FAST AND ROBUST TRIPLE TENSOR DECOMPOSITION WITH DATA CORRUPTION

Nguyen Quy Dang[⋆♯], *Do Minh Nhat*[⋆♯], *Thanh Trung Le*[⋆], *Nguyen Linh Trung*[⋆], *Karim Abed-Meraim*[†]

[⋆] VNU University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam
[†] PRISME Laboratory, University of Orleans, France

ABSTRACT

In this paper, we study the problem of tensor triple decomposition in the presence of data corruption and propose a fast and robust algorithm, called TriTD-ADMM. Data corruption is modeled as a sparse residual component penalized by the ℓ_1-norm regularization. To address it efficiently, we design an effective alternating direction method of multipliers (ADMM) equipped with a proximal operator for the ℓ_1-regularizer and adaptive penalty updates. Moreover, we develop a reshape–permute acceleration strategy (RPAS) that replaces expensive Kronecker-based computations with efficient operations, thereby further improving optimization efficiency. Experimental results on real-world datasets show that TriTD-ADMM achieves competitive accuracy and significant speedups, ranking as the fastest method on seven out of eight benchmarks.

***Index Terms*—** Tensor decomposition, triple decomposition, ADMM, data corruption, ℓ_1-norm regularization.

1. INTRODUCTION

Tensors (multiway arrays) offer a natural representation for multivariate and high-dimensional datasets across space, time, and modalities [1]. Accordingly, tensor decomposition have enabled significant advances in compression, completion, and representation learning, with applications ranging from biomedical signal processing, hyperspectral imaging, streaming video, and communication systems [2–6], among others. Most of these applications are built upon two foundational tensor formats: CP and Tucker. The former factorizes a tensor into rank-1 components, offering strong identifiability and uniqueness guarantees under mild conditions, whereas the latter trades identifiability for flexibility by introducing a core tensor [1]. However, with the emergence of large-scale and increasingly complex structure datasets, striking a balance among compactness, effectiveness and computational scalability still remains a challenge.

Recently, triple tensor decomposition (TriTD) was proposed as a more flexible alternative to CP and Tucker [7]. TriTD factorizes a third-order tensor into three smaller core tensors of the same order and introduces the notion of triple rank, which can be strictly lower than the corresponding CP or Tucker ranks. Consequently, TriTD has found some nice applications such as spatio-temporal traffic completion [8], link-load recovery [9], hyperspectral super-resolution [10], and online video completion [11].

Related Works: The original TriTD solver adopts a Tikhonov-regularized alternating least-squares (ALS) method [7], which offers efficient least-squares recovery for third-order tensors and establishes the foundation for later TriTD methods. Building on this framework, Chen *et al.* in [8] introduced a Barzilai–Borwein gradient algorithm for spatio–temporal traffic completion, enabling scalable recovery of large-scale Internet traffic data with accelerated convergence. Wu *et al.* in [12] incorporated manifold regularization and nonnegativity for image compression and representation, demonstrating the benefit of preserving intrinsic geometric structure while enforcing physically meaningful nonnegative factors of TriTD for real-world data. Liao *et al.* in [13] integrated a hypergraph prior to capture higher-order sample relations, improving clustering performance by explicitly modeling complex inter-sample dependencies. Cui *et al.* in [10] adapted TriTD to hyperspectral super-resolution, showing that low-rank modeling can effectively fuse multispectral and hyperspectral data for high-quality reconstruction. Ming *et al.* in [9] applied TriTD to link-load recovery in communication networks, achieving reasonable load estimation and underscoring its practical relevance for network management. Most recently, Thanh *et al.* in [11] proposed an adaptive TriTD variant for streaming tensors and demonstrated its effectiveness in video completion tasks. Despite these advances, existing TriTD methods primarily rely on squared-error objectives, which are highly sensitive to data corruption. Moreover, they still depend on large Kronecker matrix constructions, which significantly inflate memory usage and computational cost when handling large-scale tensors. These limitations motivate the development of a more efficient and robust TriTD method capable of handling corrupted and large-scale data.

Contributions: In this paper, we introduce TriTD-ADMM, a fast and robust method for triple tensor decomposition based on the alternating direction method of multipliers (ADMM). Our approach updates the core tensors via ridge-regularized least squares (RLS) applied to mode-wise unfoldings. To accelerate computation, we propose a reshape–permute acceleration strategy (RPAS) that avoids explicit Kronecker matrix construction and yields smaller Gram matrices for efficient

[♯] Quy Dang and Minh Nhat are co-first authors. Corresponding author: Thanh Trung Le (thanhletrung@vnu.edu.vn)

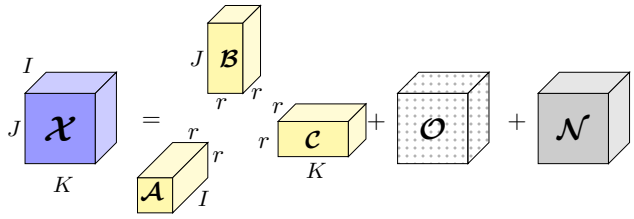

Fig. 1. Robust triple tensor decomposition with sparse corruption $\boldsymbol{\mathcal{O}}$ and an additive noise $\boldsymbol{\mathcal{N}}$.

