axioms

MDPI

Article

Efficient Tensor Robust Principal Analysis via Right-Invertible Matrix-Based Tensor Products

Zhang Huang *, Jun Feng and Wei Li

School of Mathematical Sciences, Chengdu University of Technology, Chengdu 610059, China; fengjun@cdut.edu.cn (J.F.); lovelylw@126.com (W.L.)

* Correspondence: huangzhang16@cdut.edu.cn

Abstract: In this paper, we extend the definition of tensor products from using an invertible matrix to utilising right-invertible matrices, exploring the algebraic properties of these new tensor products. Based on this novel definition, we define the concepts of tensor rank and tensor nuclear norm, ensuring consistency with their matrix counterparts, and derive a singular value thresholding ($*_{L,R}$ SVT) formula to approximately solve the subproblems in the alternating direction method of multipliers (ADMM), which is integral to our proposed tensor robust principal component analysis ($*_{LR}$ TRPCA) algorithm. The computational complexity of the $*_{LR}$ TRPCA algorithm is $O(k \cdot (n_1 n_2 n_3 + p \cdot \min(n_1^2 n_2, n_1 n_2^2)))$ for k iterations. According to this complexity analysis, by using a right-invertible matrix that selects p rows from the n_3 rows of the invertible matrix used in the tensor product with an invertible matrix, the computational load is approximately reduced to p/n_3 of what it would be with an invertible matrix, highlighting the efficiency gain in terms of computational resources. We apply this efficient algorithm to grayscale video denoising and motion detection problems, where it demonstrates significant improvements in processing speed while maintaining comparable quality levels to existing methods, thereby providing a promising approach for handling multi-linear data and offering valuable insights for advanced data analysis tasks.

Keywords: tensor robust principal component analysis (TRPCA); right-invertible matrices; tensor nuclear norm; image processing

MSC: 15A69; 62H25; 65F20; 65F50

check for updates

Academic Editor: Jorge Delgado Gracia

Received: 16 December 2024
Revised: 14 January 2025
Accepted: 23 January 2025
Published: 28 January 2025

Citation: Huang, Z.; Feng, J.; Li, W. Efficient Tensor Robust Principal Analysis via Right-Invertible Matrix-Based Tensor Products. *Axioms* **2025**, *14*, 99. https://doi.org/10.3390/axioms14020099

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In this paper, we advance tensor robust principal component analysis (TRPCA) with several key contributions. First, we introduce new tensor products for third-order tensors using right-invertible operators, generalizing existing methods like the t-product and c-product. Second, we propose TRPCA models that enhance computational efficiency and robustness through these new products and the tensor nuclear norm. Third, we provide theoretical guarantees for recovering low-rank and sparse components accurately. Finally, our numerical experiments show superior performance in applications such as grayscale video recovery and motion detection compared to c-product-based methods.

Principal component analysis (PCA) is a fundamental technique in data analysis that is widely used for dimensionality reduction and feature extraction. It is particularly effective in uncovering low-dimensional structures in high-dimensional data, which are prevalent in various fields, such as images, text, videos, and bioinformatics. PCA simplifies the complexity of high-dimensional data while retaining trends and patterns by transforming

the data into fewer dimensions, which act as summaries of features. This type of data presents several challenges that PCA mitigates: computational expense and an increased error rate due to multiple test corrections when testing each feature for association with an outcome. Despite its computational efficiency and robustness to minor noise, PCA is highly sensitive to gross corruptions or outliers, which are common in real-world datasets. Recent advancements in robust PCA have addressed these limitations, providing more accurate and stable results in the presence of outliers [1–6].

To address this limitation, numerous robust variants of PCA have been developed, but many suffer from high computational costs [7–9]. Among these robust variants, robust principal component analysis (RPCA) stands out as a significant advancement [1]. RPCA is the first polynomial-time algorithm with strong recovery guarantees [10]. Given an observed matrix $X \in \mathbb{R}^{n_1 \times n_2}$, which can be decomposed as $X = L_0 + E_0$, where L_0 is low-rank and E_0 is sparse, RPCA demonstrates that if the singular vectors of L_0 meet certain incoherence conditions, L_0 and E_0 can be accurately recovered with high probability by solving the following convex optimisation problem:

$$\min_{L,E} \|L\|_* + \lambda \|E\|_1, \quad \text{s.t. } X = L + E. \tag{1}$$

Here, $\|L\|_*$ represents the nuclear norm (the sum of the singular values of L), and $\|E\|_1$ denotes the ℓ_1 norm (the sum of the absolute values of all entries in E) [1,3]. The parameter λ is typically set to $1/\sqrt{\max(n_1, n_2)}$, which performs well in practice [1]. Algorithmically, this problem can be efficiently solved at a cost comparable to that of PCA [7,9]. Recent research has focused on improving the efficiency and robustness of RPCA, with novel approaches such as nonconvex RPCA (N-TRPCA) and learned robust PCA (LRPCA) showing promising results [6,11].

However, a primary limitation of RPCA is its applicability only to two-way (matrix) data. Real-world data, especially in modern applications, are often multidimensional and represented as multiway arrays or tensors [12,13]. For example, a colour image is a three-way object with column, row, and colour modes, whereas a grayscale video is indexed by two spatial variables and one temporal variable [12]. To apply RPCA, one must first reshape the multiway data into a matrix, a process that often leads to information loss and performance degradation [3,14]. This limitation has prompted researchers to extend RPCA to handle tensor data directly, leveraging its multidimensional structure [6,15]. Such extensions aim to preserve the inherent structures of the data and avoid the pitfalls associated with reshaping, thus improving the robustness and accuracy of RPCA in handling high-dimensional data [16,17].

To address these issues, recent research has focused on extending robust principal component analysis (RPCA) to handle tensor data directly, leveraging its multidimensional structure. One notable approach is tensor robust principal component analysis (TRPCA), which aims to precisely recover a low-rank tensor corrupted by sparse errors [3,15]. Lu et al. [15] introduced a new tensor nuclear norm based on the tensor–tensor product (t-product) [18], which generalizes the matrix–matrix product. This new tensor nuclear norm serves as the convex envelope of the tensor average rank within the unit ball of the tensor spectral norm, providing a strong theoretical foundation for TRPCA [15]. Building on this, many authors have improved upon TRPCA algorithms based on the tensor–tensor product (t-product). For instance, Zhu et al. [19] proposed a fast TRPCA method that estimates the multirank vector and adopts a Riemannian optimization strategy to more efficiently recover low-rank and sparse tensors. Furthermore, Cai et al. [20] developed a generalized framework for estimating a latent low-rank plus sparse tensor using fast Riemannian optimization, which can simultaneously estimate both the low-rank and sparse

components under suitable conditions, enhancing recovery performance while reducing computational costs.

These authors have constructed and improved TRPCA algorithms based on the tensor t-product, with applications in image denoising, hyperspectral data analysis, medical imaging, and other fields. All of these methods fall under the category of TRPCA techniques defined by tensor products based on invertible matrices. To the best of our knowledge, there has been no exploration of TRPCA algorithms utilizing tensor products defined using non-invertible matrices.

Unlike previous TRPCA algorithms that rely on tensor–tensor products (t-products) and other tensor products defined by invertible matrices, our approach introduces new tensor products for third-order tensors through right-invertible matrices. These new tensor products generalize the concept of tensor products defined by invertible linear transforms. Based on the new defined tensor product, we propose a novel TRPCA model that utilises a new concept of tensor products defined via right-invertible linear operators and offer several advantages in terms of computational efficiency and robustness.The primary aim of our current work is to validate the feasibility of TRPCA based on these new tensor products. Further improvements to achieve even greater computational efficiency will be explored in subsequent research.

Our contributions are summarised as follows:

1. Motivated by the new tensor product in Reference [21], which is a natural generalisation of the tensor product with invertible linear transforms, such as the t-product and c-product, we rigorously deduce a new class of tensor products for third-order tensors via right-invertible linear operators. These new tensor products generalise existing tensor product definitions and preserve many fundamental properties of the tensor product defined by invertible linear transforms. This forms the foundation for extending models, optimisation methods, and theoretical analysis techniques from t-products to new products with the right-invertible linear operators.
2. Novel TRPCA models that leverage new tensor products to increase computational efficiency and robustness are given. Equipped with the tensor nuclear norm, this model provides a more efficient and robust solution for tensor data.
3. We provide theoretical guarantees for the exact recovery of low-rank and sparse components via the new tensor nuclear norm. Under certain assumption conditions, the solution to the convex TRPCA model perfectly recovers the underlying low-rank component L_0 and sparse component E_0. The recovery guarantees of RPCA [1] fall into our special cases.
4. Numerical experiments are presented to validate the efficiency and effectiveness of the proposed algorithms in various applications, including grayscale video recovery and motion detection. Our experiments show the superiority of the new TRPCA over TRPCA on the basis of the c-product.

This paper is organised as follows: Section 2 introduces the necessary background and notation. Section 3 defines the new tensor products and explores their properties. Section 4 presents the TRPCA model and the theoretical guarantees. Section 5 provides numerical experiments and applications. Finally, Section 6 concludes the paper and discusses future directions.

2. Preliminaries

2.1. Notations

Throughout this paper, vectors are denoted by italic lowercase letters such as $\vec{u}$ and $\vec{v}$, matrices by uppercase letters such as A and B, and tensors by calligraphic capital letters such as $\mathscr{A}$ and $\mathscr{B}$. Scalars are denoted by lowercase letters, such as a. The notation

$\mathbb{F}^{n_1\times n_2\times n_3}$ represents the set of $n_1 \times n_2 \times n_3$ tensors over the field $\mathbb{F}$, where $\mathbb{F}$ can be either the field of real numbers $\mathbb{R}$ or the field of complex numbers $\mathbb{C}$. This notation can also be written as $\mathbb{F}_{n_3}^{n_1\times n_2}$ when emphasising two particular dimensions.

For a third-order tensor $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1\times n_2}$, the (i,j,k)-th entry is denoted as $\mathscr{A}_{ijk}$ or a_{ijk}. Let $\mathbf{a}_{ij}$ represent the i,j-th tube of $\mathscr{A}$, that is, $\mathbf{a}_{ij} = \mathscr{A}_{i_1,i_2,:}$. A slice of a tensor is a two-dimensional section defined by fixing all but two indices. The i_3-th frontal slice is denoted as $\mathscr{A}^{(i_3)}$, the i_1-th horizontal slice is denoted as $\mathscr{A}_{i_1,:,:}$, and the i_2-th lateral slice is denoted as $\mathscr{A}_{:,i_2,:}$.

The inner product between matrices A and B in $\mathbb{F}^{n_1\times n_2}$ is defined as $\langle A, B\rangle = \mathrm{Tr}(A^*B)$, where A^* denotes the conjugate transpose of A and $\mathrm{Tr}(\cdot)$ denotes the matrix trace. The inner product between tensors $\mathscr{A}$ and $\mathscr{B}$ in $\mathbb{F}_{n_3}^{n_1\times n_2}$ is defined as $\langle \mathscr{A}, \mathscr{B}\rangle = \sum_{i_3=1}^{n_3}\langle \mathscr{A}^{(i_3)}, \mathscr{B}^{(i_3)}\rangle$. Several norms are used for vectors, matrices, and tensors. The ℓ_1 norm of a tensor $\mathscr{A}$ is defined as $\|\mathscr{A}\|_1 = \sum_{i_1,i_2,i_3} |a_{i_1i_2i_3}|$. The Frobenius norm is defined as $\|\mathscr{A}\|_F = \sqrt{\sum_{i_1,i_2,i_3} |a_{i_1i_2i_3}|^2}$. These norms reduce to the corresponding vector or matrix norms when $\mathscr{A}$ is a vector or a matrix. For a vector $\vec{v} \in \mathbb{F}^n$, the ℓ_2 norm is defined as $\|\vec{v}\|_2 = \sqrt{\sum_i |v_i|^2}$. The matrix nuclear norm is defined as $\|A\|_* = \sum_i \sigma_i(A)$.

2.2. Right-Invertible Operator

Consider a matrix M of size $p \times n_3$ with $p \le n_3$. If M is a right-invertible matrix, then there exists a matrix N of size $n_3 \times p$ such that $M_{p\times n_3}N_{n_3\times p} = I_{p\times p}$. When $p = n$, M, as a right-invertible matrix, is an invertible matrix. A right-invertible matrix is a matrix with linearly independent rows. One way to construct a right-invertible matrix is by removing some rows from an invertible matrix. Let us consider an $n \times n$ discrete cosine transform (DCT) matrix C. Suppose that we want to form a new matrix C' of size $(n-m)\times n$ by removing the last m rows from the original matrix C, where $m \le n$. This operation is equivalent to truncating the matrix. The resulting matrix C' will no longer be square and thus does not have an inverse effect in the traditional sense. However, we can find a right inverse G such that $C'_{(n-m)\times n}G_{n\times(n-m)} = I_{n-m}$ where I_{n-m} is the $(n-m)\times(n-m)$ identity matrix. In general, the right inverse of a matrix is not unique. To construct G, we can use the Moore–Penrose pseudoinverse. For a given matrix A, its pseudoinverse A^+ is defined as $A^+ = A^\dagger(AA^\dagger)^{-1}$ provided that $A^\dagger A$ is invertible, where $A^\dagger$ denotes the conjugate transpose of A. Therefore, the right inverse G of C' is given by: $G = C'^\dagger(C'C'^\dagger)^{-1}$. Thus, we form a right-invertible pair (C', G).

