Effective Tensor-Tensor Product-Based Tensor Recovery and Its Efficient Non-Convex Optimization Framework

by

© *Jingjing Zheng*

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Doctor of Philosophy

Department of *Computer Science*

Memorial University of Newfoundland

June 2023

St. John's Newfoundland

Abstract

Low-rank tensor recovery methods have increasingly received attention due to their successful applications in dimensionality reduction and data analysis. A fundamental problem is often asked: how to define tensor rank effectively and reasonably? Several tensor rank defining ways (such as the CP rank, the Tucker rank, and the tensor product-based rank) have been explored to answer this question. Among them, the tensor product-based defining way has recently become increasingly popular because of its natural generalization of the matrix-matrix product. Although the tensor product-based rank has successfully studied the low rankness within the tensor data, there are still two challenges. (1) Transpose Invariance and Slice Permutations Invariance of the tensor product-based rank do not hold, causing the performance degradation in tensor recovery and limiting its applications in the real world. (2) Since the calculating tensor singular value decomposition (t-SVD) is required in tensor recovery with the product-based rank, leading to high computational cost for a large tensor. In this dissertation, we focus on solving these two problems, and the main contributions are summarized as follows:

- We analyze Transpose Variability in tensor recovery from the view of theory and experiment. The Weighted Tensor Average Rank is proposed and applied to the third-order tensor robust principal component analysis to eliminate the transpose variability. The experimental results indicate that the proposed method can more exactly explore the low dimensional structure within the tensor data.

- We study Slice Permutations Variability (SPV) in tensor recovery from the view of theory and experiment. We propose a novel tensor recovery algorithm by Minimum

Hamiltonian Circle for SPV to handle slice permutation variability. The experimental results demonstrate the effectiveness of the proposed algorithm in eliminating SPV.

- Besides, we propose a novel tensor recovery framework with a developing rank estimation strategy that utilizes a dual low-rank constraint, reducing total cost at each iteration of the developing algorithm to $\mathcal{O}(N^3 \log N + \kappa N^3)$ from $\mathcal{O}(N^4)$ achieved with standard methods, where κ was the estimation of tensor rank and far less than N. The experiments on the synthetic and real-world data demonstrate the effectiveness and efficiency of the proposed method.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisors, Dr. Xianta Jiang and Dr. Xiaoqin Zhang, for their unwavering support and invaluable guidance throughout my academic journey.

I would also like to extend my appreciation to all the members of the supervisory committee (Dr. Xianta Jiang, Dr. Xiaoqin Zhang, Dr. Yuanzhu Chen, Dr. Antonina Kolokolova) and the examiners who dedicated their time to review my dissertation and participate in my defense. Their constructive feedback and guidance have significantly contributed to the development and refinement of this dissertation.

Furthermore, I would like to acknowledge the valuable assistance of my colleagues, including Shuo Wang, Mohammed Asfour, Wenting Hu, Wenzhe Wang, Ziwei Huang, and Wanglong Lu. Their support and collaboration have been instrumental in shaping my research. I would also like to express my deepest appreciation to my friends, Ruixin Song, Xiaomiao Song, and Shuaishuai Li, for the wonderful moments we shared in Newfoundland. Your companionship and joyful experiences have made my journey more fulfilling.

Lastly, I would like to dedicate a special thank you to my family. Their unwavering support and encouragement have been my pillar of strength throughout this academic endeavor. Obtaining my Ph.D. degree would not have been possible without their love and belief in my abilities.

I am deeply grateful to everyone mentioned above for their contributions, and encouragement. Without their support, this achievement would not have been possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

With the rapid advances of data-intensive applications in various engineering and scientific fields, there is a growing explosion of high-dimensional data, including images and videos, which are difficult to store, transmit, and process. Therefore, many low-rank matrix methods have been proposed for efficiently handling and understanding such complicated data by exploiting low-dimensional structures in such high-dimensional data [9, 10, 8, 12, 83, 77, 21, 76, 103].

Principal Component Analysis (PCA) [21, 76] was first proposed and widely used for dimension reduction and data analysis. The traditional PCA, which employs the Frobenius norm, is robust against small noise perturbations but sensitive to gross sparse errors. Consequently, when the data is corrupted by gross sparse errors, PCA fails to work [8, 12]. To solve this issue, Robust PCA (RPCA) [8, 12] was proposed, which employs ℓ_0-norm (*i.e.*, the number of non-zero entries in a matrix) to quantify the gross sparse errors present in the data. These matrix-based low-rank methods have achieved remarkable success in various applications, including dimension reduction, as well as image and video processing.

Currently, massive amounts of high-dimensional data, including images, videos, hyperspectral data, and 3-D range data, have become available due to dramatic advances in hardware for data [14]. Such real-world data/signals are naturally represented as multidimensional arrays (known as tensors[1]). An easy way to deal with such tensor data is first to transform the data tensor into 2D matrices, then perform the matrix-based methods on the matrices. However, as Liu *et al.* [45] points out, the essential structures in the tensor data will be lost when a higher-order tensor is transformed into a 2D matrix. Exploiting low-dimensional structures in such tensor data in an effective way has become increasingly important.

1.2 Low-Rank Tensor Recovery

In recent years, many low-rank tensor recovery methods have been proposed [25, 45, 50, 89, 101, 86, 85, 7, 52]. Different with matrix case, as there is no unified defining way for tensor rank, how to define a tensor rank appropriately is an important problem in low-rank tensor recovery. General speaking, there are three common ways to define the tensor rank functions: 1) the CANDECOMP/PARAFAC (CP) rank [40, 43, 11], 2) Tucker rank [43, 17], and 3) tensor tubal rank [47, 30]. Inspired by the definition of the matrix rank, Kiers [40] defined the CP rank of the tensor $\mathcal{X}$ as the minimum number of rank-one decomposition. However,

[1] In this thesis, the tensor is the generalization of the matrix to the higher order. For example, the color image can be regarded as a third-order tensor because of its RGB channels. Vector and matrix can be regarded as first-order and second-order tensors, respectively.

computing CP rank for a given tensor $\mathcal{X}$ is NP-hard [50], limiting the application of CP rank in the real world. In addition, due to the breakthroughs in low-rank matrix recovery [9, 10, 8, 12, 83, 77, 21, 76, 103], the method based on Tucker decomposition (the unfolding matrices of the tensor) becomes more popular than the one based on CP decomposition. For example, in [25], the rank of the tensor (Tucker rank) was defined as the sum of the ranks of the different unfolding matrices. Besides, since the corresponding tensor rank minimization problem is an NP-hard problem, Gandy *et al.* utilized the sum of nuclear norms of the different unfolding matrices (SNN) instead of the sum of ranks for tensor recovery. However, as stated in [50], SNN is not the convex envelope of the sum of the ranks. Therefore, a weighted sum of the ranks of the unfolding matrices is considered in [45]. Since the tensor recovery methods based on the weighted sum of ranks suffer from the high computation cost of the computing of singular value decompositions (SVDs) for the large unfolding matrices, an efficient matrix factorization method for tensor recovery is developed in [49].

More recently, tensor nuclear norm (TNN) based on the tensor-tensor product (t-product) has attracted more attention because of its effectiveness in tensor recovery, particularly for tensor completion [96, 51] and tensor robust principal component analysis [50]. The resulting TNN-based models can exactly recover the true value of the problem under some conditions as stated in [96, 51, 50]. However, the conditions are hardly satisfied in the real world. In addition, since the information of the data is concentrated in the components corresponding to a few largest singular values [64, 50], the larger singular values should be penalized mildly, and the smaller ones should be penalized heavily. Whereas the TNN-based methods treat the singular values with an equal penalty, leading to the over-penalization for large singular values and therefore suffering from performance degradation. Therefore, many low-rank recovery methods with non-convex surrogates of ℓ_0-norm have been proposed to solve this issue [74, 38, 84], in which the rank function and nuclear norm can be regarded as ℓ_0-norm and ℓ_1-norm of a singular vector, respectively. Since computing t-SVD in each iteration is required, these methods cost much computation and cannot be used to handle large-scale tensor data efficiently, where the computational complexity of t-SVD for a tensor with the size of $I_1 \times I_2 \times I_3$ is $\mathcal{O}(I_{(1)} I_{(2)}^2 I_3 + I_1 I_2 I_3 \log I_3)$, $I_{(1)} = \max(I_1, I_2)$ and $I_{(2)} = \min(I_1, I_2)$. To address this issue, Zhou *et al.* [102] have proposed Tensor Completion by Tensor Factorization (TCTF), which achieved significant improvement in terms of running time if the given tensor data is low-rankness, and their proposed rank-decreasing scheme can estimate the t-product-based tensor rank precisely.

1.3 Motivations

Although the t-product-based defining way for tensor rank is getting increasingly popular and has achieved great success, there are still several challenges as follows: (1) as we will discuss in this dissertation, two important properties (including Transpose Invariance and Slice Permutations Invariance) to the t-product-based tensor rank that do not hold. Because the t-product-based tensor recovery does not satisfy the Transpose Invariance, some information within the data will be lost if only the low rankness of tensor data along with one direction is considered. For slice permutation invariance, it is derived from a reasonable assumption about the algorithm, *i.e.*, changing data order should not affect the effectiveness of the algorithm. We call these two interesting problems as Transpose Variability (TV) and Slice Permutations Variability (SPV) in tensor recovery, respectively. (2) The TCTF [102] is based on a basic hypothesis that the low-rank tensor can be approximatively decomposed to the t-product of two skinny tensors $\mathcal{A} \in \mathbb{R}^{I_1 \times \kappa \times I_3}$ and $\mathcal{B} \in \mathbb{R}^{\kappa \times I_2 \times I_3}$ (κ is an estimation of the t-product-based tensor rank), causing its over-reliance on the rank estimation strategy. On the other hand, because of the lacking a rank-increasing scheme, the rank estimation strategy given in [102] often underestimates the true rank, which leads to performance degradation in TCTF. (3) Currently, the common non-convex surrogate functions of the ℓ_0-norm, including ℓ_p-norm $(0 < p < 1)$ [23], ETP [26], Geman [27], Laplace [71] and Logarithm [24], have been widely applied in the field of low-rank recovery and achieved more satisfied performance [29, 61, 53, 89, 82]. But these non-convex low-rank recovery methods require calculating tensor singular value decomposition (t-SVD), leading to high computational cost for a large tensor. An efficient and effective tensor completion framework for a wide range of surrogate functions is necessary to address the over-penalization problem in the TNN-based methods and reduce the computational cost caused by t-SVD for the large tensor simultaneously.

1.4 Contributions

In this dissertation, two questions are mainly asked: (i) How to define an effective tensor rank based on the tensor-tensor product? (ii) How to solve the low-rank tensor recovery model effectively and efficiently?

To solve the above problems, we give a new tensor rank called Weighted Tensor Average Rank (WTAR) and a novel algorithm to eliminate transpose variability and slice permutation variability in tensor recovery, respectively. Besides, we propose a novel tensor recovery method with a dual low-rank constraint strategy, which aims to avoid the high computational cost in the standard t-SVD-based method, and to achieve superior recovery results simultaneously.

The main contributions of this dissertation are three-folds as follows:

- We initially explore an intriguing phenomenon known as TV (Tensor Variability). To address TV in tensor recovery, we introduce a novel tensor rank called Weighted Tensor Average Rank (WTAR), which enables us to analyze the low-rank structure of tensor data from different dimensions of tensor data. We apply WTAR to the third-order tensor robust principal component analysis to investigate its effectiveness. The experimental results indicate that the proposed method is effective. The findings related to this research have been published in an international journal [93].

- We discuss the SPV of several critical tensor recovery problems theoretically and experimentally. The conclusion shows a vast gap between results by tensor recovery for tensor data with different slice sequences. To overcome SPV in DFT-based tensor recovery, we develop a novel tensor recovery algorithm by Minimum Hamiltonian Circle for SPV (TRSPV), exploiting low dimensional subspace structures within data tensor more exactly, which work has been published in an international conference [100]. Furthermore, we extend our study to explore SPV in other t-product-based methods and propose a general solution to mitigate SPV in such methods. To the best of our knowledge, this is the first work to extensively discuss SPV in the context of tensor recovery and provide an effective solution to mitigate its impact. The experimental results demonstrate the effectiveness of the proposed algorithm in eliminating SPV in tensor recovery.

- We propose a novel tensor completion framework that aims to overcome the reliance on rank estimation strategies used in the standard tensor factorization-based tensor recovery and tackle the computational burden associated with the standard t-SVD-based tensor recovery. To this end, we propose a new tensor norm with a dual low-rank constraint, which utilizes the low tensor average rank prior and tensor tubal rank information at the same time. In the proposed tensor norm, a series of surrogate functions of the tensor tubal rank can be used to achieve better performance in harness low-rankness within tensor data. It is proven theoretically that the resulting tensor completion model can effectively avoid performance degradation caused by inaccurate rank estimation. Meanwhile, attributed to the proposed dual low-rank constraint, the t-SVD of a smaller tensor instead of the original big one is computed by using a sample trick. Based on this, the total cost at each iteration of the optimization algorithm is reduced to $\mathcal{O}(N^3 \log N + \kappa N^3)$ from $\mathcal{O}(N^4)$ achieved with standard methods, where κ is the estimation of the true tensor rank and far less than N. Our method was evaluated on synthetic and real-world data, and it demonstrated superior performance and efficiency over several existing state-of-the-art tensor completion methods. This study has been published as a preprint on arXiv [99].

1.5 Organization

The dissertation contains seven chapters and one appendix. Each of the chapters is described below.

- Chapter 1 mainly gives an introduction to the low-rank tensor recovery, and summarizes the motivation and contribution as well.
- Chapter 2 provides a comprehensive overview of the related work in this paper, organized into four main parts. (1) The first part introduces various tensor rank functions,

discussing their properties and applications. (2) The second part focuses on several prominent low-rank tensor recovery models. It presents a detailed explanation of Tensor Principal Component Analysis (TPCA), Tensor Robust Principal Component Analysis (TRPCA), and Tensor Completion (TC). (3) The third part delves into optimization algorithms commonly used in tensor recovery. It covers essential methods such as the Tensor Singular Value Threshold (TSVT), Generalized Tensor Singular Value Threshold (GTSVT), Block Coordinate Descent (BCD), and Alternating Direction Method of Multipliers (ADMM). (4) The final part introduces three representative applications of the low-rank tensor recovery models in computer vision.

- Chapter 3 delves into an in-depth exploration of Total Variation (TV) in tensor recovery. It introduces the Weighted Tensor Average Rank (WTAR) as a novel approach to address TV in tensor recovery. The WTAR is specifically applied to the third-order tensor robust principal component analysis, allowing for an investigation of its effectiveness in studying the low-rankness present in the tensor data.

- Chapter 4 focuses on the investigation of Slice Permutation Variability (SPV) in Discrete Fourier Transformation (DFT)-based tensor recovery, which is a specific case of a special case of t-product-based tensor recovery. Specifically, we discuss SPV in several key DFT-based tensor recovery problems theoretically and experimentally. A novel algorithm called TRSPV. This algorithm is specifically applied to third-order tensor robust principal component analysis to investigate its effectiveness.

- Chapter 5 studies the issue of SPV in other t-product-based tensor recovery further. Specifically, we deeply analysis SPV in three cases in t-product-based tensor recovery, including DFT-based methods, Discrete Cosine Transform (DCT)-based methods, and Random Orthogonal Matrix (ROM)-based methods experimentally. We provide a general solver to overcome the issue of SPV in t-product-based tensor recovery. The experimental results demonstrate the effectiveness of the proposed algorithm in eliminating SPV.

- Chapter 6 gives a novel efficiency and effective tensor recovery framework with a developed rank estimation method that involves the convex surrogate and a series of non-convex surrogates. We compare several state-of-art methods on tensor completion problems to investigate the effectiveness of the proposed method.
- Chapter 7 gives the conclusions and future works.
- Appendix presents some definitions and symbols used in this dissertation. Some tensor computations and the definitions of specific tensors are given in A.1 and A.2, respectively.

1.6 Author Contributions

- Jingjing Zheng, Xiaoqin Zhang*, Wenzhe Wang, Xianta Jiang. Handling Slice Permutations Variability in Tensor Recovery. *AAAI Conference on Artificial Intelligence*, 2022

  Author Contributions: investigation and methodology: Jingjing Zheng; writing—original draft: Jingjing Zheng; experiments: Jingjing Zheng, Wenzhe Wang; writing—review and editing: Jingjing Zheng, Xianta Jiang, Xiaoqin Zhang; funding acquisition: Xianta Jiang, Xiaoqin Zhang.

- Xiaoqin Zhang*, Jingjing Zheng, Li Zhao, Zhengyuan Zhou, Zhouchen Lin. Tensor Recovery With Weighted Tensor Average Rank. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

  Author Contributions: investigation and methodology: Jingjing Zheng; writing—original draft: Jingjing Zheng; experiments: Jingjing Zheng; writing—review and editing: Xiaoqin Zhang, Jingjing Zheng, Li Zhao, Zhengyuan Zhou, Zhouchen Lin; funding acquisition: Xiaoqin Zhang.

- Jingjing Zheng, Wenzhe Wang, Xiaoqin Zhang*, Xianta Jiang. A Novel Tensor Factorization-Based Method with Robustness to Inaccurate Rank Estimation, *arXiv: 2305.11458*, 2023. It corresponds to the content in Chapter 6.

  Author Contributions: investigation: Jingjing Zheng, Wenzhe Wang; methodology: Jingjing Zheng; writing—original draft: Jingjing Zheng, Wenzhe Wang; experiments: Wenzhe Wang, Jingjing Zheng; writing—review and editing: Jingjing Zheng, Xianta Jiang, Xiaoqin Zhang; funding acquisition: Xiaoqin Zhang, Xianta Jiang.

Chapter 2

Related works

In recent years, a massive amount of high-dimensional data, including images, videos, hyper-spectral data, and 3-D range data, became available due to the dramatic advance in hardware for data [14]. The real-world data or signals are often naturally represented as multidimensional arrays, which are commonly referred to as tensors. These tensors often lie in some low-dimensional sub-spaces or manifolds approximately. To exploit such low-dimensional structures in tensor data, low-rank tensor recovery methods have been proposed and widely applied in color images and videos denoising [50], image inpainting [78, 66], video background modeling [8, 98, 50], and hyperspectral image restoration [25].

Exploiting low-dimensional structures in such tensor data has become increasingly important. As a powerful computational tool for tensor analysis, low-rank tensor recovery has been widely applied in color images and videos denoising [50], image inpainting [78, 66], video background modeling [8, 98, 50], and hyperspectral image restoration [25].

In this chapter, I will introduce the three different ways of defining the tensor rank including CP rank, Tucker rank, and tensor product-based rank first. Then, I will introduce

Table 2.1: Notations

Notations	Descriptions	Notations	Descriptions
$\mathbb{R}$	real field	$\mathbb{C}$	complex field
$\mathbb{A}, \mathbb{B}, \mathbb{D}, \cdots$	sets	$\lvert\mathbb{A}\rvert$	the number of elements of set $\mathbb{A}$
$a, b, c, \cdots$	scalars	$\mathrm{Im}(a)$, $\mathrm{Re}(a)$, $\mathrm{Conj}(a)$	imaginary part of a, real part of a, and conjugate of a, respectively
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \cdots$	vectors	$[\boldsymbol{a}]_i, a_i$	i-th element of $\boldsymbol{a}$
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \cdots$	matrices	$a_{i,j}$, $[\boldsymbol{A}]_{i,j}$	(i, j)-th element of matrix $\boldsymbol{A}$
$[\boldsymbol{A}]_{i,:}$, $\boldsymbol{a}_{i,:}$	the i-th row vector of $\boldsymbol{A}$	$[\boldsymbol{A}]_{:,j}$, $\boldsymbol{a}_{:,j}$	the j-th column vector of $\boldsymbol{A}$
$\mathbf{0}$	null tensor	$\boldsymbol{I}$	identity matrix
$\boldsymbol{F}_N$	$N \times N$ DFT matrix	$\boldsymbol{A}^T$	conjugate transpose of $\boldsymbol{A}$
$\boldsymbol{A} \longrightarrow \boldsymbol{B}$	$\boldsymbol{B}$ can be obtained by elementary row or column transformations of $\boldsymbol{A}$	$\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \cdots$	tensors
$\bar{\boldsymbol{\mathcal{A}}}$	the result of DFT on $\boldsymbol{\mathcal{A}}$ along the 3-rd dimension	$[\boldsymbol{\mathcal{A}}]_{i_1,i_2,\cdots,i_h}$, $a_{i_1,i_2,\cdots,i_h}$	$(i_1, i_2, \cdots, i_h)$-th element in $\boldsymbol{\mathcal{A}}$
$[\boldsymbol{\mathcal{A}}]_{i_1,i_2,:}$	(i_1, i_2)-th tube	$[\boldsymbol{\mathcal{A}}]_{i_1,:,:}$, $[\boldsymbol{\mathcal{A}}]_{:,i_2,:}$, $[\boldsymbol{\mathcal{A}}]_{:,:,i_3}$	i_1-th horizontal slice, i_2-th lateral slice, and i_3-th frontal slice, respectively
$\boldsymbol{\mathcal{A}}^{\dagger}$	pseudo-inverse of $\boldsymbol{\mathcal{A}}$	$\bar{\boldsymbol{A}}$, $\bar{\boldsymbol{A}}_i$	$\bar{\boldsymbol{A}} = \mathrm{bdiag}(\bar{\boldsymbol{\mathcal{A}}})$, $\bar{\boldsymbol{A}}_i = [\bar{\boldsymbol{\mathcal{A}}}]_{:,:,i}$
$\boldsymbol{\mathcal{A}}^{T_f}$	frontal slice-wise conjugate of $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{I_1 \times I_2 \times I_3}$, *i.e.*, $[\boldsymbol{\mathcal{A}}^{T_f}]_{:,:,i} = [\boldsymbol{\mathcal{A}}]^T_{:,:,i}$ for $i = 1, 2, \cdots, I_3$	$\boldsymbol{\mathcal{A}}^{T_{\boldsymbol{L}}}$, $\boldsymbol{\mathcal{A}}^T$	$\boldsymbol{\mathcal{A}}^{T_{\boldsymbol{L}}} = \boldsymbol{L}^{-1}(\boldsymbol{L}(\boldsymbol{\mathcal{A}})^{T_f})$ and $\boldsymbol{\mathcal{A}}^T = \mathrm{ifft}(\bar{\boldsymbol{\mathcal{A}}}^{T_f}, [], 3)$, respectively.
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \cdots$	functions	$\vec{\mathrm{A}}, \vec{\mathrm{B}}, \vec{\mathrm{C}}, \cdots$	ordered sequences

the application of tensor product-based rank in tensor recovery, involving TPCA (Tensor Principal Component Analysis), TRPCA (Tensor Robust Principal Component Analysis), and Tensor Completion (TC), because of its superior performance in studying the low-rankness of the tensor data. Subsequently, the optimization algorithms for solving the convex and non-convex approximation of these models and their variants have been introduced. Several typical applications in computer vision of these models (including color images and video denoising, image inpainting, and video background modeling) will be introduced in the final. I have summarized the symbols in this dissertation relating to matrices, tensors, and sets in Table 2.1. Some related notations and definitions are provided in the Appendix for detailed explanations, and I will utilize footnotes to indicate their location.

2.1 Tensor Rank

In matrix recovery, the rank function, *i.e.*, $\mathrm{rank}(\cdot)$, is commonly employed to assess the correlations between the columns and rows of the matrix, *i.e.*, the low-rankness of the matrix. As a result, $\mathrm{rank}(\cdot)$ has been utilized in early works [37, 8] to address the tensor case by transforming the tensor into 2D matrices. However, taking the color image with size of $512 \times 512 \times 3$ as an example, when we transform a tensor into the matrix

$$\begin{pmatrix} \boldsymbol{Y}^r \\ \boldsymbol{Y}^g \\ \boldsymbol{Y}^b \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_1^r & \boldsymbol{y}_2^r & \cdots & \boldsymbol{y}_{512}^r \\ \boldsymbol{y}_1^g & \boldsymbol{y}_2^g & \cdots & \boldsymbol{y}_{512}^g \\ \boldsymbol{y}_1^b & \boldsymbol{y}_2^b & \cdots & \boldsymbol{y}_{512}^b \end{pmatrix}, \tag{2.1}$$

the essential structure in $\begin{pmatrix} \boldsymbol{y}_k^r & \boldsymbol{y}_k^g & \boldsymbol{y}_k^b \end{pmatrix}$ for $k = 1, 2, \cdots, 512$ will be lost after vectorization, where $\boldsymbol{Y}^r = \begin{pmatrix} \boldsymbol{y}_1^r & \boldsymbol{y}_2^r & \cdots & \boldsymbol{y}_{512}^r \end{pmatrix}$, $\boldsymbol{Y}^g = \begin{pmatrix} \boldsymbol{y}_1^g & \boldsymbol{y}_2^g & \cdots & \boldsymbol{y}_{512}^g \end{pmatrix}$ and $\boldsymbol{Y}^b = \begin{pmatrix} \boldsymbol{y}_1^b & \boldsymbol{y}_2^b & \cdots & \boldsymbol{y}_{512}^b \end{pmatrix}$ are RGB channels of the image, respectively. Besides, some correlation information between different channels will be lost, such as the correlation between $\boldsymbol{y}_1^b$ and $\boldsymbol{y}_2^g$.

Therefore, three different types of tensor rank functions, including CP rank, Tucker rank, and tensor tubal rank, are proposed to explore the low rankness within the tensor data more accurately. The inspiration for these three tensor rank functions comes from three equivalent definitions of matrix rank, which are as follows:

D1: the minimum number of rank one decomposition of the given matrix;

D2: the number of orthonormal column (or row) vectors of the given matrix;

D3: the number of non-zero singular values of the given matrix.

In this section, we are going to introduce these three different definitions of tensor rank function in turn.

2.1.1 CP Rank

From the **D1**, a matrix $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2}$ with rank R can be decomposed into

$$\boldsymbol{X} = \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \boldsymbol{a}_r^{(2)T} = \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)}, \tag{2.2}$$

where $\boldsymbol{a}_r^{(j)} \in \mathbb{R}^{I_j}$ for $j = 1, 2$, and the symbol $\circ$ denotes the outer product defined in Definition A.1. Canonical Polyadic (CP) Decomposition can be obtained by extending (2.2) to the tensor case[31, 32], which is introduced in Definition 2.1.

Definition 2.1. *(Canonical Polyadic Decomposition) For $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_h}$, the Canonical Polyadic (CP) Decomposition of $\boldsymbol{\mathcal{A}}$ can be denoted as*

$$\boldsymbol{\mathcal{A}} = \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(h)} = \sum_{r=1}^{R} g_{r,r,\cdots,r} \boldsymbol{u}_r^{(1)} \circ \boldsymbol{u}_r^{(2)} \circ \cdots \circ \boldsymbol{u}_r^{(h)} = \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \cdots \times_h \boldsymbol{U}_h, \tag{2.3}$$

where $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R \times R \times \cdots R}$ is called as core tensor[1] *being diagonal tensor whose $(r, r, \cdots, r)$-th element is $g_{r,r,\cdots,r} = \|\boldsymbol{a}_r^{(1)}\|_2 \|\boldsymbol{a}_r^{(2)}\|_2 \cdots \|\boldsymbol{a}_r^{(h)}\|_2$, and $\boldsymbol{U}_k = [\frac{\boldsymbol{a}_1^{(k)}}{\|\boldsymbol{a}_1^{(k)}\|_2}, \frac{\boldsymbol{a}_2^{(k)}}{\|\boldsymbol{a}_2^{(k)}\|_2}, \cdots, \frac{\boldsymbol{a}_R^{(k)}}{\|\boldsymbol{a}_R^{(k)}\|_2}] \in \mathbb{R}^{I_k \times R}$ is a matrix with orthogonal columns for $k = 1, 2, \cdots, h$.*

The symbol $\times_n$ in (2.3) denotes the Mode-n product defined as follows.

Definition 2.2. *(Mode-n product)[48] Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_h}$ and $\boldsymbol{B} \in \mathbb{R}^{L \times I_n}$. Then the mode-n product of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{B}$ is defined as $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_n \boldsymbol{B} \in \mathbb{R}^{I_1 \times I_2 \times \cdots I_{n-1} \times L \times I_{n+1} \cdots \times I_h}$, where*

$$[\boldsymbol{\mathcal{A}} \times_n \boldsymbol{B}]_{i_1,\cdots,i_{n-1},l,i_{n+1},\cdots,i_h} = \sum_{i_n} [\boldsymbol{\mathcal{A}}]_{i_1,i_2,\cdots,i_h} [\boldsymbol{B}]_{l,i_n}. \tag{2.4}$$

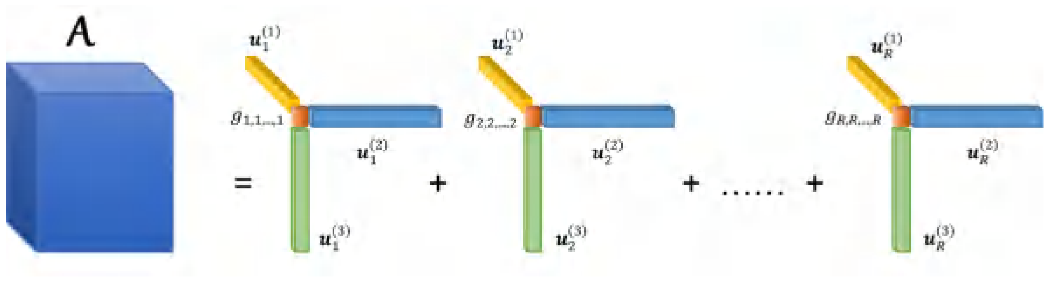


Figure 2.1: Illustration of CP Decomposition for a third-order tensor.

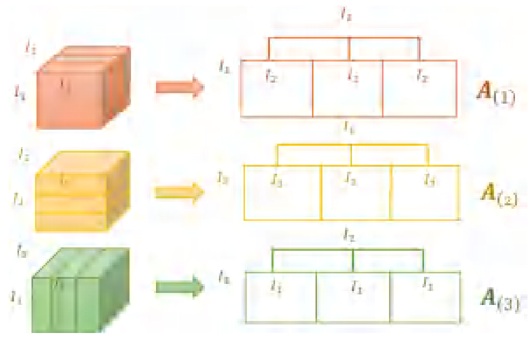


Figure 2.2: Illustration of the Kiers Method-Based Mode-n Unfolding for a third-order tensor.

From the definition of Mode-n product, we know that $\boldsymbol{C}_{(n)} = \boldsymbol{B}\boldsymbol{A}_{(n)}$ if $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_n \boldsymbol{B}$, where $\boldsymbol{A}_{(n)}$ is Kiers Method-Based Mode-n Unfolding[2] of $\boldsymbol{\mathcal{A}}$. To enhance understanding, I give illustrations of CP Decomposition and Kiers Method-Based Mode-n Unfolding in Figs. 2.1-2.2, respectively.

Based on CP decomposition, Kolda and Bader [43] have adopt the minimum number of tensor rank-one decomposition (CP decomposition) of the given tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_h}$ as the CP rank:

$$\operatorname{rank}_{\text{cp}}(\boldsymbol{\mathcal{X}}) = \min\{R | \boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(h)}, \boldsymbol{a}_r^{(j)} \in \mathbb{R}^{I_j} \text{ for } j = 1, 2, \cdots, h\}. \tag{2.5}$$

From (2.5), it is evident that the definition of CP rank is equivalent to the rank function when $h = 2$. However, computing the CP rank is generally NP-hard, which greatly restricts its application in tensor recovery. Therefore, a new way of defining the tensor rank based on Tucker Decomposition is proposed, and it has received more extensive attention compared to the CP rank.

2.1.2 Tucker Rank

Definition 2.3. *[72] (Tucker Decomposition) For $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_h}$, the Tucker Decomposition of $\boldsymbol{\mathcal{A}}$ can be denoted as*

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \cdots \times_h \boldsymbol{U}_h = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_h=1}^{R_h} g_{r_1, r_2, \cdots, r_n} \boldsymbol{u}_{r_1}^{(1)} \circ \boldsymbol{u}_{r_2}^{(2)} \circ \cdots \circ \boldsymbol{u}_{r_h}^{(h)}, \tag{2.6}$$

where the core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots R_h}$ is full tensor, and $\boldsymbol{U}_k = [\boldsymbol{u}_1^{(k)}, \boldsymbol{u}_2^{(k)}, \cdots, \boldsymbol{u}_{R_k}^{(k)}] \in \mathbb{R}^{I_k \times R_k}$ is a matrix with orthogonal columns for $k = 1, 2, \cdots, h$.

[1] Here, the diag elements of $\boldsymbol{\mathcal{G}}$ play the similar roles as singular values in the matrix.

[2] Please refer to Definition A.5 for the specific definition of Kiers Method-Based Mode-n Unfolding.

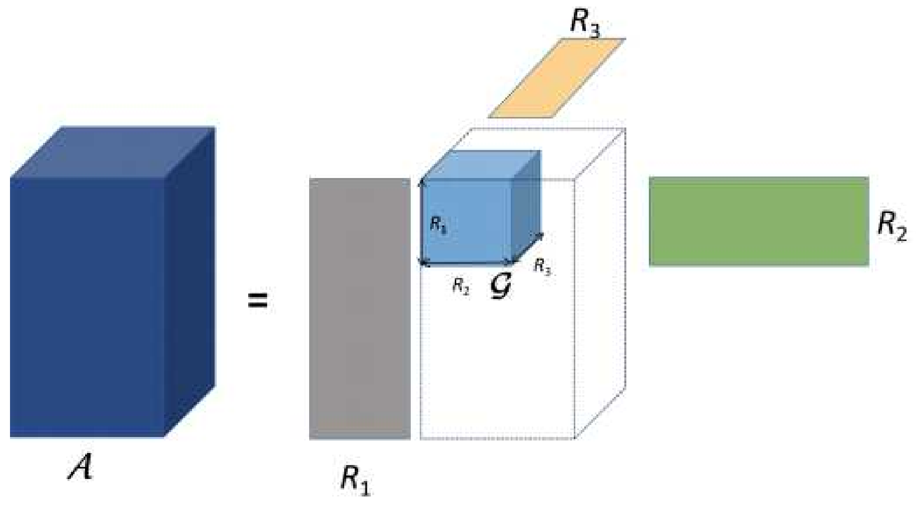


Figure 2.3: Illustration of Tucker Decomposition for a third-order tensor.

To better understand Tucker Decomposition, I give its illustration in Fig. 2.3.

According to the definition of Mode-n product, we can see that there has a matrix $\boldsymbol{H} \in \mathbb{R}^{I_1 \cdots I_{n-1} I_{n+1} \cdots I_h \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_h}$ such that $\boldsymbol{A}_{(n)} = \boldsymbol{U}_n \hat{\boldsymbol{A}}_{(n)} \boldsymbol{H}$, where $\hat{\boldsymbol{\mathcal{A}}} = \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \cdots \times_{n-1} \boldsymbol{U}_{n-1}$. Thus, we have $\operatorname{rank}(\boldsymbol{A}_{(n)}) \leq R_n (1 \leq n \leq h)$. The Tucker rank [3][43] of tensor $\boldsymbol{\mathcal{A}}$ is defined as

$$\operatorname{rank}_{\mathrm{tr}}(\boldsymbol{\mathcal{A}}) = (\operatorname{rank}(\boldsymbol{A}_{(1)}), \operatorname{rank}(\boldsymbol{A}_{(2)}), \cdots, \operatorname{rank}(\boldsymbol{A}_{(h)}).$$

Comparing definitions of CP Decomposition and Tucker Decomposition, it can be easily concluded that

$$\operatorname{rank}(\boldsymbol{A}_{(n)}) \leq \operatorname{rank}_{\mathrm{cp}}(\boldsymbol{\mathcal{A}}) (1 \leq n \leq h). \tag{2.7}$$

Therefore, if $\boldsymbol{\mathcal{A}}$ is a low CP rank tensor, $\boldsymbol{A}_{(n)}$ should be low rank for $n = 1, 2, \cdots, h$.

Based on the Tucker rank, Gandy *et al.* given a new tensor rank function that is defined as $\sum_{n=1}^{h} \operatorname{rank}(\boldsymbol{A}_{(n)})$ [25]. By considering different unfolding modes, this tensor rank function

[3] In some literature [48], it is called as the multilinear rank or n-rank.

can effectively characterize the correlation information across different tensor slices. It allows for the incorporation of inter-slice relationships, enhancing the ability of the rank function to capture the underlying structure of the tensor. Taking into account the variation in the low-rankness of $\boldsymbol{A}_{(n)}$ for different n, Liu *et al.* [45] give a weighted sum of the ranks of the unfolding matrices $\sum_{n=1}^{h} \alpha_n \mathrm{rank}(\boldsymbol{A}_{(n)})$, where α_n $(n = 1, 2, \cdots, h)$ are weight parameters satisfied $\sum_{n=1}^{h} \alpha_n = 1$.

It is worth noting that the weights play an important role in the weighted sum of ranks-based methods, and the best choice for the weights is unknown if without any prior. Thus, a new tensor rank based on the maximum rank of a set of unfolding matrics is proposed to promote the low-rankness of unfolding matrics of the recovered tensor [94].

2.1.3 Tensor Rank Based on Tensor-Tensor Product

Recently, the rank based on the tensor-tensor product (t-product) has received more and more attention because of its effectiveness in tensor recovery [33, 97]. The t-product of any two third-order tensors is defined as follows.

Definition 2.4. *(t-product) [42] Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_2 \times L \times I_3}$. Then the* t*-product $\boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}$ is defined to be a tensor of size $I_1 \times L \times I_3$,*

$$\boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}} = \mathrm{fold}(\mathrm{bcirc}(\boldsymbol{\mathcal{A}}) \cdot \mathrm{unfold}(\boldsymbol{\mathcal{B}})), \tag{2.8}$$

where

$$\mathrm{bcirc}(\boldsymbol{\mathcal{A}}) = \begin{pmatrix} [\boldsymbol{\mathcal{A}}]_{:,:,1} & [\boldsymbol{\mathcal{A}}]_{:,:,I_3} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,2} \\ [\boldsymbol{\mathcal{A}}]_{:,:,2} & [\boldsymbol{\mathcal{A}}]_{:,:,1} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,3} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{\mathcal{A}}]_{:,:,I_3} & [\boldsymbol{\mathcal{A}}]_{:,:,I_3-1} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,1} \end{pmatrix},$$

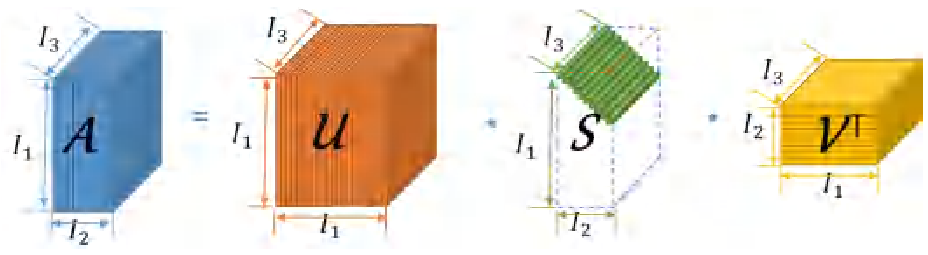


Figure 2.4: Illustration of t-SVD for a third-order tensor.

$$\text{unfold}(\boldsymbol{\mathcal{A}}) = \begin{pmatrix} [\boldsymbol{\mathcal{A}}]_{:,:,1} \\ [\boldsymbol{\mathcal{A}}]_{:,:,2} \\ \vdots \\ [\boldsymbol{\mathcal{A}}]_{:,:,I_h} \end{pmatrix} \in \mathbb{R}^{I_1 I_3 \times I_2},$$

and $\text{fold}(\cdot)$ *is its inverse operator* i.e., $\text{fold}(\text{unfold}(\boldsymbol{\mathcal{A}})) = \boldsymbol{\mathcal{A}}$.

Then, the tensor version of Singular Value Decomposition (t-SVD) can be given based on the t-product. From [50], we know that any tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ can be factorized as $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{S}} * \boldsymbol{\mathcal{V}}^T$ as illustrated in Fig. 2.4, where $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{I_1 \times I_1 \times I_3}$ and $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{I_2 \times I_2 \times I_3}$ are orthogonal[4], and $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is a f-diagonal tensor[5]. As $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}$ implies $\bar{\boldsymbol{\mathcal{C}}} = \bar{\boldsymbol{\mathcal{A}}} \odot_f \bar{\boldsymbol{\mathcal{B}}}$ from [50], we can calculate the t-SVD of $\boldsymbol{\mathcal{A}}$ by $\bar{\boldsymbol{\mathcal{A}}} = \bar{\boldsymbol{\mathcal{U}}} \odot_f \bar{\boldsymbol{\mathcal{S}}} \odot_f \bar{\boldsymbol{\mathcal{V}}}^{T_f}$, where $\odot_f$ stands for frontal slices product defined as follows.

Definition 2.5. *(Frontal slices product) For* $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{C}^{I_1 \times I_2 \times I_3}$, *the frontal slices product of* $\boldsymbol{\mathcal{A}}$ *and* $\boldsymbol{\mathcal{B}}$ *is defined as* $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \odot_f \boldsymbol{\mathcal{B}}$, *where* $[\boldsymbol{\mathcal{C}}]_{:,:,i_3} = [\boldsymbol{\mathcal{A}}]_{:,:,i_3} [\boldsymbol{\mathcal{B}}]_{:,:,i_3}$, $i_3 = 1, 2, \cdots, I_3$.

Algorithm 2.1 presents the details of computing t-SVD for a given tensor $\boldsymbol{\mathcal{Y}}$.

After obtaining $\boldsymbol{\mathcal{S}}$ by t-SVD, we can define the rank of $\boldsymbol{\mathcal{A}}$ as the number of non-zero

[4] Please refer to Definition A.18 for the specific definition of the orthogonal tensor.

[5] Please refer to Definition A.19 for the specific definition of the f-diagonal tensor.

Algorithm 2.1: t-SVD [50]

```
Input: $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\lambda > 0$.
Output: $\boldsymbol{\mathcal{U}}$, $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{V}}$.
1. Compute the result of DFT on $\boldsymbol{\mathcal{Y}}$ along the 3-rd dimension by using the Matlab
  command $\bar{\boldsymbol{\mathcal{Y}}} = \mathrm{fft}(\boldsymbol{\mathcal{Y}}, [], 3)$.
2. Compute each frontal slice of $\bar{\boldsymbol{\mathcal{U}}}$, $\bar{\boldsymbol{\mathcal{S}}}$ and $\bar{\boldsymbol{\mathcal{V}}}$ from $\bar{\boldsymbol{\mathcal{Y}}}$ by
for $i = 1, ..., \lfloor \frac{I_3+1}{2} \rfloor$ do
$[\bar{\boldsymbol{U}}_i, \bar{\boldsymbol{S}}_i, \bar{\boldsymbol{V}}_i] = \mathrm{SVD}(\bar{\boldsymbol{Y}}_i)$;
end for
for $i = \lfloor \frac{I_3+1}{2} \rfloor + 1, \cdots, I_3$ do
$\bar{\boldsymbol{U}}_i = \mathrm{Conj}(\bar{\boldsymbol{U}}_{(I_3-i+2)})$;
$\bar{\boldsymbol{S}}_i = \bar{\boldsymbol{S}}_{I_3-i+2}$;
$\bar{\boldsymbol{V}}_i = \mathrm{Conj}(\bar{\boldsymbol{V}}_{I_3-i+2})$;
end for
3. $\boldsymbol{\mathcal{U}} = \mathrm{ifft}(\bar{\boldsymbol{\mathcal{U}}}, [], 3)$, $\boldsymbol{\mathcal{S}} = \mathrm{ifft}(\bar{\boldsymbol{\mathcal{S}}}, [], 3)$ and $\boldsymbol{\mathcal{V}} = \mathrm{ifft}(\bar{\boldsymbol{\mathcal{V}}}, [], 3)$, where ifft is the
  inverse operation of fft.
```

singular tubes in $\boldsymbol{\mathcal{S}}$:

$$\mathrm{rank_t}(\boldsymbol{\mathcal{A}}) = |\{i|[\boldsymbol{\mathcal{S}}]_{i,i,:} \neq \mathbf{0}\}| = |\{i|[\bar{\boldsymbol{\mathcal{S}}}]_{i,i,:} \neq \mathbf{0}\}| = |\{i|[\boldsymbol{\mathcal{S}}]_{i,i,1} \neq 0\}|, \tag{2.9}$$

which is known as tensor tubal rank [50]. The first equality in (2.9) follows from $\bar{\boldsymbol{\mathcal{S}}} = \boldsymbol{\mathcal{S}} \times_3 \boldsymbol{F}_{I_3}$ and $\boldsymbol{\mathcal{S}} = \bar{\boldsymbol{\mathcal{S}}} \times_3 \boldsymbol{F}_{I_3}^{-1}$, where $\boldsymbol{F}_{I_3}$ is the $I_3 \times I_3$ discrete Fourier matrix. And the second equality holds from $[\boldsymbol{\mathcal{S}}]_{i,i,1} = \frac{1}{I_3} \sum_{j=1}^{I_3} [\bar{\boldsymbol{\mathcal{S}}}]_{i,i,j}$. Additionally, considering the number of non-zero singular elements in $\boldsymbol{\mathcal{S}}$, we can define the rank of a tensor as

$$\mathrm{rank_a}(\boldsymbol{\mathcal{A}}) = \frac{1}{I_3}|\{(i,i,i_3)|[\bar{\boldsymbol{\mathcal{S}}}]_{i,i,i_3} \neq 0\}| = \frac{1}{I_3}\mathrm{rank}(\bar{\boldsymbol{A}}) = \frac{1}{I_3}\mathrm{rank}(\mathrm{bcirc}(\boldsymbol{\mathcal{A}})), \tag{2.10}$$

which is known as the average tensor rank [50]. The last equality in (2.10) holds from the property of the DFT [50]. From $\mathrm{rank_t}(\boldsymbol{\mathcal{A}}) = |\{i|[\bar{\boldsymbol{\mathcal{S}}}]_{i,i,:} \neq \mathbf{0}\}|$ and $\mathrm{rank_a}(\boldsymbol{\mathcal{A}}) = \frac{1}{I_3}|\{(i,i,i_3)|[\bar{\boldsymbol{\mathcal{S}}}]_{i,i,i_3} \neq 0\}|$, we can easily conclude that

$$\mathrm{rank_a}(\boldsymbol{\mathcal{A}}) \leq \mathrm{rank_t}(\boldsymbol{\mathcal{A}}). \tag{2.11}$$

Besides, from $\mathrm{rank_a}(\boldsymbol{\mathcal{A}}) = \frac{1}{I_3}\mathrm{rank}(\mathrm{bcirc}(\boldsymbol{\mathcal{A}}))$ and (2.7), we have

$$\mathrm{rank_a}(\boldsymbol{\mathcal{A}}) \leq \max \mathrm{rank_{tr}}(\boldsymbol{\mathcal{A}}) \leq \mathrm{rank_{cp}}(\boldsymbol{\mathcal{A}}). \tag{2.12}$$

Therefore, the low average rank assumption is more easily satisfied in the real world [50].

Furthermore, from (2.9) and (2.10), we can observe that the tensor product-based rank allows us to analyze the low-rank properties of $[\bar{\boldsymbol{\mathcal{A}}}]_{:,:,i_3}$ that correspond to different frequency information from the 3rd direction of $\boldsymbol{\mathcal{A}}$ for different i_3, simultaneously. Specifically, $[\bar{\boldsymbol{\mathcal{A}}}]_{:,:,i_3}$ for small i_3 captures the low-frequency information within all $[\boldsymbol{\mathcal{A}}]_{i_1,i_2,:}$, representing the small changes in $[\boldsymbol{\mathcal{A}}]_{i_1,i_2,:}$, while $[\bar{\boldsymbol{\mathcal{A}}}]_{:,:,i_3}$ for larger i_3 captures the high-frequency information within all $[\boldsymbol{\mathcal{A}}]_{i_1,i_2,:}$, representing the fast-changing parts and noise in $[\boldsymbol{\mathcal{A}}]_{i_1,i_2,:}$. Therefore, the tensor product-based tensor rank can effectively distinguish the detailed information from the noise within the tensor.

Drawing inspiration from the expression $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}} = ((\boldsymbol{\mathcal{A}} \times_3 \boldsymbol{F}_{I_3}) \odot_f (\boldsymbol{\mathcal{B}} \times_3 \boldsymbol{F}_{I_3})) \times_3 \boldsymbol{F}_{I_3}^{-1}$, various definitions for tensor product and tensor rank can be established by replacing $\boldsymbol{F}_{I_3}$ with other invertible linear transforms. In this dissertation, the tensor average rank and tensor tubal rank are referred to as DFT-based methods due to their reliance on the Discrete Fourier Transform (DFT).

