Robust Tensor Tracking with Missing Data and Outliers: Novel Adaptive CP Decomposition and Convergence Analysis

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

Abstract—Canonical Polyadic (CP) decomposition is a powerful multilinear algebra tool for analyzing multiway (a.k.a. tensor) data and has been used for various signal processing and machine learning applications. When the underlying tensor is derived from data streams, adaptive CP decomposition is required. In this paper, we propose a novel method called robust adaptive CP decomposition (RACP) for dealing with high-order incomplete streaming tensors that are corrupted by outliers. At each time instant, RACP first performs online outlier rejection to accurately detect and remove sparse outliers, and then performs tensor factor tracking to efficiently update the tensor basis. A unified adaptive algorithm, streaming tensor, missing data, outlier.

Index Terms—CANDECOMP/PARAFAC (CP) decomposition, adaptive algorithm, streaming tensor, missing data, outlier.

I. INTRODUCTION

Nowadays, many modern datasets can be represented by multiway arrays which are referred to as tensors, e.g., a color video surveillance sequence can be represented as a 4th-order tensor of dimensionality, width × height × channel × time. Accordingly, tensor decomposition, which factorizes a tensor into a sequence of basic components, has become a popular analysis tool for processing high-dimensional and multivariate data [1]–[4]. CANDECOMP/PARAFAC (CP) decomposition and Tucker decomposition are well-known and widely-used types of tensor decomposition [1], [5]. Under CP decomposition, a tensor can be expressed as a linear combination of rank-1 tensors, which are formed by an outer product of vectors. As a result, this decomposition offers several nice properties [1], [4]. Among them are the following: (i) CP only requires a linear space complexity w.r.t. the tensor order; (ii) hence, it can avoid the “curse of dimensionality”, a phenomenon whereby the memory storage grows drastically when the dimension increases; (iii) under certain conditions, its expression is essentially unique up to a permutation and scale. The merits of CP decomposition have already been demonstrated in various applications, such as wireless communications [6]–[8], neuroscience [9]–[11], and remote sensing [12]–[14].

In recent years, the demand for adaptive (i.e., online) processing has been increasing due to the fact that many applications generate a huge number of data streams over time [15]–[17]. Such data streams are often with high veracity and high velocity. Veracity requires robust algorithms so as to deal with uncertain, noisy and imperfect data, while velocity requires online real-time processing [15]. These characteristics lead to several critical computational issues: (i) increase in size of the data streams over time, (ii) time-dependent and varying models, and (iii) uncertainty and incompleteness. A robust variant of tensor decomposition for tensors derived from such data streams, namely robust tensor tracking (RTT), has been emerging as a good approach. The main goal of this paper is to propose a scalable and effective method for RTT under the CP model.

A. Related Work

Many methods for CP decomposition have been proposed, and standard algorithms and their applications have been nicely surveyed in [1], [2], [5], [18]. However, most CP algorithms are either sensitive to data imperfection or designed only for batch processing. Online or adaptive algorithms are needed when tensors are derived from data streams. The very first adaptive CP algorithms were developed by Nion and Sidiroopoulos in [19] more than a decade ago. Specifically, the authors proposed to track the low-dimensional subspace of the underlying streaming tensor and then reconstruct the loading factors by exploiting its Khatri-Rao structure. Since then, many adaptive CP algorithms have been proposed for factorizing tensors derived from data streams, such as [20]–[23], to name a few. Vandecappelle et al. in [20] developed a nonlinear least-squares (NLS)-based adaptive CP algorithm for factorizing streaming tensors of order 3. In [21], Zhou et al. introduced the so-called OLCP algorithm which is capable of tracking higher-order streaming tensors. Smith et al. in [22] proposed an adaptive algorithm specifically for streaming sparse tensors, namely CP-stream. In [23], Rambhatla et al. introduced another adaptive CP algorithm called TensorNOODL using online dictionary learning. Nevertheless, none of them are designed for dealing with missing data and outliers.
In the context of missing data, there exist some online robustification methods for batch tensor decompositions. Morteza et al. developed an incremental CP algorithm for RTT with third-order tensors, called TeCPSGD [24]. Thanks to the stochastic gradient descent method, the algorithm can efficiently update the loading factors. Kasai proposed an effective recursive estimator, OLSTEC, to learn low-rank components of the underlying data streams [25]. It yields a better estimation accuracy than TeCPSGD, but its complexity is much higher. Both TeCPSGD and OLSTEC are, however, not capable of tracking higher-order streaming tensors. To overcome this drawback, we recently proposed a new adaptive algorithm that is able to handle higher-order incomplete streaming tensors, called ACP [26], [27]. In spite of their computational merits, the above algorithms are sensitive to outliers.

To deal with outliers, Zhang et al. introduced an online Bayesian-based CP algorithm, namely BRST [28]. To capture sparse components or outliers affecting the tensor, BRST uses a Bayesian statistical model. However, BRST has high computational complexity and hence proves to be inefficient when dealing with fast-arriving and big data streams. Najafi et al. proposed another robust estimator for adaptive CP decomposition, called OR-MSTC [29]. Leveraging the alternating direction method of multipliers (ADMM), OR-MSTC is capable of handling gross corruptions in multi-aspect streaming tensor data. Lee et al. developed the so-called SOFIA method which is specifically designed for dealing with seasonal tensor streams with missing values and sparse corruptions [30]. SOFIA employs the Holt-Winters procedure, a well-known forecasting model for time series capable of dealing with trend and seasonality [31]. Convergence analysis of OR-MSTC and SOFIA is, however, not available.

Some other studies attempted to extend online robust PCA and subspace learning for high-order tensor data. Hu et al. proposed an incremental tensor subspace learning algorithm, called IRTSA, and applied it to robust visual tracking in video streams [32]. Li et al. presented a robust algorithm that can update the tensor dictionary and detect anomalies in an online manner, namely RTSL [33]. Sobral et al. introduced an online stochastic tensor algorithm for learning low-rank structure and sparse components in the tensor data [34]. Another incremental tensor decomposition was designed for video background and foreground separation in [35]. Li et al. developed an adaptive algorithm for robust low-rank tensor learning, called ORLTM [36]. Very recently, Dimitris et al. proposed the first robust online Tucker decomposition that can deal with streaming tensors in the presence of outliers [37]. However, none of the above algorithms are designed for handling missing data. The problem of robust tensor tracking for high-order incomplete streaming tensors remains largely unexplored.

B. Main Contributions

Since there exist several robustification methods for batch tensor decompositions with performance guarantees (see, for examples, [38]–[41]), we designed our algorithm in such a way that it casts such robustness guarantees on RTT. Our method involves the two well-known optimization frameworks: block-coordinate descent (BCD) [42] and majorization-minimization (MM) [43], [44]. To adapt to online learning, the iteration step of MM coincides with the arrival of a new tensor slice over time. Specifically, at each time instant, we decompose RACP into two stages: (i) online outlier rejection and (ii) tensor factor tracking. In the first stage, sparse outliers living in the underlying data streams are first detected by optimizing an \(\ell_1\)-norm regularized loss function. Since the proposed loss function not only promotes sparsity but also remains convex, its convergence is guaranteed. Next, based on the past estimates, the second stage enables us to update the tensor basis by minimizing a majorizing surrogate of the main objective function. Accordingly, an efficient recursive estimator is developed to update the loading factors as well as to track their variation over time.

Our main contributions are summarized as follows. First, we propose a scalable and effective online CP algorithm with ability to (i) estimate low-rank components of streaming tensors derived from imperfect and noisy data streams due to missing observations and outlier corruptions, (ii) adapt the changes of the underlying data streams in dynamic and nonstationary environments, (iii) separate and reject sparse outliers in an online fashion with high accuracy, (iv) be capable of tracking tensor components derived from large data streams, and (v) easily incorporate prior information to deal with specific constraints on the tensor model, e.g., smoothness and nonnegativity.

Secondly, we show that RACP is a provable adaptive CP algorithm with a convergence guarantee. Under mild conditions, we prove that the sequence of solutions generated by RACP converges asymptotically to a stationary point of the empirical loss function. Moreover, the asymptotic variation of the solutions and the almost-sure convergence of the objective function values are also analyzed. To the best of our knowledge, this is a pioneer convergence analysis for RTT algorithms in the presence of missing data and outliers.

Finally, we provide several experiments on both synthetic and real data to illustrate the effectiveness of RACP and its variant in comparison with state-of-the-art algorithms.

Compared to our companion work on tensor tracking in [27], there are several differences between ACP and RACP. First, ACP is not designed for handling sparse corruptions, thus its tracking ability diminishes considerably when observations are corrupted by outliers. By contrast, RACP which is a robust version of ACP is capable of tracking the underlying tensor model as well as detecting sparse corruptions successfully over time.

Technically, the data model and the objective function considered in Section II are different from that in [27] due to the presence of sparse outliers. A \(\ell_1\)-norm regularization term and a truncated sliding window are particularly introduced in this work, which leads to several different technical specifications in optimization methodology and convergence analysis. More concretely, we here derive an ADMM solver to estimate the tensor dictionary coefficients and sparse outliers while ACP adopts a randomized least-squares method for this task.

