Applied Mathematics and Computation 367 (2020) 124783

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Low-rank tensor train for tensor robust principal component analysis

Jing-Hua Yang[a], Xi-Le Zhao[a,*], Teng-Yu Ji[b], Tian-Hui Ma[c], Ting-Zhu Huang[a]

[a] The School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, PR China
[b] Department of Applied Mathematics, School of Science, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, PR China
[c] School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, PR China

ARTICLE INFO

Article history:
Received 22 September 2018
Revised 14 May 2019
Accepted 23 September 2019
Available online 4 October 2019

Keywords:
Tensor robust principal component analysis
Tensor train rank
High-dimensional data
Alternating direction method of multipliers

ABSTRACT

Recently, tensor train rank, defined by a well-balanced matricization scheme, has been shown the powerful capacity to capture the hidden correlations among different modes of a tensor, leading to great success in tensor completion problem. Most of the high-dimensional data in the real world are more likely to be grossly corrupted with sparse noise. In this paper, based on tensor train rank, we consider a new model for tensor robust principal component analysis which aims to recover a low-rank tensor corrupted by sparse noise. The alternating direction method of multipliers algorithm is developed to solve the proposed model. A tensor augmentation tool called ket augmentation is used to convert lower-order tensors to higher-order tensors to enhance the performance of our method. Experiments of simulated data show the superiority of the proposed method in terms of PSNR and SSIM values. Moreover, experiments of the real rain streaks removal and the real stripe noise removal also illustrate the effectiveness of the proposed method.

1. Introduction

Principal component analysis (PCA), as a classical data analysis and dimension reduction method, has been widely applied in various applications, such as computer vision [1–4], diffusion magnetic resonance imaging (MRI) [5,6], hyperspectral image recovery [7,8], and video recovery [9–13]. PCA focuses on reconstructing the low-rank component from the original data with noise corruption. According to the dimensions of the data, there are mainly two kinds of PCA methods: the matrix-based method and the tensor-based method.

Matrix-based PCA decomposes a matrix $D \in R^{n_1 \times n_2}$ into the sum of a low-rank component Z and a noise component S, i.e., $D = Z + S$. When S is small multidimensional Gaussian noise, traditional PCA [14] seeks the best rank-k estimate of Z by minimizing

$$\begin{aligned} &\arg\min_{Z} \|D - Z\|_F^2, \\ &\quad s.t. \quad \operatorname{rank}(Z) \le k. \end{aligned} \tag{1}$$

* Corresponding author.
E-mail addresses: yangjinghua110@126.com (J.-H. Yang), xlzhao122003@163.com (X.-L. Zhao), tengyu_j66@126.com (T.-Y. Ji), nkmth0307@126.com (T.-H. Ma), tingzhuhuang@126.com (T.-Z. Huang).

https://doi.org/10.1016/j.amc.2019.124783
0096-3003/© 2019 Elsevier Inc. All rights reserved.

However, traditional PCA cannot effectively handle large Gaussian noise and severe outliers that are common in practical data. Consequently, robust PCA (RPCA) [15] overcomes this shortcoming by modeling S as a sparse component, i.e.,

$$\begin{aligned} &\arg\min_{Z,S} \ \|Z\|_* + \|S\|_1, \\ &\quad s.t.\ D = Z + S, \end{aligned} \tag{2}$$

where $\|Z\|_* = \sum_r \sigma_r(Z)$ denotes the nuclear norm of Z, $\sigma_r(Z)$ $(r = 1, 2, \ldots, \min(n_1, n_2))$ is the rth singular value of Z, $\|S\|_1 = \sum_{ij} |s_{ij}|$ denotes the l_1-norm of S and s_{ij} is the (i, j)th element of S. The minimization problem (2) is motivated by the fact that nuclear norm and l_1-norm provide the tightest convex relaxation for the rank of matrix and l_0-norm, respectively.

Tensor PCA focuses on dimension reduction and analysis for high-dimensional data. In practice, we often encounter high-dimensional data, such as color images, videos, and medical data. Traditional RPCA methods process the high-dimensional data by transforming it into a matrix [16]. Such an operation seriously destroys the intrinsic tensor structure of high-dimensional data and increases the computational cost of data analysis. Recently, tensor RPCA (TRPCA) was developed based on tensor algebra [17–19]. TRPCA decomposes an lth-order tensor $\mathcal{D} \in R^{n_1\times n_2\times\cdots\times n_l}$ into the sum of a low-rank tensor $\mathcal{Z}$ and the sparse noise $\mathcal{S}$, i.e.,

$$\begin{aligned} &\arg\min_{\mathcal{Z},\mathcal{S}} \ \mathrm{rank}(\mathcal{Z}) + \lambda\|\mathcal{S}\|_0, \\ &\quad s.t.\ \mathcal{D} = \mathcal{Z} + \mathcal{S}. \end{aligned} \tag{3}$$

A central issue in TRPCA is the definition of the tensor rank. However, the definition of a tensor rank is not unique compared with the matrix rank. Two classical tensor rank definitions are CANDECOMP/PARAFA (CP) rank and Tucker rank [20]. CP rank [20] is defined as the smallest number of rank-one tensors formed by the vector outer product. However, the minimization of CP rank is NP-hard, and it is hard to establish a solvable relaxation form for it [21]. Tucker rank [20] is defined as

$$\mathrm{rank}_{tc}(\mathcal{Z}) := (\mathrm{rank}(Z_{(1)}), \mathrm{rank}(Z_{(2)}), \ldots, \mathrm{rank}(Z_{(l)})),$$

where $Z_{(i)} \in R^{n_i\times(n_1\cdots n_{i-1}n_{i+1}\cdots n_l)}$ is the mode-i matricization of $\mathcal{Z}$. In order to effectively minimize the Tucker rank, Liu et al. [22] proposed the sum of nuclear norms (SNN), $\sum_{i=1}^{l}\alpha_i\|Z_{(i)}\|_*$, as the convex surrogate of Tucker rank, where $\{\alpha_i\}_{i=1}^{l}$ are positive constants satisfying $\sum_{i=1}^{l}\alpha_i = 1$. Based on this surrogate, Huang et al. [17] proposed the following TRPCA:

$$\begin{aligned} &\arg\min_{\mathcal{Z},\mathcal{S}} \ \sum_{i=1}^{l}\alpha_i\|Z_{(i)}\|_* + \lambda\|\mathcal{S}\|_1, \\ &\quad s.t.\ \mathcal{D} = \mathcal{Z} + \mathcal{S}, \end{aligned} \tag{4}$$

where $\|\mathcal{S}\|_1$ is the sum of the absolute values of all entries in $\mathcal{S}$. However, Tucker rank cannot appropriately capture the global correlation of a tensor. The reason is that only a single mode represents the matrix row in $Z_{(i)}$ which is an unbalanced matricization scheme (one mode versus the rest) [23]. For instance, when all the modes have the same dimension ($n_1 = \cdots = n_l = n$), the dimension of $Z_{(i)}$ is $n \times n^{l-1}$. Looking at the matrix, Tucker and its convex relaxation cannot fully capture the correlation between high-dimensional data. Thus, its low-rankness does not make the optimization problem (4) efficient in addressing the rank optimization problem (3).

Recently, based on the tensor-tensor product and tensor singular value decomposition (t-SVD), the tensor tubal rank and its convex surrogate tensor nuclear norm (TNN) are proposed to characterize the informational and structural complexity of multilinear data [24,25]. For a third-order tensor $\mathcal{Z} \in R^{n_1\times n_2\times n_3}$, Lu et al. [18] applied TNN to TRPCA

$$\begin{aligned} &\arg\min_{\mathcal{Z},\mathcal{S}} \ \sum_{i=1}^{n_3}\alpha_i\|\bar{Z}^{(i)}\|_* + \lambda\|\mathcal{S}\|_1, \\ &\quad s.t.\ \mathcal{D} = \mathcal{Z} + \mathcal{S}, \end{aligned} \tag{5}$$

where $\bar{Z}^{(i)}$ is the ith frontal slice of $\bar{Z} = \mathrm{fft}(\mathcal{Z}, [\,], 3)$ and fft denotes the Fast Fourier Transform; see more details in [24,26]. By the definition of TNN, the correlations along the first and the second modes are characterized by the t-SVD while that along the third mode is encoded by the embedded circular convolution [27]. This implies that the TNN lacks a direct measure of the low-rankness of the third dimension.