Definition 2.6. *(t-product induced by invertible linear transform) [52] Let* $\boldsymbol{L} : \mathbb{R}^{I_1 \times I_2 \times I_3} \longrightarrow$

$\mathbb{R}^{I_1 \times I_2 \times I_3}$ *be a invertible linear transform such that*

$$\boldsymbol{L}(\boldsymbol{\mathcal{A}}) = \boldsymbol{\mathcal{A}} \times_3 \boldsymbol{L}, \tag{2.13}$$

which satisfies

$$\boldsymbol{L}^T \boldsymbol{L} = \boldsymbol{L}\boldsymbol{L}^T = \ell_{\boldsymbol{L}}. \tag{2.14}$$

Here, $\ell_{\boldsymbol{L}} > 0$ *is a constant. Its inverse mapping is defined as*

$$\boldsymbol{L}^{-1}(\boldsymbol{\mathcal{A}}) = \boldsymbol{\mathcal{A}} \times_3 \boldsymbol{L}^{-1}. \tag{2.15}$$

For $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ *and* $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_2 \times L \times I_3}$*, the* t*-product based on the invertible linear transform* $\boldsymbol{L}$ *is defined as*

$$\boldsymbol{\mathcal{A}} *_{\boldsymbol{L}} \boldsymbol{\mathcal{B}} = \boldsymbol{L}^{-1}(\boldsymbol{L}(\boldsymbol{\mathcal{A}}) \odot_f \boldsymbol{L}(\boldsymbol{\mathcal{B}})). \tag{2.16}$$

The computation of the corresponding t-SVD is given as follows can be obtained by Algorithm 2.2. The tensor tubal rank and tensor average rank based on the invertible linear

Algorithm 2.2: t-SVD induced by the invertible linear transform L [52]

```
**Input:** $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\lambda > 0$.
**Output:** $\boldsymbol{\mathcal{U}}$, $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{V}}$.
1. Compute $\bar{\boldsymbol{\mathcal{Y}}}_{\boldsymbol{L}} = \boldsymbol{L}(\boldsymbol{\mathcal{Y}})$.
2. Compute each frontal slice of $\bar{\boldsymbol{\mathcal{U}}}_{\boldsymbol{L}}$, $\bar{\boldsymbol{\mathcal{S}}}_{\boldsymbol{L}}$ and $\bar{\boldsymbol{\mathcal{V}}}_{\boldsymbol{L}}$ from $\bar{\boldsymbol{\mathcal{Y}}}_{\boldsymbol{L}}$ by
**for** $i = 1, ..., I_3$ **do**
$[[\bar{\boldsymbol{\mathcal{U}}}_{\boldsymbol{L}}]_{:,:,i}, [\bar{\boldsymbol{\mathcal{S}}}_{\boldsymbol{L}}]_{:,:,i}, [\bar{\boldsymbol{\mathcal{V}}}_{\boldsymbol{L}}]_{:,:,i}] = \text{SVD}([\bar{\boldsymbol{\mathcal{Y}}}_{\boldsymbol{L}}]_{:,:,i})$;
**end for**
3. $\boldsymbol{\mathcal{U}} = \boldsymbol{L}^{-1}(\bar{\boldsymbol{\mathcal{U}}}_{\boldsymbol{L}})$, $\boldsymbol{\mathcal{S}} = \boldsymbol{L}^{-1}(\bar{\boldsymbol{\mathcal{S}}}_{\boldsymbol{L}})$ and $\boldsymbol{\mathcal{V}} = \boldsymbol{L}^{-1}(\bar{\boldsymbol{\mathcal{V}}}_{\boldsymbol{L}})$.
```

transform $\boldsymbol{L}$ are defined as

$$\operatorname{rank}_{\mathrm{t},\boldsymbol{L}}(\boldsymbol{\mathcal{A}}) = |\{i|[\bar{\boldsymbol{\mathcal{S}}}_{\boldsymbol{L}}]_{i,i,:} \neq \mathbf{0}\}|,$$

and

$$\operatorname{rank}_{\mathrm{a},\boldsymbol{L}}(\boldsymbol{\mathcal{A}}) = \frac{1}{\ell_{\boldsymbol{L}}}|\{(i,j,k)|[\bar{\boldsymbol{\mathcal{S}}}_{\boldsymbol{L}}]_{i,j,k} \neq 0\}| = \frac{1}{\ell_{\boldsymbol{L}}}\sum_{i_3}^{I_3} \operatorname{rank}([\bar{\mathcal{A}}_{\boldsymbol{L}}]_{:,:,i_3}),$$

respectively. The invertible linear transform $\boldsymbol{L}$ in (2.16) can be the Discrete Cosine Transform (DCT) and Random Orthogonal Matrix (ROM) [52].

2.2 Low-Rank Tensor Recovery Based on t-Product

The rise of low-rank models in recent years started roughly with introducing of the matrix completion (MC) problem [9, 10]. But in fact, principal component analysis (PCA) [76] was given and widely used in data dimensionality reduction long before MC was proposed. In this part, I will introduce three basic tensor recovery models based on the tensor average rank that extend from the matrix models, including PCA, Robust PCA, and MC, to recover a low-rank tensor from the observation tensor with various perturbations.

2.2.1 Tensor Principal Component Analysis

Assuming $\boldsymbol{\mathcal{Y}}$ is a tensor data with small noise perturbation, based on the low-rank prior in the tensor data $\mathcal{Y}$, we have $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{E}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, where $\boldsymbol{\mathcal{X}}$ is low tensor average rank, and $\mathcal{E}$ represents the noise and redundant information in the tensor data. The goal of Tensor Principal Component Analysis (TPCA) is to look for a low-rank approximation and approximately recover the tensor data from the noised observation $\boldsymbol{\mathcal{Y}}$, thus can be formulized as

$$\boldsymbol{\mathcal{X}}_{\mathrm{opt}} = \underset{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}}{\arg\min} \ \lambda \mathrm{rank}_{\mathrm{a}}(\boldsymbol{\mathcal{X}}) + \frac{1}{2} \|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2, \tag{2.17}$$

where $\boldsymbol{\mathcal{X}}$ is low rank tensor, and $\lambda > 0$ is parameter to balance the low-rankness and fidelity of $\boldsymbol{\mathcal{X}}$. Form [93], we have $\bar{\boldsymbol{X}}_{\mathrm{opt}} = [\bar{\boldsymbol{U}}_{\bar{\boldsymbol{Y}}}]_{:,1:k}[\bar{\boldsymbol{S}}_{\bar{\boldsymbol{Y}}}]_{1:k,1:k}([\bar{\boldsymbol{V}}_{\bar{\boldsymbol{Y}}}]_{:,1:k})^T$, where $\bar{\boldsymbol{U}}_{\bar{\boldsymbol{Y}}}$, $\bar{\boldsymbol{S}}_{\bar{\boldsymbol{Y}}}$, $\bar{\boldsymbol{V}}_{\bar{\boldsymbol{Y}}}$ can be obtained by SVD of $\bar{\boldsymbol{Y}}$ and k is the minimal integer such that $[\bar{\boldsymbol{S}}_{\bar{\boldsymbol{Y}}}]_{k,k} > \sqrt{2\lambda}$. Since the information of the data is concentrated in the components corresponding to a few largest singular values [64, 50], we can remove the noise and redundant information in the tensor data by solving (2.17).

2.2.2 Tensor Robust Principal Component Analysis

In TPCA, the Frobenius norm is imposed on the tensor $\mathcal{E}$ denoting the noise within the data to characterize the magnitude of small noise perturbation. By using the Frobenius norm, the principal components are robust to small noise perturbation, but sensitive to the outliers [50]. Based on the sparsity prior to the outliers, for given tensor data $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, Tensor Robust Principal Component Analysis (TRPCA) [50] is given as

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \mathrm{rank}_{\mathrm{a}}(\boldsymbol{\mathcal{L}}) + \lambda \|\boldsymbol{\mathcal{S}}\|_0 \quad s.t.\ \boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}}, \tag{2.18}$$

where $\boldsymbol{\mathcal{L}}$ is low-rank and $\boldsymbol{\mathcal{S}}$ is sparse. Compared to the Frobenius norm, the ℓ_0-norm can characterize the magnitude of the sparse tensor better.

TRPCA aims to exactly recover the low-rank tensor from tensor data with gross corruptions. When both gross sparse errors and small entry-wise noise appear in the tensor data $\boldsymbol{\mathcal{P}}$, $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}} + \boldsymbol{\mathcal{E}}$ holds, where $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{E}}$ ($\|\boldsymbol{\mathcal{E}}\|_F \leq \delta$) stand for the gross sparse errors and small entry-wise noise, respectively. The tensor version of Stable Principal Component

Pursuit (TSPCP) [103] can be employed to handle such cases:

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}\|_F^2 + \alpha \mathrm{rank}_{\mathrm{a}}(\boldsymbol{\mathcal{L}}) + \gamma\|\boldsymbol{\mathcal{S}}\|_0, \tag{2.19}$$

which can be converted to TPCA when $\gamma \longrightarrow \infty$.

2.2.3 Tensor Completion

From the above discussion, TRPCA and its variant (TSPCP) can effectively handle scenarios where gross sparse errors are present and the support set of noised elements is unknown. In situations where the support set of noised elements is known, the TRPCA problem can be transformed into a Tensor Completion (TC) problem [96, 51]. The goal of TC is to recover a low-rank tensor $\boldsymbol{\mathcal{X}}$ from tensor data with missing entries.

Suppose $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is an approximate low tensor average rank tensor, and $\mathbf{P}_\Omega$ is a linear project operator on the support set Ω composed of the locations corresponding to the observed entries in $\boldsymbol{\mathcal{M}}$, *i.e.*,

$$[\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}})]_{i_1,i_2,i_3} = \begin{cases} [\boldsymbol{\mathcal{M}}]_{i_1,i_2,i_3}, & \text{if} \quad (i_1,i_2,i_3) \in \Omega; \\ 0, & \text{if} \quad (i_1,i_2,i_3) \notin \Omega. \end{cases}$$

To recover a low-rank tensor $\boldsymbol{\mathcal{X}}$ that satisfy $\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) = \mathbf{P}_\Omega(\boldsymbol{\mathcal{X}})$, TC can be formulated as

$$\min_{\boldsymbol{\mathcal{X}}} \mathrm{rank}_{\mathrm{a}}(\boldsymbol{\mathcal{X}}) \qquad s.t.\ \mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) = \mathbf{P}_\Omega(\boldsymbol{\mathcal{X}}). \tag{2.20}$$

If $\boldsymbol{\mathcal{M}}$ is contaminated by the small entry-wise noise,we can express it as $\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) = \mathbf{P}_\Omega(\boldsymbol{\mathcal{X}}) + \mathbf{P}_\Omega(\boldsymbol{\mathcal{E}})$, where $\boldsymbol{\mathcal{E}}$ represents the noise tensor. To handle this scenario, Robust TC (RTC) is proposed, which is formulated as follows:

$$\min_{\boldsymbol{\mathcal{X}}} \lambda \mathrm{rank}_{\mathrm{a}}(\boldsymbol{\mathcal{X}}) + \frac{1}{2}\|\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) - \mathbf{P}_\Omega(\boldsymbol{\mathcal{X}})\|_F^2. \tag{2.21}$$

Since $\mathrm{rank_a}(\cdot)$ and ℓ_0 norm involved in TRPCA, TSPCP, and TC are discrete that lead to the NP-hard problems, we consider their convex and non-convex approximation in the next.

2.2.4 Convex and Non-Convex Approximation

2.2.4.1 Convex Approximation

The core idea of convex approximation adopted in low-rank recovery is using the convex envelopes of the tensor average rank and l_0-norm to replace the tensor average rank and l_0-norm in the tensor recovery model, respectively. In [50], Lu *et al.* have proved the tensor average norm defined as $\|\boldsymbol{\mathcal{A}}\|_{*,a} = \frac{1}{I_3}\|\bar{\boldsymbol{A}}\|_*$ is the convex envelope of the tensor average rank within the unit ball of the tensor spectral norm, where the tensor spectral norm of $\boldsymbol{\mathcal{A}}$ is defined as $\|\boldsymbol{\mathcal{A}}\|_2 = \|\bar{\boldsymbol{A}}\|_2$.

Therefore, the convex approximation of TRPCA and TSPCP can be written as

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \|\boldsymbol{\mathcal{L}}\|_* + \lambda\|\boldsymbol{\mathcal{S}}\|_1 \quad s.t.\ \boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}} \tag{2.22}$$

and

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}\|_F^2 + \alpha\|\boldsymbol{\mathcal{L}}\|_{*,a} + \gamma\|\boldsymbol{\mathcal{S}}\|_1, \tag{2.23}$$

respectively.

For TC, its convex version can be formulated as

$$\min_{\boldsymbol{\mathcal{X}}} \|\boldsymbol{\mathcal{X}}\|_* \qquad s.t.\ \mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) = \mathbf{P}_\Omega(\boldsymbol{\mathcal{X}}). \tag{2.24}$$

Since (2.22), (2.23) and (2.24) are convex, we can use an iterative algorithm to solve them, which will be introduced in the next section. Because of the orthogonal property of the invertible linear transforms, some essential properties in matrix recovery models are satisfied in (2.22) and (2.24) as well, such as their exact recovery guarantee [96, 50, 51].

Table 2.2: Examples of surrogate functions of ℓ_0, where $\gamma > 0$.

Name	$\mathcal{G}(x)$
ℓ_p [23]	$x^p, 0 < p < 1$
Geman [27]	$\frac{x}{x+\gamma}$
Laplace [71]	$(1 - \exp(-\frac{x}{\gamma}))$
LOG [57]	$\log(\gamma + x)$
Logarithm [24]	$\frac{1}{\log(\gamma+1)} \log(\gamma x + 1)$
ETP [26]	$\frac{1-\exp(-\gamma x)}{1-\exp(-\gamma)}$

2.2.4.2 Non-Convex Approximation

Although the exact recovery of convex approximation-based methods is guaranteed in theory [52], the conditions for the exact recovery are hardly met in the real world. Besides, the convex approximation-based methods treat the singular values with an equal penalty, leading to the over-penalization of large singular values. A number of non-convex-based tensor recovery methods have been proposed to solve this issue [39, 74, 38, 44, 84, 36, 68]. Kong *et al.* [44] have proposed a new tensor Schatten-p norm for getting a better approximation to the tensor nuclear norm, leading to a better tensor completion performance. The corresponding theoretical analysis has provided the performance guarantees for the resulting model. Jiang *et al.* proposed a non-convex approximation named partial sum of the tensor nuclear norm (PSTNN) that only penalizes the small singular values and leaves the large ones to preserve the low-rank structure of the tensor effectively [39]. Furthermore, Xu *et al.* [84] proposed a non-convex surrogate strategy for tensor multi-rank by using the Laplace function, in which the weight for each singular value is updated adaptively. The basic idea of these non-convex-based tensor completion methods is to replace the tensor average norm with its non-convex surrogate functions. Therefore, the non-convex approximation of TRPCA and RTC can be summarized in the following models:

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \|\boldsymbol{\mathcal{L}}\|_{*,\mathcal{G}} + \lambda\|\boldsymbol{\mathcal{S}}\|_{\mathcal{G}} \quad s.t.\ \boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}}, \tag{2.25}$$

and

$$\min_{\boldsymbol{\mathcal{X}}} \lambda\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} + \frac{1}{2}\|\mathbf{P}_{\Omega}(\boldsymbol{\mathcal{M}}) - \mathbf{P}_{\Omega}(\boldsymbol{\mathcal{X}})\|_F^2, \tag{2.26}$$

respectively. Here, $\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} = \frac{1}{I_3}\sum_{i=1}^{r}\mathcal{G}(\sigma_i(\bar{\boldsymbol{X}}))$, $\|\boldsymbol{\mathcal{S}}\|_{\mathcal{G}} = \sum_{i_1}\sum_{i_2}\sum_{i_3}\mathcal{G}([\boldsymbol{\mathcal{S}}]_{i_1,i_2,i_3})$, $\mathcal{G} : \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ is an increasing function listed in Table 2.2, and $r = \mathrm{rank}(\bar{\boldsymbol{X}})$.

As stated in [74], if select a proper γ in $\mathcal{G}(\cdot)$, we have $\sigma_i(\bar{\boldsymbol{X}}_j) \leq \mathcal{G}(\sigma_i(\bar{\boldsymbol{X}}_j)) \leq 1$ for $\sigma_i(\bar{\boldsymbol{X}}_j) \leq 1$, which implies

$$\|\boldsymbol{\mathcal{X}}\|_* = \|\boldsymbol{\mathcal{X}}\|_{*,a}{}^{6} \leq \|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} \leq \mathrm{rank_a}(\boldsymbol{\mathcal{X}}) \leq \mathrm{rank_t}(\boldsymbol{\mathcal{X}})$$

on the set $\{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times I_3} | \|\boldsymbol{\mathcal{X}}\| \leq 1\}$. Therefore, it can be concluded that $\|\cdot\|_{*,\mathcal{G}}$ is a better approximation of tensor average rank and tensor tubal rank than tensor average norm and tensor tubal norm. The related non-convex optimization algorithms were proposed to solve the resulting generalized non-convex approximation of TRPCA, TC and their variants [53, 74, 93].

2.3 Optimization Algorithm

In the following, I will introduce several iterative algorithms, which are commonly used in the optimization problem of tensor recovery.

[6] From [50], we can see that tensor nuclear norm is identity to tensor average norm.

2.3.1 Convex Algorithm

From the mathematical formulations of the above tensor recovery models, including TRPCA and TC, we encounter scenarios where multiple variables are involved. Since the optimization for the multiple variables simultaneously could be expensive in practice, we adopt iterative optimization methods to solve for the optimal variables. Taking (2.22) and (2.23) as the examples, we are going to introduce two basic iterative optimization algorithms, including the Block Coordinate Descent (BCD) Method and the Alternating Direction Method of Multipliers (ADMM) and their application in convex optimization problem of tensor recovery. And there are two convex sub-problems involved in:

$$\underset{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{I_1\times I_2\times I_3}}{\arg\min}\ \gamma\|\boldsymbol{\mathcal{X}}\|_1+\frac{1}{2}\|\boldsymbol{\mathcal{Y}}-\boldsymbol{\mathcal{X}}\|_F^2, \tag{2.27}$$

and

$$\underset{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{I_1\times I_2\times I_3}}{\arg\min}\ \alpha\|\boldsymbol{\mathcal{X}}\|_*+\frac{1}{2}\|\boldsymbol{\mathcal{Y}}-\boldsymbol{\mathcal{X}}\|_F^2. \tag{2.28}$$

The optimal solution of (2.27) can be obtained by soft thresholding shrinkage operator $(\boldsymbol{\mathcal{Y}}-\gamma)_+$, each element of which is defined as $\max([\boldsymbol{\mathcal{Y}}]_{i_1,i_2,i_3}-\gamma,0)$. And the optimal solution of (2.28) can be obtained by the following conclusion.

Theorem 2.1. *[50] For any $\lambda>0$ and $\boldsymbol{\mathcal{Y}}\in\mathbb{R}^{I_1\times I_2\times I_3}$, then the tensor singular value thresholding operator obeys*

$$\mathcal{D}(\boldsymbol{\mathcal{Y}},\lambda)\in\underset{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{I_1\times I_2\times I_3}}{\arg\min}\ \lambda\|\boldsymbol{\mathcal{X}}\|_*+\frac{1}{2}\|\boldsymbol{\mathcal{Y}}-\boldsymbol{\mathcal{X}}\|_F^2, \tag{2.29}$$

where $\mathcal{D}(\boldsymbol{\mathcal{Y}},\lambda)$ is obtained by TSVT (Algorithm 2.3).

Algorithm 2.3: Tensor Singular Value Thresholding (TSVT) [50]

```
**Input:** $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\lambda > 0$.
**Output:** $\mathcal{D}(\boldsymbol{\mathcal{Y}}, \lambda)$.
1. Compute $\bar{\boldsymbol{\mathcal{Y}}}$ by performing DFT on $\boldsymbol{\mathcal{Y}}$ along the 3-rd dimension.
2. Perform matrix SVT on each frontal slice of $\bar{\boldsymbol{\mathcal{Y}}}$ by
**for** $i = 1, ..., \lfloor \frac{I_3+1}{2} \rfloor$ **do**
$[\bar{\boldsymbol{U}}_i, \bar{\boldsymbol{S}}_i, \bar{\boldsymbol{V}}_i] = \text{SVD}(\bar{\boldsymbol{\mathcal{Y}}}_i)$;
$\bar{\boldsymbol{W}}_i = \bar{\boldsymbol{U}}_i (\bar{\boldsymbol{S}}_i - \lambda)_+ \bar{\boldsymbol{V}}_i^T$;
**end for**
**for** $i = \lfloor \frac{I_3+1}{2} \rfloor + 1, \cdots, I_3$ **do**
$\bar{\boldsymbol{W}}_i = \text{Conj}(\bar{\boldsymbol{W}}_{I_3 - i + 2})$;
**end for**
3. Compute $\mathcal{D}(\boldsymbol{\mathcal{Y}}, \lambda)$ by performing inverse DFT on $\bar{\boldsymbol{\mathcal{W}}}$ along the 3-rd dimension.
```

2.3.1.1 Block Coordinate Descent Method

Let us consider the unconstrained problem

$$\min_{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n} \mathcal{F}(\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n). \tag{2.30}$$

According to the block coordinate descent method (BCD) framework, we can solve (2.30) iteratively as follows.

Given $(\boldsymbol{X}_1^{(t)}, \boldsymbol{X}_2^{(t)}, \cdots, \boldsymbol{X}_n^{(t)})$, we can get $\boldsymbol{X}_k^{(t+1)} (k = 1, 2, \cdots, n)$ by

$$\boldsymbol{X}_k^{(t+1)} = \arg\min_{\boldsymbol{X}_k} \mathcal{F}(\boldsymbol{X}_1^{(t+1)}, \cdots, \boldsymbol{X}_{k-1}^{(t+1)}, \boldsymbol{X}_k, \boldsymbol{X}_{k+1}^{(t)}, \cdots, \boldsymbol{X}_n^{(t)}), \tag{2.31}$$

producing the next iterate $(\boldsymbol{X}_1^{(t+1)}, \boldsymbol{X}_2^{(t+1)}, \cdots, \boldsymbol{X}_n^{(t+1)})$.

Here, we take the problem (2.32) (Tensor Stable Principal Component Pursuit) as an example to explain BCD.

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}\|_F^2 + \alpha\|\boldsymbol{\mathcal{L}}\|_* + \gamma\|\boldsymbol{\mathcal{S}}\|_1. \tag{2.32}$$

From the framework of BCD, the problem (2.32) can be iteratively solved as follows.

Step1 Given $\boldsymbol{\mathcal{L}}^{(t)}$, we update $\boldsymbol{\mathcal{S}}$ by

$$\begin{aligned}\boldsymbol{\mathcal{S}}^{(t+1)} &= \arg\min_{\boldsymbol{\mathcal{S}}} \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}}^{(t)} - \boldsymbol{\mathcal{S}}\|_F^2 + \alpha\|\boldsymbol{\mathcal{L}}^{(t)}\|_* + \gamma\|\boldsymbol{\mathcal{S}}\|_1 \\ &= \arg\min_{\boldsymbol{\mathcal{S}}} \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}}^{(t)} - \boldsymbol{\mathcal{S}}\|_F^2 + \gamma\|\boldsymbol{\mathcal{S}}\|_1 \\ &= (\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}}^{(t)} - \gamma)_+. \end{aligned} \tag{2.33}$$

Step2 Given $\boldsymbol{\mathcal{S}}^{(t+1)}$, we update $\boldsymbol{\mathcal{L}}$ by

$$\begin{aligned}\boldsymbol{\mathcal{L}}^{(t+1)} &= \arg\min_{\boldsymbol{\mathcal{L}}} \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}^{(t+1)}\|_F^2 + \alpha\|\boldsymbol{\mathcal{L}}\|_* + \gamma\|\boldsymbol{\mathcal{S}}^{(t+1)}\|_1 \\ &= \arg\min_{\boldsymbol{\mathcal{L}}} \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}^{(t+1)}\|_F^2 + \alpha\|\boldsymbol{\mathcal{L}}\|_* \\ &= \mathcal{D}(\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{S}}^{(t+1)}, \alpha). \end{aligned} \tag{2.34}$$

Therefore, we can get the optimal solution of (2.32), *i.e.*, $(\hat{\boldsymbol{\mathcal{L}}}, \hat{\boldsymbol{\mathcal{S}}})$, by repeating the above steps until the algorithm convergence.

2.3.1.2 Alternating Direction Method of Multipliers

Let us consider the following constrained problem:

$$\min_{\boldsymbol{A},\boldsymbol{B}} \mathcal{F}_1(\boldsymbol{A}) + \mathcal{F}_2(\boldsymbol{B}),\ s.t.\ \mathcal{G}_1(\boldsymbol{A}) + \mathcal{G}_2(\boldsymbol{B}) = \boldsymbol{C}. \tag{2.35}$$

The augmented Lagrangian function of (2.35) can be written as

$$\mathcal{L}_\mu(\boldsymbol{A},\boldsymbol{B},\boldsymbol{Y}) = \mathcal{F}_1(\boldsymbol{A}) + \mathcal{F}_2(\boldsymbol{B}) + \langle \mathcal{G}_1(\boldsymbol{A}) + \mathcal{G}_2(\boldsymbol{B}) - \boldsymbol{C}, \boldsymbol{Y}\rangle + \frac{\mu}{2}\|\mathcal{G}_1(\boldsymbol{A}) + \mathcal{G}_2(\boldsymbol{B}) - \boldsymbol{C}\|_F^2, \tag{2.36}$$

where $\mu > 0$ is parameter, and $\boldsymbol{Y}$ is the Lagrange multiplier. According to the framework of the alternating direction method of multipliers (ADMM), we can solve (2.35) iteratively:

Given $(\boldsymbol{A}^{(t)}, \boldsymbol{B}^{(t)}, \boldsymbol{Y}^{(t)})$, we can get $\boldsymbol{A}^{(t+1)}$ and $\boldsymbol{B}^{(t+1)}$ by

$$\boldsymbol{A}^{(t+1)} = \arg\min_{\boldsymbol{A}} \mathcal{L}_\mu(\boldsymbol{A}, \boldsymbol{B}^{(t)}, \boldsymbol{Y}^{(t)}) \tag{2.37}$$

and

$$\boldsymbol{B}^{(t+1)} = \arg\min_{\boldsymbol{B}} \mathcal{L}_\mu(\boldsymbol{A}^{(t+1)}, \boldsymbol{B}, \boldsymbol{Y}^{(t)}), \tag{2.38}$$

respectively.

Given $(\boldsymbol{A}^{(t+1)}, \boldsymbol{B}^{(t+1)}, \boldsymbol{Y}^{(t)})$, we update $\boldsymbol{Y}$ by

$$\boldsymbol{Y}^{(t+1)} = \boldsymbol{Y}^{(t)} + \mu(\mathcal{G}_1(\boldsymbol{A}^{(t+1)}) + \mathcal{G}_2(\boldsymbol{B}^{(t+1)}) - \boldsymbol{C}^{(t+1)}) \tag{2.39}$$

From (2.22) and (2.24), we can see that the convex approximation of TRPCA and TC have the form (2.35), and thus can be solved by ADMM further. Here, I take (2.22) as an example to explain the framework of ADMM. The augmented Lagrangian function of (2.22) can be written as

$$\mathcal{L}_\mu(\boldsymbol{\mathcal{L}}, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{Y}}) = \|\boldsymbol{\mathcal{L}}\|_* + \lambda\|\boldsymbol{\mathcal{S}}\|_1 + \langle \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Y}} \rangle + \frac{\mu}{2}\|\boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{P}}\|_F^2, \tag{2.40}$$

where $\mu > 0$ is parameter, and $\boldsymbol{\mathcal{Y}}$ is the Lagrange multiplier. Then, we can solve (2.22) iteratively:

Step 1 Given $(\boldsymbol{\mathcal{S}}^{(t)}, \boldsymbol{\mathcal{Y}}^{(t)})$, we can get $\boldsymbol{\mathcal{L}}^{(t+1)}$ by

$$\begin{aligned}
\boldsymbol{\mathcal{L}}^{(t+1)} &= \arg\min_{\boldsymbol{\mathcal{L}}} \mathcal{L}_\mu(\boldsymbol{\mathcal{L}}, \boldsymbol{\mathcal{S}}^{(t)}, \boldsymbol{\mathcal{Y}}^{(t)}) \\
&= \arg\min_{\boldsymbol{\mathcal{L}}} \|\boldsymbol{\mathcal{L}}\|_* + \langle \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Y}} \rangle + \frac{\mu}{2}\|\boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{P}}\|_F^2 \\
&= \arg\min_{\boldsymbol{\mathcal{L}}} \frac{1}{\mu}\|\boldsymbol{\mathcal{L}}\|_* + \frac{1}{2}\|\boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}}^{(t)} - \boldsymbol{\mathcal{P}} + \frac{1}{\mu}\boldsymbol{\mathcal{Y}}^{(t)}\|_F^2 \\
&= \mathcal{D}(-\boldsymbol{\mathcal{S}}^{(t)} + \boldsymbol{\mathcal{P}} - \frac{1}{\mu}\boldsymbol{\mathcal{Y}}^{(t)}, \frac{1}{\mu})
\end{aligned} \tag{2.41}$$

Step 2 Given $(\boldsymbol{\mathcal{L}}^{(t+1)}, \boldsymbol{\mathcal{Y}}^{(t)})$, we get $\boldsymbol{\mathcal{S}}^{(t+1)}$ by

$$\begin{aligned}
\boldsymbol{\mathcal{S}}^{(t+1)} &= \arg\min_{\boldsymbol{\mathcal{S}}} \mathcal{L}_\mu(\boldsymbol{\mathcal{L}}^{(t+1)}, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{Y}}^{(t)}) \\
&= \arg\min_{\boldsymbol{\mathcal{S}}} \lambda\|\boldsymbol{\mathcal{S}}\|_1 + \langle \boldsymbol{\mathcal{L}}^{(t+1)} + \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Y}}^{(t)} \rangle + \frac{\mu}{2}\|\boldsymbol{\mathcal{L}}^{(t+1)} + \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{P}}\|_F^2 \\
&= \arg\min_{\boldsymbol{\mathcal{S}}} \frac{\lambda}{\mu}\|\boldsymbol{\mathcal{S}}\|_1 + \frac{1}{2}\|\boldsymbol{\mathcal{L}}^{(t+1)} + \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{P}} + \frac{1}{\mu}\boldsymbol{\mathcal{Y}}^{(t)}\|_F^2 \\
&= (-\boldsymbol{\mathcal{L}}^{(t+1)} + \boldsymbol{\mathcal{P}} - \frac{1}{\mu}\boldsymbol{\mathcal{Y}}^{(t)} - \frac{\lambda}{\mu})_+. \qquad (2.42)
\end{aligned}$$

Step 3 Given $(\boldsymbol{\mathcal{L}}^{(t+1)}, \boldsymbol{\mathcal{S}}^{(t+1)}, \boldsymbol{\mathcal{Y}}^{(t)})$, we update $\boldsymbol{\mathcal{Y}}$ by

$$\boldsymbol{\mathcal{Y}}^{(t+1)} = \boldsymbol{\mathcal{Y}}^{(t)} + \mu(\boldsymbol{\mathcal{L}}^{(t+1)} + \boldsymbol{\mathcal{S}}^{(t+1)} - \boldsymbol{\mathcal{P}}). \qquad (2.43)$$

Step 4 For given $\rho > 1$, we update $\mu^{(t+1)}$ by

$$\mu = \min(\rho\mu, \bar{\mu}), \qquad (2.44)$$

where $\bar{\mu}$ is the upper bound of μ.

Therefore, we can get the optimal solution of (2.22), *i.e.*, $(\hat{\boldsymbol{\mathcal{L}}}, \hat{\boldsymbol{\mathcal{S}}})$ by repeating the above steps until the algorithm convergence, where t is for iteration number of the algorithm. The stop criterion in the algorithm is set as

$$\frac{\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}}^{(t+1)} - \boldsymbol{\mathcal{S}}^{(t+1)}\|_F}{\text{dual\_norm}},$$

where $\text{dual\_norm} = \max(\|\boldsymbol{\mathcal{P}}\|_2, \|\boldsymbol{\mathcal{P}}\|_\infty)$.

2.3.2 Non-Convex Algorithm

In this subsection, I am going to introduce several optimization algorithms including Iterative Reweighted t-TNN Algorithm and Generalized Tensor Singular Values Thresholding-based iterative algorithms for solving the non-convex approximation of the tensor recovery models.

2.3.2.1 Iterative Reweighted t-TNN Algorithm

By extending Iteratively Reweighted Nuclear Norm (IRNN) algorithm [53] to the tensor case, Wang *et al.* have proposed Iterative Reweighted t-TNN Algorithm (IR-t-TNN) [74] to solve the following general tensor recovery problem:

$$\min_{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{I_1\times I_2\times I_3}} \lambda\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} + \mathcal{L}(\boldsymbol{\mathcal{X}}), \tag{2.45}$$

where $\mathcal{G}(\cdot)$ satisfies Assumption 2.1, and $\mathcal{L}(\cdot)$ is a loss function such that $\|\nabla\mathcal{L}(\boldsymbol{\mathcal{X}}) - \nabla\mathcal{L}(\boldsymbol{\mathcal{Y}})\|_F \leq l(\mathcal{L})\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\|_F$ for some constant $l(\mathcal{L}) > 0$ (In other words, $\mathcal{L}(\cdot)$ is Lipschitz gradient continuous.).

Assumption 2.1. $\mathcal{G}(\cdot) : [0, +\infty) \to [0, +\infty)$ *satisfies*

A1 $\mathcal{G}(x)$ *is a continuous, monotonically non-decreasing, and concave function;*

A2 $\mathcal{G}(0) = 0, \lim_{x\longrightarrow+\infty} \dfrac{\mathcal{G}(x)}{x} = 0$,

Since $\mathcal{G}(\cdot)$ is concave on $[0, +\infty)$, we have

$$\mathcal{G}(\sigma_i(\bar{\boldsymbol{X}}_j)) \leq \mathcal{G}(\sigma_i(\bar{\boldsymbol{X}}_j^{(t)})) + w_{i,i,j}^{(t)}(\sigma_i(\bar{\boldsymbol{X}}_j) - \sigma_i(\bar{\boldsymbol{X}}_j^{(t)})),$$

where $w_{i,i,j}^{(t)}$ is the supergradient of $\mathcal{G}(x)$ at $x = \sigma_i(\bar{\boldsymbol{X}}_j^{(t)})$. Therefore, we have

$$\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} \leq \|\boldsymbol{\mathcal{X}}^{(t)}\|_{*,\mathcal{G}} + \sum_{k=1}^{r_j}\sum_{j=1}^{I_3} w_{i,i,j}^{(t)}(\sigma_i(\bar{\boldsymbol{X}}_j) - \sigma_i(\bar{\boldsymbol{X}}_j^{(t)})).$$

On the other hand, we have

$$\mathcal{L}(\boldsymbol{\mathcal{X}}) \leq \mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}) + \langle\nabla\mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}), \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\rangle + \frac{\mu}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\|_F^2,$$

where $\mu \geq l(\mathcal{L})$. Therefore, Wang *et al.* turn to solve the following relaxed problem:

$$\begin{aligned}
\boldsymbol{\mathcal{X}}^{(t+1)} &= \arg\min_{\boldsymbol{\mathcal{X}}} \lambda(\|\boldsymbol{\mathcal{X}}^{(t)}\|_{*,\mathcal{G}} + \sum_{k=1}^{r_j}\sum_{j=1}^{I_3} w_{i,i,j}(\sigma_i(\bar{\boldsymbol{X}}_j) - \sigma_i(\bar{\boldsymbol{X}}_j^{(t)}))) + \mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}) \\
&\qquad + \langle \nabla\mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}), \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\rangle + \frac{\mu}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\|_F^2 \\
&= \arg\min_{\boldsymbol{\mathcal{X}}} \lambda \sum_{k=1}^{r_j}\sum_{j=1}^{I_3} w_{i,i,j}^{(t)}\sigma_i(\bar{\boldsymbol{X}}_j) + \langle \nabla\mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}), \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\rangle + \frac{\mu}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\|_F^2 \\
&= \arg\min_{\boldsymbol{\mathcal{X}}} \lambda \sum_{k=1}^{r_j}\sum_{j=1}^{I_3} w_{i,i,j}^{(t)}\sigma_i(\bar{\boldsymbol{X}}_j) + \frac{\mu}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)} + \frac{1}{\mu}\nabla\mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)})\|_F^2. \qquad (2.46)
\end{aligned}$$

Since $w_{i,i,j}^{(t)}$ is the supergradient of $\mathcal{G}(x)$ at $x = \sigma_i(\bar{\boldsymbol{X}}_j^{(t)})$, $0 \leq w_{1,1,j}^{(t)} \leq \cdots \leq w_{k,k,j}^{(t)} \leq \cdots \leq w_{\min(I_1,I_2),\min(I_1,I_2),j}^{(t)}$ $(j = 1, 2, \cdots, I_3)$ from the antimonotone property of supergradient [53]. Thus, we have

$$\boldsymbol{\mathcal{X}}^{(t+1)} = \mathcal{D}_{\boldsymbol{\mathcal{W}}^{(t)}}(\boldsymbol{\mathcal{X}}^{(t)} - \frac{1}{\mu}\nabla\mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}), \lambda), \qquad (2.47)$$

where

$$\mathcal{D}_{\boldsymbol{\mathcal{W}}}(\boldsymbol{\mathcal{Y}}, \lambda) = \arg\min_{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}} \lambda\|\boldsymbol{\mathcal{X}}\|_{*,\boldsymbol{\mathcal{W}}} + \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2. \qquad (2.48)$$

And Weighted Tensor Singular Value Thresholding (WTSVT) has been proposed to solve (2.48): $\mathcal{D}_{\boldsymbol{\mathcal{W}}}(\boldsymbol{\mathcal{Y}}, \lambda) = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{S}}_{\boldsymbol{\mathcal{W}},\lambda} * \boldsymbol{\mathcal{V}}^T$, where $\boldsymbol{\mathcal{S}}_{\boldsymbol{\mathcal{W}},\lambda} = \text{ifft}((\bar{\boldsymbol{\mathcal{S}}} - \lambda\boldsymbol{\mathcal{W}})_+, [], 3)$ [74, 53], and $\boldsymbol{\mathcal{U}}$, $\bar{\boldsymbol{\mathcal{S}}}$, and $\boldsymbol{\mathcal{V}}$ can be obtained by the t-SVD of $\boldsymbol{\mathcal{Y}}$: $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{S}} * \boldsymbol{\mathcal{V}}^T$.

Taking $\mathcal{L}(\boldsymbol{\mathcal{X}})$ as $\frac{1}{2}\|\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) - \mathbf{P}_\Omega(\boldsymbol{\mathcal{X}})\|_F^2$, we can solve the non-convex approximation of RTC by IR-t-TNN (as well as its convex approximation).

2.3.2.2 Generalized Tensor Singular Values Threshold (GTSVT)-Based Iterative Algorithms

In the IRNN algorithm, the both $\|\boldsymbol{\mathcal{X}}^{(t)}\|_{*,\mathcal{G}} + \sum_{k=1}^{r_j}\sum_{j=1}^{I_3} w_{i,i,j}^{(t)}(\sigma_i(\bar{\boldsymbol{X}}_j) - \sigma_i(\bar{\boldsymbol{X}}_j^{(t)}))$ and $\mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}) + \langle \nabla\mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}), \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\rangle + \frac{\mu}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\|_F^2$ are used to replace $\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}}$ and $\mathcal{L}(\boldsymbol{\mathcal{X}})$, respectively, for easy solving of (2.45).

Inspired by GSVT algorithm [54], we can utilize a tighter estimation for the objective function in (2.45) than the objective function in (2.46), and solve (2.45) iteratively as follow:

$$\begin{aligned}\boldsymbol{\mathcal{X}}^{(t+1)} =& \arg\min_{\boldsymbol{\mathcal{X}}} \lambda\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} + \mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}) + \langle \nabla\mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}), \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\rangle + \frac{\mu}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\|_F^2 \\ =& \arg\min_{\boldsymbol{\mathcal{X}}} \lambda\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} + \langle \nabla\mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)}), \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\rangle + \frac{\mu}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)}\|_F^2 \\ =& \arg\min_{\boldsymbol{\mathcal{X}}} \lambda\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} + \frac{\mu}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^{(t)} + \frac{1}{\mu}\nabla\mathcal{L}(\boldsymbol{\mathcal{X}}^{(t)})\|_F^2. \end{aligned} \tag{2.49}$$

Therefore, Zhang *et al.* [93] provides an algorithm (Algorithm 2.4) that solves the following problem directly:

$$\arg\min_{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{I_1\times I_2\times I_3}} \lambda\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} + \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2, \tag{2.50}$$

which can be used for solving (2.45), and the non-convex approximation of other tensor recovery models, including TC, TRPCA, and TSPCP (as well as their convex approximation). The optimal solution of (2.50) has been analysed in the Theorem 2.2.

Theorem 2.2. *[93] For any $\lambda > 0$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1\times I_2\times I_3}$, if $\mathcal{G}$ is increasing on $[0, +\infty)$, then the tensor singular value thresholding operator obeys*

$$\mathcal{D}_{\mathcal{G}}(\boldsymbol{\mathcal{Y}}, \lambda) \in \arg\min_{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{I_1\times I_2\times I_3}} \lambda\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} + \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2, \tag{2.51}$$

where $\mathcal{D}_{\mathcal{G}}(\boldsymbol{\mathcal{Y}}, \lambda)$ is obtained by GTSVT (Algorithm 2.4), and $\mathcal{T}_{\mathcal{G}}(\bar{\boldsymbol{S}}_i, \lambda)$ in Algorithm 2.4 is defined as

$$\mathcal{T}_{\mathcal{G}}(\bar{\boldsymbol{S}}_i, \lambda) = \arg\min_{\boldsymbol{S}\in\mathbb{R}^{I_1\times I_2}} \frac{1}{2}\|\bar{\boldsymbol{S}}_i - \boldsymbol{S}\|_F^2 + \lambda\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\mathcal{G}(|s_{i_1 i_2}|). \tag{2.52}$$

The problem (2.52) is a basic optimization problem in low-rank recovery and is widely studied. For example, in [54], Lu *et al.* propose Algorithm 2.5 that that solves (2.52) to

Algorithm 2.4: Generalized Tensor Singular Value Thresholding (GTSVT) [93]

```
Input: $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\lambda > 0$.
Output: $\mathcal{D}_{\mathcal{G}}(\boldsymbol{\mathcal{Y}}, \lambda)$.
1. Compute $\bar{\boldsymbol{\mathcal{Y}}}$ by performing DFT on $\boldsymbol{\mathcal{Y}}$ along the 3-rd dimension.
2. Perform matrix SVT on each frontal slice of $\bar{\boldsymbol{\mathcal{Y}}}$ by
for $i = 1, ..., \lfloor \frac{I_3+1}{2} \rfloor$ do
    $[\bar{\boldsymbol{U}}_i, \bar{\boldsymbol{S}}_i, \bar{\boldsymbol{V}}_i] = \text{SVD}([\bar{\boldsymbol{\mathcal{Y}}}]_{:,:,i})$;
    $\bar{\boldsymbol{W}}_i = \bar{\boldsymbol{U}}_i \mathcal{T}_{\mathcal{G}}(\bar{\boldsymbol{S}}_i, \lambda) \bar{\boldsymbol{V}}_i^T$;
end for
for $i = \lfloor \frac{I_3+1}{2} \rfloor + 1, \cdots, I_3$ do
    $\bar{\boldsymbol{W}}_i = \text{Conj}(\bar{\boldsymbol{W}}_{I_3-i+2})$;
end for
3. Compute $\mathcal{D}_{\mathcal{G}}(\boldsymbol{\mathcal{Y}}, \lambda)$ by performing inverse DFT on $\bar{\boldsymbol{\mathcal{W}}}$ along the 3-rd dimension.
```

global optimality. Furthermore, Zhang *et al.* [90] develop a fixed point algorithm (Algorithm 2.6) that solves (2.52) to global optimality with a guaranteed super-linear convergence rate, when the surrogate function $\mathcal{G}(\cdot)$ satisfies Assumption 2.2.

Assumption 2.2. $\mathcal{G}(\cdot) : [0, +\infty) \to [0, +\infty)$ *satisfies* $\mathbf{A1} - \mathbf{A3}$*:*

A1 $\mathcal{G}(x)$ *is strictly concave and increasing, and* $\mathcal{G}(0) = 0$*;*

A2 $\mathcal{G}'(x)$ *is strictly convex;*

A3 $\mathcal{G}''(x)$ *is continuous on* $(0, +\infty)$.

Besides, from the introduction of the iterative algorithms including BCD and ADMM, we can see that the basic idea of these iterative algorithms is to convert a complex optimization

Algorithm 2.5: Lu's work [54]

```
Input: A real number $y > 0$, a number of iterations $\kappa > 0$ and a tolerance $\tau > 0$.
Output: $x_{\mathrm{L}}^{*} = \arg\min_x \mathcal{F}_y(x) = \frac{1}{2}(y-x)^2 + \lambda\mathcal{G}(|x|)$.
// Find $\hat{x}_{\mathrm{L}}$ by fixed point iteration
Initialize $x_{\mathrm{L}}^{(0)} = y$, $x_{\mathrm{L}}^{(1)} = \mathcal{J}_1(x_{\mathrm{L}}^{(0)})$ and $t = 1$. //$\mathcal{J}_1(x) = y - \lambda\mathcal{G}'(x)$
while $|x_{\mathrm{L}}^{(t)} - x_{\mathrm{L}}^{(t-1)}| > \tau$ & $t < \kappa$ do
    $x_{\mathrm{L}}^{(t+1)} = \mathcal{J}_1(x_{\mathrm{L}}^{(t)})$.
    if $x_{\mathrm{L}}^{(t+1)} < 0$ then
        return $\hat{x}_{\mathrm{L}} = 0$.
    else
        $\hat{x}_{\mathrm{L}} = x_{\mathrm{L}}^{(t+1)}$.
    end
    Let $t = t + 1$.
end
Compare $\mathcal{F}_y(0)$ and $\mathcal{F}_y(\hat{x}_{\mathrm{L}})$ to identify the optimal solution $x_{\mathrm{L}}^{*}$.
```

problem into several optimization sub-problems, which can be easily solved. Based on that idea, we can solve the non-convex approximation of the tensor recovery models by combing ADMM (or BCD) with GTSVT.

Here, let us take ADMM as an example to solve the following problem.

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \|\boldsymbol{\mathcal{L}}\|_{*,\mathcal{G}} + \lambda\|\boldsymbol{\mathcal{S}}\|_{\mathcal{G}} \quad s.t.\ \boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}}, \tag{2.53}$$

The augmented Lagrangian function of (2.53) can be written as

$$\mathcal{L}_\mu(\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}},\boldsymbol{\mathcal{Y}}) = \|\boldsymbol{\mathcal{L}}\|_{*,\mathcal{G}} + \lambda\|\boldsymbol{\mathcal{S}}\|_{\mathcal{G}} + \langle \boldsymbol{\mathcal{L}}+\boldsymbol{\mathcal{S}}-\boldsymbol{\mathcal{P}},\boldsymbol{\mathcal{Y}}\rangle + \frac{\mu}{2}\|\boldsymbol{\mathcal{L}}+\boldsymbol{\mathcal{S}}-\boldsymbol{\mathcal{P}}\|_F^2. \tag{2.54}$$

Therefore, we can solve (2.53) iteratively:

Algorithm 2.6: Generalized Accelerating Iterative Algorithm (GAI)

```
Input: A real number $y > 0$, a threshold $\lambda > 0$, and a tolerance $\tau > 0$.
Output: $\mathcal{T}_{\mathcal{G}}(y, \lambda) = x^{*}_{\mathrm{G}}$[7].
Let
    $\begin{cases} \mathcal{F}_y(x) = \frac{1}{2}(y-x)^2 + \lambda\mathcal{G}(x) \ , \\ \mathcal{J}_1(x) = y - \lambda\mathcal{G}'(x), \\ \mathcal{J}_2(x) = \mathcal{J}_1(x) - \frac{(\mathcal{J}_1(\mathcal{J}_1(x))-\mathcal{J}_1(x))(\mathcal{J}_1(x)-x)}{\mathcal{J}_1(\mathcal{J}_1(x))-2\mathcal{J}_1(x)+x} \ . \end{cases}$
$a_0 \leftarrow \max\{x | \mathcal{J}_1'(x) = 1 \text{ or } x = 0\}$.
if $\mathcal{F}_y'(a_0) < 0$ then
    // Find $\hat{x}_{\mathrm{G}}$ by fixed point iteration
    Initialize $x_{\mathrm{G}}^{(0)} \leftarrow y$, $t \leftarrow 0$
    while $|\mathcal{J}_1(\mathcal{J}_1(x_{\mathrm{G}}^{(t)})) - 2\mathcal{J}_1(x_{\mathrm{G}}^{(t)}) + x_{\mathrm{G}}^{(t)}| > \tau$ do
        $x_{\mathrm{G}}^{(t+1)} = \mathcal{J}_2(x_{\mathrm{G}}^{(t)})$
        $t \leftarrow t + 1$
    end
    $\hat{x}_{\mathrm{G}} = \mathcal{J}_1(x_{\mathrm{G}}^{(t)})$
else
    return $\hat{x}_{\mathrm{G}} = a_0$
end
If $\mathcal{F}_y(0) > \mathcal{F}_y(\hat{x}_{\mathrm{G}})$, return $x^{*}_{\mathrm{G}} = \hat{x}_{\mathrm{G}}$; otherwise return $x^{*}_{\mathrm{G}} = 0$.
```

Step 1 Given $(\boldsymbol{\mathcal{S}}^{(t)}, \boldsymbol{\mathcal{Y}}^{(t)})$, we can get $\boldsymbol{\mathcal{L}}^{(t+1)}$ by

$$\begin{aligned}\boldsymbol{\mathcal{L}}^{(t+1)} &= \arg\min_{\boldsymbol{\mathcal{L}}} \mathcal{L}_{\mu}(\boldsymbol{\mathcal{L}}, \boldsymbol{\mathcal{S}}^{(t)}, \boldsymbol{\mathcal{Y}}^{(t)}) \\ &= \arg\min_{\boldsymbol{\mathcal{L}}} \frac{1}{\mu}\|\boldsymbol{\mathcal{L}}\|_{*,\mathcal{G}} + \frac{1}{2}\|\boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}}^{(t)} - \boldsymbol{\mathcal{P}} + \frac{1}{\mu}\boldsymbol{\mathcal{Y}}^{(t)}\|_F^2 \\ &= \mathcal{D}_{\mathcal{G}}(-\boldsymbol{\mathcal{S}}^{(t)} + \boldsymbol{\mathcal{P}} - \frac{1}{\mu}\boldsymbol{\mathcal{Y}}^{(t)}, \frac{1}{\mu}) \end{aligned} \tag{2.55}$$

Step 2 Given $(\boldsymbol{\mathcal{L}}^{(t+1)}, \boldsymbol{\mathcal{Y}}^{(t)})$, we get $\boldsymbol{\mathcal{S}}^{(t+1)}$ by

$$\begin{aligned}\boldsymbol{\mathcal{S}}^{(t+1)} &= \arg\min_{\boldsymbol{\mathcal{S}}} \mathcal{L}_{\mu}(\boldsymbol{\mathcal{L}}^{(t+1)}, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{Y}}^{(t)}) \\ &= \arg\min_{E} \frac{\lambda}{\mu}\|\boldsymbol{\mathcal{S}}\|_1 + \frac{1}{2}\|\boldsymbol{\mathcal{L}}^{(t+1)} + \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{P}} + \frac{1}{\mu}\boldsymbol{\mathcal{Y}}^{(t)}\|_F^2 \\ &= \mathcal{T}_{\mathcal{G}}(-\boldsymbol{\mathcal{L}}^{(t+1)} + \boldsymbol{\mathcal{P}} - \frac{1}{\mu}\boldsymbol{\mathcal{Y}}^{(t)}, \frac{\lambda}{\mu}), \end{aligned} \tag{2.56}$$

where $[\mathcal{T}_{\mathcal{G}}(-\boldsymbol{\mathcal{L}}^{(t+1)} + \boldsymbol{\mathcal{P}} - \frac{1}{\mu}\boldsymbol{\mathcal{Y}}^{(t)}, \frac{\lambda}{\mu})]_{i,j,k} = \mathcal{T}_{\mathcal{G}}([-\boldsymbol{\mathcal{L}}^{(t+1)} + \boldsymbol{\mathcal{P}} - \frac{1}{\mu}\boldsymbol{\mathcal{Y}}^{(t)}]_{i,j,k}, \frac{\lambda}{\mu})$ by Algorithm (2.6).

