}(t+1), \\ \mathbf{B}^T(t+1) = [\mathbf{B}^T(t), \mathbf{b}^T(t+1)]. \end{cases}$
```

the price paid in terms of estimation performance is small when the model changes slowly, as expected.

Table I summarizes the steps proposed in this section. Towards fleshing out complete adaptive algorithms, the following key issues should be addressed:

i) the observed matrix $\mathbf{X}^{(1)}(t)$ has to be properly windowed so as to weight past observations;
ii) pseudoinverse matrices should be recursively updated;
iii) SVDs should be replaced by SVD tracking algorithms;
iv) operations having a complexity increasing with time should be avoided.

In the next section, we focus on i) and we show how an exponential window or a sliding window can be used.

V. Choice of the Window

A. Exponential Window

Estimation of the PARAFAC components of $\mathcal{X}(t+1)$ can be done by minimization of the following exponential window least squares (EWLS) criterion:

$$\min_{\{\mathbf{H}(t+1), \mathbf{B}(t+1)\}} (\phi^{\mathrm{EW}}(t+1)) \tag{10}$$

with

$$\phi^{\mathrm{EW}}(t+1) = \sum_{\tau=1}^{t+1} \lambda^{t+1-\tau} \left\| \mathbf{x}(\tau) - \mathbf{H}(t+1)\mathbf{b}^T(\tau) \right\|^2 \tag{11}$$

where λ is the forgetting factor. This window implies that at any time, all previous observed slices are involved with a different weight. Let us define the weighted observed matrix by

$$\mathbf{X}_{\mathrm{EW}}(t+1) \overset{\mathrm{def}}{=} \mathbf{X}^{(1)}(t+1)\mathbf{\Lambda}(t+1) \tag{12}$$

where $\mathbf{\Lambda}(t+1) = \mathrm{diag}([\lambda^{t/2}, \lambda^{t-1/2}, \ldots, \lambda^{1/2}, 1])$ is the weighting matrix. The exponential window implies the following update rule for $\mathbf{X}_{\mathrm{EW}}$:

$$\mathbf{X}_{\mathrm{EW}}(t+1) = [\lambda^{1/2}\mathbf{X}_{\mathrm{EW}}(t), \mathbf{x}(t+1)]. \tag{13}$$

B. Truncated Window

For a truncated window of length $N > R$, we denote by $\mathbf{X}_{\mathrm{TW}}^{(1)}(t+1)$ the $IK \times N$ (unweighted) matrix consisting of the last N columns of $\mathbf{X}^{(1)}(t+1)$

$$\mathbf{X}_{\mathrm{TW}}^{(1)}(t+1) = [\mathbf{x}(t+2-N), \ldots, \mathbf{x}(t), \mathbf{x}(t+1)]. \tag{14}$$

Let us define by $\mathcal{X}_{\mathrm{TW}}(t+1) \in \mathbb{C}^{I \times N \times K}$ the tensor built from the N most recent slices, of which $\mathbf{X}_{\mathrm{TW}}^{(1)}(t+1)$ is a matrix representation. Estimation of the PARAFAC decomposition of $\mathcal{X}_{\mathrm{TW}}(t+1)$ can be done by minimization of the following truncated window least squares (TWLS) criterion

$$\min_{\{\mathbf{H}(t+1), \mathbf{B}_{\mathrm{TW}}(t+1)\}} (\phi^{\mathrm{TW}}(t+1)) \tag{15}$$

with

$$\phi^{\mathrm{TW}}(t+1) = \sum_{\tau=1}^{N} \lambda^{N-\tau} \left\| \mathbf{x}(u+\tau) - \mathbf{H}(t+1)\mathbf{b}^T(u+\tau) \right\|^2 \tag{16}$$

where λ is the forgetting factor, $u = t+1-N$ and $\mathbf{B}_{\mathrm{TW}}(t+1)$ consists of the last N rows of $\mathbf{B}(t+1)$. This window implies that at any time, the N most recent slices are involved with a different weight. The case $\lambda = 1$ corresponds to a rectangular sliding window, while the case $\lambda < 1$ is known as the exponential decaying sliding window.

Let us define the weighted observed matrix $\mathbf{X}_{\mathrm{TW}}(t+1)$, built from $\mathbf{X}_{\mathrm{TW}}^{(1)}(t+1)$ as follows:

$$\mathbf{X}_{\mathrm{TW}}(t+1) \overset{\mathrm{def}}{=} \mathbf{X}_{\mathrm{TW}}^{(1)}(t+1)\mathbf{\Lambda} \tag{17}$$

where $\mathbf{\Lambda} = \mathrm{diag}([\lambda^{N-1/2}, \lambda^{N-2/2}, \ldots, \lambda^{1/2}, 1])$. The update rule for $\mathbf{X}_{\mathrm{TW}}$ is

$$[\lambda^{N/2}\mathbf{x}(t+1-N), \mathbf{X}_{\mathrm{TW}}(t+1)] = [\lambda^{1/2}\mathbf{X}_{\mathrm{TW}}(t), \mathbf{x}(t+1)]. \tag{18}$$

Given the windowed observed data, we now adapt the skeleton of Table I to build the PARAFAC-SDT and PARAFAC-RLST algorithms.

VI. PARAFAC-SDT Algorithm

A. Preliminaries

For an exponential window, let us consider the two following factorizations of the weighted matrix $\mathbf{X}_{\mathrm{EW}}(t)$:

$$\begin{cases} \mathbf{X}_{\mathrm{EW}}(t) = (\mathbf{A}(t) \odot \mathbf{C}(t))\mathbf{B}^T(t)\mathbf{\Lambda}(t) \\ \mathbf{X}_{\mathrm{EW}}(t) = \mathbf{U}_{\mathrm{EW}}(t)\Sigma_{\mathrm{EW}}(t)\mathbf{V}_{\mathrm{EW}}^H(t). \end{cases} \tag{19}$$

The first factorization results from substitution of $\mathbf{X}^{(1)}(t)$ in (12) by its PARAFAC decomposition. The second is the economy-size SVD of $\mathbf{X}_{\mathrm{EW}}(t)$, where $\mathbf{U}_{\mathrm{EW}}(t) \in \mathbb{C}^{IK \times R}$, $\mathbf{\Sigma}_{\mathrm{EW}}(t) \in \mathbb{C}^{R \times R}$ and $\mathbf{V}_{\mathrm{EW}}(t) \in \mathbb{C}^{J(t) \times R}$.

Similarly, for a truncated window, we get

$$\begin{cases} \mathbf{X}_{\mathrm{TW}}(t) = (\mathbf{A}(t) \odot \mathbf{C}(t))\mathbf{B}_{\mathrm{TW}}^T(t)\mathbf{\Lambda} \\ \mathbf{X}_{\mathrm{TW}}(t) = \mathbf{U}_{\mathrm{TW}}(t)\Sigma_{\mathrm{TW}}(t)\mathbf{V}_{\mathrm{TW}}^H(t) \end{cases} \tag{20}$$

where $\mathbf{U}_{\text{TW}}(t) \in \mathbb{C}^{IK\times R}$, $\mathbf{\Sigma}_{\text{TW}}(t) \in \mathbb{C}^{R\times R}$ and $\mathbf{V}_{\text{TW}}(t) \in \mathbb{C}^{N\times R}$.

Under the conditions of Theorem 1, $\mathbf{X}_{\text{EW}}(t)$ and $\mathbf{X}_{\text{TW}}(t)$ have rank-R, almost surely [29]. Thus, from (19), there exists a nonsingular matrix $\mathbf{W}_{\text{EW}}(t) \in \mathbb{C}^{R\times R}$ such that

$$\begin{cases} \mathbf{A}(t) \odot \mathbf{C}(t) = \mathbf{E}_{\text{EW}}(t)\mathbf{W}_{\text{EW}}(t) \\ \mathbf{B}^T(t)\mathbf{\Lambda}(t) = \mathbf{W}_{\text{EW}}^{-1}(t)\mathbf{V}_{\text{EW}}^H(t) \end{cases} \tag{21}$$

where $\mathbf{E}_{\text{EW}}(t) \overset{\text{def}}{=} \mathbf{U}_{\text{EW}}(t)\Sigma_{\text{EW}}(t)$. Similarly, from (20), there exists a nonsingular matrix $\mathbf{W}_{\text{TW}}(t) \in \mathbb{C}^{R\times R}$ such that

$$\begin{cases} \mathbf{A}(t) \odot \mathbf{C}(t) = \mathbf{E}_{\text{TW}}(t)\mathbf{W}_{\text{TW}}(t) \\ \mathbf{B}_{\text{TW}}^T(t)\mathbf{\Lambda} = \mathbf{W}_{\text{TW}}^{-1}(t)\mathbf{V}_{\text{TW}}^H(t) \end{cases} \tag{22}$$

where $\mathbf{E}_{\text{TW}}(t) \overset{\text{def}}{=} \mathbf{U}_{\text{TW}}(t)\Sigma_{\text{TW}}(t)$. It was established in [29] that the matrix $\mathbf{W}_{\text{EW}}(t)$ that links both equations in a system of the form (21) can be found by solving a simultaneous-diagonalization problem of a set of R matrices of size $R \times R$. The same remark holds for $\mathbf{W}_{\text{TW}}(t)$ and system (22). However, solving such a problem for each time index should be avoided in real-time applications. Instead, the objective of PARAFAC-SDT is to update $\mathbf{W}_{\text{EW}}$, $\mathbf{W}_{\text{TW}}$ and their respective inverses in a recursive way.