More recently, the tensor train (TT) rank has become an active research topic thanks to its definition from a well-balanced matricization scheme. For an lth-order tensor $\mathcal{Z} \in R^{n_1\times n_2\times\cdots\times n_l}$, the TT rank is defined as

$$\mathrm{rank}_{tt}(\mathcal{Z}) := (\mathrm{rank}(Z_{[1]}), \mathrm{rank}(Z_{[2]}), \ldots, \mathrm{rank}(Z_{[l-1]})),$$

where $Z_{[i]} \in R^{\Pi_{k=1}^{i} n_k \times \Pi_{k=i+1}^{l} n_k}$ is the mode-$(1, 2, \ldots, k)$ matricization of $\mathcal{Z}$ (see Section 2.1). It is worth reminding that $Z_{[i]}$ is obtained by matricizing along the first k modes and the rest $l-k$ modes. Compared with Tucker rank, TT rank can complement the correlations between different modes, by providing the mean of the correlation between a few modes (rather than a single mode) and the rest of the tensor. Inspired by its desired nature, Lee and Cichocki [28] used low TT rank for the singular value decomposition (SVD) of large-scale matrices. Rauhut et al. [29] used low TT rank to achieve the steepest descent iteration of the large-scale least-squares problem. Directly minimizing the TT rank is NP-hard. Thus, TT nuclear norm (TTNN) [23], as the convex surrogate of the TT rank, is defined as $\|\mathcal{Z}\|_* = \sum_{i=1}^{l-1} \alpha_i \|Z_{[i]}\|_*$. Particularly, Bengua et al. [23] applied TTNN to the low-rank tensor completion problem with good performance.

In this paper, we incorporate the advantages of TTNN into the TRPCA problem by considering the following TTNN-based TRPCA model:

$$\begin{aligned} &\arg\min_{\mathcal{Z},\mathcal{S}} \sum_{i=1}^{l-1} \alpha_i \|Z_{[i]}\|_* + \lambda \|\mathcal{S}\|_1, \\ &\quad s.t.\ \mathcal{D} = \mathcal{Z} + \mathcal{S}, \end{aligned} \tag{6}$$

where α_i are positive weight parameters satisfying $\sum_{i=1}^{l-1} \alpha_i = 1$, λ is a positive parameter. The alternating direction method of multipliers (ADMM) algorithm is developed to solve the proposed model. Moreover, a tensor augmentation technique ket augmentation (KA) is introduced to enhance the performance of our method. Numerical experiments are conducted on synthetic data including the recovery of color images, MRI images, hyperspectral images, and color videos. It is worth mentioning that the problems of rain streaks removal of videos and stripe noise removal of hyperspectral images are also tested to prove the effectiveness of the proposed method. Extensive numerical experiments reveal the superiority of the proposed method over the compared methods.

The paper proceeds as follow. In Section 2, we introduce the corresponding notations and preliminaries. In Section 3, we apply the ADMM to solve the proposed model. In Section 4, numerical experiments are reported. Finally, we summarize this paper in Section 5.

2. Notations and preliminaries

In this section, we describe the notations and preliminaries used throughout the paper.

2.1. Notations

A tensor is a high-dimensional array and its order (or mode) is the number of its dimensions. We denote scalars as lowercase letters, i.e., z, vectors as boldface lowercase letters, i.e., $\mathbf{z}$, matrices as capital letters, i.e., Z, and tensors as calligraphic letters, i.e., $\mathcal{Z}$.

The Frobenius norm of an lth-order tensor $\mathcal{Z} \in R^{n_1 \times n_2 \times \cdots \times n_l}$ is $\|\mathcal{Z}\|_F = \sqrt{\Sigma_{n_1} \Sigma_{n_2} \cdots \Sigma_{n_l} z^2_{n_1 n_2 \cdots n_l}}$, where $z_{n_1 n_2 \cdots n_l}$ is the $(n_1, n_2, \ldots, n_l)$th element of tensor $\mathcal{Z}$.

Mode-i matricization (also known as mode-i unfolding or flattening) of a tensor $\mathcal{Z} \in R^{n_1 \times n_2 \times \cdots \times n_l}$ is the process of unfolding or reshaping the tensor into a matrix $Z_{(i)} \in R^{n_i \times (n_1 \cdots n_{i-1} n_{i+1} \cdots n_l)}$. The Tucker rank of the tensor $\mathcal{Z}$ is a vector $\mathbf{r} = (r_1, r_2, \cdots, r_l)$, where r_l is the rank of the corresponding matrix $Z_{(i)}$.

Mode-$(1, 2, \ldots, k)$ matricization of a tensor $\mathcal{Z} \in R^{n_1 \times n_2 \times \cdots n_l}$ is denoted as $Z_{[i]} \in R^{p_i \times q_i}$ $(p_i = \Pi_{k=1}^{i} n_k, q_i = \Pi_{k=i+1}^{l} n_k)$. In MATLAB, it can be implemented by the reshape function

$$Z_{[i]} = \text{reshape}_{[i]}(\mathcal{Z}, p_i, q_i). \tag{7}$$

The inverse operator of reshape is denoted as "unreshape", i.e., $\mathcal{Z} = \text{unreshape}_{[i]}(Z_{[i]})$. The TT rank is defined as the vector $\mathbf{r} = (r_1, r_2, \cdots, r_{l-1})$, where r_i is the matrix rank of $Z_{[i]}$. The detailed description of TT can be found in [30].

2.2. Ket augmentation

Ket augmentation (KA) [23] is a tensor augmentation technique that essentially represents a lower-order tensor to a higher-order one. A significant property of KA is that the augmented tensor exhibits the local data structure more clearly than the original one under the TT decomposition [31]. If the tensor is slightly correlated, its augmented version has low TT rank. Therefore, KA can fully explore the potential of TT rank-based optimization and is a useful preprocessing step for TT rank minimization.

The procedure of KA. We use KA to transform a lower-order tensor to a higher-order one by rearranging the elements of the tensor data. Given a tensor $\mathcal{Z} \in R^{m \times n \times p}$, the KA procedure involves three steps. First, we factorize $m = m_1 \times m_2 \times \cdots \times m_q$ and $n = n_1 \times n_2 \times \cdots \times n_q$, and reshape $\mathcal{Z}$ to $\mathcal{Z}_1$ of size $m_1 \times m_2 \times \cdots \times m_q \times n_1 \times n_2 \times \cdots \times n_q \times p$. Second, we permute the order of dimensions of $\mathcal{Z}_1$ to generate $\mathcal{Z}_2$ of size $m_1 \times n_1 \times m_2 \times n_2 \times \cdots \times m_q \times n_q \times p$. Third, we reshape $\mathcal{Z}_2$ to the augmented result $\tilde{\mathcal{Z}}$ of size $m_1 n_1 \times m_2 n_2 \times \cdots \times m_q n_q \times p$.

We give an example to explain the procedure of KA. Given a tensor $\mathcal{Z} \in R^{8 \times 27 \times 4}$, first, we factorize $8 = 2 \times 2 \times 2$ and $27 = 3 \times 3 \times 3$, and reshape $\mathcal{Z}$ to $\mathcal{Z}_1$ of size $2 \times 2 \times 2 \times 3 \times 3 \times 3 \times 4$. Second, we permute the order of dimensions of $\mathcal{Z}_1$ to generate $\mathcal{Z}_2$ of size $2 \times 3 \times 2 \times 3 \times 2 \times 3 \times 4$. Third, we reshape $\mathcal{Z}_2$ to the augmented result $\tilde{\mathcal{Z}}$ of size $6 \times 6 \times 6 \times 4$.

Interesting readers can refer to [23] for an extensive overview.