RLS updates. To handle data corruption, we incorporate convex ℓ_1-norm regularization through soft-thresholding with lightweight dual updates. The ADMM framework, enhanced with adaptive penalty parameter updates, provides a simple yet effective solver for both primal and dual variable. Extensive experiments on eight real-world datasets demonstrate that TriTD-ADMM achieves competitive estimation accuracy while delivering strong runtime performance.

Notations: Scalars are written in lowercase letters (e.g., x), and vectors in bold lowercase (e.g., $\mathbf{x}$). Bold uppercase letters (e.g., $\mathbf{X}$) denote matrices, whereas bold calligraphic letters (e.g., $\boldsymbol{\mathcal{X}}$) represent tensors. The symbols $\circ$, $\odot$, and $\otimes$ indicate the outer, Hadamard, and Khatri-Rao products, respectively. Matrix transpose is indicated by $(\cdot)^\top$. The notation $\|\cdot\|$ is use to denote the norm of a vector, matrix, or tensor. The mode-n unfolding of $\mathbf{X}$ is written as $\boldsymbol{\mathcal{X}}_{(n)}$. For $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$, the operator $\mathrm{permute}(\boldsymbol{\mathcal{X}}, [\pi_1, \pi_2, \pi_3])$ reorders its modes according to a permutation, e.g., $\mathrm{permute}(\boldsymbol{\mathcal{X}}, [2, 3, 1])$ maps $(i, j, k) \mapsto (j, k, i)$. Similarly, $\mathrm{reshape}(\boldsymbol{\mathcal{Z}}, [d_1, d_2, \ldots, d_m])$ changes only the view of the data into dimensions $d_1 \times d_2 \times \cdots \times d_m$.

2. TRIPLE TENSOR DECOMPOSITION

Given a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$, its triple decomposition (TriTD) is defined as

$$\boldsymbol{\mathcal{X}} = \sum_{p=1}^{r}\sum_{q=1}^{r}\sum_{s=1}^{r} \boldsymbol{\mathcal{A}}_{:,p,q} \circ \boldsymbol{\mathcal{B}}_{s,:,q} \circ \boldsymbol{\mathcal{C}}_{s,p,:}, \tag{1}$$

where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I\times r\times r}$, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{r\times J\times r}$, and $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{r\times r\times K}$ are core tensors (loading factors) [7]. The smallest integer r that satisfies (1) is called the triple rank, denoted by $\mathrm{TriRank}(\boldsymbol{\mathcal{X}})$. For brevity, we express (1) as $\boldsymbol{\mathcal{X}} = \mathrm{TriTD}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}})$. The TriTD model also exhibits several useful properties.

Compactness: If $\boldsymbol{\mathcal{X}}$ admits the Tucker/HOSVD decomposition with Tucker rank $[r_1, r_2, r_3]$, then its triple rank is upper bounded as $\mathrm{TriRank}(\boldsymbol{\mathcal{X}}) \le \mathrm{middle}\{r_1, r_2, r_3\}$, i.e., no larger than the median of its Tucker rank parameters [7, Theorem 3.4]. Furthermore, CP/PARAFAC can be viewed as a special case of TriTD, and the triple rank also satisfies $\mathrm{TriRank}(\boldsymbol{\mathcal{X}}) \le \mathrm{CPRank}(\boldsymbol{\mathcal{X}})$ [7, Theorem 3.3]. Together, these bounds demonstrate the compactness of TriTD: it can provide a lower-dimensional representation than CP and Tucker. In particular, any dataset that admits a low-rank CP or Tucker approximation can likewise be well-approximated by a low-rank triple decomposition.