Step 3 Given $(\boldsymbol{\mathcal{L}}^{(t+1)}, \boldsymbol{\mathcal{S}}^{(t+1)}, \boldsymbol{\mathcal{Y}}^{(t)})$, we update $\boldsymbol{\mathcal{Y}}$ by

$$\boldsymbol{\mathcal{Y}}^{(t+1)} = \boldsymbol{\mathcal{Y}}^{(t)} + \mu(\boldsymbol{\mathcal{L}}^{(t+1)} + \boldsymbol{\mathcal{S}}^{(t+1)} - \boldsymbol{\mathcal{P}}). \tag{2.57}$$

Step 4 For given $\rho > 1$, we update $\mu^{(t+1)}$ by

$$\mu = \min(\rho\mu, \bar{\mu}), \tag{2.58}$$

where $\bar{\mu}$ is the upper bound of μ.

2.3.3 Tensor Factorization-Based Algorithm

From the above discussion, we can see that solving the convex and non-convex approximation of these tensor models requires computing t-SVD of a tensor with the size of $I_1 \times I_2 \times I_3$

[69], which costs $\mathcal{O}(I_{(1)}I_{(2)}^2 I_3 + I_1 I_2 I_3 \log I_3)$ and cannot be used to handle large scale tensor data efficiently. To solve these issues, Zhou *et al.* [102] have proposed Tensor Completion by Tensor Factorization (TCTF):

$$\min_{\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}}} \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}\|_F \qquad s.t.\ \mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) = \mathbf{P}_\Omega(\boldsymbol{\mathcal{X}}), \tag{2.59}$$

where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \kappa \times I_3}$, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{\kappa \times I_2 \times I_3}$, and κ obtained by the rank estimation strategy [102] is the estimation for the tensor tubal rank of $\boldsymbol{\mathcal{M}}$. The optimization problem (2.59) can be solved by using ADMM. The total costs at each iteration of the algorithm is $\mathcal{O}(\kappa I_1 I_2 I_3 + I_1 I_2 I_3 \log I_3)$, which achieved significant improved performance for the case of $\kappa \ll I_{(2)}$.

2.4 Representative Applications

In this section, we are going to introduce several representative applications of TRPCA and TC, including image and video denoising, image inpainting, and background subtraction.

2.4.1 Image and Video Denoising

With the development of multimedia, as a basic task in the field of computer vision, images, and video denoising is always a research hotspot. The goal of image and video denoising is to remove various kinds of noise from one or multiple noised images and videos and preserve the details and texture information within the observed visual data.

In the past decade, a large number of image and video denoising methods have been proposed [70, 20, 60, 13, 62, 15, 22, 5, 2], in which the noise with specific distribution (such as Zero-Mean Gaussian Noise and Zero-Mean Gaussian-Impulse Mixed Noise) are assumed to approximate the noise in the real world. Generally speaking, the existing denoising

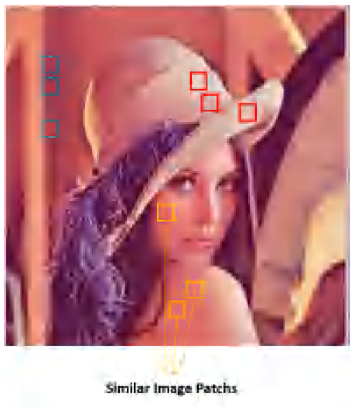


Figure 2.5: Self-similarity of natural image: there are some similar image patchs in one image

methods are mainly classified into local methods [70, 20, 60, 13, 62, 15, 22] and non-local methods [5, 2]. Local methods usually perform kernel convolution operations on the local spatial domain of the noised visual data. Since the local methods do not use the global information and structure in the visual data, resulting in blurring and losing of details in the denoised images. In contrast, non-local methods often achieve a better denoising performance by using the self-similarity property of natural images as shown in Fig. (2.5) [5]. For example, Buades *et al.* [5] have proposed the Non-local Mean Method (NLM) that greatly facilitated the development of the field of image denoising. The NLM makes full use of the self-similarity of natural images, and the value of the center of the reference image block can be estimated after a simple weighted average of these similar blocks, in which the similarity is calculated by the Euclidean distance between the image blocks.

Inspired by NLM, a large number of non-local similar block-based image denoising methods have emerged [56, 19, 16, 28, 91, 92, 69]. The main idea of non-local block-based image denoising methods is to improve the image denoising effect by exploiting the similarity between different similar blocks of the whole image (or within a large domain).

In addition, a potential assumption is that if the matrix formed by column vectorization of those similar blocks is approximated to a low-rank matrix. This low-rank assumption is directly or indirectly used in image denoising algorithms [56, 19, 16, 28, 91]. However, as stated in [92], it will lead to the loss of the spatial information within the image blocks if staking similar blocks into the matrix. To deal with this issue, tensor-based denoising methods have been proposed [92, 69], in which the similar blocks are stacked into tensors instead of the matrix. The tensor-based non-local denoising methods consist of three basic steps: patch grouping [13], low-rank tensor recovery, and aggregation. The detail of each step is explained as follows:

(1) **Patch grouping**: We divide the noisy data $\mathcal{P}$ into a set of overlapping images blocks. Then, we search for K non-local similar image blocks of the given reference image blocks across the whole data by utilizing block matching [13]. For the case of Zero-Mean Gaussian-Impulse Mixed Noise, since the impulse noise will seriously affect block matching results, Zhang *et al.* [91] give use the adaptive center-weighted median filter (ACWMF) to detect the random-valued impulse noise before utilizing block matching and introduce a characteristic tensor to record the position corresponding to impulse noise.

(2) **Low-rank recovery**: Stacking the obtained non-local similar image blocks and the given reference image blocks together, we get a tensor $\mathcal{P}_s$. For the case of mixed noise, $\mathcal{P}_s$ satisfies

$$\mathcal{P}_s = \mathcal{X}_s + \mathcal{Z}_s + \mathcal{E}_s,$$

where $\mathcal{X}_s$ is a clean tensor with low-rankness, $\mathcal{Z}_s$ stand for small noise such as zero-mean Gaussian noise, and $\mathcal{E}_s$ is a sparse tensor composed of impulse noise within $\mathcal{P}_s$.

Therefore, we have

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \|\boldsymbol{\mathcal{X}}_s\|_* + \lambda\|\boldsymbol{\mathcal{E}}_s\|_1 \quad s.t. \ \|\boldsymbol{\mathcal{P}}_s - \boldsymbol{\mathcal{X}}_s - \boldsymbol{\mathcal{E}}_s\|_F \leq \epsilon, \tag{2.60}$$

where λ is a penalty parameter to balance the low-rankness of $\boldsymbol{\mathcal{X}}_s$ and the sparseness of $\boldsymbol{\mathcal{E}}_s$. The problem (2.60) can be converted to

$$(\hat{\boldsymbol{\mathcal{X}}}_{\boldsymbol{s}}, \hat{\boldsymbol{\mathcal{E}}}_{\boldsymbol{s}}) = \arg\min_{\boldsymbol{\mathcal{X}}_s,\boldsymbol{\mathcal{E}}_s} \alpha\|\boldsymbol{\mathcal{X}}_s\|_* + \gamma\|\boldsymbol{\mathcal{E}}_s\|_1 + \frac{1}{2}\|\boldsymbol{\mathcal{P}}_s - \boldsymbol{\mathcal{X}}_s - \boldsymbol{\mathcal{E}}_s\|_F. \tag{2.61}$$

If $\lambda \longrightarrow \infty$, $\|\boldsymbol{\mathcal{E}}_s\|_0 \longrightarrow 0$, and (2.61) is converted to

$$\hat{\boldsymbol{\mathcal{X}}}_{\boldsymbol{s}} = \arg\min_{\boldsymbol{\mathcal{X}}_s} \|\boldsymbol{\mathcal{X}}_s\|_* + \frac{1}{2}\|\boldsymbol{\mathcal{P}}_s - \boldsymbol{\mathcal{X}}_s\|_F. \tag{2.62}$$

Therefore, in addition to the mixed noise, (2.61) can be also used to deal with the case of zero-mean Gaussian noise.

(3) **Aggregation**: We reconstruct the denoised image $\hat{\boldsymbol{\mathcal{X}}}$ by aggregating all the denoised patches $\hat{\boldsymbol{\mathcal{X}}}_{\boldsymbol{s}}$ together.

(4) **Adding back a part of removed noise**: To remain the image detail, we add back a part of removed noise by

$$\hat{\boldsymbol{\mathcal{P}}} = \hat{\boldsymbol{\mathcal{X}}} + \alpha_0(\boldsymbol{\mathcal{P}} - \hat{\boldsymbol{\mathcal{X}}} - \hat{\boldsymbol{\mathcal{E}}}).$$

We repeat the above steps until the algorithm convergence. The whole progress is presented in Fig. 2.6. From the above process, we can see that low-rank recovery plays a crucial role in non-local image denoising. By replacing (2.61) with other variants of TRPCA, we get various non-local image denoising methods.

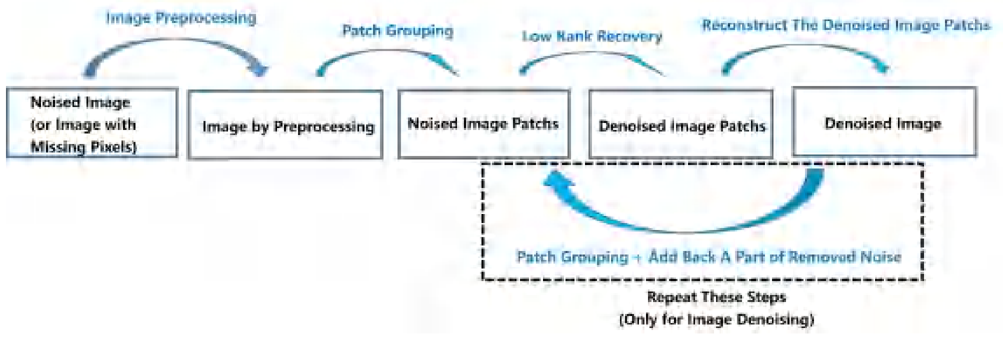


Image Preprocessing
Patch Grouping
Low Rank Recovery
Reconstruct The Denoised Image Patchs
Noised Image (or Image with Missing Pixels)
Image by Preprocessing
Noised Image Patchs
Denoised Image Patchs
Denoised Image
Patch Grouping + Add Back A Part of Removed Noise
Repeat These Steps (Only for Image Denoising)

Figure 2.6: Sequence diagram for image recovery

2.4.2 Image and Video Inpainting

The goal of image and video inpainting is to recover the images and videos from the observed visual data with missing elements. Inspired by the non-local similar block-based image denoising methods, Song *et al.* [67] have proposed the non-local-based image and video inpainting method to make full use of the self-similarity property in natural images. The detail of each step is explained as follows:

(1) **Pre-processing**: Since the missing elements in the images will seriously affect block matching results, Song *et al.* [67] use a triangular-based linear interpolation algorithm [75] to estimate the values in the missing positions. By doing this, we get a preliminary estimate for the clean images, which is denoted as $\hat{\mathcal{P}}$.

(2) **Patch grouping**: Similar to the non-local image denoising method, we get a set of overlapping images blocks by dividing the noisy data $\mathcal{P}$, and then searches for K non-local similar images blocks of the given reference image blocks across $\hat{\mathcal{P}}$ by utilizing block matching [13].

(3) **Low-rank recovery**: Stacking the obtained non-local similar images blocks and the

given reference image blocks together, we get a tensor $\boldsymbol{\mathcal{P}}_s$, which satisfies

$$\mathrm{P}_\Omega(\boldsymbol{\mathcal{P}}_s) = \mathrm{P}_\Omega(\boldsymbol{\mathcal{X}}_s),$$

where $\boldsymbol{\mathcal{X}}_s$ is a recovered tensor with low-rankness, and P_Ω is a linear project operator on the support set Ω composed of the locations corresponding to the observed entries in $\boldsymbol{\mathcal{P}}_s$. Therefore, we have

$$\hat{\boldsymbol{\mathcal{X}}}_{\boldsymbol{s}} = \underset{\boldsymbol{\mathcal{X}}_s}{\arg\min} \|\boldsymbol{\mathcal{X}}_s\|_* \quad s.t.\ \mathrm{P}_\Omega(\boldsymbol{\mathcal{P}}_s) = \mathrm{P}_\Omega(\boldsymbol{\mathcal{X}}_s). \tag{2.63}$$

(4) **Aggregation**: We reconstruct the denoised $\hat{\boldsymbol{\mathcal{X}}}$ by aggregating all the denoised patches $\hat{\boldsymbol{\mathcal{X}}}_{\boldsymbol{s}}$ together.

Similarly, by replacing (2.63) with other variants of TC, we can obtain various non-local image inpainting methods.

2.4.3 Background Subtraction

The task of background modeling is to separate the foreground (moving objects) $\boldsymbol{\mathcal{E}}$ and the background $\boldsymbol{\mathcal{X}}$ in a video $\boldsymbol{\mathcal{P}}$. Therefore, we have

$$\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{E}}.$$

Since the moving objects in the video occupy only a small portion of pixels, the tensor $\boldsymbol{\mathcal{E}}$ corresponding to the foreground in the video is sparse. Besides, since the static background changes slightly, the tensor $\boldsymbol{\mathcal{X}}$ corresponding to the background should be a low-rank tensor. Therefore, background modeling can be regarded as a TRPCA problem [50], which is formulized to

$$(\hat{\boldsymbol{\mathcal{X}}}, \hat{\boldsymbol{\mathcal{E}}}) = \underset{\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{E}}}{\arg\min} \|\boldsymbol{\mathcal{X}}\|_* + \lambda\|\boldsymbol{\mathcal{E}}\|_1 \quad s.t.\ \boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{E}}, \tag{2.64}$$

where $\hat{\boldsymbol{\mathcal{X}}}$ is the separated background.

2.5 Summary

In this chapter, I have introduced various tensor rank functions, namely CP rank, Tucker rank, and t-product-based tensor rank. These rank functions correspond to the three equivalent definitions of matrix rank, *i.e.*, **D1**-**D3**, respectively. In other words, they can be seen as extensions of the three equivalent definitions of matrix rank. Unlike the matrix case, these three tensor functions are equivalent only when tensor order $h = 2$. Because of the superior performance of t-product-based methods in studying the low-rankness of the tensor data, we have mainly introduced several basic t-product-based-tensor recovery methods, their optimization, and several representative application in computer vision, including color images and video denoising, image inpainting, and video background modeling.

From the discussion on low-rank recovery-based non-local image inpainting and denoising, we can observe the critical role played by low-rank recovery models. By utilizing different variants of TRPCA (or TC), we obtain various non-local image denoising methods (or image inpainting methods). However, it is important to note that this dissertation does not aim to provide specific image denoising or inpainting algorithms. Rather, the goal is to investigate an effective approach for defining the tensor rank function that better characterizes the low-rank structure in tensor data. Therefore, in the remaining part of this dissertation, when conducting experiments on data denoising and inpainting, we will not employ any non-local strategies presented in Chapter 2.4. Instead, the entire noisy data (or data with missing elements) will be considered as the observation tensor in the tensor recovery model, such as $\mathcal{P}$ in (2.22), unless otherwise specified.

Chapter 3

Handling Transpose Variability in t-Product-Based Tensor Recovery

3.1 Introduction

Although the tensor ranks based on t-product are effective and widely used, there are still a few limitations: (1) The tensor tubal rank is based on the Discrete Fourier Transformation (DFT) in the 3-rd dimension of the tensor. As a result, the tensor tubal ranks of the resulting tensors obtained by performing different transpose operators on the tensor may be different, which may lead to the tensor recovery results relying on the transpose operators. In this paper, this issue is referred to as Transpose Variability of Tensor Recovery (TVTR). A tensor recovery algorithm has TVTR property if the results of the algorithm are relying on the transpose operators. It can be reasonably imagined that some information within tensor data (the relationship of various views from different dimensions of tensor data) will be lost if only one dimension is considered in a tensor recovery algorithm with TVTR property. (2)

Although it is proven that the true value of the models can be exactly recovered under certain conditions for TRPCA based on ℓ_1-norm (*i.e.*, relax ℓ_0-norm and rank function to ℓ_1-norm and nuclear norm respectively.) These strong conditions often cannot be guaranteed in the real world.

To overcome the aforementioned limitations, this paper focuses on the recovery of a low-rank tensor from a third-order data tensor contaminated by both gross sparse errors and small entry-wise dense noise. The contributions of this work are three-fold. First, to our best knowledge, TVTR is firstly discussed in this paper. Second, to deal with TVTR, a new tensor rank called Weighted Tensor Average Rank (WTAR) is given. Meanwhile, WTAR is applied to the tensor-robust principal component analysis, and a new low-rank tensor recovery model called Tensor Recovery based on WTAR (TRWTAR) is obtained. In addition, we prove that the worst-case error bounds of the recovered tensor are established by TRWTAR (in Theorem 3.3). Third, inspired by the literature on non-convex optimization [23, 57, 24, 71, 87, 26] (see Table 2.2), this paper provides a general algorithm that solves both the convex surrogate and a series of non-convex surrogates of the proposed framework (not limited to the surrogate functions in Table 2.2). The study results contribute to the broad landscape of tensor recovery by delineating an effective measure of tensor rank and providing theoretical and algorithmic advances in robust tensor recovery problems.

3.2 Transpose Variability in Tensor Recovery

It can be seen from Definitions A.13 and A.16 that the tensor tubal rank and tensor nuclear norm are based on t-SVD, in which discrete Fourier transform is applied on the 3-rd dimension of the tensor. Therefore, the transpose operations of the tensor directly affect the tensor recovery methods based on the two norms (including the tensor tubal rank and tensor nuclear norm). An example is given in the following to illustrate this point: Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{2\times 2\times 2}$, in which $[\boldsymbol{\mathcal{A}}]_{:,:,1} = \begin{pmatrix} -0.1241 & 1.4090 \\ 1.4897 & 1.4172 \end{pmatrix}$, $[\boldsymbol{\mathcal{A}}]_{:,:,2} = \begin{pmatrix} 0.6715 & 0.7172 \\ -1.2075 & 1.6302 \end{pmatrix}$.

Definition 3.1. *(Mode-1 conjugate transpose) The conjugate transpose of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{I_1\times I_2\times I_3}$ is denoted as $\boldsymbol{\mathcal{A}}^{T_1} \in \mathbb{C}^{I_1\times I_3\times I_2}$, which is obtained by conjugate transposing each of the horizontal slices.*

Definition 3.2. *(Mode-2 conjugate transpose) The conjugate transpose of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{I_1\times I_2\times I_3}$ is denoted as $\boldsymbol{\mathcal{A}}^{T_2} \in \mathbb{C}^{I_3\times I_2\times I_1}$, which is obtained by conjugate transposing each of the lateral slices.*

Definition 3.3. *(Mode-3 Conjugate transpose) The conjugate transpose of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{I_1\times I_2\times I_3}$ is denoted as $\boldsymbol{\mathcal{A}}^{T_3} \in \mathbb{C}^{I_2\times I_1\times I_3}$, which is obtained by conjugate transposing each of the frontal slices.*

For a third-order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times I_3}$, six tensors are obtained by all possible transpose operations for $\boldsymbol{\mathcal{A}}$: $\boldsymbol{\mathcal{A}} = (\boldsymbol{\mathcal{A}}^{T_1})^{T_1} \in \mathbb{R}^{I_1\times I_2\times I_3}$, $\boldsymbol{\mathcal{B}}_1 = \boldsymbol{\mathcal{A}}^{T_2} \in \mathbb{R}^{I_3\times I_2\times I_1}$, $\boldsymbol{\mathcal{B}}_2 = (\boldsymbol{\mathcal{A}}^{T_2})^{T_3} \in \mathbb{R}^{I_2\times I_3\times I_1}$, $\boldsymbol{\mathcal{B}}_3 = \boldsymbol{\mathcal{A}}^{T_1} \in \mathbb{R}^{I_1\times I_3\times I_2}$, $\boldsymbol{\mathcal{B}}_4 = (\boldsymbol{\mathcal{A}}^{T_1})^{T_3} \in \mathbb{R}^{I_3\times I_1\times I_2}$ and $\boldsymbol{\mathcal{B}}_5 = \boldsymbol{\mathcal{A}}^{T_3} \in \mathbb{R}^{I_2\times I_1\times I_3}$. From the top row of Table 3.1, it can be seen that the tensor singular values of $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{A}}^{T_2}$, and $\boldsymbol{\mathcal{A}}^{T_1}$ are different. Considering the following key optimal problem in low-rank tensor recovery

$$\mathcal{D}(\boldsymbol{\mathcal{Y}}, \lambda) = \underset{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{I_1\times I_2\times I_3}}{\arg\min}\ \lambda\|\boldsymbol{\mathcal{X}}\|_* + \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2, \tag{3.1}$$

this is the proximal operator of the tensor nuclear norm. To solve the problem shown in Eq.(3.1), Liu *et al.* proposed an optimal algorithm (see the Algorithm 2.3) in [50]. The

	$\boldsymbol{\mathcal{A}}$	$\boldsymbol{\mathcal{A}}^{T_2}$	$(\boldsymbol{\mathcal{A}}^{T_2})^{T_3}$	$\boldsymbol{\mathcal{A}}^{T_1}$	$(\boldsymbol{\mathcal{A}}^{T_1})^{T_3}$	$\boldsymbol{\mathcal{A}}^{T_3}$
$\boldsymbol{\sigma}(\cdot)$	$\begin{pmatrix} 3.2988 \\ 0.4407 \end{pmatrix}$	$\begin{pmatrix} 3.1631 \\ 0.9183 \end{pmatrix}$	$\begin{pmatrix} 3.1631 \\ 0.9183 \end{pmatrix}$	$\begin{pmatrix} 3.0859 \\ 1.2910 \end{pmatrix}$	$\begin{pmatrix} 3.0859 \\ 1.2910 \end{pmatrix}$	$\begin{pmatrix} 3.2988 \\ 0.4407 \end{pmatrix}$
$\boldsymbol{\sigma}(\text{bcirc}(\cdot))$	$\begin{pmatrix} 3.7558 \\ 2.8417 \\ 0.5970 \\ 0.2843 \end{pmatrix}$	$\begin{pmatrix} 3.7505 \\ 2.5758 \\ 1.2586 \\ 0.5780 \end{pmatrix}$	$\begin{pmatrix} 3.7505 \\ 2.5758 \\ 1.2586 \\ 0.5780 \end{pmatrix}$	$\begin{pmatrix} 3.3331 \\ 2.8387 \\ 1.5338 \\ 1.0482 \end{pmatrix}$	$\begin{pmatrix} 3.3331 \\ 2.8387 \\ 1.5338 \\ 1.0482 \end{pmatrix}$	$\begin{pmatrix} 3.7558 \\ 2.8417 \\ 0.5970 \\ 0.2843 \end{pmatrix}$

Table 3.1: First row: the tensor singular values of six tensors that are obtained by all possible transpose operations for $\boldsymbol{\mathcal{A}}$. Second row: the singular values of the block circulate matrix of the six tensors.

Algorithm 2.3 reveals that $\mathcal{D}(\boldsymbol{\mathcal{B}}_i, \lambda)$ is not equivalent to the transpose of $\mathcal{D}(\boldsymbol{\mathcal{A}}, \lambda)$ for some λ, when $\boldsymbol{\sigma}(\boldsymbol{\mathcal{A}}) \neq \boldsymbol{\sigma}(\boldsymbol{\mathcal{B}}_i)$. Therefore, it can be concluded that the tensor nuclear norm-based tensor recovery methods have TVTR property. Note that, as stated in [50], Eq.(3.1) is equivalent to

$$\mathcal{D}(\boldsymbol{\mathcal{Y}}, \lambda) = \underset{\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}}{\arg\min} \lambda \|\boldsymbol{\mathcal{X}}\|_{*,\mathrm{a}} + \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2. \tag{3.2}$$

Therefore, for the tensor average nuclear norm-based tensor recovery methods, a similar conclusion can be obtained.

As discussed above, the effectiveness of the tensor recovery methods based on the two norms (including tensor nuclear norm and tensor average nuclear norm) is affected by the transpose operations on the data tensor, but this is ignored by traditional tensor recovery methods. An intuitive approach to solve this problem is to consider all possible transpose operations in the definition of tensor rank.

3.3 Proposed Methods

3.3.1 Weighted Tensor Average Rank

In this section, the TVTR is discussed in detail, and a new tensor rank is given to better explore the low-rank structure within a data tensor.

Definition 3.4. *(Weighted tensor tubal rank) Define weighted tensor tubal rank* $\mathrm{rank}_{\mathrm{wt}}(\cdot)$ *as:*

$$\mathrm{rank}_{\mathrm{wt}}(\boldsymbol{\mathcal{A}}) = \sum_{k=1}^{3} \alpha_k \mathrm{rank}_{\mathrm{t}}(\boldsymbol{\mathcal{A}}^{T_k}), \tag{3.3}$$

where $\alpha_k(k = 1, 2, 3)$ *indicates the weights which sum to* 1.

Definition 3.5. *(Weighted tensor average rank) Define weighted average tensor rank* $\mathrm{rank}_{\mathrm{wa}}(\cdot)$ *as:*

$$\mathrm{rank}_{\mathrm{wa}}(\boldsymbol{\mathcal{A}}) = \sum_{k=1}^{3} \alpha_k \mathrm{rank}_{\mathrm{a}}(\boldsymbol{\mathcal{A}}^{T_k}), \tag{3.4}$$

where $\alpha_k(k = 1, 2, 3)$ *indicates the weights which sum to* 1.

Definition 3.6. *(Weighted tensor nuclear norm) Define weighted tensor nuclear norm* $\|\cdot\|_{\mathrm{wt}}$ *as:*

$$\|\boldsymbol{\mathcal{A}}\|_{\mathrm{wt}} = \sum_{k=1}^{3} \alpha_k \|\boldsymbol{\mathcal{A}}^{T_k}\|_{*}, \tag{3.5}$$

where $\alpha_k(k = 1, 2, 3)$ *indicates the weights which sum to* 1.

Definition 3.7. *(Weighted tensor average nuclear norm) Define weighted tensor average nuclear norm* $\|\cdot\|_{\mathrm{wa}}$ *as:*

$$\|\boldsymbol{\mathcal{A}}\|_{\mathrm{wa}} = \sum_{k=1}^{3} \alpha_k \|\boldsymbol{\mathcal{A}}^{T_k}\|_{*,a}, \tag{3.6}$$

where $\alpha_k(k = 1, 2, 3)$ *indicates the weights which sum to* 1.

Property 3.1. *For* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$*,* $\boldsymbol{\sigma}(\mathrm{bcirc}(\boldsymbol{\mathcal{A}})) = \boldsymbol{\sigma}(\mathrm{bcirc}(\boldsymbol{\mathcal{A}}^{T_3}))$*.*

Proof.

$$\mathrm{bcirc}(\boldsymbol{\mathcal{A}}^{T_3}) = \begin{pmatrix} [\boldsymbol{\mathcal{A}}]^T_{:,:,1} & [\boldsymbol{\mathcal{A}}]^T_{:,:,I_3} & \cdots & [\boldsymbol{\mathcal{A}}]^T_{:,:,2} \\ [\boldsymbol{\mathcal{A}}]^T_{:,:,2} & [\boldsymbol{\mathcal{A}}]^T_{:,:,1} & \cdots & [\boldsymbol{\mathcal{A}}]^T_{:,:,3} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{\mathcal{A}}]^T_{:,:,I_3} & [\boldsymbol{\mathcal{A}}]^T_{:,:,I_3-1} & \cdots & [\boldsymbol{\mathcal{A}}]^T_{:,:,1} \end{pmatrix}$$

$$\longrightarrow \begin{pmatrix} [\boldsymbol{\mathcal{A}}]^T_{:,:,1} & [\boldsymbol{\mathcal{A}}]^T_{:,:,2} & \cdots & [\boldsymbol{\mathcal{A}}]^T_{:,:,I_3} \\ [\boldsymbol{\mathcal{A}}]^T_{:,:,2} & [\boldsymbol{\mathcal{A}}]^T_{:,:,3} & \cdots & [\boldsymbol{\mathcal{A}}]^T_{:,:,1} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{\mathcal{A}}]^T_{:,:,I_3} & [\boldsymbol{\mathcal{A}}]^T_{:,:,1} & \cdots & [\boldsymbol{\mathcal{A}}]^T_{:,:,I_3-1} \end{pmatrix}$$

$$= \begin{pmatrix} [\boldsymbol{\mathcal{A}}]_{:,:,1} & [\boldsymbol{\mathcal{A}}]_{:,:,2} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,I_3} \\ [\boldsymbol{\mathcal{A}}]_{:,:,2} & [\boldsymbol{\mathcal{A}}]_{:,:,3} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,1} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{\mathcal{A}}]_{:,:,I_3} & [\boldsymbol{\mathcal{A}}]_{:,:,1} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,I_3-1} \end{pmatrix}^T$$

$$\longrightarrow \begin{pmatrix} [\boldsymbol{\mathcal{A}}]_{:,:,1} & [\boldsymbol{\mathcal{A}}]_{:,:,I_3} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,2} \\ [\boldsymbol{\mathcal{A}}]_{:,:,2} & [\boldsymbol{\mathcal{A}}]_{:,:,1} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,3} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{\mathcal{A}}]_{:,:,I_3} & [\boldsymbol{\mathcal{A}}]_{:,:,I_3-1} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,1} \end{pmatrix}^T = \mathrm{bcirc}(\boldsymbol{\mathcal{A}})^T. \tag{3.7}$$

Therefore, Property 1 holds. $\square$

Theorem 3.1. *For* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$*, if* $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$*,* $\|\boldsymbol{\mathcal{A}}\|_{\mathrm{wa}} = \|\boldsymbol{\mathcal{A}}^{T_s}\|_{\mathrm{wa}}$ *for* $s = 1, 2, 3$*.*

Proof. For $s = 1$, since $(\boldsymbol{\mathcal{A}}^{T_1})^{T_1} = \boldsymbol{\mathcal{A}}$ and $(\boldsymbol{\mathcal{A}}^{T_1})^{T_2} = (\boldsymbol{\mathcal{A}}^{T_2})^{T_3}$, $\|(\boldsymbol{\mathcal{A}}^{T_1})^{T_1}\|_{*,a} = \|\boldsymbol{\mathcal{A}}\|_{*,a} = \|\boldsymbol{\mathcal{A}}^{T_3}\|_{*,a}$, $\|(\boldsymbol{\mathcal{A}}^{T_1})^{T_2}\|_{*,a} = \|(\boldsymbol{\mathcal{A}}^{T_2})^{T_3}\|_{*,a} = \|\boldsymbol{\mathcal{A}}^{T_2}\|_{*,a}$, and $\|(\boldsymbol{\mathcal{A}}^{T_1})^{T_3}\|_{*,a} = \|\boldsymbol{\mathcal{A}}^{T_1}\|_{*,a}$ by

Property 3.1.
Therefore, $\|\boldsymbol{\mathcal{A}}^{T_1}\|_{\mathrm{wa}} = \sum_{k=1}^{3} \frac{1}{3}\|(\boldsymbol{\mathcal{A}}^{T_1})^{T_k}\|_{*,a} = \dfrac{\|(\boldsymbol{\mathcal{A}}^{T_1})^{T_1}\|_{*,a} + \|(\boldsymbol{\mathcal{A}}^{T_1})^{T_2}\|_{*,a} + \|(\boldsymbol{\mathcal{A}}^{T_1})^{T_3}\|_{*,a}}{3} =$ $\sum_{k=1}^{3} \frac{1}{3}\|\boldsymbol{\mathcal{A}}^{T_k}\|_{*,a} = \|\boldsymbol{\mathcal{A}}\|_{\mathrm{wa}}$.

For $s = 2$, since $(\boldsymbol{\mathcal{A}}^{T_2})^{T_1} = (\boldsymbol{\mathcal{A}}^{T_1})^{T_3}$ and $(\boldsymbol{\mathcal{A}}^{T_2})^{T_2} = \boldsymbol{\mathcal{A}}$, $\|(\boldsymbol{\mathcal{A}}^{T_2})^{T_1}\|_{*,a} = \|(\boldsymbol{\mathcal{A}}^{T_1})^{T_3}\|_{*,a} = \|\boldsymbol{\mathcal{A}}^{T_1}\|_{*,a}$, $\|(\boldsymbol{\mathcal{A}}^{T_2})^{T_2}\|_{*,a} = \|\boldsymbol{\mathcal{A}}\|_{*,a} = \|\boldsymbol{\mathcal{A}}^{T_3}\|_{*,a}$, and $\|(\boldsymbol{\mathcal{A}}^{T_2})^{T_3}\|_{*,a} = \|\boldsymbol{\mathcal{A}}^{T_2}\|_{*,a}$ by Property 3.1.
Therefore, $\|\boldsymbol{\mathcal{A}}^{T_2}\|_{\mathrm{wa}} = \sum_{k=1}^{3} \frac{1}{3}\|(\boldsymbol{\mathcal{A}}^{T_2})^{T_k}\|_{*,a} = \dfrac{\|(\boldsymbol{\mathcal{A}}^{T_2})^{T_1}\|_{*,a} + \|(\boldsymbol{\mathcal{A}}^{T_2})^{T_2}\|_{*,a} + \|(\boldsymbol{\mathcal{A}}^{T_2})^{T_3}\|_{*,a}}{3} =$ $\sum_{k=1}^{3} \frac{1}{3}\|\boldsymbol{\mathcal{A}}^{T_k}\|_{*,a}$.

For $s = 3$, since $(\boldsymbol{\mathcal{A}}^{T_3})^{T_1} = (\boldsymbol{\mathcal{A}}^{T_2})^{T_3}$ and $(\boldsymbol{\mathcal{A}}^{T_3})^{T_2} = (\boldsymbol{\mathcal{A}}^{T_1})^{T_3}$, $\|\boldsymbol{\mathcal{A}}^{T_3}\|_{\mathrm{wa}} = \sum_{k=1}^{3} \frac{1}{3}\|(\boldsymbol{\mathcal{A}}^{T_3})^{T_k}\|_{*,a} =$ $\dfrac{\|(\boldsymbol{\mathcal{A}}^{T_3})^{T_1}\|_{*,a} + \|(\boldsymbol{\mathcal{A}}^{T_3})^{T_2}\|_{*,a} + \|(\boldsymbol{\mathcal{A}}^{T_3})^{T_3}\|_{*,a}}{3} = \dfrac{\|(\boldsymbol{\mathcal{A}}^{T_2})^{T_3}\|_{*,a} + \|(\boldsymbol{\mathcal{A}}^{T_1})^{T_3}\|_{*,a} + \|(\boldsymbol{\mathcal{A}}^{T_3})^{T_3}\|_{*,a}}{3} =$ $\sum_{k=1}^{3} \frac{1}{3}\|\boldsymbol{\mathcal{A}}^{T_k}\|_{*,a}$.

□

Since $\|\boldsymbol{\mathcal{A}}\|_* = \|\boldsymbol{\mathcal{A}}\|_{*,a}$ as stated in [50], we have $\|\boldsymbol{\mathcal{A}}\|_{\mathrm{wa}} = \|\boldsymbol{\mathcal{A}}\|_{\mathrm{wt}}$. Therefore, the following theorem is derived.

Theorem 3.2. *For $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times I_3}$, if $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$, $\|\boldsymbol{\mathcal{A}}\|_{\mathrm{wt}} = \|\boldsymbol{\mathcal{A}}^{T_s}\|_{\mathrm{wt}}$ for $s = 1, 2, 3$.*

3.3.2 Tensor Robust Principal Component Analysis with Weighted Tensor Average Rank

Based on the definition of $\mathrm{rank}_{\mathrm{wt}}(\cdot)$, TPRCA with weighted tensor average rank is defined as follows:

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \mathrm{rank}_{\mathrm{wt}}(\boldsymbol{\mathcal{L}}) + \lambda\|\boldsymbol{\mathcal{S}}\|_0 \quad s.t.\ \|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}\|_F \le \delta, \tag{3.8}$$

where $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}} + \boldsymbol{\mathcal{Z}}$; $\boldsymbol{\mathcal{L}}$ is low-rank; $\boldsymbol{\mathcal{S}}$ is sparse, and $\boldsymbol{\mathcal{Z}}$ is a small noisy perturbation and $\|\boldsymbol{\mathcal{Z}}\|_F \leq \delta$. Since $\mathrm{rank}_{\mathrm{wt}}(\cdot)$ and ℓ_0-norm is discrete, the continuous version of (3.8) is considered, which is defined as follows,

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \|\boldsymbol{\mathcal{L}}\|_{\mathrm{wa},\mathcal{G}} + \lambda\|\boldsymbol{\mathcal{S}}\|_{\mathcal{G}} \quad s.t.\ \|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}\|_F \leq \delta, \tag{3.9}$$

where $\|\boldsymbol{\mathcal{L}}\|_{\mathrm{wa},\mathcal{G}} = \sum_{k=1}^{3} \frac{\alpha_k}{n_k} \sum_{i=1}^{r_k} \mathcal{G}(\sigma_i(\mathrm{bcirc}(\boldsymbol{\mathcal{L}}^{T_k})))$, $\|\boldsymbol{\mathcal{S}}\|_{\mathcal{G}} = \sum_{i_1,i_2,i_3} \mathcal{G}(|[\boldsymbol{\mathcal{S}}]_{i_1,i_2,i_3}|)$, and $\mathcal{G} : \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ is an increasing function. Note that all the surrogate functions of ℓ_0 listed in Table 2.2 satisfy this condition.

Remark 3.1. *From Property 3.1, we can get the same conclusion with Theorem 3.1 easily for* $\|\cdot\|_{\mathrm{wa},\mathcal{G}}$, i.e., *if* $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$, $\|\boldsymbol{\mathcal{A}}\|_{\mathrm{wa},\mathcal{G}} = \|\boldsymbol{\mathcal{A}}^{T_s}\|_{\mathrm{wa},\mathcal{G}}$ *for* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times I_3}$ *and* $s = 1, 2, 3$.

3.3.3 Tensor Recovery with Non-convex Penalty

3.3.3.1 ℓ_p Minimization Formulation

Taking $\mathcal{G}(\cdot)$ in (3.9) as ℓ_p-norm, then (3.9) is turned to

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \|\boldsymbol{\mathcal{L}}\|_{\mathrm{wa},p}^p + \lambda\|\boldsymbol{\mathcal{S}}\|_{p,p}^p \quad s.t.\ \|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}\|_F \leq \delta, \tag{3.10}$$

where $\|\boldsymbol{\mathcal{L}}\|_{\mathrm{wa},p}^p = \sum_{k=1}^{3} \frac{\alpha_k}{n_k} (\sum_{i=1}^{r_k} \sigma_i(\mathrm{bcirc}(\boldsymbol{\mathcal{L}}^{T_k}))^{\frac{1}{p}})^p$, $r_k = \mathrm{rank}(\mathrm{bcirc}(\boldsymbol{\mathcal{L}}^{T_k}))$, and $\|\boldsymbol{\mathcal{S}}\|_{p,p}^p = (\sum_{i_1,i_2,i_3} |[\boldsymbol{\mathcal{S}}]_{i_1,i_2,i_3}|^{\frac{1}{p}})^p$. For convenience, (3.10) is referred to as TRWTAR-ℓ_p (where TRWTAR and ℓ_p stand for Tensor Recovery with Weighted Tensor Average Rank and ℓ_p-norm respectively). It is easy to see that, for $p = 1$, (3.10) reduces to

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \|\boldsymbol{\mathcal{L}}\|_{\mathrm{wa}} + \lambda\|\boldsymbol{\mathcal{S}}\|_1 \quad s.t.\ \|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}\|_F \leq \delta, \tag{3.11}$$

which is referred to as Tensor Recovery with Weighted Tensor Average Nuclear Norm (TRWTANN).

3.3.3.2 Worst-Case Error Bound

Here, an error bound is established under the transformed ℓ_p minimization problem (3.10).

Lemma 3.1. *[6] Let $w_1, w_2, \ldots, w_n$ be n positive numbers such that $\sum_{k=1}^{n} w_k = 1$. Then, for any real numbers s and t such that $0 < s < t < \infty$, and for any $a_1, \ldots, a_n \geq 0$, we have:*

$$\Big(\sum_{k=1}^{n} w_k a_k^s\Big)^{\frac{1}{s}} \leq \Big(\sum_{k=1}^{n} w_k a_k^t\Big)^{\frac{1}{t}}, \tag{3.12}$$

if and only if $a_1 = a_2 = \cdots = a_n$.

Theorem 3.3. *Let $(\boldsymbol{\mathcal{L}}_0, \boldsymbol{\mathcal{S}}_0)$ be the pair of true low-rank and sparse tensors, and $\hat{\boldsymbol{\mathcal{L}}}$ be the solution to the optimization problem* (3.10)*. If the average of the entries of the sparse component $\boldsymbol{\mathcal{S}}_0$ is bounded by T, and the carnality of the support $\boldsymbol{\mathcal{S}}_0$ is bounded by m, then* $\mathrm{Err}(\hat{\boldsymbol{\mathcal{L}}}) = \dfrac{\|\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}\|_F}{M} \leq \sqrt[p]{\frac{2mT^p + \frac{(2\delta)^p}{M^{\frac{p}{2}-1}}}{M^p(1-\frac{1}{\lambda})}}$*, where $\lambda > 1$, $M = \prod_{k=1}^{3} I_k$. Remark $I^{(1)} = I_2$, $I^{(2)} = I_1$ and $I^{(3)} = I_3$.*

Proof. Let $(\hat{\boldsymbol{\mathcal{L}}}, \hat{\boldsymbol{\mathcal{S}}})$ be the optimal solution of (3.10), $\hat{\boldsymbol{\mathcal{Z}}} = \boldsymbol{\mathcal{P}} - \hat{\boldsymbol{\mathcal{S}}} - \hat{\boldsymbol{\mathcal{L}}}$, and $\boldsymbol{\mathcal{Z}}_0 = \boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{S}}_0 - \boldsymbol{\mathcal{L}}_0$. By optimality, we have

$$||\hat{\boldsymbol{\mathcal{L}}}||_{\mathrm{wa},p}^p + \lambda||\boldsymbol{\mathcal{P}} - \hat{\boldsymbol{\mathcal{Z}}} - \hat{\boldsymbol{\mathcal{L}}}||_{p,p}^p \leq ||\boldsymbol{\mathcal{L}}_0||_{\mathrm{wa},p}^p + \lambda||\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{Z}}_0 - \boldsymbol{\mathcal{L}}_0||_{p,p}^p. \tag{3.13}$$

Next, recall that a function $\mathcal{F}(\cdot)$ is sub-additive if $\mathcal{F}(x+y) \leq \mathcal{F}(x) + \mathcal{F}(y)$. According to the result in [63], a concave function $\mathcal{F} : [0, \infty) \to [0, \infty)$ with $\mathcal{F}(0) \geq 0$ is sub-additive. Thus, for $0 < p < 1$, $\mathcal{F}(x) = |x|^p$ is concave (x is a scalar here), $|x|^p$ is a sub-additive function. Since the sum of sub-additive functions is sub-additive, $\mathcal{F}(x) = \|x\|_p^p, x \in \mathbb{R}^n$ is also sub-additive, thereby implying $\|x\|_p^p - \|y\|_p^p \leq \|x-y\|_p^p$. Consequently, Equation (3.13) implies that

$$\begin{aligned} ||\boldsymbol{\mathcal{P}} - \hat{\boldsymbol{\mathcal{Z}}} - \hat{\boldsymbol{\mathcal{L}}}||_{p,p}^p &\leq \frac{1}{\lambda}(||\boldsymbol{\mathcal{L}}_0||_{\text{wa},p}^p - ||\hat{\boldsymbol{\mathcal{L}}}||_{\text{wa},p}^p) + ||\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{Z}}_0 - \boldsymbol{\mathcal{L}}_0||_{p,p}^p \\ &\leq \frac{1}{\lambda}||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_{\text{wa},p}^p + ||\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{Z}}_0 - \boldsymbol{\mathcal{L}}_0||_{p,p}^p, \end{aligned} \tag{3.14}$$

where the last inequality is derived from the linearity property in the definition of $\|\cdot\|_{\text{wa},p}^p$ on tensors. Based on this inequality, $||\hat{\boldsymbol{\mathcal{L}}} - \boldsymbol{\mathcal{L}}_0||_{p,p}^p$ can be bounded as follows:

$$\begin{aligned} &||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_{p,p}^p \leq ||\boldsymbol{\mathcal{P}} - \hat{\boldsymbol{\mathcal{Z}}} - \hat{\boldsymbol{\mathcal{L}}}||_{p,p}^p + ||\boldsymbol{\mathcal{P}} - \hat{\boldsymbol{\mathcal{Z}}} - \boldsymbol{\mathcal{L}}_0||_{p,p}^p \\ \leq& ||\boldsymbol{\mathcal{P}} - \hat{\boldsymbol{\mathcal{Z}}} - \hat{\boldsymbol{\mathcal{L}}}||_{p,p}^p + ||\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{Z}}_0 - \boldsymbol{\mathcal{L}}_0||_{p,p}^p + ||\hat{\boldsymbol{\mathcal{Z}}} - \boldsymbol{\mathcal{Z}}_0||_{p,p}^p \\ \leq& \frac{1}{\lambda}||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_{\text{wa},p}^p + 2||\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{Z}}_0 - \boldsymbol{\mathcal{L}}_0||_{p,p}^p + ||\hat{\boldsymbol{\mathcal{Z}}} - \boldsymbol{\mathcal{Z}}_0||_{p,p}^p \\ =& \frac{1}{\lambda}\sum_{k=1}^{3}\frac{\alpha_k}{I^{(k)}}\Big(\sum_{i=1}^{r_k}(\sigma_i^{(k)})^p\Big) + 2||\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{Z}}_0 - \boldsymbol{\mathcal{L}}_0||_{p,p}^p + ||\hat{\boldsymbol{\mathcal{Z}}} - \boldsymbol{\mathcal{Z}}_0||_{p,p}^p, \end{aligned} \tag{3.15}$$

where the third inequality is derived from substituting the inequality (3.14) into the current inequality; r_k is the rank of the matrix $\text{bcirc}((\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}})^{T_k})$, and $\sigma_1^{(k)}, \sigma_2^{(k)}, \dots, \sigma_{r_k}^{(k)}$ are the r_k singular values of the matrix $\text{bcirc}((\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}})^{T_k})$.