2.3. Framework of ADMM

In this section, we recall the general convergence result of ADMM [32]. Consider the following convex optimization problem with separable structure:

$$\begin{aligned} &\arg\min_{\mathbf{x},\mathbf{y}}\ f(\mathbf{x})+g(\mathbf{y}),\\ &s.t.\ A\mathbf{x}+B\mathbf{y}=\mathbf{b},\ \ \mathbf{x}\in X,\ \ \mathbf{y}\in Y, \end{aligned} \tag{8}$$

where f: $\mathbb{R}^m\to\mathbb{R}$ and g: $\mathbb{R}^n\to\mathbb{R}$ are closed proper convex functions, $X\subseteq\mathbb{R}^m$ and $Y\subseteq\mathbb{R}^n$ are closed convex sets, $A\in\mathbb{R}^{l\times m}$ and $B\in\mathbb{R}^{l\times n}$ are matrices, and $\mathbf{b}\in\mathbb{R}^l$ is a given vector. The augmented Lagrangian function of (8) is

$$L(\mathbf{x},\mathbf{y},\mathbf{z})=f(\mathbf{x})+g(\mathbf{y})+\langle\mathbf{z},A\mathbf{x}+B\mathbf{y}-\mathbf{b}\rangle+\frac{\beta}{2}\|A\mathbf{x}+B\mathbf{y}-\mathbf{b}\|_2^2, \tag{9}$$

where $\mathbf{z}$ is the Lagrangian multiplier and β is a penalty parameter. ADMM iterates as

$$\begin{cases} \mathbf{x}^{k+1}=\arg\min_{\mathbf{x}} f(\mathbf{x})+\langle\mathbf{z}^k,A\mathbf{x}\rangle+\frac{\beta}{2}\|A\mathbf{x}+B\mathbf{y}^k-\mathbf{b}\|_2^2,\\ \mathbf{y}^{k+1}=\arg\min_{\mathbf{y}} g(\mathbf{y})+\langle\mathbf{z}^k,B\mathbf{y}\rangle+\frac{\beta}{2}\|A\mathbf{x}^{k+1}+B\mathbf{y}-\mathbf{b}\|_2^2,\\ \mathbf{z}^{k+1}=\mathbf{z}^k+\tau\beta(A\mathbf{x}^{k+1}+B\mathbf{y}^{k+1}-\mathbf{b}), \end{cases} \tag{10}$$

where τ is the step length and the superscript k refers to the iteration index. The following theorem establishes the convergence of ADMM.

Lemma 1 (Theorem B.1 in [1]). *Assume that the solution set of (8) is nonempty and there exists $(\mathbf{x}_0,\mathbf{y}_0)\in ri(dom f\times dom g)\cap P$, where P is the constraint set in (8). Assume also that both A^TA and B^TB are positive definite.* Let *$\{(\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k)\}$ be generated from the ADMM algorithm.If the step length $\tau\in(0,(1+\sqrt{5})/2)$, then the sequence $\{(\mathbf{x}^k,\mathbf{y}^k)\}$ converges to an optimal solution to (8) and $\{\mathbf{z}^k\}$ converges to an optimal solution to the dual problem of (8). Therefore, the sequence $\{(\mathbf{x}^k,\mathbf{y}^k,\mathbf{z}^k)\}$ generated from the ADMM algorithm is convergent.*

3. The proposed algorithm

In this section, we develop ADMM [33–35] for solving the convex optimization problem (6). First, we covert (6) to the following problem by introducing auxiliary variables $U_i(i=1,2,\dots,l-1)$ and $\mathcal{Y}$:

$$\begin{aligned} &\arg\min_{\mathcal{Z},\mathcal{S}}\sum_{i=1}^{l-1}\alpha_i\|U_i\|_*+\lambda\|\mathcal{Y}\|_1,\\ &s.t.\ U_i=Z_{[i]},\mathcal{D}=\mathcal{Z}+\mathcal{S},\mathcal{Y}=\mathcal{S}. \end{aligned} \tag{11}$$

The linear constraints can be reformulated as the following matrix-vector multiplication form:

$$\begin{pmatrix} I & I\\ 0 & I\\ I & 0\\ \vdots & \\ I & 0 \end{pmatrix}\begin{pmatrix}\mathbf{z}\\ \mathbf{s}\end{pmatrix}+\begin{pmatrix} 0 & 0 & 0 & \dots & 0\\ -I & 0 & 0 & \dots & 0\\ 0 & -I & 0 & \dots & 0\\ \vdots & & & & \\ 0 & 0 & 0 & \dots & -I \end{pmatrix}\begin{pmatrix}\mathbf{y}\\ \mathbf{u}_1\\ \mathbf{u}_2\\ \vdots\\ \mathbf{u}_{l-1}\end{pmatrix}=\begin{pmatrix}\mathbf{d}\\ 0\\ 0\\ \vdots\\ 0\end{pmatrix}, \tag{12}$$

where I denotes the identify matrix, $\mathbf{z}$, $\mathbf{s}$, $\mathbf{y}$, $\{\mathbf{u}_i\}_{i=1}^{l-1}$ and $\mathbf{d}$ denote the vectorization of $\mathcal{Z}$, $\mathcal{S}$, $\mathcal{Y}$, $\{U_i\}_{i=1}^{l-1}$ and $\mathcal{D}$, respectively. We separate all the variables into two groups, $(\mathcal{Z},\ \mathcal{S})$ and $(\mathcal{Y},\ \{U_i\}_{i=1}^{l-1})$, and decompose the objective function as $f+g$ with $f=0$ and $g=\sum_{i=1}^{l-1}\alpha_i\|U_i\|_*+\lambda\|\mathcal{Y}\|_1$. Then the minimization problem (11) fits the framework of ADMM (8). The corresponding augmented Lagrangian function of (11) is given by

$$\begin{aligned} &\mathcal{L}(\mathcal{Z},\mathcal{S},\mathcal{Y},\{U_i\}_{i=1}^{l-1},\{C_i\}_{i=1}^{l-1},\mathcal{E},\mathcal{J})\\ &=\sum_{i=1}^{l-1}\left(\alpha_i\|U_i\|_*+\langle C_i,U_i-Z_{[i]}\rangle+\frac{\beta_i}{2}\|U_i-Z_{[i]}\|_F^2\right)\\ &\quad+\lambda\|\mathcal{Y}\|_1+\langle\mathcal{J},\mathcal{Y}-\mathcal{S}\rangle+\frac{\sigma}{2}\|\mathcal{Y}-\mathcal{S}\|_F^2+\langle\mathcal{E},\mathcal{D}-\mathcal{Z}-\mathcal{S}\rangle+\frac{\gamma}{2}\|\mathcal{D}-\mathcal{Z}-\mathcal{S}\|_F^2, \end{aligned} \tag{13}$$

where C_i, $\mathcal{E}$, and $\mathcal{J}$ are Lagrangian multipliers and β_i, γ, and σ are penalty parameters.

Now, we establish the convergence of the proposed algorithm. We show that our model satisfies the assumptions in Lemma 1, which implies the convergence of the proposed ADMM solver. The proof is divided into three parts. First, we show that the solution set of (6) is nonempty. It is clear that the objective function of (6), denote by $E(\mathcal{Z},\mathcal{S})$, is proper, continuous, and convex. According to the Weierstrass' theorem [36,37], it remains only to show the coercivity of $E(\mathcal{Z},\mathcal{S})$, i.e., for every sequence $\{(\mathcal{Z}^k, \mathcal{S}^k)\}$ such that $\|\mathcal{Z}^k\|_F + \|\mathcal{S}^k\|_F \to \infty$, we have $\lim_{k\to\infty} E(\mathcal{Z}^k, \mathcal{S}^k) = \infty$. We prove it by contradiction. Suppose that there exists a subsequence of $\{(\mathcal{Z}^k, \mathcal{S}^k)\}$ (also denoted as $\{(\mathcal{Z}^k, \mathcal{S}^k)\}$) that $\{E(\mathcal{Z}^k, \mathcal{S}^k)\}$ is bounded, we have that $\sum_{i=1}^{l-1} \alpha_i \|Z_{[i]}\|_*$ and $\|\mathcal{S}\|_1$ are bounded. Using the equivalence of norms we deduce that $\{\|\mathcal{Z}^k\|_F\}$ and $\{\|\mathcal{S}^k\|_F\}$ are bounded. Then $\{(\mathcal{Z}^k, \mathcal{S}^k)\}$ is a bounded sequence, which is a contradiction. So the solution set of (6) is nonempty. Second, it is clear that P in our model is an affine space, then there exists $(\mathcal{Z}_0, \mathcal{S}_0) \in \mathrm{ri}(\mathrm{dom} f \times \mathrm{dom} g) \cap P$. Third, we have that A^TA and B^TB are positive definite, since both A and B in (12) are full column rank, where A and B denote the coefficient matrices of the variables $(\mathbf{z}^T, \mathbf{s}^T)^T$ and $(\mathbf{y}^T, \mathbf{u}_1^T, \mathbf{u}_2^T, \ldots, \mathbf{u}_{l-1}^T)^T$ in (12), respectively. According to the Lemma 1, the sequence $\{\mathcal{Z}^k, \mathcal{S}^k, \mathcal{Y}^k, \{U_i\}_{i=1}^{l-1}, \{C_i\}_{i=1}^{l-1}, \mathcal{E}^k, \mathcal{J}^k\}$ generated from the proposed ADMM algorithm is convergent.