At time $t+1$, (21) becomes

$$\begin{cases} \mathbf{A}(t+1) \odot \mathbf{C}(t+1) = \mathbf{E}_{\text{EW}}(t+1)\mathbf{W}_{\text{EW}}(t+1) \\ \mathbf{B}^T(t+1)\mathbf{\Lambda}(t+1) = \mathbf{W}_{\text{EW}}^{-1}(t+1)\mathbf{V}_{\text{EW}}^H(t+1) \end{cases} \tag{23}$$

and (22) becomes

$$\begin{cases} \mathbf{A}(t+1) \odot \mathbf{C}(t+1) = \mathbf{E}_{\text{TW}}(t+1)\mathbf{W}_{\text{TW}}(t+1) \\ \mathbf{B}_{\text{TW}}^T(t+1)\mathbf{\Lambda} = \mathbf{W}_{\text{TW}}^{-1}(t+1)\mathbf{V}_{\text{TW}}^H(t+1). \end{cases} \tag{24}$$

The main idea of PARAFAC-SDT is to link (21) to (23) and (22) to (24) by capitalizing on the time-shift structure of $\mathbf{B}(t+1)$. Hence, we show that exploitation of the common block between $\mathbf{B}(t)$ and $\mathbf{B}(t+1)$ leads to the recursive updates we are looking for. Before derivation of these update rules, the first step consists of the estimation of $\mathbf{E}_{\text{EW}}(t+1)$ and $\mathbf{V}_{\text{EW}}(t+1)$ from that of $\mathbf{E}_{\text{EW}}(t)$ and $\mathbf{V}_{\text{EW}}(t)$, i.e., we have to track the SVD of the weighted matrix $\mathbf{X}_{\text{EW}}$. The same remark holds for $\mathbf{X}_{\text{TW}}$.

From now on, subscripts EW and TW will be omitted when the same properties hold for both windows.

B. *Steps of PARAFAC-SDT*

1) Step 1: SVD-Tracking: First, we notice that for both windows, left and right subspaces have to be tracked. Also, we need to compute orthonormal right subspace basis vectors because this property on $\mathbf{V}$ will be used in the next step to derive the recursive update of $\mathbf{W}$. In the exponential window case, $\mathbf{X}_{\text{EW}}$ has a growing dimension and so has $\mathbf{V}_{\text{EW}}$. In the truncated window case, $\mathbf{E}_{\text{TW}}$ and $\mathbf{V}_{\text{TW}}$ both have fixed dimensions. For both windows, one solution to update $\mathbf{E}$ and $\mathbf{V}$ is to perform a single step of the classical Bi-Iteration technique [34].

However, with an exponential window, this technique involves complexities growing with time for *three* operations out of *four*, because of the growing dimension of $\mathbf{X}_{\text{EW}}$. Though many adaptive algorithms proposed in the literature deal with the exponential-window case, most of them are designed to track the subspace associated to the fixed dimension only. One solution to handle this problem is to combine Bi-SVD1 algorithm [6] with the time-updating recursion for the growing orthonormal right singular basis matrix given in [6, eq. (11b)]. In this way, only one step involves a complexity growing linearly with time, instead of three.

For a truncated window, complexities are fixed, but the dominant cost of Bi-Iteration is $16RIKN$ (if $R \ll min(IK, N)$). Several algorithms for sliding-window SVD tracking problems with similar robustness and lower complexities have been proposed, see, e.g., [7], [8]. In practice, we will use SWASVD (with a slight modification to include λ) proposed in [7], of which the dominant cost (under $R \ll IK$) is $23(N+IK)R^2$.

2) Step 2: Recursive Updates of $\mathbf{W}$ *and* $\mathbf{W}^{-1}$*:* To derive recursive updates of $\mathbf{W}$ and $\mathbf{W}^{-1}$, we use orthonormality of the right subspace basis vectors in $\mathbf{V}$, obtained from the first step. For both windows, we have

$$\begin{cases} \mathbf{B}^T(t)\mathbf{\Lambda}(t) = \mathbf{W}^{-1}(t)\mathbf{V}^H(t) \\ \mathbf{B}^T(t+1)\mathbf{\Lambda}(t+1) = \mathbf{W}^{-1}(t+1)\mathbf{V}^H(t+1). \end{cases} \tag{25}$$

where the indices TW and EW have been omitted and $\mathbf{\Lambda}(t+1)$ resumes to $\mathbf{\Lambda}$ in the TW case. In this system, previous estimates $\mathbf{B}(t)$, $\mathbf{W}^{-1}(t)$, $\mathbf{V}(t)$ are known and $\mathbf{V}(t+1)$ is known from Step 1. Moreover, due to the time-shift structure, $\mathbf{B}(t)$ and $\mathbf{B}(t+1)$ have a block in common, denoted $\tilde{\mathbf{B}}(t)$.

If an exponential window was used in step 1, $\mathbf{V}_{\text{EW}}(t)$ and $\mathbf{V}_{\text{EW}}(t+1)$ have dimensions $J(t) \times R$ and $J(t+1) \times R$, respectively. So are the dimensions of $\mathbf{B}(t)$ and $\mathbf{B}(t+1)$. In this case the common block $\tilde{\mathbf{B}}(t)$ is $\mathbf{B}(t)$ itself.

If a truncated window of length $N > R$ was used, then $\mathbf{V}_{\text{TW}}$ and $\mathbf{B}_{\text{TW}}$ have dimensions $N \times R$ for every time index. In this case, the common block $\tilde{\mathbf{B}}(t)$ has dimensions $N-1 \times R$ and is given by $\tilde{\mathbf{B}}(t) = [\mathbf{B}_{\text{TW}}(t)]_{2:N,:} = [\mathbf{B}_{\text{TW}}(t+1)]_{1:N-1,:}$.

Let us now define the following matrices, according to the window under consideration.

- (EW):

$$\begin{cases} \tilde{\mathbf{V}}(t) = \mathbf{V}_{\text{EW}}(t) \\ \hat{\mathbf{V}}(t+1) = [\mathbf{V}_{\text{EW}}(t+1)]_{1:J(t),:} \\ \hat{\mathbf{v}}(t+1) = [\mathbf{V}_{\text{EW}}(t+1)]_{J(t+1),:}. \end{cases}$$

- (TW):

$$\begin{cases} \tilde{\mathbf{V}}(t) = [\mathbf{V}_{\text{TW}}(t)]_{2:N,:} \\ \tilde{\mathbf{v}}(t) = [\mathbf{V}_{\text{TW}}(t)]_{1,:} \\ \hat{\mathbf{V}}(t+1) = [\mathbf{V}_{\text{TW}}(t+1)]_{1:N-1,:} \\ \hat{\mathbf{v}}(t+1) = [\mathbf{V}_{\text{TW}}(t+1)]_{N,:}. \end{cases}$$

For both windows, identification of the common block $\tilde{\mathbf{B}}(t)$ in (25) yields

$$\lambda^{1/2}\mathbf{W}^{-1}(t)\tilde{\mathbf{V}}^H(t) = \mathbf{W}^{-1}(t+1)\hat{\mathbf{V}}^H(t+1). \tag{26}$$

It follows that

$$\mathbf{W}^{-1}(t+1) = \lambda^{1/2}\mathbf{W}^{-1}(t)\tilde{\mathbf{V}}^H(t)\left(\hat{\mathbf{V}}^H(t+1)\right)^{\dagger} \tag{27}$$

and

$$\mathbf{W}(t+1) = \lambda^{-1/2}\hat{\mathbf{V}}^H(t+1)\left(\tilde{\mathbf{V}}^H(t)\right)^{\dagger}\mathbf{W}(t). \tag{28}$$

The task is now to avoid explicit computation of the pseudoinverse in (27) and (28). For both windows, $\tilde{\mathbf{V}}(t+1)$ results from deleting the last row $\tilde{\mathbf{v}}(t+1)$ of the orthogonal matrix $\mathbf{V}(t+1)$ produced by the first step. From the matrix inversion Lemma for rank-1 updates, we have

$$\left(\tilde{\mathbf{V}}^H(t+1)\right)^\dagger = \tilde{\mathbf{V}}(t+1)\left(\mathbf{I}_R + \frac{\tilde{\mathbf{v}}^H(t+1)\tilde{\mathbf{v}}(t+1)}{1-\|\tilde{\mathbf{v}}(t+1)\|^2}\right) \quad (29)$$

which is substituted in (27) to get the recursive update of $\mathbf{W}^{-1}$.

Let us now define a recursive update for $\mathbf{W}$. In the exponential window case, $\tilde{\mathbf{V}}(t) = \mathbf{V}_{\mathrm{EW}}(t)$ is a unitary matrix so $\left(\tilde{\mathbf{V}}^H(t)\right)^\dagger = \mathbf{V}_{\mathrm{EW}}(t)$. In the truncated window case, $\tilde{\mathbf{V}}(t)$ results from deleting the first row $\tilde{\mathbf{v}}(t)$ of the orthogonal matrix $\mathbf{V}_{\mathrm{TW}}(t)$. From the same matrix inversion Lemma, we get

$$\left(\tilde{\mathbf{V}}^H(t)\right)^\dagger = \tilde{\mathbf{V}}(t)\left(\mathbf{I}_R + \frac{\tilde{\mathbf{v}}^H(t)\tilde{\mathbf{v}}(t)}{1-\|\tilde{\mathbf{v}}(t)\|^2}\right) \quad (30)$$

which is substituted in (28) to get the recursive update of $\mathbf{W}$.