Since $\|\boldsymbol{\mathcal{Z}}_0\|_F \leq \delta$ and $\|\hat{\boldsymbol{\mathcal{Z}}}\|_F \leq \delta$, $\|\boldsymbol{\mathcal{Z}}_0 - \hat{\boldsymbol{\mathcal{Z}}}\|_F \leq 2\delta$. According to Lemma 3.1 and setting $w_j = \frac{1}{M}, \forall j = 1, \dots, M$, we have:

$$||\hat{\boldsymbol{\mathcal{Z}}} - \boldsymbol{\mathcal{Z}}_0||_{p,p}^p \leq M\left(\frac{\|\hat{\boldsymbol{\mathcal{Z}}} - \boldsymbol{\mathcal{Z}}_0\|_F}{\sqrt{M}}\right)^p \leq \frac{(2\delta)^p}{M^{\frac{p}{2}-1}}.$$

By Lemma 3.1, and setting $w_j = \frac{1}{r_k}, \forall j = 1, \dots, r_k$, we have:

$$\frac{(\sigma_1^{(k)})^p + (\sigma_2^{(k)})^p + \cdots + (\sigma_{r_k}^{(k)})^p}{r_k} \leq \left(\sqrt{\frac{(\sigma_1^{(k)})^2 + (\sigma_2^{(k)})^2 + \cdots + (\sigma_{r_k}^{(k)})^2}{r_k}}\right)^p, \tag{3.16}$$

thereby leading to

$$\begin{aligned} \sum_{i=1}^{r_k}(\sigma_i^{(k)})^p \leq r_k^{1-\frac{p}{2}}\Big(\sum_{i=1}^{r_k}(\sigma_i^{(k)})^2\Big)^{\frac{p}{2}} &= r_k^{1-\frac{p}{2}}||\text{bcirc}((\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}})^{T_k})||_F^p \\ &= r_k^{1-\frac{p}{2}}(I^{(k)})^{\frac{p}{2}}||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_F^p. \end{aligned} \tag{3.17}$$

Applying this inequality to the final line in (3.15) results in:

$$||\hat{\boldsymbol{\mathcal{L}}} - \boldsymbol{\mathcal{L}}_0||_{p,p}^p \leq \frac{1}{\lambda}\sum_{k=1}^{3}\frac{\alpha_k}{(I^{(k)})^{1-\frac{p}{2}}} r_k^{1-\frac{p}{2}} ||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_F^p + 2mT^p + \frac{(2\delta)^p}{M^{\frac{p}{2}-1}}. \tag{3.18}$$

Since $||\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{Z}}_0 - \boldsymbol{\mathcal{L}}_0||_{p,p}^p = ||\boldsymbol{\mathcal{S}}_0||_{p,p}^p \leq mT^p$, according to the generalized power-mean inequality in Lemma 3.1 (by setting $s = p, t = 1$), we have:

$$\Big(\frac{||\boldsymbol{\mathcal{S}}_0||_{p,p}^p}{m}\Big)^{\frac{1}{p}} \leq \frac{||\boldsymbol{\mathcal{S}}_0||_{1,1}^1}{m} \leq T. \tag{3.19}$$

Next, we show that $||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_F^p \leq ||\hat{\boldsymbol{\mathcal{L}}} - \boldsymbol{\mathcal{L}}_0||_{p,p}^p$. Denoting $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}$ and based on the fact that $||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_F \leq ||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_1$, we have:

$$\begin{aligned} \|\boldsymbol{\mathcal{L}}\|_F &= \sqrt{\sum_{i_1,i_2,i_3} [\boldsymbol{\mathcal{L}}]_{i_1,i_2,i_3}^2} \leq \sum_{i_1,i_2,i_3} |[\boldsymbol{\mathcal{L}}]_{i_1,i_2,i_3}| \\ &= \Big(\Big(\sum_{i_1,i_2,i_3} |[\boldsymbol{\mathcal{L}}]_{i_1,i_2,i_3}|\Big)^p\Big)^{\frac{1}{p}} \leq \Big(\sum_{i_1,i_2,i_3} |[\boldsymbol{\mathcal{L}}]_{i_1,i_2,i_3}|^p\Big)^{\frac{1}{p}} = \|\boldsymbol{\mathcal{L}}\|_{p,p}, \end{aligned} \tag{3.20}$$

where the second inequality is derived from the fact that $\mathcal{F}(x) = x^p (0 < p < 1)$ is a sub-additive function. Raising both sides to the power of p yields $||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_F^p \leq ||\hat{\boldsymbol{\mathcal{L}}} - \boldsymbol{\mathcal{L}}_0||_{p,p}^p$. Combining this inequality with Equation (3.18), we have:

$$||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_F^p \leq \frac{1}{\lambda}\sum_{k=1}^{3}\frac{\alpha_k}{(I^{(k)})^{1-\frac{p}{2}}} r_k^{1-\frac{p}{2}} ||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_F^p + 2mT^p + \frac{(2\delta)^p}{M^{\frac{p}{2}-1}}. \tag{3.21}$$

Rearranging the terms, we have

$$||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_F^p \leq \frac{2mT^p + \frac{(2\delta)^p}{M^{\frac{p}{2}-1}}}{1 - \frac{1}{\lambda}\sum_{k=1}^{3}\frac{\alpha_k}{(I^{(k)})^{1-\frac{p}{2}}} r_k^{1-\frac{p}{2}}} \leq \frac{2mT^p + \frac{(2\delta)^p}{M^{\frac{p}{2}-1}}}{1 - \frac{1}{\lambda}\sum_{k=1}^{3}\frac{\alpha_k}{(I^{(k)})^{1-\frac{p}{2}}} (I^{(k)})^{1-\frac{p}{2}}}$$

(since $r_k \leq I^{(k)}$ and $1 - \frac{p}{2} > 0$), and therefore

$$||\boldsymbol{\mathcal{L}}_0 - \hat{\boldsymbol{\mathcal{L}}}||_F \leq \sqrt[p]{\frac{2mT^p + \frac{(2\delta)^p}{M^{\frac{p}{2}-1}}}{1 - \frac{1}{\lambda}}}, \tag{3.22}$$

provided that $\lambda > 1$.

□

To give an intuitive understanding of Theorem 3.3, consider the two most simplest cases:

- For $p = 1$, $\delta = 0$, we have:

$$\mathrm{Err}(\hat{\boldsymbol{\mathcal{L}}}) = \frac{2mT}{M(1-\frac{1}{\lambda})},$$

  where $\frac{m}{M} << 1$ is the sparsity coefficient, and T is bounded. Usually, the entries in visual data are typically bounded by a constant that is not too large, *i.e.*, the biggest value of the entry is 255 for images. Thus, the error bound is rather small, indicating rather good recovery.

- For $p = 1, T = 0$, we have:

$$\mathrm{Err}(\hat{\boldsymbol{\mathcal{L}}}) = \frac{2\delta}{M^{\frac{1}{2}}(1-\frac{1}{\lambda})},$$

  where $\frac{1}{M^{\frac{1}{2}}(1-\frac{1}{\lambda})} << 1$ for $\lambda = \infty$. As suggested above, λ in (3.11) should be set to a large enough value.

3.3.4 Optimization Based on Alternating Direction Method

This section introduces a general optimization algorithm for solving (3.9), which can be reduced to

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \alpha \sum_{k=1}^{3} \alpha_k \|\boldsymbol{\mathcal{L}}^{T_k}\|_{*,\mathcal{G}} + \beta \|\boldsymbol{\mathcal{S}}\|_{\mathcal{G}} + \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}\|_F^2. \tag{3.23}$$

To simplify (3.23), a series of auxiliary tensors $\boldsymbol{\mathcal{M}}_k (k = 1, 2, 3)$ are introduced to replace $\boldsymbol{\mathcal{L}}^{T_k}$ and to remove the correlation of $\boldsymbol{\mathcal{L}}^{T_k}$. Then, (3.23) can be rewritten to:

$$\begin{gathered}\min_{\boldsymbol{\mathcal{M}}_k,\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}\|_F^2 + \alpha \sum_{k=1}^{3} \alpha_k \|\boldsymbol{\mathcal{M}}_k\|_{*,\mathcal{G}} + \beta \|\boldsymbol{\mathcal{S}}\|_{\mathcal{G}} \\ s.t. \boldsymbol{\mathcal{L}}^{T_k} = \boldsymbol{\mathcal{M}}_k, k = 1, 2, 3.\end{gathered} \tag{3.24}$$

To relax the above equality constraints, this paper applies the Augmented Lagrange Multiplier (ALM) method to the above problem, and the following augmented Lagrangian function is obtained:

$$\mathcal{L}_\mu(\boldsymbol{\mathcal{M}}_k, \boldsymbol{\mathcal{L}}, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{Q}}_k) = \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}\|_F^2 + \alpha \sum_{k=1}^{3} \alpha_k \|\boldsymbol{\mathcal{M}}_k\|_{*,\mathcal{G}} + \beta\|\boldsymbol{\mathcal{S}}\|_{\mathcal{G}}$$
$$+ \Sigma_{k=1}^3(\langle \boldsymbol{\mathcal{Q}}_k, \boldsymbol{\mathcal{L}}^{T_k} - \boldsymbol{\mathcal{M}}_k\rangle + \frac{\mu_k}{2}\|\boldsymbol{\mathcal{L}}^{T_k} - \boldsymbol{\mathcal{M}}_k\|_F^2), \tag{3.25}$$

where μ_i is a positive scalar, and $\boldsymbol{\mathcal{Q}}_k$ is Lagrange multiplier tensor. According to the framework of the alternating direction method (ADM) [3], the above optimization problem can be iteratively solved as follows.

Step1 Given $\boldsymbol{\mathcal{L}}^{(t)}$ and $\boldsymbol{\mathcal{Q}}_k^{(t)}$, update $\boldsymbol{\mathcal{M}}_k, k = 1, 2, 3$ by

$$\boldsymbol{\mathcal{M}}_k^{(t+1)} = \arg\min_{\boldsymbol{\mathcal{M}}_k} \frac{\mu_k}{2}\|(\boldsymbol{\mathcal{L}}^{(t)})^{T_k} - \boldsymbol{\mathcal{M}}_k + \frac{1}{\mu_k}\boldsymbol{\mathcal{Q}}_k^{(s)}\|_F^2 + \alpha\alpha_k\|\boldsymbol{\mathcal{M}}_k\|_{*,\mathcal{G}}$$
$$= \mathcal{D}_{\mathcal{G}}((\boldsymbol{\mathcal{L}}^{(t)})^{T_k} + \frac{1}{\mu_k}\boldsymbol{\mathcal{Q}}_k^{(t)}, \frac{\alpha\alpha_k}{\mu_k}). \tag{3.26}$$

Step2 Given $\boldsymbol{\mathcal{M}}_k^{(t+1)}$, $\boldsymbol{\mathcal{S}}^{(t)}$ and $\boldsymbol{\mathcal{Q}}_k^{(t)}, k = 1, 2, 3$, update $\boldsymbol{\mathcal{L}}$ by

$$\boldsymbol{\mathcal{L}}^{(t+1)} = \arg\min_{\boldsymbol{\mathcal{L}}} \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{S}}^{(t)}\|_F^2 + \sum_{k=1}^{3} \frac{\mu_k}{2}\|\boldsymbol{\mathcal{L}}^{T_k} - \boldsymbol{\mathcal{M}}_k^{(t+1)} + \frac{1}{\mu_k}\boldsymbol{\mathcal{Q}}_k^{(t)}\|_F^2 \tag{3.27}$$

Calculate the partial derivative of the above formulation with respect to $\boldsymbol{\mathcal{L}}$, and set it to zero.

$$-\boldsymbol{\mathcal{P}} + \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}}^{(t)} + \Sigma_{k=1}^3 \mu_k(\boldsymbol{\mathcal{L}} - (\boldsymbol{\mathcal{M}}_k^{(t+1)} - \frac{1}{\mu_k}\boldsymbol{\mathcal{Q}}_k^{(t)})^{T_k}) = \mathbf{0}$$

By rearranging the term with $\boldsymbol{\mathcal{L}}$, we have

$$\boldsymbol{\mathcal{L}}^{(t+1)} = \frac{\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{S}}^{(t)} + \Sigma_{k=1}^3 \mu_k(\boldsymbol{\mathcal{M}}_k^{(t+1)} - \frac{1}{\mu_k}\boldsymbol{\mathcal{Q}}_k^{(t)})^{T_k}}{1 + \Sigma_{k=1}^3 \mu_k} \tag{3.28}$$

Step3 Given $\boldsymbol{\mathcal{L}}^{(t+1)}$, update $\boldsymbol{\mathcal{S}}$ by

$$\boldsymbol{\mathcal{S}}^{(t+1)} = \arg\min_{\boldsymbol{\mathcal{S}}} \frac{1}{2}\|\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}}^{(t+1)} - \boldsymbol{\mathcal{S}}\|_F^2 + \beta\|\boldsymbol{\mathcal{S}}\|_{\mathcal{G}} = \mathcal{T}_{\mathcal{G}}(\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{L}}^{(t+1)}, \beta). \tag{3.29}$$

Step4 Given $\boldsymbol{\mathcal{Q}}_k^{(t)}$, $\boldsymbol{\mathcal{L}}^{(t+1)}$ and $\boldsymbol{\mathcal{M}}_k^{(t+1)}, k = 1, 2, 3$, update $\boldsymbol{\mathcal{L}}$ by

$$\boldsymbol{\mathcal{Q}}_k^{(t+1)} = \boldsymbol{\mathcal{Q}}_k^{(t)} + \mu_k((\boldsymbol{\mathcal{L}}^{(t+1)})^{T_k} - \boldsymbol{\mathcal{M}}_k^{(t+1)}), \forall k \tag{3.30}$$

3.4 Experiments

In this section, four sets of experiments are conducted to illustrate the effectiveness of our proposed methods. The first set of experiments is performed on the color image data contaminated by zero-mean Gaussian noise, and the proposed methods including TRWTANN and TRWTAR-ℓ_p are compared with several state-of-the-art low-rank tensor recovery methods, including SNN[25], Liu's work (called Liu for short in the following)[45], SRALT-ℓ_p [89], KBR [80] and TRPCA [50]. The second and third sets of experiments are performed on the color image data and video, respectively. All of them are contaminated by the mixture of zero-mean Gaussian noise and random valued impulse noise in different noise levels to test the seven methods. To illustrate the robustness of the proposed methods to outliers in the visual data and their effectiveness in practical applications, in the fourth set of experiments, all seven methods are tested on background subtraction. The source code of SRALT-ℓ_p[1] and KBR[2] are provided by their authors, while the source code of the remaining methods including SNN, Liu's work, and TRPCA are provided by the LibADMM toolbox[3]. The parameters of all methods are tuned to the best for each case. In addition, $\alpha_k (1 \leq k \leq 3)$ in our methods are set to $\frac{1}{3}$.

[1] https://github.com/18357710774/SRALT_code

[2] https://github.com/XieQi2015/KBR-TC-and-RPCA

[3] https://github.com/canyilu/LibADMM-toolbox

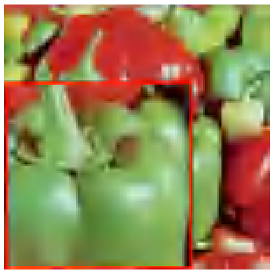
(a)

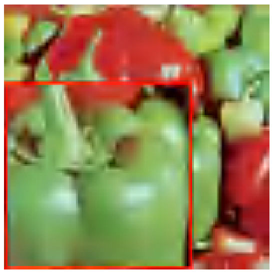
(b)

(c)

(d)

(e)


(f)


(g)

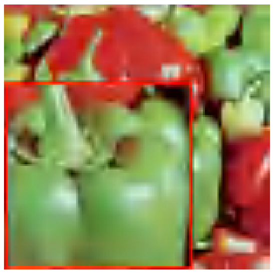

(h)

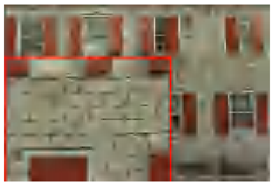
Figure 3.1: Denoised results on “Peppers”, $\delta = 20$. (a) Noised image (b) SNN. (c) Liu. (d) SRALT-ℓ_p. (e) KBR. (f) TRPCA. (g) TRWTANN. (h) TRWTAR-ℓ_p.

3.4.1 Zero-Mean Gaussian Noise: Color Image Denoising

The clean color image with a size of $I_1 \times I_2 \times 3$ can be approximated by low-rank tensor $\boldsymbol{\mathcal{L}}_0 \in \mathbb{R}^{I_1 \times I_2 \times 3}$, and the zero-mean Gaussian noise can be regarded as small entry-wise perturbations $\boldsymbol{\mathcal{Z}}_0 \in \mathbb{R}^{I_1 \times I_2 \times 3}$, which is a tensor with the entries independently sampled from a $\mathcal{N}(0, \delta^2)$ distribution (the noised image can be obtained by $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{L}}_0 + \boldsymbol{\mathcal{Z}}_0$). In this part, all the seven methods (including SNN, Liu, SRALT-ℓ_p, KBR, TRPCA, TRWTANN, and TRWTAR-ℓ_p) are applied to color image recovery in which the color image is contaminated by zero-mean Gaussian noise. All methods are performed on House, Lena, Peppers, F16,

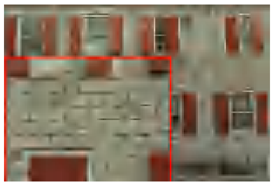
	$\delta = 5$							$\delta = 10$						
	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p
House	29.77	29.58	**36.06**	35.16	31.43	31.88	34.37	28.36	28.25	29.38	29.70	31.07	**31.52**	31.04
Peppers	32.02	31.76	**34.70**	32.21	31.91	32.77	34.55	28.29	28.29	28.66	29.49	30.81	**31.06**	30.97
Lena	32.71	32.57	**35.51**	33.80	32.69	33.53	35.32	28.50	28.56	29.17	30.29	31.46	31.74	**31.82**
Baboon	31.73	31.16	**34.12**	27.65	29.72	30.67	33.15	28.04	27.92	28.22	26.36	28.64	**29.10**	28.66
F16	33.39	33.25	**36.40**	33.27	33.53	34.42	36.39	28.71	28.78	30.15	30.02	31.97	32.12	**32.54**
Kodak image1	33.19	33.06	35.95	33.31	33.99	35.01	**37.24**	28.37	28.57	28.76	30.07	31.62	31.71	**32.35**
Kodak image2	32.68	32.61	36.55	33.90	34.76	35.71	**37.43**	28.29	28.53	28.75	30.75	32.50	32.42	**33.00**
Kodak image3	32.80	32.71	36.85	34.36	34.64	35.60	**37.41**	28.32	28.57	28.89	30.80	32.29	32.29	**32.92**
Kodak image12	33.04	32.99	37.22	36.14	35.51	36.43	**38.12**	28.39	28.65	29.87	31.58	32.95	32.78	**33.71**
Average	32.37	32.19	35.93	33.31	33.13	34.00	**36.00**	28.36	28.46	29.10	29.90	31.48	31.64	**31.89**
	$\delta = 15$							$\delta = 20$						
	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p
House	22.98	22.96	**29.53**	27.11	27.14	26.50	28.94	22.46	22.49	**28.40**	24.78	26.95	26.56	26.59
Peppers	26.23	25.68	28.85	27.22	27.75	27.73	**28.96**	25.24	24.88	27.36	25.43	26.90	27.17	**27.47**
Lena	26.74	26.22	28.96	27.87	28.52	28.58	**29.87**	25.69	25.38	28.22	25.93	27.59	28.02	**28.31**
Baboon	23.30	22.34	25.72	24.73	25.03	24.95	**26.29**	22.65	21.86	**24.84**	23.52	24.54	24.65	24.48
F16	27.14	26.34	28.52	27.59	28.94	28.87	**30.31**	26.00	25.54	28.42	25.58	27.88	28.17	**28.58**
Kodak image1	26.47	26.13	28.35	27.66	28.58	28.71	**29.61**	25.02	25.13	26.53	25.79	27.22	27.54	**28.05**
Kodak image2	27.45	27.79	**30.77**	28.63	30.25	30.39	30.60	25.91	26.76	26.87	26.76	28.56	29.00	**29.71**
Kodak image3	27.16	27.40	30.36	28.41	29.75	29.94	**30.41**	25.61	26.32	27.35	26.44	28.14	28.61	**29.27**
Kodak image12	27.57	27.98	29.53	28.74	30.78	30.95	**31.23**	25.99	26.86	29.67	26.58	28.89	29.38	**30.35**
Average	26.12	25.87	28.96	27.55	28.53	28.51	**29.58**	24.95	25.02	27.52	25.65	27.41	27.68	**28.09**
	$\delta = 25$							$\delta = 30$						
	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p
House	21.34	21.36	**25.24**	23.13	24.22	23.95	24.01	20.94	20.86	22.30	21.62	**24.19**	23.93	24.07
Peppers	24.06	23.74	24.20	23.98	25.04	25.40	**25.51**	23.39	23.14	21.37	22.82	24.63	25.04	**25.36**
Lena	24.66	24.34	25.32	24.27	25.87	26.25	**26.26**	23.98	23.70	22.22	22.90	25.46	25.96	**26.26**
Baboon	21.23	20.85	**22.92**	22.34	22.42	22.69	22.54	20.82	20.44	20.79	21.36	22.27	**22.56**	22.54
F16	24.90	24.45	**27.44**	24.01	26.13	26.34	26.38	24.14	23.84	24.67	22.72	25.65	26.02	**26.32**
Kodak image1	24.25	24.24	23.50	24.24	25.76	26.14	**26.15**	23.39	23.59	20.98	22.96	25.10	25.40	**25.76**
Kodak image2	25.65	26.15	23.20	25.23	27.87	28.20	**28.23**	24.68	25.41	20.63	23.92	26.91	27.17	**27.71**
Kodak image3	25.15	25.50	23.62	24.80	27.13	27.57	**27.71**	24.17	24.71	20.93	23.80	26.23	26.62	**27.07**
Kodak image12	25.73	26.23	27.70	24.88	28.24	28.60	**28.70**	24.70	25.44	23.96	23.48	27.14	27.43	**28.13**
Average	24.11	24.09	24.79	24.10	25.85	26.13	**26.17**	23.36	23.46	21.98	22.84	25.29	25.57	**25.91**

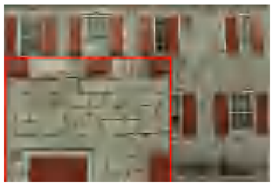
Table 3.2: Color image denoising results (PSNR) by different methods for the case of zero-mean Gaussian noise.

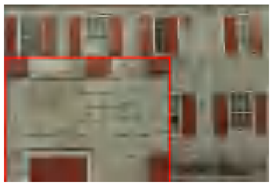

(a)

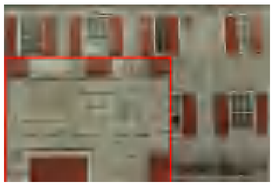
(b)

(c)


(d)


(e)


(f)


(g)

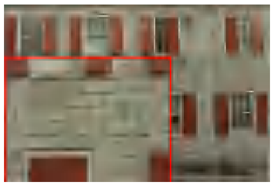

(h)

Figure 3.2: Denoised results on “kodak image1”, $\delta = 30$. (a) Noised image (b) SNN. (c) Liu. (d) SRALT-ℓ_p. (e) KBR. (f) TRPCA. (g) TRWTANN. (h) TRWTAR-ℓ_p.

Baboon, and the $1 - 3^{th}$ and 12^{th} images from the Kadak PhotoCD [4]. Meanwhile, the standard deviations of zero-mean Gaussian noise δ are set to $\delta = \{5, 10, 15, 20, 25, 30\}$.

Table 3.2 shows the Peak Signal-To-Noise Ratio (PSNR) results of different methods when the image data is corrupted by zero-mean Gaussian noise and the highest PSNR values are marked in bold. The visual quality performance of all the methods is reported in Figs.3.1-3.2. From these results, the following observations are made. First, the PSNR results of the proposed methods (TRWTANN and TRWTAR-ℓ_p) and the other five methods (SNN, Liu, SRALT-ℓ_p, KBR, and TRPCA) indicate that TRWTANN and TRWTAR-ℓ_p achieve the best denoising performance in most cases. Specially, for the case of $\delta = 15$, TRWTAR-ℓ_p even outperforms the five comparing methods by at least 1dB on average PSNR. This illustrates the effectiveness of the methods based on the weighted tensor average rank for handling

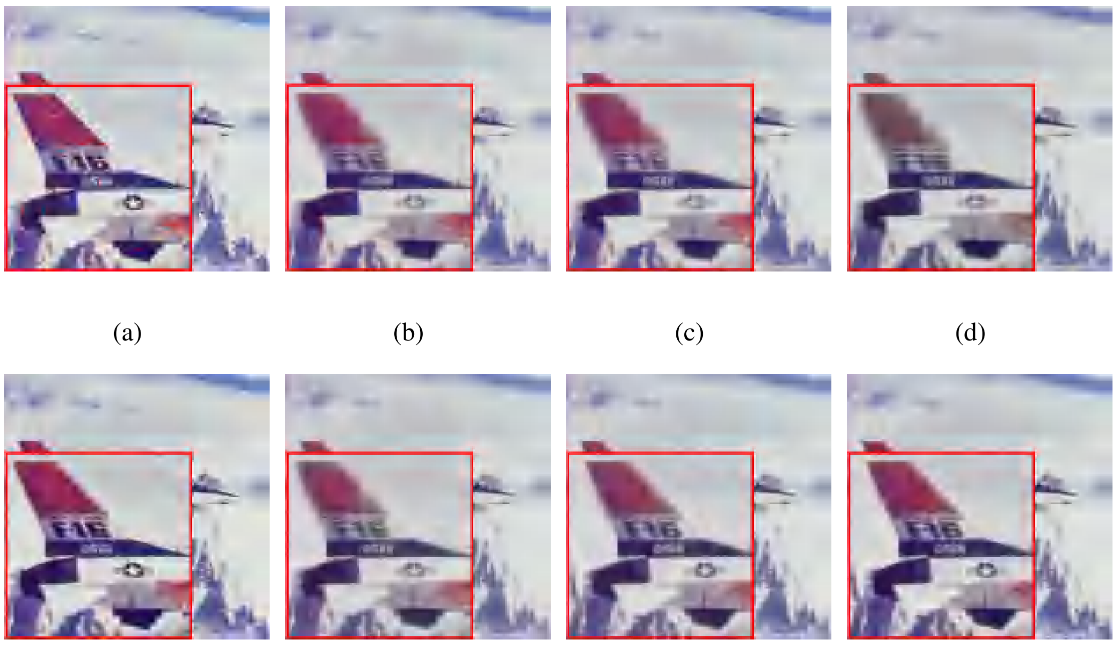
[4] https://webpages.tuni.fi/foi/GCF-BM3D/index.html

Gaussian noise. Besides, the PSNR results of TRWTANN and TRWTAR-ℓ_p indicate that using the non-convex surrogate strategy given in this paper can improve the effectiveness of the original method (TRWTANN) significantly. In addition, from Figs.3.1-3.2, it can be seen that the three tensor recovery methods based on t-product (including TRPCA, TRWTANN, and TRWTAR-ℓ_p) retain more information and details about the image, while the denoised images obtained by SNN and Liu appear some white stripes. For the remaining two methods including SRALT-ℓ_p and KBR, there are still some residual noise within the denoised image. This validates the effectiveness of the methods based on the t-product.

3.4.2 Zero-Mean Gaussian-Impulse Mixed Noise: Color Image Denoising

In this part, the proposed models are applied to image recovery, where the color image is contaminated by the mixture of zero-mean Gaussian noise $\boldsymbol{\mathcal{Z}}_0$ and random valued impulse noise. Because the clean color image can be approximated by low-rank tensors, and the random valued impulse noise with density level c can be regarded as sparse errors $\boldsymbol{\mathcal{S}}_0$[5], the noise can be removed from the color images $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{L}}_0 + \boldsymbol{\mathcal{Z}}_0 + \boldsymbol{\mathcal{S}}_0$ by all the seven methods (including SNN, Liu, SRALT-ℓ_p, KBR, TRPCA, TRWTANN, and TRWTAR-ℓ_p). All the methods are tested on the testing image set that contains House, Lena, Peppers, F16, Baboon, and the $1-3^{th}$ and 12^{th} images from the Kadak PhotoCD. Meanwhile, the noise is set to zero-mean Gaussian noise with standard deviations δ and random-valued impulse noise with density level c. Besides, in this experiment, (δ, c) is set to $(\delta, c) = \{(0, 5\%), (5, 5\%), (5, 10\%), (15, 10\%), (15, 15\%), (30, 15\%)\}$.

[5] $3cI_1I_2$ entries in $\boldsymbol{\mathcal{S}}_0$ uniformly distributed in $[0, 255]$, and the remain entries in $\boldsymbol{\mathcal{S}}_0$ are zeros.


(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.3: Denoised results on “F16”, $(\delta, c) = (15, 10\%)$. (a) Noised image (b) SNN. (c) Liu. (d) SRALT-ℓ_p. (e) KBR. (f) TRPCA. (g) TRWTANN. (h) TRWTAR-ℓ_p.

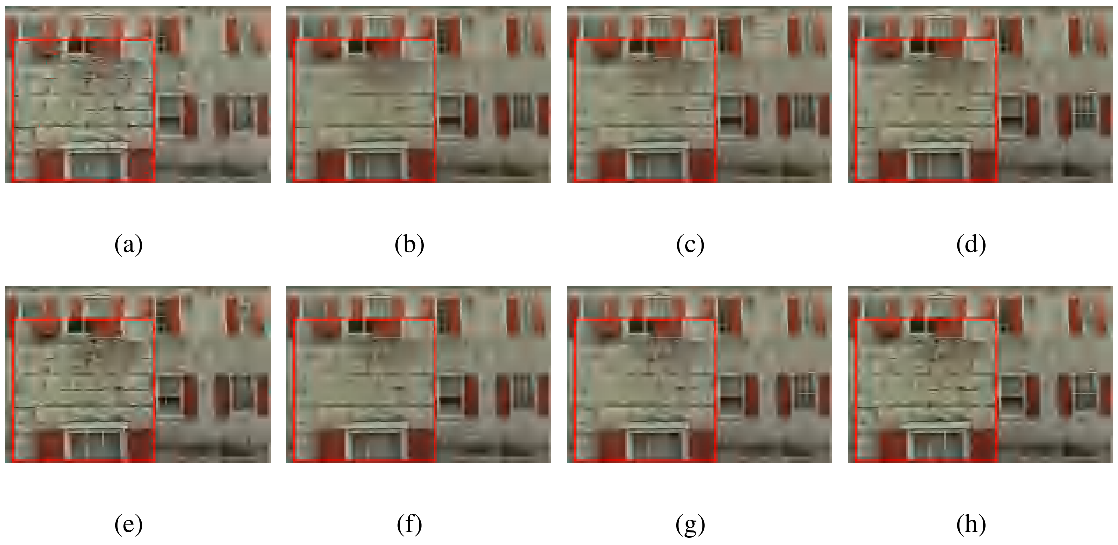


Figure 3.4: Denoised results on “Kodak image1”, $(\delta, c) = (15, 10\%)$. (a) Noised image (b) SNN. (c) Liu. (d) SRALT-ℓ_p. (e) KBR. (f) TRPCA. (g) TRWTANN. (h) TRWTAR-ℓ_p.

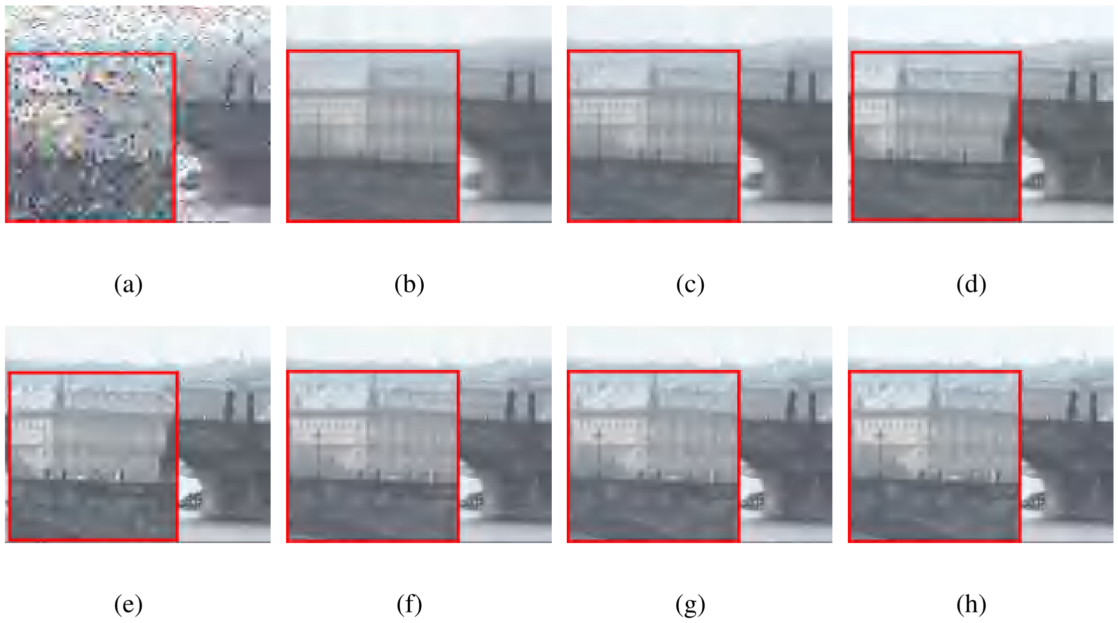


Figure 3.5: Denoised results on “bridge-close”, $(\delta, c) = (10, 20\%)$. (a) Noised data (b) SNN. (c) Liu. (d) SRALT-ℓ_p. (e) KBR. (f) TRPCA. (g) TRWTANN. (h) TRWTAR-ℓ_p.

	$(\delta, c) = (0, 5\%)$							$(\delta, c) = (5, 5\%)$						
	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p
Baboon	29.72	29.61	26.81	27.06	29.50	29.43	**29.99**	28.32	28.20	25.73	26.56	28.34	28.76	**29.34**
F16	35.84	36.03	34.66	33.91	35.71	36.40	**37.69**	31.81	31.79	32.19	32.09	32.22	33.17	**34.01**
House	30.36	31.15	32.78	**39.41**	31.53	32.63	32.77	29.73	30.19	30.99	**33.51**	30.65	30.85	30.71
Lena	34.66	34.73	35.56	35.61	34.88	35.48	**36.31**	31.37	31.27	32.47	32.80	31.87	32.82	**33.31**
Peppers	33.09	33.21	33.56	33.15	32.62	33.58	**34.40**	30.68	30.61	29.68	31.38	30.69	31.46	**31.92**
Kodak image1	34.23	35.07	30.15	33.15	**38.68**	37.11	38.61	30.49	30.82	27.67	31.99	32.15	33.20	**33.99**
Kodak image2	34.39	35.67	27.56	35.21	37.21	37.80	**39.49**	31.42	31.54	26.00	33.15	32.12	33.62	**34.25**
Kodak image3	35.37	35.69	30.35	35.87	35.99	37.40	**38.96**	31.41	31.55	27.71	33.38	31.91	33.45	**34.06**
Kodak image12	36.09	36.48	35.12	38.29	38.48	38.98	**40.46**	31.63	31.81	32.23	34.78	32.56	34.32	**34.85**
Average	33.86	34.48	31.84	34.63	34.96	35.42	**36.52**	30.76	30.86	29.41	32.18	31.39	32.41	**32.94**
	$(\delta, c) = (10, 10\%)$							$(\delta, c) = (15, 10\%)$						
	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p
Baboon	23.23	23.16	20.48	24.46	22.74	23.95	**25.26**	22.67	22.57	23.04	23.15	22.21	23.41	**23.55**
F16	26.97	26.72	22.43	28.21	26.50	27.67	**28.89**	25.92	25.73	24.96	25.67	25.65	26.84	**26.85**
House	23.24	24.03	23.40	**26.87**	24.58	24.97	26.06	22.77	23.51	**24.56**	24.30	23.92	24.55	24.13
Lena	27.14	27.09	24.24	28.60	27.11	28.00	**29.04**	26.08	25.96	26.32	26.16	26.05	27.14	**27.22**
Peppers	26.48	26.34	24.25	27.81	25.80	26.88	**27.91**	25.49	25.36	**26.29**	25.60	24.94	26.06	26.11
Kodak image1	26.66	26.50	23.85	27.88	27.37	27.87	**28.78**	25.25	25.23	25.85	25.61	25.93	26.12	**26.65**
Kodak image2	28.21	28.24	27.23	29.22	28.37	29.34	**29.44**	26.56	26.74	26.02	26.88	27.05	27.49	**27.50**
Kodak image3	28.03	28.10	26.55	29.23	27.98	29.05	**29.30**	26.31	26.52	26.66	26.81	26.63	27.15	**27.30**
Kodak image12	28.33	28.38	25.71	29.60	28.89	**29.94**	29.92	26.61	26.80	27.61	26.93	27.40	27.79	**27.85**
Average	26.48	26.51	24.24	27.99	26.59	27.52	**28.29**	25.30	25.38	25.70	25.68	25.53	26.28	**26.35**
	$(\delta, c) = (15, 15\%)$							$(\delta, c) = (30, 15\%)$						
	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p
Baboon	20.64	20.81	21.67	22.27	19.98	22.38	**22.41**	19.67	19.78	20.01	19.71	19.04	**20.12**	20.06
F16	24.02	24.04	24.48	25.18	23.04	25.41	**25.85**	22.48	22.53	**22.88**	21.21	21.82	22.83	22.87
House	21.66	21.70	20.05	**24.90**	21.64	23.16	22.96	20.47	20.36	18.03	**20.59**	20.28	20.58	20.38
Lena	24.69	24.76	25.21	25.55	24.04	26.19	**26.38**	22.84	22.88	22.82	21.09	22.32	**23.50**	23.48
Peppers	23.89	23.95	23.98	24.71	22.73	25.04	**25.29**	22.12	22.14	22.01	21.01	21.10	22.36	**22.48**
Kodak image1	24.40	24.46	25.49	25.02	24.19	25.59	**25.94**	22.65	22.67	23.06	21.11	22.56	23.09	**23.14**
Kodak image2	26.74	26.74	26.03	26.44	26.23	26.86	**27.36**	24.65	24.62	24.13	22.00	24.73	24.68	**24.84**
Kodak image3	26.39	26.28	25.52	26.18	25.49	26.58	**27.10**	23.98	23.85	22.92	22.07	23.69	24.15	**24.30**
Kodak image12	26.84	26.83	27.06	26.22	26.55	27.17	**27.82**	24.61	24.61	24.49	21.92	24.75	24.79	**24.99**
Average	24.36	24.40	24.39	25.16	23.77	25.38	**25.68**	22.61	22.60	22.26	21.19	22.25	22.90	**22.95**

Table 3.3: Color image denoising results (PSNR) by different methods for the case of zero-mean Gaussian-impulse mixed noise.

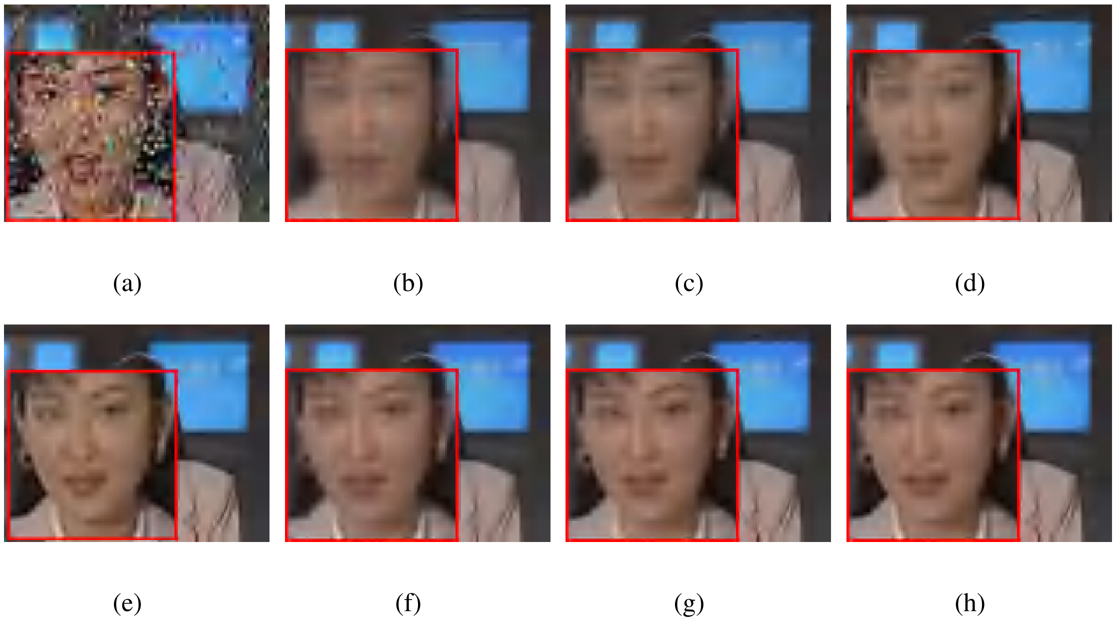


Figure 3.6: Denoised results on “akiyo”, $(\delta, c) = (10, 20\%)$. (a) Noised data (b) SNN. (c) Liu. (d) SRALT-ℓ_p. (e) KBR. (f) TRPCA. (g) TRWTANN. (h) TRWTAR-ℓ_p.

(δ, c)	Video sequence	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p
$(5, 10\%)$	templete	23.24	24.12	23.88	24.09	24.26	25.05	**25.18**
	grandma	28.07	29.98	31.20	32.32	32.07	32.55	**32.74**
	akiyo	27.02	28.40	29.46	29.94	30.53	31.04	**31.28**
	bus	23.20	22.81	20.94	20.58	23.45	24.52	**24.75**
	mobile	21.07	21.46	20.39	18.01	21.48	22.73	**23.03**
	bridge-close	27.06	28.52	29.75	31.09	30.78	31.17	**31.31**
	bridge-far	30.87	31.89	33.49	35.36	34.82	35.06	**35.77**
	Average	25.79	26.74	27.01	27.34	28.19	28.87	**29.15**
$(10, 20\%)$	templete	19.71	20.67	20.85	20.66	21.09	21.44	**21.52**
	grandma	25.44	26.71	28.21	28.72	28.42	28.77	**29.27**
	akiyo	24.20	24.82	26.18	27.60	27.16	27.52	**27.99**
	bus	19.51	20.44	19.73	19.95	21.41	**21.85**	21.73
	mobile	17.88	18.52	18.64	17.39	21.15	**20.84**	20.18
	bridge-close	24.88	25.92	28.07	29.43	28.59	29.00	**29.68**
	bridge-far	28.19	28.84	31.97	32.21	31.48	31.52	**32.59**
	Average	22.83	23.70	24.80	25.13	25.61	25.84	**26.13**

Table 3.4: Results on video data with Gaussian noise and random-valued impulse noise.

All the methods are evaluated by the PSNR value and visual results. From Table 3.3, it can be seen that the proposed methods (TRWTANN and TRWTAR-ℓ_p) outperform SNN, Liu, SRALT-ℓ_p, KBR, and TRPCA by a large margin in all cases on PSNR values. As shown in Figs.3.3-3.4, the proposed methods retain more details in the denoised images. These results indicate the superiority of the proposed methods. The performance superiority is achieved by considering different tensor transpose operations in the progress of estimating the latent low-rank tensor, which makes use of the information within the tensor data as much as possible. This illustrates that the new tensor rank (WTAR) given in this paper (see Definitions 3.4-3.5) is more reasonable in real applications than others.

3.4.3 Zero-Mean Gaussian-Impulse Mixed Noise: Video Sequence Denoising

Similar to the case of color image denoising, video sequence denoising can also be regarded as a low-rank tensor recovery problem. In this case, each color frame of the video is folded in the third dimension of the data tensor $\hat{\boldsymbol{\mathcal{L}}}_0 \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times 3}$ (corresponding to the color video with the size of $I_1 \times I_2 \times I_3 \times 3$) to obtain clean tensor data $\boldsymbol{\mathcal{L}}_0 \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times 3}$. Then, TRWTANN and TRWTAR-ℓ_p are compared with the other five methods including SNN, Liu, SRALT-ℓ_p, KBR, and TRPCA on the video sequences contaminated by mixed noise to demonstrate the effectiveness of the proposed model. In this experiment, (δ, c) is set to $(\delta, c) = \{(5, 10\%), (10, 20\%)\}$. Seven wildly used test videos are taken from the YUV Video Sequences to form the testing video set [6], including templete, grandma, akiyo, bus, mobile, bridge-close, and bridge-far. The size of each frame is 144×176, and only the first

[6] http://trace.eas.asu.edu/yuv/

Video Clip	SNN	Liu	SRALT-ℓ_p	KBR	TRPCA	TRWTANN	TRWTAR-ℓ_p
Airport	0.2632	0.2787	0.3485	0.0859	0.1972	0.3770	**0.3837**
Hall	0.4258	0.5440	0.5408	**0.5548**	0.4412	0.5534	0.5492
Office	0.3158	0.5278	0.5081	**0.5763**	0.1874	0.5552	0.5736
Pedestrian	0.2957	**0.4882**	0.4661	0.4124	0.3177	0.4554	0.4546
Smoke	0.1138	**0.6249**	0.6063	0.5160	0.0233	0.5515	0.5881
Average	0.2829	0.4927	0.4940	0.4291	0.2334	0.4985	**0.5098**

Table 3.5: Background subtraction results of different methods.

30 frames of each video are chosen for testing.

All the methods are also evaluated by the PSNR value and visual results, and the evaluation results are listed in Table 3.4 and Figs.3.5-3.6. From these results, the following observations can be obtained: (1) The methods based on t-product (including TRPCA, TRWTANN, and TRWTAR-ℓ_p) obtains better results than other methods (including SNN, Liu, SRALT-ℓ_p, and KBR) in the case of video denoising. As shown in Figs.3.5-3.6, the methods based on t-product retain more information about the video. This is because all three methods based on t-product have a recovery guarantee. Also, they can find the low-rank subspace of tensor data more exactly and utilize the information within the real data more effectively than other low-rank tensor recovery methods under mixed noise. (2) The proposed methods (including TRWTANN and TRWTAR-ℓ_p) are more effective than other comparing five methods. Specially, TRWTAR-ℓ_p outperforms other comparing methods by at least 0.5dB on average PSNR value. This indicates that the proposed methods guarantee a more accurate low-rank recovery than other comparing methods, and they are more robust against noise and outliers. (3) In most cases, the results obtained by TRWTAR-ℓ_p are better than those obtained by TRWTANN, indicating the effectiveness of the general algorithm given in this paper.

3.4.4 Background Subtraction

In this part, the proposed models are applied to the background subtraction task that aims to separate the foreground objects from the background. The background of each frame of the video is static and similar, and it can be regarded as a low-rank tensor $\boldsymbol{\mathcal{L}}_0$. Meanwhile, the moving foreground objects can be regarded as sparse noise $\boldsymbol{\mathcal{S}}_0$, because they occupy only a fraction of pixels in the video. Therefore, all the seven methods including SNN, Liu, SRALT-ℓ_p, KBR, TRPCA, TRWTANN, and TRWTAR-ℓ_p are tested on the five video sequences [7] to deal with the case of background subtraction.

To measure the background modeling output quantitatively, $S(\boldsymbol{A}, \boldsymbol{B}) = \frac{\boldsymbol{A} \bigcap \boldsymbol{B}}{\boldsymbol{A} \bigcup \boldsymbol{B}}$ is used to calculate the similarity between the estimated foreground regions and the ground truths. The quantitative results of different methods are listed in Table 3.5, and it can be seen that the proposed model achieves the best results. Also, the following observations can be made. First, TRPCA performs poorly in this experiment. This is because the exact recovery [50] and the stable recovery of TRPCA require that the support Ω of the true latent sparse tensor is uniformly distributed. However, this condition is not met in the background subtraction application because the moving foreground objects are composed of several contiguous regions. The proposed methods (TRWTANN and TRWTAR-ℓ_p) can fix this problem well. This is because they consider different transpose operators to make use of the information within the tensor data effectively, and they perform stably against the outliers. In addition, it should be noted that Liu needs some additional effort to tune the weighted parameters empirically. By contrast, in our methods, all of $\alpha_k (1 \leq k \leq 3)$ are set to $\frac{1}{3}$ so that the proposed methods can be applied to real applications more easily.

[7] http://perception.i2r.a-star.edu.sg/bk model/bk index.html

3.5 Summary

In this work, TVTR is discussed at first. It is discovered that if different transpose operators are performed on the observation tensor, different results will be obtained by the tensor recovery algorithm with TVTR property. To solve this issue, TRWTANN is taken to study the resulting tensor by a series of transpose operators on the observation sensor, and the information within the tensor data is utilized more effectively. Besides, to balance the solvability and effectiveness of TRWTANN, the non-convex version (3.9) of TRWTANN, *i.e.*, TRWTAR-ℓ_p, is investigated. Then, the worst-case error bounds of the recovered tensor are given, and a non-convex optimization algorithm based on generalized tensor singular value thresholding (GSVT) is designed to solve the proposed model (3.11) and its non-convex version (3.9). The experimental results validate the effectiveness of the proposed methods. The work presented in this chapter has been published [93].

It is worth noting that the t-product-based tensor recovery still has the TV for other invertible linear transforms because the t-product-based rank is only based on the invertible linear transforms along with the third dimension of the tensor. But, we still can adopt the same idea with the weighted tensor average rank for other cases, *i.e.*, considering the low t-product-based tensor rank from different dimensions of data tensors.

Chapter 4

Handling Slice Permutations Variability in DFT-Based Tensor Recovery

4.1 Introduction

Although tensor recovery based on t-product is effective and widely used, there are still some limitations: as shown in Fig. 4.1, rearranging frontal slices sequence order of tensor will have a significant influence on the effectiveness of tensor recovery, in which $\hat{\mathcal{X}}^*$ is obtained by arranging the low-rank approximation of $\hat{\mathcal{Y}}$ ($\hat{\mathcal{Y}}$ is obtained by rearranging $\mathcal{Y}$ in randomly frontal slices sequence order) in original frontal slices sequence order. Note that the gap of two mean PSNR (Peak Signal to Noise Ratio) results even achieve 3dB. We called this phenomenon as Slice Permutations Variability (SPV) in tensor recovery.