Next, we propose to use a truncated window of a flexible size \(L_t\) varying from 1 to \(t\), instead of using an exponential
one as in [27]. In the adaptive signal processing literature, it is well known that the exponential window is only useful for stationary and slowly time-varying environments where the underlying low-rank model is either static or changes slowly with time [45]. To enhance the tracking ability of RACP in more complicated scenarios (e.g., fast time-varying or abrupt changes at some points), the use of a truncated window is preferable. Accordingly, a more elaborate recursive rule for updating each tensor factor is designed up to row-wise level which can further support parallel and distributed processing and implementations. It also helps accelerate the tracking process, e.g., we can ignore or skip the update of some factor rows without affecting the others if the corresponding observations are seriously disrupted by strong corruptions.

C. Paper Organization & Notations

The rest of this paper is organized as follows. Section II formulates the RTT problem of interest. Section III presents the proposed RACP algorithm and its convergence analysis is established in Section IV. Section V provides experiments to evaluate the performance of RACP. Section VI concludes the paper. For clarity, the frequently used notations are summarized in Table I.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$, $X$, $\mathcal{X}$</td>
<td>scalar, vector, matrix, and tensor</td>
</tr>
<tr>
<td>$x_{i_1, \ldots, i_N}$ or $[X]_{i_1, \ldots, i_N}$</td>
<td>$(i_1, \ldots, i_N)$-th entry of $X$</td>
</tr>
<tr>
<td>$x = vec(X)$</td>
<td>vectorization of $X$</td>
</tr>
<tr>
<td>$X = diag(x)$</td>
<td>diagonal matrix $X$ with $x$ on the main diagonal</td>
</tr>
<tr>
<td>$x(i, :)$, $x(:, j)$</td>
<td>$i$-th row and $j$-th column of $X$</td>
</tr>
<tr>
<td>$X^T$, $X^{-1}$, $X^#$</td>
<td>transpose, inverse, and pseudo-inverse of $X$</td>
</tr>
<tr>
<td>$X^{(n)}$</td>
<td>mode-$n$ unfolding of $X$</td>
</tr>
<tr>
<td>$U^{(n)}$</td>
<td>$n$-th loading factor/matrix</td>
</tr>
<tr>
<td>$\circ$, $\otimes$, $\oplus$</td>
<td>outer, Khatri-Rao, and Hadamard product</td>
</tr>
<tr>
<td>$\mathcal{X}_{n1}$, $\mathcal{Y}$</td>
<td>concatenation of $\mathcal{X}$ with $\mathcal{Y}$ along the dimension $n$</td>
</tr>
<tr>
<td>$\mathcal{X} \otimes_{n=1}^N U^{(n)}$</td>
<td>$\mathcal{X} \otimes U^{(1)} \otimes \cdots \otimes U^{(N)}$</td>
</tr>
<tr>
<td>$\mathcal{Y}$</td>
<td>Euclidean norm</td>
</tr>
</tbody>
</table>

II. PROBLEM STATEMENT

In this study, we consider an incomplete streaming tensor $\mathcal{X}[t] \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times t}$ whose slices are serially observed with time. At each time $t$, $\mathcal{X}[t]$ is particularly obtained by concatenating a new incoming “slice” $\mathcal{X}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times 1}$ into the previous $\mathcal{X}[t-1]$ along the time dimension, i.e., $\mathcal{X}[t] = \mathcal{X}[t-1] \oplus_{N+1} \mathcal{X}_t$. In particular, we assume to observe the tensor slice $\mathcal{X}_t$ satisfying the following model:

$$\mathcal{P}_t \otimes \mathcal{X}_t = \mathcal{P}_t \otimes (\mathcal{Y}_t + \mathcal{O}_t + \mathcal{N}_t),$$

where $\mathcal{P}_t$ is a binary mask tensor, $\mathcal{Y}_t$ is a low-rank tensor, $\mathcal{O}_t$ is a sparse tensor containing outliers, $\mathcal{N}_t$ is a Gaussian noise tensor, and all these tensors are of the same size with $\mathcal{X}_t$.

Specifically, the observation mask $\mathcal{P}_t$ indicates whether the $(i_1, i_2, \ldots, i_N)$-th entry of $\mathcal{X}_t$ is observed or missing, i.e.,

$$p_{i_1, i_2, \ldots, i_N} = \begin{cases} 0, & \text{if } x_{i_1, i_2, \ldots, i_N} \text{ is missing}, \\ 1, & \text{otherwise}. \end{cases}$$

The low-rank tensor $\mathcal{Y}_t$ is generated according to the following model:

$$\mathcal{Y}_t = \left( \mathcal{I} \bigotimes_{n=1}^N \mathcal{X}_n^{(n)} \right)_{N+1} \mathcal{U}_t^\top,$$

where $\mathcal{I} \in \mathbb{R}^{r \times r \times \cdots \times r}$ is an identity tensor, $\mathcal{U}_t \in \mathbb{R}^{r \times 1}$ is a weight vector and $\{\mathcal{U}_t^{(n)}\}_{n=1}^N$, with $\mathcal{U}_t^{(n)} \in \mathbb{R}^{1 \times N}$, are loading factors. For short, we write $\mathcal{D} = \mathcal{U}_t \otimes \mathcal{U}_{t-1} \otimes \cdots \otimes \mathcal{U}_1$ and denote $\mathcal{D} = \{([\mathcal{U}_t^{(1)}])^\top, ([\mathcal{U}_t^{(2)}])^\top, \ldots, ([\mathcal{U}_t^{(N)}])^\top\}^\top$ the tensor dictionary containing all loading factors.

Next, we define a loss function $\ell(\cdot)$ that not only promotes sparsity but also preserves convexity. For a fixed $\mathcal{D}$ and a tensor slice $\mathcal{X}$ under a binary observation mask $\mathcal{P}$, the loss function w.r.t. $\mathcal{D}$ and $\{\mathcal{P}, \mathcal{X}\}$ is defined as

$$\ell(\mathcal{D}, \mathcal{P}, \mathcal{X}, u) = \min_{\mathcal{O}, \mathcal{U}} \ell(\mathcal{D}, \mathcal{P}, \mathcal{X}, \mathcal{O}, u),$$

with

$$\ell(D, P, X, O, u) = \|O\|_{1} + \frac{\rho}{2} \|P \odot (X - O) - H \times_{N+1} U^t\|_F^2,$$

where $H = \mathcal{I} \bigotimes_{n=1}^N \mathcal{X}_n^{(n)}$. The $\ell_1$-norm is to promote the sparsity on $\mathcal{O}$ and $\rho > 0$ is a regularized parameter.

Now, given a streaming set of incomplete tensor slices $\{\mathcal{P}_k \oplus \mathcal{X}_k\}_{k=1}^K$, robust tensor tracking (RTT) can be stated as the following optimization problem:

$$\mathcal{D}_t = \arg\min_{\mathcal{D}} \sum_{k=L_t}^{t+L_t-1} \lambda^{t-k} \ell(\mathcal{D}, \mathcal{P}_k, \mathcal{X}_k),$$

where $L_t$ is the length of a sliding window and $\lambda$ is a forgetting factor. When $L_t = t$, $\lambda = 1$, the minimization of (4) boils down to its counterpart in batch setting. When $0 < L_t < t$ or $\lambda < 1$, it reduces the impact of past observations, and hence facilitates the tracking ability of RTT estimators in time-varying conditions.

We make some assumptions to support the proposed algorithm in Section III. First, entries of tensor slices $\{\mathcal{X}_t\}_{t=1}^T$ are Frobenius-norm bounded, i.e., $\|\mathcal{X}_t\|_F \leq M_x < \infty \forall t$. This prevents arbitrarily large values in observations and ill-conditioned computation. Next, the tensor rank $r$ is assumed to remain unchanged over time. In addition, tensor factors $\{\mathcal{U}_t^{(n)}\}_{n=1}^N$ are bounded and full column rank, i.e., $\text{rank}(\mathcal{U}_t^{(n)}) = r < I_n$ and $\|\mathcal{U}_t^{(n)}\|_F \leq N_U < \infty \forall n$. Besides, the variation between two consecutive time instants is small, i.e., $0 \leq \sin(\theta(\mathcal{U}_t^{(n)}, \mathcal{U}_{t-1}^{(n)})) \ll 1 \forall n, t$, where $\theta(\mathcal{U}_t^{(n)}, \mathcal{U}_{t-1}^{(n)})$ denotes the canonical angle (the largest principal angle) between two subspaces spanning $\mathcal{U}_t^{(n)}$ and $\mathcal{U}_{t-1}^{(n)}$, respectively. This assumption permits the estimation of the outliers and the coefficient vector from the previous estimation with reasonable accuracy. Under these assumptions, our optimization algorithm is capable of accurately estimating tensor factors, but also successfully tracking their variation over time.