Now, we can define two functions to convert between tube tensors $\mathbb{F}_{n_3}^{1\times 1}$ and vectors in $\mathbb{F}^{n_3}$. The function vec maps a tube tensor $\mathscr{A} \in \mathbb{F}_{n_3}^{1\times 1}$ to a vector in $\mathbb{F}^{n_3}$, defined as:

$$\mathrm{vec} : \mathbb{F}_{n_3}^{1\times 1} \to \mathbb{F}^{n_3}, \quad \mathscr{A} \mapsto \mathrm{vec}(\mathscr{A}),$$

where $\mathrm{vec}(\mathscr{A})$ is the vector obtained by stacking the elements of $\mathscr{A}$ into a column vector. The function unvec maps a vector $\vec{v} \in \mathbb{F}^{n_3}$ to a tube tensor $\mathscr{A} \in \mathbb{F}_{n_3}^{1\times 1}$, defined as:

$$\mathrm{unvec} : \mathbb{F}^{n_3} \to \mathbb{F}_{n_3}^{1\times 1}, \quad \vec{v} \mapsto \mathrm{unvec}(\vec{v}),$$

where $\mathrm{unvec}(\vec{v})$ is the tube tensor obtained by reshaping the vector $\vec{v}$ into a $\mathbb{F}_{n_3}^{1\times 1}$ tensor.

Definition 1. *The mode-3 (matrix) product between $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1\times n_2}$ and an $p \times n_3$ matrix U is defined as*

$$(\mathscr{A} \times_3 U)_{ijk} = \sum_{s=1}^{n_3} \mathscr{A}_{ijs}u_{ks}, \quad i = 1,\dots,n_1,\ j = 1,\dots,n_2,\ k = 1,\dots,p.$$

Definition 2. *Let M be a right-invertible matrix of size $p \times n_3$. We define the right-invertible operator L on tube tensors as*

$$L : \mathbb{F}_{n_3}^{1\times 1} \to \mathbb{F}_{p}^{1\times 1}, \quad \mathscr{A} \mapsto \mathscr{A} \times_3 M$$

and its inverse as

$$R : \mathbb{F}_{p}^{1\times 1} \to \mathbb{F}_{n_3}^{1\times 1}, \quad \mathscr{B} \mapsto \mathscr{B} \times_3 N$$

where N is the right inverse matrix of M.

Using the functions vec and unvec, we can also express the operators L and R in terms of vector operations

$$L : \mathbb{F}_{n_3}^{1\times 1} \to \mathbb{F}_{p}^{1\times 1}, \quad \mathscr{A} \mapsto unvec(M \cdot vec(\mathscr{A}))$$

$$R : \mathbb{F}_{p}^{1\times 1} \to \mathbb{F}_{n_3}^{1\times 1}, \quad \mathscr{B} \mapsto unvec(N \cdot vec(\mathscr{B}))$$

It can be seen that $R(L(\mathscr{A}))$ is not equal to the original tensor $\mathscr{A}$ in general because NM is not an identity matrix, although MN is an identity matrix.

3. Definition and Properties of the New Tensor Product

Definition 3 (Face-wise Product [22])**.** *Let $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1\times k}$ and $\mathscr{B} \in \mathbb{F}_{n_3}^{k\times n_2}$ be two third-order tensors. The face-wise product between $\mathscr{A}$ and $\mathscr{B}$ is defined as:*

$$(\mathscr{A} \triangle \mathscr{B})^{(i_3)} = \mathscr{A}^{(i_3)} \mathscr{B}^{(i_3)}, \quad i_3 = 1, \ldots, n_3.$$

Here, $\mathscr{A}^{(i_3)}$ and $\mathscr{B}^{(i_3)}$ denote the i_3-th frontal slices of $\mathscr{A}$ and $\mathscr{B}$, respectively, which are matrices of sizes $n_1 \times k$ and $k \times n_2$. The product $\mathscr{A}^{(i_3)}\mathscr{B}^{(i_3)}$ is the standard matrix multiplication.

Definition 4. *Let M be a right-invertible matrix of size $p \times n_3$. We define the right-invertible operator L on two third-order tensors as*

$$L : \mathbb{F}_{n_3}^{n_1\times n_2} \to \mathbb{F}_{p}^{n_1\times n_2}, \quad \mathscr{A} \mapsto \mathscr{A} \times_3 M$$

and its inverse as

$$R : \mathbb{F}_{p}^{n_1\times n_2} \to \mathbb{F}_{n_3}^{n_1\times n_2}, \quad \mathscr{B} \mapsto \mathscr{B} \times_3 N$$

where N is the right inverse matrix of M.

In the following, if no confusion arises, we do not distinguish the right-invertible operator L defined by the same matrix M on different tensor spaces.

Proposition 1. *The operator L is surjective (onto), the operator R is injective (one-to-one), and LR is the identity mapping.*

This proposition establishes the fundamental properties of the operators L and R. The surjectivity of L and injectivity of R ensure the mapping properties required for consistency and invertibility. These properties are important in proving the associativity of new tensor products. The proof of this proposition can be found in Appendix A.1.

Definition 5. *Let L be the right-invertible operator between $\mathbb{F}_{n_3}^{n_1\times n_2}$ and $\mathbb{F}_{p}^{n_1\times n_2}$, and R be its inverse between $\mathbb{F}_{p}^{n_1\times n_2}$ and $\mathbb{F}_{n_3}^{n_1\times n_2}$. We define the kernel of the right-invertible operator L as*

$$\begin{aligned}\ker(L) &= \{\mathscr{A} \in \mathbb{F}_{n_3}^{n_1\times n_2} \mid L(\mathscr{A}) = 0 \in \mathbb{F}_{p}^{n_1\times n_2}\} \\ &= \{\mathscr{A} \in \mathbb{F}_{n_3}^{n_1\times n_2} \mid \mathscr{A} \times_3 M = 0 \in \mathbb{F}_{p}^{n_1\times n_2}\}\end{aligned}$$

Proposition 2. *Let L and R be defined as above; then, we have*

$$\ker(L) = \{\mathscr{B} \mid \exists \mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2} \text{ such that } \mathscr{B} = (I - RL)(\mathscr{A})\}$$

where I is the identity map on $\mathbb{F}_{n_3}^{n_1 \times n_2}$.

This proposition characterizes the kernel of the operator L, providing essential insights into the null space of the operator, and its proof is detailed in Appendix A.2. It plays a crucial role in ensuring that our tensor robust principal component analysis (TRPCA) model can accurately decompose tensors into low-rank and sparse components by identifying elements that are mapped to zero by L. Through the kernel space, we can also assess the decomposition error in TRPCA, providing insights into cases where the decomposition might not be exact.

Using the functions *vec* and *unvec*, we can give an equivalent description of the operators L and R in terms of tube tensors

$$L : \mathbb{F}_{n_3}^{n_1 \times n_2} \to \mathbb{F}_{p}^{n_1 \times n_2}, \quad \mathscr{A}_{i,j,:} \mapsto \text{unvec}(M\text{vec}(\mathscr{A}_{i,j,:}))$$

for all $i = 1, 2, \ldots, n_1$ and $j = 1, 2, \ldots, n_2$,

$$R : \mathbb{F}_{p}^{n_1 \times n_2} \to \mathbb{F}_{n_3}^{n_1 \times n_2}, \quad \mathscr{B}_{i,j,:} \mapsto \text{unvec}(N\text{vec}(\mathscr{B}_{i,j,:}))$$

for all $i = 1, 2, \ldots, n_1$ and $j = 1, 2, \ldots, n_2$.

Definition 6. *Let L be the right-invertible operator defined by matrix M of size $p \times n_3$ between $\mathbb{F}_{n_3}^{n_1 \times k}$ and $\mathbb{F}_{n_3}^{n_1 \times k}$, and R be its inverse defined by matrix N of size $n_3 \times p$ between $\mathbb{F}_{p}^{n_1 \times n_2}$ and $\mathbb{F}_{n_3}^{n_1 \times n_2}$. The right-invertible operator product for two third-order tensors $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times k}$ and $\mathscr{B} \in \mathbb{F}_{n_3}^{k \times n_2}$ based on L, R is denoted by*

$$*_{L,R} : \mathbb{F}_{n_3}^{n_1 \times k} \times \mathbb{F}_{n_3}^{k \times n_2} \longrightarrow \mathbb{F}_{n_3}^{n_1 \times n_2}$$

and is defined as

$$\mathscr{A} *_{L,R} \mathscr{B} = ((\mathscr{A} \times_3 M) \triangle (\mathscr{B} \times_3 M)) \times_3 N$$

If there is no confusion and for simplicity, we denote

$$\mathscr{A} *_{L,R} \mathscr{B} = R(L(\mathscr{A}) \triangle L(\mathscr{B}))$$

Proposition 3. *Let L be the right-invertible operator between $\mathbb{F}_{n_3}^{n_1 \times n_2}$ and $\mathbb{F}_{p}^{n_1 \times n_2}$, and R be its inverse between $\mathbb{F}_{p}^{n_1 \times n_2}$ and $\mathbb{F}_{n_3}^{n_1 \times n_2}$. For $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times r}$ and $\mathscr{B} \in \mathbb{F}_{n_3}^{r \times n_2}$, the (i, j)-th tube fibre of $\mathscr{A} *_{L,R} \mathscr{B}$ can be computed as the appropriate sum of tube fibre products; i.e.,*

$$(\mathscr{A} *_{L,R} \mathscr{B})_{i,j} = \sum_{k=1}^{r} \mathbf{a}_{ik} *_{L,R} \mathbf{b}_{kj},$$

where $\mathbf{a}_{ik}$ and $\mathbf{b}_{kj}$ are the tube fibres of $\mathscr{A}$ and $\mathscr{B}$, respectively.

This proposition provides a method to compute the (i, j)-th tube fibre of the tensor product $\mathscr{A} *_{L,R} \mathscr{B}$ using sums of tube fibre products. This approach allows us to define new tensor products from a different perspective, simplifying both the theoretical formulation and the implementation of these operations in programming. It facilitates the development of efficient algorithms for tensor computations, enhancing the practical applicability of our

tensor robust principal component analysis (TRPCA) model. For the proof of Proposition 3, please refer to Appendix A.3.

Now, we provide the algorithm for computing the $*_{L,R}$ product of two tensors in Algorithm 1.

Algorithm 1 Compute the $*_{L,R}$ product of two tensors

Require: $n_1 \times k \times n_3$ tensor $\mathscr{A}$ and $k \times n_2 \times n_3$ tensor $\mathscr{B}$, right-invertible linear operators pair (L, R)
Ensure: $n_1 \times n_2 \times n_3$ tensor $\mathscr{C}$
1: $\hat{\mathscr{A}} \leftarrow L(\mathscr{A}), \hat{\mathscr{B}} \leftarrow L(\mathscr{B})$
2: **for** $i = 1$ to p **do**
3: $\quad \hat{\mathscr{C}}^{(i)} \leftarrow \hat{\mathscr{A}}^{(i)} \hat{\mathscr{B}}^{(i)}$
4: **end for**
5: $\mathscr{C} \leftarrow R(\hat{\mathscr{C}})$

The $*_{L,R}$ product retains several properties of the tensor product defined the via invertible matrix [22]. However, owing to the generally non-invertible properties of the right-invertible matrix used to define the $*_{L,R}$ product, several properties exhibit notable differences.

Lemma 1. *If $\mathscr{A}, \mathscr{B}, \mathscr{C}$ are third-order tensors of appropriate sizes, then the following statements are true:*

- ***Associative Property**: $(\mathscr{A} *_{L,R} \mathscr{B}) *_{L,R} \mathscr{C} = \mathscr{A} *_{L,R} (\mathscr{B} *_{L,R} \mathscr{C})$*
- ***Distributive Property**:*

$$\mathscr{A} *_{L,R} (\mathscr{B} + \mathscr{C}) = (\mathscr{A} *_{L,R} \mathscr{B}) + (\mathscr{A} *_{L,R} \mathscr{C})$$

$$(\mathscr{B} + \mathscr{C}) *_{L,R} \mathscr{A} = (\mathscr{B} *_{L,R} \mathscr{A}) + (\mathscr{C} *_{L,R} \mathscr{A})$$

- ***Noncommutative Property**: In general, $\mathscr{A} *_{L,R} \mathscr{B} \neq \mathscr{B} *_{L,R} \mathscr{A}$*

The proof of this lemma can be found in Appendix A.4, where the proof of associativity shows that the operator LR is the identity operator. This also indicates that, under this definition, if L is a left-invertible operator, the associative property does not hold.

Theorem 1 (Commutativity of the $*_{L,R}$ product for tube tensors)**.** *If $\mathscr{A}, \mathscr{B} \in \mathbb{F}^{1\times 1\times n_3}$, then the $*_{L,R}$ product is commutative. Specifically,*

$$\mathscr{A} *_{L,R} \mathscr{B} = \mathscr{B} *_{L,R} \mathscr{A}.$$

The proof of this theorem can be found in Appendix A.5. Based on this theorem, we can define more complex algebraic structures involving the new tensor product $*_{L,R}$. The commutativity property for tube tensors allows us to extend the framework to include tensor modules and other algebraic constructs. This extension enriches the theoretical foundation of our work and provides a robust basis for developing advanced tensor operations and applications.

Definition 7 ($*_{L,R}$ conjugate transpose)**.** *Let L be the right-invertible operator between $\mathbb{C}_{n_3}^{n_1\times n_2}$ and $\mathbb{C}_{p}^{n_1\times n_2}$. The conjugate transpose of a third-order tensor $\mathscr{A} \in \mathbb{C}_{n_3}^{n_1\times n_2}$, denoted by $\mathscr{A}^H$, is the tensor $\mathscr{A}^H \in \mathbb{C}_{n_3}^{n_2\times n_1}$ obtained by conjugate transposing of each of the frontal slices in the non-invertible transformed domain. Specifically,*

$$L(\mathscr{A}^H)^{(i)} = (L(\mathscr{A}^{(i)}))^H, \quad \text{for } i = 1, 2, \ldots, n_3,$$

where $L(\mathscr{A}^{(i)})$ denotes the i-th frontal slice of $\mathscr{A}$ and $(L(\mathscr{A}^{(i)}))^H$ denotes the conjugate transpose of the i-th frontal slice. However, since L is not necessarily invertible, different tensors may transform into the same tensor. Therefore, the conjugate transpose of $\mathscr{A}$ is not necessarily unique.