Thus, ADMM is based on the following iterative scheme:

$$\begin{cases} (\mathcal{Z}^{k+1}, \mathcal{S}^{k+1}) = \arg\min_{\mathcal{Z},\mathcal{S}} \mathcal{L}(\mathcal{Z}, \mathcal{S}, U_i^k, \mathcal{Y}^k, C_i^k, \mathcal{E}^k, \mathcal{J}^k), \\ (\mathcal{Y}^{k+1}, U_i^{k+1}) = \arg\min_{\mathcal{Y},\mathcal{U}_i} \mathcal{L}(\mathcal{Z}^{k+1}, \mathcal{S}^{k+1}, U_i, \mathcal{Y}, C_i^k, \mathcal{E}^k, \mathcal{J}^k), \\ C_i^{k+1} = C_i^k + \tau\beta_i(U_i^{k+1} - Z_{[i]}^{k+1}), \\ \mathcal{E}^{k+1} = \mathcal{E}^k + \tau\gamma(\mathcal{D}^{k+1} - \mathcal{Z}^{k+1} - \mathcal{S}^{k+1}), \\ \mathcal{J}^{k+1} = \mathcal{J}^k + \tau\sigma(\mathcal{Y}^{k+1} - \mathcal{S}^{k+1}), \end{cases} \tag{14}$$

where the superscript k refers to the iteration index. Following, we give the details of solving each subproblem.

1. $(\mathcal{Z}, \mathcal{S})$**-subproblem** The $(\mathcal{Z}, \mathcal{S})$-subproblem is a least squares problem

$$\begin{aligned}(\mathcal{Z}^{k+1}, \mathcal{S}^{k+1}) &= \arg\min_{\mathcal{Z},\mathcal{S}} \sum_{i=1}^{l-1}\left(\langle C_i^k, U_i^k - Z_{[i]}\rangle + \frac{\beta_i}{2}\|U_i^k - Z_{[i]}\|_F^2\right) + \langle \mathcal{J}^k, \mathcal{Y}^k - \mathcal{S}\rangle + \frac{\sigma}{2}\|\mathcal{Y}^k - \mathcal{S}\|_F^2 \\ &\quad + \langle \mathcal{E}^k, \mathcal{D} - \mathcal{Z} - \mathcal{S}\rangle + \frac{\gamma}{2}\|\mathcal{D} - \mathcal{Z} - \mathcal{S}\|_F^2 \\ &= \arg\min_{\mathcal{Z},\mathcal{S}} \sum_{i=1}^{l-1} \frac{\beta_i}{2}\|U_i^k - Z_{[i]} + C_i^k/\beta_i\|_F^2 + \frac{\sigma}{2}\|\mathcal{Y}^k - \mathcal{S} + \mathcal{J}^k/\sigma\|_F^2 + \frac{\gamma}{2}\|\mathcal{D} - \mathcal{Z} - \mathcal{S} + \mathcal{E}^k/\gamma\|_F^2. \end{aligned} \tag{15}$$

The objective function of (15) is represented by $F(\mathcal{Z}, \mathcal{S})$. Using the optimal condition $\partial F/\partial \mathcal{Z} = 0$ and $\partial F/\partial \mathcal{S} = 0$ [38], we have

$$\left(\sum_{i=1}^{l-1}\beta_k + \gamma\right)\mathcal{Z} + \gamma\mathcal{S} = \sum_{i=1}^{l-1}\beta_i(\text{unreshape}_{[i]}(U_i^k + C_i^k)/\beta_i) + \gamma(\mathcal{D} + \mathcal{E}^k/\gamma) \tag{16}$$

and

$$\gamma\mathcal{Z} + (\gamma + \sigma)\mathcal{S} = \gamma(\mathcal{D} + \mathcal{E}^k/\gamma) + \sigma(\mathcal{Y}^k + \mathcal{J}^k/\sigma). \tag{17}$$

Then the $\mathcal{Z}$ and $\mathcal{S}$ can be exactly obtained as following:

$$\mathcal{Z}^{k+1} = \left(\gamma\mathcal{N}^k - (\gamma + \sigma)\mathcal{M}^k\right)/\left(\gamma^2 - \left(\sum_{i=1}^{l-1}\beta_i + \gamma\right)(\gamma + \sigma)\right) \tag{18}$$

and

$$\mathcal{S}^{k+1} = \left(\gamma\mathcal{M}^k - \left(\sum_{i=1}^{l-1}\beta_i + \gamma\right)\mathcal{N}^k\right)/\left(\gamma^2 - \left(\sum_{i=1}^{l-1}\beta_i + \gamma\right)(\gamma + \sigma)\right), \tag{19}$$

where $\mathcal{M}^k = \sum_{i=1}^{l-1}\beta_i(\text{unreshape}_{[i]}(U_i^k + C_i^k)/\beta_i) + \gamma(\mathcal{D} + \mathcal{E}^k/\gamma)$ and $\mathcal{N}^k = \gamma(\mathcal{D} + \mathcal{E}^k/\gamma) + \sigma(\mathcal{Y}^k + \mathcal{J}^k/\sigma)$. The computational complexities of updating the variables $\mathcal{Z}$ and $\mathcal{S}$ are $O(\Pi_{i=1}^{l} n_i)$.

The variables $\mathcal{Y}$ and $\mathcal{U}_i$ are decoupled with each other, so they can be solved separately.

2. $\mathcal{Y}$**-subproblem**The $\mathcal{Y}$-subproblem is

$$\begin{aligned}\mathcal{Y}^{k+1} &= \arg\min_{\mathcal{Y}} \lambda\|\mathcal{Y}\|_1 + \langle \mathcal{J}^k, \mathcal{Y} - \mathcal{S}^{k+1}\rangle + \frac{\sigma}{2}\|\mathcal{Y} - \mathcal{S}^{k+1}\|_F^2 \\ &= \arg\min_{\mathcal{Y}} \lambda\|\mathcal{Y}\|_1 + \frac{\sigma}{2}\|\mathcal{Y} - \mathcal{S}^{k+1} + \mathcal{J}^k/\sigma\|_F^2. \end{aligned} \tag{20}$$

It has the following closed-form solution by the soft shrinkage operator [39]:

$$\mathcal{Y}^{k+1} = \max(|\mathcal{S}^{k+1} - \mathcal{J}^k/\sigma| - \frac{\lambda}{\sigma}, 0) \circ \frac{\mathcal{S}^{k+1} - \mathcal{J}^k/\sigma}{|\mathcal{S}^{k+1} - \mathcal{J}^k/\sigma|}, \tag{21}$$

where $\circ$ denotes the Hadamard product and the division is performed component-wise. The convention $0 \circ \frac{0}{0} = 0$ is assumed. The complexities of computing $\mathcal{Y}$ is $O(\Pi_{i=1}^{l} n_i)$.

3. U_i-subproblem The U_i-subproblem is

$$\begin{aligned} U_i^{k+1} &= \arg\min_{U_i} \sum_{i=1}^{l-1} \alpha_i \|U_i\|_* + \langle C_i^k, U_i - Z_{[i]}^{k+1}\rangle + \frac{\beta_i}{2}\|U_i - Z_{[i]}^{k+1}\|_F^2 \\ &= \arg\min_{U_i} \sum_{i=1}^{l-1} \left(\alpha_i \|U_i\|_* + \frac{\beta_i}{2}\|U_i - Z_{[i]}^{k+1} + C_i^k/\beta_i\|_F^2 \right). \end{aligned} \tag{22}$$

Since the U_i-subproblem can be decomposed into $l-1$ independent subproblems for U_i, it can be solved in parallel as

$$\arg\min_{U_i} \alpha_i \|U_i\|_* + \frac{\beta_i}{2}\|U_i - Z_{[i]}^{k+1} + C_i^k/\beta_i\|_F^2, \tag{23}$$

which has the closed-form solution [40]

$$U_i^{k+1} = U\Sigma_{\alpha_i/\beta_i}V^T, \tag{24}$$

where $Z_{[i]}^{k+1} - C_i^k/\beta_i = U\Sigma V^T$, $\Sigma_{\alpha_i/\beta_i} = \mathrm{diag}(\max(\Sigma_{r,r} - \alpha_i/\beta_i, 0))$, and $\Sigma_{r,r}$ is the rth singular value of Σ. Its complexity is $O(\Sigma_{i=1}^{l-1}\min(p_i^2 q_i, p_i q_i^2))$ ($p_i = \Pi_{k=1}^{i} n_k, q_i = \Pi_{k=i+1}^{l} n_k$) operations. Finally, we summarize the proposed algorithm in Algorithm 1. The total cost of computing all the variables at each iteration is $O(\Pi_{i=1}^{l} n_i + \Sigma_{i=1}^{l-1}\min(p_i^2 q_i, p_i q_i^2))$.