3) Step 3: Updates of $\mathbf{A}$ *and* $\mathbf{C}$*:* First, we form the matrix $\mathbf{H}(t+1) = \mathbf{E}(t+1)\mathbf{W}(t+1)$, from $\mathbf{E}(t+1) = \mathbf{U}(t+1)\mathbf{\Sigma}(t+1)$ produced by step 1 and from $\mathbf{W}(t+1)$ produced by step 2. $\mathbf{H}(t+1)$ is an estimate of $\mathbf{A}(t+1) \odot \mathbf{C}(t+1)$ and one could possibly extract $\mathbf{A}(t+1)$ and $\mathbf{C}(t+1)$ from $\mathbf{H}(t+1)$ by following the procedure described in step 4 of Table I. Instead, the recursive updates of $\mathbf{c}_r$ and $\mathbf{a}_r$ consist of tracking the first left and right singular vectors of $\mathbf{H}_r(t+1)$. A single Bi-SVD iteration applied to these extreme singular vectors resumes to the following substeps:

Step 3a. $\mathbf{a}_r^*(t+1) = \mathbf{H}_r^H(t+1)\mathbf{c}_r(t)$.

Step 3b. $\mathbf{c}_r(t+1) = \mathbf{H}_r(t+1)\mathbf{a}_r(t+1)/\|\mathbf{H}_r(t+1) \times \mathbf{a}_r(t+1)\|$.

4) Step 4: Update of $\mathbf{B}$*:* The updated new row $\mathbf{b}^T(t+1) = \mathbf{W}^{-1}(t+1)\tilde{\mathbf{v}}^H(t+1)$ is finally appended to $\mathbf{B}(t)$ to build $\mathbf{B}(t+1)$, where $\mathbf{W}^{-1}(t+1)$ is given by (27).

C. Complexity

The PARAFAC-SDT algorithm is summarized in Table II, with the complexity associated to each step. The complexities are expressed in terms of Real FLoating point OPeration (flop) counts. For instance, the scalar product of d dimensional complex vectors involves $8d$ flops. The choice of an exponential window with this algorithm involves a time-dependent complexity because of step 1, since $\mathbf{V}(t)$ has an increasing horizontal dimensions. In this sense, it is preferable to combine PARAFAC-SDT with a truncated window. In the latter case, the global complexity is $16R^3+R^2(31IK+31N+40)+R(32IK+10K+20)$. On the same basis, the complexity of a *single* iteration of the batch ALS algorithm, applied on the tensor built from the last N slices, is $88R^3+R^2(64NK+64IK+64IN+24)+R(24INK+8NK+8IK+8IN)$ [35]. From these results, it is clear that PARAFAC-SDT has a much lower complexity than ALS.

VII. PARAFAC-RLST Algorithm

A. Principle of the Algorithm

In this section, we derive a very different tracking algorithm, which we term PARAFAC-RLST (PARAFAC via Recursive Least Squares Tracking). The principle of PARAFAC-RLST algorithm follows the skeleton given in Table I. As a starting point, an initial estimate of $\mathbf{b}(t+1)$ is given by (8), where $\mathbf{H}^\dagger(t)$ is known from the previous tracking step. Then, the recursive updates of $\mathbf{H}(t+1)$ and $\mathbf{H}^\dagger(t+1)$ are derived from the minimization of $\phi^{\mathrm{EW}}(t+1)$ or $\phi^{\mathrm{TW}}(t+1)$ defined in Section V. Finally, $\mathbf{b}(t+1)$ is reestimated by substituting $\mathbf{H}^\dagger(t)$ by $\mathbf{H}^\dagger(t+1)$ in (8) and $\mathbf{B}(t+1)$ is built by appending the new row $\mathbf{b}(t+1)$ to $\mathbf{B}(t)$. From now on, subscripts EW and TW will be omitted when the same properties hold for both windows.

B. Steps of PARAFAC-RLST

1) Update of $\mathbf{H}$*:* Let us derive a recursive update for $\mathbf{H}(t+1)$, given an initial estimate of $\mathbf{b}(t+1)$. Let $\nabla\phi(t+1) \in \mathbb{C}^{IK\times R}$ denote the gradient of $\phi(t+1)$ with respect to $\mathbf{H}(t+1)$:

- in the exponential window case:

$$\nabla\phi^{(EW)}(t+1) = 2\sum_{\tau=1}^{t+1}\lambda^{t+1-\tau}\left(\mathbf{x}(\tau) - \mathbf{H}(t+1)\mathbf{b}^T(\tau)\right)\mathbf{b}^*(\tau) \quad (31)$$

- in the truncated window case:

$$\nabla\phi^{(TW)}(t+1) = 2\sum_{\tau=1}^{N}\lambda^{N-\tau}\big(\mathbf{x}(u+\tau) - \mathbf{H}(t+1)\mathbf{b}^T(u+\tau)\big)\mathbf{b}^*(u+\tau). \quad (32)$$

After solving $\nabla\phi(t+1) = \mathbf{0}$, we get the following:

- in the exponential window case:

$$\mathbf{H}(t+1) = \mathbf{R}_{\mathrm{EW}}(t+1)\mathbf{P}_{\mathrm{EW}}^{-1}(t+1) \quad (33)$$

where

$$\begin{cases}\mathbf{R}_{\mathrm{EW}}(t+1) = \sum_{\tau=1}^{t+1}\lambda^{t+1-\tau}\mathbf{x}(\tau)\mathbf{b}^*(\tau)\\ \mathbf{P}_{\mathrm{EW}}(t+1) = \sum_{\tau=1}^{t+1}\lambda^{t+1-\tau}\mathbf{b}^T(\tau)\mathbf{b}^*(\tau)\end{cases}$$

- in the truncated window case:

$$\mathbf{H}(t+1) = \mathbf{R}_{\mathrm{TW}}(t+1)\mathbf{P}_{\mathrm{TW}}^{-1}(t+1) \quad (34)$$

where

$$\begin{cases}\mathbf{R}_{\mathrm{TW}}(t+1) = \sum_{\tau=1}^{N}\lambda^{N-\tau}\mathbf{x}(u+\tau)\mathbf{b}^*(u+\tau)\\ \mathbf{P}_{\mathrm{TW}}(t+1) = \sum_{\tau=1}^{N}\lambda^{N-\tau}\mathbf{b}^T(u+\tau)\mathbf{b}^*(u+\tau).\end{cases} \quad (35)$$

The recursive updates for $\mathbf{R}_{\mathrm{EW}}$, $\mathbf{P}_{\mathrm{EW}}$, $\mathbf{R}_{\mathrm{TW}}$ and $\mathbf{P}_{\mathrm{TW}}$ immediately follow: see (36)–(37) at the bottom of the next page. The rank-1 update structure of $\mathbf{R}_{\mathrm{EW}}$ and $\mathbf{P}_{\mathrm{EW}}$ is characteristic for an exponential window, where the data matrix has increasing

TABLE II
SUMMARY OF PARAFAC-SDT ALGORITHM

INPUTS at time t
Old estimates: $\mathbf{A}(t)$, $\mathbf{B}(t)$, $\mathbf{C}(t)$, $\mathbf{W}(t)$, $\mathbf{W}^{-1}(t)$, $\mathbf{E}(t)$ and $\mathbf{V}(t)$
Observations: $\mathbf{X}^{(1)}(t)$ and $\mathbf{X}^{(1)}(t+1) = [\mathbf{X}^{(1)}(t), \mathbf{x}(t+1)]$
Windowing: choose λ and apply a TW or an EW on $\mathbf{X}^{(1)}(t+1)$ to build $\mathbf{X}(t+1)$ (=$\mathbf{X}_{TW}(t+1)$ or $\mathbf{X}_{EW}(t+1)$)
UPDATES at time t+1:

STEP 1: Track the SVD of $\mathbf{X}(t+1)$
First option: for (EW) or (TW), do one step of Bi-iteration SVD.
Better options:
(EW): choose Bi-SVD1 [6], with the modification [6, Eq.(11b)] to compute orthogonal right subspace too. Complexity: growing linearly with time.
(TW): choose SWASVD1 [7] with a slight modification to take λ into account. Complexity: $R^2(23IK + 23N) + 16RIK$.