If $\mathscr{A}^H$ is the conjugate transpose of $\mathscr{A}$, then $\mathscr{A}^H + \mathscr{N}$ where $\mathscr{N} \in \ker(L)$ is also a conjugate transpose of $\mathscr{A}$. This is because for any $\mathscr{N} \in \ker(L)$, we have $L(\mathscr{N}) = 0$; thus, $L(\mathscr{A}^H + \mathscr{N}) = L(\mathscr{A}^H) + L(\mathscr{N}) = L(\mathscr{A}^H) = (L(\mathscr{A}))^H$. Therefore, $\mathscr{A}^H + \mathscr{N}$ satisfies the same properties as $\mathscr{A}^H$ in the non-invertible transformed domain, making it another valid conjugate transpose of $\mathscr{A}$.

Proposition 4. *The multiplication reversal property of the $*_{L,R}$ conjugate transpose holds: $\mathscr{A}^H *_{L,R} \mathscr{B}^H = [\mathscr{B} *_{L,R} \mathscr{A}]^H$.*

For the proof, please refer to Appendix A.6.

Definition 8 ($*_{L,R}$ identity tensor)**.** *Let L be the right-invertible operator between $\mathbb{C}_{n_3}^{n\times n}$ and $\mathbb{C}_{p}^{n\times n}$. The identity tensor $\mathscr{I} \in \mathbb{F}_{n_3}^{n\times n}$ is defined such that its frontal slices in the non-invertible transformed domain are all identity matrices. Formally, for $i = 1, 2, \ldots, p$,*

$$L(\mathscr{I})_{:,:,i} = I_n,$$

where I_n denotes the $n \times n$ identity matrix.

It is clear that $\mathscr{A} *_{L,R} \mathscr{I} = \mathscr{I} *_{L,R} \mathscr{A} = \mathscr{A}$. It is easy to say that $\mathscr{A} *_{L,R} (\mathscr{I} + \mathscr{N}) = (\mathscr{I} + \mathscr{N}) *_{L,R} \mathscr{A} = \mathscr{A}$, for all $\mathscr{N} \in Ker(L)$.

Definition 9 ($*_{L,R}$ unitary tensor)**.** *Let $\mathscr{Q} \in \mathbb{C}_{n_3}^{n\times n}$. $\mathscr{Q}$ is said to be unitary if $\mathscr{Q}^H *_{L,R} \mathscr{Q} = \mathscr{Q} *_{L,R} \mathscr{Q}^H = \mathscr{J}$.*

It is clear that if $\mathscr{Q}$ is unitary, then $\mathscr{Q} + \mathscr{N}$ for all $\mathscr{N} \in Ker(L)$ is also unitary according to the definition above.

Definition 10 (F-diagonal/F-upper/F-lower tensor)**.** *Let L be the right-invertible operator between $\mathbb{C}_{n_3}^{n_1\times n_2}$ and $\mathbb{C}_{p}^{n_1\times n_2}$ and $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1\times n_2}$. Then, $\mathscr{A}$ is called an F-diagonal/F-upper/F-lower tensor if all frontal slices $L(\mathscr{A})^{(i)}, i = 1, 2, \ldots, p$ of $\mathscr{A}$ are diagonal/upper triangular/lower triangular matrices in the non-invertible transformed domain.*

Having established the requisite algebraic framework, we now define a tensor singular value decomposition based on the $*_{L,R}$ product, denoted as the $*_{L,R}$ SVD. This formulation closely parallels the tensor singular value decomposition based on the tensor product via an invertible matrix, as introduced in Reference [22].

Theorem 2 ($*_{L,R}$ SVD)**.** *Let L be the right-invertible operator defined by matrix M of size $p \times n_3$ between $\mathbb{F}_{n_3}^{n_1\times n_2}$ and $\mathbb{F}_{p}^{n_1\times n_2}$, and R be its inverse defined by matrix N of size $n_3 \times p$ between $\mathbb{F}_{p}^{n_1\times n_2}$ and $\mathbb{F}_{n_3}^{n_1\times n_2}$. Let $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1\times n_2}$, then we have the following $*_{L,R}$ SVD:*

$$RL(\mathscr{A}) = \mathscr{U} *_{L,R} \mathscr{S} *_{L,R} \mathscr{V}^H$$

where $\mathscr{U} \in \mathbb{F}_{n_3}^{n_1\times n_1}$ and $\mathscr{V} \in \mathbb{F}_{n_3}^{n_2\times n_2}$ are unitary tensors and $\mathscr{S} \in \mathbb{F}_{n_3}^{n_1\times n_2}$ is an F-diagonal tensor. The main steps for computing the tensor $*_{L,R}$ SVD are summarised in Algorithm 2.

Algorithm 2 $*_{L,R}$ SVD under the $*_{L,R}$ product

Input: $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$, right-invertible linear operators pair $(L_{n_3 \times p}, R_{p \times n_3})$
Output: $\mathscr{U} \in \mathbb{F}_{n_3}^{n_1 \times n_1}$, $\mathscr{S} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$, $\mathscr{V} \in \mathbb{F}_{n_3}^{n_2 \times n_2}$

1: $\hat{\mathscr{A}} = L(\mathscr{A})$
2: **for** $i = 1$ to p **do**
3: $\quad [\hat{\mathscr{U}}^{(i)}, \hat{\mathscr{S}}^{(i)}, \hat{\mathscr{V}}^{(i)}] = \mathrm{svd}(\hat{\mathscr{A}}^{(i)})$
4: **end for**
5: $\mathscr{U} = R(\hat{\mathscr{U}})$, $\mathscr{S} = R(\hat{\mathscr{S}})$, $\mathscr{V} = R(\hat{\mathscr{V}})$

Based on this theorem, we can apply the $*_{L,R}$ SVD to various practical applications, such as image compression. By decomposing a tensor representing an image into its singular values, we can retain only the most significant components, leading to efficient storage and transmission while preserving the essential features of the image. This approach leverages the structure provided by the $*_{L,R}$ SVD to achieve high compression ratios with minimal loss of quality.

Definition 11. *Define the $*_{L,R}$ SVD error $\mathscr{E}$ of the $*_{L,R}$ SVD decomposition as $\mathscr{E} = \mathscr{A} - \mathscr{U} *_{L,R} \mathscr{S} *_{L,R} \mathscr{V}^H$, where $RL(\mathscr{A}) = \mathscr{U} *_{L,R} \mathscr{S} *_{L,R} \mathscr{V}^H$.*

When L is an invertible linear operator, the $*_{L,R}$ SVD error $\mathscr{E}$ of the $*_{L,R}$ SVD decomposition is a zero tensor.

Now, we can define the tensor multirank and tubal rank associated with the tensor $*_{L,R}$ product and the $*_{L,R}$ SVD as follows.

It is known that the singular values of a matrix have a decreasing order property. Let $RL(\mathscr{A}) = \mathscr{U} *_{L,R} \mathscr{S} *_{L,R} \mathscr{V}^H$ be the $*_{L,R}$ SVD of $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$. The entries on the diagonal of the first frontal slice $\mathscr{S}(:,:,1)$ have the same decreasing property, i.e.,

$$\mathscr{S}(i,i,1) \geq \mathscr{S}(j,j,1) \geq 0, \quad \forall 1 \geq i < j \geq n',$$

where $n' = \min(n_1, n_2)$. This property holds since the matrix N in the definition of the $*_{L,R}$ product gives

$$\mathscr{S}(i,i,1) = Nvec(\hat{\mathscr{S}}(i,i,:)),$$

and the entries on the diagonal of $\hat{\mathscr{S}}(:,:,j)$ are the singular values.

Definition 12 ($*_{L,R}$ tubal rank)**.** *Let L be the right-invertible operator defined by matrix M of size $p \times n_3$ between $\mathbb{F}_{n_3}^{n_1 \times n_2}$ and $\mathbb{F}_{p}^{n_1 \times n_2}$, and R be its inverse defined by matrix N of size $n_3 \times p$ between $\mathbb{F}_{p}^{n_1 \times n_2}$ and $\mathbb{F}_{n_3}^{n_1 \times n_2}$. Let $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$, the $*_{L,R}$ tubal rank of $\mathscr{A}$ under the $*_{L,R}$ product is defined by the number of nonzero tubes of $\mathscr{S}$, where $\mathscr{S}$ is obtained from the $*_{L,R}$ SVD, i.e.,*

$$rank_{*_{L,Rt}}(\mathscr{A}) = card(\{i \mid \mathscr{S}(i,i,:) \neq 0\}) = card(\{i \mid \mathscr{S}(i,i,1) \neq 0\})$$

Definition 13 ($*_{L,R}$ tensor average rank)**.** *For a tensor $\mathscr{A} \in \mathbb{F}^{n_1 \times n_2 \times n_3}$, the $*_{L,R}$ tensor average rank of $\mathscr{A}$, denoted as $rank_{*_{L,Ra}}(\mathscr{A})$, is defined as*

$$rank_{*_{L,Ra}}(\mathscr{A}) = \frac{1}{p} \sum_{i=1}^{p} rank(L(\mathscr{A})^{(i)})$$

where $L(\mathscr{A})^{(i)}$ represents the i-th frontal slice of $\mathscr{A}$ in the non-invertible transformed domain and L is the right-invertible linear operator used in the definition of the $*_{L,R}$ product.

Based on the above definition, we derive the following two theorems regarding the $*_{L,R}$ tubal rank and the $*_{L,R}$ tensor average rank.

Theorem 3. *Let $A \in \mathbb{F}^{n_3 \times n_3}$ be an invertible matrix, and let $M_{p\times n_3}$ be the matrix formed by retaining the first p rows of M_1. Then, for any tensor $\mathscr{A}$,*

$$rank_{*_{LRtp_1}}(\mathscr{A}) \leq rank_{*_{LRtp_2}}(\mathscr{A})$$

with $p_1 \leq p_2$, where $rank_{*_{LRtp_i}}(\mathscr{A})$ is the $*_{L,R}$ tubal rank of $\mathscr{A}$ associated with $M_{p_i\times n_3}$, $i = 1, 2$.

Note that similar conclusions hold for the $*_{L,R}$ tensor average rank.

Theorem 4. *For any tensor $\mathscr{A} \in \mathbb{R}_{n_3}^{n_1\times n_2}$, the relationship between the $*_{L,R}$ tensor average rank and the $*_{L,R}$ tensor tube rank of $\mathscr{A}$ satisfies*

$$rank_{*_{L,Ra}}(\mathscr{A}) \leq rank_{*_{L,Rt}}(\mathscr{A})$$

This theorem demonstrates that the $*_{L,R}$ tensor average rank is always less than or equal to the $*_{L,R}$ tensor tube rank. This relationship indicates that the $*_{L,R}$ tensor tube rank can be seen as an upper bound on the average rank of the frontal slices of the tensor. This result is significant in tensor decomposition and low-rank approximation problems, especially in the design and analysis of robust tensor decomposition algorithms.

Theorem 5. *The $*_{L,R}$ average rank of a tensor $\mathscr{A}$ is less than or equal to its CP rank.*

$$rank_{*_{L,Ra}}(\mathscr{A}) \leq rank_{CP}(\mathscr{A})$$

For the proof, please refer to Appendix A.8. Recall that the inequality $\text{rank}_{\text{CP}}(\mathscr{A}) \leq \text{rank}_{\text{Tucker}}(\mathscr{A})$ holds, which implies

Corollary 1. *The $*_{L,R}$ average rank of a tensor $\mathscr{A}$ is less than or equal to its Tucker rank.*

$$rank_{*_{L,Ra}}(\mathscr{A}) \leq rank_{Tucker}(\mathscr{A})$$

These theorems generalize the rank properties under the $*_{L,R}$ product, extending the understanding of tensor ranks from the traditional t-product context. These results provide a theoretical foundation for analysing tensors using the new product. Moreover, this facilitates the construction of relationships between different types of ranks and nuclear norms, which are essential for developing robust tensor decomposition methods and optimization algorithms. This advancement enhances our ability to model and analyse complex tensor data efficiently.

4. Tensor Robust Principal Component Analysis with Novel Tensor Multiplication

In the field of image processing, the low-rank property of images is an important characteristic. However, the low-rank function is nonconvex and highly sensitive to parameter choices and noise, making the optimisation problem challenging to solve. Because the properties of the rank function are poor, researchers have proposed the use of the nuclear norm, which approximates the tensor rank. This convex relaxation of the tensor rank not only simplifies the optimisation problem but also improves its convergence properties and robustness. In the following, the $*_{L,R}$ nuclear norm is defined as a measure that captures the

low-rank structure of the tensor after the right-invertible linear operator pair $(L_{p\times n_3}, R_{n_3\times p})$ is applied.

4.1. Tensor Nuclear Norm with New Tensor Multiplication

Definition 14. *Let $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1\times n_2}$ and $M_{n_3\times p}$ be a right-invertible matrix for defining the $*_{L,R}$ product, and then define the nuclear norm of $\mathscr{A}$ under the $*_{L,R}$ product as follows:*

$$\|\mathscr{A}\|_{*_{L,R}} = \sum_{i=1}^{p} \|L(\mathscr{A})^{(i)}\|_*$$

By this definition, we have $\|\mathscr{A} + \mathscr{N}\|_{*_{L,R}} = \|\mathscr{A}\|_{*_{L,R}}$ for all $\mathscr{N} \in Ker(L)$.

Since both definitions are based on slices, combining the conclusion that the nuclear norm is the convex envelope of the matrix rank, the convex envelope of the tensor average rank $\text{rank}_{*_{L,Ra}}(\mathscr{A})$ in the meaning of the $*_{L,R}$ product is the tensor nuclear norm $\|\mathscr{A}\|_{*_{L,R}}$.