Matricization (Unfolding): Let $\mathbf{A} \triangleq \boldsymbol{\mathcal{A}}_{(1)} \in \mathbb{R}^{I\times r^2}$, $\mathbf{B} \triangleq \boldsymbol{\mathcal{B}}_{(2)} \in \mathbb{R}^{J\times r^2}$, and $\mathbf{C} \triangleq \boldsymbol{\mathcal{C}}_{(3)} \in \mathbb{R}^{K\times r^2}$. The unfolding matrices of $\boldsymbol{\mathcal{X}}$ can be expressed in the following bilinear forms:

$$\boldsymbol{\mathcal{X}}_{(1)} = \mathbf{AF}, \quad \boldsymbol{\mathcal{X}}_{(2)} = \mathbf{BG}, \text{ and } \boldsymbol{\mathcal{X}}_{(3)} = \mathbf{CH}, \tag{2}$$

where $\mathbf{F} \in \mathbb{R}^{r^2\times JK}$, $\mathbf{G} \in \mathbb{R}^{r^2\times IK}$ and $\mathbf{H} \in \mathbb{R}^{r^2\times IJ}$ are matrices of Kronecker structure. For brevity, we provide only the closed-form expression of $\mathbf{F}$:

$$\mathbf{F} = (\mathbf{I}_r \otimes \bar{\mathbf{B}})(\mathbf{C}^\top \otimes \mathbf{I}_r), \tag{3}$$

where $\bar{\mathbf{B}} \in \mathbb{R}^{r\times Jr}$ with element $\bar{\mathbf{B}}(r_2, j + (r_1 - 1)J) = \boldsymbol{\mathcal{B}}(r_1, j, r_2)$ for $1 \le r_1, r_2 \le r$ and $1 \le j \le J$. The closed-form expressions of $\mathbf{G}$ and $\mathbf{H}$ are analogous, see [8] for details. Thanks to (2), Qi *et al.* in [7] obtained an alternating least-squares (ALS) framework for computing $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, and $\boldsymbol{\mathcal{C}}$.

Alternating Least Squares (ALS): ALS offers an iterative scheme for estimating the core tensors using least-squares solvers. At each iteration, $\boldsymbol{\mathcal{A}}$ is updated as

$$\boldsymbol{\mathcal{A}}_{(1)} \leftarrow \left(\boldsymbol{\mathcal{X}}_{(1)}\mathbf{F}^\top\right)\left(\mathbf{F}\mathbf{F}^\top + \alpha\mathbf{I}_{r^2}\right)^{-1}, \tag{4}$$

where α is a small regularization parameter. The updates for $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$ are obtained analogously via $\mathbf{G}$ and $\mathbf{H}$ respectively. See [7] for their derivations and other ALS variants for TriTD.

3. PROPOSED METHOD

In this study, we investigate the robust TriTD in the presence of data corruption. Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ denote the observed tensor, which is contaminated by sparse corruption $\boldsymbol{\mathcal{O}}$ and a Gaussian noise $\boldsymbol{\mathcal{N}}$:

$$\boldsymbol{\mathcal{X}} = \mathrm{TriTD}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}) + \boldsymbol{\mathcal{O}} + \boldsymbol{\mathcal{N}}, \tag{5}$$

where $\boldsymbol{\mathcal{L}} = \mathrm{TriTD}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}})$ represent the structured low-rank component.

Optimization Framework: We propose to estimate components $\{\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{O}}\}$ by solving the following minimization

$$\begin{aligned} \underset{\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{C}},\boldsymbol{\mathcal{O}},\boldsymbol{\mathcal{E}}}{\operatorname{argmin}} \ & \lambda\|\boldsymbol{\mathcal{E}}\|_1 + \frac{\alpha_1}{2}\|\boldsymbol{\mathcal{A}}\|_F^2 + \frac{\alpha_2}{2}\|\boldsymbol{\mathcal{B}}\|_F^2 + \frac{\alpha_3}{2}\|\boldsymbol{\mathcal{C}}\|_F^2 \\ \text{subject to} \quad & \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{O}} \quad \text{and} \quad \boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{O}}, \end{aligned} \tag{6}$$

where λ and $\{\alpha_i\}_{i=1}^3$ are regularization parameters. The first term $\|\boldsymbol{\mathcal{E}}\|_1$ using ℓ_1-norm is to enforce the sparsity on the outlier component $\boldsymbol{\mathcal{E}}$. The last three terms using ℓ_2-norm are used as the regularization to improve stability and prevent ill-posed issues. The corresponding augmented Lagrangian is then given by

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{E}}; \boldsymbol{\mathcal{Y}}_L, \boldsymbol{\mathcal{Y}}_O) &= \lambda\|\boldsymbol{\mathcal{E}}\|_1 + \langle \boldsymbol{\mathcal{Y}}_O,\ \boldsymbol{\mathcal{E}} - \boldsymbol{\mathcal{O}}\rangle \\ &+ \frac{\mu_O}{2}\|\boldsymbol{\mathcal{E}} - \boldsymbol{\mathcal{O}}\|_F^2 + \frac{\alpha_1}{2}\|\boldsymbol{\mathcal{A}}\|_F^2 + \frac{\alpha_2}{2}\|\boldsymbol{\mathcal{B}}\|_F^2 + \frac{\alpha_3}{2}\|\boldsymbol{\mathcal{C}}\|_F^2 \\ &+ \langle \boldsymbol{\mathcal{Y}}_L,\ \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{O}}\rangle + \frac{\mu_L}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{O}}\|_F^2. \end{aligned} \tag{7}$$