Algorithm 1 ADMM for solving (6).

```
Input: the observed tensor $\mathcal{D}$, parameters $\lambda$, $f$, $\gamma$, and $\sigma$.
Output: the restored tensor $\mathcal{Z}$ and sparse noise tensor $\mathcal{S}$.
1: Initialize $\mathcal{Z} = \mathcal{D}$, $U_i$, $\mathcal{Y}$, $C_i$, $\mathcal{E}$, $\mathcal{J}$, maximum iterations $K = 200$, and $\tau = 1.1$.
2: While $\|\mathcal{Z}^{k+1} - \mathcal{Z}^k\|_F/\|\mathcal{Z}^k\|_F > \varepsilon$ and $k \le K$ Do
3: Updating $\mathcal{Z}$ and $\mathcal{S}$ via (18) and (19);
4: Updating $\mathcal{Y}$ via (21);
5: Updating $U_i$ via (24);
6: Updating Multipliers $C_i$, $\mathcal{E}$, and $\mathcal{J}$ via (14);
7: End Do
```

4. Numerical experiments

In this section, we evaluate the performance of the proposed TTNN-based method (denoted as "TTNN") for restoring observed high-dimensional images as simulation experiments including color images, MRI images, hyperspectral images, and color videos. We also test the real world video rain streaks and hyperspectral image stripes removal problems. We compare the results with two TRPCA methods, including the method based on Tucker rank [41] (denoted as "SNN") and the method based on tensor tubal rank [18] (denoted as "TNN"). All test tensors are normalized between [0, 1] to allow a fair quantitative evaluation.

The quality of recovered tensors is measured by the peak signal-to-noise ratio (PSNR) [42] and the structural similarity index (SSIM) [43], which are defined as

$$\mathrm{PSNR} = 10\log_{10}\frac{N(Z_{\max})^2}{\|Z - Z^*\|_F^2}$$

and

$$\mathrm{SSIM} = \frac{(2\mu_Z\mu_{Z^*})(2\sigma_{ZZ^*} + C_2)}{(\mu_Z^2 + \mu_{Z^*}^2 + C_1)(\sigma_Z^2 + \sigma_{Z^*}^2 + C_2)},$$

where Z^* is one band (frame) of the true tensor, Z is one band (frame) of the recovered tensor, N denotes the total number of pixels in the image, $Z_{\max}$ is the maximum pixel value of the image, μ_Z and μ_{Z^*} are the mean values of images Z and Z^*, σ_Z and σ_{Z^*} are the standard variances of Z and Z^*, σ_{ZZ^*} is the covariance of Z and Z^*, and C_1 and $C_2 > 0$ are constants. The PSNR (dB) and SSIM values for a higher-order tensor are obtained by calculating average PSNR and SSIM values for all bands (frames). Higher PSNR and SSIM values imply better image quality.

The convergence criterion of our proposed algorithm is defined by computing the relative error of the tensor $\mathcal{Z}$ between two successive iterations as follows:

$$\frac{\|\mathcal{Z}^{k+1} - \mathcal{Z}^k\|_F}{\|\mathcal{Z}^k\|_F} \le 10^{-4}. \tag{25}$$

In our experiment, the weights α_i are defined as

$$\alpha_i = \frac{\delta_i}{\sum_{i=1}^{l-1}\delta_i} \quad \text{with} \quad \delta_i = \min(\Pi_{k=1}^{i} n_k, \Pi_{k=i+1}^{l} n_k), \tag{26}$$

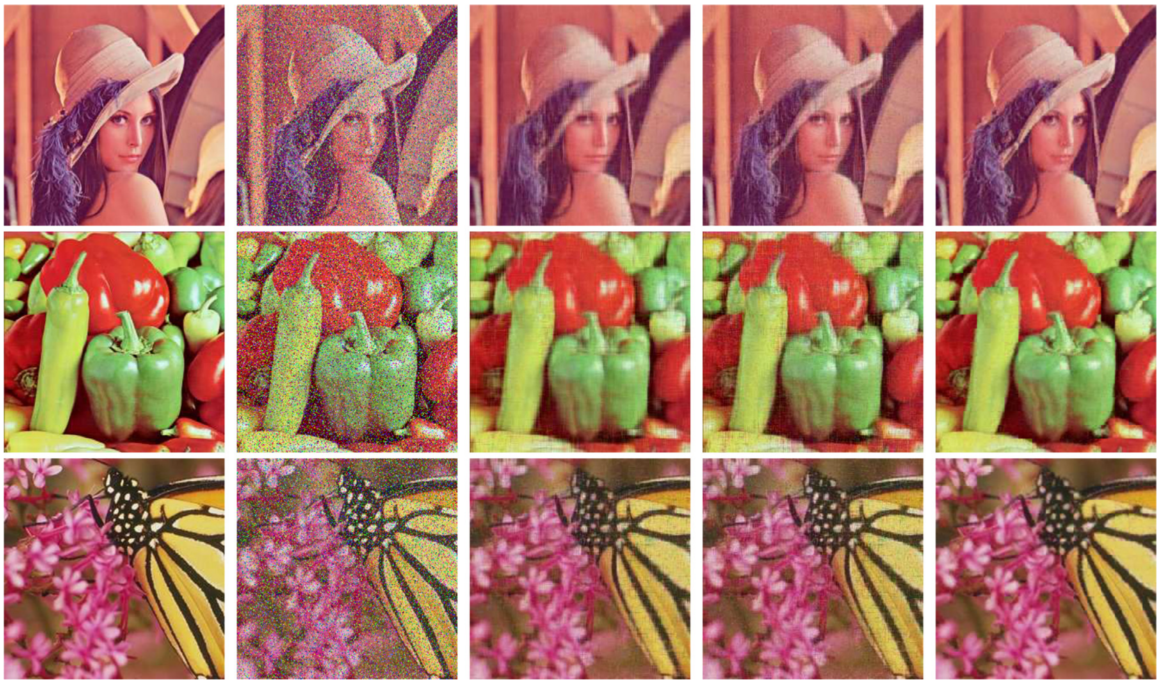

Fig. 1. Restored results of color images with the noise level 30%. From top to bottom: Lena, Peppers, and Monarch. From left to right: the original data, the observed data, the recovered results by SNN, TNN, and TTNN, respectively.

where $i = 1, \ldots, l-1$. We choose $\beta_i = f\alpha_i$, where f is empirically chosen from one of the following values in {0.1, 1, 1.5, 2}. In addition, we empirically set the regularization parameter $\lambda \in [0.01, 0.1]$ with increment of 0.01 and penalty parameters $\gamma \in [0.001, 0.003]$ and $\sigma \in [0.001, 0.01]$ with increment of 0.001. For the compared method SNN, we empirically set regularization parameters α_1, $\alpha_2 \in [5, 30]$ with increment of 2 and $\alpha_3 \in [0.2, 6]$ with increment of 0.4 and select the penalty parameter from the set $\{10^{-5}, 10^{-4}, 10^{-3}\}$. The parameter in TNN is optimized according to the author's suggestion in [18]. For synthetic data, we optimize parameters of each method to attain the highest PSNR value in all experiments. For real data, we choose the parameters to get a good visual quality.

All numerical experiments are performed on Windows 10 64-bit and MATLAB R2012a running on a desktop equipped with an Intel(R) Core(TM) i7-6700M CPU with 3.40 GHz and 8 GB of RAM.