The standard alternating direction method of multipliers (ADMM) is used in the process of solving the tensor robust principal component analysis with a tensor nuclear norm problem. A crucial step in this process is the computation of the proximal operator of the tensor nuclear norm (TNN).

The optimization subproblem for the $*_{L,R}$ TRPCA can be formulated as follows:

$$\min_{\mathscr{X}\in\mathbb{R}_{n_3}^{n_1\times n_2}} \tau\|\mathscr{X}\|_{*_{L,R}} + \frac{1}{2}\|\mathscr{X} - \mathscr{Y}\|_F^2, \tag{2}$$

where $\|\cdot\|_{*_{L,R}}$ denotes the tensor nuclear norm induced by the $*_{L,R}$ product, $\|\cdot\|_F$ represents the Frobenius norm, and $\tau > 0$ is a regularization parameter.

We provide an approximate solution for this optimization problem [23], which can be interpreted as the proximal operator of the matrix nuclear norm. Let $RL(\mathscr{Y}) = \mathscr{U} *_{L,R} \mathscr{S} *_{L,R} \mathscr{V}^*$ represent the $*_{L,R}$ singular value decomposition ($*_{L,R}$ SVD) of tensor $\mathscr{Y} \in \mathbb{R}_{n_3}^{n_1\times n_2}$. For each $\tau > 0$, we define the $*_{L,R}$ tensor singular value thresholding ($*_{L,R}$ SVT) operator as follows:

$$R(\mathscr{D}_\tau(L(\mathscr{Y}))) = \mathscr{U} *_{L,R} \mathscr{S}_\tau *_{L,R} \mathscr{V}^*, \tag{3}$$

where

$$\mathscr{S}_\tau = R((\hat{\mathscr{S}} - \tau)_+, [3]). \tag{4}$$

where $R(\cdot, [k])$ is the R operator in the $*_{L,R}$ product, which applies the soft-thresholding operation to the frontal slices of the tensor along the k-th dimension. Specifically, for each frontal slice $\mathscr{S}(:,:,i)$ of the tensor $\mathscr{S}$, the soft-thresholding operation is given by

$$(L(\mathscr{S})(:,:,i) - \tau)_+ = \max(L(\max(\mathscr{S})(:,:,i) - \tau, 0).$$

Note that $L(\mathscr{S})$ represents the singular value tensor obtained from the SVD decomposition of $L(\mathscr{Y})$ in the non-invertible transformed domain. The notation r_+ denotes the positive part of r, i.e., $r_+ = \max(r, 0)$. This operator applies a soft-thresholding rule to the singular values of the frontal slices of $L(\mathscr{Y})$, effectively shrinking these values towards zero. The $*_{L,R}$ SVT operator is the proximity operator associated with the $*_{L,R}$ TNN.

This formulation ensures that the solution $R(\mathscr{D}_\tau(L(\mathscr{Y})))$ is a low-rank approximation of the original tensor $\mathscr{Y}$ while maintaining a balance between rank minimisation and the data fidelity term. The $*_{L,R}$ SVT operator effectively reduces the rank of the tensor by thresholding the singular values, making it a powerful tool in tensor completion and robust principal component analysis tasks. We summarize the operators that will be used in the subsequent sections in Table 1.

Table 1. Summary of Operators Employed in Subsequent Analysis.

Operator	Definition
$*_{L,R}$ Singular Value Decomposition ($*_{L,R}$ SVD)	$R_L(\mathscr{Y}) = \mathscr{U} *_{L,R} \mathscr{S} *_{L,R} \mathscr{V}^*$
$*_{L,R}$ Tensor Singular Value Thresholding ($*_{L,R}$ SVT operator)	$R(\mathscr{D}_\tau(L(\mathscr{Y}))) = \mathscr{U} *_{L,R} \mathscr{S}_\tau *_{L,R} \mathscr{V}^*$
$*_{L,R}$ Soft-thresholding operation	$(L(\mathscr{S})(:,:,i) - \tau)_+ = \max(L(\mathscr{S})(:,:,i) - \tau, 0)$

Based on the aforementioned operator, we summarize the algorithm for $*_{L,R}$-based Tensor Singular Value Thresholding ($*_{L,R}$ SVT) as shown in Algorithm 3.

Algorithm 3 $*_{L,R}$-based Tensor Singular Value Thresholding ($*_{L,R}$ SVT)

Input:$\mathscr{Y} \in \mathbb{R}^{n_1 \times n_2}_{n_3}, \tau > 0$,Right-invertible linear operators pair $(L_{p \times n_3}, R_{n_3 \times p})$
Output: $R(\mathscr{D}_\tau(L(\mathscr{A})))$ as defined in Equation (3).

1: Compute $\bar{\mathscr{Y}} = L(\mathscr{Y})$
2: **for** $i = 1$ to p **do**
3: $\quad [U, S, V] = \text{svd}(\bar{\mathscr{Y}}^{(i)})$
4: $\quad \bar{\mathscr{W}}^{(i)} = U \cdot (S - \tau)_+ \cdot V^*$
5: **end for**
6: Compute $R(\mathscr{D}_\tau(L(\mathscr{Y}))) = R(\bar{\mathscr{W}})$

Theorem 6. *Let $\mathscr{A} \in \mathbb{F}^{n_1 \times n_2}_{n_3}$, $\tau > 0$, and L be a right invertible linear operator. Let the matrix representation of L be $M_{p \times n_3}$, which consists of the first p rows of an orthogonal matrix. Then,*

$$R(\mathscr{D}_\tau(L(\mathscr{A}))) = \arg \min_{\mathscr{Y} \in \mathbb{F}^{n_1 \times n_2}_{n_3}} \left\{ \tau \|\mathscr{Y}\|_{*_{L,R}} + \frac{1}{2} \|L(\mathscr{A} - \mathscr{Y})\|_F^2 \right\},$$

where $\|\cdot\|_{*_{L,R}}$ denotes the tensor nuclear norm based on the $*_{L,R}$ product, $\|\cdot\|_F^2$ denotes the Frobenius norm, and $\tau > 0$ is a regularisation parameter.

For the proof of this theorem, we refer the reader to Appendix A.9. Notably, the closed-form expression for the $*_{L,R}$-based SVT operator $D_\tau(\mathscr{Y})$ can be viewed as a natural generalization of the t-SVT operator introduced in [3].

Theorem 7. *Let $\mathscr{A} \in \mathbb{F}^{n_1 \times n_2}_{n_3}$ and $\tau > 0$. Let $M_{p \times n_3}$ be the matrix corresponding to L, satisfying $\| \frac{(\mathscr{A} - \mathscr{Y}) \times_3 M\|_F^2}{(\mathscr{A} - \mathscr{Y})\|_F^2} \approx 1$, then*

$$R(\mathscr{D}_\tau(L(\mathscr{A}))) \approx \arg \min_{\mathscr{Y} \in \mathbb{F}^{n_1 \times n_2}_{n_3}} \left\{ \tau \|\mathscr{Y}\|_{*_{L,R}} + \frac{1}{2} \|\mathscr{A} - \mathscr{Y}\|_F^2 \right\},$$

where $\|\cdot\|_{*_{L,R}}$ denotes the tensor nuclear norm based on the $*_{L,R}$ product, $\|\cdot\|_F^2$ denotes the Frobenius norm, and $\tau > 0$ is a regularisation parameter.

The theorem only provides an approximate solution to the optimization Problem 2. We have experimentally verified its feasibility to a certain extent; however, further research is required to obtain more precise solutions and computational methods. The proof of this theorem can be found in Appendix A.10.

4.2. ADMM Algorithm for Tensor Robust Principal Component Analysis with $*_{L,R}$ Tensor Product

We consider the following optimisation problem

$$\min_{\mathscr{L},\mathscr{E}} \|\mathscr{L}\|_{*_{L,R}} + \lambda\|\mathscr{E}\|_1 \quad \text{subject to} \quad \mathscr{L} + \mathscr{E} = \mathscr{X} \tag{5}$$

where $\mathscr{L}$ is the low-rank tensor; $\mathscr{E}$ is the sparse tensor; $\mathscr{X}$ is the observed tensor; $\|\mathscr{L}\|_{*_{L,R}}$ is the tensor nuclear norm (low-rank regularisation term); $\|\mathscr{E}\|_1$ is the tensor L_1 norm (sparsity regularisation term); and λ is the balancing parameter between low rank and sparsity.

To solve this problem via the alternating direction method of multipliers (ADMM), we introduce the augmented Lagrangian function as follows:

$$\mathscr{L}_\mu(\mathscr{L},\mathscr{E},\mathscr{Y}) = \|\mathscr{L}\|_{*_{L,R}} + \lambda\|\mathscr{E}\|_1 + \langle\mathscr{Y},\mathscr{L}+\mathscr{E}-\mathscr{X}\rangle + +\frac{\mu}{2}\|\mathscr{L}+\mathscr{E}-\mathscr{X}\|_F^2 \tag{6}$$

where $\mathscr{Y}$ is the Lagrange multiplier and μ is the penalty parameter.

The ADMM algorithm updates $\mathscr{L}$, $\mathscr{E}$, and $\mathscr{Y}$ iteratively.

Update $\mathscr{L}$:

$$\mathscr{L}^{k+1} = \arg\min_{\mathscr{L}} \|\mathscr{L}\|_{*_{L,R}} + \frac{\mu}{2}\|\mathscr{L}+\mathscr{E}^k-\mathscr{X}+\frac{1}{\mu}\mathscr{Y}^k\|_F^2 \tag{7}$$

Update $\mathscr{E}$:

$$\mathscr{E}^{k+1} = \arg\min_{\mathscr{E}} \lambda\|\mathscr{E}\|_1 + \frac{\mu}{2}\|\mathscr{L}^{k+1}+\mathscr{E}-\mathscr{X}+\frac{1}{\mu}\mathscr{Y}^k\|_F^2 \tag{8}$$

Update $\mathscr{Y}$:

$$\mathscr{Y}^{k+1} = \mathscr{Y}^k + \mu(\mathscr{L}^{k+1}+\mathscr{E}^{k+1}-\mathscr{X}) \tag{9}$$

Convergence Conditions:

$$\|\mathscr{L}^{k+1}-\mathscr{L}^k\|_F < \epsilon, \|\mathscr{E}^{k+1}-\mathscr{E}^k\|_F < \epsilon, \|\mathscr{L}^{k+1}+\mathscr{E}^{k+1}-\mathscr{X}\|_F < \epsilon,$$

where ϵ is a small positive tolerance.

In the following, we present the solution of the subproblem in the ADMM.

Solution of Update $\mathscr{L}$: Approximate solution of the subproblem to update the low-rank tensor $\mathscr{L}$

$$\mathscr{L}^{k+1} = \arg\min_{\mathscr{L}} \|\mathscr{L}\|_{*_{L,R}} + \frac{\mu}{2}\|\mathscr{L}+\mathscr{E}^k-\mathscr{X}+\frac{1}{\mu}\mathscr{Y}^k\|_F^2 \tag{10}$$

First, we set

$$\mathscr{Z} = \mathscr{E}^k - \mathscr{X} + \frac{1}{\mu}\mathscr{Y}^k$$

and then, the objective function can be rewritten as

$$\mathscr{L}^{k+1} = \arg\min_{\mathscr{L}} \|\mathscr{L}\|_{*_{L,R}} + \frac{\mu}{2}\|\mathscr{L}-(-\mathscr{Z})\|_F^2$$

This problem can be solved via the tensor singular value thresholding operation $R(\mathscr{D}_\tau(L(\mathscr{A})))$, as defined in Equation (3); here, we denote it as $*_{L,R}$ SVT. The $*_{L,R}$ SVT operation is defined as

$$*_{L,R}\,\mathrm{SVT}(\mathscr{T},\tau) \approx \arg\min_{\mathscr{L}} \|\mathscr{L}\|_{*_{L,R}} + \frac{1}{2\tau}\|\mathscr{L}-\mathscr{T}\|_F^2$$

where τ is the threshold parameter. In our subproblem, $\mathscr{T} = -\mathscr{Z}$ and $\tau = \frac{1}{\mu}$; thus,

$$\mathscr{L}^{k+1} = *_{L,R}\ \mathrm{SVT}\left(-\mathscr{Z}, \frac{1}{\mu}\right)$$

Substituting back $\mathscr{Z}$, we obtain

$$\mathscr{L}^{k+1} = *_{L,R}\ \mathrm{SVT}\left(\mathscr{X} - \mathscr{E}^k - \frac{1}{\mu}\mathscr{Y}^k, \frac{1}{\mu}\right)$$

Solution of Update $\mathscr{E}$: Subproblem 2 can be solved via the soft-thresholding operator

$$\mathscr{E}^{k+1} = \mathrm{SoftShrink}\left(\mathscr{X} - \mathscr{L}^{k+1} - \frac{1}{\mu}\mathscr{Y}^k, \frac{\lambda}{\mu}\right)$$

where the soft-thresholding operator is defined as

$$\mathrm{SoftShrink}(x, \tau) = \mathrm{sign}(x) \cdot \max(|x| - \tau, 0)$$

This operation effectively shrinks the singular values of the tensor towards zero, promoting a low-rank structure in the tensor $\mathscr{X}$.

We summarize the above algorithm steps as Algorithm 4, in which we specify the values of some parameters.

Algorithm 4 Solve (5) via ADMM

Input: $\mathscr{X} \in \mathbb{R}^{n_1 \times n_2}_{n_3}, \tau > 0$, Right-invertible linear operators pair $(L_{p \times n_3}, R_{n_3 \times p})$
Initialize: $\rho = 1.1$, $\mu_0 = 1e^{-3}$, $\mu_{\max} = 1e^{10}$, $\epsilon = 1e^{-8}$.
Output: $\mathscr{L}, \mathscr{E}$.