Here, $\boldsymbol{\mathcal{Y}}_O$ and $\boldsymbol{\mathcal{Y}}_L$ denote the Lagrange multipliers (dual variables), while μ_O and μ_L are the corresponding penalty parameters. To find a stationary point of (7), we adopt the following procedure consisting of: (i) minimizing $\mathcal{L}(\cdot)$ with respect to each primal variable while keeping the others fixed, and (ii) updating the dual variables by maximizing $\mathcal{L}(\cdot)$. Our method,

Algorithm 1 Robust Triple Tensor Decomposition via ADMM (TriTD-ADMM)

Require: Tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$; target rank r; sparsity weight $\lambda > 0$; tolerance $\varepsilon > 0$; iterations MAXITER; parameters $\{\alpha_i\}_{i=1}^{3}$.

Ensure: Factors $(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}})$; sparse component $\boldsymbol{\mathcal{O}}$; auxiliary copy $\boldsymbol{\mathcal{E}}$

Operators: $\mathrm{TriTD}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}})$, $\mathrm{RPAS}(\cdot,\cdot)$ of two core tensors $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$ or $\boldsymbol{\mathcal{C}}$; and $\mathrm{soft}(\boldsymbol{\mathcal{X}}, \tau) = \mathrm{sign}(\boldsymbol{\mathcal{X}}) \odot \max(|\boldsymbol{\mathcal{X}}| - \tau, 0)$;

1: **Initialize:** $\boldsymbol{\mathcal{A}}^0, \boldsymbol{\mathcal{B}}^0, \boldsymbol{\mathcal{C}}^0$ (e.g., at random or SVD-based); $\boldsymbol{\mathcal{O}}^0 = \boldsymbol{\mathcal{E}}^0 = \mathbf{0}$; dual variables $\boldsymbol{\mathcal{Y}}_L^0 = \boldsymbol{\mathcal{Y}}_O^0 = \mathbf{0}$

2: Set penalties $\mu_L^0, \mu_O^0 > 0$, growth rates $\rho_L, \rho_O > 1$, caps $\mu_L^{\max}, \mu_O^{\max}$

3: **for** $\ell = 0$ **to** MAXITER **do**

Low-rank Update (ALS over TriTD factors)

4: $\quad \boldsymbol{\mathcal{T}}^{\ell} \leftarrow \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{O}}^{\ell} + \boldsymbol{\mathcal{Y}}_L^{\ell}/\mu_L^{\ell}$

5: $\quad \mathbf{F}^{\ell} \leftarrow \mathrm{RPAS}(\boldsymbol{\mathcal{B}}^{\ell}, \boldsymbol{\mathcal{C}}^{\ell})$

6: $\quad \mathbf{G}^{\ell} \leftarrow \mathrm{RPAS}(\boldsymbol{\mathcal{A}}^{\ell}, \boldsymbol{\mathcal{C}}^{\ell})$

7: $\quad \mathbf{H}^{\ell} \leftarrow \mathrm{RPAS}(\boldsymbol{\mathcal{A}}^{\ell}, \boldsymbol{\mathcal{B}}^{\ell})$

8: $\quad \mathbf{A}^{\ell+1} \leftarrow \big(\boldsymbol{\mathcal{T}}_{(1)}^{\ell}\,(\mathbf{F}^{\ell})^{\top}\big)\big(\mathbf{F}^{\ell}(\mathbf{F}^{\ell})^{\top} + \alpha_1 \mathbf{I}_{r^2}\big)^{-1}$

9: $\quad \mathbf{B}^{\ell+1} \leftarrow \big(\boldsymbol{\mathcal{T}}_{(2)}^{\ell}\,(\mathbf{G}^{\ell})^{\top}\big)\big(\mathbf{G}^{\ell}(\mathbf{G}^{\ell})^{\top} + \alpha_2 \mathbf{I}_{r^2}\big)^{-1}$

10: $\quad \mathbf{C}^{\ell+1} \leftarrow \big(\boldsymbol{\mathcal{T}}_{(3)}^{\ell}\,(\mathbf{H}^{\ell})^{\top}\big)\big(\mathbf{H}^{\ell}(\mathbf{H}^{\ell})^{\top} + \alpha_3 \mathbf{I}_{r^2}\big)^{-1}$

11: $\quad \boldsymbol{\mathcal{L}}^{\ell+1} \leftarrow \mathrm{TriTD}(\boldsymbol{\mathcal{A}}^{\ell+1}, \boldsymbol{\mathcal{B}}^{\ell+1}, \boldsymbol{\mathcal{C}}^{\ell+1})$