4.1. Synthetic data

In the simulated experiments, taking the noise level 30% as an example, the observed image is obtained by randomly setting 30% of the pixels to random values [0, 1], and the positions of the corrupted pixels are unknown.

4.1.1. Color images

In this section, we evaluate the proposed method on color images. The size of the test data is $256 \times 256 \times 3$. In the low-rank term of (6), the third-order tensor $\mathcal{Z} \in R^{256\times256\times3}$ is transformed into a ninth-order $\tilde{\mathcal{Z}} \in R^{4\times4\times4\times4\times4\times4\times4\times4\times3}$ to explore the TT low-rankness by using KA. Fig. 1 shows the experiment results by SNN, TNN, and the proposed TTNN. It is clear that the restored results by the proposed method are visually better than those by SNN and TNN. Table 1 summarizes the recovered quantitative results of different noise levels for SNN, TNN, and TTNN, respectively. From this table, one can observe that our method obtains the best results with respect to PSNR and SSIM.

4.1.2. MRI images

We use MRI data of size $256 \times 256 \times 10$ as the test data in this subsection. The third-order tensor is converted to a ninth-order tensor of size $4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 10$ by using KA. Fig. 2 shows a band of the test MRI image recovered by all compared methods. Clearly, the restored results obtained by the TTNN are visually better than those obtained by SNN and TNN. Fig. 3 shows the PSNR and SSIM values of every frame. Note that every frame recovered by the proposed method is higher than that recovered by SNN and TNN. Table 2 shows the average PSNR and SSIM values of the MRI image with different sparse noise levels, and they are consistent with the visual comparison.

Table 1
The average PSNR (dB) and SSIM values obtained by SNN, TNN, and TTNN for color images with different noise levels.

Image	Method	Sparse Noise Levels					
		10%		30%		50%	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Lena	Observed	18.73	0.3636	13.98	0.1495	11.72	0.0809
	SNN	30.57	0.8890	25.37	0.7644	20.43	0.5234
	TNN	29.87	0.9288	25.35	0.6892	17.51	0.2261
	TTNN	**34.98**	**0.9627**	**28.20**	**0.8457**	**23.13**	**0.7002**
Peppers	Observed	17.93	0.3575	13.19	0.1528	10.93	0.0863
	SNN	29.24	0.8319	23.00	0.7014	17.79	0.4144
	TNN	26.98	0.8936	22.22	0.5684	15.19	0.1794
	TTNN	**30.86**	**0.9174**	**24.61**	**0.7762**	**20.08**	**0.5923**
Pallon	Observed	18.48	0.2705	13.68	0.0901	11.44	0.0473
	SNN	31.86	0.8542	27.81	0.8188	21.95	0.5950
	TNN	31.14	0.9319	27.02	0.7164	17.73	0.1743
	TTNN	**36.19**	**0.9577**	**30.27**	**0.8711**	**23.64**	**0.6834**
Carnev	Observed	15.62	0.2294	10.83	0.0823	8.59	0.0431
	SNN	29.16	0.7617	24.09	0.5688	19.36	0.2082
	TNN	26.67	0.8521	23.84	0.4336	14.09	0.1027
	TTNN	**31.90**	**0.9124**	**26.28**	**0.6537**	**20.21**	**0.2435**
Cat	Observed	18.13	0.4235	13.34	0.1713	11.15	0.0905
	SNN	31.47	0.9010	25.37	0.7346	19.48	0.4206
	TNN	32.58	0.9534	25.25	0.7098	16.79	0.2475
	TTNN	**35.70**	**0.9712**	**28.42**	**0.8607**	**22.30**	**0.6096**
Monarch	Observed	18.31	0.4546	13.60	0.2141	11.34	0.1218
	SNN	29.86	0.9127	22.44	0.7084	17.48	0.3980
	TNN	28.79	0.9432	22.28	0.6408	15.75	0.2546
	TTNN	**32.97**	**0.9605**	**24.76**	**0.8676**	**20.01**	**0.6055**
Lochness	Observed	17.29	0.3458	12.49	0.1336	10.30	0.0685
	SNN	30.56	0.8452	25.90	0.6914	18.50	0.3979
	TNN	30.92	0.8936	25.55	0.6321	15.62	0.1928
	TTNN	**33.66**	**0.9372**	**27.81**	**0.7776**	**20.41**	**0.5442**

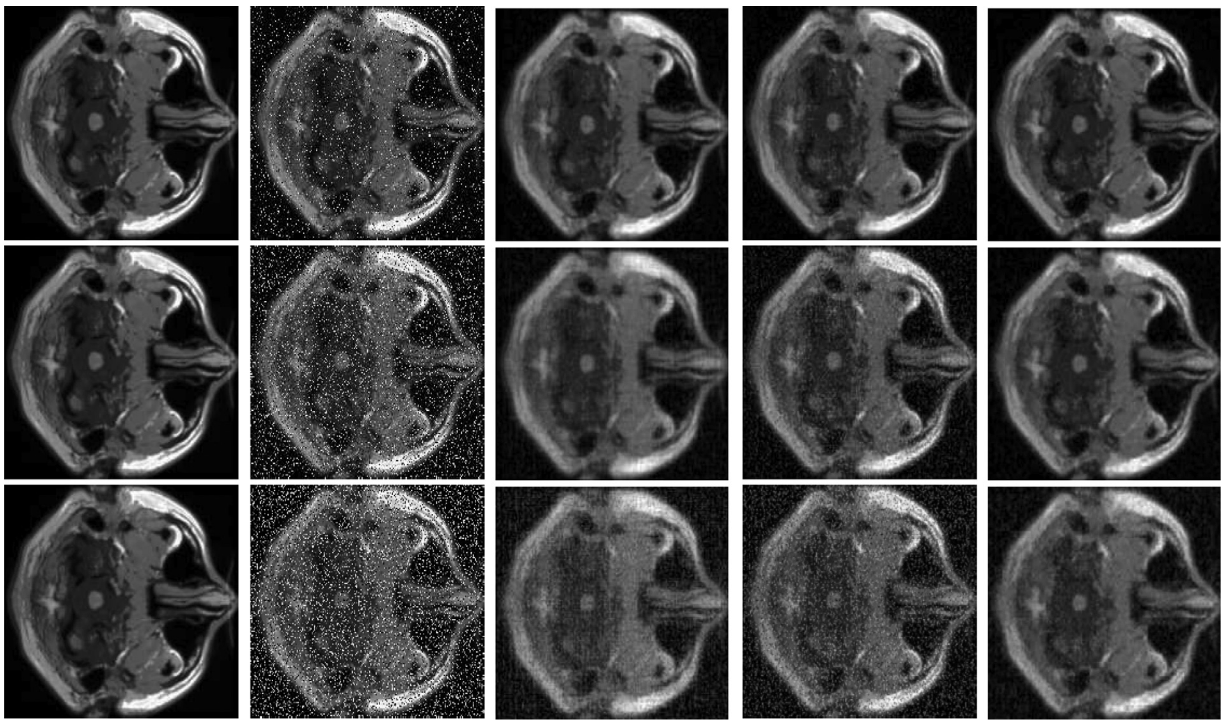

Fig. 2. Restored results of MRI image with the noise levels 10%, 20%, and 30%, respectively. From top to down: the noise level 10%, 20%, and 30%. From left to right: the original data, the observed data, the recovered results by SNN, TNN, and TTNN, respectively.

Table 2
The average PSNR (dB) and SSIM values obtained by SNN, TNN, and TTNN for MRI image with different noise levels.