This paper focuses on this new problem which has not been explored so far to the best of our knowledge. Our contributions are three-fold:

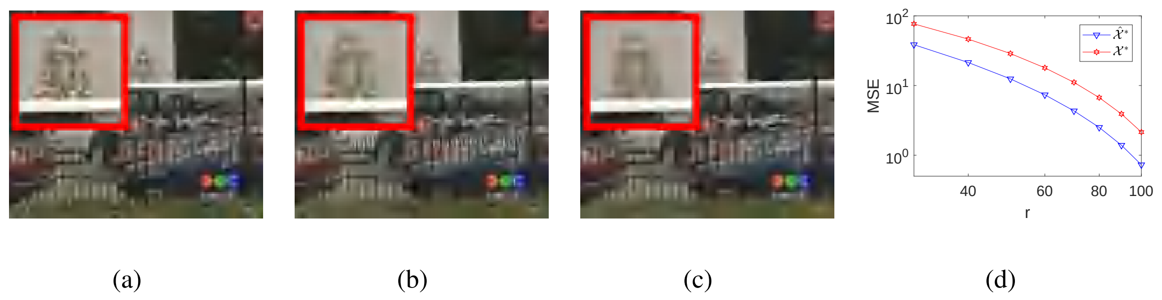


Figure 4.1: Color video ('bus') (modeled as a tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{144\times176\times90}$) can be approximated by low tubal rank tensor. Here, only the first frame of visual results in (a)-(b) are presented. (a) The first frame of the original video (b) approximation by tensor $\boldsymbol{\mathcal{X}}^* \in \mathbb{R}^{144\times176\times90}$ with tubal rank $r = 30$. (MPSNR=32.45dB) (c) approximation by tensor $\hat{\boldsymbol{\mathcal{X}}}^* \in \mathbb{R}^{144\times176\times90}$ with tubal rank $r = 30$. (MPSNR=29.27dB) (d) MSE results of $\boldsymbol{\mathcal{X}}^*$ and $\hat{\boldsymbol{\mathcal{X}}}^*$ comparison for different r.

- We study SPV and Slice Permutations Invariance (SPI) of tensor recovery theoretically and experimentally for the first time. A tensor recovery algorithm has SPI, *i.e.*whatever how to change the slice order of data tensor, the solution of the algorithm will not be changed. We prove that the tensor recovery algorithm has SPI property under certain conditions.

- When the conditions are not met, to make tensor recovery more stable for slice permutations on data tensor, we propose a tensor recovery algorithm for SPV (TRSPV) to solve a basic problem (Tensor Principal Component Analysis) in tensor recovery. In the proposed algorithm, we find a better sequence of tensor slices by solving a Minimum Hamiltonian Circle problem. Based on the new sequence obtained by the proposed algorithm, we can extract the intrinsic low-dimensional structure of high-dimensional tensor data more exactly.

- We conduct experiments to examine the SPV of TRPCA, the goal of which is to recover a low-rank tensor from a high-dimensional data tensor with chaos slices sequence despite both small entry-wise noise and gross sparse errors. An extension of TRSPV, Robust Principal Component Analysis for SPV (TRPCA-SPV), is proposed to deal with this problem. The experimental results show a much better performance of TRPCA-SPV compared with the existing state-of-the-art tensor recovery algorithms, and a huge gap between the results of TRPCA-SPV and TRPCA.

4.2 Slice Permutations Variability in Tensor Recovery

4.2.1 SPI of the Sum of Nuclear Norms

For matrix recovery, as we all know, singular values of the matrix will not be affected by any row or column transformations on the matrix, which means it does not make any influence on the effectiveness of matrix recovery to rearrange the data sequence. And we call it to row or column transformations invariance in matrix recovery (Property 4.2 and Theorem 4.1). Therefore, for tensor recovery based on the unfolding matrices of the tensor, SPV is satisfied naturally (Property 4.3 and Theorem 4.2). Please refer to the supplementary material of this paper for the detailed proof of these conclusions.

Definition 4.1. *[95] $\boldsymbol{P} \in \mathbb{R}^{N\times N}$ is a permutation matrix if each row and each column of $\boldsymbol{P}$ has unique non-zero entries* 1.

Property 4.1. *[95] If $\boldsymbol{P} \in \mathbb{R}^{N\times N}$ is a permutation matrix, then $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}$.*

Property 4.2. *For $\boldsymbol{A} \in \mathbb{R}^{I_1\times I_2}$, then nuclear norm satisfies row (or column) permutations* *invariance,* i.e.$\|\boldsymbol{PA}\|_* = \|\boldsymbol{A}\|_*$ *for any permutation matrix* $\boldsymbol{P} \in \mathbb{R}^{I_1\times I_1}$ *(or* $\|\boldsymbol{AP}\|_* = \|\boldsymbol{A}\|_*$ *for any permutation matrix* $\boldsymbol{P} \in \mathbb{R}^{I_2\times I_2}$*).*

Proof. $\|\boldsymbol{PA}\|_* = \|\boldsymbol{A}\|_*$ by Property 4.1 and the unitary invariant norm property.

Similarly, we can get $\|\boldsymbol{AP}\|_* = \|\boldsymbol{A}\|_*$ for any permutation matrix $\boldsymbol{P} \in \mathbb{R}^{I_2\times I_2}$. □

Theorem 4.1. *For* $\boldsymbol{Y} \in \mathbb{R}^{I_1\times I_2}$*,* $\mathcal{D}(\boldsymbol{Y},\tau) = \boldsymbol{P}^{-1}\mathcal{D}(\boldsymbol{PY},\tau)$ *for any permutation matrix* $\boldsymbol{P} \in \mathbb{R}^{I_1\times I_1}$ *(and* $\mathcal{D}(\boldsymbol{Y},\tau) = \mathcal{D}(\boldsymbol{YP},\tau)\boldsymbol{P}^{-1}$ *for any permutation matrix* $\boldsymbol{P} \in \mathbb{R}^{I_2\times I_2}$*), where* $\mathcal{D}(\boldsymbol{Y},\tau) = \arg\min_X \frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{X}\|_F^2 + \tau\|\boldsymbol{X}\|_*$*, and* $\boldsymbol{P}^{-1}$ *is inverse operator of* $\boldsymbol{P}$*.*

Proof.

$$\begin{aligned}\boldsymbol{P}^{-1}\mathcal{D}(\boldsymbol{PY},\tau) &= \boldsymbol{P}^{-1}\arg\min_{\boldsymbol{Z}} \frac{1}{2}\|\boldsymbol{PY}-\boldsymbol{Z}\|_F^2 + \tau\|\boldsymbol{Z}\|_* \\ &= \arg\min_{\boldsymbol{X}} \frac{1}{2}\|\boldsymbol{PY}-\boldsymbol{PX}\|_F^2 + \tau\|\boldsymbol{PX}\|_* \\ &= \arg\min_{\boldsymbol{X}} \frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{X}\|_F^2 + \tau\|\boldsymbol{X}\|_*, \end{aligned} \tag{4.1}$$

where the second equation holds by letting $\boldsymbol{X} = \boldsymbol{P}^{-1}\boldsymbol{Z}$, and the third equation holds by the Property 4.1 and Property 4.2.

Similarly, we can get $\mathcal{D}(\boldsymbol{Y},\tau) = \mathcal{D}(\boldsymbol{YP},\tau)\boldsymbol{P}^{-1}$ for any permutation matrix $\boldsymbol{P} \in \mathbb{R}^{I_2\times I_2}$. □

Definition 4.2. *[4] Let* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times I_3}$*,* $\mathrm{C} = \{i_1, i_2, ..., i_{I_3}, i_1\}$ *is a circle on* $\boldsymbol{\mathcal{A}}$ *which composed of* 1, 2, 3,..., I_3*. And we regard* $\{i_1, i_2, ..., i_{I_3}, i_1\}$, $\{i_2, i_3, ..., i_{I_3}, i_1, i_2\}$,...,$\{i_{I_3}, i_1, ..., i_{I_3-2}, i_{I_3-1}, i_{I_3}\}$ *as the same circle.*

Definition 4.3. *Let* $\mathrm{C}_k = \{i_1, i_2, ..., i_{I_k}, i_1\}$ *is a circle on* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times I_3}$ *which composed of* 1, 2, 3,..., I_k*. And we call* $\vec{\mathrm{O}}_k = \{i_1, i_2, ..., i_{I_k}\}$ *is an obtained ordered array from* C_k*.*

Define $\vec{\mathrm{O}}(i)$ *is the* i*-th number of the ordered array,* $\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(k)}_{\vec{\mathbf{O}}_k}(k=1,2,3)$ *are the results by horizontal slice permutations, lateral slice permutations and frontal slice permutations on* $\boldsymbol{\mathcal{A}}$ *according to* $\vec{\mathrm{O}}_k$*, respectively,* i.e.$[\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(1)}_{\vec{\mathbf{O}}_1}]_{i,:,:} = [\boldsymbol{\mathcal{A}}]_{\vec{\mathbf{O}}_1(i),:,:}$, $[\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(2)}_{\vec{\mathbf{O}}_2}]_{:,i,:} = [\boldsymbol{\mathcal{A}}]_{:,\vec{\mathbf{O}}_2(i),:}$ *and* $[\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{O}}_3}]_{:,:,i} = [\boldsymbol{\mathcal{A}}]_{:,:,\vec{\mathbf{O}}_3(i)}$ *for* $i = 1,2,3,...,I_k$*. (If there is no danger of ambiguity, these are abbreviated to* $\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(k)}(k=1,2,3)$*.) Besides, we use* $\boldsymbol{\mathcal{A}}^{\vec{\mathbf{O}}}$ *to represent* $\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{O}}}$ *for convenience.*

Property 4.3. *For* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$*, then* $\sum_{i=1}^{3} \alpha_i \|(\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(k)})_{(i)}\|_* = \sum_{i=1}^{3} \alpha_i \|\boldsymbol{\mathcal{A}}_{(i)}\|_*$ *for any slice permutations* $\mathcal{P}^{(k)}$ i.e.$(k=1,2,3)$*, where* $\boldsymbol{\mathcal{A}}_{(i)}$ *represents the mode-*i *unfolding matrix of* $\boldsymbol{\mathcal{A}}$*,* $\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(k)}(k=1,2,3)$ *stands for the result by performing horizontal slice permutations, lateral slice permutations, and frontal slice permutations on* $\boldsymbol{\mathcal{A}}$*, respectively.*

Proof. For any slice permutations $\mathcal{P}^{(k)}(k=1,2,3)$, exist permutation marries $\boldsymbol{P}_i$ and $\boldsymbol{Q}_i$ makes $\mathrm{unfold}_i(\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(k)}) = \boldsymbol{P}_i \boldsymbol{A}_{(i)} \boldsymbol{Q}_i$ for $i = 1,2,3$. Therefore, $\sum_{i=1}^{3} \alpha_i \|\mathrm{unfold}_i(\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(k)})\|_* = \sum_{i=1}^{3} \alpha_i \|\boldsymbol{P}_i \boldsymbol{A}_{(i)} \boldsymbol{Q}_i\|_* = \sum_{i=1}^{3} \alpha_i \|\boldsymbol{A}_{(i)}\|_*$. □

Theorem 4.2. $\mathcal{S}_\tau(\boldsymbol{\mathcal{Y}}) = \mathcal{S}_\tau(\boldsymbol{\mathcal{Y}} \circ \mathcal{P}^{(k)}) \circ (\mathcal{P}^{(k)})^{-1}(k=1,2,3)$*, where* $\mathcal{S}_\tau(\boldsymbol{\mathcal{Y}}) = \arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 + \tau \sum_{i=1}^{3} \frac{1}{3}\|\boldsymbol{X}_{(i)}\|_*$*, and* $(\mathcal{P}^{(k)})^{-1}$ *is an inverse operator of* $\mathcal{P}^{(k)}$*.*

Proof.

$$
\begin{aligned}
&\mathcal{S}_\tau(\boldsymbol{\mathcal{Y}} \circ \mathcal{P}^{(k)}) \circ (\mathcal{P}^{(k)})^{-1} \\
=&(\arg\min_{\boldsymbol{\mathcal{Z}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} \circ \mathcal{P}^{(k)} - \boldsymbol{\mathcal{Z}}\|_F^2 + \tau \sum_{i=1}^{3} \alpha_i \|\boldsymbol{Z}_{(i)}\|_*) \circ (\mathcal{P}^{(k)})^{-1} \\
=&\arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} \circ \mathcal{P}^{(k)} - \boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(k)}\|_F^2 + \tau \sum_{i=1}^{3} \alpha_i \|\mathrm{unfold}_i(\boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(k)})\|_* \\
=&\arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 + \tau \sum_{i=1}^{3} \alpha_i \|\boldsymbol{X}_{(i)}\|_*, \qquad (4.2)
\end{aligned}
$$

where the second equation holds by letting $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{Z}} \circ (\mathcal{P}^{(k)})^{-1}$, and the third equation holds by the property of $\mathcal{P}^{(k)}$ and Property 4.3. □

4.2.2 SPI of Tensor Nuclear Norm

In this part, we study the SPI of the tensor nuclear norm. And we have the following conclusions. Please refer to the supplementary material of this paper for the detailed proof of these conclusions.

Property 4.4. *(Horizontal SPI of tensor nuclear norm) Tensor nuclear norm satisfies HSPI (Horizontal SPI),* i.e.$\|\boldsymbol{\mathcal{A}}\|_* = \|\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(1)}\|_*$, *for any horizontal slice permutations* $\mathcal{P}^{(1)}$.

Proof. By the definition of $\mathrm{bcirc}(\boldsymbol{\mathcal{A}})$, exist two permutation matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ such that $\mathrm{bcirc}(\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(1)}) = \boldsymbol{P} \cdot \mathrm{bcirc}(\boldsymbol{\mathcal{A}}) \cdot \boldsymbol{Q}$. Therefore, $\|\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(1)}\|_* = \|\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(1)}\|_{a,*} = \frac{1}{I_3}\|\mathrm{bcirc}(\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(1)})\|_* = \frac{1}{I_3}\|\boldsymbol{P} \cdot \mathrm{bcirc}(\boldsymbol{\mathcal{A}}) \cdot \boldsymbol{Q}\|_*$. By Property 4.2, $\frac{1}{I_3}\|\boldsymbol{P} \cdot \mathrm{bcirc}(\boldsymbol{\mathcal{A}}) \cdot \boldsymbol{Q}\|_* = \frac{1}{I_3}\|\mathrm{bcirc}(\boldsymbol{\mathcal{A}})\|_* = \|\boldsymbol{\mathcal{A}}\|_{a,*} = \|\boldsymbol{\mathcal{A}}\|_*$. Thus $\|\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(1)}\|_* = \|\boldsymbol{\mathcal{A}}\|_*$.

□

Property 4.5. *(Lateral SPI of tensor nuclear norm) tensor nuclear norm satisfies LSPI (Lateral SPI),* i.e.$\|\boldsymbol{\mathcal{A}}\|_* = \|\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(2)}\|_*$, *for any lateral slices permutations* $\mathcal{P}^{(2)}$.

Proof. Similar to the proof of Property 4.4. □

Property 4.6. *For same circle* $\mathrm{C}^1 = \{i_1, i_2, ..., i_{I_3}, i_1\}$ *and* $\mathrm{C}^2 = \{i_k, i_{k+1}, ..., i_{I_3}, ..., i_{k-1}, i_k\}$,

$$\|\boldsymbol{\mathcal{A}}^{\vec{\mathrm{O}}^1}\|_* = \|\boldsymbol{\mathcal{A}}^{\vec{\mathrm{O}}^2}\|_*,$$

where $\vec{\mathrm{O}}^1 = \{i_1, i_2, ..., i_{I_3}\}$ *is obtained by* C^1, *and* $\vec{\mathrm{O}}^2 = \{i_k, i_{k+1}, ..., i_{I_3}, ..., i_{k-1}\}$ *is obtained by* C^2.

Proof.

$$\begin{aligned}
\operatorname{bcirc}(\boldsymbol{\mathcal{A}}^{\vec{\mathbf{O}}^1}) &= \begin{pmatrix} [\boldsymbol{\mathcal{A}}]_{:,:,i_1} & [\boldsymbol{\mathcal{A}}]_{:,:,i_{I_3}} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,i_3} & [\boldsymbol{\mathcal{A}}]_{:,:,i_2} \\ [\boldsymbol{\mathcal{A}}]_{:,:,i_2} & [\boldsymbol{\mathcal{A}}]_{:,:,i_1} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,i_4} & [\boldsymbol{\mathcal{A}}]_{:,:,i_3} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ [\boldsymbol{\mathcal{A}}]_{:,:,i_{I_3-1}} & [\boldsymbol{\mathcal{A}}]_{:,:,i_{I_3-2}} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,i_1} & [\boldsymbol{\mathcal{A}}]_{:,:,i_{I_3}} \\ [\boldsymbol{\mathcal{A}}]_{:,:,i_{I_3}} & [\boldsymbol{\mathcal{A}}]_{:,:,i_{I_3-1}} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,i_2} & [\boldsymbol{\mathcal{A}}]_{:,:,i_1} \end{pmatrix} \\
&\longrightarrow \begin{pmatrix} [\boldsymbol{\mathcal{A}}]_{:,:,i_k} & [\boldsymbol{\mathcal{A}}]_{:,:,i_{k-1}} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,i_{k+2}} & [\boldsymbol{\mathcal{A}}]_{:,:,i_{k+1}} \\ [\boldsymbol{\mathcal{A}}]_{:,:,i_{k+1}} & [\boldsymbol{\mathcal{A}}]_{:,:,i_k} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,i_{k+3}} & [\boldsymbol{\mathcal{A}}]_{:,:,i_{k+2}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ [\boldsymbol{\mathcal{A}}]_{:,:,i_{k-2}} & [\boldsymbol{\mathcal{A}}]_{:,:,i_{k-3}} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,i_k} & [\boldsymbol{\mathcal{A}}]_{:,:,i_{k-1}} \\ [\boldsymbol{\mathcal{A}}]_{:,:,i_{k-1}} & [\boldsymbol{\mathcal{A}}]_{:,:,i_{k-2}} & \cdots & [\boldsymbol{\mathcal{A}}]_{:,:,i_{k+1}} & [\boldsymbol{\mathcal{A}}]_{:,:,i_k} \end{pmatrix} \\
&= \operatorname{bcirc}(\boldsymbol{\mathcal{A}}^{\vec{\mathbf{O}}^2}). \qquad (4.3)
\end{aligned}$$

Therefore $\|\boldsymbol{\mathcal{A}}^{\vec{\mathbf{O}}^1}\|_* = \|\boldsymbol{\mathcal{A}}^{\vec{\mathbf{O}}^1}\|_{a,*} = \frac{1}{I_3}\|\operatorname{bcirc}(\boldsymbol{\mathcal{A}}^{\vec{\mathbf{O}}^1})\|_* = \frac{1}{I_3}\|\operatorname{bcirc}(\boldsymbol{\mathcal{A}}^{\vec{\mathbf{O}}^2})\|_* = \|\boldsymbol{\mathcal{A}}^{\vec{\mathbf{O}}^2}\|_{a,*} = \|\boldsymbol{\mathcal{A}}^{\vec{\mathbf{O}}^2}\|_*$. □

The symbols and definitions used in Property 4.6 are explained in Definitions 4.2-4.3.

Theorem 4.3. *For same circle* $\mathrm{C}^1 = \{i_1, i_2, ..., i_{I_3}, i_1\}$ *and* $\mathrm{C}^2 = \{i_k, i_{k+1}, ..., i_{I_3}, ..., i_{k-1}, i_k\}$,

$$\mathcal{D}(\boldsymbol{\mathcal{Y}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{O}}^1}, \tau) \circ \mathcal{P}^{(3)^{-1}}_{\vec{\mathbf{O}}^1} = \mathcal{D}(\boldsymbol{\mathcal{Y}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{O}}^2}, \tau) \circ \mathcal{P}^{(3)^{-1}}_{\vec{\mathbf{O}}^2} \qquad (4.4)$$

where $\mathcal{D}(\boldsymbol{\mathcal{A}}, \tau) = \arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{X}}\|_F^2 + \tau\|\boldsymbol{\mathcal{X}}\|_*$, $\vec{\mathbf{O}}^1 = \{i_1, i_2, ..., i_{I_3}\}$ *is obtained by* C^1*, and* $\vec{\mathbf{O}}^2 = \{i_k, i_{k+1}, ..., i_{I_3}, ..., i_{k-1}\}$ *is obtained by* C^2.

Proof.

$$
\begin{aligned}
&(\arg\min_{\boldsymbol{\mathcal{Z}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^1} - \boldsymbol{\mathcal{Z}}\|_F^2 + \tau\|\boldsymbol{\mathcal{Z}}\|_*) \circ \mathcal{P}^{(3)^{-1}}_{\vec{\mathbf{o}}^1} \\
=&\arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^1} - \boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^1}\|_F^2 + \tau\|\boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^1}\|_* \\
=&\arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 + \tau\|\boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^1}\|_*,
\end{aligned}
$$

where the first equation holds by letting $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{Z}} \circ (\mathcal{P}^{(3)}_{\vec{\mathbf{o}}^1})^{-1}$.

By Property 4.6, $\|\boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^1}\|_* = \|\boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^2}\|_*$. Therefore,

$$
\begin{aligned}
&\arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 + \tau\|\boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^1}\|_* \\
=&\arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 + \tau\|\boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^2}\|_* \\
=&\arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^2} - \boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^2}\|_F^2 + \tau\|\boldsymbol{\mathcal{X}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^2}\|_* \\
=&(\arg\min_{\boldsymbol{\mathcal{Z}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}^2} - \boldsymbol{\mathcal{Z}}\|_F^2 + \tau\|\boldsymbol{\mathcal{Z}}\|_*) \circ \mathcal{P}^{(3)^{-1}}_{\vec{\mathbf{o}}^2},
\end{aligned}
$$

where the third equation holds by letting $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} \circ (\mathcal{P}^{(3)}_{\vec{\mathbf{o}}^2})$. The conclusion holds. □

Property 4.7. *For $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, if $I_3 \leq 3$, then tensor nuclear norm satisfies frontal slice permutations invariance (FSPI),* i.e.$\|\boldsymbol{\mathcal{A}}\|_* = \|\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}}\|_*$ *for any frontal slice permutations $\mathcal{P}^{(3)}_{\vec{\mathbf{o}}}$.*

Proof. For $I_3 = 2$, let $[\boldsymbol{\mathcal{B}}]_{:,:,1} = [\boldsymbol{\mathcal{A}}]_{:,:,2}$ and $[\boldsymbol{\mathcal{B}}]_{:,:,2} = [\boldsymbol{\mathcal{A}}]_{:,:,1}$. Thus $\mathrm{bcirc}(\boldsymbol{\mathcal{B}}) = \begin{pmatrix} [\boldsymbol{\mathcal{B}}]_{:,:,1} & [\boldsymbol{\mathcal{B}}]_{:,:,2} \\ [\boldsymbol{\mathcal{B}}]_{:,:,2} & [\boldsymbol{\mathcal{B}}]_{:,:,1} \end{pmatrix} = \begin{pmatrix} [\boldsymbol{\mathcal{A}}]_{:,:,2} & [\boldsymbol{\mathcal{A}}]_{:,:,1} \\ [\boldsymbol{\mathcal{A}}]_{:,:,1} & [\boldsymbol{\mathcal{A}}]_{:,:,2} \end{pmatrix} \longrightarrow \begin{pmatrix} [\boldsymbol{\mathcal{A}}]_{:,:,1} & [\boldsymbol{\mathcal{A}}]_{:,:,2} \\ [\boldsymbol{\mathcal{A}}]_{:,:,2} & [\boldsymbol{\mathcal{A}}]_{:,:,1} \end{pmatrix} = \mathrm{bcirc}(\boldsymbol{\mathcal{A}})$. Therefore, $\|\boldsymbol{\mathcal{A}}\|_* = \|\boldsymbol{\mathcal{A}}\|_{*,a} = \|\boldsymbol{\mathcal{B}}\|_{*,a} = \|\boldsymbol{\mathcal{B}}\|_*$.

For $I_3 = 3$, there is only one circle. Therefore, $\|\boldsymbol{\mathcal{A}}\|_* = \|\boldsymbol{\mathcal{A}} \circ \mathcal{P}^{(3)}_{\vec{\mathbf{o}}}\|_*$ by Property 4.6.

Thus, the conclusion holds.

□

Theorem 4.4. *For $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, if $I_3 \leq 3$, then*

$$\mathcal{D}(\mathcal{Y}, \tau) = \mathcal{D}(\mathcal{Y} \circ \mathcal{P}^{(k)}, \tau) \circ \mathcal{P}^{(k)^{-1}} \tag{4.5}$$

for $k = 1, 2, 3$.

Proof.

$$\begin{aligned} \mathcal{D}(\mathcal{Y} \circ \mathcal{P}^{(k)}, \tau) \circ (\mathcal{P}^{(k)})^{-1} =& (\arg\min_{\mathcal{Z}} \frac{1}{2}\|\mathcal{Y} \circ \mathcal{P}^{(k)} - \mathcal{Z}\|_F^2 + \tau\|\mathcal{Z}\|_*) \circ (\mathcal{P}^{(k)})^{-1} \\ =& \arg\min_{\mathcal{X}} \frac{1}{2}\|\mathcal{Y} \circ \mathcal{P}^{(k)} - \mathcal{X} \circ \mathcal{P}^{(k)}\|_F^2 + \tau\|\mathcal{X} \circ \mathcal{P}^{(k)}\|_* \\ =& \arg\min_{\mathcal{X}} \frac{1}{2}\|\mathcal{Y} - \mathcal{X}\|_F^2 + \tau\|\mathcal{X}\|_*, \end{aligned} \tag{4.6}$$

where the second equation holds by letting $\mathcal{X} = \mathcal{Z} \circ (\mathcal{P}^{(k)})^{-1}$, and the third equation holds by the property of $\mathcal{P}^{(k)}$ and Property 4.4, 4.2.2 and 4.7. □

Although, for $I_3 > 3$, we have taken an example that contradicts the SPI of tensor recovery based on the tensor-tensor product (see Fig. 4.1). By Theorem 4.4, it can be seen that tensor nuclear norm-based tensor recovery satisfies slice permutations invariance for $I_3 \leq 3$.

4.3 The Proposed Method for SPV

In the following, we consider the case of $I_3 > 3$.

4.3.1 Tensor Principal Component Analysis for SPV

Consider the following key problem:

$$\min_{\boldsymbol{\mathcal{X}},\vec{\mathbf{O}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}}^{\vec{\mathbf{O}}} - \boldsymbol{\mathcal{X}}\|_F^2 + \tau\|\boldsymbol{\mathcal{X}}\|_*. \tag{4.7}$$

Since $\|\boldsymbol{\mathcal{X}}\|_* = \|\boldsymbol{\mathcal{X}}\|_{*,a}$ [50], therefore (4.7) can be converted to

$$\begin{aligned}&\min_{\boldsymbol{\mathcal{X}},\vec{\mathbf{O}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}}^{\vec{\mathbf{O}}} - \boldsymbol{\mathcal{X}}\|_F^2 + \tau\|\boldsymbol{\mathcal{X}}\|_{*,a}\\ =&\min_{\boldsymbol{\mathcal{X}},\vec{\mathbf{O}}} \frac{1}{2I_3}\|\mathrm{bcirc}(\boldsymbol{\mathcal{Y}}^{\vec{\mathbf{O}}}) - \mathrm{bcirc}(\boldsymbol{\mathcal{X}})\|_F^2 + \frac{\tau}{I_3}\|\mathrm{bcirc}(\boldsymbol{\mathcal{X}})\|_*.\end{aligned} \tag{4.8}$$

From

$$\mathrm{bcirc}(\boldsymbol{\mathcal{Y}}^{\vec{\mathbf{O}}}) = \begin{pmatrix} [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(1)} & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(I_3)} & \cdots & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(2)} \\ [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(2)} & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(1)} & \cdots & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(3)} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(I_3)} & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(I_3-1)} & \cdots & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(1)} \end{pmatrix}$$
$$\longrightarrow \begin{pmatrix} [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(1)} & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(2)} & \cdots & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(I_3)} \\ [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(2)} & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(3)} & \cdots & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(1)} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(I_3)} & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(1)} & \cdots & [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(I_3-1)} \end{pmatrix},$$

it can be seen that $\mathrm{bcirc}(\boldsymbol{\mathcal{Y}}^{\vec{\mathbf{O}}})$ will be approximated to a lower rank matrix and get a better low-rank eastimation of $\boldsymbol{\mathcal{Y}}$ when adjacent $[\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(i)}$ and $[\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(i+1)}$ are more similar (mark $[\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(I_3+1)} = [\boldsymbol{\mathcal{Y}}]_{:,:,\vec{\mathbf{O}}(1)}$ for convenience). Therefore, we convert (4.7) to the following problem:

$$\arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}}^{\vec{\mathbf{O}}^*} - \boldsymbol{\mathcal{X}}\|_F^2 + \tau\|\boldsymbol{\mathcal{X}}\|_*, \tag{4.9}$$

where $\vec{\mathrm{O}}^*$ is obtained by $\mathbf{C}^*(\boldsymbol{\mathcal{Y}})$. Therefore we solve (4.7) via Algorithm 4.1 approximately by Theorem 2.1[1]. The symbols and definitions used in Algorithm 4.1 are explained in Definitions 4.4-4.5.

Definition 4.4. *Let* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\mathbf{C} = \{i_1, i_2, ..., i_{I_3}, i_1\}$ *is a circle on* $\boldsymbol{\mathcal{A}}$ *which composed of* 1, 2, 3,..., I_3. *We call* $\mathbf{C}(i_s, i_t) = \{i_s, i_{s+1}, ..., i_t\}$ *as a walk from* i_s *to* i_t *on* $\mathbf{C}$, *and* $\mathbf{C}^{-1}(i_s, i_t) = \{i_t, i_{t-1}, ..., i_s\}$ *as inverse of walk* $\mathbf{C}(i_s, i_t)$. *Assume* $\mathbf{C}(i_1, i_l) = \{i_1, i_2, ..., i_l\}$ *and* $\mathbf{C}(i_l, i_{l+k}) = \{i_l, i_{l+1}, ..., i_{l+k}\}$ *are two walks on circle* $\mathbf{C}$, *mark* $\mathbf{C}(i_1, i_l) \bigcup \mathbf{C}(i_l, i_{l+k}) = \{i_1, i_2, ..., i_l, i_{l+1}, ..., i_{l+k}\}$.

Definition 4.5. *Let* $\mathbf{C} = \{i_1, i_2, ..., i_{I_3}, i_1\}$ *is a circle on* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ *which composed of* 1, 2, 3,..., I_3., *and* $\boldsymbol{W}(\boldsymbol{\mathcal{A}})$ *is a weight matrix in which* $w_{i,j}(\boldsymbol{\mathcal{A}}) = \| [\boldsymbol{\mathcal{A}}]_{:,:,i} - [\boldsymbol{\mathcal{A}}]_{:,:,j} \|_F$ *is weight of* $[\boldsymbol{\mathcal{A}}]_{:,:,i}$ *and* $[\boldsymbol{\mathcal{A}}]_{:,:,j}$ *for* $i \neq j$, *and* $w_{i,j}(\boldsymbol{\mathcal{A}}) = \infty$ *for* $i = j$. *Mark* $w(\boldsymbol{\mathcal{A}}, \mathbf{C}) = \sum_{k=1}^{I_3-1} w_{i_k, i_{k+1}}(\boldsymbol{\mathcal{A}}) + w_{i_{I_3}, i_1}(\boldsymbol{\mathcal{A}})$, $\mathbf{C}^*(\boldsymbol{\mathcal{A}}) = \arg\min_{\mathbf{C}} w(\boldsymbol{\mathcal{A}}, \mathbf{C})$ *and* $c^*(\boldsymbol{\mathcal{A}}) = \min_{\mathbf{C}} w(\boldsymbol{\mathcal{A}}, \mathbf{C})$.

A key point to $\mathcal{D}^{\mathrm{SPV}}(\boldsymbol{\mathcal{Y}}, \tau)$ is to find $\mathbf{C}^*(\boldsymbol{\mathcal{Y}})$. And a simplest idea for getting $\mathbf{C}^*(\boldsymbol{\mathcal{Y}})$ is that, when we get $\mathbf{C}^{(k-1)}$, we can make appropriate modifications for the circle $\mathbf{C}^{(k)}$ to get another circle $\mathbf{C}^{(k)}$ with a smaller $w(\boldsymbol{\mathcal{Y}}, \mathbf{C}^{(k)})$ as Fig. 4.2 [4]. Repeat the above process until $\mathbf{C}^{(k)}$ convergence to $\mathbf{C}^*(\boldsymbol{\mathcal{Y}})$.

4.3.2 Tensor Robust Principal Component Analysis for SPV

Consider the following problem:

$$(\hat{\boldsymbol{\mathcal{L}}}, \hat{\boldsymbol{\mathcal{S}}}, \vec{\mathrm{O}}^*) = \min_{\boldsymbol{\mathcal{L}}, \boldsymbol{\mathcal{S}}, \vec{\mathrm{O}}} \|\boldsymbol{\mathcal{L}}\|_* + \lambda \|\boldsymbol{\mathcal{S}}\|_1 \qquad s.t. (\boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{S}})^{\vec{\mathrm{O}}} = \boldsymbol{\mathcal{L}}, \tag{4.10}$$

[1] It is worth noting that we convert (4.7) to a minimum Hamiltonian circle problem.

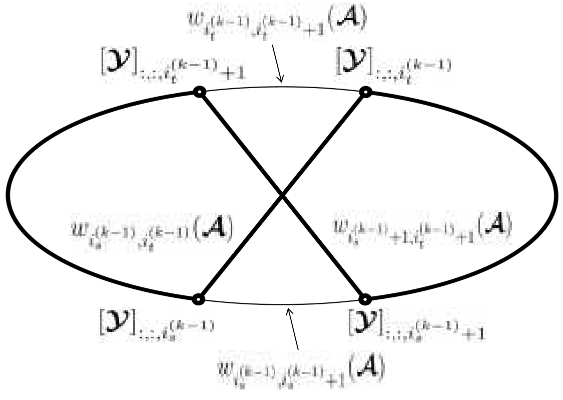

$w_{i_t^{(k-1)}, i_t^{(k-1)}+1}(\mathcal{A})$
$[\mathcal{Y}]_{:,:,i_t^{(k-1)}+1}$
$[\mathcal{Y}]_{:,:,i_t^{(k-1)}}$
$w_{i_a^{(k-1)}, i_t^{(k-1)}}(\mathcal{A})$
$w_{i_a^{(k-1)}+1, i_t^{(k-1)}+1}(\mathcal{A})$
$[\mathcal{Y}]_{:,:,i_a^{(k-1)}}$
$[\mathcal{Y}]_{:,:,i_a^{(k-1)}+1}$
$w_{i_a^{(k-1)}, i_a^{(k-1)}+1}(\mathcal{A})$

Figure 4.2: Obtaining $\mathbf{C}^*(\boldsymbol{\mathcal{Y}})$ by solving a minimum Hamiltonian circle problem.

where $\boldsymbol{\mathcal{L}}$ is low-rank, and $\boldsymbol{\mathcal{S}}$ is sparse. And Algorithm 4.2 based on alternating direction method (ADM) [3] is proposed for solving (4.10). It is worth noting that, for fixed $\vec{\mathrm{O}}$, (4.10) degenerate to TRPCA (which means (4.10) can exactly recover the low-rank and sparse components from their sum for the fixed $\vec{\mathrm{O}}$.).

4.4 Experiments

This section includes three parts: in the first two parts, we compare the proposed algorithm (TRPCA-SPV) with several existing state-of-the-art tensor recovery methods (including RPCA[2][8], SNN[3][25], Liu's work [3](called Liu for short)[9] and TRPCA[3] [50]) on image sequence recovery task and image classification task to evaluate the effectiveness of the algorithms to alleviate SPV problem on tensor recovery. And the third part is conducted in order to evaluate the performance of TRPCA-SPV with different values of the parameter κ.

[2] https://github.com/dlaptev/RobustPCA

[3] https://github.com/canyilu/LibADMM-toolbox

4.4.1 Image Sequence Recovery

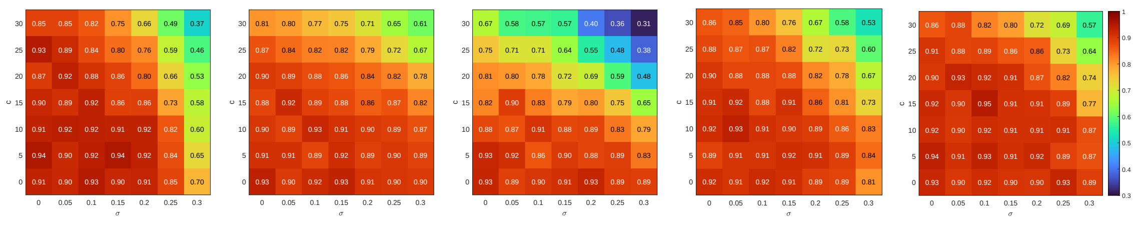
In this part, all five methods are tested on two hyperspectral image databases including Pavia University [4] and Botswana[4].

Each image with a dimension of $I_1 \times I_2$ is contaminated by the mixture of zero mean Gaussian noise and random valued impulse noise, in which standard deviations of zero-mean Gaussian δ is set as $\delta = 5 : 10 : 25$ and random-valued impulse noise with density level c is set as $c = 0.05 : 0.1 : 0.25$. For Pavia University, we empirically set $\lambda = \frac{1}{\sqrt{\max(I_1, I_2)}}$ for RPCA (which deals with each band separately), $\lambda = [\frac{240}{3}, \frac{240}{3}, \frac{240}{3}]$ for SNN, and $\lambda = 330 \times [0.2, 0.1, 0.7]$ for Liu. For Botswana, we empirically set $\lambda = \frac{0.9}{\sqrt{\max(I_1, I_2)}}$ for RPCA (which deals with each band separately), $\lambda = [\frac{340}{3}, \frac{340}{3}, \frac{340}{3}]$ for SNN, and $\lambda = 370 \times [0.3, 0.1, 0.6]$ for Liu.

For TRPCA, the parameter λ is tuned to $\lambda = \frac{0.9}{\sqrt{\max(I_1, I_2)I_3}}$ and $\lambda = \frac{0.8}{\sqrt{\max(I_1, I_2)I_3}}$ for Pavia University and Botswana respectively, in which I_3 is the number of spectral bands. For TRPCA-SPV, the parameter λ is tuned to $\lambda = \frac{0.9}{\sqrt{\max(I_1, I_2)I_3}}$ for the two databases.

The Mean Peak Signal-To-Noise Ratio (MPSNR) value $\frac{1}{I_3}\sum_{i=1}^{I_3} \mathrm{PSNR}_i$ is used to evaluate the methods, where PSNR_i is the Peak Signal-To-Noise Ratio (PSNR) result of i-th restored band. From Table 4.1, there are some observations as follows: TRPCA-SPV outperforms the compared methods by a wide margin in most of cases. Specifically, for Pavia University, TRPCA-SPV outperforms other methods by more than 3 dB on the case of small noise levels. This demonstrates the superiority of our TRPCA-SPV in tensor recovery. For the case of TRPCA-SPV v.s. TRPCA, TRPCA-SPV can attain much better results compared

[4] http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensin_Scenes



(a) (b) (c) (d) (e)

Figure 4.3: Classification accuracies of the 5 algorithms on ORL database: (a) RPCA (b) SNN (c) Liu (d) TRPCA (e) TRPCA-SPV

to TRPCA. The gap between MPSNR results by TRPCA-SPV and TRPCA even achieves 5dB in the case of $\delta = 5$ and $c = 0.05 : 0.1 : 0.25$. This illustrates the huge affecting of SPV on TRPCA, and TRPCA-SPV can eliminate it well.

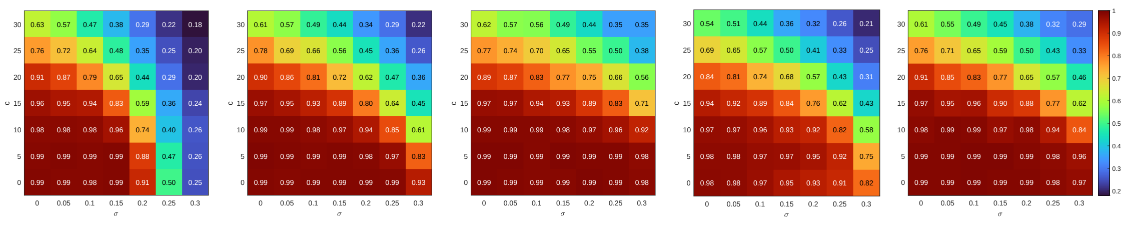
4.4.2 Image Classification

In this part, image classification is conducted on two datasets including ORL database[5] and CMU PIE database [6].

Each image with the size of $I_1 \times I_2$ is contaminated by the mixed noise, in which δ is set as $\delta = 0 : 5 : 30$ and c is set as $c = 0 : 0.05 : 0.3$. For each noise level, all five algorithms are used to recover the low-rank tensor structure from the noised images. The performance of the algorithms is evaluated by classification accuracy via k nearest neighbor (kNN), where $k = 1$ in the experiments. For each dataset, 90% of samples are randomly selected as the training set, and the rest are taken as the testing set. The experiments are repeated 10 times, and the average values of the accuracy of all methods are reported in Fig. 4.3-4.4. For RPCA

[5] https://cam-orl.co.uk/facedatabase.html

[6] https://www.ri.cmu.edu/project/pie-database/



(a) (b) (c) (d) (e)

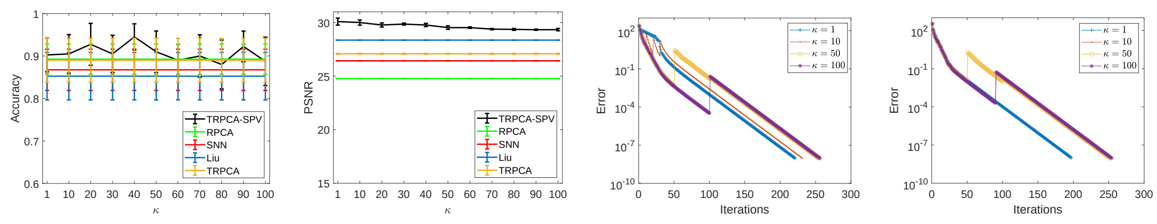
Figure 4.4: Classification accuracy result on CMU PIE database: (a) RPCA (b) SNN (c) Liu (d) TRPCA (e) TRPCA-SPV

and TRPCA, the parameter λ is set to $\lambda = 1/\sqrt{\max(I_1 I_2, I_3)}$ and $\lambda = 1/\sqrt{\max(I_1, I_2)I_3}$ respectively as suggested in [50], in which I_3 is the number of samples. For TRPCA-SPV, the parameter λ is set to $\lambda = 1/\sqrt{\max(I_1, I_2)I_3}$ as well. For Liu, we find that it does not perform well when λ_i's are set to the values suggested in theory [34]. We empirically set it as $70 \times [0.2, 0.3, 0.5]$. For SNN, we empirically set $\lambda = [\frac{70}{3}, \frac{70}{3}, \frac{70}{3}]$. All results are presented in Fig. 4.3-4.4. The cell with more dark red corresponds to higher classification accuracy.

From Fig. 4.3-4.4, there are some observations as follows: In general, TRPCA-SPV achieves more stable and better performance compared to other methods (RPCA, SNN, Liu, and TRPCA). In addition, TRPCA-SPV can attain better results compared to TRPCA, because TRPCA-SPV exploits the low-rank structure within the tensor data more exactly.

4.4.3 Sensitivity Analysis of Parameters

In this part, an experiment is conducted with two datasets (including ORL database and Pavia University), in which each image in datasets contaminated by the mixed noise with $\delta = 15$ and $c = 0.15$, to investigate the influence of the parameter κ.



(a) (b) (c) (d)

Figure 4.5: Sensitivity analysis of parameter κ for TRPCA-SPV on (a) ORL database and (b) Pavia University; Convergence analysis for TRPCA-SPV with different κ on (c) ORL database and (d) Pavia University.

The experiments for each parameter κ are repeated 10 times, and the results obtained by the different methods are shown in Fig. 4.5 (a) and (b), from which we have the following observations: (1) In general, the results by TRPCA-SPV are robust against to the parameter κ. (2) For all cases of TRPCA-SPV, the results by TRPCA-SPV are much better than TRPCA.

In addition, from Fig. 4.5 (c) and (d), the curve by TRPCA-SPV is shocked depending on the parameter κ at the beginning and tends to stable with more iterations of the algorithm, in which

$$\text{Error} = \max(\|\boldsymbol{\mathcal{L}}^{(k+1)} - \boldsymbol{\mathcal{L}}^{(k)}\|_\infty, \|\boldsymbol{\mathcal{S}}^{(k+1)} - \boldsymbol{\mathcal{S}}^{(k)}\|_\infty, \|\boldsymbol{\mathcal{L}}^{(k+1)} + (\boldsymbol{\mathcal{S}}^{(k+1)})^{\vec{\mathbf{O}}^*} - \boldsymbol{\mathcal{P}}^{\vec{\mathbf{O}}^*}\|_\infty). \tag{4.11}$$

4.5 Summary

This chapter focuses on solving a new problem (SPV in tensor recovery) that has not been explored so far. We aim to accurately recover a low-rank tensor from a high-dimensional tensor data with chaos tensor slices sequence. The example given in Figure 1 shows a huge gap between results by tensor recovery using tensors with different slices sequence. To deal with this issue, TRSPV is proposed. Furthermore, we discuss the SPV of several key tensor recovery problems theoretically. To this end, we first study the row (or column) permutation invariance of a key low-rank matrix recovery problem (Principal Component Analysis). Then, the SPI of several key tensor recovery problems are discussed theoretically, and we get the following results: (1) Tensor recovery based on the weighted sum of the nuclear norm of the unfolding matrices has SPI. (2) For $I_3 \leq 3$, DFT-based tensor recovery has SPI. For the case of $I_3 > 3$, experimental results show the effectiveness of the proposed algorithm and eliminate SPV in DFT-based tensor recovery well.