1Since the CP format can be viewed as a special case of the Tucker format, we have the following equivalence:

$$\mathcal{I} \bigotimes_{n=1}^N \mathcal{X}_n^{(n)} \equiv \sum_{i=1}^{N} \mathcal{U}^{(1)}(i) \odot \mathcal{U}^{(2)}(i) \odot \cdots \odot \mathcal{U}^{(N)}(i),$$

where $\mathcal{U}^{(n)}(i)$ is the $i$-th column of $\mathcal{U}^{(n)} \in \mathbb{R}^{I_n \times r}$.

2In batch setting, the weight vector $\mathcal{U}_t$ in (2) is seen as the $t$-th row of the last loading factor $\mathcal{U}^{(N+1)} \in \mathbb{R}^{I_{N+1} \times \cdots \times \cdots \times r}$ of the underlying tensor $\mathcal{X}[t]$. 
III. PROPOSED METHODS

In this section, we first propose the robust adaptive CP (RACP) algorithm for the RTT problem in the presence of missing data and outliers. Then, we introduce two simple extensions of RACP in order to deal with the smoothness condition and nonnegative constraints.

A. Proposed RACP Algorithm

Finding the global optimal solution of (4) is difficult since \( f_t(\cdot) \) is nonconvex. We here adapt it using the majorization-minimization (MM) framework [43], which has been successfully applied to several signal processing problems in general [44] and online learning problems in particular [46]–[49]. In essence, we decompose it into two main stages: (i) online outlier rejection and (ii) tensor factor tracking.

On the arrival of \( \mathbf{X}_t \) at each time \( t \), we first estimate the outlier tensor \( O_t \) and the coefficient vector \( \mathbf{u}_t \) based on the old estimation \( D_{t-1} \). Specifically, we solve the following optimization:

\[
\{O_t, \mathbf{u}_t\} = \underset{O, \mathbf{u}}{\text{argmin}} \tilde{f}(D_{t-1}, P_t, \mathbf{X}_t, O, \mathbf{u}).
\]

(5)

From the past statistics \( \{D_k, \mathbf{P}_k, \mathbf{X}_k, O_k, \mathbf{u}_k\}_{k \geq 1} \), the set of loading factors \( D_k = \{U^{(n)}_k\}_{n=1}^N \) can be updated by minimizing the following majorizing surrogate \( \tilde{f}_t(\cdot) \):

\[
\tilde{f}_t(D) = \frac{1}{L_t} \sum_{k=t-L_t+1}^t \lambda^{-k} \tilde{f}(D, \mathbf{P}_k, \mathbf{X}_k, O_k, \mathbf{u}_k).
\]

(6)

that locally approximates \( f_t(\cdot) \). Note that \( \tilde{f}_t(D) \) is not only first-order surrogate, but also a majorant function of \( f_t(D) \), that is, for all \( t \) and \( D \), we always have \( f_t(D) \leq \tilde{f}_t(D) \) and the error function \( e_t(D) = \tilde{f}_t(D) - f_t(D) \) is Lipschitz continuous. In fact, \( f_t(D) \) and \( \tilde{f}_t(D) \) converge almost-surely to the same limit, and the solution \( D_t \), which minimizes \( \tilde{f}_t(D) \), is exactly that of \( f_t(D) \) when \( t \to \infty \). The results will be later proven in our convergence analysis.

In what follows, we propose two solvers for minimizing (5) and (6) efficiently.

Stage 1: Online Outlier Rejection

To estimate \( O_t \) and \( \mathbf{u}_t \), we recast (5) into the following standard matrix-vector form:

\[
\{o_t, u_t\} = \underset{o, u}{\text{argmin}} \|o\|_1 + \frac{\rho}{2} \|P_t(x_t - o - H_{t-1}u)\|_2^2,
\]

(7)

where \( o_t = \text{vec}(O_t), x_t = \text{vec}(X_t) \), the observation mask matrix \( P_t = \text{diag}(\text{vec}(P_t)) \), and \( H_{t-1} \) is of a Khatri-Rao structure, i.e., \( H_{t-1} = \bigodot_{n=1}^N U^{(n)}_{t-1} \).

Since both terms of (7) are convex, it can be efficiently solved by several methods with convergence guarantees. Here, we use an ADMM solver to minimize (7) due to its simple interpretation and moderate convergence rate [50]. At the \( i \)-th iteration, we particularly read

\[
u^i = (H_{t-1}^\top P_t H_{t-1})^{-1} H_{t-1}^\top P_t (x_t - o^{i-1} - z^{i-1}/\rho), \]

(8)

\[
r^i = \alpha P_t (x_t - H_{t-1}u^i) + (1 - \alpha) o^{i-1} \]

(9)

\[
o^{i-1} = S_\rho(r^i - z^{i-1}/\rho), \]

(10)

\[
z^i = z^{i-1} + \rho(o^i - r^i), \]

(11)

where \( S_\rho(\cdot) \) is the soft-thresholding operator of the \( \ell_1 \)-norm defined as \( S_\rho(x) = \max(0, |x| - \rho) \) and \( \alpha \in [1.5, 1.8] \) is a relaxation parameter. The procedure is stopped when residuals are small, i.e., \( \|P_t(x_t - H_{t-1}u^i - o^i)\|_2 \leq \epsilon^{res} \) and \( \|o^i - r^i\|_2 \leq \epsilon^{out} \) where \( \epsilon^{res}, \epsilon^{out} > 0 \) are predefined accuracy parameters or when the procedure reaches the maximum number of iterations.

After the sparse outlier \( O_t \) is detected, we reduce the effect of \( O_t \) on the tracking process by the following outlier removal

\[
\mathbf{P}_t \odot \tilde{\mathbf{X}}_t = \mathbf{P}_t \odot (\mathbf{X}_t - O_t).
\]

(12)

In some cases, we can skip the corrupted entries in \( \mathbf{X}_t \) by re-updating the observation mask \( \mathbf{P}_t \) as

\[
p_{t|i_1 \ldots i_N} = \left\{ \begin{array}{ll}
0, & \text{if } x_{i_1 \ldots i_N} \text{ is missing or outlier}, \\
1, & \text{otherwise}.
\end{array} \right.
\]

(13)

Here, the removal step (12) still holds under the new binary mask \( \mathbf{P}_t \). This approach stems from the following observations. In the context of subspace tracking (ST), rejecting outliers can facilitate the tracking ability of ST estimators since only “clean” measurements involve the process [49].

Our next stage for estimating the tensor basis can indeed boil down to the ST problem with missing data, so the outlier rejection mechanism of (13) can improve performance. Please see Fig. 7 for an illustration that the outlier rejection mechanism can help improve the convergence rate of RACP when the fraction of corrupted entries is not too large. We refer to the mechanism (13) as a heuristic modification of the standard outlier removal (12) in RACP.

Stage 2: Estimation of factors \( \{U^{(n)}_t\}_{n=1}^N \)

The optimization (6) can be effectively solved by using the block-coordinate descent (BCD) technique. The main idea is to minimize alternately the surrogate \( \tilde{f}_t(\cdot) \) w.r.t. each factor \( U^{(n)}_t \) while fixing the remaining factors (hereafter denoted as \( \tilde{f}_t(U^{(n)}_t, \ldots) \) for short), that is,

\[
U^{(n)}_t = \underset{U^{(n)}_t}{\text{argmin}} \tilde{f}_t(U^{(n)}_t, \ldots).
\]

(15)

Minimization (15) is equivalent to

\[
\underset{U^{(n)}_t}{\text{argmin}} \sum_{k=t-L_t+1}^t \lambda^{-k} \left\| P^{(n)}_k \circ \left( \tilde{X}^{(n)}_k - U^{(n)}_t (W^{(n)}_k)^\top \right) \right\|_F^2,
\]

(16)

where \( \tilde{X}^{(n)}_k \) and \( P^{(n)}_k \) are the mode-\( n \) unfoldings of \( \tilde{X}_k \) and \( P_k \) respectively, and \( W^{(n)}_k \) is given by

\[
W^{(n)}_k = \left( \bigodot_{i=1}^{n-1} U^{(i)}_t \right) \odot \left( \bigodot_{i=n+1}^N U^{(i)}_{t-1} \right) \odot u^\top_k.
\]

(17)

The minimization of (16) can be decomposed into subproblems for each row \( u^{(n)}_m \) of \( U^{(n)}_t \), \( m = 1, 2, \ldots, N_t \), as

\[
\underset{u^{(n)}_m}{\text{argmin}} \sum_{k=t-L_t+1}^t \lambda^{-k} \left\| P^{(n)}_k \circ \left( \tilde{X}^{(n)}_{k,m} - W^{(n)}_k (u^{(n)}_m)^\top \right) \right\|_F^2.
\]

(18)

In practice, we can use the non-linear Jacobi iteration scheme to update (17) as \( W^{(n)}_k = \left( \bigodot_{i=1, i\neq m}^{n} U^{(i)}_{t,m} \right) \odot u^\top_k \). This scheme can be useful for parallel and/or distributed processing.
Algorithm 1: Robust Adaptive CP Decomposition (RACP)

Input:
- Tensor slices \( \{ \mathcal{P}_t \otimes \mathcal{X}_t \}_{t=1}^\infty \), \( \mathcal{X}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \).
- CP rank \( r \), forgetting factor \( \lambda \in [0,1] \).
- Predefined parameters: penalty \( \rho > 0 \), precision \( \epsilon_{\text{in}}, \epsilon_{\text{out}} > 0 \), maximum iteration \( K \), relaxation \( \alpha \in [1.5, 1.8] \), and \( \delta > 0 \).