Method	Sparse Noise Levels					
	10%		20%		30%	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Observed	17.21	0.3820	14.23	0.2366	12.42	0.1631
SNN	29.06	0.8607	24.32	0.6990	21.10	0.5231
TNN	30.99	0.8572	24.49	0.6110	19.78	0.4269
TTNN	**32.53**	**0.9314**	**28.23**	**0.8231**	**24.08**	**0.6618**

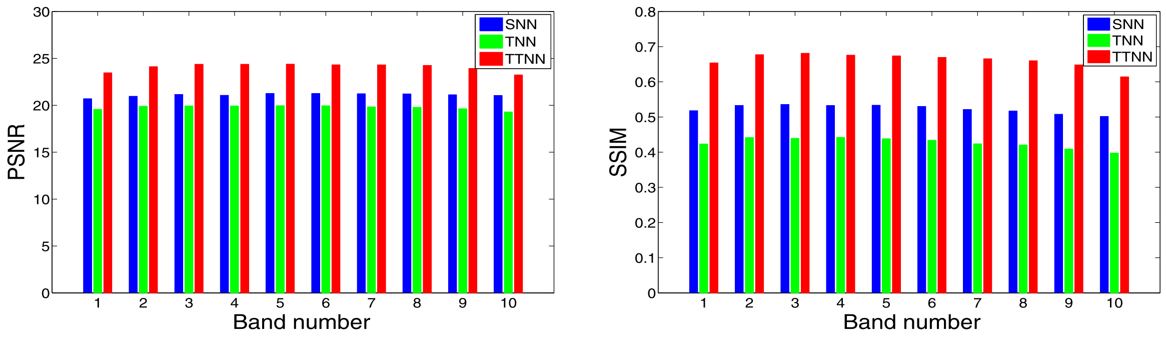

Fig. 3. The PSNR and SSIM values of all bands of the recovered MRI image (the noise level 30%) by SNN, TNN, and TTNN, respectively.

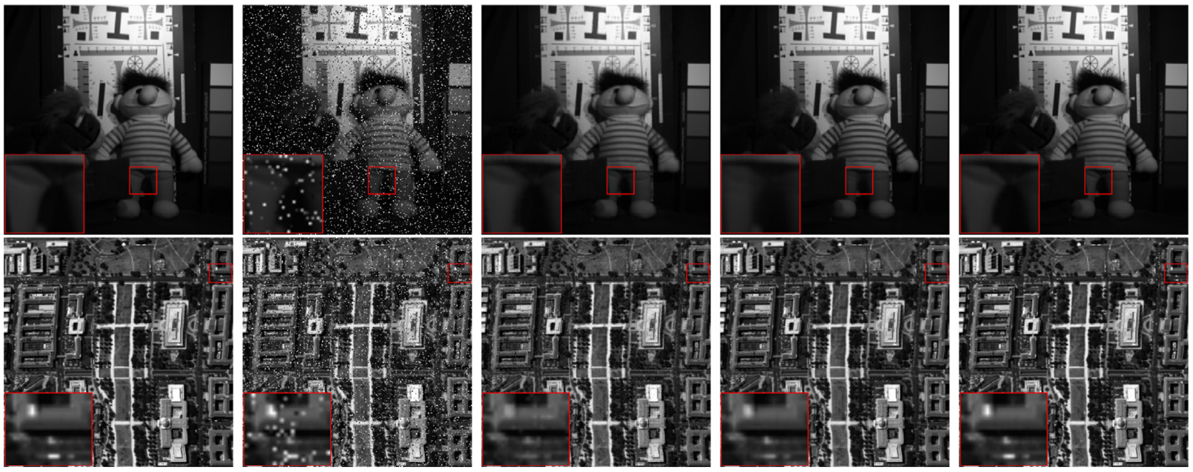

Fig. 4. Restored results of hyperspectral images with the noise level 10%. From left to right: the original data, the observed data, the recovered results by SNN, TNN, and TTNN, respectively.

4.1.3. *Multispectral images*

In this subsection, the *Toy*[1] and *WashtonDC*[2] data are used to test the performance of different methods. We only select a part of them (of size $256 \times 256 \times 10$) as the testing multispectral images. Fig. 4 shows a band of the test hyperspectral images reconstructed by the proposed method and two compared method. It is observed that the proposed method is able to produce visually superior results than the compared methods. The PSNR and SSIM values of each band of the reconstructed multispectral images for the noise level 10% are shown in Fig. 5. We can see that the PSNR and SSIM values in all bands obtained by the proposed method are better than those obtained by the compared methods. In addition, Table 3 shows numerical comparisons of different methods for recovering multispectral images with different sparse noise levels. In contrast, our advantages are more obvious when the noise level is lower.

[1] http://www1.cs.columbia.edu/CAVE/databases/multispectral.
[2] https://engineering.purdue.edu/biehl/MultiSpec/hyperspectral.html.

Table 3
The average PSNR (dB) and SSIM values obtained by SNN, TNN, and TTNN for multispectral images with different noise levels.

Image	Method	Sparse Noise Levels					
		10%		20%		30%	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Toy	Observed	16.00	0.2496	12.99	0.1490	11.20	0.1042
	SNN	36.41	0.9623	31.68	0.8635	28.01	0.8687
	TNN	41.66	0.9953	39.30	0.9912	35.78	0.9572
	TTNN	**50.33**	**0.9987**	**41.71**	**0.9951**	**36.27**	**0.9848**
WashtonDC	Observed	17.95	0.6130	14.91	0.4363	13.15	0.3288
	SNN	28.01	0.9382	24.21	0.8503	21.57	0.7545
	TNN	35.54	0.9919	32.84	0.9843	29.06	0.9559
	TTNN	**44.58**	**0.9986**	**36.31**	**0.9939**	**29.98**	**0.9742**

Table 4
The average PSNR (dB) and SSIM values obtained by SNN, TNN, and TTNN for color videos with different noise levels.

Video	Method	Sparse Noise Levels					
		10%		20%		30%	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bus	Observed	18.00	0.489	15.01	0.3362	13.24	0.2499
	SNN	25.86	0.9405	24.27	0.8654	22.05	0.6664
	TNN	24.16	0.9028	22.81	0.8419	22.03	0.7592
	TTNN	**31.81**	**0.9702**	**27.14**	**0.9322**	**23.95**	**0.8112**
Mobile	Observed	17.37	0.5785	14.41	0.4315	12.58	0.3333
	SNN	22.55	0.8423	20.60	0.6900	18.01	0.5244
	TNN	19.84	0.7659	18.56	0.6711	17.63	0.5527
	TTNN	**27.59**	**0.9244**	**23.81**	**0.8516**	**21.05**	**0.7283**
News	Observed	17.19	0.3398	14.23	0.2197	12.46	0.1577
	SNN	28.26	0.9250	26.69	0.9018	24.64	0.7944
	TNN	25.94	0.9095	24.65	0.881	23.44	0.8427
	TTNN	**38.25**	**0.9901**	**34.52**	**0.9810**	**29.71**	**0.9530**

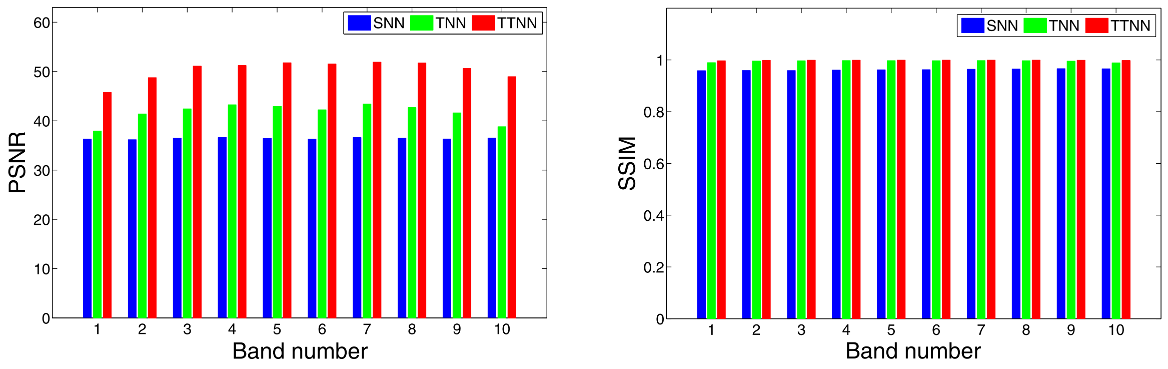

Fig. 5. The PSNR and SSIM values of all bands of the recovered multispectral image *Toy* (the noise level 10%) by SNN, TNN, and TTNN, respectively.

4.1.4. Color videos

In this section, we test the proposed method on three color videos, including *bus, mobile*, and *news*.[3] The size of all test videos is $243 \times 256 \times 3 \times 27$. We reshape the tensor to a ninth-order tensor of size $6 \times 6 \times 6 \times 6 \times 6 \times 6 \times 6 \times 6 \times 3$ for experiments.