1. Update $\mathscr{L}_{k+1}$ by

$$\mathscr{L}_{k+1} = \arg\min_{\mathscr{L}} \|\mathscr{L}\|_{*_{L,R}} + \frac{\mu_k}{2}\left\|\mathscr{L} + \mathscr{E}_k - \mathscr{X} + \frac{\mathscr{Y}_k}{\mu_k}\right\|_F^2$$

2. Update $\mathscr{E}_{k+1}$ by

$$\mathscr{E}_{k+1} = \arg\min_{\mathscr{E}} \lambda\|\mathscr{E}\|_1 + \frac{\mu_k}{2}\left\|\mathscr{L}_{k+1} + \mathscr{E} - \mathscr{X} + \frac{\mathscr{Y}_k}{\mu_k}\right\|_F^2$$

3. Update $\mathscr{Y}_{k+1} = \mathscr{Y}_k + \mu_k(\mathscr{L}_{k+1} + \mathscr{E}_{k+1} - \mathscr{X})$
4. Update $\mu_{k+1} = \min(\rho\mu_k, \mu_{\max})$
5. Check convergence conditions

$$\|\mathscr{L}_{k+1} - \mathscr{L}_k\|_\infty \le \epsilon,\ \|\mathscr{E}_{k+1} - \mathscr{E}_k\|_\infty \le \epsilon,\ \|\mathscr{L}_{k+1} + \mathscr{E}_{k+1} - \mathscr{X}\|_\infty \le \epsilon$$

4.3. Computational Complexity Analysis

In this section, we analyse the computational complexity of the proposed method, which uses the alternating direction method of multipliers (ADMM) to solve the tensor robust principal component analysis (TRPCA) problem. The initialisation step, involving the setup of the low-rank component $\mathscr{L}$, the sparse noise component $\mathscr{E}$, and the penalty term $\mathscr{Y}$, has a complexity of $O(n_1 n_2 n_3)$, where n_1, n_2, and n_3 are the dimensions of the tensor. Each ADMM iteration consists of updating $\mathscr{L}_{k+1}$ by performing singular value decomposition (SVD) on each of the p slices of the non-invertible transformed domain, resulting in a complexity of $O(p \cdot \min(n_1^2 n_2, n_1 n_2^2))$. Updating $\mathscr{E}_{k+1}$ via soft thresholding has a complexity of $O(n_1 n_2 n_3)$, whereas updating $\mathscr{Y}_{k+1}$ and checking the convergence con-

ditions both have a complexity of $O(n_1n_2n_3)$. Assuming k iterations, the total complexity of the ADMM process is $O(k \cdot (n_1n_2n_3 + p \cdot \min(n_1^2n_2, n_1n_2^2)))$. From the complexity analysis, it is evident that the first dimension p of the right invertible matrix has a linear relationship with the computational complexity. To more clearly demonstrate the efficiency advantages of our algorithm, we provide a comparison of algorithm complexity in Table 2.

Table 2. Complexity comparison of $*_{L,R}$ TRPCA algorithms (TRPCA-(p/n_3)M) and classical TRPCA (TRPCA-M).

Algorithm	Description	Complexity
TRPCA-(p/n_3)M	Using the first p rows from an invertible matrix M of order n_3	$O(k \cdot (n_1n_2n_3 + p \cdot \min(n_1^2n_2, n_1n_2^2)))$
TRPCA-M	Classical algorithm using all rows from an invertible matrix M of order n_3	$O(k \cdot (n_1n_2n_3 + n_3 \cdot \min(n_1^2n_2, n_1n_2^2)))$

Note: The main difference between the two algorithms lies in the number of rows used from the invertible matrix M. TRPCA-(p/n_3)L uses only the first p rows, while TRPCA-M uses all n_3 rows. This results in a reduced computational complexity for TRPCA-(p/n_3)M when $p < n_3$.

From a simple analysis, we can see that the space complexity of both algorithms is the same. Both algorithms require storing the input tensor $\mathscr{X}$, the low-rank component $\mathscr{L}$, the sparse error component $\mathscr{E}$, and the dual variable $\mathscr{Y}$, leading to a space complexity of $O(n_1n_2n_3)$.

5. Numerical Experiments

In this section, we apply the $*_{L,R}$ TRPCA algorithm to both randomly generated synthetic data and real-world data for grey video denoising and motion detection. This approach allows us to evaluate the performance of the algorithm and demonstrate its advantages. All the experiments were conducted on a personal computer (Dell, Xiamen, China) equipped with an Intel(R) Core(TM) i7-9750H 2.30 GHz processor and 8 GB of RAM. The experiments were implemented using Python in the PyCharm 2021.3.3 (Community Edition) environment. We applied the $*_{L,R}$ TRPCA algorithm to recover the original tensor from those corrupted by random noise. To simulate sparse noise interference in real-world applications, we added sparse noise to the original data tensor $\mathscr{X}$. Specifically, we first created a zero matrix sparse_noise with the same shape as $\mathscr{X}$. Next, we randomly selected 1% of the elements in $\mathscr{X}$ as noise points, ensuring that each position was chosen only once. For these selected positions, we assigned random values drawn from a uniform distribution $[0, 1]$. Finally, we added the generated sparse noise matrix sparse_noise to the original data $\mathscr{X}$ to obtain the noisy data $\mathscr{X}_{\text{bar}}$.

We use the peak signal-to-noise ratio (PSNR) for each slice along the third dimension, defined as

$$\text{PSNR}_k = 10\log_{10}\left(\frac{\|\mathscr{M}(:,:,k)\|_2^2}{\frac{1}{n_1n_2}\|\mathscr{X}(:,:,k) - \hat{\mathscr{M}}(:,:,k)\|_F^2}\right), \tag{11}$$

To evaluate the recovery performance of each slice, $\mathscr{M}(:,:,k)$ represents the k-th slice along the third dimension of the original tensor data; $\hat{\mathscr{M}}(:,:,k)$ represents the k-th slice along the third dimension of the recovered tensor data after processing; $\mathscr{X}(:,:,k)$ represents the k-th slice along the third dimension of the observed tensor data, which may contain noise or other distortions; and n_1, n_2 represents the sizes of the first and second dimensions of the tensor slices. The PSNR is a measure of the quality of the recovered data compared with the original data. Higher PSNR values indicate better recovery performance.

The relative squared error (RSE) for each slice along the third dimension, used to measure the relative error between the recovered data and the original data, is defined as follows:

$$\mathrm{RSE}_k = \frac{\|\mathscr{X}(:,:,k) - \hat{\mathscr{M}}(:,:,k)\|_F^2}{\|\mathscr{X}(:,:,k) - \bar{\mathscr{X}}(:,:,k)\|_F^2}, \tag{12}$$

where $\bar{\mathscr{X}}(:,:,k)$ represents the mean of the k-th slice along the third dimension of the observed tensor data, i.e., the average value of all the elements at each position in the slice. The RSE measures the relative squared error between the recovered data and the original data. Lower RSE values indicate better recovery performance. We choose the PSNR, relative squared error (RSE), and runtime as the evaluation metrics to assess the denoising performance of various $*_{L,R}$ TRPCA methods.

The right-invertible matrix is obtained by removing some rows from a 200×200 discrete cosine transform (DCT) matrix. The size parameter denotes the number of rows retained from the DCT matrix. To present the numerical results, we set the size p to 20, 40, 60, 100, and 200. The resulting right-invertible matrices are denoted as (20/200)C, (40/200)C, (60/200)C, (100/200)C, and C, respectively. $*_{L,R}$ TRPCA algorithms constructed based on these right-invertible matrices are referred to as TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, TRPCA-(100/200)C, and TRPCA-C. When the size is 200, it corresponds to the classical TRPCA method, which is based on the c-product [22], and in this paper, we denote it as TRPCA-C. The classical TRPCA-C method, which is based on the c-product method, outperforms robust principal component analysis (RPCA) and sparse nonnegative matrix factorisation (SNN) by providing better recovery accuracy and noise reduction in the processed data by Reference [15]. Therefore, in our comparative experiments, we exclusively compare TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, TRPCA-(100/200)C against the classical TRPCA-C. On the basis of the related literature and our experimental calculations, we set λ to $\lambda = 30 \times (\max(m,n) \times l)^{-0.5}$.

5.1. Synthetic Data

In this section, we evaluate the performance of the proposed $*_{L,R}$ nuclear norm-based TRPCA method using synthetic data. The synthetic data are generated as a random tensor of dimensions $100 \times 100 \times 200$ normalised to the range [0, 1]. Sparse noise is added to the original tensor, affecting approximately 1% of the elements. The noisy tensor is then used as input to the TRPCA algorithm. For simplicity, we will use the condition $\frac{\|\mathscr{X} \times_3 M\|_F^2}{\|\mathscr{X}\|_F^2}$ to approximate the analysis of $\frac{\|(\mathscr{X}-\mathscr{Y}) \times_3 M\|_F^2}{\|(\mathscr{X}-\mathscr{Y})\|_F^2}$.

The synthetic data are generated as a random tensor $\mathscr{X}$ of dimensions $100 \times 100 \times 200$, normalized to the range [0, 1]. We compute the ratio $\frac{\|\mathscr{X} \times_3 M\|_F^2}{\|\mathscr{X}\|_F^2}$ for different sizes of the right-invertible matrix M.

Table 3 shows the computed ratios for different right-invertible matrix M. These ratios indicate that for artificial random data, the ratio $\frac{\|\mathscr{X} \times_3 M\|_F^2}{\|\mathscr{X}\|_F^2}$ approaches 1 as the truncated size of M increases.

Table 3. Ratio $\frac{\|\mathscr{X} \times_3 M\|_F^2}{\|\mathscr{X}\|_F^2}$ for Various M Values, with $\mathscr{X}$ as a Random Tensor.

Right-invertible matrix M	(20/200)C	(40/200)C	(60/200)C	(100/200)C	C
Ratio	0.9550	0.9639	0.9695	0.9788	1.0000

The experiment involves varying the size of the right-invertible linear operator matrix, denoted as p, from 20 to 200 in steps. For each size, the TRPCA algorithm is applied to the noisy tensor, and the results are evaluated via the peak signal-to-noise ratio (PSNR)

and relative squared error (RSE) metrics. The PSNR and RSE values are calculated for all frames of the tensor.

The results show that the proposed method effectively recovers the low-rank component of the tensor while maintaining computational efficiency and stability. The PSNR values increase as the size of the right-invertible matrix increases, indicating better recovery performance. The RSE values also decrease, confirming the improved accuracy of the low-rank approximation.

Table 4 summarizes the runtime of the TRPCA algorithms: TRPCA-C, TRPCA-(100/200)C, TRPCA-(40/200)C, and TRPCA-(20/200)C, applied to synthetic data. As the truncation size increases, the runtime also increases. However, the improvements in PSNR (Peak Signal-to-Noise Ratio) and reductions in RSE (Relative Squared Error) justify the additional computational cost.

Table 4. Runtime Comparison of $*_{L,R}$ TRPCA Algorithms (TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, and TRPCA-(100/200)C) and Classical TRPCA Algorithm (TRPCA-C) Applied to Synthetic Data.

Algorithm	TRPCA-(20/200)C	TRPCA-(40/200)C	TRPCA-(60/200)C	TRPCA-(100/200)C	TRPCA-C
Runtime (s)	128.2	150.9	258.8	314.2	694.7

Figures 1 and 2 provide visualisations of the PSNR and RSE values for $*_{L,R}$ TRPCA Algorithms (TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, TRPCA-(100/200)C) and the classical TRPCA algorithm (TRPCA-C). For synthetic data, the tensor recovery performance of the $*_{L,R}$ TRPCA algorithms has a noticeable gap compared with the classical algorithm TRPAC-C.

5.2. Video Denoising and Anomaly Detection

In these experiments, we used the `Test001` subset from the `UCSD_Anomaly_Dataset` directory. The UCSD anomaly detection dataset is a widely used benchmark for evaluating anomaly detection algorithms in video surveillance. `Test001` includes 200 frames from a video sequence, which we used to construct a video tensor. For more information about the dataset, please refer to UCSD Anomaly Detection Dataset (http://www.svcl.ucsd.edu/projects/anomaly/, accessed on 22 January 2025).

The video data are represented as a tensor of dimensions $158 \times 238 \times 200$. We compute the ratio $\frac{\|\mathscr{X} \times_3 M\|_F^2}{\|\mathscr{X}\|_F^2}$ for different sizes of the right-invertible matrix M, where $\mathscr{X}$ is the video tensor.

Table 5 shows the computed ratios for different right-invertible matrix M. These ratios indicate that for real data, the ratio $\frac{\|\mathscr{X} \times_3 M\|_F^2}{\|\mathscr{X}\|_F^2}$ approaches 1 more closely than synthetic data do. This confirms that real data are more consistent with the condition $\frac{\|\mathscr{X} \times_3 M\|_F^2}{\|\mathscr{X}\|_F^2} \approx 1$.

Table 5. Ratio $\frac{\|\mathscr{X} \times_3 M\|_F^2}{\|\mathscr{X}\|_F^2}$ for Various M Values, with $\mathscr{X}$ as a video Tensor.

Right-invertible matrix M	(20/200)C	(40/200)C	(60/200)C	(100/200)C	C
Ratio	0.9824	0.9923	0.9957	0.9984	1.0000

Figure 3 shows the PSNR values for $*_{L,R}$ TRPCA algorithms (TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, and TRPCA-(100/200)C), and the classical TRPCA algorithm (TRPCA-C) across various frames. Figure 4 shows the RSE values for $*_{L,R}$ TRPCA algorithms (TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, and TRPCA-(100/200)C), and the classical TRPCA algorithm (TRPCA-C) across various frames. Com-

bining the information from the previous text and the new details about Figures 3 and 4, we can analyse the experimental results as follows.