Algorithm 4.1: Tensor recovery for SPV (TRSPV)

```
Input: $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, and $\mathrm{Iternum}$.
Output: $\mathbf{C}^*(\boldsymbol{\mathcal{Y}})$ and $\mathcal{D}^{\mathrm{SPV}}(\boldsymbol{\mathcal{Y}}, \tau)$
Compute weight matrix $W$;
Initialize circle $\mathbf{C}^{(0)} = \{i_1^{(0)}, i_2^{(0)}, ..., i_{I_3}^{(0)}, i_1^{(0)}\}$, and $k = 0$;
while $k \leq \mathrm{Iternum}$ do
    $k = k + 1$;
    if there are different $i_s^{(k-1)}$,$i_t^{(k-1)}$,$i_s^{(k-1)} + 1$,$i_t^{(k-1)} + 1$ in $\mathbf{C}^{(k-1)}$ which make
     $w_{i_s^{(k-1)},i_t^{(k-1)}}(\boldsymbol{\mathcal{Y}}) + w_{i_s^{(k-1)}+1,i_t^{(k-1)}+1}(\boldsymbol{\mathcal{Y}}) <$
     $w_{i_s^{(k-1)},i_s^{(k-1)}+1}(\boldsymbol{\mathcal{Y}}) + w_{i_t^{(k-1)},i_t^{(k-1)}+1}(\boldsymbol{\mathcal{Y}})$ then
        $\mathbf{C}^{(k)} = \{i_t^{(k-1)}, i_s^{(k-1)}\} \bigcup$
        $\mathbf{C}^{(k-1)^{-1}}(i_{t+1}^{(k-1)}, i_s^{(k-1)}) \bigcup \{i_{t+1}^{(k-1)}, i_{s+1}^{(k-1)}\}$
        $\bigcup \mathbf{C}^{(k-1)}(i_{s+1}^{(k-1)}, i_t^{(k-1)})$;
    else
        $\mathbf{C}^{(k)} = \mathbf{C}^{(k-1)}$;
        break;
    end
end
Obtain $\mathbf{C}^*(\boldsymbol{\mathcal{Y}}) = \mathbf{C}^{(k)}$, and compute $\mathcal{D}^{\mathrm{SPV}}(\boldsymbol{\mathcal{Y}}, \tau) = \mathcal{D}(\boldsymbol{\mathcal{Y}}^{\vec{\mathbf{O}}^*}, \tau)$, where $\vec{\mathbf{O}}^*$ obtained
 by $\mathbf{C}^*(\boldsymbol{\mathcal{Y}})$;
```

Algorithm 4.2: TRPCA for SPV (TRPCA-SPV)

```
Initialize: $\boldsymbol{\mathcal{L}}^{(0)} = \boldsymbol{\mathcal{S}}^{(0)} = \boldsymbol{\mathcal{Q}}^{(0)} = \boldsymbol{\mathcal{Y}}^{(0)} = \mathbf{0}$, $\rho > 1$, $\mu_0 = 1e-3$, $\epsilon = 1e-8$, $\kappa > 0$.
while not converged do
    1. Update $\vec{\mathrm{O}}^*$ by
    If $\kappa = 1$ or $k \mod \kappa = 1$, update $\vec{\mathrm{O}}^*$ by $\mathrm{C}^*(\boldsymbol{\mathcal{M}}^{(k)})$, where
      $\boldsymbol{\mathcal{M}}^{(k)} = \boldsymbol{\mathcal{P}} - \boldsymbol{\mathcal{S}}^{(k)} - \frac{\boldsymbol{\mathcal{Q}}^{(k)}}{\mu_k}$;
    2.Update $\boldsymbol{\mathcal{L}}^{(k+1)}$ by $\boldsymbol{\mathcal{L}}^{(k+1)} = \arg\min_{\boldsymbol{\mathcal{L}}} \|\boldsymbol{\mathcal{L}}\|_* + \frac{\mu_k}{2}\|\boldsymbol{\mathcal{L}} - (\boldsymbol{\mathcal{M}}^{(k)})^{\vec{\mathrm{O}}^*}\|_F^2$;
    3. Update $\boldsymbol{\mathcal{S}}^{(k+1)}$ by
      $\boldsymbol{\mathcal{S}}^{(k+1)} = \arg\min_{\boldsymbol{\mathcal{S}}} \lambda\|\boldsymbol{\mathcal{S}}^{\vec{\mathrm{O}}^*}\|_1 + \frac{\mu_k}{2}\|\boldsymbol{\mathcal{L}}^{(k+1)} + \boldsymbol{\mathcal{S}}^{\vec{\mathrm{O}}^*} - \boldsymbol{\mathcal{P}}^{\vec{\mathrm{O}}^*} + (\frac{\boldsymbol{\mathcal{Q}}^{(k)}}{\mu_k})^{\vec{\mathrm{O}}^*}\|_F^2$;
    4.$(\boldsymbol{\mathcal{Q}}^{(k+1)})^{\vec{\mathrm{O}}^*}$=$(\boldsymbol{\mathcal{Q}}^{(k)})^{\vec{\mathrm{O}}^*} + \mu(\boldsymbol{\mathcal{L}}^{(k+1)} + (\boldsymbol{\mathcal{S}}^{(k+1)})^{\vec{\mathrm{O}}^*} - \boldsymbol{\mathcal{P}}^{\vec{\mathrm{O}}^*})$;
    5.Update $\mu_{k+1}$ by $\mu_{k+1} = \min(\rho\mu_k, \mu_{\max})$;
    6. Check the convergence conditions
    $\|\boldsymbol{\mathcal{L}}^{(k+1)} - \boldsymbol{\mathcal{L}}^{(k)}\|_\infty \leq \epsilon$, $\|(\boldsymbol{\mathcal{S}}^{(k+1)})^{\vec{\mathrm{O}}^*} - (\boldsymbol{\mathcal{S}}^{(k)})^{\vec{\mathrm{O}}^*}\|_\infty \leq \epsilon$,
      $\|\boldsymbol{\mathcal{L}}^{(k+1)} + (\boldsymbol{\mathcal{S}}^{(k+1)})^{\vec{\mathrm{O}}^*} - \boldsymbol{\mathcal{P}}^{\vec{\mathrm{O}}^*}\|_\infty \leq \epsilon$;
end
```

		Botswana					Pavia University				
δ	c	RPCA	SNN	Liu	TRPCA	TRPCA-SPV	RPCA	SNN	Liu	TRPCA	TRPCA-SPV
5	5%	29.90	34.52	36.82	32.06	**38.44**	27.56	29.82	32.03	30.65	**36.60**
	15%	29.04	33.02	35.34	30.06	**37.11**	26.90	29.21	31.60	28.07	**35.39**
	25%	27.73	30.81	32.92	28.78	**34.98**	25.55	27.96	30.53	26.03	**33.48**
15	5%	28.11	30.91	32.42	31.11	**34.21**	25.58	27.19	28.07	30.22	**31.51**
	15%	27.32	29.47	30.92	28.99	**32.34**	24.77	26.43	28.35	27.17	**30.38**
	25%	25.78	27.23	28.48	27.18	**29.67**	23.20	24.96	26.99	24.67	**27.76**
25	5%	26.84	29.17	30.37	29.34	**31.65**	23.63	25.12	26.94	28.50	**29.02**
	15%	26.05	27.55	28.67	26.83	**29.77**	22.74	24.30	26.30	25.21	**27.49**
	25%	24.29	25.14	26.06	24.18	**26.79**	21.11	22.76	24.77	22.49	**24.81**

Table 4.1: MPSNR results by different methods on Botswana and Pavia University.

Chapter 5

Handling Slice Permutations Variability in t-Product-Based Tensor Recovery

5.1 Introduction

In the previous chapter, we discussed that Discrete Fourier Transform (DFT)-based tensor recovery exhibits frontal Slice Permutations Variability (SPV), meaning that rearranging the order of frontal slices in a tensor significantly affects the effectiveness of tensor recovery. We believe that the root cause of SPV in tensor tubal rank-based recovery methods lies in the definition of the t-product. To investigate the SPV in t-product-based tensor recovery methods, including Tensor Principal Component Analysis (TPCA) and Tensor Factorization (TF), for best κ-tensor rank estimating, we conducted experiments using seven gray videos[1]: *bridge-far*, *grandma*, *akiyo*, *bridge-close*, *templet*, *bus*, and *mobile*. For each video, we

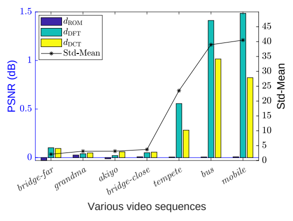
[1] http://trace.eas.asu.edu/yuv/

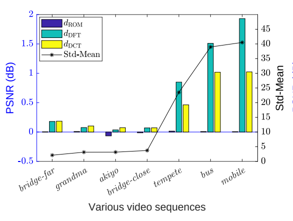
used only the first 50 frames. These videos were chosen to test the influence of frontal slice permutations on a series of t-product-based methods, including DFT-based methods, Discrete Cosine Transform (DCT)-based methods, and Random Orthogonal Matrix (ROM)-based methods. In Figure 5.1, we illustrate the relationship between SPV and the transforms used, as well as the variation in frontal slices of the data tensor. More specifically, we have the following two observations. (1) The impact of SPV on ROM-based methods, in terms of mean Peak Signal-to-Noise Ratio (PSNR) results, is relatively mild compared to DFT- and DCT-based methods. This suggests that the severity of SPV in t-product-based methods depends on the specific transform employed. For the video sequence *mobile*, the SPV-induced gap in PSNR reaches up to 1.9 dB. This is because both DFT and DCT represent the frequency domain. Each frontal slice obtained through DFT or DCT applied to the tensor tubes corresponds to coefficients at different frequencies. As a result, SPV becomes more pronounced in DFT-based and DCT-based methods when dealing with video sequences exhibiting higher std-mean values (the mean of standard deviations of each tube in the data tensor).

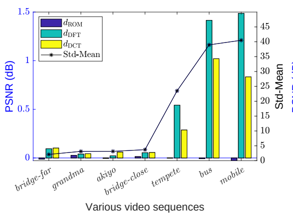
In the previous chapter, we explored a method to find an optimal tensor frontal slice permutation that results in a tensor with similar adjacent frontal slices, leading to a lower average rank tensor. However, this method has certain limitations:

(1) Solving a Minimum Hamiltonian Cycle problem to find the optimal cycle is NP-hard, making it challenging to obtain the optimal solution. Additionally, even slight disturbances in the slice sequence order can lead to worse tensor recovery results, as observed in Figure 5.2(b).

(2) The example presented in Figure 5.2(a) demonstrates that a tensor with a smaller


(a) ($\kappa = 30$, TPCA)


(b) ($\kappa = 50$, TPCA)


(c) ($\kappa = 30$, TF)

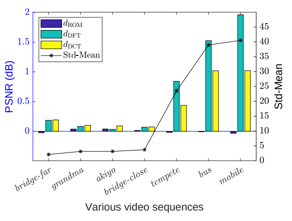
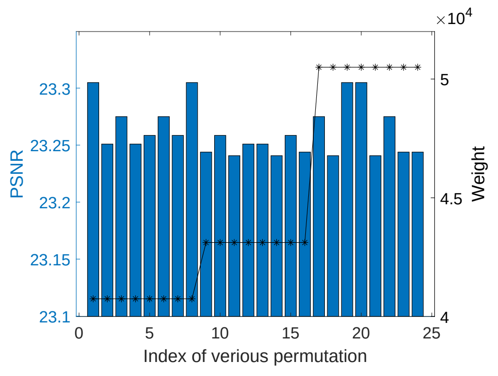

(d) ($\kappa = 50$, TF)

Figure 5.1: The differences in mean PSNR (Peak Signal to Noise Ratio) results, *i.e.*, d_{ROM}, d_{DFT}, and d_{DCT}, by performing random frontal slice permutation on the data tensor for DFT-based methods, DCT-based methods, and ROM-based methods are presented as the bars. The std-mean results presented as the black line are the mean of standard deviations of each tube in the data tensor.


(a) DCT

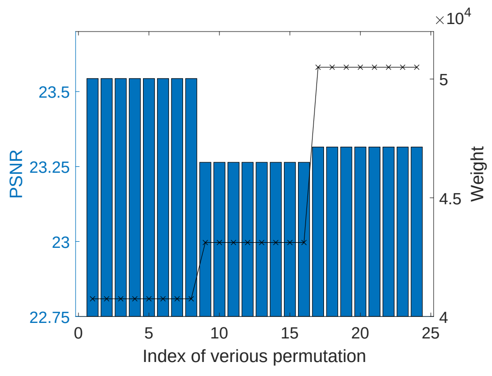

(b) DFT

Figure 5.2: Left Y Axis: the PSNR results of the best $\kappa = 50$-rank approximation by different SVD-based TPCA when the observation tensor suffers various permutations. Right Y Axis: the weight of the cycle that corresponds to the Euclidean distance of adjacent frontal slices.

weight does not necessarily have a lower DCT-based rank. This finding emphasizes that the method proposed in [100] is not suitable for DCT-based methods, as indicated by the results obtained.

As a result, there is a need to develop an effective general solver that can effectively handle SPV in t-product-based tensor recovery across various transform methods.

In this chapter, we introduce a generalized framework to address the issue of Slice Permutations Variability (SPV) in t-product-based tensor recovery methods. The framework revolves around finding a unitary matrix $\boldsymbol{U}$ that enables the tensor $\boldsymbol{\mathcal{A}} \times_3 \boldsymbol{U}$ to be approximated by a tensor with a lower rank. By tackling SPV and presenting this generalized solution, our work aims to enhance the robustness and effectiveness of t-product-based tensor recovery methods across various transform methods.

5.2 Proposed Framework for Handling SPV in Tensor Recovery

5.2.1 Formulation of Proposed Framework

Let us consider the following constrained problems:

$$\min_{\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{E}}} \mathcal{F}(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}) \qquad s.t.\ \mathcal{G}_k(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}) \leq \mathbf{0}(k = 1, 2, \cdots, n), \tag{5.2}$$

where $\mathcal{G}_k(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}) \leq \mathbf{0}$ represents that each element of $\mathcal{G}_k(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}})$ is less or equal than 0, and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, and $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ represent the observation tensor, low rank estimation of $\boldsymbol{\mathcal{Y}}$, and noise in the observation tensor, respectively.

Algorithm 5.1: BCD for solving the proposed t-SVD-based model (5.7).

```
**Input:** $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $k, \varepsilon > 0$.
**Output:** $\boldsymbol{\mathcal{X}}$.
while not converged do
1. Compute $\boldsymbol{\mathcal{X}}^{(t+1)}$ by $\boldsymbol{\mathcal{X}}^{(t+1)} = \arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{U}\boldsymbol{\mathcal{Y}}_{(3)} - \boldsymbol{\mathcal{X}}_{(3)}\|_F^2 + \lambda\|\boldsymbol{\mathcal{X}}\|_{\diamond,\boldsymbol{L}}$;
2. Compute $\boldsymbol{U}^{(t+1)}$ by $\boldsymbol{U}^{(t+1)} = \arg\min_{\boldsymbol{U}} \frac{1}{2}\|\boldsymbol{U}\boldsymbol{\mathcal{Y}}_{(3)} - \boldsymbol{\mathcal{X}}_{(3)}\|_F^2 s.t. \boldsymbol{I} = \boldsymbol{U}^T\boldsymbol{U}$;
3. Check the convergence condition: $\|\boldsymbol{\mathcal{X}}^{(t+1)} - \boldsymbol{\mathcal{X}}^{(t)}\|_\infty < \varepsilon$,
 $\|\boldsymbol{U}^{(t+1)} - \boldsymbol{P}^{(t)}\|_\infty < \varepsilon$;
4. $t = t + 1$.
end while
```

The most straightforward approach to finding a good frontal slice order for $\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}$ is by seeking a permutation matrix $\boldsymbol{P}$ such that $(\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}) \times_3 \boldsymbol{P}$ can be approximated by a lower-rank tensor. To achieve this, we aim to minimize $\mathcal{F}(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}, (\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}) \times_3 \boldsymbol{P})$ for $\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}$, while satisfying $\mathcal{G}_k(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}, (\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}) \times_3 \boldsymbol{P}) \leq \boldsymbol{0}(k = 1, 2, \cdots, n)$. Thus, we consider the following problem:

$$\min_{\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{E}},\boldsymbol{P}} \mathcal{F}(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}, (\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}) \times_3 \boldsymbol{P}) \qquad s.t.\ \mathcal{G}_k(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}, (\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}) \times_3 \boldsymbol{P}) \leq \boldsymbol{0}(k = 1, 2, \cdots, n). \tag{5.3}$$

Here, $\boldsymbol{P}$ is a permutation matrix satisfying $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}$. To simplify the problem, we do not strictly require that $\boldsymbol{P}$ is a permutation matrix, but it should still satisfy $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}$. Therefore, we introduce a unitary matrix $\boldsymbol{U}$, and consider the following problem

Algorithm 5.2: BCD for solving the proposed Tensor Factorization model (5.10).

```
**Input:** $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\kappa, \varepsilon > 0$.
**Output:** $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$.
**while** not converged **do**
1. Compute $\boldsymbol{\mathcal{A}}^{(t+1)}$ by $\boldsymbol{\mathcal{A}}^{(t+1)} = \arg\min_{\boldsymbol{\mathcal{A}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} \times_3 \boldsymbol{U}^{(t)} - \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}^{(t)}\|_F^2$;
2. Compute $\boldsymbol{\mathcal{B}}^{(t+1)}$ by $\boldsymbol{\mathcal{B}}^{(t+1)} = \arg\min_{\boldsymbol{\mathcal{B}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} \times_3 \boldsymbol{U}^{(t)} - \boldsymbol{\mathcal{A}}^{(t+1)} * \boldsymbol{\mathcal{B}}\|_F^2$;
3. Compute $\boldsymbol{U}^{(t+1)}$ by
   $$\boldsymbol{U}^{(t+1)} = \arg\min_{\boldsymbol{U}} \frac{1}{2}\|\boldsymbol{U}\boldsymbol{\mathcal{Y}}_{(3)} - [\boldsymbol{\mathcal{A}}^{(t+1)} * \boldsymbol{\mathcal{B}}^{(t+1)}]_{(3)}\|_F^2 \qquad s.t.\ \boldsymbol{I} = \boldsymbol{U}^T\boldsymbol{U}; \quad (5.1)$$
4. Check the convergence condition: $\|\boldsymbol{\mathcal{A}}^{(t+1)} - \boldsymbol{\mathcal{A}}^{(t)}\|_\infty < \varepsilon$, $\|\boldsymbol{\mathcal{B}}^{(t+1)} - \boldsymbol{\mathcal{B}}^{(t)}\|_\infty < \varepsilon$,
   $\|\boldsymbol{U}^{(t+1)} - \boldsymbol{U}^{(t)}\|_\infty < \varepsilon$;
5. $t = t + 1$.
**end while**
```

instead:

$$\begin{aligned}&\min_{\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{E}},\boldsymbol{U}} \mathcal{F}(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}, (\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}) \times_3 \boldsymbol{U}) \\ &s.t.\ \mathcal{G}_k(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{X}}, (\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{E}}) \times_3 \boldsymbol{U}) \leq \mathbf{0} (k = 1, 2, \cdots, n),\ \boldsymbol{I} = \boldsymbol{U}^T\boldsymbol{U}. \end{aligned} \tag{5.4}$$

By solving (5.4), we obtain the low-rank estimation of $\boldsymbol{\mathcal{Y}}$, *i.e.*, $\hat{\boldsymbol{\mathcal{X}}} \times_3 \hat{\boldsymbol{U}}^T$, where $(\hat{\boldsymbol{\mathcal{X}}}, \hat{\boldsymbol{\mathcal{E}}}, \hat{\boldsymbol{U}})$ represents the optimal solution of (5.4).

Therefore, using the framework presented in (5.4), we can address the SPV in Tensor Robust Principal Component Analysis (TRPCA) as follows:

$$\min_{\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{E}},\boldsymbol{U}} \|\boldsymbol{\mathcal{E}}\|_1 + \tau\|\boldsymbol{\mathcal{X}}\|_{*,\boldsymbol{L}} \qquad s.t.\ \boldsymbol{\mathcal{Y}} \times_3 \boldsymbol{U} = \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{E}} \times_3 \boldsymbol{U},\ \boldsymbol{I} = \boldsymbol{U}^T\boldsymbol{U}, \tag{5.5}$$

where $\|\boldsymbol{\mathcal{X}}\|_{*,\boldsymbol{L}}$ denotes a regularization term calculated as $\sum_{i=1}^{I_3} \frac{1}{I_3}\|[\boldsymbol{L}(\boldsymbol{\mathcal{X}})]_{:,:,i}\|_*$, and $\boldsymbol{L}$ can be any invertible transforms, either real or complex.

5.2.2 The Proposed Framework for Learning t-Product-Based Rank and t-Product

Here, we use TPCA as an example to illustrate our framework for learning the t-product and t-product-based rank. The TPCA model can be formulated as follows:

$$\arg\min_{\boldsymbol{\mathcal{X}}} \|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 \qquad s.t.\ \|\boldsymbol{\mathcal{X}}\|_{\diamond,\boldsymbol{L}} \leq \kappa. \tag{5.6}$$

In this formulation, $\|\boldsymbol{\mathcal{X}}\|_{\diamond,\boldsymbol{L}}$ represents the tensor rank of $\boldsymbol{\mathcal{X}}$, which is defined as the non-zero tubes of $\boldsymbol{L}(\boldsymbol{\mathcal{S}})$ obtained by performing frontal slices-wise Singular Value Decomposition (SVD) on $\boldsymbol{L}(\boldsymbol{\mathcal{X}})$. This can be expressed as $[\boldsymbol{L}(\boldsymbol{\mathcal{X}})]_{:,:,i} = [\boldsymbol{L}(\boldsymbol{\mathcal{U}})]_{:,:,i}[\boldsymbol{L}(\boldsymbol{\mathcal{S}})]_{:,:,i}[\boldsymbol{L}(\boldsymbol{\mathcal{V}})]_{:,:,i}^T$ for $i = 1, 2, \cdots, I_3$. Using the framework presented in (5.4), we obtain the following SVD-based TPCA model:

$$(\hat{\boldsymbol{\mathcal{X}}}, \hat{\boldsymbol{U}}) = \arg\min_{\boldsymbol{\mathcal{X}},\boldsymbol{U}} \|\boldsymbol{\mathcal{Y}} \times_3 \boldsymbol{U} - \boldsymbol{\mathcal{X}}\|_F^2 \qquad s.t.\ \|\boldsymbol{\mathcal{X}}\|_{\diamond,\boldsymbol{L}} \leq \kappa,\ \boldsymbol{I} = \boldsymbol{U}^T\boldsymbol{U}. \tag{5.7}$$

To solve (5.7), a Block Coordinate Descent (BCD)-based optimization algorithm (Algorithm 5.1) can be used, which involves performing SVD. Let $\boldsymbol{\mathcal{Z}}$ be $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} \times_3 \boldsymbol{U}^T$, we can reformulate the problem as follows:

$$(\hat{\boldsymbol{\mathcal{Z}}}, \hat{\boldsymbol{U}}) = \arg\min_{\boldsymbol{\mathcal{Z}},\boldsymbol{U}} \|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{Z}}\|_F^2 \qquad s.t.\ \|\boldsymbol{\mathcal{Z}}\|_{\diamond,\boldsymbol{LU}} \leq \kappa,\ \boldsymbol{I} = \boldsymbol{U}^T\boldsymbol{U}. \tag{5.8}$$

Furthermore, we can observe that $\hat{\boldsymbol{\mathcal{X}}} \times_3 \hat{\boldsymbol{U}}^T = \hat{\boldsymbol{\mathcal{Z}}}$. Thus, solving (5.7) is equivalent to learning an appropriate transform $\boldsymbol{LU}$ and tensor norm $\|\cdot\|_{\diamond,\boldsymbol{LU}}$ for better exploiting the low-rank property in the observation tensor.

The definition of the t-product is closely related to the transform, leading to a similar conclusion for the t-product. Let us consider the tensor factorization-based TPCA (TFTPCA):

$$\min_{\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{A}} *_{\boldsymbol{L}} \boldsymbol{\mathcal{B}}\|_F^2, \tag{5.9}$$

where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times\kappa\times I_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{\kappa\times I_2\times I_3}$ for given κ. Using the proposed framework, we can formulate the following model:

$$(\hat{\boldsymbol{\mathcal{A}}}, \hat{\boldsymbol{\mathcal{B}}}, \hat{\boldsymbol{U}}) = \arg\min_{\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}},\boldsymbol{U}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} \times_3 \boldsymbol{U} - \boldsymbol{\mathcal{A}} *_{\boldsymbol{L}} \boldsymbol{\mathcal{B}}\|_F^2 \quad s.t.\ \boldsymbol{I} = \boldsymbol{U}^T\boldsymbol{U}, \tag{5.10}$$

which can be solved by Algorithm 5.2. Consequently, we obtain an approximate tensor factorization of $\boldsymbol{\mathcal{Y}}$ given by

$$(\hat{\boldsymbol{\mathcal{A}}} *_{\boldsymbol{L}} \hat{\boldsymbol{\mathcal{B}}}) \times_3 \hat{\boldsymbol{U}}^T = (\boldsymbol{L}\hat{\boldsymbol{U}})^{-1}(\boldsymbol{L}(\hat{\boldsymbol{\mathcal{A}}}) \odot_f \boldsymbol{L}(\hat{\boldsymbol{\mathcal{B}}})) = \hat{\boldsymbol{U}}^T(\hat{\boldsymbol{\mathcal{A}}}) *_{\boldsymbol{L}\hat{\boldsymbol{U}}} \hat{\boldsymbol{U}}^T(\hat{\boldsymbol{\mathcal{B}}}), \tag{5.11}$$

and the $(\hat{\boldsymbol{U}}^T(\hat{\boldsymbol{\mathcal{A}}}), \hat{\boldsymbol{U}}^T(\hat{\boldsymbol{\mathcal{B}}}), \hat{\boldsymbol{U}})$ is the optimal solution of (5.12) as well.

$$\min_{\boldsymbol{\mathcal{C}},\boldsymbol{\mathcal{D}},\boldsymbol{U}} \frac{1}{2}\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{C}} *_{\boldsymbol{LU}} \boldsymbol{\mathcal{D}}\|_F^2 \qquad s.t.\ \boldsymbol{I} = \boldsymbol{U}^T\boldsymbol{U} \tag{5.12}$$

5.2.3 Optimization

In this part, we are going to solve the proposed TRPCA model (5.5) by using ADMM, where μ is a positive scalar, and Λ is Lagrange multiplier tensor. According to the framework of ADMM, the above optimization problem can be iteratively solved by minimizing the Lagrangian function of (5.5), *i.e.*, the function (5.13), as follows.

$$\begin{aligned}\mathcal{L}_\mu(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{E}}, \boldsymbol{U}, \Lambda) =& \|\boldsymbol{\mathcal{X}}\|_{*,\boldsymbol{L}} + \lambda\|\boldsymbol{\mathcal{E}}\|_1 + \langle \boldsymbol{U}\boldsymbol{\mathcal{E}}_{(3)} + \boldsymbol{\mathcal{X}}_{(3)} - \boldsymbol{U}\boldsymbol{\mathcal{Y}}_{(3)}, \Lambda\rangle \\ &+ \frac{\mu}{2}\|\boldsymbol{U}\boldsymbol{\mathcal{E}}_{(3)} + \boldsymbol{\mathcal{X}}_{(3)} - \boldsymbol{U}\boldsymbol{\mathcal{Y}}_{(3)}\|_F^2,\end{aligned} \tag{5.13}$$

Step 1 Update $\boldsymbol{\mathcal{X}}^{(t+1)}$ by

$$\begin{aligned}\boldsymbol{\mathcal{X}}^{(t+1)} &= \arg\min_{\boldsymbol{\mathcal{X}}} \mathcal{L}_\mu(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{E}}^{(t)}, \boldsymbol{P}^{(t)}, \boldsymbol{\Lambda}^{(t)}) \\ &= \arg\min_{\boldsymbol{\mathcal{X}}} \|\boldsymbol{\mathcal{X}}\|_{*,\boldsymbol{L}} + \langle \boldsymbol{U}^{(t)}\boldsymbol{\mathcal{E}}^{(t)}_{(3)} + \boldsymbol{\mathcal{X}}_{(3)} - \boldsymbol{U}^{(t)}\boldsymbol{\mathcal{Y}}_{(3)}, \boldsymbol{\Lambda}^{(t)}\rangle \\ &\quad + \frac{\mu}{2}\|\boldsymbol{U}^{(t)}\boldsymbol{\mathcal{E}}^{(t)}_{(3)} + \boldsymbol{\mathcal{X}}_{(3)} - \boldsymbol{U}^{(t)}\boldsymbol{\mathcal{Y}}_{(3)}\|_F^2 \\ &= \arg\min_{\boldsymbol{\mathcal{X}}} \|\boldsymbol{\mathcal{X}}\|_{*,\boldsymbol{L}} + \frac{\mu}{2}\|\boldsymbol{U}^{(t)}\boldsymbol{\mathcal{E}}^{(t)}_{(3)} + \boldsymbol{\mathcal{X}}_{(3)} - \boldsymbol{U}^{(t)}\boldsymbol{\mathcal{Y}}_{(3)} + \frac{1}{\mu}\boldsymbol{\Lambda}^{(t)}\|_F^2 \end{aligned} \tag{5.14}$$

Step 2 Calculate $\boldsymbol{\mathcal{E}}^{(t+1)}$ by

$$\begin{aligned}\boldsymbol{\mathcal{E}}^{(t+1)} &= \arg\min_{\boldsymbol{\mathcal{E}}} \mathcal{L}_\mu(\boldsymbol{\mathcal{X}}^{(t+1)}, \boldsymbol{\mathcal{E}}, \boldsymbol{U}^{(t)}, \boldsymbol{\Lambda}^{(t)}) \\ &= \arg\min_{\boldsymbol{\mathcal{E}}} \lambda\|\boldsymbol{\mathcal{E}}\|_1 + \frac{\mu}{2}\|\boldsymbol{U}^{(t)}\boldsymbol{\mathcal{E}}_{(3)} + \boldsymbol{\mathcal{X}}^{(t+1)}_{(3)} - \boldsymbol{U}^{(t)}\boldsymbol{\mathcal{Y}}_{(3)} + \frac{1}{\mu}\boldsymbol{\Lambda}^{(t)}\|_F^2 \end{aligned} \tag{5.15}$$

Step 3 Calculate $\boldsymbol{U}^{(t+1)}$ by

$$\begin{aligned}\boldsymbol{U}^{(t+1)} &= \arg\min_{\boldsymbol{U}} \mathcal{L}_\mu(\boldsymbol{\mathcal{X}}^{(t+1)}, \boldsymbol{\mathcal{E}}^{(t+1)}, \boldsymbol{U}, \boldsymbol{\Lambda}^{(t)}) \qquad s.t.\ \boldsymbol{I} = \boldsymbol{U}^T\boldsymbol{U} \\ &= \arg\min_{\boldsymbol{U}} \frac{\mu}{2}\|\boldsymbol{U}\boldsymbol{\mathcal{E}}^{(t+1)}_{(3)} + \boldsymbol{\mathcal{X}}^{(t+1)}_{(3)} - \boldsymbol{U}\boldsymbol{\mathcal{Y}}_{(3)} + \frac{1}{\mu}\boldsymbol{\Lambda}^{(t)}\|_F^2 \qquad s.t.\ \boldsymbol{I} = \boldsymbol{U}^T\boldsymbol{U} \end{aligned} \tag{5.16}$$

Let $(\boldsymbol{\mathcal{X}}^{(t+1)}_{(3)} + \frac{1}{\mu}\boldsymbol{\Lambda}^{(t)})(\boldsymbol{\mathcal{Y}}_{(3)} - \boldsymbol{\mathcal{E}}^{(t+1)}_{(3)})^T = \boldsymbol{U}_s\boldsymbol{\Sigma}_s\boldsymbol{V}_s^T$ be the SVD of $(\boldsymbol{\mathcal{X}}^{(t+1)}_{(3)} + \frac{1}{\mu}\boldsymbol{\Lambda}^{(t)})(\boldsymbol{\mathcal{Y}}_{(3)} - \boldsymbol{\mathcal{E}}^{(t+1)}_{(3)})^T$. The optimal solution of (5.16) can be given by $\boldsymbol{U}^{(t+1)} = \boldsymbol{U}_s\boldsymbol{V}_s^T$ [104].

Step 4 Calculate $\boldsymbol{\Lambda}^{(t+1)}$ by

$$\boldsymbol{\Lambda}^{(t+1)} = \boldsymbol{\Lambda}^{(t)} + \mu^{(t)}(\boldsymbol{U}^{(t+1)}\boldsymbol{\mathcal{E}}^{(t+1)}_{(3)} + \boldsymbol{\mathcal{X}}^{(t+1)}_{(3)} - \boldsymbol{U}^{(t+1)}\boldsymbol{\mathcal{Y}}_{(3)}). \tag{5.17}$$

Step 5 Update $\mu^{(t+1)}$ by

$$\mu^{(t+1)} = \min(\rho\mu^{(t)}, \bar{\mu}), \tag{5.18}$$

where $\rho > 1$ and $\bar{\mu}$ is upper bound of $\mu^{(t+1)}$.

Algorithm 5.3: ADMM for solving the proposed TRPCA model (5.5).

```
Input: $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\varepsilon > 0$.
Output: $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$.
while not converged do
1. Compute $\boldsymbol{\mathcal{X}}^{(t+1)}$ by (5.14);
2. Compute $\boldsymbol{\mathcal{E}}^{(t+1)}$ by (5.15);
3. Compute $\boldsymbol{U}^{(t+1)}$ by (5.16);
4. Calculate $\boldsymbol{\Lambda}^{(t+1)}$ by (5.17);
5. Update $\mu^{(t+1)}$ by (5.18);
6. Check the convergence condition: $\|\boldsymbol{\mathcal{X}}^{(t+1)} - \boldsymbol{\mathcal{X}}^{(t)}\|_\infty < \varepsilon$, $\|\boldsymbol{\mathcal{E}}^{(t+1)} - \boldsymbol{\mathcal{E}}^{(t)}\|_\infty < \varepsilon$,
   $\|\boldsymbol{U}^{(t+1)} - \boldsymbol{U}^{(t)}\|_\infty < \varepsilon$, $\|\boldsymbol{\mathcal{X}}^{(t+1)} + \boldsymbol{\mathcal{E}}^{(t+1)} - \boldsymbol{\mathcal{Y}} \times_3 \boldsymbol{U}^{(t+1)}\|_\infty < \varepsilon$;
7. $t = t + 1$.
end while
```

5.3 Experimental Results

In this section, we conducted two kinds of experiments to evaluate the effectiveness of the proposed methods in tensor recovery applications, specifically video reconstruction and image sequence recovery. To demonstrate the capability of our methods in mitigating the performance degradation caused by slice permutation, we tested our methods on tensor data that has undergone random slice permutation. For clarity, we denote the methods tested on tensor data with the original frontal slice order by adding the '-Original' abbreviation. The mean of the Peak Signal-To-Noise Ratio (MPSNR) is employed as the performance metric for all methods in tensor recovery, and the best results are highlighted in bold.

$\kappa = 30$									
	ROM			DFT			DCT		
methods	mobile	bus	tempete	mobile	bus	tempete	mobile	bus	tempete
t-SVD	21.19	25.55	25.24	21.23	25.70	25.32	21.23	25.71	25.30
t-SVD-Original	21.20	25.56	25.24	22.71	27.11	25.88	22.05	26.73	25.58
t-SVD-Ours	**22.08**	**27.00**	**25.72**	**23.11**	**28.27**	**26.27**	**22.22**	**27.57**	**25.76**
TF	21.18	25.57	25.23	21.23	25.70	25.34	21.22	25.71	25.29
TF-Original	21.16	25.56	25.23	22.71	27.11	25.88	22.05	26.73	25.58
TF-Ours	**22.13**	**27.07**	**25.73**	**23.09**	**28.17**	**26.26**	**22.22**	**27.43**	**25.75**

$\kappa = 50$									
	ROM			DFT			DCT		
methods	mobile	bus	tempete	mobile	bus	tempete	mobile	bus	tempete
t-SVD	23.76	28.56	28.51	23.80	28.71	28.58	23.79	28.72	28.54
t-SVD-Original	23.76	28.57	28.52	25.74	30.22	29.43	24.81	29.74	29.00
t-SVD-Ours	**24.92**	**30.07**	**29.15**	**26.24**	**31.30**	**29.82**	**25.08**	**30.58**	**29.21**
TF	23.76	28.58	28.53	23.77	28.70	28.59	23.79	28.72	28.57
TF-Original	23.73	28.57	28.51	25.74	30.22	29.43	24.81	29.74	29.00
TF-Ours	**24.94**	**30.12**	**29.15**	**26.27**	**31.18**	**29.83**	**25.08**	**30.53**	**29.21**

Table 5.1: MPSNR results by different methods

5.3.1 Video Reconstruction

In this section, we compared our proposed methods, including the SVD-based method presented in Algorithm 5.1 and the TF-based method presented in Algorithm 5.2, with the traditional SVD-based method given in (5.6) and TF-based method given in (5.9), for video reconstruction. We evaluated all methods on three video sequences, including *mobile*, *bus*, and *tempete*.

All results are presented in Table 5.1, which shows that our methods get the best performance in TPCA for all cases. Particularly, the performance gap between our methods and the other methods becomes more significant when the video sequence exhibits larger deviations. This indicates that our methods effectively mitigate the slice permutation variability. Moreover, our methods outperform the other methods even when the videos are in their

methods	$\mu^{(0)}$	ρ	λ
TRPCA-ROM	$1e-8$	1.6	$14/\sqrt{\max(I_1, I_2)I_3}$
TRPCA-DFT	$1e-3$	1.9	$1.3/\sqrt{\max(I_1, I_2)I_3}$
TRPCA-DCT	$1e-6$	1.7	$14/\sqrt{\max(I_1, I_2)I_3}$
TRPCA-ROM-our	$5e-7$	1.1	$14/\sqrt{\max(I_1, I_2)I_3}$
TRPCA-DFT-our	$1e-3$	1.1	$\lambda = 1.4/\sqrt{\max(I_1, I_2)I_3}$
TRPCA-DCT-our	$1e-4$	1.2	$\lambda = 14/\sqrt{\max(I_1, I_2)I_3}$

Table 5.2: Parameter setting for different methods.

methods	$c = 0.05$	$c = 0.1$	$c = 0.15$	$c = 0.2$	$c = 0.25$	$c = 0.3$
TRPCA-ROM	31.80	31.28	30.14	28.03	23.71	19.14
TRPCA-ROM-our	**45.41**	**45.02**	**44.26**	**42.37**	**34.40**	**23.09**
TRPCA-DFT	33.96	33.57	33.19	28.15	21.62	17.33
TRPCA-DFT-our	**45.52**	**45.07**	**44.31**	**42.09**	**33.05**	**21.94**
TRPCA-DCT	34.72	34.35	33.75	30.18	23.58	18.31
TRPCA-DCT-our	**43.89**	**44.35**	**44.16**	**40.35**	**28.25**	**20.23**

Table 5.3: MPSNR results by different methods for TRPCA on *Pavia University*

original frame order. For instance, in both *mobile* and *bus* sequences with the original frame order, the MPSNR results obtained by our methods surpass the other methods by more than 0.5 dB for DFT and more than 1 dB for ROM.

methods	$c = 0.05$	$c = 0.1$	$c = 0.15$	$c = 0.2$	$c = 0.25$	$c = 0.3$
TRPCA-ROM	31.78	31.06	30.46	29.42	28.71	27.47
TRPCA-ROM-our	**47.41**	**46.95**	**45.99**	**44.91**	**42.79**	**39.96**
TRPCA-DFT	37.66	36.57	35.41	33.98	27.29	20.66
TRPCA-DFT-our	**49.51**	**48.43**	**46.64**	**44.06**	**39.57**	**29.19**
TRPCA-DCT	36.37	35.75	34.45	33.64	32.4	30.89
TRPCA-DCT-our	**47.22**	**46.57**	**45.63**	**44.41**	**39.97**	**35.98**

Table 5.4: MPSNR results by different methods for TRPCA on *Botswana*

methods	$c = 0.05$	$c = 0.1$	$c = 0.15$	$c = 0.2$	$c = 0.25$	$c = 0.3$
TRPCA-ROM	32.49	32.10	31.67	30.88	29.96	28.51
TRPCA-ROM-our	**34.79**	**34.50**	**34.19**	**33.91**	**33.65**	**33.20**
TRPCA-DFT	35.19	34.45	31.81	24.44	19.68	17.40
TRPCA-DFT-our	**37.09**	**36.84**	**36.50**	**33.32**	**23.59**	**18.45**
TRPCA-DCT	33.05	32.52	31.94	31.23	30.10	27.27
TRPCA-DCT-our	**35.45**	**34.66**	**34.74**	**34.23**	**34.15**	**32.71**

Table 5.5: MPSNR results by different methods for TRPCA on *Indian_pines*

5.3.2 Image Sequence Recovery

In this part, we compared the performance of the proposed methods, including TRPCA-ROM-our, TRPCA-DFT-our, and TRPCA-DCT-our (our TRPCA method presented in Algorithm 5.3 by using ROM, DFT, and DCT, respectively) with TRPCA-ROM, TRPCA-DFT, and TRPCA-DCT[2] (TRPCA method given in [50] using ROM, DFT, and DCT, respectively) in image sequence recovery. We evaluated all methods on three hyperspectral image databases, including *Pavia University*[3], *Botswana*[3], and *Indian_pines*[3]. Each data has dimensions of $I_1 \times I_2 \times I_3$ and is contaminated by random-valued impulse noise with density level $c = 0.05 : 0.1 : 0.25$, where I_3 represents the number of spectral bands. The parameters for all methods are empirically set and presented in Table 5.2.

The MRSNR results are presented in Table 5.3-5.5, where our proposed method consistently outperforms other methods significantly across all cases. Particularly, our method achieves more than a 10 dB improvement over other methods for most cases on the *Pavia University* and *Botswana* datasets. These results demonstrated the advantage of our methods over traditional TRPCA methods for image sequence recovery when the order of the image sequence is perturbed.

[2] https://github.com/canyilu/LibADMM-toolbox

[3] http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensin_Scenes

5.4 Summary

This paper aims to investigate the issue of slice permutation variability (SPV) in t-product-based tensor recovery methods and focuses on accurately recovering low-rank tensors from high-dimensional tensor data with perturbations in the order of tensor slice sequences. To address this problem, we propose a generalized framework (5.4) that tackles SPV in t-product-based tensor recovery methods. We apply this framework to TPCA (both SVD-based and TF-based approaches) and TRPCA and employ BCD and ADMM algorithms to solve the resulting models, respectively. The experimental results demonstrate the effectiveness of the proposed methods in mitigating SPV in t-product-based tensor recovery.

Chapter 6

A Novel Tensor Factorization-Based Method for Tensor Recovery

6.1 Introduction

Although the t-product-based tensor completion methods have achieved great success, there are still two challenges: (1) The TCTF [102] is based on a basic hypothesis that the tensor with tensor tubal rank κ can be approximately decomposed to the t-product of two skinny tensors $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \kappa \times I_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{\kappa \times I_2 \times I_3}$ (κ could be obtained by estimating the tensor tubal rank), so it overly relies on the rank estimation strategy. On the other hand, due to the lack of a rank-increasing scheme, the rank estimation strategy given in [102] often underestimates the true rank, causing performance degradation in TCTF. (2) Currently, iterative re-weighted tensor nuclear norm algorithms [74, 68] and generalized tensor singular value thresholding [93] were proposed to solve the non-convex approximation of tensor recovery. These algorithms require computing the t-SVD of the original large tensor in each iteration, causing a high computational cost. Therefore, it is necessary to develop an efficient and effective tensor recovery framework for a wide range of surrogate functions.

To address the issues mentioned above, this paper proposes a novel tensor completion framework with a new tensor norm that utilizes a dual low-rank constraint (TCDLR), and its goal is to avoid the high computational cost in the standard t-SVD-based method and achieve superior recovery results. More specifically, the features of the proposed method are as follows:

- First, a new tensor norm with a dual low-rank constraint is given to utilize the low-rank prior and the true tubal-rank information at the same time. Based on the proposed tensor norm, a series of surrogate functions (possibly non-convex) of the tensor tubal rank in the resulting tensor completion model (TCDLR) are allowed to achieve better performance in harnessing low-rankness within the tensor data and solving the over-penalization problem in the TNN-based tensor completion. Besides, Property 6.1, Theorem 6.1, and synthetic experiments confirm that TCDLR can be less negatively affected by the misestimation of tensor rank compared with standard tensor factorization-based methods.
- Second, an optimization algorithm is developed to solve TCDLR efficiently, in which the t-SVD of a smaller tensor instead of the big one is computed by using a simple trick. As a result, the total cost at each iteration of the developing algorithm is reduced to $\mathcal{O}(I_1 I_2 I_3 \log I_3 + \kappa I_1 I_2 I_3)$ from $\mathcal{O}(I_{(1)} I_{(2)}^2 I_3 + I_1 I_2 I_3 \log I_3)$ achieved with standard t-SVD-based methods, where $I_{(1)} = \max(I_1, I_2)$ and $I_{(2)} = \min(I_1, I_2)$. Here, $\kappa \ll I_{(2)}$ is the estimation of tensor rank. The convergence of the optimization algorithm was

analyzed experimentally.

- Third, since the tensor rank of real tensor data is unknown, a novel rank estimation method is proposed, which adopts an increasing and decreasing strategy to estimate the tensor rank more precisely. By combining TCDLR with the proposed rank estimation method, an efficient tensor completion framework (TCDLR-RE) is established to accurately and effectively recover the principal components (the low-rank tensor). The experiments demonstrate the high efficiency and effectiveness of TCDLR-RE.

6.2 Proposed Tensor Completion Framework

6.2.1 Formulation of TCDLR

Although TCTF can well address the issue of high computational complexity caused by t-SVD for large tensors at each iteration, its over-reliance on the rank estimation strategy often leads to degraded recovery accuracy. According to Property 6.1 (ii), the low-rank estimation of TCTF and TC-RE will deviate significantly from the true value when the rank estimation deviates from the true rank. Even worse, the true rank is difficult to be estimated accurately, especially under a low sampling rate. Therefore, in addition to achieving a better rank estimation, a new effective tensor completion model needs to be developed.

Property 6.1. *For* $\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$*, then*

(i) $\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 \geq \frac{1}{I_3} \sum_i (\sigma_i(\bar{\boldsymbol{Y}}) - \sigma_i(\bar{\boldsymbol{X}}))^2$;

(ii) $\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}}\|_F^2 \geq \frac{1}{I_3} \sum_{i=1}^{I_3} \sum_{\operatorname{rank}(\boldsymbol{\mathcal{Y}}) < j \leq \operatorname{rank}(\boldsymbol{\mathcal{X}})} \sigma_j(\bar{\boldsymbol{X}}_i)^2$ *if* $\operatorname{rank}(\boldsymbol{\mathcal{Y}}) < \operatorname{rank}(\boldsymbol{\mathcal{X}})$.

The fundamental idea of our approach is to calculate the t-SVD of the obtained smaller tensor instead of the original tensor by utilizing tensor tubal rank information that could be provided by rank estimation methods, thus reducing the computational complexity of the original t-SVD-based methods. Therefore, a new tensor norm is introduced,

$$\|\boldsymbol{\mathcal{X}}\|_{*,(\kappa,\mathcal{G})} = \begin{cases} \infty, & \text{if } \operatorname{rank}(\boldsymbol{\mathcal{X}}) > \kappa; \\ \|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}}, & \text{if } \operatorname{rank}(\boldsymbol{\mathcal{X}}) \leq \kappa, \end{cases} \tag{6.1}$$

and the following generalized framework (TCDLR) is established:

$$\min_{\boldsymbol{\mathcal{X}}} \|\boldsymbol{\mathcal{X}}\|_{*,(\kappa,\mathcal{G})} \qquad s.t.\ \mathbf{P}_\Omega(\boldsymbol{\mathcal{X}}) = \mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}), \tag{6.2}$$

where $\mathcal{G}$ can be any of the surrogate functions listed in Table 2.2. By minimizing $\|\boldsymbol{\mathcal{X}}\|_{*,(\kappa,\mathcal{G})}$, both the low-rank prior and tensor tubal rank information $\operatorname{rank}(\boldsymbol{\mathcal{X}}) \leq \kappa$ are considered. Theorem 6.1 shows the robustness of TCDLR to inaccurate rank estimations for κ. Noting that since $\|\cdot\|_{*,\mathcal{G}}$ is a better approximation of the tensor tubal rank than the tensor nuclear norm, the optimal solution to (6.4) is expected to be low rank. When $\operatorname{rank}(\widetilde{\boldsymbol{\mathcal{X}}}) \leq \kappa < \operatorname{rank}(\mathring{\boldsymbol{\mathcal{X}}})$, the estimation κ provides a better prior for the true tensor tubal rank, which helps to recover the low-rank tensor more accurately. Later, the effectiveness of TCDLR in tensor recovery will be demonstrated by experiments further.

$$\min_{\boldsymbol{\mathcal{X}}} \operatorname{rank}(\boldsymbol{\mathcal{X}}) \qquad s.t.\ \mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) = \mathbf{P}_\Omega(\boldsymbol{\mathcal{X}}), \tag{6.3}$$

$$\min_{\boldsymbol{\mathcal{X}}} \|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} \qquad s.t.\ \mathbf{P}_\Omega(\boldsymbol{\mathcal{X}}) = \mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}). \tag{6.4}$$

Theorem 6.1. *Let $\mathring{\boldsymbol{\mathcal{X}}}$, $\hat{\boldsymbol{\mathcal{X}}}$ and $\widetilde{\boldsymbol{\mathcal{X}}}$ be the optimal solutions to* (6.4)*,* (6.2)*, and* (6.3)*, respectively. Then, we have:*

(i) $\operatorname{rank}(\mathring{\boldsymbol{\mathcal{X}}}) \geq \operatorname{rank}(\widetilde{\boldsymbol{\mathcal{X}}})$ *holds;*

(ii) *If* $\kappa \geq \operatorname{rank}(\mathring{\boldsymbol{\mathcal{X}}})$*,* $\hat{\boldsymbol{\mathcal{X}}}$ *is an optimal solution of* (6.4) *and* $\mathring{\boldsymbol{\mathcal{X}}}$ *is an optimal solution of* (6.2)*;*

(iii) *If* $\kappa = \operatorname{rank}(\widetilde{\boldsymbol{\mathcal{X}}})$*,* $\hat{\boldsymbol{\mathcal{X}}}$ *is an optimal solution of* (6.3)*;*

(iv) $\kappa < \operatorname{rank}(\widetilde{\boldsymbol{\mathcal{X}}})$ *holds if and only if* $\|\hat{\boldsymbol{\mathcal{X}}}\|_{*,(\kappa,\mathcal{G})} = \infty$.

As it will be shown later, benefiting from the proposed dual low-rank constraint, TCDLR can avoid performing the t-SVD operation for the original bigger tensor that causes high time consumption.

6.2.2 The Developing Optimization Algorithm for Solving TCDLR

6.2.2.1 A Trick to Efficiently Solve TCDLR

Since there exist $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \kappa \times I_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{\kappa \times I_2 \times I_3}$ such that $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}$ and $\|\boldsymbol{\mathcal{X}}\|_{*,(\kappa,\mathcal{G})} = \|\boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}\|_{*,\mathcal{G}}$ when $\operatorname{rank}(\boldsymbol{\mathcal{X}}) \leq \kappa$, the following problem is considered:

$$\min_{\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}}} \|\boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}\|_{*,\mathcal{G}} \qquad s.t.\ \mathbf{P}_{\Omega}(\boldsymbol{\mathcal{M}}) = \mathbf{P}_{\Omega}(\boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}), \tag{6.5}$$

where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \kappa \times I_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{\kappa \times I_2 \times I_3}$. If $(\hat{\boldsymbol{\mathcal{A}}}, \hat{\boldsymbol{\mathcal{B}}})$ is an optimal solution to (6.5), we have $\|\hat{\boldsymbol{\mathcal{A}}} * \hat{\boldsymbol{\mathcal{B}}}\|_{*,(\kappa,\mathcal{G})} = \|\hat{\boldsymbol{\mathcal{A}}} * \hat{\boldsymbol{\mathcal{B}}}\|_{*,\mathcal{G}} \leq \|\hat{\boldsymbol{\mathcal{X}}}\|_{*,(\kappa,\mathcal{G})}$, where $\hat{\boldsymbol{\mathcal{X}}}$ is the optimal solution to (6.2). Thus, it can be concluded that $\hat{\boldsymbol{\mathcal{A}}} * \hat{\boldsymbol{\mathcal{B}}}$ is an optimal solution to (6.2).