Outputs of step 1:
(EW): $\mathbf{U}(t+1) \in \mathbb{C}^{KI\times R}$, $\mathbf{\Sigma}(t+1) \in \mathbb{C}^{R\times R}$, $\mathbf{V}(t+1) \in \mathbb{C}^{J(t+1)\times R}$
(TW): $\mathbf{U}(t+1) \in \mathbb{C}^{KI\times R}$, $\mathbf{\Sigma}(t+1) \in \mathbb{C}^{R\times R}$, $\mathbf{V}(t+1) \in \mathbb{C}^{N\times R}$

STEP 2: Update $\mathbf{W}$, $\mathbf{W}^{-1}$
Build matrices:
(EW): $\check{\mathbf{V}}(t) = \mathbf{V}(t)$, $\tilde{\mathbf{V}}(t+1) = [\mathbf{V}(t+1)]_{1:J(t),:}$, $\tilde{\mathbf{v}}(t+1) = [\mathbf{V}(t+1)]_{J(t+1),:}$
(TW): $\check{\mathbf{V}}(t) = [\mathbf{V}(t)]_{2:N,:}$, $\check{\mathbf{v}}(t) = [\mathbf{V}(t)]_{1,:}$, $\tilde{\mathbf{V}}(t+1) = [\mathbf{V}(t+1)]_{1:N-1,:}$ and $\tilde{\mathbf{v}}(t+1) = [\mathbf{V}(t+1)]_{N,:}$

Updates	**EW complexity**	**TW complexity**
-Build $\mathbf{Z} = \lambda^{-1/2}\tilde{\mathbf{V}}^H(t+1)\check{\mathbf{V}}(t)$	$8R^2J(t)$	$8R^2(N-1)$
-Update $\mathbf{W}$		
EW: $\mathbf{W}(t+1) = \mathbf{Z}\mathbf{W}(t)$	$8R^3$	
TW: $\mathbf{W}(t+1) = \mathbf{Z}\left(\mathbf{I}_R + \frac{\check{\mathbf{v}}^H(t)\check{\mathbf{v}}(t)}{1-\lVert\check{\mathbf{v}}(t)\rVert^2}\right)\cdot\mathbf{W}(t)$		$8R^3 + 23R^2 + 10R$
-Update $\mathbf{W}^{-1}$		
EW or TW: $\mathbf{W}^{-1}(t+1) = \mathbf{W}^{-1}(t)\mathbf{Z}^H\cdot\left(\mathbf{I}_R + \frac{\tilde{\mathbf{v}}^H(t+1)\tilde{\mathbf{v}}(t+1)}{1-\lVert\tilde{\mathbf{v}}(t+1)\rVert^2}\right)$	$8R^3 + 23R^2 + 10R$	$8R^3 + 23R^2 + 10R$

STEP 3: Update $\mathbf{A}$ and $\mathbf{C}$

Updates	**EW complexity**	**TW complexity**
$\mathbf{H}(t+1) = \mathbf{E}(t+1)\mathbf{W}(t+1)$	$8R^2IK$	$8R^2IK$
For $r = 1\ldots R$ Do	$R\times$	$R\times$
$\mathbf{H}_r(t+1) = \text{unvec}([\mathbf{H}(t+1)]_{:,r})$		
$\mathbf{a}_r^*(t+1) = \mathbf{H}_r^H(t+1)\mathbf{c}_r(t)$	$8KI$	$8KI$
$\mathbf{c}_r(t+1) = \frac{\mathbf{H}_r(t+1)\mathbf{a}_r(t+1)}{\lVert\mathbf{H}_r(t+1)\mathbf{a}_r(t+1)\rVert}$	$8KI + 10K$	$8KI + 10K$

STEP 4: Update $\mathbf{B}$

Updates	**EW complexity**	**TW complexity**
$\mathbf{b}^T(t+1) = \mathbf{W}^{-1}(t+1)\tilde{\mathbf{v}}^H(t+1)$ and set $\mathbf{B}(t+1)^T = [\mathbf{B}(t)^T, \mathbf{b}^T(t+1)]$	$8R^2$	$8R^2$

OUTPUTS
Matrices $\mathbf{A}(t+1)$, $\mathbf{B}(t+1)$ and $\mathbf{C}(t+1)$ now stand for estimates of the PARAFAC decomposition of $\mathcal{X}(t+1)$.

dimensions. The rank-2 update structure of $\mathbf{R}_{\mathrm{TW}}$ and $\mathbf{P}_{\mathrm{TW}}$ is characteristic for a truncated window, where the new data are appended while the oldest are deleted. The matrix $\mathbf{Q}_{\mathrm{EW}} \overset{\mathrm{def}}{=} \mathbf{P}_{\mathrm{EW}}^{-1}$ in (33) can thus be efficiently computed in a recursive way from the matrix inversion lemma:

$$\mathbf{Q}_{\mathrm{EW}}(t+1) = \lambda^{-1}\mathbf{Q}_{\mathrm{EW}}(t) - \frac{\lambda^{-2}\mathbf{Q}_{\mathrm{EW}}(t)\mathbf{b}^T(t+1)\mathbf{b}^*(t+1)\mathbf{Q}_{\mathrm{EW}}(t)}{1+\lambda^{-1}\mathbf{b}^*(t+1)\mathbf{Q}_{\mathrm{EW}}(t)\mathbf{b}^T(t+1)}. \quad (38)$$

Similarly, $\mathbf{Q}_{\mathrm{TW}} \overset{\mathrm{def}}{=} \mathbf{P}_{\mathrm{TW}}^{-1}$ can be computed by applying this Lemma twice, as follows:

$$\mathbf{Q}_{\mathrm{TW}}(t+1) = \tilde{\mathbf{Q}} + \frac{\lambda^N\tilde{\mathbf{Q}}\mathbf{b}^T(u)\mathbf{b}^*(u)\tilde{\mathbf{Q}}}{1-\lambda^N\mathbf{b}^*(u)\tilde{\mathbf{Q}}\mathbf{b}^T(u)} \quad (39)$$

where

$$\tilde{\mathbf{Q}} = \lambda^{-1}\mathbf{Q}_{\mathrm{TW}}(t) - \frac{\lambda^{-2}\mathbf{Q}_{\mathrm{TW}}(t)\mathbf{b}^T(t+1)\mathbf{b}^*(t+1)\mathbf{Q}_{\mathrm{TW}}(t)}{1+\lambda^{-1}\mathbf{b}^*(t+1)\mathbf{Q}_{\mathrm{TW}}(t)\mathbf{b}^T(t+1)}. \quad (40)$$

Finally, the update of $\mathbf{H}(t+1)$ in (33) or (34) follows from the recursively updated matrices $\mathbf{R}_{\mathrm{EW}}(t+1)$, $\mathbf{P}_{\mathrm{EW}}^{-1}(t+1)$ or $\mathbf{R}_{\mathrm{TW}}(t+1)$, $\mathbf{P}_{\mathrm{TW}}^{-1}(t+1)$.

2) Update of $\mathbf{H}^\dagger$*:* Suppose now that, given the previous estimate of $\mathbf{H}(t+1)$, we want to estimate the last row $\mathbf{b}(t+1)$ of

$$\begin{cases}\mathbf{R}_{\mathrm{EW}}(t+1) = \lambda\mathbf{R}_{\mathrm{EW}}(t) + \mathbf{x}(t+1)\mathbf{b}^*(t+1)\\ \mathbf{P}_{\mathrm{EW}}(t+1) = \lambda\mathbf{P}_{\mathrm{EW}}(t) + \mathbf{b}^T(t+1)\mathbf{b}^*(t+1),\end{cases} \quad (36)$$

$$\begin{cases}\mathbf{R}_{\mathrm{TW}}(t+1) = \lambda\mathbf{R}_{\mathrm{TW}}(t) + \mathbf{x}(t+1)\mathbf{b}^*(t+1) - \lambda^N\mathbf{x}(u)\mathbf{b}^*(u)\\ \mathbf{P}_{\mathrm{TW}}(t+1) = \lambda\mathbf{P}_{\mathrm{TW}}(t) + \mathbf{b}^T(t+1)\mathbf{b}^*(t+1) - \lambda^N\mathbf{b}^T(u)\mathbf{b}^*(u).\end{cases} \quad (37)$$

TABLE III
SUMMARY OF PARAFAC-RLST ALGORITHM

INPUTS at time t
Old estimates: $\mathbf{A}(t)$, $\mathbf{B}(t)$, $\mathbf{C}(t)$, $\mathbf{P}(t)$, $\mathbf{R}(t)$, $\mathbf{P}^{-1}(t)$ and $\mathbf{R}^{\dagger}(t)$.