Sparse Update

12: $\quad \boldsymbol{\mathcal{R}}_1^{\ell} \leftarrow \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{L}}^{\ell+1} + \boldsymbol{\mathcal{Y}}_L^{\ell}/\mu_L^{\ell}, \quad \boldsymbol{\mathcal{R}}_2^{\ell} \leftarrow \boldsymbol{\mathcal{E}}^{\ell} - \boldsymbol{\mathcal{Y}}_O^{\ell}/\mu_O^{\ell}$

13: $\quad \boldsymbol{\mathcal{O}}^{\ell+1} \leftarrow \dfrac{\mu_L^{\ell}\boldsymbol{\mathcal{R}}_1^{\ell} + \mu_O^{\ell}\boldsymbol{\mathcal{R}}_2^{\ell}}{\mu_L^{\ell} + \mu_O^{\ell}}$

Auxiliary copy (proximal step)

14: $\quad \boldsymbol{\mathcal{R}}_3 \leftarrow \boldsymbol{\mathcal{O}}^{\ell+1} + \boldsymbol{\mathcal{Y}}_O^{\ell}/\mu_O^{\ell}$

15: $\quad \boldsymbol{\mathcal{E}}^{\ell+1} \leftarrow \mathrm{soft}(\boldsymbol{\mathcal{R}}_3, \lambda/\mu_O^{\ell})$

Dual Ascent

16: $\quad \boldsymbol{\mathcal{R}}_L^{\ell+1} \leftarrow \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{L}}^{\ell+1} - \boldsymbol{\mathcal{O}}^{\ell+1}, \quad \boldsymbol{\mathcal{R}}_O^{\ell+1} \leftarrow \boldsymbol{\mathcal{O}}^{\ell+1} - \boldsymbol{\mathcal{E}}^{\ell+1}$

17: $\quad \boldsymbol{\mathcal{Y}}_L^{\ell+1} \leftarrow \boldsymbol{\mathcal{Y}}_L^{\ell} + \mu_L^{\ell}\,\boldsymbol{\mathcal{R}}_L^{\ell+1}, \quad \boldsymbol{\mathcal{Y}}_O^{\ell+1} \leftarrow \boldsymbol{\mathcal{Y}}_O^{\ell} + \mu_O^{\ell}\,\boldsymbol{\mathcal{R}}_O^{\ell+1}$

Adaptive Penalty Update

18: $\quad \mu_L^{\ell+1} \leftarrow \min(\rho_L \mu_L^{\ell}, \mu_L^{\max})$

19: $\quad \mu_O^{\ell+1} \leftarrow \min(\rho_O \mu_O^{\ell}, \mu_O^{\max})$

Stopping Rule

20: $\quad \mathrm{err}^{\ell+1} \leftarrow \|\boldsymbol{\mathcal{R}}_O^{\ell+1}\|_F + \|\boldsymbol{\mathcal{R}}_L^{\ell+1}\|_F$

21: $\quad$ **if** $\mathrm{err}^{\ell+1} \le \varepsilon$ **then break**

22: **return** $\{\boldsymbol{\mathcal{A}}^{\ell+1}, \boldsymbol{\mathcal{B}}^{\ell+1}, \boldsymbol{\mathcal{C}}^{\ell+1}; \boldsymbol{\mathcal{O}}^{\ell+1}\}$

called TriTD-ADMM, is summarized in Algorithm 1.

Reshape–Permute Acceleration Strategy (RPAS): In Algorithm 1, the most computational cost come from the low-rank update stage, which involves solving ridge least-squares problems (see lines 8, 9, and 10). This requires constructing a design matrix $\{\mathbf{F}, \mathbf{G}, \mathbf{H}\}$ from two of the three cores. Using the standard formulation (3) to compute them directly is costly, as it involves two Kronecker products followed by a large matrix–matrix multiplication.

To alleviate this bottleneck, we introduce the Reshape Permute Acceleration Strategy (RPAS). Here, we retain the same ridge least-squares subproblems, $\mathbf{Z}^{\ell+1} = (\boldsymbol{\mathcal{T}}_{(i)}^{\ell}(\mathbf{M}^{\ell})^{\top})\,(\mathbf{M}^{\ell}(\mathbf{M}^{\ell})^{\top} + \alpha\,\mathbf{I}_{r^2})^{-1}$, where $(\mathbf{Z}, \mathbf{M}, i) \in \{(\mathbf{A}, \mathbf{F}, 1), (\mathbf{B}, \mathbf{G}, 2), (\mathbf{C}, \mathbf{H}, 3)\}$. However, we compute $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{H}$ in a *Kronecker-free* manner using only `reshape`/`permute` operators and a single general matrix multiplication per mode. Specifically, in mode 1, the matrix $\mathbf{F}$ can be efficiently computed from $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$ as

$$\mathbf{F} = \mathrm{reshape}\Big(\,\mathrm{reshape}\big(\,\mathrm{permute}(\boldsymbol{\mathcal{B}}, [2,3,1]), [Jr,\, r]\big) \times \mathrm{reshape}\big(\boldsymbol{\mathcal{C}}, [r,\, rK]\big), [r^2, JK]\Big), \quad (8)$$