Theorem 6.2. *Let* $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}$*, where* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \kappa \times I_3}$ *and* $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{\kappa \times I_2 \times I_3}$*. If* $\boldsymbol{\mathcal{B}}^T = \boldsymbol{\mathcal{Q}}^T * \boldsymbol{\mathcal{R}}^T$ *is the QR decomposition of* $\boldsymbol{\mathcal{B}}^T$ *and* $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{R}}$*,*

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{Z}} * \boldsymbol{\mathcal{Q}}$$

and

$$\mathcal{D}_{\mathcal{G}}(\boldsymbol{\mathcal{Y}},\tau)=\mathcal{D}_{\mathcal{G}}(\boldsymbol{\mathcal{Z}},\tau)*\boldsymbol{\mathcal{Q}}$$

hold, where $\boldsymbol{\mathcal{Z}}\in\mathbb{R}^{I_1\times\kappa\times I_3}$, $\boldsymbol{\mathcal{Q}}\in\mathbb{R}^{\kappa\times I_2\times I_3}$, *and*

$$\mathcal{D}_{\mathcal{G}}(\boldsymbol{\mathcal{Y}},\tau)=\arg\min_{\boldsymbol{\mathcal{X}}}\frac{1}{2}\|\boldsymbol{\mathcal{Y}}-\boldsymbol{\mathcal{X}}\|_F^2+\tau\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}}.$$

According to Theorem 6.2, the t-SVD of the large tensor can be avoided by constructing $\mathcal{D}_{\mathcal{G}}(\boldsymbol{\mathcal{A}}*\boldsymbol{\mathcal{B}},\tau)$. To achieve this goal, this paper introduces an auxiliary tensor $\boldsymbol{\mathcal{X}}$ such that $\boldsymbol{\mathcal{X}}=\boldsymbol{\mathcal{A}}*\boldsymbol{\mathcal{B}}$. Meanwhile, (6.5) is convert to

$$\min_{\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{X}}}\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}}\qquad s.t.\ \mathbf{P}_\Omega(\boldsymbol{\mathcal{M}})=\mathbf{P}_\Omega(\boldsymbol{\mathcal{A}}*\boldsymbol{\mathcal{B}}),\quad \boldsymbol{\mathcal{X}}=\boldsymbol{\mathcal{A}}*\boldsymbol{\mathcal{B}},$$

which is equivalent to (6.6) when $\mu=+\infty$.

$$\min_{\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{X}}}\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}}+\frac{\mu}{2}\|\boldsymbol{\mathcal{A}}*\boldsymbol{\mathcal{B}}-\boldsymbol{\mathcal{X}}\|_F^2\qquad s.t.\ \mathbf{P}_\Omega(\boldsymbol{\mathcal{M}})=\mathbf{P}_\Omega(\boldsymbol{\mathcal{A}}*\boldsymbol{\mathcal{B}})$$

When (6.6) is solved by iterative algorithms such as the Alternating Direction Method of Multiplier (ADMM) [46], the t-SVD operation is only involved in solving

$$\mathcal{D}_{\mathcal{G}}(\boldsymbol{\mathcal{A}}*\boldsymbol{\mathcal{B}},\tau)=\arg\min_{\boldsymbol{\mathcal{X}}}\tau\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}}+\frac{1}{2}\|\boldsymbol{\mathcal{A}}*\boldsymbol{\mathcal{B}}-\boldsymbol{\mathcal{X}}\|_F^2.\tag{6.6}$$

According to Theorem 6.2, a fast solver for (6.6) is presented in Algorithm 6.1.

6.2.2.2 The Developing Optimization Algorithm based on ADMM

By utilizing the above trick, TCDLR can be solved as follows. To simplify (6.6), an auxiliary tensor $\boldsymbol{\mathcal{E}}$ is introduced. Then, (6.6) can be rewritten to:

$$\min_{\boldsymbol{\mathcal{E}}\in\mathbb{I},\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{X}}}\frac{\mu}{2}\|\boldsymbol{\mathcal{A}}*\boldsymbol{\mathcal{B}}-\boldsymbol{\mathcal{X}}\|_F^2+\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}}\qquad s.t.\ \mathbf{P}_\Omega(\boldsymbol{\mathcal{M}})=\boldsymbol{\mathcal{A}}*\boldsymbol{\mathcal{B}}+\boldsymbol{\mathcal{E}},\tag{6.7}$$

```
Algorithm 6.1: Fast Solver to (6.6)
  Input: $\mathcal{A} \in \mathbb{R}^{I_1\times\kappa\times I_3}$, $\mathcal{B} \in \mathbb{R}^{I_1\times\kappa\times I_3}$,$\lambda > 0$.
  Output: $\mathcal{S}_{\lambda,\mathcal{G}}(\mathcal{A} * \mathcal{B})$, $\mathcal{Z}$ ,$\mathcal{Q}$ and $\mathcal{R}$.
  1. Compute $\mathcal{R}$ and $\mathcal{Q}$ by QR decomposition [41] of $(\mathcal{B}^{(t+1)})^T$: $\mathcal{B}^T = \mathcal{Q}^T * \mathcal{R}^T$;
  2. Compute $\mathcal{Z} = \mathcal{A} * \mathcal{R}$;
  3. Obtain $\mathcal{S}_{\lambda,\mathcal{G}}(\mathcal{Z})$ by Generalized Tensor Singular Value Thresholding
     (GTSVT)[93]
  4. Compute $\mathcal{S}_{\lambda,\mathcal{G}}(\mathcal{A} * \mathcal{B}) = \mathcal{S}_{\lambda,\mathcal{G}}(\mathcal{Z}) * \mathcal{Q}$.
```

where $\mathbb{I} = \{\mathcal{E}|\mathbf{P}_\Omega(\mathcal{E}) = \mathbf{0}\}$.

The lagrangian function of (6.7) is formulated as

$$\begin{aligned}\mathcal{L}_\mu(\mathcal{X},\mathcal{E},\mathcal{A},\mathcal{B},\mathcal{Y}) =&\|\mathcal{X}\|_{*,\mathcal{G}} + \frac{\mu}{2}\|\mathcal{X} - \mathcal{A} * \mathcal{B}\|_F^2 + \langle \mathbf{P}_\Omega(\mathcal{M}) - \mathcal{A} * \mathcal{B} - \mathcal{E}, \mathcal{Y}\rangle \\ &+ \frac{\mu}{2}\|\mathbf{P}_\Omega(\mathcal{M}) - \mathcal{A} * \mathcal{B} - \mathcal{E}\|_F^2, \end{aligned} \tag{6.8}$$

where $\mathcal{Y}$ is Lagrange multiplier, and μ is a positive scalar. Therefore, (6.7) is iteratively solved by ADMM as follows.

Step 1 Calculate $\mathcal{A}^{(t+1)}$ by

$$\begin{aligned}\mathcal{A}^{(t+1)} &= \arg\min_{\mathcal{A}} \mathcal{L}_\mu(\mathcal{X}^{(t)},\mathcal{E}^{(t)},\mathcal{A},\mathcal{B}^{(t)},\mathcal{Y}^{(t)}) \\ &= \arg\min_{\mathcal{A}} \|\mathcal{X}^{(t)} - \mathcal{A} * \mathcal{B}^{(t)}\|_F^2 + \|\mathbf{P}_\Omega(\mathcal{M}) - \mathcal{A} * \mathcal{Q}^{(t)} - \mathcal{E}^{(t)} + \frac{1}{\mu^{(t)}}\mathcal{Y}^{(t)}\|_F^2 \\ &= \arg\min_{\mathcal{A}} \|\mathcal{C}^{(t+1)} - \mathcal{A} * \mathcal{B}^{(t)}\|_F^2 = \mathcal{C}^{(t+1)} * (\mathcal{B}^{(t)})^T * (\mathcal{B}^{(t)} * (\mathcal{B}^{(t)})^T)^\dagger \end{aligned} \tag{6.9}$$

where $\mathcal{C}^{(t+1)} = (\mathbf{P}_\Omega(\mathcal{M}) - \mathcal{E}^{(t)} + \mathcal{X}^{(t)} + \frac{\mathcal{Y}^{(t)}}{\mu^{(t)}})/2$.

Algorithm 6.2: Solve (6.2) by the Developing Optimization Algorithm

```
**Input:** The observed tensor $\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) \in \mathbb{R}^{I_1\times I_2\times I_3}$, the support set $\Omega$, $\rho = 1.3$,
    $\bar{\mu} = 10^{14}$, $\varepsilon = 10^{-9}$, $\kappa$.
**Output:** $\boldsymbol{\mathcal{X}}^{(t+1)}$
**Initialize:** $\mu^{(0)}$, $\boldsymbol{\mathcal{B}}^{(0)}$, $\boldsymbol{\mathcal{E}}^{(0)}$, $\boldsymbol{\mathcal{X}}^{(0)}$, $\boldsymbol{\mathcal{Y}}^{(0)}$, and $t = 0$
**While not converge do**
1. Compute $\boldsymbol{\mathcal{A}}^{(t+1)}$ by (6.16);
2. Compute $\boldsymbol{\mathcal{B}}^{(t+1)}$ by (6.10);
3. Calculate $\boldsymbol{\mathcal{Q}}^{(t+1)}$, $\boldsymbol{\mathcal{Z}}^{(t+1)}$ and $\boldsymbol{\mathcal{X}}^{(t+1)}$ by Algorithm 6.1;
4. Calculate $\boldsymbol{\mathcal{P}}^{(t+1)}$ by $\boldsymbol{\mathcal{P}}^{(t+1)} = \boldsymbol{\mathcal{Z}}^{(t+1)} * \boldsymbol{\mathcal{Q}}^{(t+1)}$;
5. Calculate $\boldsymbol{\mathcal{E}}^{(t+1)}$ by (6.12);
6. Calculate $\boldsymbol{\mathcal{Y}}^{(t+1)}$ by (6.13);
7. Calculate $\mu^{(t+1)}$ by (6.14);
8. Check the convergence condition: $\|\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) - \boldsymbol{\mathcal{P}}^{(t)} - \boldsymbol{\mathcal{E}}^{(t)}\|_\infty < \varepsilon$,
   $\|\boldsymbol{\mathcal{P}}^{(t+1)} - \boldsymbol{\mathcal{P}}^{(t)}\|_\infty < \varepsilon$ ,$\|\boldsymbol{\mathcal{E}}^{(t+1)} - \boldsymbol{\mathcal{E}}^{(t)}\|_\infty < \varepsilon$;
9. $t = t + 1$.
**end while**
```

Step 2 Calculate $\boldsymbol{\mathcal{B}}^{(t+1)}$ by

$$\begin{aligned}\boldsymbol{\mathcal{B}}^{(t+1)} &= \arg\min_{\boldsymbol{\mathcal{B}}} \mathcal{L}_\mu(\boldsymbol{\mathcal{X}}^{(t)}, \boldsymbol{\mathcal{E}}^{(t)}, \boldsymbol{\mathcal{A}}^{(t+1)}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{Y}}^{(t)})\\ &= \arg\min_{\boldsymbol{\mathcal{B}}} \|\boldsymbol{\mathcal{C}}^{(t+1)} - \boldsymbol{\mathcal{A}}^{(t+1)} * \boldsymbol{\mathcal{B}}\|_F^2 = ((\boldsymbol{\mathcal{A}}^{(t+1)})^T * \boldsymbol{\mathcal{A}}^{(t+1)})^\dagger * (\boldsymbol{\mathcal{A}}^{(t+1)})^T * \boldsymbol{\mathcal{C}}^{(t+1)}\end{aligned} \tag{6.10}$$

Step 3 Calculate $\boldsymbol{\mathcal{X}}^{(t+1)}$ by

$$\boldsymbol{\mathcal{X}}^{(t+1)} = \arg\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2}\|\boldsymbol{\mathcal{A}}^{(t+1)} * \boldsymbol{\mathcal{B}}^{(t+1)} - \boldsymbol{\mathcal{X}}\|_F^2 + \frac{1}{\mu^{(t)}}\|\boldsymbol{\mathcal{X}}\|_{*,\mathcal{G}} = \mathcal{D}_{\mathcal{G}}(\boldsymbol{\mathcal{A}}^{(t+1)} * \boldsymbol{\mathcal{B}}^{(t+1)}, \frac{1}{\mu^{(t)}}), \tag{6.11}$$

and obtain $\boldsymbol{\mathcal{Z}}^{(t+1)}$ and $\boldsymbol{\mathcal{Q}}^{(t+1)}$ by Algorithm 6.1.

Step 4 Calculate $\boldsymbol{\mathcal{E}}^{(t+1)}$ by

$$\boldsymbol{\mathcal{E}}^{(t+1)} = \arg\min_{\boldsymbol{\mathcal{E}}} \frac{\mu^{(t)}}{2}\|\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) - \boldsymbol{\mathcal{P}}^{(t+1)} + \frac{\boldsymbol{\mathcal{Y}}^{(t)}}{\mu^{(t)}} - \boldsymbol{\mathcal{E}}\|_F^2, \qquad s.t.\ \boldsymbol{\mathcal{E}} \in \mathbb{I}, \tag{6.12}$$

where $\boldsymbol{\mathcal{P}}^{(t+1)} = \boldsymbol{\mathcal{Z}}^{(t+1)} * \boldsymbol{\mathcal{Q}}^{(t+1)}$.

Step 5 Calculate $\boldsymbol{\mathcal{Y}}^{(t)}$ by

$$\boldsymbol{\mathcal{Y}}^{(t+1)} = \mu^{(t)}(\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) - \boldsymbol{\mathcal{P}}^{(t+1)} - \boldsymbol{\mathcal{E}}^{(t+1)}) + \boldsymbol{\mathcal{Y}}^{(t)}. \tag{6.13}$$

Step 6 Calculate $\mu^{(t+1)}$ by

$$\mu^{(t+1)} = \min(\bar{\mu}, \rho\mu^{(t)}), \tag{6.14}$$

where $\bar{\mu}$ is the upper bound of $\mu^{(t+1)}$, and $\rho > 1$.

According to $\boldsymbol{\mathcal{A}}^{(t)} * \boldsymbol{\mathcal{B}}^{(t)} = \boldsymbol{\mathcal{Z}}^{(t)} * \boldsymbol{\mathcal{Q}}^{(t)}$, $\mathrm{range}(\boldsymbol{\mathcal{Z}}^{(t)})$ and $\mathrm{range}(\boldsymbol{\mathcal{Q}}^{(t)})$ are the column space and row space of $\boldsymbol{\mathcal{A}}^{(t)} * \boldsymbol{\mathcal{B}}^{(t)}$, respectively. Then, with $(\boldsymbol{\mathcal{B}}^{(t+1)})^T = (\boldsymbol{\mathcal{Q}}^{(t+1)})^T * (\boldsymbol{\mathcal{R}}^{(t+1)})^T$, we have

$$\begin{aligned} &\min_{\boldsymbol{\mathcal{B}}} \|\boldsymbol{\mathcal{C}}^{(t+1)} - \boldsymbol{\mathcal{C}}^{(t+1)} * (\boldsymbol{\mathcal{Q}}^{(t)})^T * \boldsymbol{\mathcal{B}}\|_F^2 \\ \le &\min_{\boldsymbol{\mathcal{B}}} \|\boldsymbol{\mathcal{C}}^{(t+1)} - \boldsymbol{\mathcal{C}}^{(t+1)} * (\boldsymbol{\mathcal{B}}^{(t)})^T * (\boldsymbol{\mathcal{B}}^{(t)} * (\boldsymbol{\mathcal{B}}^{(t)})^T)^\dagger * \boldsymbol{\mathcal{B}}\|_F^2. \end{aligned} \tag{6.15}$$

Therefore, $\boldsymbol{\mathcal{A}}^{(t+1)}$ is updated by

$$\arg\min_{\boldsymbol{\mathcal{A}}} \|\boldsymbol{\mathcal{C}}^{(t+1)} - \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{Q}}^{(t)}\|_F^2 = \boldsymbol{\mathcal{C}}^{(t+1)}(\boldsymbol{\mathcal{Q}}^{(t)})^T \tag{6.16}$$

for easy computation. The whole procedure of the algorithm is presented in Algorithm 6.2.

Table 6.1: Computational complexity of Algorithm 6.2

Step no.	Operation	Complexity
1	$\overbrace{\boldsymbol{\mathcal{A}}^{(t+1)}}^{I_1\times\kappa\times I_3} = \arg\min_{\boldsymbol{\mathcal{A}}} \mu\| \overbrace{\boldsymbol{\mathcal{C}}^{(t+1)}}^{I_1\times I_2\times I_3} - \overbrace{\boldsymbol{\mathcal{A}}}^{I_1\times\kappa\times I_3} * \overbrace{\boldsymbol{\mathcal{B}}^{(t)}}^{\kappa\times I_2\times I_3} \|_F^2$	$\mathcal{O}(I_1I_2I_3\log I_3 + \kappa I_1I_2I_3)$
2	$\overbrace{\boldsymbol{\mathcal{B}}^{(t+1)}}^{\kappa\times I_2\times I_3} = \arg\min_{\boldsymbol{\mathcal{B}}} \mu\| \overbrace{\boldsymbol{\mathcal{C}}^{(t+1)}}^{I_1\times I_2\times I_3} - \overbrace{\boldsymbol{\mathcal{A}}^{(t+1)}}^{I_1\times\kappa\times I_3} * \overbrace{\boldsymbol{\mathcal{B}}}^{\kappa\times I_2\times I_3} \|_F^2$	$\mathcal{O}(I_1I_2I_3\log I_3 + \kappa I_1I_2I_3)$
3	$[\overbrace{(\boldsymbol{\mathcal{Q}}^{(t+1)})^T}^{I_2\times\kappa\times I_3}, \overbrace{(\boldsymbol{\mathcal{R}}^{(t+1)})^T}^{\kappa\times\kappa\times I_3}] = \mathrm{QR}(\overbrace{(\boldsymbol{\mathcal{B}}^{(t+1)})^T}^{I_2\times\kappa\times I_3})$	$\mathcal{O}(I_2\kappa I_3\log I_3 + I_2\kappa^2 I_3)$
3	$\overbrace{\boldsymbol{\mathcal{Z}}^{(t+1)}}^{I_1\times\kappa\times I_3} = \overbrace{\boldsymbol{\mathcal{A}}^{(t+1)}}^{I_1\times\kappa\times I_3} * \overbrace{\boldsymbol{\mathcal{R}}^{(t+1)}}^{\kappa\times\kappa\times I_3}$	$\mathcal{O}((I_1+\kappa)\kappa I_3\log I_3 + I_1\kappa^2 I_3)$
3	$\mathcal{S}_{\frac{1}{\mu},\mathcal{G}}(\overbrace{\boldsymbol{\mathcal{Z}}^{(t+1)}}^{I_1\times\kappa\times I_3})$	$\mathcal{O}(I_1\kappa I_3\log I_3 + I_1\kappa^2 I_3)$
4	$\overbrace{\boldsymbol{\mathcal{P}}^{(t+1)}}^{I_1\times I_2\times I_3} = \overbrace{\boldsymbol{\mathcal{A}}^{(t+1)}}^{I_1\times\kappa\times I_3} * \overbrace{\boldsymbol{\mathcal{B}}^{(t+1)}}^{\kappa\times I_2\times I_3}$	$\mathcal{O}(I_1I_2I_3\log I_3 + I_1\kappa I_2I_3)$
total	total cost at each iteration	$\mathcal{O}(I_1I_2I_3\log I_3 + \kappa I_1I_2I_3)$

6.2.3 Complexity Analysis

According to Algorithm 6.2, since TCDLR only requires computing the t-SVD of $\boldsymbol{\mathcal{Z}}$ with a smaller size, it avoids performing the t-SVD operation with a high time complexity of $\mathcal{O}(\kappa I_1I_2I_3)$. The step-by-step computational complexity of the proposed algorithm is summarized in Table 6.1. The total cost at each iteration (requires computing of FFT [50]) is $\mathcal{O}(I_1I_2I_3\log I_3 + \kappa I_1I_2I_3)$, as shown in Table 6.1.

6.2.4 Rank Estimation

In the following, a method is proposed to estimate κ in (6.5) for a lower bound $\kappa_{\min}$ and an upper bound $\kappa_{\max}$ of κ.

6.2.4.1 Increase Strategy

According to Theorem 6.1 (iv), $\kappa < \operatorname{rank}(\widetilde{\boldsymbol{\mathcal{X}}})$ holds and κ should be increased when $\|\hat{\boldsymbol{\mathcal{X}}}\|_{*,(\kappa,\mathcal{G})} = \infty$. In the following, the case of $\|\hat{\boldsymbol{\mathcal{X}}}\|_{*,(\kappa,\mathcal{G})} = \infty$ is considered.

Since $\|\hat{\boldsymbol{\mathcal{X}}}\|_{*,(\kappa,\mathcal{G})} = \infty$, $\kappa < \operatorname{rank}(\boldsymbol{\mathcal{X}})$ holds for $\forall \boldsymbol{\mathcal{X}} \in \{\boldsymbol{\mathcal{X}} | \mathrm{P}_\Omega(\boldsymbol{\mathcal{M}}) = \mathrm{P}_\Omega(\boldsymbol{\mathcal{X}})\}$, (6.5) has no solution. A preliminary idea for the critical condition of increasing κ is $\|\boldsymbol{\mathcal{C}}^{(t)} - \boldsymbol{\mathcal{A}}^{(t)} * \boldsymbol{\mathcal{B}}^{(t)}\|_F \nrightarrow 0$ when $t \to \infty$. However, it is difficult to determine whether $\|\boldsymbol{\mathcal{C}}^{(t)} - \boldsymbol{\mathcal{A}}^{(t)} * \boldsymbol{\mathcal{B}}^{(t)}\|_F$ converges to 0 and the large value of $\|\boldsymbol{\mathcal{C}}^{(t)} - \boldsymbol{\mathcal{A}}^{(t)} * \boldsymbol{\mathcal{B}}^{(t)}\|_F$ should be allowed in the early iteration of the algorithm. According to [97], $\widetilde{\boldsymbol{\mathcal{A}}} * \widetilde{\boldsymbol{\mathcal{B}}}$ is the rank-κ estimation of $\boldsymbol{\mathcal{C}}^{(t)}$ if $(\widetilde{\boldsymbol{\mathcal{A}}}, \widetilde{\boldsymbol{\mathcal{B}}})$ is the the optimal solution to $\min_{\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}}} \|\boldsymbol{\mathcal{C}}^{(t)} - \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}\|_F$. Therefore, we increases κ only if some important components of $\boldsymbol{\mathcal{C}}^{(t)}$ are lost in $\boldsymbol{\mathcal{A}}^{(t)} * \boldsymbol{\mathcal{B}}^{(t)}$.

Defining $\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{C}}^{(t)} - \boldsymbol{\mathcal{A}}^{(t)} * \boldsymbol{\mathcal{B}}^{(t)} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, according to [73], if $\bar{\boldsymbol{D}}_i$ is Gaussian distributed for given i (in this case, $\bar{\boldsymbol{D}}_i$ contains less meaningful information), $\sigma_1(\frac{\bar{\boldsymbol{D}}_i - \hat{\mu}_i(\bar{\boldsymbol{D}}_i)}{\hat{\delta}_i(\bar{\boldsymbol{D}}_i)}) \leq \sqrt{I_1} + \sqrt{I_2} + h (h \geq 1)$ holds with a high probability, where $\hat{\mu}_i(\bar{\boldsymbol{D}}_i)$ is the mean value of all elements in $\bar{D}_i$ and $\hat{\delta}_i(\bar{\boldsymbol{D}}_i) = \sqrt{\frac{1}{I_1 I_2 - 1} \sum_{w=1}^{I_1} \sum_{j=1}^{I_2} ([\bar{\boldsymbol{D}}_i]_{w,j} - \hat{\mu}_i)^2}$. Here, this paper uses $\bar{s}_{\max}(\frac{\bar{\boldsymbol{D}}_i - \hat{\mu}_i}{\hat{\delta}_i}) = \|\boldsymbol{p}^* \cdot \frac{\bar{\boldsymbol{D}}_i - \hat{\mu}_i}{\hat{\delta}_i}\|_2$ to estimate $\sigma_1(\frac{\bar{\boldsymbol{D}}_i - \hat{\mu}_i(\bar{\boldsymbol{D}}_i)}{\hat{\delta}_i(\bar{\boldsymbol{D}}_i)})$, where $\boldsymbol{p}^*$ is a $1 \times I_1$ vector whose entries are independent standard normal random variables, $\hat{\delta}_i = \sqrt{\frac{1}{w-1} \sum_{j=1}^{w} (\boldsymbol{s}_j - \hat{\mu}_i)^2}$, $\hat{\mu}_i = \frac{1}{w} \sum_{j=1}^{w} \boldsymbol{s}_j$ and $\boldsymbol{s} \in \mathbb{R}^w$ is a vector sampled from $\bar{\boldsymbol{D}}_i$.

Thus, when $\bar{s}_{\max}(\frac{\bar{\boldsymbol{D}}_i - \hat{\mu}_i}{\hat{\delta}_i}) > \sqrt{I_1} + \sqrt{I_2} + h$ $(h \geq 1)$, $\kappa_i^{(t)}$ is increased to

$$\kappa_i^{(t+1)} = \min(\kappa_i^{(t)} + l, \kappa_{\max})(l > 0). \tag{6.17}$$

By performing the QR decomposition of $[\bar{\boldsymbol{Q}}_i^{(t+1)}; \boldsymbol{P} \cdot \bar{\boldsymbol{D}}_i]^T$, we have

$$[\bar{\boldsymbol{Q}}_i^{(t+1)}; \boldsymbol{P} \cdot \bar{\boldsymbol{D}}_i]^T = \tilde{\boldsymbol{Q}}_i^T \tilde{\boldsymbol{R}}_i^T,$$

where $\boldsymbol{P}$ is an $l \times I_1$ matrix whose entries are independent standard normal random variables. Thus, we augment

$$\bar{\boldsymbol{Q}}_i^{(t+1)} \leftarrow \tilde{\boldsymbol{Q}}_i \in \mathbb{C}^{\kappa_i^{(t+1)} \times I_2} \tag{6.18}$$

and

$$\bar{\boldsymbol{Z}}_i^{(t+1)} \leftarrow [\bar{\boldsymbol{Z}}_i^{(t+1)}, \mathbf{0}]\tilde{\boldsymbol{R}}_i \in \mathbb{C}^{I_1 \times \kappa_i^{(t+1)}}. \tag{6.19}$$

6.2.4.2 Decrease Strategy

Note that in the computing process $\mathcal{S}_{\frac{1}{\mu},\mathcal{G}}(\boldsymbol{\mathcal{Z}}^{(t+1)})$, the singular values of $\bar{\boldsymbol{Z}}_i^{(t+1)}$ will be obtained first. Therefore, it is assumed that $\lambda_{1,i} \geq \lambda_{2,i} \geq \cdots \geq \lambda_{\kappa_i^{(t+1)},i}$ are singular values of $\bar{\boldsymbol{Z}}_i^{(t+1)}$. Then, the quotient sequence $\tilde{\lambda}_{j,i} = \lambda_{j,i}/\lambda_{j+1,i}(j = 1, \ldots, \kappa_i^{(t+1)} - 1)$ is computed. Suppose $s_i = \arg\max_{1 \leq j < r_i} \tilde{\lambda}_{j,i}$ and $\tau_i = \frac{(\kappa_i^{(t+1)}-1)\tilde{\lambda}_{s_i,i}}{\sum_{j \neq s_i} \tilde{\lambda}_{j,i}}$, if $\tau_i \geq 10$ indicates a large drop in the magnitude of singular values [102], $\kappa_i^{(t+1)}$ should be updated as follows [8]:

$$\kappa_i^{(t+1)} = \max(\tilde{\kappa}_i, \kappa_{\min}), \tag{6.20}$$

where $\tilde{\kappa}_i$ satisfies $\sum_{j=1}^{\tilde{\kappa}_i} \lambda_{j,i} / \sum_{j=1}^{r_i} \lambda_{j,i} \geq 95\%$.

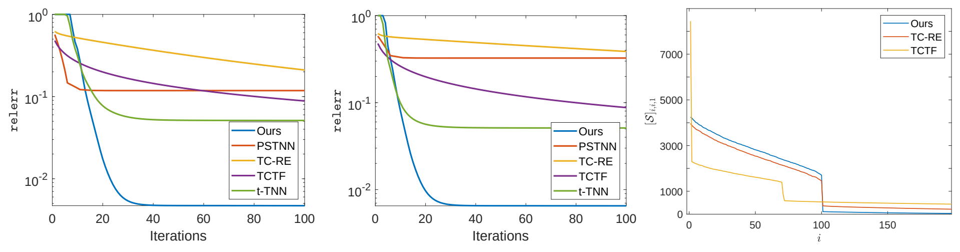
Let $\bar{\boldsymbol{Z}}_i^{(t+1)} = \boldsymbol{Q}_{\boldsymbol{Z}} \cdot \boldsymbol{R}_{\boldsymbol{Z}}$ and $\boldsymbol{R}_{\boldsymbol{Z}} = \tilde{\boldsymbol{U}} \cdot \tilde{\boldsymbol{S}} \cdot \tilde{\boldsymbol{V}}^T$ be the QR decomposition of $\bar{\boldsymbol{Z}}_i^{(t+1)}$ and the SVD of R_Z, respectively. This paper updates

$$\bar{\boldsymbol{Z}}_i^{(t+1)} \leftarrow \boldsymbol{Q}_{\boldsymbol{Z}} \cdot [\tilde{\boldsymbol{U}}]_{:,1:\kappa_i^{(t+1)}} \cdot [\tilde{\boldsymbol{S}}]_{1:\kappa_i^{(t+1)},1:\kappa_i^{(t+1)}}, \tag{6.21}$$

and

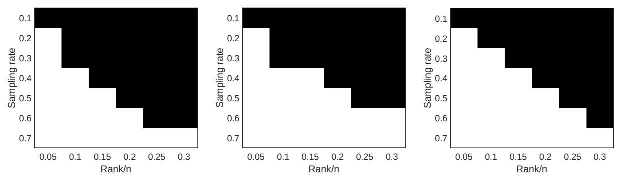
$$\bar{\boldsymbol{Q}}_i^{(t+1)} \leftarrow [\tilde{\boldsymbol{V}}]^T_{:,1:\kappa_i^{(t+1)}} \cdot \bar{\boldsymbol{Q}}_i^{(t+1)}. \tag{6.22}$$

Different from [102], in the proposed rank estimation strategy, $\kappa_i^{(t+1)}$, $\bar{\boldsymbol{Z}}_i^{(t+1)}$, and $\bar{\boldsymbol{Q}}_i^{(t+1)}$ are adjusted for each slice by considering the difference of each slice in the tensor. Based on



(a) (b) (c)

Figure 6.1: The relative error with iterations for (a) $N = 1000$ and (b) $N = 3000$, and (c) the plot of the singular values (i.e., $[\boldsymbol{\mathcal{S}}]_{i,i,1}$) of the recovered low-rank tensor by different methods for $N = 1000$, under the tubal rank $\bar{r} = 0.1 \times N$ and a sampling rate of 30%.



(a) t-TNN [51] (b) TC-RE [65] (c) TCDLR-RE

Figure 6.2: Comparison of the recovery capacity in $1000 \times 1000 \times 3$ tensor with varying tubal ranks and sampling rates. The white regions denote successful recovery with $\texttt{relerr} \leq 10^{-2}$; the black regions denote failed recovery with $\texttt{relerr} > 10^{-2}$.

Algorithm 6.2 and the proposed rank estimation method, Algorithm 6.3 (TCDLR-RE) is given.

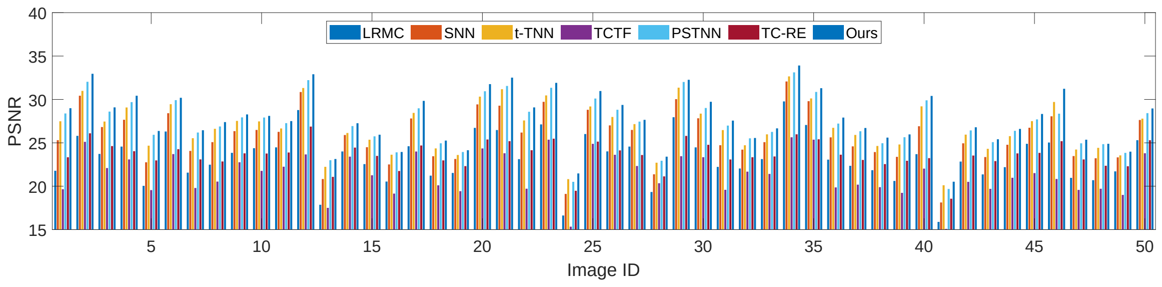
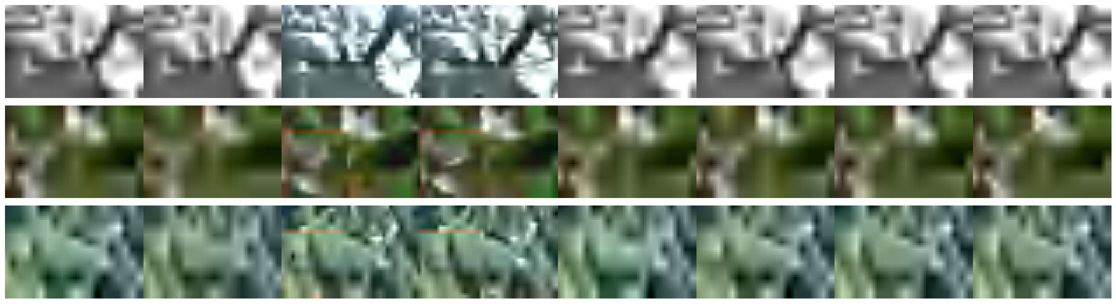


Figure 6.3: Comparison of the PSNR values on the randomly selected 50 images of the Berkeley Segmentation Dataset.



(a) Original (b) LRMC[10] (c) SNN[45] (d) t-TNN[51] (e) TCTF[102] (f) PSTNN[39] (g) TC-RE[65] (h) TCDLR-RE

Figure 6.4: Completion of the visual results on the Berkeley Segmentation Dataset with a sampling rate of 30%: (a) The original image and the results by different methods including (b) LRMC, (c) SNN, (d) t-TNN, (e) TCTF, (f) PSTNN, (g) TC-RE, and (h) TCDLR-RE, respectively.

6.3 Experiments

To verify the effectiveness and efficiency of TCDLR-RE for tensor completion, it was compared with six state-of-the-art methods, including Low-Rank Matrix Completion (LRMC)[1][10], SNN[1][45], t-TNN[1][51], TCTF[2][102], PSTNN[3] [39], and TC-RE [65] on both synthetic and real-world data. For TCDLR-RE, ℓ_p was selected as the non-convex penalty function in $\|\boldsymbol{\mathcal{X}}\|_{*,(\kappa,g)}$, and the parameters were set as $\kappa_{\max} = 0.5 \times \min(I_1, I_2)$, $\kappa_{\min} = 25$, and $p = 0.8$. For TCTF, the parameters suggested in [102] did not lead to good performance on a large dataset, so they were empirically tuned, as will be discussed later. The parameter settings of SNN, PSTNN, and TC-RE are consistent with the suggestions by the authors, and LRMC and t-TNN are free parameters. For fairness, these methods were run respectively 10 times in these experiments, and the average results were reported for each method. All the experiments were conducted on a personal computer running Windows 10 operating system and MATLAB (R2020b) (Intel Core i7-8700 3.20-GHz CPU and 16 GB memory).

6.3.1 Synthetic Experiments

All tensor product-based methods including t-TNN, TCTF, PSTNN, TC-RE, and TCDLR-RE were tested with synthetic data. The tensors of size $N \times N \times 3$ with varying $N = \{1000, 2000, 3000, 4000\}$ were considered. The low-rank tensor data $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{N \times N \times 3}$ were generated with a tensor tubal rank $\bar{r}$ by $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{M}}_1 * \boldsymbol{\mathcal{M}}_2$, where the entries of $\boldsymbol{\mathcal{M}}_1 \in$

[1] https://github.com/canyilu/LibADMM-toolbox

[2] https://panzhous.github.io/assets/code/TCTF_code.rar

[3] https://github.com/zhaoxile/Multi-dimensional-imaging-data-recovery-via-minimizing-the-partial-sum-of-tubal-nuclear-norm

$\mathbb{R}^{N\times\bar{r}\times 3}$ and $\boldsymbol{\mathcal{M}}_2 \in \mathbb{R}^{\bar{r}\times N\times 3}$ were independently sampled from an $\mathcal{N}(0,1)$. Then, $3cN^2$ elements of $\boldsymbol{\mathcal{M}}$ were sampled uniformly to construct $\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}})$, where c is the sampling rate. This paper takes the relative error (`relerr`)

$$\texttt{relerr} = \| \hat{\boldsymbol{\mathcal{X}}} - \boldsymbol{\mathcal{M}} \|_F^2 \;/\; \| \boldsymbol{\mathcal{M}} \|_F^2$$

and the running time to evaluate the effectiveness and efficiency of different algorithms, where $\hat{\boldsymbol{\mathcal{X}}}$ is the recovered tensor of $\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}})$. Following [102], the initialized rank $\kappa^{(0)} = [1.5\bar{r}, 1.5\bar{r}, 1.5\bar{r}]$ was set for TCTF and TCDLR-RE. All experimental results are presented in Table 6.2 and Figs. 6.1-6.2.

In Table 6.2, our methods (TCDLR and TCDLR-RE) are compared with other four tensor-product-based methods in terms of the running time and `relerr`, where κ in TCDLR is set to $0.5n$ according to the low tubal rank prior. The best two results for each case are shown in bold. It can be seen from Table 6.2 and Fig. 6.1 that:

(1) In all cases, TCDLR-RE and TCDLR achieve the best performance on `relerr` attributed to the proposed dual low-rank constraint and rank estimation. The superiority of TCDLR on `relerr` demonstrates its robustness to κ. Besides, as shown in Fig. 6.1 (c), for TCTF, the singular values $[\boldsymbol{\mathcal{S}}]_{i,i,1}$ $(i = 1, 2, \cdots, n)$ decrease significantly at $i = 2$ and $i = 72$, and for TC-RE and TCDLR-RE, the singular values decrease significantly at $i = \bar{r}$, which indicates that the proposed TCDLR-RE and TC-RE estimated the tensor tubal rank more accurately than TCTF. In addition, owing to introducing the proposed dual low-rank constraint, the singular values $[\boldsymbol{\mathcal{S}}]_{i,i,1}$ $(i > \bar{r})$ of TCDLR-RE are much close to zeros than those of TC-RE.

(2) Table 6.2 shows that TCDLR-RE and TCTF achieved the best performance in terms

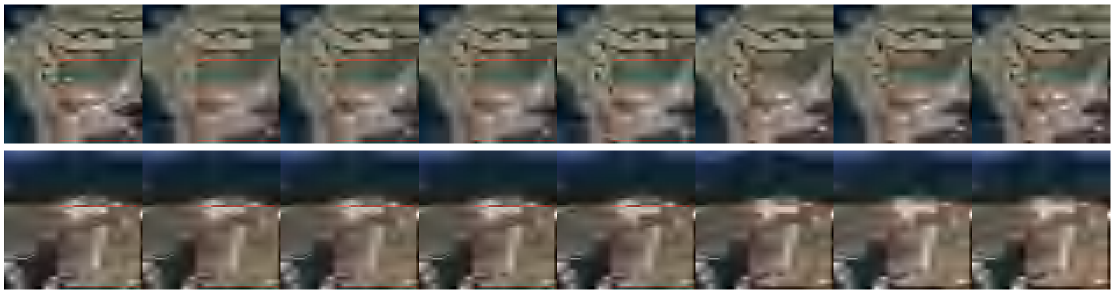
of running time, and this is because they have the same low computational complexity ($\mathcal{O}(\kappa I_1 I_2 I_3)$) for each iteration. Attributed to the proposed rank estimation strategy, TCDLR-RE performed better in running time than TCDLR. The running time of TCDLR is about two times that of TCDLR-RE. Besides, Fig. 6.1 (a)-(b) show that less running time obtained is not only because TCDLR-RE has low computational complexity for each iteration but also because it converges to the true solution $\boldsymbol{\mathcal{M}}$ with fewer iterations than other methods.

In Fig. 6.2, TCDLR-RE is compared with t-TNN and TC-RE which achieved better performance on `relerr` than other compared methods. Fig. 6.2 presents the results of `relerr` with varying c and $\bar{r}$ for fixed $N = 1000$. It was determined that a trial is successful if `relerr` $\leq 10^{-2}$, *i.e.,* the cases corresponding to the white regions were regarded as successful recovery to $\boldsymbol{\mathcal{M}}$. It can be seen from Fig 6.2 that the region of correct recovery in Fig 6.2 (c) is broader than that in Fig. 6.2 (a)-(b). These results demonstrate the effectiveness and efficiency of TCDLR-RE.

6.3.2 Real-World Applications

In this subsection, all seven tensor completion algorithms were tested in terms of image and video inpainting. $cI_1I_2I_3$ elements were sampled uniformly from the data tensor $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ to generate the observation matrices or the observation tensor $\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}})$.

Some implementation details are provided: (1) In these experiments, the sampling rate was set to $c = 0.3$. The PSNR (the peak signal-to-noise ratio), MPSNR (mean PSNR), and running time were adopted to evaluate the effectiveness and efficiency of the method. The



(a) Original (b) LRMC[10] (c) SNN [45] (d) t-TNN [51] (e) TCTF [102] (f) PSTNN[39] (g) TC-RE[65] (h) TCDLR-RE

Figure 6.5: Completion of visual results on the DOTA-v2.0 Dataset with a sampling rate of 30%: (a) The original image and the results obtained by different methods including (b) LRMC (c) SNN, (d) t-TNN, (e) TCTF, (f) PSTNN, (g) TC-RE, and (h) TCDLR-RE, respectively.

best results for each case are shown in bold. Assuming that $\hat{\boldsymbol{\mathcal{X}}}$ is the recovered tensor from $\mathbf{P}_{\Omega}(\boldsymbol{\mathcal{M}})$, the PSNR value of $\hat{\boldsymbol{\mathcal{X}}}$ is formulated as

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{I_1 I_2 I_3 \|\boldsymbol{\mathcal{M}}\|_{\infty}^2}{\|\hat{\boldsymbol{\mathcal{X}}} - \boldsymbol{\mathcal{M}}\|_F^2}\right),$$

and the mean PSNR is defined as the average PSNR results of m selected images. (2) For the matrix recovery-based method (LRMC), matrix completion was performed on each frontal slice of the observed tensor, and the results were combined to obtain the recovered tensor. (3) For TCTF, the initialized rank was set to $\kappa^{(0)} = [30, 30, 30]$ for the dataset with a small size (*i.e.*, $I_1,\ I_2 < 700$) as suggested in [102]. Since $\kappa^{(0)} = [30, 30, 30]$ did not perform well for a large dataset, this paper empirically set the initialized rank as $\kappa^{(0)} = [70, 70, 70]$ in this case. (4) The initialized rank in TCDLR-RE was set as $\kappa^{(0)} = [0.05, 0.05, \cdots, 0.05] \times \min(I_1, I_2)$.

6.3.2.1 Image Inpainting

In this part, all algorithms were tested on two color image databases of different sizes: the Berkeley Segmentation Dataset [59] and the DOTA Dataset[4][79, 18].

The MPSNR and time consumption of the seven competing inpainting algorithms on the Berkeley Segmentation Dataset are reported in Table 6.3. Fig. 6.3 compares the PSNR values of different algorithms on 50 randomly selected images. From Table 6.3 and Fig. 6.3, it can be seen that TCDLR-RE achieves the best performance in tensor recovery and took the least running time. Meanwhile, the visual quality of the seven algorithms is reported in Fig. 6.4, from which it can be seen that the visual quality of TCDLR-RE is more convincing. Specifically, the enlarged area in Fig. 6.4 indicates that our TCDLR-RE well restores the eye of the eagle, the spots of the ladybug, and the scales and fins of the fish. Compared with LRMC and TCTF, there is less visible noise in the recovered images by TCDLR-RE.

Fig. 6.5 and Table 6.4 present the experimental results of all algorithms on the DOTA-v2.0 dataset. As shown in Table 6.4, the PSNR and running time of the seven tensor completion algorithms on ten images indicate that: (1) the average PSNR value of TCDLR-RE is over 0.5 dB larger than those of the comparison methods; (2) TCDLR-RE runs much faster than other methods. Especially, the average running times of SNN, t-TNN, and PSTNN are about six times that of TCDLR-RE. Compared with TC-RE, TCDLR-RE even runs 40 times faster. In addition, the visual recovery results given in Fig. 6.5 indicate that the TCDLR-RE can retain more detail within the image data than other methods. Besides, there is more spot noise caused by the image inpainting algorithm on the recovered images by LRMC and TCTF.

[4] https://captain-whu.github.io/DOTA/index.html

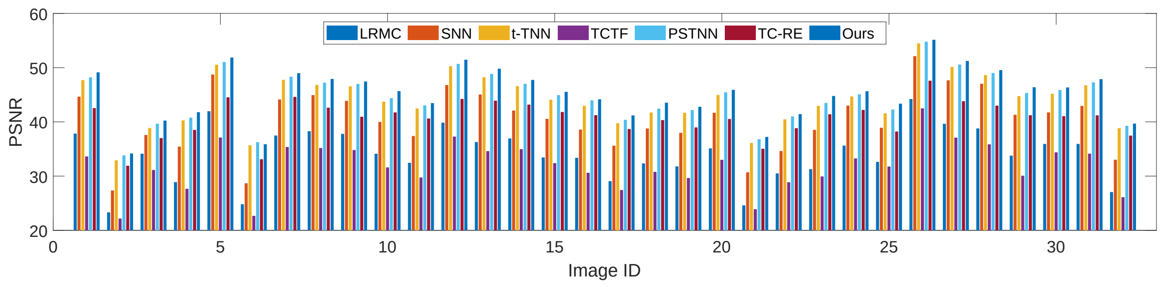


Figure 6.6: Comparison of the PSNR values of all methods on 32 HSIs from the CAVE database.

6.3.2.2 HSI Inpainting

Here, the performance of all methods was evaluated on two different hyperspectral images (HSIs) databases: the CAVE database[5] [81] and the BGU iCVL Hyperspectral Image Dataset (BiHID) [6] [1].

The experimental results on the CAVE database and the BiHID are presented in Fig. 6.6, Table 6.5 and Table 6.6, respectively. From these results, the following observations are obtained. First, TCDLR-RE achieves the best PSNR and MPRNR on both datasets. Second, on both datasets, the proposed method (TCDLR-RE) runs much faster than the comparison methods. Especially, for the BiHID, the average running time of SNN, t-TNN, and PSTNN is about six times that of TCDLR-RE, and TCDLR-RE runs even 60 times faster than TC-RE. All these results demonstrate the effectiveness and high efficiency of TCDLR-RE for HSIs.

[5] http://www.cs.columbia.edu/CAVE/databases/multispectral/

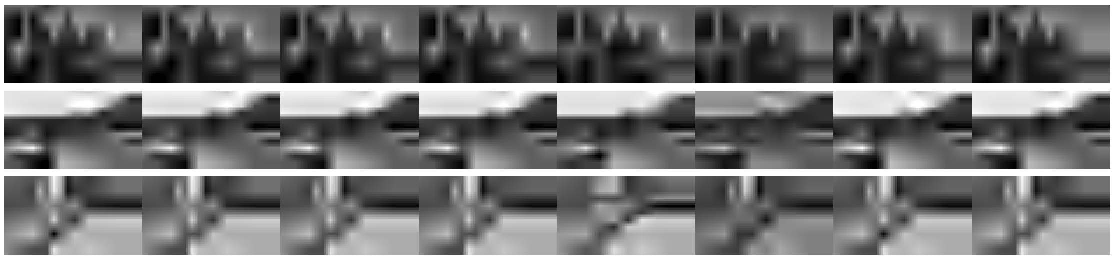
[6] http://icvl.cs.bgu.ac.il/hyperspectral/

6.3.2.3 Video Inpainting

In this part, the seven methods were tested on the first five videos in the GOT-10k video database[7][35], including "Dolphin"($1920 \times 1080 \times 100$), "City" ($1920 \times 1080 \times 80$), 'Dock'($1920 \times 1080 \times 80$), "Ship" ($1280 \times 720 \times 71$), "Handrail" ($1920 \times 1080 \times 68$), "Penguin" ($1920 \times 1080 \times 100$), "Leg" ($1280 \times 720 \times 100$), "Chicken" ($1920 \times 1080 \times 100$), "Bird" ($1920 \times 1080 \times 97$), and "Swan" ($1280 \times 720 \times 100$). The first 30 frames of each video sequence were taken and converted to the gray format. In this way, a tensor $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ was constructed for a gray video sequence with a frame size of $I_1 \times I_2$, where I_3 is the number of frames in the video sequence.

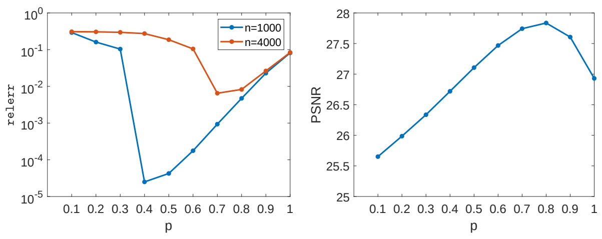
All experimental results are given in Table 6.7 and Fig. 6.7 to show the effectiveness and efficiency of TCDLR-RE. From the PSNR results given in Table 6.7, it can be seen that TCDLR-RE performs the best on video inpainting in most of the cases, and it runs the fastest. Especially, compared with TCTF, our method can achieve at least 1.5 times speed-up; compared with other tensor completion methods (including SNN, t-TNN, PSTNN, and TC-RE), our method even achieves more than 10 times speed-up. Fig. 6.7 presents the visual analysis for the three testing videos. The enlarged area of recovery results indicates that TCDLR-RE performs better in restoring the details of letters and ships. These experimental results further demonstrate the superiority of our method for large-scale data.