Observations: $\mathbf{X}^{(1)}(t)$ and $\mathbf{X}^{(1)}(t+1) = [\mathbf{X}^{(1)}(t), \mathbf{x}(t+1)]$
Choose a window (TW or EW) and λ
UPDATES at time t+1:

Updates	complexity
STEP 1: Initial estimate for $\mathbf{b}^T(t+1)$	
EW: $\mathbf{b}^T(t+1) = \mathbf{P}_{EW}(t)\mathbf{R}^{\dagger}_{EW}(t)\mathbf{x}(t+1)$	$8RKI(R+1)$
TW: $\mathbf{b}^T(t+1) = \mathbf{P}_{TW}(t)\mathbf{R}^{\dagger}_{TW}(t)\mathbf{x}(t+1)$	$8RKI(R+1)$
STEP 2: Update $\mathbf{R}$, $\mathbf{P}$, $\mathbf{R}^{\dagger}$ and $\mathbf{P}^{-1}$	
————STEP 2a: Update $\mathbf{R}_{EW}$ or $\mathbf{R}_{TW}$ ————	
EW: $\mathbf{R}_{EW}(t+1) = \lambda\mathbf{R}_{EW}(t+1) + \mathbf{x}(t+1)\mathbf{b}^*(t+1)$	$8RKI$
TW: $\mathbf{R}_{TW}(t+1) = \lambda\mathbf{R}_{TW}(t+1) + \mathbf{x}(t+1)\mathbf{b}^*(t+1) - \lambda^N\mathbf{x}(u)\mathbf{b}^*(u)$	$16RKI$
————STEP 2b: Update $\mathbf{P}_{EW}$ or $\mathbf{P}_{TW}$ ————	
EW: $\mathbf{P}_{EW}(t+1) = \lambda\mathbf{P}_{EW}(t) + \mathbf{b}^T(t+1)\mathbf{b}^*(t+1)$	$8R^2$
TW: $\mathbf{P}_{TW}(t+1) = \lambda\mathbf{P}_{TW}(t) + \mathbf{b}^T(t+1)\mathbf{b}^*(t+1) - \lambda^N\mathbf{b}^T(u)\mathbf{b}^*(u)$	$16R^2$
————STEP 2c: Update $\mathbf{R}^{\dagger}_{EW}$ or $\mathbf{R}^{\dagger}_{TW}$ ————	
EW: Apply the Matrix Pseudo-Inversion Lemma once to compute $\mathbf{R}^{\dagger}_{EW}(t+1) = (\lambda\mathbf{R}_{EW}(t) + \mathbf{x}(t+1)\mathbf{b}^*(t+1))^{\dagger}$	$48IKR$
TW: Apply this Lemma twice to compute $\mathbf{R}^{\dagger}_{TW}(t+1) = \left(\lambda\mathbf{R}_{TW}(t) + \mathbf{x}(t+1)\mathbf{b}^*(t+1) - \lambda^N\mathbf{x}(u)\mathbf{b}^*(u)\right)^{\dagger}$	$96IKR$
————STEP 2d: Update $\mathbf{P}^{-1}_{EW}$ or $\mathbf{P}^{-1}_{TW}$ ————	
EW: Apply Matrix Inversion Lemma as in (38) to compute $\mathbf{P}^{-1}_{EW}(t+1) = \left(\lambda\mathbf{P}_{EW}(t) + \mathbf{b}^T(t+1)\mathbf{b}^*(t+1)\right)^{-1}$	$24R^2 + 10R$
TW: Apply this Lemma twice as in (39) and (40) to compute $\mathbf{P}_{TW}(t+1) = \left(\lambda\mathbf{P}_{TW}(t) + \mathbf{b}^T(t+1)\mathbf{b}^*(t+1) - \lambda^N\mathbf{b}^T(u)\mathbf{b}^*(u)\right)^{-1}$	$48R^2 + 20R$
STEP 3: Update $\mathbf{H}$, $\mathbf{A}$ and $\mathbf{C}$	
EW: $\mathbf{H}(t+1) = \mathbf{R}_{EW}(t+1)\mathbf{P}^{-1}_{EW}(t+1)$	$8R^2IK$
TW: $\mathbf{H}(t+1) = \mathbf{R}_{TW}(t+1)\mathbf{P}^{-1}_{TW}(t+1)$	$8R^2IK$
For $r = 1\ldots R$ Do	$R\times$
$\mathbf{H}_r(t+1) = \text{unvec}([\mathbf{H}(t+1)]_{:,r})$	
$\mathbf{a}^*_r(t+1) = \mathbf{H}^H_r(t+1)\mathbf{c}_r(t)$	$8KI$
$\mathbf{c}_r(t+1) = \frac{\mathbf{H}_r(t+1)\mathbf{a}_r(t+1)}{\lVert\mathbf{H}_r(t+1)\mathbf{a}_r(t+1)\rVert}$	$8KI + 10K$
STEP 4: Update $\mathbf{B}$	
EW: $\mathbf{b}^T(t+1) = \mathbf{P}_{EW}(t+1)\mathbf{R}^{\dagger}_{EW}(t+1)\mathbf{x}(t+1)$	$8R^2 + 8RIK$
TW: $\mathbf{b}^T(t+1) = \mathbf{P}_{TW}(t+1)\mathbf{R}^{\dagger}_{TW}(t+1)\mathbf{x}(t+1)$	$8R^2 + 8RIK$
Then $\mathbf{B}^T(t+1) = [\mathbf{B}^T(t), \mathbf{b}^T(t+1)]$	

OUTPUTS
Matrices $\mathbf{A}(t+1)$, $\mathbf{B}(t+1)$ and $\mathbf{C}(t+1)$ now stand for estimates of the PARAFAC decomposition of $\mathcal{X}(t+1)$.

$\mathbf{B}(t+1)$. Since $\mathbf{H}^{\dagger}(t+1) = \mathbf{P}(t+1)\mathbf{R}^{\dagger}(t+1)$, the least-squares estimate of $\mathbf{b}(t+1)$ is given by

$$\mathbf{b}^T(t+1) = \mathbf{P}_{\text{EW}}(t+1)\mathbf{R}^{\dagger}_{\text{EW}}(t+1)\mathbf{x}(t+1) \tag{41}$$

in the exponential window case and by

$$\mathbf{b}^T(t+1) = \mathbf{P}_{\text{TW}}(t+1)\mathbf{R}^{\dagger}_{\text{TW}}(t+1)\mathbf{x}(t+1) \tag{42}$$

in the truncated window case. The task is now to find a recursive update for $\mathbf{R}^{\dagger}_{\text{EW}}$ and $\mathbf{R}^{\dagger}_{\text{TW}}$

$$\mathbf{R}^{\dagger}_{\text{EW}}(t+1) = (\lambda\mathbf{R}_{\text{EW}}(t) + \mathbf{x}(t+1)\mathbf{b}^*(t+1))^{\dagger}, \tag{43}$$

$$\mathbf{R}^{\dagger}_{\text{TW}}(t+1) = \big(\lambda\mathbf{R}_{\text{TW}}(t) + \mathbf{x}(t+1)\mathbf{b}^*(t+1) - \lambda^N\mathbf{x}(u)\mathbf{b}^*(u)\big)^{\dagger}. \tag{44}$$

Since $\mathbf{R}^{\dagger}_{\text{EW}}(t+1)$ has a rank-1 update structure, it can efficiently be computed by applying the matrix pseudoinversion Lemma given in the Appendix . Given the rank-2 update structure of $\mathbf{R}^{\dagger}_{\text{TW}}(t+1)$, it can be efficiently computed by applying this Lemma twice.

3) Updates of $\mathbf{A}$, $\mathbf{B}$, *and* $\mathbf{C}$*:* These unknown matrices are finally updated in a way similar to PARAFAC-SDT.

C. Complexity

The PARAFAC-RLST algorithm is summarized in Table III, with the complexity associated to each step. With a truncated window, the total complexity of this algorithm is $R^2(16IK + 72) + R(144IK + 10K + 20)$, and with an exponential window, it is $R^2(16KI + 40) + R(88KI + 10K + 10)$. For both windows, PARAFAC-RLST has a much lower complexity than ALS (see Section VI-C).

VIII. INITIALIZATION

In this section, we discuss how PARAFAC-SDT and PARAFAC-RLST can be initialized before entering the tracking mode and we propose two options. First, one can collect the first J_{init} slices to build the initial observed tensor and then run a batch algorithm to fit a PARAFAC model on this tensor. The delay introduced by this initialization step depends on the complexity and convergence speed of the batch algorithm used.

Another option is to build an EVD-based initialization, which is commonly used to initialize batch PARAFAC algorithms, by exploiting an ESPRIT-like idea [33]. Let us consider the tensor $\mathcal{X}$ of size $I \times 2 \times K$ built from the first two observed slices. The task is to find initial estimates of $\mathbf{A} \in \mathbb{C}^{I\times R}$, $\mathbf{C} \in \mathbb{C}^{K\times R}$ and $\mathbf{B} \in \mathbb{C}^{2\times R}$ that fit the PARAFAC model of $\mathcal{X}$. The first $I \times K$ slice can be written as $\mathbf{X}(1) = \mathbf{A}\mathbf{D}_1\mathbf{C}^T$, where $\mathbf{D}_1 = \mathrm{diag}(\mathbf{b}_1)$ and $\mathbf{b}_1$ is the first row of $\mathbf{B}$. Similarly, $\mathbf{X}(2) = \mathbf{A}\mathbf{D}_2\mathbf{C}^T$. Assume at this point that $\mathbf{A}$ and $\mathbf{C}$ are full column rank, $\mathbf{D}_1$ and $\mathbf{D}_2$ are not singular and that all diagonal elements of $\mathbf{D}_1\mathbf{D}_2^{-1}$ are distinct. Under these conditions, it was shown in [36] that the PARAFAC decomposition of $\mathcal{X}$ is unique. If $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ fit the model exactly, they can be found in a simple and noniterative way. By combining the two slices, we get $\mathbf{X}(1)\mathbf{X}^{\dagger}(2) = \mathbf{A}\mathbf{D}_1\mathbf{D}_2^{-1}\mathbf{A}^{\dagger}$, so $\mathbf{A}$ can be estimated as the R principal eigenvectors of $\mathbf{X}(1)\mathbf{X}^{\dagger}(2)$. Similarly, $\mathbf{C}$ can be found from the EVD of $\mathbf{X}^{\dagger}(1)\mathbf{X}(2)$. Given $\mathbf{A}$ and $\mathbf{C}$, $\mathbf{B}$ is finally estimated in the least squares sense. In presence of noise, though the so-obtained matrices may not fit the model optimally, they are generally good starting points, obtained at a significantly lower complexity than batch initialization.