For short, we denote (8) as $\mathbf{F} = \mathrm{RPAS}(\boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}})$. Similarly, in mode 2 and mode 3, we also obtain $\mathbf{G} = \mathrm{RPAS}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{C}})$ and $\mathbf{H} = \mathrm{RPAS}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}})$. With this approach, the cost of constructing $\mathbf{F}, \mathbf{G}$, and $\mathbf{H}$ is reduced from $\mathcal{O}(n^3 r^4)$ flops to $\mathcal{O}(n^2 r^3)$ flops, assuming $I = J = K = n$. Thus, the overall computational complexity of TriTD-ADMM is then given as follows.

Computational complexity: At each iteration, TriTD-ADMM updates loading factors $\{\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}\}$ via: (i) constructing a mode-specific design $\mathbf{M}$ (i.e., $\mathbf{M}$ can be $\mathbf{F}, \mathbf{G}, \mathbf{H}$) using RPAS which costs $\mathcal{O}(n^2 r^3)$ flops; (ii) computing a data–design product $\boldsymbol{\mathcal{T}}_{(i)}\mathbf{M}^{\top}$ with complexity $\mathcal{O}(n^3 r^2)$ flops; and (iii) solving a small ridge system with Gram matrix $\mathbf{M}\mathbf{M}^{\top}$ of size $r^2 \times r^2$, requiring $\mathcal{O}(n^2 r^4 + r^6)$ flops. The dual variable updates require an additional $\mathcal{O}(n^3)$ flops. Accordingly, the total computational complexity of TriTD-ADMM is $\mathcal{O}(3\,n^3 r^2 + 3n^2 r^4 + 3r^6)$ flops for each iteration.

Convergence Analysis: The following lemma establishes the convergence of TriTD-ADMM. Due to the space limit, we omit the proof here and will provide it in the extended journal.

Lemma 1 *Denote* $\mathcal{S}^{(t)} = \{\boldsymbol{\mathcal{X}}^{(t)}, \boldsymbol{\mathcal{A}}^{(t)}, \boldsymbol{\mathcal{B}}^{(t)}, \boldsymbol{\mathcal{C}}^{(t)}, \boldsymbol{\mathcal{O}}^{(t)}, \boldsymbol{\mathcal{E}}^{(t)}, \boldsymbol{\mathcal{Y}}_L^{(t)}, \boldsymbol{\mathcal{Y}}_O^{(t)}\}_{t\in\mathbb{N}}$ *the solution generated by Algorithm 1 at each iteration t. When* $t \to \infty$*, the sequence* $\{\mathcal{S}^{(t)}\}_{t=1}^{\infty}$ *converges almost surely to a stationary point of* (7).

4. EXPERIMENTS

In this section, we conduct several numerical experiments on real-world datasets to demonstrate the efficacy of TriTD-ADMM, in comparison with state-of-the-art tensor methods.[1]

Real-World Datasets: In this work, we investigate two tasks with real datasets: tensor completion and video background modeling. For tensor completion, we evaluate performance on four datasets with diverse spatiotemporal structures: `sensor` (52×4×2994),[2] `taxi` (265×265×500),[3] `network` (23×23× 1152),[4] and `chicago` (77×77×2016).[5] The evaluation metrics are run time (wall-clock time) and relative reconstruction error, defined as

$$\mathrm{RRE} = \|\boldsymbol{\mathcal{X}}_{\text{true}} - \boldsymbol{\mathcal{X}}_{\text{recover}}\|_F / \|\boldsymbol{\mathcal{X}}_{\text{true}}\|_F. \quad (9)$$

For the video background modeling, we use four video sequences from the CDnet 2014 dataset: `Highway` (240×320

[1] Our codes are available at `https://github.com/dangnq2501/Triple-Tensor-Decomposition-with-ADMM`

[2] sensor data: `https://db.csail.mit.edu/labdata/labdata.html`

[3] taxi data: `https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page`

[4] network data: `https://www.cs.utexas.edu/ yzhang/research/AbileneTM/`

[5] chicago data: `https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew`

Table 1. Comparison of tensor-based methods for tensor completion.