Output: Loading factors \( \{ U^{(n)}_t \}_{n=1}^N \).

Initialization:
- \( \{ U_0^{(n)} \}_{n=1}^N \) is initialized randomly,
- \( \{ S_0^{(n)} \}_{n=1} = \delta I_{Nt} \).

for \( t = 1, 2, \ldots \) do

Stage 1: Online Outlier Rejection

\( \mathcal{H}_{t-1} = \bigodot_{n=1}^N U_{t-1}^{(n)} \)
\( o^0, z^0, u^0 \rightarrow 0 \)
for \( i = 1, 2, \ldots, K \) do

\( u^i = (\mathcal{H}_{t-1}^{i-1} \mathcal{P}_t \mathcal{H}_{t-1}) \# \mathcal{H}_{t-1}^{i-1} \mathcal{P}_t (x_t - o^{i-1} - z^{i-1} / \rho) \)
\( r^i = \alpha (\mathcal{P}_t (x_t - \mathcal{H}_{t-1} u^i) + (1 - \alpha) o^{i-1} \)
\( o^i = S_{t/\rho} (r^i - z^{i-1} / \rho) \)
\( z^i = z^{i-1} + \rho (o^i - r^i) \)
if stopping criteria are met break

end

Outlier Removal (Re-update of \( \mathcal{P}_t \) in (13) is optional)
\( \mathcal{P}_t \otimes \hat{\mathcal{X}}_t = \mathcal{P}_t \otimes (\mathcal{X}_t - \mathcal{O}_t) \)

Stage 2: Estimation of \( \{ U_1^{(n)} \}_{n=1}^N \)

for \( n = 1, 2, \ldots, N \) do

\( W_t^{(n)} = \left( \bigodot_{i=1}^n U_i^{(1)} \right) \circ \left( \bigodot_{i=n+1}^N U_i^{(n)} \right) \circ u_k^{(n)} \)
\( \hat{W}_t^{(n)} = \left( \left( W_t^{(n)} \right)^T \right)^T \)
for \( m = 1, 2, \ldots, n \) do

\( \hat{P}_{t,m}^{(n)} = \left[ \begin{array}{c} P_{t,m}^{(n)} \\ 0 \\ \cdots \\ 0 \end{array} \right] \)
\( \hat{X}_{t,m}^{(n)} = \left[ \begin{array}{c} X_{t,m}^{(n)} \\ \cdots \\ X_{t,m}^{(n)} \end{array} \right] \)
\( S_{t,m}^{(n)} = \lambda_{t-1,1,1}^{(n)} + \left( \hat{W}_t^{(n)} \right)^T \hat{P}_{t,m}^{(n)} \hat{W}_t^{(n)} \)
\( V_{t,m}^{(n)} = \left( \left( S_{t,m}^{(n)} \right)^{-1} \right)^T \)
\( \delta_{t,m}^{(n)} = \hat{P}_{t,m}^{(n)} \left( S_{t,m}^{(n)} - \hat{W}_t^{(n)} (u_{t-1,m}^{(n)})^T \right) \)
\( u_{t,m}^{(n)} = u_{t-1,m}^{(n)} + \left( \delta_{t,m}^{(n)} \right)^T \left( V_{t,m}^{(n)} \right)^T \)
end
end

Stage 3: (Optional) Normalization and re-estimation of \( u_t \)

Column-wise Normalization:
\[ U^{(n)}_{(t)} \] = \[ \frac{U^{(n)}_{(t)}}{\| U^{(n)}_{(t)} \|_2} \]

Re-estimation of \( u_t \):
\[ u_t = (H_t^T P_t H_t) \# H_t^T P_t (x_t - o_t) \]
where \( H_t = \bigodot_{n=1}^N U_t^{(n)} \)

where \( \hat{X}_{t,m}^{(n)} \) is the \( m \)-th row of \( \hat{X}_{t,m}^{(n)} \), and the row-mask matrix is given by \( \hat{P}_{t,m}^{(n)} = \text{diag}(\hat{P}_{t,m}^{(n)}(m_i,:)) \). Here, we introduce an efficient recursive least-squares (RLS) solver to minimize (18) effectively (see Algorithm 1 and the Appendix for its derivation).

Stage 3 (Optional): Normalization and re-estimation of \( u_t \)

In order to avoid numerical problems, we can perform the column-wise normalization on the updated factors \( \{ U_t^{(n)} \}_{n=1}^N \).

Therefore, the recursive rule for updating \( u_t^{(n)} \) becomes
\[ u_t^{(n)} = u_{t-1,m}^{(n)} + (\delta_{t,m}^{(n)})^T \left( V_{t,m}^{(n)} \right)^T, \]
2) Nonnegative Constraint: It is known that nonnegative tensor factorization (NTF) offers interesting properties, e.g., the resulting expression appears to be purely additive and the loading factors are “sparse” in general [52].

One of the simplest ways is to project the estimates (i.e., \( \mathbf{u}_t \) and \( \{ \mathbf{U}^{(n)} \}_{n=1}^{N} \)) on their nonnegative orthant at the end of each step of RACP, as introduced by Nguyen et al. in [53]. This approach offers a low complexity and yields a reasonable performance in some cases. However, it may not be optimal nor guarantee convergence in general. In this task, we aim to customize the updates of \( \mathbf{u}_t \) and \( \{ \mathbf{U}^{(n)} \}_{n=1}^{N} \) in order to deal with nonnegativity at each time \( t \).

In step 1, we replace the exact LS solution (8) with the following nonnegative least-squares (NNLS) problem:

\[
\mathbf{u}^i = \arg \min_{\mathbf{u}} \| \mathbf{P}_t (\mathbf{x}_t - \mathbf{o}^i - \mathbf{H}_{t-1} \mathbf{u}) \|^2 \quad \text{s.t.} \quad [\mathbf{u}]_j \geq 0 \quad \forall \mathbf{j}. \tag{26}
\]

Here, we can apply any provable NNLS algorithm to solve (26). The reader is referred to [54], [55] for good surveys on numerical methods for NNLS. In this work, we adopt the widely-used algorithm of Lawson and Hanson [55] which is implemented as the function \texttt{lsqnonneg} in MATLAB.

In step 2, the \( m \)-th row of \( \mathbf{U}^{(n)} \) can be derived by minimizing the following constrained version of (18):

\[
\mathbf{u}_{t,m}^{(n)} = \arg \min_{\mathbf{u}_{t,m}^{(n)}} \sum_{k=t-L+1}^{\infty} \lambda^{-k} \| \mathbf{P}_{t,m} (\mathbf{z}_{t,m}^{(n)})^\top - \mathbf{W}^{(n)} \mathbf{u}_{t,m}^{(n)} \|^2 \quad \text{s.t.} \quad [\mathbf{u}_{t,m}^{(n)}]_j \geq 0 \quad \forall \mathbf{j}. \tag{27}
\]

In order to solve (27), we apply the projected gradient method (i.e. proximal gradient on indicator function [56]). More concretely, the iterative procedure for updating \( \mathbf{u}_{t,m}^{(n)} \) is given by

\[
\mathbf{u}_t = \left( \mathbf{I}_r - \frac{\mathbf{S}_{t,m}^{(n)}}{\| \mathbf{S}_{t,m}^{(n)} \|^2} \right) \mathbf{u}_{t-1} - \frac{\mathbf{d}_{t,m}^{(n)}}{\| \mathbf{S}_{t,m}^{(n)} \|^2}, \quad \text{where} \quad l \text{ denotes the iteration index.} \tag{28}
\]

where \( l \) denotes the iteration index. We refer to this modification of RACP as NRACP.

IV. PERFORMANCE ANALYSIS

In this section, we present a theoretical convergence analysis for the proposed RACP method in Algorithm 1 while assuming that the underlying tensor dictionary \( \mathbf{D} \) does not change over time. Inspired by the recent results of our companion studies on robust subspace tracking [49] and tensor tracking [27], we establish a unified theoretical approach to analyse the convergence of the objective values \( \{ f_t (\mathbf{D}_t) \}_{t=1}^{\infty} \), as well as the solutions \( \{ \mathbf{D}_t \}_{t=1}^{\infty} \) generated by RACP.