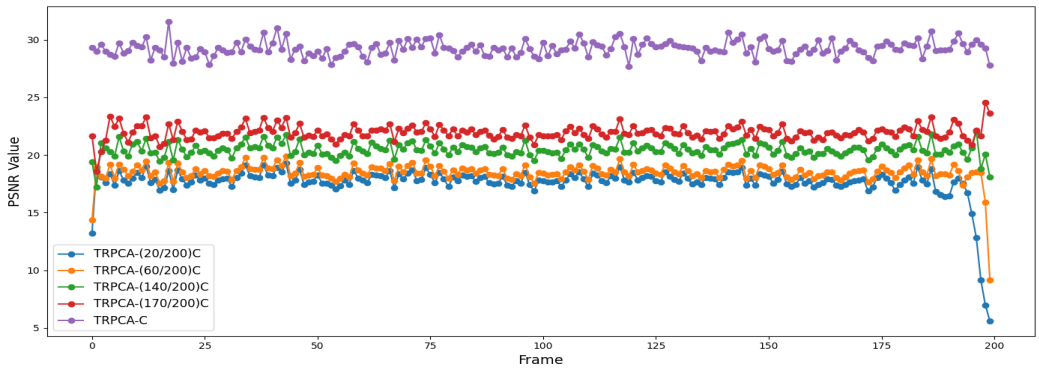


Figure 1. Comparative analysis of PSNR values for $*_{L,R}$ TRPCA algorithms (TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, and TRPCA-(100/200)C), and classical TRPCA algorithm (TRPCA-C) across frames on random tensor.

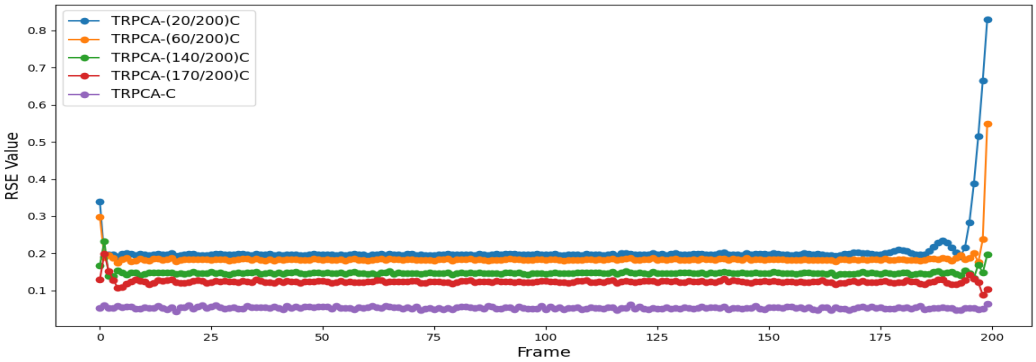


Figure 2. Comparative analysis of RSE values for $*_{L,R}$ TRPCA algorithms (TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, and TRPCA-(100/200)C), and classical TRPCA algorithm (TRPCA-C) across frames on random tensor.

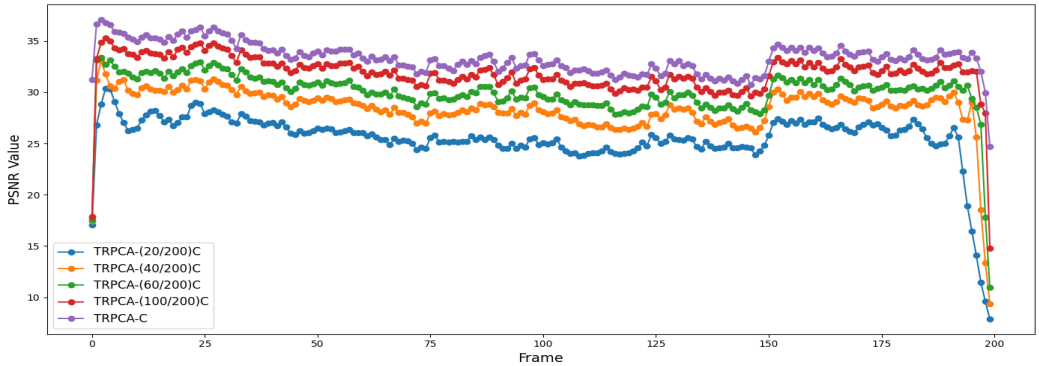


Figure 3. Comparative analysis of PSNR values for TRPAC-C, TRPCA-(100/200)C, TRPCA-(40/200)C, and TRPCA-(20/200)C algorithms across video frames.

We can deduce that for most frames, the PSNR values remain relatively close across $*_{L,R}$ TRPCA algorithms: TRPCA-(100/200)C, TRPCA-(40/200)C, and TRPCA-(20/200)C, and the classical TRPCA algorithm (TRPCA-C), suggesting consistent recovery quality. Interestingly, in certain specific frames, TRPCA-(20/200)C and TRPCA-(40/200)C manage to achieve slightly higher PSNR values than TRPAC-C and TRPCA-(100/200)C, suggesting a possible advantage in terms of recovery performance.

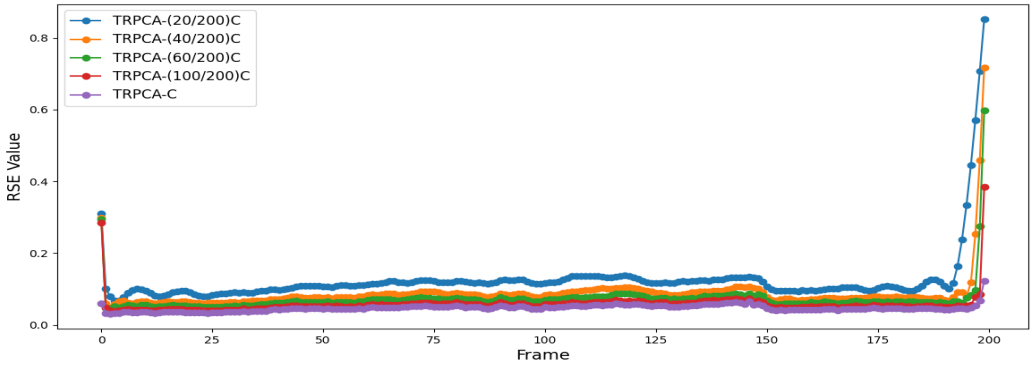


Figure 4. Comparative analysis of RSE values for TRPAC-C, TRPCA-(100/200)C, TRPCA-(40/200)C, and TRPCA-(20/200)C algorithms across video frames.

Similarly, Figure 4 shows the relative squared error (RSE) values for $*_{L,R}$ TRPCA algorithms (TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, and TRPCA-(100/200)C), and the classical TRPCA algorithm (TRPCA-C) across various frames. Upon examining this figure, we notice that for most frames, the RSE values are relatively close across different algorithms, further supporting the notion of consistent recovery quality. Once again, in some specific frames, TRPCA-(20/200)C and TRPCA-(40/200)C display slightly lower RSE values than TRPAC-C and TRPCA-(100/200)C, reinforcing the trend observed in the PSNR analysis.

In Table 6, we present the runtime analysis for the $*_{L,R}$ TRPCA method applied to video sequences with different $*_{L,R}$ TRPCA algorithms (TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, and TRPCA-(100/200)C), and the classical TRPCA algorithm (TRPCA-C). The runtime for each size is measured in seconds. As shown in the Table, the runtime increases significantly with increasing DCT truncation size used to define the $*_{L,R}$ TRPCA algorithm. This trend indicates that larger DCT truncation sizes, although providing better recovery performance, come at the cost of increased computational time. Therefore, a trade-off between recovery quality and computational efficiency must be considered when selecting the appropriate DCT truncation size to the $*_{L,R}$ TRPCA algorithm for practical applications.

Table 6. Comparison of runtime for $*_{L,R}$ TRPCA algorithms (TRPCA-(20/200)C, TRPCA-(40/200)C, TRPCA-(60/200)C, and TRPCA-(100/200)C), and the classical TRPCA algorithm (TRPCA-C) Applied to video Data.

Algorithm	TRPCA-(20/200)C	TRPCA-(40/200)C	TRPCA-(60/200)C	TRPCA-(100/200)C	TRPCA-C
Runtime (s)	402.63	550.03	664.75	1254.44	2026.47

On the basis of the numerical results, several conclusions can be drawn regarding the impact of varying truncation levels in the definition of the $*_{L,R}$ TRPCA algorithm using different-sized right-invertible matrices derived from the discrete cosine transform (DCT).

First, in Figure 5, the top row presents the original images from frames 40, 70, and 120, whereas the second row shows the corresponding noisy versions. The other row shows the recovery results achieved via the $*_{L,R}$ TRPCA method with a right-invertible matrix constructed from the different truncations of the DCT matrix. The bottom row shows the recovery results of classical TRPCA (TRPCA-C). In Figure 6, the top row again features the original images from the same set of frames, followed by the noisy versions. The other row shows the residual results after denoising via the $*_{L,R}$ TRPCA method with a right-invertible matrix constructed from the different truncations of the DCT matrix and classical TRPCA (TRPCA-C). Notably, these recovered images are significantly closer to the original

images, suggesting that incorporating more DCT coefficients enhances the recovery process, albeit at the cost of increased computational complexity. These figures highlight the varying recovery efficacy across different frames within the video sequence. For example, frame 120, where a child quickly rides a bicycle into the scene, yields less satisfactory recovery results than slower-motion scenes such as frames 40 and 70 do. This observation highlights the impact of anomalous situations on video recovery, particularly how fast movements generally compromise recovery accuracy. Although the recovery results are not as good for larger sizes, this method can still detect anomalous situations effectively.

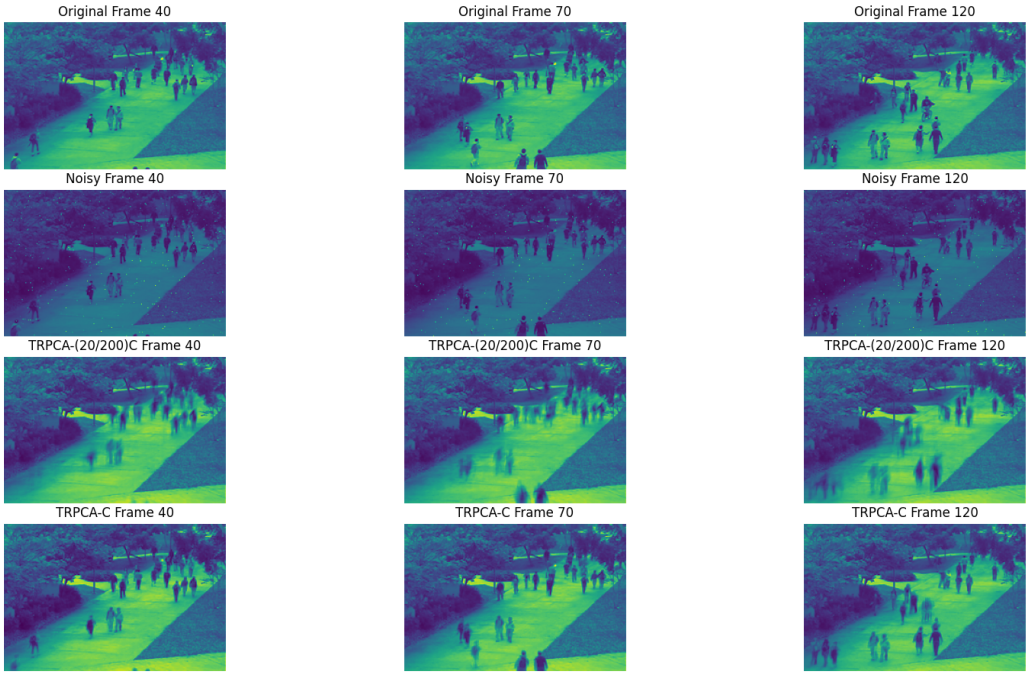


Figure 5. Comparison of denoised images with TRPAC-C (classic TRPCA) and TRPCA-(20/200)C($*_{L,R}$ TRPCA) and frames = 40, 70, 120.

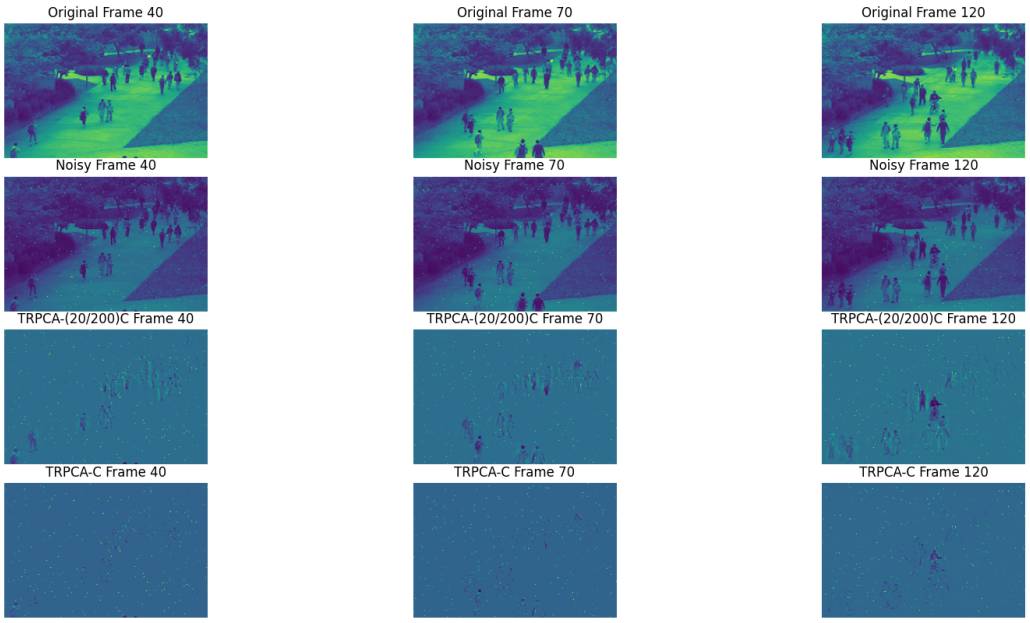


Figure 6. Comparison of residuals after denoising with TRPAC-C (classic TRPCA) and TRPCA-(20/200)C($*_{L,R}$ TRPCA) and frames = 40, 70, 120.