Dataset	sensor		taxi		network		chicago	
Tensor size	$52 \times 4 \times 2994$		$265 \times 265 \times 500$		$23 \times 23 \times 1152$		$77 \times 77 \times 2016$	
Evaluation metric	RRE	Time(s)	RRE	Time(s)	RRE	Time(s)	RRE	Time(s)
Sofia	0.341	15.95	0.584	598.24	0.963	12.01	0.352	194.36
TRLRF	0.316	25.58	**0.280**	1799.52	**0.126**	41.06	0.311	1318.22
RC-FCTN	0.337	**2.46**	0.380	128.44	1.083	5.08	**0.247**	29.30
TTNN	0.558	4.45	0.307	340.42	0.999	7.39	0.316	264.73
TriTD-ADMM (Ours)	**0.279**	2.53	0.338	**53.90**	0.143	**1.72**	0.321	**20.69**

$\times$300), Office (240$\times$320$\times$300), PETS2006 (240$\times$360 $\times$300), and Sofa (240$\times$360$\times$300).[6] Here, we select only 300 consecutive frames from each video sequence to construct the tensor $\mathcal{X}$.

We compare the performance of TriTD-ADMM against several strong baselines, including Sofia [14], TRLRF [15], RC-FCTN [16] and TTNN [17]. Their algorithm parameters were fine-tuned to achieve their best performance.

Experimental Results: Our results are presented statistically in Tab. 1 and graphically in Fig. 2. Tab. 1 summarizes the completion performance of tensor-based methods with 10% missing observations. In this setting, our method treated the missing elements as corrupted data and we fixed the triple rank to 5 for all datasets. We can see that, TriTD-ADMM consistently achieves the lowest or near-lowest runtime across all four datasets while maintaining competitive completion accuracy. Particularly on the sensor dataset (small scale), TriTD-ADMM attains the most accurate completion with a near-minimum runtime of 2.53 s, within 3% of the fastest method, RC-FCTN (2.46 s), and significantly faster than Sofia (15.95 s, **6.30**$\times$), TRLRF (25.58 s, **10.11**$\times$) and TTNN (4.45 s, **1.76**$\times$). On the taxi dataset (large scale), TriTD-ADMM is the fastest method (53.90 s) with reasonable RRE (0.338), outperforming Sofia (598.24 s, **11.10**$\times$), TRLRF (1799.52 s, **33.39**$\times$), TTNN (340.42 s, **6.32**$\times$), and RC-FCTN (128.44 s, **2.38**$\times$). On the network and chicago datasets (moderate scale), TriTD-ADMM maintains competitive estimation accuracy while consistently ranking among the fastest methods.

Fig. 2 illustrates the video background modeling on four BMC sequences (Highway, Office, PETS2006, and Sofa). In this task, our method represents the video background as the low-rank component $\mathcal{L}$ and the foreground objects as the sparse component $\mathcal{O}$, following the data model in (5). As shown in the last column of Fig. 2, TriTD-ADMM (Triple) effectively separates the video foreground from the background across all four sequences. TTNN and TRLRF perform reasonably well when the video frames contain fewer moving objects (e.g., PETS2006), but they tend to absorb moving objects into the background in more dynamic scenes such as Highway. RC-FCTN and Sofia perform poorly, failing to achieve reliable video background. With respect to run time, TriTD-ADMM is the fastest method in all video sequences, with run times of 33.68 s (Highway), 37.05 s (Sofa), 43.98 s (Office), and 35.93 s (PETS2006). Our method is faster than **1.29**$-$**2.58** times than RC-FCTN (50.64 s$-$92.62 s), **5.15**$-$**6.38** times than TTNN (201.47 s$-$229.23 s), **26.1**$-$**33.8** times than TRLRF (1031.97 s$-$1215.11 s), and **9.64**$-$**11.32** times than Sofia (370.57 s$-$424.15 s). These results highlight the efficacy of our method in this task.

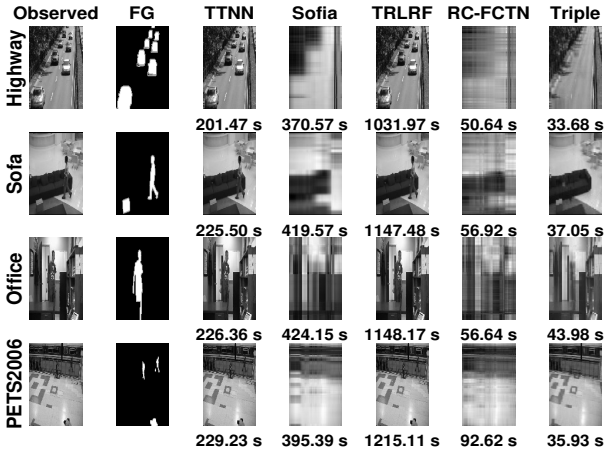

Fig. 2. Video background modeling on four BMC sequences with running time (s). *FG denotes video foreground.*

5. CONCLUSIONS AND FUTURE WORKS

We proposed a fast and robust triple tensor decomposition method via ADMM, designed to effectively handle data corruption. Our method achieves state-of-the-art performance in tensor completion and video background modeling, while achieving substantial computational speedups through the RPAS acceleration strategy and efficient ADMM optimzation framework. Experiments on real-world datasets demonstrate that our method achieves a good balance between accuracy and efficiency, particularly on moderate and large scale datasets. Nonetheless, the current solver requires a fixed triple rank, which can limit performance when low-rank components vary rapidly. Promising directions for future work include: (i) automatic or adaptive rank selection, (ii) stochastic or online ADMM for streaming data, and (iii) system-level optimizations such as GPU-friendly reshape–permute kernels and batched Cholesky/PCG solvers for RLS updates.