A. Assumptions

In order to facilitate the convergence analysis, we make the following assumptions:

\((A1)\): Low-rank components \( \{ \mathbf{Y}_t \}_{t=1}^{\infty} \) of the observed tensor slices \( \{ \mathbf{X}_t \}_{t=1}^{\infty} \) are assumed to be deterministic and bounded. Entries of noise tensors \( \{ \mathbf{N}_t \}_{t=1}^{\infty} \) are zero-mean, independently and identically distributed (i.i.d.) with a small finite covariance, and bounded. Entries of \( \mathbf{X}_t \) are Frobenius-norm bounded, i.e., \( \| \mathbf{X}_t \|_F \leq \mathcal{M}_x < \infty \), for all \( t \).

\((A2)\): The tensor factors \( \{ \mathbf{U}^{(n)} \}_{n=1}^{N} \) remain unchanged over time, i.e., the tensor dictionary \( \mathbf{D} \) is fixed. The loading factors are Frobenius-norm bounded and the tensor rank \( r \) is fixed.

\((A3)\): Observation masks \( \{ \mathbf{P}_t \}_{t=1}^{\infty} \) are independent of \( \{ \mathbf{X}_t \}_{t=1}^{\infty} \), and their entries follow a uniform distribution. The number of observed entries of \( \mathbf{X}_t \) should be larger than the lower bound \( \mathcal{O}(rL \log(L)) \), where \( L = I_1 I_2 \ldots I_N \). Every row of the mode-\( n \) unfolding \( \mathbf{X}_t^{(n)} \) of \( \mathbf{X}_t \) is observed in at least \( r \) entries, for \( n = 1, 2, \ldots, N \). In addition, each observed entry of \( \mathbf{X}_t \) is corrupted by outliers independently of others, i.e., the index of outliers is also uniformly random.

\((A4)\): The surrogate function \( \tilde{f}_t (\cdot) \) is \( m \)-strongly multi-block convex, i.e., its second-order derivative w.r.t. each factor is positive-definite, \( \nabla^2_{\mathbf{u}} \tilde{f}_t (\mathbf{u}^{(n)}) \geq \mathbf{I} \) with \( m > 0 \). Among these assumptions, (A1) and (A2) are common for analysing the convergence of online learning algorithms, such as [24], [46], [49]. Indeed, (A1) holds in many situations, e.g., real data, such as audio, image and video data, are often bounded. (A2) is a strong assumption as it requires the tensor dictionary to be constant with time. Also, the bound in (A2) prevents arbitrarily large values in \( \mathbf{U}^{(n)} \) and ill-conditioned computation. Along with (A1), it is interpreted as the simplest possible data model in (robust) tensor tracking where tensor slices are assumed to be generated from a stationary process. Theoretically, stationary processes are often “easier” to model and analyse than nonstationary ones as their statistical properties remain constant over time. Accordingly, stationarity has become a common assumption underlying many statistical procedures in general and tracking tools in particular to study their convergence and asymptotic behavior. In this work, a novel theoretical approach is established to analyse the convergence behavior of RACP in stationary environments. We leave the convergence analysis of RACP under a nonstationary model where the tensor dictionary is time-varying to a future work. Assumption (A3) is also common, under which the index of missing entries is uniformly random. Moreover, with respect to the imputation of missing values and recovery of low-rank components, the uniform randomness allows the sequence of binary masks \( \{ \mathbf{P}_t \}_{t=1}^{\infty} \) to admit stable recovery [57]. The next two constraints of (A3) are fundamental conditions to prevent the underdetermined imputation problem [58]–[60]. The last constraint of (A3) plays a similar role as the first one but accounting for sparse outliers. Assumption (A4) allows us to derive several nice results in the convergence analysis. In fact, as the Hessian matrix of \( \tilde{f}_t (\cdot) \) w.r.t. each factor is

The four assumptions (A1)-(A4) are used for the purpose of convergence analysis only. The proposed RACP algorithm can work well in many other scenarios (see Section V for an illustration).
already positive semidefinite, (A4) can be achieved with a good initialization $D_0$ or by simply adding a convex regularization term to $\tilde{f}(\cdot)$ or $\tilde{f}_\tau(\cdot)$.

B. Main Results

Given the assumptions of (A1)-(A4), our main theoretical result can be stated in the following theorem:

**Theorem 1.** Given (A1)-(A4), $\ell_t = t$ and let $D_t$ be the solution generated by Algorithm 1 at each time $t$. When $t \to \infty$,

- $f_t(D_t) - \tilde{f}_t(D_t) \xrightarrow{a.s.} 0$;
- $\nabla f_t(D_t) \xrightarrow{a.s.} 0$.

Accordingly, $D_t$ is almost surely a stationary point of $f_t(\cdot)$ when $t$ tends to infinity.

The proof of this theorem follows immediately Proposition 1 and Lemmas 1 and 2, to be stated shortly. We detail their proofs in our supplementary material attached to this manuscript.

**Proposition 1 (Key Properties).** Given (A1)-(A4), $\ell_t = t$, and denote the error function $e_t := \tilde{f}_t - f_t$. If $\{D_t, \Omega_t, u_t\}_{t=1}^\infty$ is a sequence of variables generated by Algorithm 1, then

(a) Boundedness: $\{D_t, \Omega_t, u_t\}_{t=1}^\infty$ are uniformly bounded;
(b) Forward Monotonicity: $f_t(D_t) \geq f_t(D_{t+1})$;
(c) Backward Monotonicity: $|f_t(D_t) - f_{t-1}(D_{t-1})| = O(1/t)$;
(d) Stability of estimates: $|f_t(D_t) - f_{t-1}(D_{t-1})| = O(1/t)$.

**Proof Sketch.** Part (a) can be derived from applying the same arguments of Proposition 1 in our companion work on tensor tracking [27]. Part (b) and (c) are trivial due to the proposed iteration scheme. Part (d) can be obtained by exploiting the Lipschitz continuity and multi-block convexity of the surrogate function $\tilde{f}_t(\cdot)$. We indicate Part (e) by using Part (d) and the Lipschitz continuity of $f_t(\cdot)$ and $\tilde{f}_t(\cdot)$.

**Lemma 1 (Almost sure convergence).** The sequence of $\{f_t(D_t)\}_{t=1}^\infty$ converges almost surely as $t \to \infty$. The sequence of objective values $\{f_t(D_t)\}_{t=1}^\infty$ converges to the same limit of its surrogate $\{\tilde{f}_t(D_t)\}_{t=1}^\infty$, i.e.,

$$f_t(D_t) \to \tilde{f}_t(D_t) \quad a.s.$$

**Proof Sketch.** We first prove that

$$\sum_{t=1}^\infty \mathbb{E}\left[\delta\left(\tilde{f}_{t+1}(D_{t+1}) - \tilde{f}_t(D_t)\right) F_t\right] < \infty,$$

where $F_t = \{D_t, \Omega_t, u_t\}_{0 \leq s \leq t}$ records all past estimates of RACP at time $t$ and the indicator function $\delta$ is defined as

$$\delta_t = \begin{cases} 1 & \text{if } \mathbb{E}\left[\tilde{f}_{t+1}(D_{t+1}) - \tilde{f}_t(D_t)\right] F_t > 0, \\ 0 & \text{otherwise}. \end{cases}$$

Thanks to the quasi-martingale convergence theorem [61, page 51], (30) implies that $\{\tilde{f}_t(D_t)\}_{t=1}^\infty$ converges almost surely as $t \to \infty$.

We next prove $\{f_t(D_t)\}_{t=1}^\infty$ and $\{\tilde{f}_t(D_t)\}_{t=1}^\infty$ converge to the same limit by showing

$$\sum_{t=1}^\infty \frac{\tilde{f}_t(D_t) - f_t(D_t)}{t+1} < \infty.$$
The studies in [48] and [49] consider the problem of robust online PCA/subspace tracking which can handle data corruptions (i.e., outliers and/or missing entries). These studies are designed for tracking the time-variant subspace – an object different from ours – which leads to some differences from our analysis. In particular, their main goal is to develop provable algorithms for minimizing the expected cost function in an online manner, and then indicate that their algorithm converges to a stationary point or global optimum under certain conditions. Our optimization, however, minimizes an exponential weighted cost function constructed on the latest data streams (i.e., tensor slices). Moreover, [48] does not require the solution derived from the subspace update stage to be necessarily optimal, but full column rank only at each time \( t \) (see [48, Theorem 1]). However, it is a sufficient condition that is highly leveraged in our analysis. In addition, our object is a set of multiple loading factors, instead of a single subspace matrix as in [48], [49].

The studies most related to ours are those in [24], [25], [27], [28], [29], [30], and [31], which also investigate the tensor tracking problem. However, they consider only outlier-free streaming tensors. By contrast, we here provide a more unified convergence analysis that is able to deal with both missing data and outliers. Also, our results are stronger than those of [24], [25], which are limited to the case of third-order streaming tensors with \( \lambda = 1 \).