Nonetheless, adopting larger TRPCA truncation sizes enables superior denoising and recovery outcomes even in regions with rapid motion. Consequently, adjusting the number of DCT coefficients employed in the $*_{L,R}$ TRPCA method offers a trade-off between recovery quality and computational efficiency. Specifically, smaller sizes yield higher computational efficiency for frames with slow image operators, whereas larger sizes enhance recovery quality in the presence of fast-moving objects because of the accelerated pixel variations at the same location, which facilitates dynamic detection.

To optimise both recovery quality and computational efficiency in video sequence denoising and recovery tasks, it is advisable to segment the video according to its motion intensity. Smaller operator sizes can be applied to sections with slow motion to conserve computation time, whereas larger sizes can be utilised in parts with rapid motion to ensure better precision. This adaptive strategy strikes a balance between the two objectives, ensuring optimal performance tailored to specific application demands.

In light of these findings, it becomes evident that the $*_{L,R}$ TRPCA method offers a balance between recovery quality and computational efficiency. Smaller sizes (e.g., TRPCA-(20/200)C) tend to provide competitive recovery performance in certain situations, whereas larger sizes (e.g., TRPCA-(170/200)C) may offer advantages in other situations. By adapting the size of the right-invertible matrix constructed from the DCT according to the motion intensity in the video sequence, users can strike a balance between speed and accuracy, optimising the overall performance of the algorithm for their specific needs. This adaptability makes the $*_{L,R}$ TRPCA method a practical choice for real-world applications, particularly when factors such as dynamic detection and computational efficiency are considered.

In summary, the $*_{L,R}$ TRPCA method effectively balances recovery quality and computational efficiency, rendering it a practical choice for real-world scenarios. When prioritising anomaly detection, smaller sizes should be considered because of their enhanced computational efficiency. Thus, careful consideration must be given to balancing recovery quality and computational efficiency in practical applications to attain optimal performance.

6. Conclusions

This paper introduces a novel method for tensor robust principal component analysis (TRPCA) by extending traditional tensor product definitions through right-invertible linear operators. This extension significantly enhances the flexibility of linear operators and computational efficiency, reducing both runtime and storage requirements compared to existing tensor product algorithms. A new class of tensor products for third-order tensors is introduced via right-invertible linear operators, providing a more flexible framework for tensor operations. Leveraging these new tensor products, we propose a TRPCA model that increases computational efficiency and robustness. The use of the tensor nuclear norm in this model offers an efficient and robust solution for tensor data, enabling better recovery of low-rank and sparse components.

Theoretical analyses establish conditions under which the proposed convex TRPCA model can perfectly recover the underlying low-rank and sparse components. These recovery guarantees generalize those of traditional robust principal component analysis (RPCA), making them special cases within this broader framework. Extensive numerical experiments validate the efficiency and effectiveness of the proposed algorithms in various applications, including image recovery and background modelling. The results demonstrate that the $*_{L,R}$ TRPCA method outperforms traditional TRPCA based on the c-product, a state-of-the-art approach. The method's high efficiency and scalability make it suitable for processing large volumes of data, such as in hyperspectral imaging and video processing.

However, our proposed algorithm has several notable limitations. Firstly, the subproblems within our algorithm only provide approximate solutions rather than exact ones.

This approximation can lead to reduced accuracy and may result in accumulated errors across multiple subproblems, impacting overall performance. Furthermore, it weakens certain theoretical guarantees that rely on exact solutions. Secondly, the adoption of right-invertible operators instead of fully invertible ones has resulted in less complete algebraic structures compared to those defined using invertible matrices. Right-invertible operators lack some desirable algebraic properties, such as full inverses and certain operational closures, complicating theoretical analysis and limiting the model's flexibility.

Despite these limitations, the $*_{L,R}$ TRPCA method provides a robust and efficient solution for tensor data analysis, maintaining high recovery quality while significantly reducing running time. The reduced computational requirements and running time make the method ideal for resource-limited environments like embedded systems, mobile devices, and remote sensing applications. Its efficiency also supports real-time data processing, valuable for surveillance systems, financial market analysis, and healthcare monitoring. Potential applications of the $*_{L,R}$ TRPCA method in new domains, such as medical imaging, climate modelling, and social network analysis, can be explored in the future. In conclusion, the $*_{L,R}$ TRPCA method offers significant advantages in terms of computational efficiency and scalability, making it a valuable tool for a wide range of applications.

Author Contributions: Conceptualization, Z.H. and J.F.; methodology, Z.H.; software, Z.H.; validation, Z.H. and W.L.; formal analysis, Z.H.; investigation, Z.H.; resources, J.F.; data curation, Z.H.; writing—original draft preparation, Z.H.; writing—review and editing, Z.H.; visualization, Z.H.; supervision, W.L.; project administration, J.F.; funding acquisition, J.F. All authors have read and agreed to the published version of the manuscript.

Funding: Feng Jun's work was supported in part by the Natural Science Foundation of Sichuan Province (Grant Nos. 2023NSFSC0020 and 2024NSFSC0083).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors wish to thank the reviewers for their suggestions and comments that helped to improve the manuscript.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Proofs of Some Results in the Article

Appendix A.1. Proof of Proposition 1

Proof. To show that L is surjective, we need to show that for any $\mathscr{B} \in \mathbb{F}_p^{n_1 \times n_2}$, there exists a $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$ such that $L(\mathscr{A}) = \mathscr{B}$. Consider $\mathscr{A} = \mathscr{B} \times_3 N$. Then, $L(\mathscr{A}) = L(\mathscr{B} \times_3 N) = (\mathscr{B} \times_3 N) \times_3 M = (\mathscr{B} \times_3 MN)$. Using the property $MN = I$, the identity matrix, we obtain $(\mathscr{B} \times_3 N) \times_3 M = \mathscr{B} \times_3 I = \mathscr{B}$. Thus, $L(\mathscr{A}) = \mathscr{B}$, showing that L is surjective.

To show that R is injective, we need to show that if $R(\mathscr{B}) = R(\mathscr{C})$, then $\mathscr{B} = \mathscr{C}$. Assume that $R(\mathscr{B}) = R(\mathscr{C})$. Then, $\mathscr{B} \times_3 N = \mathscr{C} \times_3 N$. Applying the operator L to both sides, we obtain $L(\mathscr{B} \times_3 N) = L(\mathscr{C} \times_3 N)$. We have $(\mathscr{B} \times_3 N) \times_3 M = (\mathscr{C} \times_3 N) \times_3 M$. Using the property $MN = I$, we obtain $\mathscr{B} \times_3 (MN) = \mathscr{C} \times_3 (MN)$ which simplifies to $\mathscr{B} \times_3 I = \mathscr{C} \times_3 I$. Thus, $\mathscr{B} = \mathscr{C}$. This shows that R is injective.

To show that LR is the identity mapping, we need to show that for any $\mathscr{B} \in \mathbb{F}_p^{n_1 \times n_2}$, $LR(\mathscr{B}) = \mathscr{B}$. Consider $LR(\mathscr{B}) = L(\mathscr{B} \times_3 N) = (\mathscr{B} \times_3 N) \times_3 M$. Using the property $MN = I$, we obtain $(\mathscr{B} \times_3 N) \times_3 M = \mathscr{B} \times_3 (N \times_3 M) = \mathscr{B} \times_3 (MN) = \mathscr{B} \times_3 I = \mathscr{B}$. Thus, $LR(\mathscr{B}) = \mathscr{B}$, showing that LR is the identity mapping. □

Appendix A.2. Proof of Proposition 2

Proof. First, we show that $\ker(L) \subseteq \{\mathscr{B} \mid \exists \mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$ such that $\mathscr{B} = (I - RL)(\mathscr{A})\}$. Assuming $\mathscr{B} \in \ker(L)$, then $L(\mathscr{B}) = 0$, and we need to show that there exists an $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$ such that $\mathscr{B} = (I - RL)(\mathscr{A})$. Consider $\mathscr{A} = \mathscr{B}$; then, $(I - RL)(\mathscr{B}) = \mathscr{B} - RL(\mathscr{B})$; since $\mathscr{B} \in \ker(L)$, we have $L(\mathscr{B}) = 0$. Therefore, $(I - RL)(\mathscr{B}) = \mathscr{B} - R(0) = \mathscr{B} - 0 = \mathscr{B}$. Thus, $\mathscr{B} = (I - RL)(\mathscr{B})$, which shows that $\mathscr{B} \in \{\mathscr{B} \mid \exists \mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$ such that $\mathscr{B} = (I - RL)(\mathscr{A})\}$.

On the other hand, we show that $\{\mathscr{B} \mid \exists \mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$ such that $\mathscr{B} = (I - RL)(\mathscr{A})\} \subseteq \ker(L)$. Assume $\mathscr{B} \in \{\mathscr{B} \mid \exists \mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$ such that $\mathscr{B} = (I - RL)(\mathscr{A})\}$. Then, there exists an $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$ such that $\mathscr{B} = (I - RL)(\mathscr{A}) = \mathscr{A} - RL(\mathscr{A})$. Thus, $L(\mathscr{B}) = L((I - RL)(\mathscr{A})) = L(\mathscr{A} - RL(\mathscr{A})) = L(\mathscr{A}) - L(RL(\mathscr{A}))$. By Proposition 1, LR is the identity mapping, so $L(RL(\mathscr{A})) = L(\mathscr{A})$. Thus, $L(\mathscr{B}) = L(\mathscr{A}) - L(\mathscr{A}) = 0$. This shows that $\mathscr{B} \in \ker(L)$.

Therefore, we have $\ker(L) = \{\mathscr{B} \mid \exists \mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times n_2}$ such that $\mathscr{B} = (I - RL)(\mathscr{A})\}$. This completes the proof. □

Appendix A.3. Proof of Proposition 3

Proof. According to the definition, L maps $\mathscr{A} \in \mathbb{F}_{n_3}^{n_1 \times r}$ to $\mathbb{F}_p^{n_1 \times r}$ and $\mathscr{B} \in \mathbb{F}_{n_3}^{r \times n_2}$ to $\mathbb{F}_p^{r \times n_2}$. Therefore, $L(\mathscr{A}) \in \mathbb{F}_p^{n_1 \times r}$ and $L(\mathscr{B}) \in \mathbb{F}_p^{r \times n_2}$. By definition, $\triangle$ is a matrix multiplication operation, so $L(\mathscr{A}) \triangle L(\mathscr{B}) \in \mathbb{F}_p^{n_1 \times n_2}$. Applying R to map the result back from $\mathbb{F}_p^{n_1 \times n_2}$ to $\mathbb{F}_{n_3}^{n_1 \times n_2}$, we have $R(L(\mathscr{A}) \triangle L(\mathscr{B})) \in \mathbb{F}_{n_3}^{n_1 \times n_2}$. By definition, $(\mathscr{A} *_{L,R} \mathscr{B})_{i,j}$ is the (i,j)-th tube fibre of $\mathscr{A} *_{L,R} \mathscr{B}$. We can express it as $(\mathscr{A} *_{L,R} \mathscr{B})_{i,j} = R(L(\mathscr{A}) \triangle L(\mathscr{B}))_{i,j}$. According to the properties of matrix multiplication, the (i,j)-th element of $L(\mathscr{A}) \triangle L(\mathscr{B})$ can be written as $(L(\mathscr{A}) \triangle L(\mathscr{B}))_{i,j} = \sum_{k=1}^{r} (L(\mathscr{A}))_{i,k} \cdot (L(\mathscr{B}))_{k,j}$. Apply R to each term in the sum $R((L(\mathscr{A}))_{i,k} \cdot (L(\mathscr{B}))_{k,j}) = R(L(\mathbf{a}_{ik}) \cdot L(\mathbf{b}_{kj}))$. By the definition of the $*_{L,R}$ product, we have $R(L(\mathbf{a}_{ik}) \cdot L(\mathbf{b}_{kj})) = \mathbf{a}_{ik} *_{L,R} \mathbf{b}_{kj}$. Therefore, the (i,j)-th tube fibre of $\mathscr{A} *_{L,R} \mathscr{B}$ can be expressed as $(\mathscr{A} *_{L,R} \mathscr{B})_{i,j} = \sum_{k=1}^{r} \mathbf{a}_{ik} *_{L,R} \mathbf{b}_{kj}$. This completes the proof. □

Appendix A.4. Proof of Lemma 1

Proof. **Associativity**: Let $\mathscr{A}, \mathscr{B}, \mathscr{C}$ be third-order tensors in $\mathbb{F}^{n_1 \times n_2 \times n_3}$; then, by the linear properties of operators L and R and the fact that LR is the identity operator, we have

$$\begin{aligned} \mathscr{A} *_{L,R} (\mathscr{B} *_{L,R} \mathscr{C}) &= R(L(\mathscr{A}) \triangle R(L(\mathscr{B}) \triangle L(\mathscr{C}))) \\ &= R(L(\mathscr{A}) \triangle L(\mathscr{B}) \triangle L(\mathscr{C})) \\ &= R(L(R(L(\mathscr{A}) \triangle L(\mathscr{B}))) \triangle L(\mathscr{C})) \\ &= (\mathscr{A} *_{L,R} \mathscr{B}) *_{L,R} \mathscr{C} \end{aligned}$$

Distributivity:

$$\begin{aligned} \mathscr{A} *_{L,R} (\mathscr{B} + \mathscr{C}) &= R(L(\mathscr{A}) \triangle L(\mathscr{B} + \mathscr{C})) \\ &= R(L(\mathscr{A}) \triangle (L(\mathscr{B}) + L(\mathscr{C}))) \\ &= R(L(\mathscr{A}) \triangle L(\mathscr{B}) + L(\mathscr{A}) \triangle L(\mathscr{C})) \\ &= R(L(\mathscr{A}) \triangle L(\mathscr{B})) + R(L(\mathscr{A}) \triangle L(\mathscr{C})) \\ &= \mathscr{A} *_{L,R} \mathscr{B} + \mathscr{A} *_{L,R} \mathscr{C} \end{aligned}$$