[6] CDnet 2014 data: http://jacarini.dinf.usherbrooke.ca/

6. REFERENCES

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

[2] Y. Panagakis, J. Kossaifi, G. G. Chrysos, J. Oldfield, M. A. Nicolaou, A. Anandkumar, and S. Zafeiriou, "Tensor methods in computer vision and deep learning," *Proc. IEEE*, vol. 109, no. 5, pp. 863–890, 2021.

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

[4] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "A contemporary and comprehensive survey on streaming tensor decomposition," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 11, pp. 10897–10921, 2023.

[5] M. Wang, D. Hong, Z. Han, J. Li, J. Yao, L. Gao, B. Zhang, and J. Chanussot, "Tensor decompositions for hyperspectral data processing in remote sensing: A comprehensive review," *IEEE Geosci. Remote Sens. Mag*, vol. 11, no. 1, pp. 26–72, 2023.

[6] N. Tokcan, S. S. Sofi, C. Prévost, S. Kharbech, B. Magnier, T. P. Nguyen, Y. Zniyed, and L. De Lathauwer, "Tensor decompositions for signal processing: Theory, advances, and applications," *Signal Process.*, vol. 238, p. 110191, 2025.

[7] L. Qi, Y. Chen, M. Bakshi, and X. Zhang, "The triple decomposition and tensor recovery of third order tensors," *SIAM J. Matrix Anal. Appl.*, vol. 42, no. 1, pp. 299–329, 2021.

[8] Y. Chen, X. Zhang, L. Qi, and Y. Xu, "A Barzilai–Borwein gradient algorithm for spatio-temporal internet traffic data completion via tensor triple decomposition," *J. Sci. Comput.*, vol. 88, no. 3, p. 65, 2021.

[9] Z. Ming, Z. Qin, L. Zhang, Y. Xu, and L. Qi, "Network traffic recovery from link-load measurements using tensor triple decomposition strategy for third-order traffic tensors," *J. Comput. Appl. Math.*, vol. 447, p. 115901, 2024.

[10] X. Cui, R. Dong, and J. Li, "Hyperspectral super-resolution via low rank tensor triple decomposition," *J. Ind. Manag. Optim.*, vol. 20, no. 3, pp. 942–966, 2024.

[11] T. T. Le, N. H. Thinh, L. M. Ha, T. T. T. Quynh, L. V. Ha, V.-T.-L. Hoang, K. Abed-Meraim, P. Ravier, and O. Buttelli, "Low-rank triple decomposition of streaming tensors and its application to video completion," in *Proc. 24th Int. Symp. Commun. Inf. Technol.*, pp. 97–1023, 2025.

[12] F. Wu, C. Li, and Y. Li, "Manifold regularization nonnegative triple decomposition of tensor sets for image compression and representation," *J. Optim. Theory Appl.*, vol. 192, no. 3, pp. 979–1000, 2022.

[13] Q. Liao, Q. Liu, and F. A. Razak, "Hypergraph regularized nonnegative triple decomposition for multiway data analysis," *Sci. Rep.*, vol. 14, no. 1, p. 9098, 2024.

[14] D. Lee and K. Shin, "Robust factorization of real-world tensor streams with patterns, missing values, and outliers," in *Proc. 37th IEEE Int. Conf. Data Eng.*, pp. 840–851, 2021.

[15] L. Yuan, C. Li, J. Cao, and Q. Zhao, "Rank minimization on tensor ring: An efficient approach for tensor decomposition and completion," *Mach. Learn.*, vol. 109, no. 3, pp. 603–622, 2020.

[16] Y.-Y. Liu, X.-L. Zhao, G.-J. Song, Y.-B. Zheng, and T.-Z. Huang, "Fully-connected tensor network decomposition for robust tensor completion," *Inverse Probl. Imaging.*, vol. 18, no. 1, pp. 208–238, 2024.

[17] J.-H. Yang, X.-L. Zhao, T.-Y. Ji, T.-H. Ma, and T.-Z. Huang, "Low-rank tensor train for tensor robust principal component analysis," *Appl. Math. Comput.*, vol. 367, p. 124783, 2020.