V. Experiments

In this section, we provide several experiments on both synthetic and real data to demonstrate the effectiveness of RACP and its variant. In particular, the performance of our method is evaluated in comparison with the state-of-the-art algorithms with respect to the following aspects: (i) impact of outliers, (ii) impact of missing data, and (iii) tracking ability in noisy and time-varying environments.

A. Experiment Setup

At \( t = 0 \), we randomly initialize \( \hat{U}_{t}^{(n)} \in \mathbb{R}^{T_s \times r} \) whose entries are i.i.d. from a normal distribution \( \mathcal{N}(0,1) \). \( n = 1, \ldots, N \). When \( t \geq 1 \), \( \hat{U}_{t}^{(n)} \) is varied according to the following model:

\[
\hat{U}_{t}^{(n)} = \hat{U}_{t-1}^{(n)} + \epsilon N_{t}^{(n)},
\]

where \( N_{t}^{(n)} \) is a Gaussian noise matrix (with zero-mean and unit-variance), and \( \epsilon \) is a positive time-varying factor used to control the variation of \( \hat{U}_{t}^{(n)} \) between \( t \) and \( t-1 \).

The \( t \)-th slice \( \mathcal{X}_t \) is then generated under the data model

\[
\mathcal{X}_t = \mathcal{P}_t \odot \left( \mathbf{I} \prod_{n=1}^{N} x_n \hat{U}_t^{(n)} \right) \left( \mathbf{I} \prod_{n=1}^{N} x_{n+1} \mathbf{u}_t^\top + \mathcal{O}_t + \mathcal{N}_t \right),
\]

where \( \mathcal{P}_t \) is a binary observation mask according to a Bernoulli distribution with probability of observing data \( 1 - \omega_{\text{miss}} \), \( \mathcal{N}_t \) is a Gaussian noise tensor with i.i.d. entries \( \mathcal{N}(0,\sigma^2_n) \), \( \mathcal{O}_t \) is a sparse outlier tensor whose entries are drawn uniformly from the range \( [0, \omega_{\text{outlier}}] \) and the indices of outliers also follow a Bernoulli distribution with probability \( \omega_{\text{outlier}} \), and \( \mathbf{u}_t \in \mathbb{R}^{r \times 1} \) is a standard normal random vector.

Fig. 1: Effect of data corruptions (outliers and missing values) on performance of RACP. White color denotes perfect estimation (i.e., \( \text{RE}(D,D) \leq 0.01 \)), black color denotes failure (i.e., \( \text{RE}(D,D) \geq 0.5 \)), and gray color is in between.

To evaluate the estimation accuracy, we use the metric

\[
e(\hat{Z}_{\text{estimate}}, \hat{Z}_{\text{true}}) = \frac{\| \hat{Z}_{\text{estimate}} - \hat{Z}_{\text{true}} \|_F}{\| \hat{Z}_{\text{true}} \|_F},
\]

where \( \hat{Z}_{\text{estimate}} \) (resp. \( \hat{Z}_{\text{true}} \)) refers to the estimation (resp. ground truth). Due to the permutation and scaling indeterminacy of CP decomposition, the estimation \( \hat{U}_t^{(n)} \) of \( U_t^{(n)} \) at each time \( t \) will be permuted and scaled such that it matches \( U_t^{(n)} \) before measuring the error metric (42). In particular, we derive the ordered and scaled version \( \hat{U}_{t-\epsilon}^{(n)} \) of \( U_t^{(n)} \) from

\[
\hat{U}_{t-\epsilon}^{(n)} = \hat{U}_t^{(n)} (P^{(n)})^\top (Q^{(n)})^{-1}, \quad n = 1, 2, \ldots, N,
\]

where the permutation matrix \( P^{(n)} \in \mathbb{R}^{r \times r} \) and the diagonal matrix \( Q^{(n)} \in \mathbb{R}^{r \times r} \) are obtained by

\[
\{P^{(n)}, Q^{(n)}\} = \min_{P, Q} \| \hat{U}_t^{(n)} - U_t^{(n)} PQ \|_F^2.
\]

The relative errors are then computed as

\[
\text{RE}(\hat{D}_t, D_t) = \frac{1}{N} \sum_{n=1}^{N} e(\hat{X}_{t-\epsilon}^{(n)}, U_t^{(n)}),
\]

\[
\text{RE}(\hat{X}_t, X_t) = e(\hat{X}_t, X_t),
\]

where \( \hat{X}_t \) is a reconstructed version of the true slice \( X_t \) derived from the recent updated loading factors.

B. Robustness of RACP

We first investigated the robustness of RACP against gross data corruptions. Specifically, we changed the density of outliers and missing data, and then measured the relative error between the ground truth and RACP’s estimation.

In this task, we used a synthetic \( 4 \)-th order streaming tensor of size \( 20 \times 20 \times 20 \times 1000 \) and the CP rank was set at \( r = 2 \) and \( r = 5 \). The noise level \( \sigma_n \) and the time-varying factor \( \epsilon \) were fixed at \( 10^{-3} \) and \( 10^{-2} \), respectively. We consider the case where the underlying data were corrupted by strong outliers with \( \omega_{\text{outlier}} = 10 \). The fraction of outliers (\( \omega_{\text{outlier}} \)) and missing data (\( \omega_{\text{miss}} \)) were varied in the range \([5\%, 95\%]\). Throughout our experiments, the forgetting factor \( \lambda \) was fixed at 0.5 while the window length was \( L_t = t \).

Phase transitions w.r.t. the pair of \( \{\omega_{\text{outlier}}, \omega_{\text{miss}}\} \) are shown in Fig. 1. The results indicate that there is a large region in which our estimation was successful. Particularly, RACP
The two synthetic rank-5 tensors of size $20 \times 20 \times 20 \times 1000$ and $20 \times 20 \times 20 \times 20 \times 1000$ were used in this task. The fraction of missing entries and sparse outliers were both set to 5%. The outlier intensity $A_{\text{outlier}}$ and the noise factor $\sigma_n$ were fixed at 10 and $10^{-5}$, respectively. The value of the time-varying factor $\epsilon$ was varied from $[10^{-4}, 10^{-1}]$. An abrupt change was created at $t = 600$ to assess how fast RACP converges. We can see from Fig. 2 that RACP’s convergence rate is not much affected by the value of $\epsilon$ but that its estimation accuracy is.

To demonstrate the effectiveness of the proposed algorithm, we compared the performance of RACP with the state-of-the-art adaptive CP decompositions, namely TeCPSGD [24], OLSTEC [25], and ACP [27]. To have a fair comparison, the algorithm parameters were set by default as suggested by their authors. These algorithms are dependent on a forgetting factor; we set its value at 0.7, 0.001, and 0.5 for OLSTEC, TeCPSGD, and ACP, respectively. The penalty parameter was set at $10^{-3}$ and $10^{-1}$ for OLSTEC and TeCPSGD, respectively.

Since OLSTEC and TeCPSGD are only capable of tracking third-order streaming tensors, we here used a synthetic streaming tensor of size $20 \times 20 \times 1000$ and its rank was fixed at 5. The noise level and time-varying factor were both kept at $10^{-2}$.

Next, we evaluated the tracking ability of RACP in time-varying environments. The noise level and time-varying factor were both kept at 10. In the presence of huge data corruptions (e.g., $\omega_{\text{outlier}} \geq 70\%$ and/or $\omega_{\text{miss}} \geq 70\%$), the proposed algorithm failed to track the underlying tensor model.

We next investigated the performance of RACP when loading factors are not normal in comparison with other adaptive CP algorithms. In particular, the initial factors $\{\mathbf{U}^{(n)}_0\}_{n=1}^N$ were sampled from a uniform distribution on the $(0, 1)$ interval instead of a Gaussian one. The time-varying model (40) was replaced with $\mathbf{U}^{(n)}_t = \mathbf{U}^{(n)}_{t-1} + c\mathbf{N}^{(n)}_t$ where $\mathbf{N}^{(n)}_t$ was also an i.i.d. uniform random matrix from 0 to 1. The parameter specifications were kept as in the previous experiment. Results are illustrated in Fig. 5. We can see that the proposed RACP algorithm still tracks the loading factors successfully over time while the state-of-the-art CP algorithms failed.

The experimental results in Figs. 3, 4, and 5 suggest that the outlier rejection step (e.g. Step 1 in RACP) using the ADMM solver plays an important role in the tracking process when observations are corrupted by sparse outliers. Therefore, we next evaluated the effectiveness of the proposed outlier rejection by applying the ADMM solver to other trackers: TeCPSGD and OLSTEC. We here reused the experiment setup above and created an abrupt change at $t = 600$. We can see from Fig. 6 that the combination of the ADMM solver and OLSTEC resulted in the best convergence rate and estimation accuracy. This is probably due to the effectiveness of the
Fig. 5: Non-Gaussian loading factors.

Fig. 6: Outlier rejection with different trackers.

Fig. 7: Convergence rate of RACP and its modification with the re-update of $P_t$ as defined in (13): $\omega_{\text{miss}} = 10\%$, $\omega_{\text{outlier}} = 10\%$, $A_{\text{outlier}} = 10, \sigma = 10^{-2}$, and $\epsilon = 10^{-2}$.