Similarly,

$$\begin{aligned}(\mathscr{B}+\mathscr{C})*_{L,R}\mathscr{A} &= R((L(\mathscr{B})+L(\mathscr{C}))\triangle L(\mathscr{A}))\\ &= R(L(\mathscr{B})\triangle L(\mathscr{A})+L(\mathscr{C})\triangle L(\mathscr{A}))\\ &= R(L(\mathscr{B})\triangle L(\mathscr{A}))+R(L(\mathscr{C})\triangle L(\mathscr{A}))\\ &= \mathscr{B}*_{L,R}\mathscr{A}+\mathscr{C}*_{L,R}\mathscr{A}\end{aligned}$$

Noncommutativity: In general, the $*_{L,R}$ product is not commutative. This is because $L(\mathscr{A})\triangle L(\mathscr{B}) \neq L(\mathscr{B})\triangle L(\mathscr{A})$ in most cases, leading to $\mathscr{A}*_{L,R}\mathscr{B} \neq \mathscr{B}*_{L,R}\mathscr{A}$. □

Appendix A.5. Proof of Theorem 1

Proof. Recall the definition of the $*_{L,R}$ product, $\mathscr{A}*_{L,R}\mathscr{B} = R(L(\mathscr{A})\triangle L(\mathscr{B}))$, where L is a linear operator and R is its right-invertible operator. For tube tensors $\mathscr{A},\mathscr{B} \in \mathbb{F}^{1\times 1\times n_3}$, let M, N be the matrix of the linear operator L and the right-invertible operator R, respectively; then, $\mathscr{A}*_{L,R}\mathscr{B} = R(L(\mathscr{A})\triangle L(\mathscr{B})) = unvec(N(Mvec(A)\triangle Mvec(B)))$. For vectors, the operation $\triangle$ is typically a component-wise multiplication for these vectors. Since component-wise multiplication is commutative, we have

$$(Mvec(A))\triangle(Mvec(B)) = (Mvec(B))\triangle(Mvec(A)).$$

that is, $L(\mathscr{A})\triangle L(\mathscr{B}) = L(\mathscr{B})\triangle L(\mathscr{A})$. Applying the right-invertible operator R to both sides, we obtain $R(L(\mathscr{A})\triangle L(\mathscr{B})) = R(L(\mathscr{B})\triangle L(\mathscr{A}))$. By the definition of the $*_{L,R}$ product, this implies $\mathscr{A}*_{L,R}\mathscr{B} = \mathscr{B}*_{L,R}\mathscr{A}$. Thus, the $*_{L,R}$ product is commutative for scalar tensors in $\mathbb{F}^{1\times 1\times n_3}$. □

Appendix A.6. Proof of Proposition 4

Proof. By definition, we have

$$\begin{aligned}L(\mathscr{A}^H *_{L,R}\mathscr{B}^H)^{(i)} &= ((L(\mathscr{A})^{(i)})^H(L(\mathscr{B})^{(i)})^H)\\ &= (L(\mathscr{B})^{(i)}L(\mathscr{A})^{(i)})^H\\ &= (L(\mathscr{B}*_L\mathscr{A})^{(i)}).\end{aligned}$$

The result follows from equality in the non-invertible transformed domain by definition. □

Appendix A.7. Proof of Theorem 4

Proof. Given that the $*_{L,Ra}$ SVD of $\mathscr{A}$ has k nonzero singular values in $\mathscr{A}(:,:,1)$, that is, $\text{rank}_{*_{L,Rt}}(\mathscr{A}) = \text{card}(\{i \mid \mathscr{S}(i,i,1) \neq 0\}) = k$, the inequality of the singular value of each frontal slice $\mathscr{S}(:,:,i)$ decreases. Therefore, for each i, we have:

$$\text{rank}_{*_{L,Rt}}(\mathscr{A}) \geq \text{rank}(\mathscr{A}(:,:,i))$$

Summing the above inequality over all i and taking the average, we obtain

$$\frac{1}{n_3}\sum_{i=1}^{n_3}\text{rank}_{*_{L,Rt}}(\mathscr{A}) = \text{rank}_{*_{L,Rt}}(\mathscr{A}) \geq \frac{1}{n_3}\sum_{i=1}^{n_3}\text{rank}(\mathscr{A}(:,:,i))$$

Thus, we have

$$\text{rank}_{*_{L,Rt}}(\mathscr{A}) \geq \text{rank}_{*_{L,Ra}}(\mathscr{A})$$

□

Appendix A.8. Proof of Theorem 5

Proof. Assume that the CP decomposition of the tensor $\mathscr{A}$ is $\mathscr{A} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$, where R is the CP rank Consider the i-th frontal slice $L(\mathscr{A})^{(i)}$ of the tensor $\mathscr{A}$ in the non-invertible transformed domain. According to the CP decomposition, each frontal slice can be expressed as $L(\mathscr{A})^{(i)} = \sum_{r=1}^{R} \lambda_r L(\vec{c}_r)_i \vec{a}_r \vec{b}_r^T$. Since $L(\mathscr{A})^{(i)}$ is a linear combination of R rank-1 matrices, the rank of $L(\mathscr{A})^{(i)}$ is at most R:

$$\operatorname{rank}(L(\mathscr{A})^{(i)}) \leq R.$$

According to the definition of $*_{L,R}$ tensor average rank, we have $\operatorname{rank}_{*_{L,Ra}}(\mathscr{A}) = \frac{1}{p}\sum_{i=1}^{p} \operatorname{rank}(L(\mathscr{A})^{(i)})$. Since each $L(\mathscr{A})^{(i)}$ has a rank of at most R, we obtain

$$\operatorname{rank}_{*_{L,Ra}}(\mathscr{A}) \leq \frac{1}{p}\sum_{i=1}^{p} R = R.$$

Therefore, we have $\operatorname{rank}_{*_{L,Ra}}(\mathscr{A}) \leq \operatorname{rank}_{\mathrm{CP}}(\mathscr{A})$. This completes the proof. □

Appendix A.9. Proof of Theorem 6

Proof. By using the properties of the Frobenius norm and the tensor nuclear norm under orthogonal operators:

$$\|\mathbf{A}\|_F = \|\bar{\mathbf{A}}\|_F,$$

and

$$\|\mathscr{A}\|_* = \|\tilde{\mathscr{A}}\|_*,$$

where $\bar{\mathbf{A}}$ and $\tilde{\mathscr{A}}$ are the matrices and tensors obtained through the orthogonal operators L and R.

Since L and R are orthogonal operators, they preserve the Frobenius norm and the nuclear norm, i.e.,

$$\|L\mathscr{A}\|_F = \|\mathscr{A}\|_F \quad \text{and} \quad \|R\mathscr{A}\|_F = \|\mathscr{A}\|_F,$$

and

$$\|L\mathscr{A}\|_* = \|\mathscr{A}\|_* \quad \text{and} \quad \|R\mathscr{A}\|_* = \|\mathscr{A}\|_*.$$

Therefore, the original problem can be written as follows:

$$\arg\min_{\mathscr{X}} \left(\tau \|\mathscr{X}\|_* + \frac{1}{2}\|\mathscr{X} - \mathscr{Y}\|_F^2 \right).$$

By Theorem 2.1 in Reference [7], the i-th frontal slice of $\mathscr{D}_\tau(\mathscr{Y})$ solves the i-th subproblem of the above minimisation problem. Hence, $\mathscr{D}_\tau(\mathscr{Y})$ solves the entire problem. □

Appendix A.10. Proof of Theorem 7

Proof. Let $M_{p\times n_3}$ be the matrix corresponding to the truncated orthogonal operator L. The condition $\| \frac{(\mathscr{A}-\mathscr{Y})\times_3 M\|_F^2}{(\mathscr{A}-\mathscr{Y})\|_F^2} \approx 1$ implies that

$$\begin{aligned} \tau\|\mathscr{Y}\|_{*_{L,R}} + \frac{1}{2}\|\mathscr{A} - \mathscr{Y}\|_F^2 &= \tau\|\mathscr{Y} \times_3 M\|_* + \frac{1}{2}\|\mathscr{A} - \mathscr{Y}\|_F^2 \\ &\approx \tau\|\mathscr{Y} \times_3 M\|_* + \frac{1}{2}\|(\mathscr{A} - \mathscr{Y}) \times_3 M\|_F^2. \end{aligned}$$

By Theorem 6, we have

$$R(\mathscr{D}_\tau(L(\mathscr{A}))) = \arg\min_{\mathscr{Y}\in\mathbb{F}_{n_3}^{n_1\times n_2}} \left\{ \tau\|\mathscr{Y}\times_3 M\|_* + \frac{1}{2}\|(\mathscr{A}-\mathscr{Y})\times_3 M\|_F^2 \right\}$$
$$\approx \arg\min_{\mathscr{Y}\in\mathbb{F}_{n_3}^{n_1\times n_2}} \left\{ \tau\|\mathscr{Y}\|_{*_{L,R}} + \frac{1}{2}\|\mathscr{A}-\mathscr{Y}\|_F^2 \right\}.$$

□

References

1. Candès, E.J.; Li, X.D.; Ma, Y.; Wright, J. Robust principal component analysis? *J. ACM* **2011**, *58*, 1–37. [CrossRef]
2. Liu, G.; Lin, Z.; Yan, S.; Sun, J.; Yu, Y.; Ma, Y. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 171–184. [CrossRef] [PubMed]
3. Lu, C.; Feng, J.; Chen, Y.; Liu, W.; Lin, Z.; Yan, S. Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 5249–5257
4. Netrapalli, P.; Niranjan, U.N.; Sanghavi, S.; Anandkumar, A.; Jain, P. Non-convex Robust PCA. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 5443–5457.
5. Lu, C.; Peng, X.; Wei, Y. Low-rank tensor completion with a new tensor nuclear norm induced by invertible linear transforms. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 5989–5997.
6. Geng, X.; Guo, Q.; Hui, S.; Yang, M.; Zhang, C. Tensor robust PCA with nonconvex and nonlocal regularization. *Comput. Vis. Image Underst.* **2024**, *243*, 104007. [CrossRef]
7. Cai, J.F.; Candès, E.J.; Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.* **2010**, *20*, 1956–1982. [CrossRef]
8. Fazel, M. Matrix Rank Minimization with Applications. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 2002.
9. Mu, C.; Huang, B.; Wright, J.; Goldfarb, D. Square Deal: Lower Bounds and Improved Relaxations for Tensor Recovery. In Proceedings of the 31st International Conference on Machine Learning (ICML), Beijing, China, 21–26 June 2014; pp. 73–81.
10. Wright, J.; Ganesh, A.; Rao, S.; Peng, Y.; Ma, Y. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. *Adv. Neural Inf. Process. Syst.* **2009**, *22*, 16977–16989.
11. Cai, H.; Liu, J.; Yin, W. Learned Robust PCA: A Scalable Deep Unfolding Approach for High-Dimensional Outlier Detection. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Online, 6–14 December 2021; pp. 16977–16989.
12. Kolda, T.G.; Bader, B.W. Tensor Decompositions and Applications. *SIAM Rev.* **2009**, *51*, 455–500. [CrossRef]
13. Kilmer, M.E.; Braman, K.; Hao, N.; Hoover, R.C. Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. *SIAM J. Matrix Anal. Appl.* **2013**, *34*, 148–172. [CrossRef]
14. Zhang, A.; Xia, D. Tensor SVD: Statistical and computational limits. *IEEE Trans. Inf. Theory* **2018**, *64*, 7311–7338. [CrossRef]
15. Lu, C.; Feng, J.; Chen, Y.; Liu, W.; Lin, Z.; Yan, S. Tensor robust principal component analysis with a new tensor nuclear norm. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 925–938. [CrossRef] [PubMed]
16. Xu, Z.; Yang, J.H.; Wang, L.C.; Wang, F.; Yan, H.X. Tensor robust principal component analysis with total generalized variation for high-dimensional data recovery. *Appl. Math. Comput.* **2024**, *483*, 128980. [CrossRef]
17. Tang, K.; Fan, Y.; Song, Y. Improvement of robust tensor principal component analysis based on generalized nonconvex approach. *Appl. Intell.* **2024**, *54*, 7377–7396. [CrossRef]
18. Kilmer, M.E.; Martin, C.D. Factorization strategies for third-order tensors. *Linear Algebra Appl.* **2011**, *435*, 641–658. [CrossRef]
19. Zhu, Q.L.; Wu, S.Q.; Fang, S.; Wu, Q.; Xie, S.L.; Agaian, S. Fast tensor robust principal component analysis with estimated multi-rank and Riemannian optimization. *Appl. Intell.* **2025**, *55*, 1–16. [CrossRef]
20. Cai, J.-F.; Li, J.; Xia, D. Generalized Low-Rank Plus Sparse Tensor Estimation by Fast Riemannian Optimization. *J. Am. Stat. Assoc.* **2022**, *117*, 2588–2604. [CrossRef]
21. Keegan, K.; Newman, E. Projected tensor-tensor products for efficient computation of optimal multiway data representations. *arXiv* **2024**, arXiv:2409.19402.

22. Kernfeld, E.; Kilmer, M.; Aeron, S. Tensor-tensor products with invertible linear transforms. *Linear Algebra Appl.* **2015**, *481*, 545–570. [CrossRef]
23. Lu, C.; Zhu, C.; Xu, C.; Yan, S.; Lin, Z. Generalized singular value thresholding. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, Austin, TX, USA, 25–30 January 2015; AAAI Press: Stanford, CA, USA, 2015; pp. 1805–1811.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.