IX. Numerical Experiments

A. Application: DOA and DOD Tracking in MIMO Radars

The concept of MIMO radar has recently received considerable attention [37], [38]. A MIMO radar utilizes multiple antennas at both the transmitter and receiver, but unlike conventional phased-array radars, it can transmit linearly independent waveforms. In this section, we illustrate the performance of PARAFAC-SDT and PARAFAC-RLST in the context of DOA and DOD tracking of multiple targets present in the same range-bin for a bistatic MIMO radar system where the transmit and receive arrays have colocated antennas. The system under consideration is parameterized as follows:

- a transmit array of I colocated antennas;
- a receive array of K colocated antennas;
- R targets in a particular range-bin of interest;
- $\mathbf{S} \in \mathbb{C}^{I\times M}$ holds I mutually orthogonal transmitted pulse waveforms, M being the number of samples per pulse period;
- $\{\alpha_r\}_{r=1}^R$, $\{\gamma_r\}_{r=1}^R$ are the locations of the targets with respect to transmit and receive arrays, respectively;
- $\mathbf{A}(\alpha) = [\mathbf{a}(\alpha_1), \dots, \mathbf{a}(\alpha_R)]$ is the $I \times R$ transmit steering matrix and $\mathbf{C}(\gamma) = [\mathbf{c}(\gamma_1), \dots, \mathbf{c}(\gamma_R)]$ the $K \times R$ receive steering matrix;
- J transmitted pulses;
- β_{rj} is the reflection coefficient of the $r\text{th}$ target during the $j\text{th}$ pulse.

If the steering matrices $\mathbf{C}(\gamma)$ and $\mathbf{A}(\alpha)$ are assumed constant over the duration of the J pulses, while the target reflection coefficients are varying independently from pulse to pulse (Swerling II model), the observed data model obtained after matched-filtering the received data by the orthogonal waveforms in $\mathbf{S}$ can be written as [39]

$$\mathbf{Y} = (\mathbf{A}(\alpha) \odot \mathbf{C}(\gamma))\mathbf{B}^T + \mathbf{Z} \qquad (45)$$

where $\mathbf{Y} \in \mathbb{C}^{IK\times J}$ collects the observations across the J pulses, $\mathbf{B}^T = [\mathbf{b}_1^T, \dots, \mathbf{b}_J^T]$ with $\mathbf{b}_j = [\beta_{1j}, \dots, \beta_{Rj}]$ and $\mathbf{Z}$ contains the interference and noise term.

The model (45) was established in [39] and can be considered as the generalization of the single-pulse multitarget model [40] to the multipulse Swerling II multitarget model. However, the link between (45) and PARAFAC was not recognized in [39], where the method proposed for the localization of multiple targets is a two-dimensional radar imaging method, that consists of looking for the peaks of a Capon beamformer output. In [41], we have linked (45) to PARAFAC and we have shown that batch PARAFAC algorithms allow more accurate localization of the targets than the Capon estimator of [39], at a lower complexity. As aforementioned, (45) considers that $\mathbf{A}(\alpha)$ and $\mathbf{C}(\gamma)$ are fixed during J pulse periods and that only the reflection coefficients vary from pulse to pulse.

If the localization of the targets with respect to both arrays is now supposed to vary from pulse to pulse, the PARAFAC decomposition (45) has to be tracked. For each new pulse, a new row is appended to $\mathbf{B}$, a new column is appended to the observed matrix $\mathbf{Y}$ and all entries of $\mathbf{A}(\alpha)$ and $\mathbf{C}(\gamma)$ may change. This corresponds precisely to the problem considered in this paper.

B. Simulation Results

The ith transmitted waveform, i.e., the $i\text{th}$ row of $\mathbf{S}$, is generated by $[\mathbf{S}]_{i,:} = 1 + \sqrt{-1}/\sqrt{2}[\mathbf{H}_M]_{i,:}$, where $\mathbf{H}_M$ is the $M \times M$ Hadamard matrix, and M is fixed to 256. We consider uniform linear array (ULA) transmit and receive arrays with half-wavelength inter-element spacing for both arrays. The carrier frequency is fixed to 1 GHz. Following the Swerling II target model, we assume that the reflection coefficient of the rth target obeys the complex Gaussian distribution with zero mean and unknown variance $\sigma_{\beta_r}^2$. The entries of $\mathbf{Z}$ obey the complex Gaussian distribution with zero mean and unknown variance. The signal-to-noise ratio (SNR) for a given pulse j is defined by

$$\mathrm{SNR} = 10\log_{10}\frac{\|(\mathbf{A}(\alpha^{(j)}) \odot \mathbf{C}(\gamma^{(j)}))\mathbf{b}_j^T\|^2}{\|[\mathbf{Z}]_{:,j}\|^2} \quad [\mathrm{dB}]$$

where $\{\alpha_r^{(j)}\}_{r=1}^R$, $\{\gamma_r^{(j)}\}_{r=1}^R$ are the locations of the targets with respect to transmit and receive arrays, during the $j\text{th}$ pulse.

We have conducted numerical experiments with $I = K = 6$ antennas and $R = 5$ targets. The set $\{\sigma_{\beta_r}^2\}_{r=1}^5$ holds linearly spaced values from 0.3 to 0.5. The SNR is fixed to 8 dB for each pulse. The targets have an elliptic trajectory in the plane (α, γ), see Fig. 2. The performance is assessed in the two cases $J = 100$ pulses and $J = 500$ pulses. Each target follows the same trajectory in both cases, with the same initial and final positions. Thus, the tracking problem is more difficult in the first case since the positions of the targets vary from a larger amount between two consecutive pulses.

We compare the performance of PARAFAC-RLST-TW, PARAFAC-RLST-EW and PARAFAC-SDT-TW to the batch ALS algorithm [23]. Since PARAFAC-SDT-EW has a complexity growing with time, it is not included in the loop. The forgetting factor is $\lambda = 0.7$ and the length of the truncated window is $N = 10$. To initialize the tracking procedure,

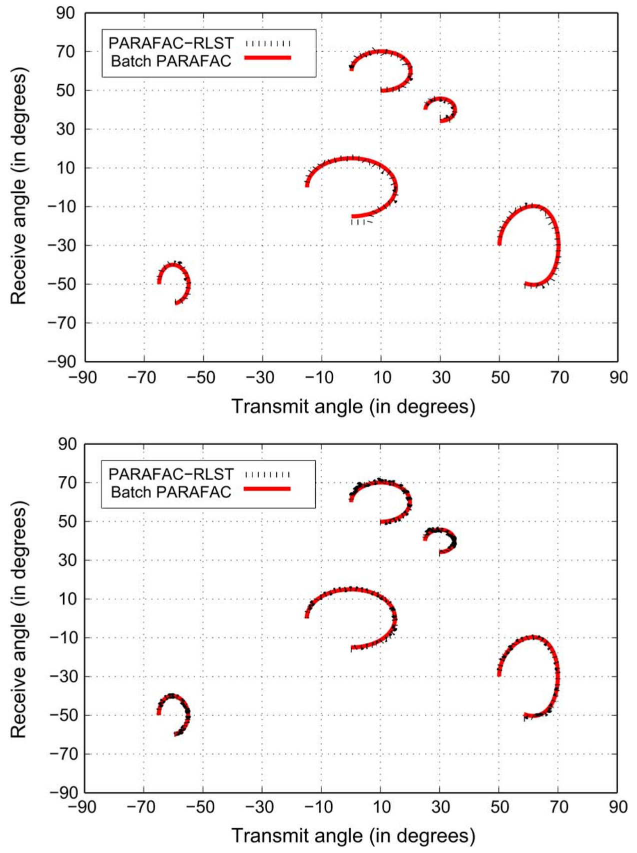


Fig. 2. Estimated trajectories of the 5 targets. $SNR = 8$ dB. $I = K = 6$ antennas. Left: $J = 100$ pulses. Right: $J = 500$ pulses.

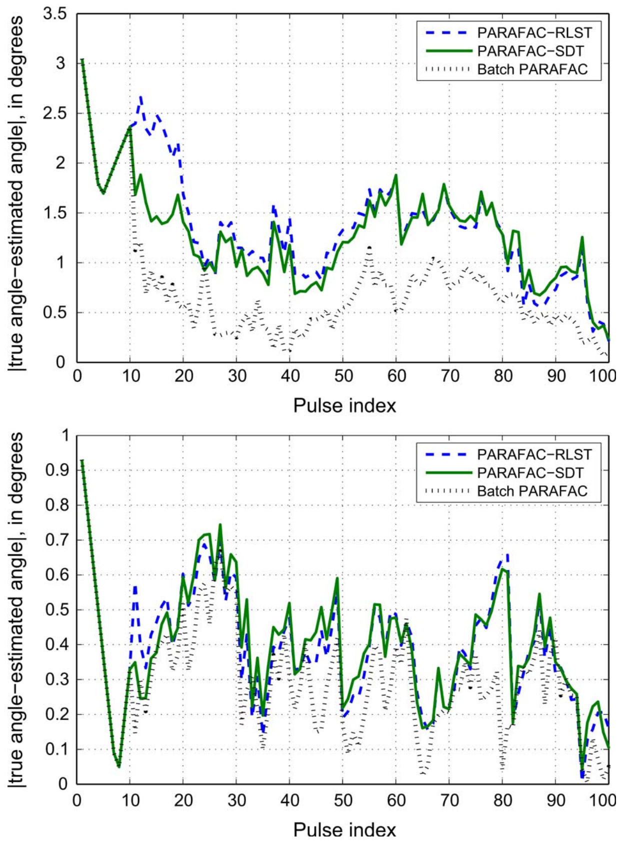


Fig. 3. Evolution of error on angle of departure, averaged over the 5 targets. $SNR = 8$ dB. $I = K = 6$ antennas. Left: $J = 100$ pulses. Right: $J = 500$ pulses (zoom on the first 100 pulses).