Fig. 8: Incomplete observations & time-varying scenarios: Performance of NRACP on a synthetic rank-5 tensor of size $50 \times 50 \times 50 \times 500$; $\sigma_n = 10^{-3}$, $A_{\text{outlier}} = 10, \omega_{\text{outlier}} = 10\%$. NRACP converges. The results are shown in Fig. 8. We can see that the relative error between the estimation and ground truth converged to an error floor. Furthermore, the missing density $\omega_{\text{miss}}$ impacted only the convergence rate of NRACP. Specifically, the lower the missing density $\omega_{\text{miss}}$ was, the faster NRACP converged. Next, we studied the robustness of NRACP against the noise variance in comparison with NSOAP [53] and NsTEF [62]. Since both algorithms are only feasible for third-order tensors without corruptions (outliers and missing values), we used a synthetic outlier-free tensor of size $50 \times 50 \times 1000$ and rank 5 for this task. The time-varying factor $\epsilon$ was set at $10^{-3}$. Both NRACP and NsTEF used random initialization while the first 50 temporal slices were used to construct a good initialization tensor for NSOAP. Performance comparison results are illustrated in Fig. 9. At a low SNR, NSOAP provided a better estimation accuracy than NRACP and NsTEF. However, the proposed NRACP outperformed NSOAP and NsTEF at the high SNR, see Fig. 9(b). In the presence of abrupt changes, the convergence rate of NRACP was fast while NSOAP and NsTEF failed to track the change.

D. Real Datasets

To demonstrate the use of RACP with real-world datasets, we consider the following tasks: (i) tracking the online low-rank approximation of real-world data streams, (ii) multichannel EEG analysis, and (iii) video background modeling and fore-
Fig. 9: Nonnegative adaptive CP decompositions: Outlier-free, full observations and an abrupt change at t = 600.

Fig. 10: Experimental results on the Intel Berkeley Lab data.

Fig. 11: Completion accuracy of adaptive CP algorithms on real-world data streams.

ground detection. See Tab. II for a summary of the real datasets used in this paper.

**Task 1: Tracking the online low-rank approximation and online data completion**

**Datasets:** In this task, we used three real datasets: Intel Berkeley Lab, Internet Traffic, and Taxi Trip Record. The first dataset is a collection of timestamped topology information gathered from 54 positions (sensors) in the Intel Berkeley Re- search Lab. Specifically, these sensors collected: temperature (in degree Celsius), humidity (ranging from 0% to 100%), light (in Lux), and voltage (in volt, ranging from 2 to 3). Accordingly, we represent the sensor data by a three-order tensor of size $54 \times 4 \times 1152$ (i.e., sensor $\times$ measurement $\times$ time). The second dataset is the link traffic data which was collected from the Internet2 backbone network Abilene. The Abilene backbone is relatively small with 12 routers, 15 links, and 144 flow entries in each traffic matrix of size $12 \times 12$. We concatenated all these traffic matrices into a tensor of size $12 \times 12 \times 48384$. The third dataset describes yellow taxi trip records in the pairs of 265 pick-up and drop-off sites in New York. Each trip record contains several attributes, such as pick-up/drop-off times and locations, elapsed trip distance, rate type, and payment method. In this work, we specifically constructed a third-order tensor of size $265 \times 265 \times 3672$ (i.e., origin $\times$ destination $\times$ time).

**Experiments & Results:** Following the same experiment setup as in subsection V-A, data corruptions were generated as follows. The locations of missing entries and sparse outliers are randomly generated with probabilities $\omega_{\text{miss}}$ and $\omega_{\text{outlier}}$, respectively. Outlier values are drawn uniformly from the range $[0, \max(\mathcal{X})]$ where $\max(\mathcal{X})$ is the largest absolute...
value in the underlying data $\mathbf{X}$. In this experiment, we chose the value of $\omega_{\text{miss}}$ and $\omega_{\text{outlier}}$ among the range $\{5\%, 10\%, 20\%, 40\%\}$. As the true rank is unknown, we first varied its value from 2 to 10 and then chose the “best” one based on the averaged reconstruction error, see Fig. 10(a) for an example. We compared the performance of RACP against the two adaptive CP algorithms TeCPSGD [24] and OLSTEC [25]. Both algorithms are dependent on the forgetting factor $\lambda$, and its value was set at 0.98, 0.001, and 0.7, respectively. The penalty parameter $\mu$ was set at 1 for both TeCPSGD and OLSTEC. The experimental result in Fig. 11 indicates that RACP outperforms TeCPSGD and OLSTEC.

**Task 2: Multichannel EEG Analysis**

**Datasets:** In this task, we used two public electroencephalogram (EEG) datasets: ERPWAVELAB\(^9\) and Epileptic EEG Data\(^10\). The former dataset contains wavelet-transformed versions of EEG signals that were collected from 14 subjects during the hand stimulation (i.e., proprioceptive pulls of the left and right hands) for inter-trial phase coherence analysis. In particular, these EEG signals were recorded using an electrode system of 64 channels with 28 measurements per subject. The continuous wavelet transform was then applied to represent these signals in the time-frequency domain. The latter dataset includes 20 EEG recordings of 6 patients diagnosed with epilepsy at the American university of Beirut medical center. The EEG data were recorded by using a system of 21 channels with a sampling rate of 500Hz. The dataset includes 3895 normal segments and 3850 abnormal segments in which there are 3034 partial seizures, 705 electrographic seizures, and 3850 abnormal segments in which there are 3034 partial seizures, 705 electrographic seizures, and 3850 normal segments and 3850 abnormal segments in which there.

**Incomplete Multichannel EEG Analysis:** Here, we used the ERPWAVELAB dataset and followed the same experimental setup as in [27], [63], [64] to demonstrate the use of RACP with real EEG signals. We constructed an EEG tensor of size $28 \times 64 \times 4392$ (i.e., $\text{measurements} \times \text{channel} \times \text{time-frequency}$). To generate incomplete observations, signals from some channels at each time were randomly assumed to be missing. As suggested in [63], [64], we set the tensor rank at $r = 3$. The performance of RACP was compared with two adaptive CP algorithms NL-PETRELS [63] and ACP [27]. We fixed the forgetting factor $\lambda$ at 0.999 and 0.5 for NL-PETRELS and ACP, respectively. As NL-PETRELS requires a warm start, we ran the batch CP-WOPT algorithm [64] with the first 1500 tensor slices, whereas random initialization was used for ACP and RACP. In this experiment, we aimed to factorize the EEG tensor into three basis components w.r.t. spatial domain, time-frequency domain, and measurement mode. As there is no real ground truth, we used the results (i.e., CP factors) derived from applying the batch CP-ALS algorithm to the EEG tensor with full observations as benchmarks. Experimental results are shown in Tab. III and Fig. 13. They indicate that RACP outperforms NL-PETRELS and provides a slightly better estimation than ACP, especially in the presence of highly incomplete observations (e.g., ≥ 40 channels are missing).

**Anomaly EEG Detection:** We demonstrate the use of RACP to detect abnormal activities in the brain (i.e., epileptic seizures) with the epileptic EEG dataset. Here, we adopted a simple but effective way to predict abnormalities in multidimensional data streams [65], i.e. by modeling the abnormality of a tensor.

![Fig. 12: Epileptic EEG Dataset.](image-url)

**Fig. 12:** Epileptic EEG Dataset.

![Fig. 13: First component of EEG factors when 40/60 EEG channels are missing.](image-url)

**Fig. 13:** First component of EEG factors when 40/60 EEG channels are missing.

<table>
<thead>
<tr>
<th>Missing channels</th>
<th>NL-PETRELS</th>
<th>ACP</th>
<th>RACP (Proposed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/64</td>
<td>0.051</td>
<td>0.063</td>
<td>0.056</td>
</tr>
<tr>
<td>10/64</td>
<td>0.062</td>
<td>0.025</td>
<td>0.023</td>
</tr>
<tr>
<td>20/64</td>
<td>0.077</td>
<td>0.011</td>
<td>0.014</td>
</tr>
<tr>
<td>30/64</td>
<td>0.121</td>
<td>0.097</td>
<td>0.086</td>
</tr>
<tr>
<td>40/64</td>
<td>0.891</td>
<td>0.132</td>
<td>0.119</td>
</tr>
<tr>
<td>50/64</td>
<td>1.325</td>
<td>1.137</td>
<td>0.982</td>
</tr>
</tbody>
</table>

**TABLE III:** Averaged errors of adaptive CP algorithms for multichannel EEG analysis from incomplete observations.
Fig. 14: The error $e_t$ over time with $\alpha = 1.5$ and $L_t = t$. Normal data which are inaccurately labelled as abnormal are referred to as “false positive”.