[7] http://got-10k.aitestunion.com/



(a) Original (b) LRMC [10] (c) SNN [45] (d) t-TNN [51] (e) TCTF [102] (f) TC-RE[65] (g) PSTNN[39] (h) TCDLR-RE

Figure 6.7: The 15th frame of the visual results in the video data. From top to bottom: “city”, “Dock”, “Handrail”.



(a) (b)

Figure 6.8: (a) The `relerr` results on the synthetic data with $N \times N \times 3$ for different p when $\text{sampling rate} = 30\%$ and $\bar{r} = 0.1N$; (b) The PSNR results on the Berkeley Segmentation Dataset for different p when $\text{sampling rate} = 30\%$.

6.3.2.4 Analysis of Parameters

In the proposed method TCDLR-RE, there are two types of parameters: the parameters in the proposed rank estimation (such as $\kappa_{\min}$ and $\kappa_{\max}$), and the parameters in the proposed tensor completion model (TCDLR), where the parameters in the proposed rank estimation can be given by the low-rank prior. Besides, according to the definition of the proposed norm, only one tunable parameter p is involved in TCDLR when $\mathcal{G}$ is ℓ_p. Therefore, in this part, experiments were conducted on both synthetic data and real-world data to investigate the influence of the parameter p. All results are presented in Fig. 6.8, from which it can be seen that our method with $p \in [0.7, 0.8]$ achieved more stable performance in tensor recovery.

6.3.3 Results Analysis

In terms of running time, TCDLR-RE runs much faster than other methods in all cases. For example, on the GOT-10k video database, TCDLR-RE runs 10 times faster than SNN, t-TNN, PSTNN, and TC-RE, three times faster than LRMC and 1.5 faster than TCTF. The reasons are analyzed as follows. (1) Owing to the proposed dual low-rank constraint, the t-SVD of a smaller tensor $\boldsymbol{\mathcal{Z}}$ is computed, and the t-SVD operation with large time consumption is avoided, thus reducing the total cost at each iteration of the algorithm from $\mathcal{O}(I_{(1)}I_{(2)}^2 I_3 + I_1 I_2 I_3 \log I_3)$ to $\mathcal{O}(\kappa I_1 I_2 I_3 + I_1 I_2 I_3 \log I_3)$, where κ is an estimation of the tensor rank. This indicates that TCDLR-RE has much lower computation complexity than traditional methods (including LRMC, SNN, t-TNN, and PSTNN) based on t-SVD for the case of data tensors with a low rank. (2) Fig. 6.1 (c) shows that the developed rank estimation method in TCDLR-RE can estimate the tensor rank accurately. The comparison of TCDLR and TCDLR-RE in Table 6.6 demonstrates the contribution of the proposed rank estimation method for reducing the running time. (3) Besides, as shown in Fig. 6.1 (a)-(b), TCDLR-RE converged to the optimal solution with fewer iterations than most of the comparison methods.

Except for TCTF, most of the tensor-based methods (including SNN, t-TNN, PSTNN, TC-RE, and TCDLR-RE) achieved a better performance than the matrix-based method (LRMC). This is because tensor-based methods can exploit the low-rank structure in the tensor data and utilize the relationship between different tensor slices well compared with the matrix-based method. Compared with tensor-based methods, TCDLR-RE achieved the best results in real-world applications. Specifically, the gap between the average PSNR values of TCDLR-RE and the other methods is about 1 dB on the BiHID. This is because (1) Compared with the traditional tensor methods, the proposed dual low-rank constraint in our methods (including TCDLR and TCDLR-RE) effectively addresses the issue of over-penalization in t-TNN-based methods and fully utilizes the low-rankness prior in tensor data, which helps to obtain the low-rank estimation more accurately. (2) Compared with the existing tensor factorization methods, including TCTF and TC-RE, the performance of TCDLR-RE is robust to the tensor rank estimation as proved by Property 6.1, Theorem 6.1 and the comparison of TCDLR with other methods in Table 6.6. (3) Compared with the estimation strategy adopted in TCTF, the proposed estimation method can estimate true tubal rank more accurately as proved by the comparison in Fig. 6.1 (c).

6.4 Summary

This work presents an efficient and effective tensor completion algorithm. First, aiming at the over-penalization issue in t-TNN-based methods and the difficulty in estimating the true tensor rank for tensor data with a small sampling rate, a novel low-rank tensor completion method with a new tensor norm is proposed, which utilizes the dual low-rank constraint. The proposed tensor norm enables the proposed methods to be more robust to inaccurate rank estimation and recover the low-rank tensor more accurately. Meanwhile, to avoid the high time consumption caused by performing the t-SVD operation on a large tensor, a trick to compute the t-SVD of a smaller tensor $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{I_1 \times \kappa \times I_3}$ instead of the original tensor of size $\mathbb{R}^{I_1 \times \kappa \times I_3}$ is used to solve TCDLR, thus reducing the total cost at each iteration to $\mathcal{O}(I_1 I_2 I_3 \log I_3 + \kappa I_1 I_2 I_3)$ from $\mathcal{O}(I_{(1)} I_{(2)}^2 I_3 + I_1 I_2 I_3 \log I_3)$ achieved by standard t-SVD-based methods. Based on this, a novel estimation method with rank-increasing and decreasing strategies is proposed for estimating the tensor rank κ. The experimental results demonstrate the high effectiveness and efficiency of the proposed methods.

Algorithm 6.3: TCDLR with the proposed rank estimation (TCDLR-RE)

```
**Input:** The tensor data $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{I_1\times I_2\times I_3}$, the observed set $\Omega$, $\rho = 1.3$, $\bar{\mu} = 10^{14}$, $\varepsilon = 10^{-9}$, $\kappa_{\min}$, $\kappa_{\max}$,
        $l = \min(I_1, I_2)/50$, and $h = 1$.
**Output:** $\boldsymbol{\mathcal{X}}^{(t+1)}$
**Initialize:** $t = 0$, $\boldsymbol{\mathcal{E}}^{(0)}$, $\boldsymbol{\mathcal{X}}^{(0)}$, $\boldsymbol{\mathcal{Y}}^{(0)}$, $\mu^{(0)}$, $\kappa_i^{(0)} \in \mathbb{N}^+$ and $\bar{\boldsymbol{Q}}_i^{(0)} \in \mathbb{C}^{I_2\times\kappa_i^{(0)}}$ for $i = 1, 2, \cdots, \lfloor\frac{I_3+1}{2}\rfloor$.
**While not converge do**
1. $\bar{\boldsymbol{C}}^{(t+1)} = \mathrm{bdiag}(\mathrm{fft}((\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) - \boldsymbol{\mathcal{E}}^{(t)} + \boldsymbol{\mathcal{X}}^{(t)} + \dfrac{\boldsymbol{\mathcal{Y}}^{(t)}}{\mu})/2, [], 3))$;
  **for** $i = 1, ..., \lfloor\frac{I_3+1}{2}\rfloor$ **do**
    2. $\bar{\boldsymbol{A}}_i^{(t+1)} = \bar{\boldsymbol{C}}_i^{(t+1)}(\bar{\boldsymbol{Q}}_i^{(t)})^*$;
    3. $\bar{\boldsymbol{B}}_i^{(t+1)} = ((\bar{\boldsymbol{A}}_i^{(t+1)})^*\bar{\boldsymbol{A}}_i^{(t+1)})^\dagger(\bar{\boldsymbol{A}}_i^{(t+1)})^*\bar{\boldsymbol{C}}_i^{(t)}$;
    4. Update $\bar{\boldsymbol{Q}}_i^{(t+1)}$ and $\bar{\boldsymbol{Z}}_i^{(t+1)}$ as follows:
       $[(\bar{\boldsymbol{Q}}_i^{(t+1)})^*, (\bar{\boldsymbol{R}}_i^{(t+1)})^*] = \mathrm{QR}((\bar{\boldsymbol{B}}_i^{(t+1)})^*)$;
       $\bar{\boldsymbol{Z}}_i^{(t+1)} = \bar{\boldsymbol{A}}_i^{(t+1)}\bar{\boldsymbol{R}}_i^{(t+1)}$;
    5. Increase $\kappa_i^{(t+1)}$ by (6.17), and adjust $\bar{\boldsymbol{Q}}_i^{(t+1)}$ and $\bar{\boldsymbol{Z}}_i^{(t+1)}$ by (6.18)
       and (6.19), respectively;
    6. $\bar{\boldsymbol{X}}_i^{(t+1)} = \mathrm{GSVT}(\bar{\boldsymbol{Z}}_i^{(t+1)})\bar{\boldsymbol{Q}}_i^{(t+1)}$, $\bar{\boldsymbol{P}}_i^{(t+1)} = \bar{\boldsymbol{Z}}_i^{(t+1)}\bar{\boldsymbol{Q}}_i^{(t+1)}$;
    7. Decrease $\kappa_i^{(t+1)}$ by (6.20), and adjust $\bar{\boldsymbol{Q}}_i^{(t+1)}$ and $\bar{\boldsymbol{Z}}_i^{(t+1)}$by (6.22)
       and (6.21), respectively;
  **end for**
  **for** $i = \lfloor\frac{I_3+1}{2}\rfloor + 1, ..., I_3$ **do**
    8. Update $\kappa_i^{(t+1)}$, $\bar{\boldsymbol{X}}_i^{(t+1)}$ and $\bar{\boldsymbol{P}}_i^{(t+1)}$ as follows:
       $\kappa_i^{(t+1)} = \kappa_{I_3-i+2}^{(t+1)}$; $\bar{\boldsymbol{X}}_i^{(t+1)} = \mathrm{Conj}(\bar{\boldsymbol{X}}_{I_3-i+2}^{(t+1)})$;
       $\bar{\boldsymbol{P}}_i^{(t+1)} = \mathrm{Conj}(\bar{\boldsymbol{P}}_{I_3-i+2}^{(t+1)})$;
  **end for**
9. Calculate $\boldsymbol{\mathcal{X}}^{(t+1)} = \mathrm{ifft}(\bar{\boldsymbol{\mathcal{X}}}^{(t+1)}, [], 3)$ and $\boldsymbol{\mathcal{P}}^{(t+1)} = \mathrm{ifft}(\bar{\boldsymbol{\mathcal{P}}}^{(t+1)}, [], 3)$;
10. Update $\boldsymbol{\mathcal{E}}^{(t+1)}$, $\boldsymbol{\mathcal{Y}}^{(t+1)}$ and $\mu^{(t+1)}$ by (6.12), (6.13) and (6.14), respectively;
11. Check the convergence condition: $\|\boldsymbol{\mathcal{P}}^{(t+1)} - \boldsymbol{\mathcal{P}}^{(t)}\|_\infty < \varepsilon$,
  $\|\boldsymbol{\mathcal{E}}^{(t+1)} - \boldsymbol{\mathcal{E}}^{(t)}\|_\infty < \varepsilon$, $\|\mathbf{P}_\Omega(\boldsymbol{\mathcal{M}}) - \boldsymbol{\mathcal{P}}^{(t)} - \boldsymbol{\mathcal{E}}^{(t)}\|_\infty < \varepsilon$
12. $t = t + 1$.
**end while**
```

Table 6.2: Comparison of `relerr` and running time (seconds) on synthetic data when the sampling rate=30% and $\bar{r} = 0.1N$.

Method	t-TNN[51]		TCTF[102]		PSTNN[39]		TC-RE[65]		TCDLR		TCDLR-RE	
Data	`relerr`	time(s)	`relerr`	time(s)	`relerr`	time(s)	`relerr`	time(s)	`relerr`	time(s)	`relerr`	time(s)
$n = 1000$	$5.24E^{-2}$	140.9	$5.29E^{-2}$	**30.4**	$1.19E^{-1}$	64.5	$5.06E^{-2}$	1338.2	$\mathbf{4.75E^{-3}}$	62.9	$\mathbf{4.75E^{-3}}$	$\mathbf{42.4}$
$n = 2000$	$5.15E^{-2}$	1591.7	$6.85E^{-2}$	**165.6**	$2.37E^{-1}$	742.9	$7.23E^{-2}$	19701.8	$\mathbf{5.28E^{-3}}$	432.0	$\mathbf{5.40E^{-3}}$	$\mathbf{228.9}$
$n = 3000$	$5.10E^{-2}$	7837.8	$6.81E^{-2}$	**481.0**	$3.24E^{-1}$	2384.2	$9.67E^{-2}$	63472.4	$\mathbf{6.25E^{-3}}$	1536.0	$\mathbf{6.52E^{-3}}$	$\mathbf{644.6}$
$n = 4000$	$5.19E^{-2}$	17870.6	$6.81E^{-2}$	**1033.2**	$3.86E^{-1}$	5543.4	$8.70E^{-2}$	131239.6	$\mathbf{7.76E^{-3}}$	3784.1	$\mathbf{8.30E^{-3}}$	$\mathbf{1706.8}$

Table 6.3: Comparison of the MPSNR and total time (seconds) on the 50 randomly selected images with a sample rate of 30% on the Berkeley Segmentation Dataset.

	LRMC[10]	SNN[45]	t-TNN[51]	TCTF[102]	PSTNN[39]	TC-RE[65]	TCDLR-RE
MPSNR	23.26	25.76	26.78	21.41	27.32	23.64	**27.83**
Average time	11.69	29.17	12.46	4.07	24.63	62.60	**3.58**

Table 6.4: Comparison of the PSNR and running time (seconds) on 10 randomly chosen images from the DOTA-v2.0 Dataset with a sampling rate of 30%.

images	LRMC[10]		SNN[45]		t-TNN[51]		TCTF[102]		PSTNN[39]		TC-RE[65]		TCDLR-RE	
	PSNR	time(s)	PSNR	time(s)	PSNR	time(s)	PSNR	time(s)	PSNR	time(s)	PSNR	time(s)	PSNR	time(s)
1	24.93	137.21	27.59	209.22	28.98	163.61	25.91	28.09	29.34	347.02	27.71	1113.40	**29.94**	**24.70**
2	26.94	128.35	29.43	198.85	30.68	161.81	27.48	28.08	30.92	320.25	29.22	1015.84	**31.75**	**24.93**
3	24.54	138.01	26.98	202.09	28.17	167.40	25.14	28.19	28.32	314.94	27.05	1111.42	**28.79**	**24.47**
4	29.19	129.75	32.56	201.79	34.15	149.11	29.18	27.75	34.83	314.63	31.04	918.00	**36.08**	**24.50**
5	24.53	139.80	27.37	211.15	29.12	184.40	24.58	27.70	29.56	314.76	27.82	1106.52	**30.17**	**24.36**
6	27.60	132.55	30.11	189.97	31.34	165.90	26.86	27.85	31.44	314.75	29.66	1024.86	**32.55**	**24.58**
7	26.00	136.96	28.23	189.99	29.20	168.21	26.47	27.45	29.41	314.46	27.98	1114.94	**29.85**	**24.71**
8	27.59	133.63	29.99	189.82	30.98	172.42	28.04	27.43	31.15	314.90	29.59	1016.91	**31.83**	**24.60**
9	24.25	128.18	26.88	193.03	28.82	165.67	25.23	27.61	29.30	314.40	27.53	1111.54	**30.10**	**24.36**
10	25.33	135.28	28.30	189.08	29.46	156.05	25.57	27.42	29.70	314.06	28.38	1073.92	**30.59**	**24.36**
Average	26.09	133.97	28.74	197.50	30.09	165.46	26.44	27.76	30.40	318.42	28.60	1060.74	**31.16**	**24.56**

Table 6.5: Comparison of the MPSNR and average time (seconds) on the CAVE dataset with a sampling rate of 30%.

	LRMC[10]	SNN[45]	t-TNN[51]	TCTF[102]	PSTNN[39]	TC-RE[65]	TCDLR-RE
MPSNR	34.04	40.36	44.00	31.86	44.57	40.69	**45.23**
Average Time	283.81	409.48	1256.16	102.28	345.62	1393.67	**71.99**

Table 6.6: Comparison of the PSNR and running time (seconds) on first 8 HSIs from the BiHID with a sampling rate of 30%.

images	LRMC[10]		SNN[45]		t-TNN[51]		TCTF[102]		PSTNN[39]		TC-RE[65]		TCDLR-RE	
	PSNR	time(s)	PSNR	time(s)	PSNR	time(s)	PSNR	time(s)	psnr	time	psnr	time	PSNR	time(s)
1	39.13	2774.19	46.75	3850.68	49.98	5370.50	36.19	695.07	50.44	5083.09	46.90	37663.53	**51.41**	**589.36**
2	36.52	2822.36	43.56	3707.55	47.45	5420.99	33.75	766.11	47.92	5080.61	40.55	38794.25	**48.75**	**585.96**
3	34.62	3211.77	43.15	3839.95	47.52	5937.54	32.68	690.66	48.03	5170.47	40.88	40202.43	**49.58**	**583.63**
4	40.22	3061.57	49.21	4044.59	51.84	5478.93	38.80	697.81	52.36	5120.01	47.43	35283.78	**53.76**	**585.62**
5	35.13	3110.88	43.27	3910.30	47.90	5867.13	32.72	700.20	48.35	5128.86	44.71	36735.30	**49.85**	**582.80**
6	42.32	3032.16	51.01	4157.88	53.18	5585.10	36.87	701.17	53.72	5127.92	47.44	32960.93	**55.26**	**581.58**
7	36.63	3098.21	45.03	4042.21	49.66	5829.48	33.90	701.07	50.17	5167.41	46.49	35307.90	**51.93**	**572.78**
8	38.42	2736.59	46.02	3807.17	47.82	5087.35	34.76	681.42	48.28	5176.80	44.33	49373.34	**49.29**	**585.03**
Average	37.87	2980.97	46.00	3920.04	49.42	5572.13	34.96	704.19	49.91	5131.90	44.84	38290.18	**51.23**	**583.35**

Table 6.7: Comparison of the PSNR and running time (seconds) on videos with a sampling rate of 30%.

videos	LRMC[10]		SNN[45]		t-TNN[51]		TCTF[102]		PSTNN[39]		TC-RE[65]		TCDLR-RE	
	PSNR	time(s)	PSNR	time(s)	PSNR	time(s)	PSNR	time(s)	psnr	time(s)	psnr	time(s)	PSNR	time(s)
Dolphin (1920 × 1080 × 30)	50.94	2074.26	**53.01**	8563.23	50.72	6134.07	45.97	728.90	50.87	5431.55	18.42	49285.94	52.37	**470.35**
City (1920 × 1080 × 30)	27.99	2158.47	27.83	7483.17	27.73	6077.03	26.29	744.79	28.01	5482.72	23.66	49053.57	**28.91**	**482.77**
Dock (1920 × 1080 × 30)	26.87	2064.83	27.22	7080.09	27.49	5922.21	25.34	660.02	27.72	5534.49	13.34	50584.37	**28.18**	**483.94**
Ship (1280 × 720 × 30)	42.09	713.79	45.46	2486.32	43.30	1186.47	34.97	327.03	43.61	1264.76	36.26	10915.48	**46.41**	**195.41**
Handrail (1920 × 1080 × 30)	35.44	2019.32	36.05	7653.02	35.34	6008.16	31.32	728.98	35.65	5378.40	16.56	48916.63	**37.40**	**487.75**
Penguin (1920 × 1080 × 30)	39.92	2062.64	42.80	7551.00	40.10	6097.18	34.83	736.58	40.38	5484.69	18.06	49622.41	**42.85**	**489.57**
Leg (1280 × 720 × 30)	34.41	680.41	37.35	2119.52	36.17	1115.00	33.94	322.13	36.49	1272.85	23.30	10310.20	**38.78**	**194.84**
Chicken (1920 × 1080 × 30)	23.89	1937.22	**25.16**	6603.46	24.41	5787.27	22.02	724.16	24.59	5390.03	15.45	49552.65	25.07	**484.86**
Bird (1920 × 1080 × 30)	27.97	2001.99	**29.19**	6640.75	28.37	5795.14	26.65	730.30	28.64	5232.23	19.24	52635.84	29.14	**487.46**
Swan (1280 × 720 × 30)	33.95	675.05	34.49	2092.67	33.82	1133.65	25.27	323.54	34.14	1252.79	28.24	10113.47	**35.63**	**193.84**
Average	34.35	1638.80	35.86	5827.32	34.74	4525.62	30.66	602.64	35.01	4172.45	21.25	38099.06	**36.47**	**397.08**

Chapter 7

Conclusions and Future Works

In this dissertation, we mainly studied TV and SPV in t-product-based methods and provided an efficient non-convex optimization for solving t-product-based tensor recovery problems. Based on our analysis for TV and SPV, we have the following conclusions:

- TV is commonly observed in t-product-based methods, as the t-product-based rank depends on the application of invertible linear transforms to only one dimension of the tensor.
- Experimental results have shown that SPV is also commonly present in t-product-based methods, mainly due to the choice of the invertible linear transform. Specifically, we have experimentally proven that Slice Permutation Invariance (SPI) does not hold for Discrete Cosine Transformation and Discrete Fourier Transformation.

Therefore, we proposed a new norm (Weighted Tensor Average Rank) for handling TV and a general solver for eliminating SPV, respectively.

In addition to addressing TV and SPV, we have also considered the issue of over-penalization caused by the tensor nuclear norm and the lack of an efficient non-convex optimization framework for t-product-based methods. To overcome these challenges, we have developed a fast non-convex optimization framework called TCDLR-RE for tensor recovery. This framework allows for the use of a wide range of surrogate functions to improve the performance of tensor recovery. Notably, TCDLR-RE can be applied to matrix completion problems since matrix completion is a special case of tensor completion. The experimental results on tensor completion and tensor robust principal component analysis have demonstrated the effectiveness of the proposed methods.

Furthermore, the proposed methods can be extended to other low-rank recovery problems, such as robust tensor completion and tensor outlier pursuit, which involve tensor data with various types of noise rather than just incomplete observations and sparse noise. Additionally, the proposed methods can be combined with non-local methods [67] to achieve better performance. However, it is important to clarify that this dissertation does not focus on providing specific image denoising or inpainting algorithms. Instead, the primary objective is to investigate an effective approach for defining the tensor rank function that better characterizes the low-rank structure in tensor data. Consequently, we have chosen not to employ any non-local strategies and have not compared the algorithms specifically designed for image denoising or inpainting, including those based on deep learning, in our experimental analysis throughout this dissertation. Besides, it is worth noting that the low-rank methods discussed in this research can be broadly applied to various computer vision problems in an unsupervised manner. Although deep learning methods have achieved impressive achievements in various applications over the years, they still face some problems, such as their interpretability and the amount of time and labor required in building and training neural networks for a specific task. Therefore, we firmly believe that research on low-rank methods continues to hold significant value.

Although we have specifically discussed TV, SPV, and the non-convex optimization framework in third-order t-product-based tensor recovery in this dissertation, the Weighted Tensor Average Rank and the general solver for eliminating SPV can be generalized to address TV and SPV in higher-order t-product-based tensor recovery by introducing more auxiliary variables. Moreover, by utilizing the higher-order tensor product defined in [58, 48], TCDLR-RE can be readily applied to higher-order tensor recovery problems. I plan to explore these generalizations in my future work. In the future, I am also interested in theoretically proving the exact recovery guarantees provided by the proposed general solver presented in Chapter 5.

Bibliography

[1] B. Arad and O. Ben-Shahar. Sparse recovery of hyperspectral signal from natural rgb images. In *European Conference on Computer Vision*, pages 19–34, 2016.

[2] S. P. Awate and R. T. Whitaker. Higher-order image statistics for unsupervised, information-theoretic, adaptive, image filtering. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 44–51. IEEE, 2005.

[3] D. Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[4] J. A. Bondy, U. S. R. Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[5] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 2, pages 60–65. Ieee, 2005.

[6] P. S. Bullen, D. S. Mitrinovic, and M. Vasic. *Means and their inequalities*, volume 31. Springer Science & Business Media, 2013.

[7] C. Cai, G. Li, H. V. Poor, and Y. Chen. Nonconvex low-rank tensor completion from noisy data. *Operations Research*, 2021.

[8] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[9] E. J. Candes and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[10] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mmathematics*, 9(6):717–772, 2009.

[11] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[12] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky. Sparse and low-rank matrix decompositions. *IFAC Proceedings Volumes*, 42(10):1493–1498, 2009.

[13] S. G. Chang, B. Yu, and M. Vetterli. Adaptive wavelet thresholding for image denoising and compression. *IEEE transactions on image processing*, 9(9):1532–1546, 2000.

[14] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, 2015.

[15] A. L. Da Cunha, J. Zhou, and M. N. Do. The nonsubsampled contourlet transform: theory, design, and applications. *IEEE transactions on image processing*, 15(10):3089–3101, 2006.

[16] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

[17] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[18] J. Ding, N. Xue, Y. Long, G.-S. Xia, and Q. Lu. Learning roi transformer for oriented object detection in aerial images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2849–2858, 2019.

[19] W. Dong, G. Shi, and X. Li. Nonlocal image restoration with bilateral variance estimation: a low-rank approach. *IEEE transactions on image processing*, 22(2):700–711, 2012.

[20] D. L. Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995.

[21] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[22] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive dct for high-

quality denoising and deblocking of grayscale and color images. *IEEE transactions on image processing*, 16(5):1395–1411, 2007.

[23] L. E. Frank and J. H. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135, 1993.

[24] J. H. Friedman. Fast sparse regression and classification. *International Journal of Forecasting*, 28(3):722–738, 2012.

[25] S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

[26] C. Gao, N. Wang, Q. Yu, and Z. Zhang. A feasible nonconvex relaxation approach to feature selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011.

[27] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):367–383, 1992.

[28] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang. Weighted nuclear norm minimization and its applications to low level vision. *International journal of computer vision*, 121(2):183–208, 2017.

[29] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2862–2869, 2014.

[30] A. Guichardet. *Symmetric Hilbert spaces and related topics: Infinitely divisible positive definite functions. Continuous products and tensor products. Gaussian and Poissonian stochastic processes*, volume 261. Springer, 2006.

[31] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[32] F. L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7(1-4):39–79, 1928.

[33] W. Hu, D. Tao, W. Zhang, Y. Xie, and Y. Yang. The twist tensor nuclear norm for video completion. *IEEE Transactions on Neural Networks and Learning Systems*, 28(12):2961–2973, 2016.

[34] B. Huang, C. Mu, D. Goldfarb, and J. Wright. Provable models for robust low-rank tensor completion. *Pacific Journal of Optimization*, 11(2):339–364, 2015.

[35] L. Huang, X. Zhao, and K. Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1562–1577, 2021.

[36] P. Jain and S. Oh. Provable tensor factorization with missing data. In *Advances in Neural Information Processing Systems*, pages 1431–1439, 2014.

[37] H. Ji, C. Liu, Z. Shen, and Y. Xu. Robust video denoising using low rank matrix completion. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1791–1798, 2010.

[38] T.-X. Jiang, T.-Z. Huang, X.-L. Zhao, and L.-J. Deng. A novel nonconvex approach to recover the low-tubal-rank tensor data: when t-svd meets pssv. *arXiv preprint arXiv:1712.05870*, 2017.

[39] T.-X. Jiang, T.-Z. Huang, X.-L. Zhao, and L.-J. Deng. Multi-dimensional imaging data recovery via minimizing the partial sum of tubal nuclear norm. *Journal of Computational and Applied Mathematics*, 372:112680, 2020.

[40] H. A. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 14(3):105–122, 2000.

[41] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover. Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. *SIAM Journal on Matrix Analysis and Applications*, 34(1):148–172, 2013.

[42] M. E. Kilmer and C. D. Martin. Factorization strategies for third-order tensors. *Linear Algebra and its Applications*, 435(3):641–658, 2011.

[43] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[44] H. Kong, X. Xie, and Z. Lin. t-schatten-p norm for low-rank tensor recovery. *IEEE Journal of Selected Topics in Signal Processing*, 12(6):1405–1419, 2018.

[45] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 35(1):208–220, 2013.

[46] R. Liu, Z. Lin, and Z. Su. Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning. In *Asian Conference on Machine Learning*, pages 116–132, 2013.

[47] X.-Y. Liu, S. Aeron, V. Aggarwal, and X. Wang. Low-tubal-rank tensor completion using alternating minimization. *IEEE Transactions on Information Theory*, 66(3):1714–1737, 2019.

[48] Y. Liu, J. Liu, Z. Long, and C. Zhu. *Tensor computation for data analysis*. Springer, 2022.

[49] Y. Liu and F. Shang. An efficient matrix factorization method for tensor completion. *IEEE Signal Processing Letters*, 20(4):307–310, 2013.

[50] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan. Tensor robust principal component analysis with a new tensor nuclear norm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):925–938, 2019.

[51] C. Lu, J. Feng, Z. Lin, and S. Yan. Exact low tubal rank tensor recovery from gaussian measurements. *arXiv preprint arXiv:1806.02511*, 2018.

[52] C. Lu, X. Peng, and Y. Wei. Low-rank tensor completion with a new tensor nuclear norm induced by invertible linear transforms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5996–6004, 2019.

[53] C. Lu, J. Tang, S. Yan, and Z. Lin. Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm. *IEEE Transactions on Image Processing*, 25(2):829–839, 2015.

[54] C. Lu, C. Zhu, C. Xu, S. Yan, and Z. Lin. Generalized singular value thresholding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[55] T. Lyche. *Numerical linear algebra and matrix factorizations*, volume 22. Springer Nature, 2020.

[56] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *2009 IEEE 12th international conference on computer vision*, pages 2272–2279. IEEE, 2009.

[57] D. Malioutov and A. Aravkin. Iterative log thresholding. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7198–7202, 2013.

[58] C. D. Martin, R. Shafer, and B. LaRue. An order-p tensor factorization with applications in imaging. *SIAM Journal on Scientific Computing*, 35(1):A474–A490, 2013.

[59] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision.*, volume 2, pages 416–423, 2001.

[60] P. Moulin and J. Liu. Analysis of multiresolution image denoising schemes using generalized gaussian and complexity priors. *IEEE transactions on Information Theory*, 45(3):909–919, 1999.

[61] T.-H. Oh, Y.-W. Tai, J.-C. Bazin, H. Kim, and I. S. Kweon. Partial sum minimization

of singular values in robust pca: Algorithm and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):744–758, 2015.

[62] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image processing*, 12(11):1338–1351, 2003.

[63] E. Schechter. *Handbook of Analysis and its Foundations*. Academic Press, 1996.

[64] F. Shang, J. Cheng, Y. Liu, Z.-Q. Luo, and Z. Lin. Bilinear factor matrix norm minimization for robust pca: Algorithms and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(9):2066–2080, 2017.

[65] Q. Shi, Y.-M. Cheung, and J. Lou. Robust tensor svd and recovery with rank estimation. *IEEE Transactions on Cybernetics*, 2021.

[66] Q. Shi, Y.-M. Cheung, Q. Zhao, and H. Lu. Feature extraction for incomplete data via low-rank tensor decomposition with feature regularization. *IEEE Transactions on Neural Networks and Learning Systems*, 30(6):1803–1817, 2018.

[67] L. Song, B. Du, L. Zhang, L. Zhang, J. Wu, and X. Li. Nonlocal patch based t-svd for image inpainting: Algorithm and error analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[68] Y. Su, X. Wu, and G. Liu. Nonconvex low tubal rank tensor minimization. *IEEE Access*, 7:170831–170843, 2019.

[69] M. Sun, L. Zhao, J. Zheng, and J. Xu. A nonlocal denoising framework based on

tensor robust principal component analysis with lp norm. In *2020 IEEE International Conference on Big Data*, pages 3333–3340, 2020.

[70] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 839–846. IEEE, 1998.

[71] J. Trzasko and A. Manduca. Highly undersampled magnetic resonance image reconstruction via homotopic ℓ_0-minimization. *IEEE Transactions on Medical Imaging*, 28(1):106–121, 2009.

[72] L. R. Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15(122-137):3, 1963.

[73] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[74] H. Wang, F. Zhang, J. Wang, T. Huang, J. Huang, and X. Liu. Generalized nonconvex approach for low-tubal-rank tensor recovery. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[75] D. Watson and G. Philip. Triangle based interpolation. *Journal of the International Association for Mathematical Geology*, 16(8):779–795, 1984.

[76] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics & Intelligent Laboratory Systems*, 2(1):37–52, 1987.

[77] J. Wright, A. Ganesh, S. R. Rao, Y. Peng, and Y. Ma. Robust principal component

analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Neural Information Processing Systems*, volume 58, pages 289–298, 2009.

[78] Y. Wu, H. Tan, Y. Li, J. Zhang, and X. Chen. A fused cp factorization method for incomplete tensors. *IEEE Transactions on Neural Networks and Learning Systems*, 30(3):751–764, 2018.

[79] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3974–3983, 2018.

[80] Q. Xie, Q. Zhao, D. Meng, and Z. Xu. Kronecker-basis-representation based tensor sparsity and its applications to tensor recovery. *IEEE transactions on pattern analysis and machine intelligence*, 40(8):1888–1902, 2017.

[81] Q. Xie, Q. Zhao, D. Meng, Z. Xu, S. Gu, W. Zuo, and L. Zhang. Multispectral images denoising by intrinsic tensor sparsity regularization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1692–1700, 2016.

[82] Y. Xie, S. Gu, Y. Liu, W. Zuo, W. Zhang, and L. Zhang. Weighted schatten p-norm minimization for image denoising and background subtraction. *IEEE transactions on image processing*, 25(10):4842–4857, 2016.

[83] H. Xu, C. Caramanis, and S. Sanghavi. Robust pca via outlier pursuit. In *Advances in Neural Information Processing Systems*, pages 2496–2504, 2010.

[84] W.-H. Xu, X.-L. Zhao, T.-Y. Ji, J.-Q. Miao, T.-H. Ma, S. Wang, and T.-Z. Huang.

Laplace function based nonconvex surrogate for low-rank tensor completion. *Signal Processing: Image Communication*, 73:62–69, 2019.

[85] 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. *Applied Mathematics and Computation*, 367:124783, 2020.

[86] F. Zhang, J. Wang, W. Wang, and C. Xu. Low-tubal-rank plus sparse tensor recovery with prior subspace information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3492–3507, 2021.

[87] T. Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11(2):1081–1107, 2010.

[88] X. Zhang. *Matrix analysis and applications*. Tsinghua University Press, 2004.

[89] X. Zhang, D. Wang, Z. Zhou, and Y. Ma. Robust low-rank tensor recovery with rectification and alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):238–255, 2019.

[90] X. Zhang, J. Zheng, D. Wang, G. Tang, Z. Zhou, and Z. Lin. Structured sparsity optimization with non-convex surrogates of $\ell_{2,0}$-norm: A unified algorithmic framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–18, 2022.

[91] X. Zhang, J. Zheng, D. Wang, and L. Zhao. Exemplar-based denoising: A unified low-rank recovery framework. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(8):2538–2549, 2019.

[92] X. Zhang, J. Zheng, Y. Yan, L. Zhao, and R. Jiang. Joint weighted tensor schatten p-norm and tensor l_p-norm minimization for image denoising. *IEEE Access*, 7:20273–20280, 2019.

[93] X. Zhang, J. Zheng, L. Zhao, Z. Zhou, and Z. Lin. Tensor recovery with weighted tensor average rank. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[94] X. Zhang, Z. Zhou, D. Wang, and Y. Ma. Hybrid singular value thresholding for tensor completion. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[95] X.-D. Zhang. *Matrix analysis and applications*. Cambridge University Press, 2017.

[96] Z. Zhang and S. Aeron. Exact tensor completion using t-SVD. *IEEE Transactions on Signal Processing*, 65(6):1511–1526, 2016.

[97] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer. Novel methods for multilinear data completion and de-noising based on tensor-svd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3842–3849, 2014.

[98] Q. Zhao, G. Zhou, L. Zhang, A. Cichocki, and S.-I. Amari. Bayesian robust tensor factorization for incomplete multiway data. *IEEE Transactions on Neural Networks and Learning Systems*, 27(4):736–748, 2015.

[99] J. Zheng, W. Wang, X. Zhang, and X. Jiang. A novel tensor factorization-based method with robustness to inaccurate rank estimation, 2023.

[100] J. Zheng, X. Zhang, W. Wang, and X. Jiang. Handling slice permutations variability in tensor recovery. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(3):3499–3507, 2022.

[101] Y.-B. Zheng, T.-Z. Huang, T.-Y. Ji, X.-L. Zhao, T.-X. Jiang, and T.-H. Ma. Low-rank tensor completion via smooth matrix factorization. *Applied Mathematical Modelling*, 70:677–695, 2019.

[102] P. Zhou, C. Lu, Z. Lin, and C. Zhang. Tensor factorization for low-rank tensor completion. *IEEE Transactions on Image Processing*, 27(3):1152–1163, 2017.

[103] Z. Zhou, X. Li, J. Wright, E. Candes, and Y. Ma. Stable principal component pursuit. In *2010 IEEE International Symposium on Information Theory*, pages 1518–1522, 2010.

[104] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.

Appendix A

A.1 Tensor Computation

A.1.1 Basic Computation

Definition A.1. *[48] (Outer Product) For vectors $\boldsymbol{a}_1 \in \mathbb{R}^{I_1}, \boldsymbol{a}_2 \in \mathbb{R}^{I_2}, \cdots, \boldsymbol{a}_h \in \mathbb{R}^{I_h}$, the outer product for vectors $\boldsymbol{a}_n \in \mathbb{R}^{I_n}, n = 1, 2, \cdots, h$ will produces a tensor as follows:*

$$\boldsymbol{\mathcal{C}} = \boldsymbol{a}_1 \circ \boldsymbol{a}_2 \circ \boldsymbol{a}_3 \circ \cdots \circ \boldsymbol{a}_h \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_h} \tag{A.1}$$

Definition A.2. *[48] (Hadamard Product) For matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ with the same size $I \times J$, the Hadamard product of them is defined as follows:*

$$\boldsymbol{C} = \boldsymbol{A} \circledast \boldsymbol{B} = \begin{pmatrix} [\boldsymbol{A}]_{1,1}[\boldsymbol{B}]_{1,1} & [\boldsymbol{A}]_{1,2}[\boldsymbol{B}]_{1,2} & \cdots & [\boldsymbol{A}]_{1,J}[\boldsymbol{B}]_{1,J} \\ [\boldsymbol{A}]_{2,1}[\boldsymbol{B}]_{2,1} & [\boldsymbol{A}]_{2,2}[\boldsymbol{B}]_{2,2} & \cdots & [\boldsymbol{A}]_{2,J}[\boldsymbol{B}]_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{A}]_{I,1}[\boldsymbol{B}]_{I,1} & [\boldsymbol{A}]_{I,2}[\boldsymbol{B}]_{I,2} & \cdots & [\boldsymbol{A}]_{I,J}[\boldsymbol{B}]_{I,J} \end{pmatrix} \tag{A.2}$$

Definition A.3. *[48] (Kronecker Product) The Kronecker product of matrices* $\boldsymbol{A} \in \mathbb{R}^{I\times J}$ *and* $\boldsymbol{B} \in \mathbb{R}^{K\times L}$ *can be defined by*

$$\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} [\boldsymbol{A}]_{1,1}\boldsymbol{B} & [\boldsymbol{A}]_{1,2}\boldsymbol{B} & \cdots & [\boldsymbol{A}]_{1,J}\boldsymbol{B} \\ [\boldsymbol{A}]_{2,1}\boldsymbol{B} & [\boldsymbol{A}]_{2,2}\boldsymbol{B} & \cdots & [\boldsymbol{A}]_{2,J}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ [\boldsymbol{A}]_{I,1}\boldsymbol{B} & [\boldsymbol{A}]_{I,2}\boldsymbol{B} & \cdots & [\boldsymbol{A}]_{I,J}\boldsymbol{B} \end{pmatrix} \tag{A.3}$$

Definition A.4. *(Tensor Inner Product) [48] For tensors* $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{C}^{I_1\times I_2\times I_3\times\cdots\times I_h}$*, the inner product is defined as*

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_h=1}^{I_h}[\boldsymbol{\mathcal{A}}]_{i_1,i_2,\cdots,i_h}\mathrm{Conj}([\boldsymbol{\mathcal{B}}]_{i_1,i_2,\cdots,i_h})$$

A.1.2 Tensor Unfolding

Definition A.5. *(Kiers Method-Based Mode-n Unfolding) [88] For an order-h tensor* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times I_3\times\cdots\times I_h}$*, the mode-n unfolding matrix of* $\boldsymbol{\mathcal{A}}$ *is defined as* $\boldsymbol{A}_{(n)} = \mathrm{unfold}_n(\boldsymbol{\mathcal{A}}) \in \mathbb{R}^{I_n\times\prod_{i\neq n} I_i}$*. The* (i_n, j)*-th element of* $\boldsymbol{A}_{(n)}$ *is defined as*

$$[\boldsymbol{A}_{(n)}]_{i_n,j} = [\boldsymbol{\mathcal{A}}]_{i_1,i_2,\cdots,i_h},$$

where $j = i_{n+1} + \sum_{p=1}^{h-2}\left[(i_{h+n-p}-1)\prod_{q=n+1}^{h+n-p-1} I_q\right]$*,* $I_{h+k} = I_k$*, and* $i_{h+k} = i_k$ *for* $k > 0$.

Definition A.6. *(Block Diag Matrix) [52] For tensors* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times I_3}$*, the block diag matrix* $\mathrm{bdiag}(\boldsymbol{\mathcal{A}})$ *of* $\boldsymbol{\mathcal{A}}$ *is defined as*

$$\mathrm{bdiag}(\boldsymbol{\mathcal{A}}) = \begin{pmatrix} [\boldsymbol{\mathcal{A}}]_{:,:,1} & & & \\ & [\boldsymbol{\mathcal{A}}]_{:,:,2} & & \\ & & \ddots & \\ & & & [\boldsymbol{\mathcal{A}}]_{:,:,I_3} \end{pmatrix} \in \mathbb{R}^{I_1 I_3\times I_2 I_3}.$$

A.1.3 Tensor Transpose

Definition A.7. *(Conjugate transpose) [50] The conjugate transpose of a tensor* $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{I_1 \times I_2 \times I_3}$ *is the tensor* $\boldsymbol{\mathcal{A}}^T \in \mathbb{C}^{I_2 \times I_1 \times I_3}$ *obtained by conjugate transposing each of the frontal slices and then reversing the order of transposed frontal slice through positions 2 to* I_3.

Definition A.8. *(Conjugate transpose induced by invertible linear transform) [52] For any invertible linear transform* $\boldsymbol{L}$ *such that*

$$\boldsymbol{L}(\boldsymbol{\mathcal{A}}) = \boldsymbol{\mathcal{A}} \times_3 \boldsymbol{L}, \tag{A.4}$$

which satisfies

$$\boldsymbol{L}^T\boldsymbol{L} = \boldsymbol{L}\boldsymbol{L}^T = \ell_{\boldsymbol{L}}. \tag{A.5}$$

Here, $\ell_{\boldsymbol{L}} > 0$ *is a constant. The conjugate transpose of a tensor* $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{I_1 \times I_2 \times I_3}$ *is the tensor* $\boldsymbol{\mathcal{A}}^{T_{\boldsymbol{L}}} \in \mathbb{C}^{I_2 \times I_1 \times I_3}$ *that satisfies* $[\boldsymbol{L}(\boldsymbol{\mathcal{A}}^{T_{\boldsymbol{L}}})]_{:,:,i_3} = [\boldsymbol{L}(\boldsymbol{\mathcal{A}})]^T_{:,:,i_3}$, $i_3 = 1, 2, \cdots, I_3$.

A.1.4 Norms

Definition A.9. *[48] (ℓ_0 Norm) Let* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_h}$, ℓ_0 *norm of* $\boldsymbol{\mathcal{A}}$ *is defined as*

$$\parallel \boldsymbol{\mathcal{A}} \parallel_0 = |\{(i_1, i_2, \cdots, i_h)||[\boldsymbol{\mathcal{A}}]_{i_1,i_2,\cdots,i_h}| > 0\}|.$$

Definition A.10. *[48] (ℓ_1-Norm) Let* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_h}$, ℓ_1*-norm of* $\boldsymbol{\mathcal{A}}$ *is defined as*

$$\parallel \boldsymbol{\mathcal{A}} \parallel_1 = \sum_{i_1,i_2,\cdots,i_h} |[\boldsymbol{\mathcal{A}}]_{i_1,i_2,\cdots,i_h}|.$$

Definition A.11. *[55] (Frobenius Norm) Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_h}$, Frobenius norm of $\boldsymbol{\mathcal{A}}$ is defined as*

$$\| \boldsymbol{\mathcal{A}} \|_F = \sqrt{\sum_{i_1, i_2, \cdots, i_h} [\boldsymbol{\mathcal{A}}]^2_{i_1, i_2, \cdots, i_h}}.$$

Definition A.12. *[55] (ℓ_2-Norm of vectors) Let $\boldsymbol{a} \in \mathbb{R}^N$, ℓ_2-norm of $\boldsymbol{a}$ is defined as*

$$\| \boldsymbol{a} \|_2 = \sqrt{\sum_n a_n^2}.$$

Definition A.13. *(Tensor tubal rank) [50] For $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the tensor tubal rank of $\boldsymbol{\mathcal{A}}$, denoted by $\mathrm{rank_t}(\boldsymbol{\mathcal{A}})$, is defined as the number of non-zero singular tubes of $\boldsymbol{\mathcal{S}}$, where $\boldsymbol{\mathcal{S}}$ is from the t-SVD of $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{S}} * \boldsymbol{\mathcal{V}}^T$. We can write $\mathrm{rank_t}(\boldsymbol{\mathcal{A}}) = |\{i | [\boldsymbol{\mathcal{S}}]_{i,i,:} \neq \mathbf{0}\}| = |\{i | [\boldsymbol{\mathcal{S}}]_{i,i,1} \neq 0\}|$. Denote $\boldsymbol{\sigma}(\mathcal{S}) = ([\boldsymbol{\mathcal{S}}]_{1,1,1}, [\boldsymbol{\mathcal{S}}]_{2,2,1}, ..., [\boldsymbol{\mathcal{S}}]_{r,r,1})^T$,in which $r = \mathrm{rank}_t(\boldsymbol{\mathcal{A}})$.*

Definition A.14. *(Tensor spectral norm) [50] For $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the tensor spectral norm is defined as $\|\boldsymbol{\mathcal{A}}\|_2 = \|\mathrm{bcirc}(\boldsymbol{\mathcal{A}})\|_2$.*

Definition A.15. *(Tensor average rank) [50] For $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the tensor average rank of $\boldsymbol{\mathcal{A}}$ is defined as $\mathrm{rank_a}(\boldsymbol{\mathcal{A}}) = \frac{1}{I_3}\mathrm{rank}(\mathrm{bcirc}(\boldsymbol{\mathcal{A}}))$.*

Definition A.16. *(Tensor average nuclear norm/ Tensor nuclear norm) [50] For $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the tensor average nuclear norm is defined as $\|\boldsymbol{\mathcal{A}}\|_{*,a} = \frac{1}{I_3}\|\mathrm{bcirc}(\boldsymbol{\mathcal{A}})\|_*$. From [50], we can know that $\|\boldsymbol{\mathcal{A}}\|_{*,a} = \|\boldsymbol{\mathcal{A}}\|_*$, where $\|\boldsymbol{\mathcal{A}}\|_*$ is the tensor nuclear norm of $\boldsymbol{\mathcal{A}}$ defined as $\|\boldsymbol{\mathcal{A}}\|_* = \|\bar{\boldsymbol{A}}\|_*$.*

A.2 Specific Tensors

Definition A.17. *(Identity tensor) [42] The tensor $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{I \times I \times I_3}$ is the tensor with the first frontal slice being the identity matrix, and other frontal slices being all zeros.*

Definition A.18. *(Orthogonal tensor) [42] A tensor* $\boldsymbol{\mathcal{Q}} \in \mathbb{C}^{I \times n \times I_3}$ *is orthogonal if it satisfies* $\boldsymbol{\mathcal{Q}}^T * \boldsymbol{\mathcal{Q}} = \boldsymbol{\mathcal{Q}} * \boldsymbol{\mathcal{Q}}^T = \boldsymbol{\mathcal{I}}$.

Definition A.19. *(f-diagonal tensor) [42] Tensor* $\boldsymbol{\mathcal{A}}$ *is called f-diagonal if each of its frontal slices is a diagonal matrix.*

Definition A.20. *(Identity tensor induced by invertible linear transform) [52] For any invertible linear transform* $\boldsymbol{L}$ *defined in* (A.4)*, if the tensor* $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{I \times I \times I_3}$ *is a tensor such that each frontal slice of* $\boldsymbol{L}(\boldsymbol{\mathcal{I}})$ *is the identity matrix,* $\boldsymbol{\mathcal{I}}$ *is called as the identity tensor induced by* $\boldsymbol{L}$*.*

Definition A.21. *(Orthogonal tensor induced by invertible linear transform) [52] For any invertible linear transform* L *defined in* (A.4)*, a tensor* $\boldsymbol{\mathcal{Q}} \in \mathbb{C}^{I \times I \times I_3}$ *is orthogonal if it satisfies* $\boldsymbol{\mathcal{Q}}^{T_L} *_{\boldsymbol{L}} \boldsymbol{\mathcal{Q}} = \boldsymbol{\mathcal{Q}} *_{\boldsymbol{L}} \boldsymbol{\mathcal{Q}}^{T_L} = \boldsymbol{\mathcal{I}}$.