the EVD-based technique described in Section VIII is used. The batch ALS algorithm is used repeatedly to compute the PARAFAC decomposition of the tensor formed from the N most recent observed slices. For the comparison between ALS and adaptive algorithms to be fair, ALS is initialized with the loading matrices estimated from the decomposition of the previous tensor, which are supposedly close to the actual solution. Consequently, a few iterations of ALS (typically 20) are needed to converge.

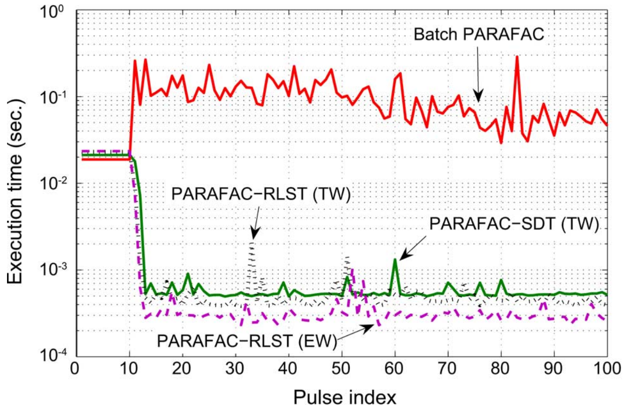
In Fig. 2, we plot the trajectories estimated by PARAFAC-RLST-TW and ALS. It turns out that both algorithms have very close performance, especially with $J = 500$ pulses. For the sake of clarity, the corresponding trajectories of PARAFAC-RLST-EW and PARAFAC-SDT-TW have not been plotted, since they are very close to those of PARAFAC-RLST-TW.

In order to highlight differences in tracking accuracy between the proposed adaptive algorithms and their batch counterpart, we plot in Fig. 3 the evolution of the absolute value of the difference between true and estimated angles of departure, averaged over all targets. As a first observation, it is clear that batch and adaptive algorithms all perform better in the case $J = 500$ than $J = 100$, since the error is always less than 1° in the first case. As a second observation, the performance gap between batch and adaptive algorithms reduces as J increases. This is expected since a high value of J corresponds to slowly varying target positions, which makes tracking easier.

The previously derived per-iteration flop counts for the various algorithms are summarized in Table IV. Unlike the adaptive algorithms, batch ALS is iterated until convergence, and the number of iterations depends on the specific problem instance and the quality of initialization. In the experiments detailed in this section, PARAFAC-ALS typically converged in 20 iterations. In Fig. 4, we compare the execution time of batch and adaptive algorithms. The execution time for the first 10 slices corresponds to the initialization. It is clear that the proposed adaptive algorithms have a very low complexity compared to their batch counterpart—the gap in terms of execution time is between two and three decades. This observation is in accordance with Table IV, which indicates that, in theory, 20 iterations of PARAFAC-ALS yield a complexity that is about 200 times higher than the one of the adaptive algorithms, for the dimensions chosen in this experiment. While differences in execution times depend on implementation, comparing algorithms relying solely on matrix algebra tools implemented in Matlab gives a good idea about complexity, especially in our case where the execution times are orders-of-magnitude apart. Finally, since the complexity of PARAFAC-RLST-EW is lower than that of PARAFAC-SDT-TW and PARAFAC-RLST-TW, with a similar

TABLE IV
COMPARISON OF THE COMPLEXITIES

Algorithm	**Complexity** (flops)	$I = 6, K = 6, N = 10, R = 5$
PARAFAC-ALS	$88R^3 + R^2(64NK + 64IK + 64IN) + R(24INK + 8NK + 8IK + 8IN)$	310640 per iteration
PARAFAC-SDT,TW	$16R^3 + R^2(31IK + 31N + 40) + R(32IK + 10K + 20)$	44810
PARAFAC-RLST,TW	$R^2(16IK + 72) + R(144IK + 10K + 20)$	42520
PARAFAC-RLST,EW	$R^2(16IK + 40) + R(88KI + 10K + 10)$	31590



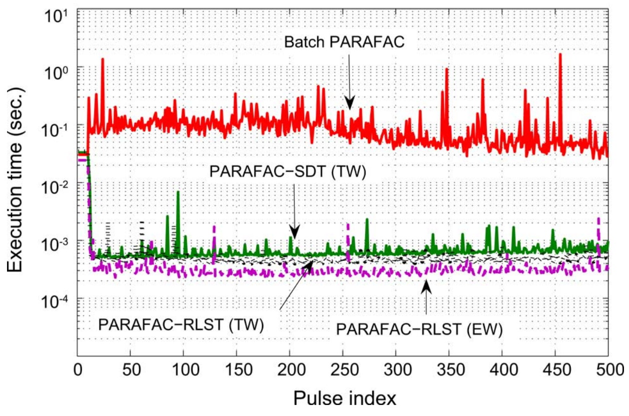


Fig. 4. Evolution of execution time. $SNR = 8$ dB. $I = K = 6$ antennas. Left: $J = 100$ pulses. Right: $J = 500$ pulses.

accuracy (at least in the MIMO radar application considered in this section), the use of this algorithm is to be preferred.

X. Conclusion

In this paper, we have proposed two adaptive algorithms to track the PARAFAC decomposition of a third-order tensor. To our knowledge, this problem has not been addressed in the literature before. The first algorithm, which we termed PARAFAC-SDT, is the tracking version of the batch algorithm based on simultaneous diagonalization proposed in [29]. The second algorithm, which we termed PARAFAC-RLST, is based on the minimization of a weighted least squares criterion. The use of PARAFAC-SDT algorithm is preferable with a truncated window, since its cost with an exponential window increases with time. The PARAFAC-RLST algorithm can be used with both windows. Through the application of multiple target tracking in a MIMO radar system, we have illustrated the excellent tracking capability of these algorithms, which offer performance very close to the well-known batch ALS algorithm, at a much lower complexity. Finally, PARAFAC-SDT and PARAFAC-RLST can be readily employed in a variety of established PARAFAC applications, such as those considered in [23], [24], [27], to deal with a time-varying wireless communication or acoustic propagation channel, or in [42], [43] for tracking the epileptic seizure localization. The derivation of adaptive algorithms to track the PARAFAC decomposition of tensors of any order for which only one mode is growing can be done in the same way as for the three-way case. The only difference in the N-way case is that the matrix $\mathbf{H}(t)$ in (4) would be the Khatri-Rao product of $N-1$ matrices. The generalization to the case where two or more dimensions are growing is left as future work.

Appendix
Pseudo-Inversion Lemma for Rank-1 Update

Let $\mathbf{A} \in \mathbb{C}^{M\times N}$, $M \geq N$, be a full column rank matrix. Consider the vectors $\mathbf{c} \in \mathbb{C}^{M\times 1}$ and $\mathbf{d} \in \mathbb{C}^{N\times 1}$. Assume that $\mathbf{A} + \mathbf{cd}^H$ is full column rank, then

$$(\mathbf{A} + \mathbf{cd}^H)^\dagger = \mathbf{A}^\dagger + \frac{1}{\beta^*}\mathbf{A}^\dagger\mathbf{h}^H\mathbf{u}^H - \frac{\beta^*}{\sigma}\mathbf{pq}^H \tag{46}$$

where $\beta = 1 + \mathbf{d}^H\mathbf{A}^\dagger\mathbf{c}, \mathbf{h} = \mathbf{d}^H\mathbf{A}^\dagger, \mathbf{u} = (\mathbf{I}_M - \mathbf{AA}^\dagger)\mathbf{c}, \mathbf{k} = \mathbf{A}^\dagger\mathbf{c}, \mathbf{p} = -(\|\mathbf{u}\|^2/\beta^*\mathbf{A}^\dagger\mathbf{h}^H + \mathbf{k}), \mathbf{q}^H = -(\|\mathbf{h}\|^2/\beta^*\mathbf{u}^H + \mathbf{h})$, and $\sigma = \|\mathbf{h}\|^2\|\mathbf{u}\|^2 + |\beta|^2$. Note that in the case $M = N$, then $\mathbf{u} = \mathbf{0}$, such that (46) then reduces to $(\mathbf{A} + \mathbf{cd}^H)^\dagger = \mathbf{A}^\dagger - 1/\beta\mathbf{kh}$, which is the inversion lemma.

References

[1] R. Roy and T. Kailath, "ESPRIT-estimation of signal parameters via rotational invariance techniques," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, pp. 984–995, 1989.
[2] R. O. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Propag.*, vol. 34, pp. 276–280, 1986.
[3] A.-J. van der Veen, "Algebraic methods for deterministic blind beamforming," *Proc. IEEE*, vol. 86, pp. 1987–2008, 1998.
[4] E. Moulines, P. Duhamel, J.-F. Cardoso, and S. Mayrargue, "Subspace methods for the blind identification of multichannel FIR filters," *IEEE Trans. Signal Process.*, vol. 43, pp. 516–525, 1995.