(streaming) slice $\mathcal{Y}_t$ by its recovery error

$$e_t = \left\| \mathcal{P}_t \otimes (\mathcal{Y}_t - \mathcal{Y}_t \prod_{n=1}^{N} U_t^{(n)} U_t^{(n)^\mathbb{R}}) \right\|_F / \left\| \mathcal{Y}_t \right\|_F, \quad (48)$$

where $\{ U_t^{(n)} \}_{n=1}^{N}$ is the set of solutions generated by RACP at time $t$. It is also worth noting that the error $e_t$ is relatively proportional to the norm of the outlier $\mathcal{O}_t$. We label $\mathcal{Y}_t$ based on the following rule

$$e_t \begin{cases} \text{abnormal} & \tau_t = \text{mean}(\{ e \}_L) + \alpha \text{std}(\{ e \}_L), \\ \text{normal} & \end{cases} \quad (49)$$

where $\{ e \}_L$ denotes the set of $e$, with $t - L_t < \tau \leq t$. We followed the method in our companion work on epileptic spike detection [66] to obtain the time-frequency representation of multichannel EEG segments (including normal data and seizures), and hence the corresponding EEG tensors of size $19 \times 20 \times 500$ (i.e., channel $\times$ scale $\times$ time). The resulting tensors were then concatenated into a huge tensor of which each channel contains 500 samples. Also, 20 wavelet scales are chosen in the range $[4, 8]$.

**Table IV:** Anomaly EEG detection results. Sensitivity and specificity measure the percentage of abnormal and normal data detected correctly, respectively. Accuracy indicates the overall performance.

<table>
<thead>
<tr>
<th>Value of $\alpha$</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>42.21%</td>
<td>53.02%</td>
<td>47.57%</td>
</tr>
<tr>
<td>0.5</td>
<td>59.74%</td>
<td>66.48%</td>
<td>63.29%</td>
</tr>
<tr>
<td>1</td>
<td>72.80%</td>
<td>74.38%</td>
<td>73.59%</td>
</tr>
<tr>
<td>1.5</td>
<td>81.58%</td>
<td>85.16%</td>
<td>83.36%</td>
</tr>
<tr>
<td>2</td>
<td>50.16%</td>
<td>53.54%</td>
<td>51.83%</td>
</tr>
</tbody>
</table>

11As indicated in the EEG dataset description report, the data of two channels Cz and Pz were omitted. Thus, we have 19 EEG channels left and each channel contains 500 samples. Also, 20 wavelet scales are chosen in the range $[4, 8]$.

12Video Sequences: http://jacarini.dinf.usherbrooke.ca.

Fig. 15: Three video sequences used in this paper.
Fig. 16: Qualitative illustration of video background modeling results.

Fig. 17: Qualitative illustration of video foreground detection results.

VI. Conclusions
In this paper, we have addressed the problem of robust tensor tracking in the presence of both missing data and outliers. Under the CP/PARAFAC model, a novel robust adaptive CP decomposition called RACP has been proposed to track the low-rank approximation of streaming tensors from uncertain, noisy, and imperfect measurements. Its convergence analysis has been established to guarantee that the solution generated by RACP converges to a stationary point asymptotically. Experimental results indicate that RACP is capable of estimating the tensor factors as well as tracking their variations over time with high accuracy, and that RACP outperformed the state-of-the-art adaptive CP algorithms in both simulated and real data tests.

Appendix: Derivations of Tensor Factor Tracking
The optimal solution of (18) can be derived by setting its derivative to zero

\[
\sum_{k=t-L_t+1}^{t} \lambda^{t-k} (W_k^{(n)})^\top P_{k,m}^{(n)} x_{k,m}^{(n)} = \sum_{k=t-L_t+1}^{t} \lambda^{t-k} (W_k^{(n)})^\top P_{k,m}^{(n)} W_k^{(n)} u_m^{(n)} \tag{50}
\]

Instead of solving (50) directly, we propose a more elegant recursive way to obtain \( u_{t,m}^{(n)} \) as follows.

First, let us denote the left hand side of (50) by \( d_{t,m}^{(n)} \), and the expression \( \sum_{k=t-L_t+1}^{t} \lambda^{t-k} (W_k^{(n)})^\top P_{k,m}^{(n)} W_k^{(n)} \) by \( S_{t,m}^{(n)} \).
Accordingly, (50) becomes
\[ S_{t,m}^{(n)}(u_{t,m}^{(n)})^\top = d_{t,m}^{(n)}. \]  
(51)

Interestingly, both \( d_{t,m}^{(n)} \) and \( S_{t,m}^{(n)} \) can be updated recursively:
\[ d_{t,m}^{(n)} = \lambda \delta d_{t,m}^{(n)} + (\hat{W}_t^{(n)})^\top \hat{P}_{t,m}^{(n)} \tilde{S}_{t,m}^{(n)}, \]  
(52)
\[ S_{t,m}^{(n)} = \lambda S_{t-1,m}^{(n)} + (\hat{W}_t^{(n)})^\top \hat{P}_{t,m}^{(n)} \hat{W}_t^{(n)}. \]  
(53)

where
\[ \hat{W}_t^{(n)} = [(\hat{W}_t^{(n)})^\top (\hat{W}_t^{(n)})^{*} ]^{T}, \]  
(54)
\[ \tilde{S}_{t,m}^{(n)} = [S_{t,m}^{(n)} - \hat{W}_t^{(n)} u_{t,m}^{(n)}]^T, \]  
(55)
\[ \hat{P}_{t,m}^{(n)} = \begin{bmatrix} \hat{P}_{t,m}^{(n)} & 0 \\ 0 & -\lambda I \end{bmatrix}. \]  
(56)

Therefore, we can rewrite (51) as
\[ S_{t,m}^{(n)}(u_{t,m}^{(n)})^\top = \lambda d_{t,m}^{(n)} + (\hat{W}_t^{(n)})^\top \hat{P}_{t,m}^{(n)} \tilde{S}_{t,m}^{(n)} = \lambda S_{t-1,m}^{(n)}(u_{t-1,m}^{(n)}) + (\hat{W}_t^{(n)})^\top \hat{P}_{t,m}^{(n)} \tilde{S}_{t,m}^{(n)}. \]

Multiplying both sides by \((S_{t,m}^{(n)})^{-1}\) results in
\[ u_{t,m}^{(n)} = u_{t,m}^{(n)} + (\delta \tilde{S}_{t,m}^{(n)})^T (\hat{W}_t^{(n)})^T, \]  
(57)

where
\[ \delta \tilde{S}_{t,m}^{(n)} = \hat{P}_{t,m}^{(n)}((\tilde{S}_{t,m}^{(n)} - \hat{W}_t^{(n)}(u_{t-1,m}^{(n)}))^T), \]  
(58a)
\[ \hat{V}_t^{(n)} = (S_{t,m}^{(n)})^{-1}(\hat{W}_t^{(n)})^T. \]  
(58b)

Collecting all rows \( u_{t,m}^{(n)} \) together, \( m = 1, 2, \ldots, I_n \), a simplified version of (57) for updating the whole \( U_t^{(n)} \) can be given by
\[ U_t^{(n)} = U_{t-1}^{(n)} + \Delta \tilde{X}_t^{(n)}(V_t^{(n)})^T. \]  
(59)

Here, the error matrix \( \Delta \tilde{X}_t^{(n)} \in \mathbb{R}^{I_n \times 2J_n} \) with \( J_n = \prod_{i \in \mathcal{I}_n} I_i \) is defined as
\[ \Delta \tilde{X}_t^{(n)} = \hat{X}_t^{(n)}(\hat{X}_t^{(n)})^T - \tilde{X}_t^{(n)}(U_{t-1,m}^{(n)} (\hat{W}_t^{(n)})^T)^T. \]  
(60)

with \( \tilde{X}_t^{(n)} = [X_t^{(n)}(X_t^{(n)})^T] \in \mathbb{R}^{I_n \times 2J_n} \) and the coefficient matrix \( V_t^{(n)} \in \mathbb{R}^{2J_n \times 2J_n} \) is computed as
\[ V_t^{(n)} = (S_t^{(n)})^{-1}(\hat{W}_t^{(n)})^T, \]  
(61)

where the matrix \( S_t^{(n)} \in \mathbb{R}^{r \times r} \) is recursively updated as follows
\[ S_t^{(n)} = \lambda S_{t-1,m}^{(n)} + (\hat{W}_t^{(n)})^T \hat{W}_t^{(n)}. \]  
(62)

In this way, we can skip several operations and save a memory storage of \( O(\sum_{n=1}^N (I_n-1)(I_n + r^2)) \). Specifically, the cost of computing (62) is \( O(r^2 \prod_{i=1, i \neq n}^N I_i) \) flops. The computation of (61) also requires a cost of \( O(r^2 \prod_{i=1, i \neq n}^N I_i) \) flops because \( S_t^{(n)} \) is of size \( r \times r \) and its inverse computation is not expensive and independent of the tensor dimension. The error matrix \( \Delta \tilde{X}_t^{(n)} \) in (60) can be derived from Step 1 by reshaping the

---

References