[5] S. Gannot and M. Moonen, "Subspace methods for multimicrophone speech dereverberation," *EURASIP J. Appl. Signal Process.*, no. 11, pp. 1074–1090, 2003.
[6] P. Strobach, "Bi-iteration SVD subspace tracking algorithms," *IEEE Trans. Signal Process.*, vol. 45, pp. 1222–1240, May 1997.
[7] R. Badeau, G. Richard, and B. David, "Sliding window adaptive SVD algorithms," *IEEE Trans. Signal Process.*, vol. 52, pp. 1–10, Jan. 2004.
[8] S. Ouyang and Y. Hua, "Bi-iterative least-square method for subspace tracking," *IEEE Trans. Signal Process.*, vol. 53, pp. 2984–2996, Aug. 2005.
[9] P. Comon and G. H. Golub, "Tracking a few extreme singular values and vectors in signal processing," *Proc. IEEE*, vol. 78, pp. 1327–1343, Aug. 1990.
[10] L. De Lathauwer, "Signal processing based on multilinear algebra," Ph. D., Faculty of Eng., K. U. Leuven, Belgium, 1997.
[11] L. R. Tucker, "The extension of factor analysis to three-dimensional matrices," in *Contributions to Mathematical Psychology*, H. Gulliksen and N. Frederiksen, Eds. New York: Holt, Rinehart and Winston, 1964, pp. 109–127.
[12] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, pp. 279–311, 1966.

[13] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000.
[14] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear image analysis for facial recognition," in *Proc. Int. Conf. Pattern Recogn. (ICPR)*, Quebec, Canada, Aug. 2002.
[15] R. Costantini, L. Sbaiz, and S. Süsstrunk, "Higher order SVD analysis for dynamic texture synthesis," *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 42–52, 2008.
[16] R. A. Harshman, "Foundations of the PARAFAC procedure: Model and conditions for an 'explanatory' multi-mode factor analysis," *UCLA Working Papers in Phonetics*, vol. 16, pp. 1–84, 1970.
[17] R. Bro, "PARAFAC: Tutorial and applications," *Chemom. Intell. Lab. Syst.*, vol. 38, pp. 149–171, 1997.
[18] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *J. Math. Phys.*, vol. 6, no. 1, pp. 164–189, 1927.
[19] F. L. Hitchcock, "Multiple invariants and generalized rank of a p-way matrix or tensor," *J. Math. Phys.*, vol. 7, no. 1, pp. 39–79, 1927.
[20] R. Cattell, "Parallel proportional profiles and other principles for determining the choice of factors by rotation," *Psychometrika*, vol. 9, no. 4, pp. 267–283, Dec. 1944.
[21] A. Smilde, R. Bro, and P. Geladi, *Multi-way Analysis. Applications in the Chemical Sciences*. Chichester, U.K.: Wiley, 2004.
[22] P. Kroonenberg, *Applied Multiway Data Analysis*. New York: Wiley Series in Probabil. Statist., 2008.
[23] N. D. Sidiropoulos, G. B. Giannakis, and R. Bro, "Blind PARAFAC receivers for DS-CDMA systems," *IEEE Trans. Signal Process.*, vol. 48, pp. 810–823, 2000.
[24] N. D. Sidiropoulos, R. Bro, and G. B. Giannakis, "Parallel factor analysis in sensor array processing," *IEEE Trans. Signal Process.*, vol. 48, pp. 2377–2388, 2000.
[25] P. Comon, "Blind identification and source separation in 2×3 under-determined mixtures," *IEEE Trans. Signal Process.*, vol. 52, no. 1, pp. 11–22, 2004.
[26] L. De Lathauwer and J. Castaing, "Blind identification of underdetermined mixtures by simultaneous matrix diagonalization," *IEEE Trans. Signal Process.*, vol. 56, no. 3, pp. 1096–1105, 2008.
[27] K. Mokios, N. D. Sidiropoulos, and A. Potamianos, "Blind speech separation using PARAFAC analysis and integer least squares," *Proc. ICASSP '06*, vol. 5, pp. 73–76, 2006.
[28] G. Tomasi and R. Bro, "A comparison of algorithms for fitting the PARAFAC model," *Comp. Stat. Data Anal.*, vol. 50, pp. 1700–1734, 2006.
[29] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization," *SIAM J. Matrix Anal. Appl.*, vol. 28, no. 3, pp. 642–666, 2006.
[30] J. B. Kruskal, "Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear Algebra Appl.*, vol. 18, pp. 95–138, 1977.
[31] A. Stegeman and N. D. Sidiropoulos, "On kruskal's uniqueness condition for the candecomp/PARAFAC decomposition," *Linear Algebra Appl.*, vol. 420, pp. 540–552, 2007.
[32] N. D. Sidiropoulos and R. Bro, "On the uniqueness of multilinear decomposition of N-way arrays," *J. Chemometr.*, vol. 14, pp. 229–239, 2000.
[33] S. Leurgans, R. Ross, and R. Abel, "A decomposition for three-way arrays," *SIAM J. Matrix Anal. Appl.*, vol. 14, no. 4, pp. 1064–1083, 1993.
[34] M. Clint and A. Jennings, "A simultaneous iteration method for the unsymmetric eigenvalue problem," *J. Inst. Math. Appl.*, vol. 8, pp. 111–121, 1971.
[35] M. Rajih, P. Comon, and R. A. Harshman, "Enhanced line search: A novel method to accelerate PARAFAC," *SIAM J. Matrix Anal. Appl., Tensor Decomposit. Appl.*, vol. 30, no. 3, pp. 1148–1171, Sep. 2008.
[36] R. A. Harshman, "Determination and proof of minimum uniqueness conditions for PARAFAC1," *UCLA Working Papers in Phonet.*, vol. 22, pp. 111–117, 1972.
[37] A. Haimovich, R. S. Blum, and L. J. Cimini , Jr, "MIMO radar with widely separated antennas," *IEEE Signal Process. Mag.*, pp. 116–129, Jan. 2008.
[38] J. Li and P. Stoica, "MIMO radar with colocated antennas," *IEEE Signal Process. Mag.*, pp. 106–114, Sep. 2007.
[39] H. Yan, J. Li, and G. Liao, "Multitarget identification and localization using bistatic MIMO radar systems," *EURASIP J. Adv. Signal Process.*, vol. 2008, no. ID 283483, 2008.
[40] L. Xu, J. Li, and P. Stoica, "Radar imaging via adaptive MIMO techniques," in *Proc. 14th Eur. Signal Process. Conf.*, Florence, Italy, Sep. 2006.
[41] D. Nion and N. D. Sidiropoulos, "A PARAFAC-based technique for detection and localization of multiple targets in a MIMO radar system," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2009.
[42] M. De Vos, A. Vergult, L. De Lathauwer, W. De Clercq, S. Van Huffel, P. Dupont, A. Palmini, and W. Van Paesschen, "Canonical decomposition of Ictal EEG reliably detects the seizure onset zone," *NeuroImage*, vol. 37, no. 3, pp. 844–854, Sep. 2007.
[43] E. Acar, C. Bingol, H. Bingol, R. Bro, and B. Yener, "Multiway analysis of epilepsy tensors," *Bioinformatics*, vol. 23, no. 13, pp. i10–i18, 2007.
[44] S. L. Campbell and C. D. Meyer, *Generalized Inverses of Linear Transformations*. New York: Dover, 1991.

Dimitri Nion (S'07–AM'08) was born in Lille, France, on September 6, 1980. He received the electronic engineering degree from ISEN, Lille, in 2003, the M.S. degree from Queen Mary University, London, U.K., in 2003, and the Ph.D. degree in signal processing from the University of Cergy-Pontoise, France, in 2007.

His research interests include linear and multilinear algebra, blind source separation, signal processing for communications, and adaptive signal processing.

Nicholas D. Sidiropoulos (F'09) received the Diploma degree from the Aristotle University of Thessaloniki, Greece, and M.S. and Ph.D. degrees from the University of Maryland at College Park (UMCP), in 1988, 1990, and 1992, respectively, all in electrical engineering.

He has been a Postdoctoral Fellow (1994–1995) and Research Scientist (1996–1997) at the Institute for Systems Research, UMCP, and has held positions as Assistant Professor, Department of Electrical Engineering, University of Virginia, Charlottesville (1997–1999), and Associate Professor, Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis (2000–2002). Since 2002, he has been a Professor with the Department of Electronic and Computer Engineering, Technical University of Crete, Chania-Crete, Greece, and an Adjunct Professor with the University of Minnesota. His current research interests are primarily in signal processing for communications, convex optimization, cross-layer resource allocation for wireless networks, and multiway analysis.

Prof. Sidiropoulos has served as Chair of the Signal Processing for Communications and Networking Technical Committee (SPCOM-TC) of the IEEE Signal Processing (SP) Society (2007–2008; Vice-Chair 2005–2006; Member 2000–2005). He is also a member of the Sensor Array and Multichannel processing Technical Committee (SAM-TC) of the IEEE SP Society (2004–2009). He has served as Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING from 2000 to 2006 and the IEEE SIGNAL PROCESSING LETTERS from 2000 to 2002. He currently serves on the editorial board of *IEEE Signal Processing Magazine*. He received the U.S. NSF/CAREER award in June 1998, and the IEEE Signal Processing Society Best Paper Award twice (in 2001 and 2007). He is a Distinguished Lecturer of the IEEE SP Society for 2008–2009.