In Fig. 6, we illustrate the restored results of one frame of *bus, mobile*, and *news* by SNN, TNN and TTNN, with the noise level 30%. It is obvious that our method visually outperforms SNN and TNN in preserving details and structure of underlying videos. The PSNR and SSIM values against the frame number are plotted in Fig. 7. It is clear that the proposed method obtains higher quality results in all frames. Table 4 summarizes the average PSNR and SSIM values of all videos

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

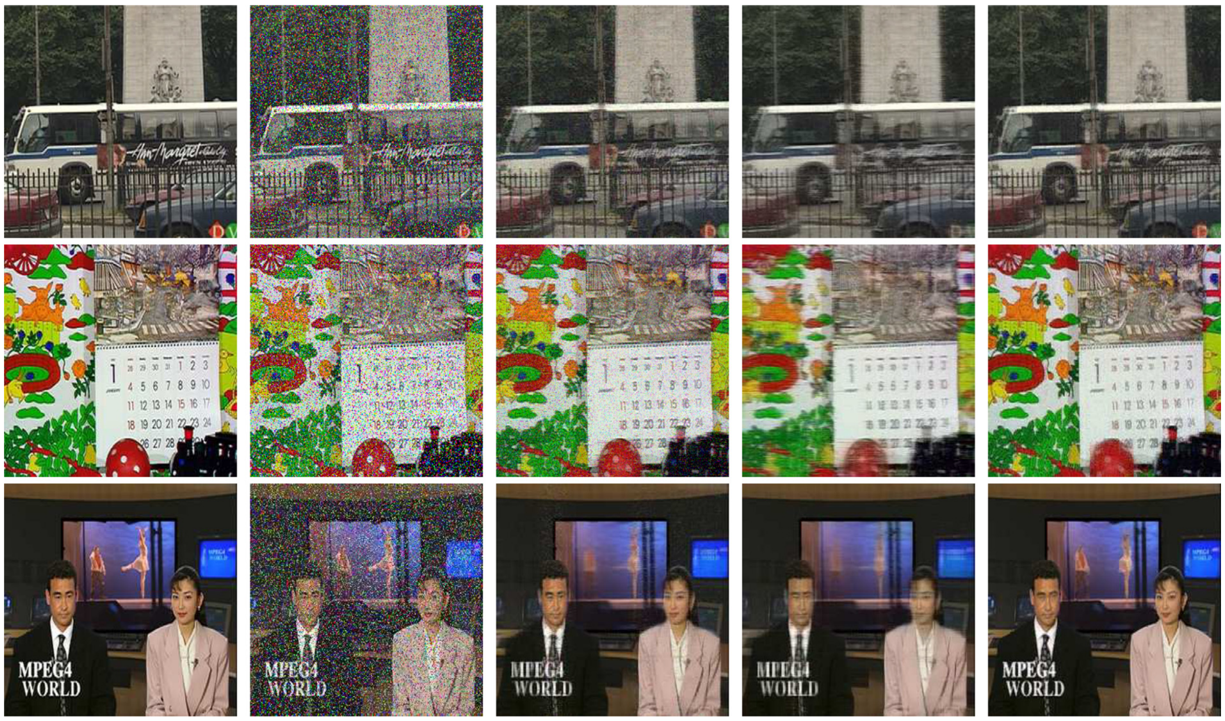

Fig. 6. One frame of the test videos recovered by different methods with the noise level 30%. From up to down: *bus*, *mobile*, and *news*. From left to right: the original data, the observed data, the recovered results by SNN, TNN, and TTNN, respectively.

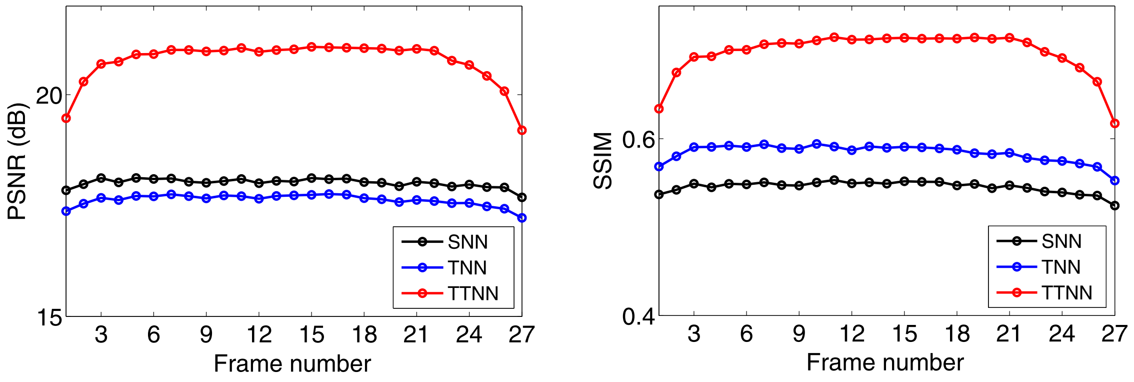

Fig. 7. The PSNR and SSIM values of all frames of the recovered video *mobile* (the noise level 30%) by SNN, TNN, and TTNN, respectively.

recovered by different methods for different noise levels. We observe that the proposed method consistently outperforms the compared methods in terms of PSNR and SSIM values.

4.2. Real data

In the previous section, we simulated random sparse noise. However, in the real world, sparse noise does not obey the above discussion, such as rain streaks in videos and stripe noise in hyperspectral images. So in this section, we test the effectiveness of the proposed method for the real-world data.

4.2.1. Video rain streaks removal

The real video is recorded by the authors of [10] on a rainy day. The size of the real video is $243 \times 256 \times 3 \times 27$. Fig. 8 shows three frames of the rain streaks removal results. Qualitatively, our method shows the best visual performance on simultaneously removing rain and preserving details. We can see that there are still many rain streaks on the results of SNN, while TNN destroys some spatial details, for instance, the leaves in the video.

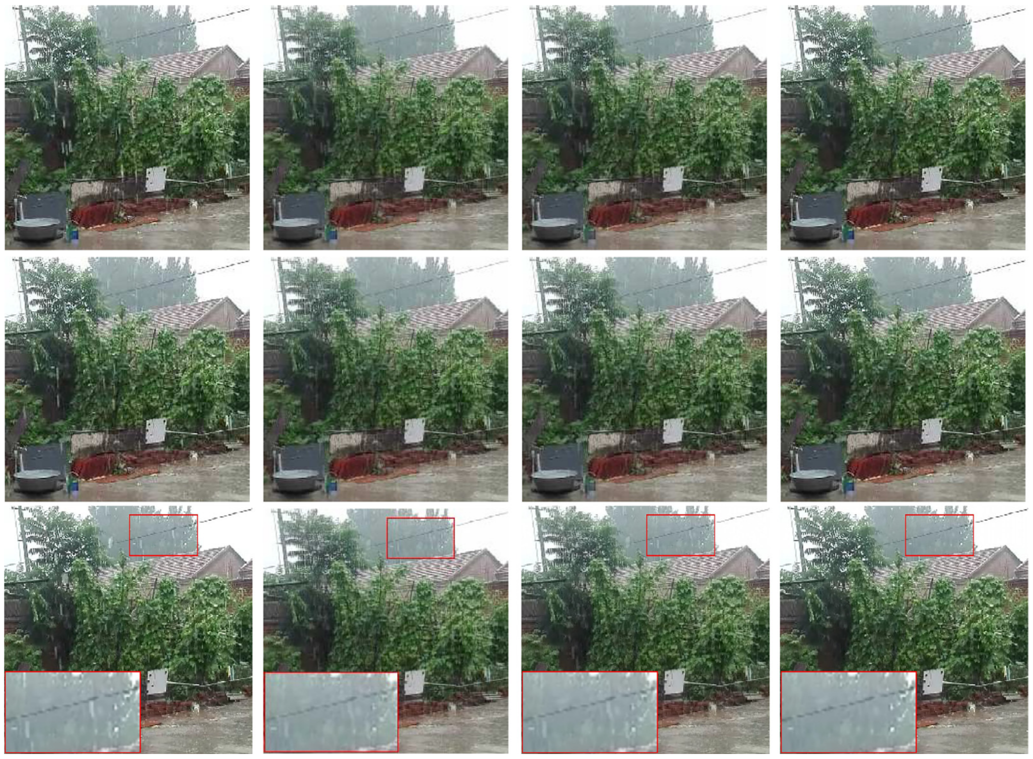

Fig. 8. Restored results on video rain streaks removal. From left to right: the real data, the recovered results by SNN, TNN, and TTNN, respectively.

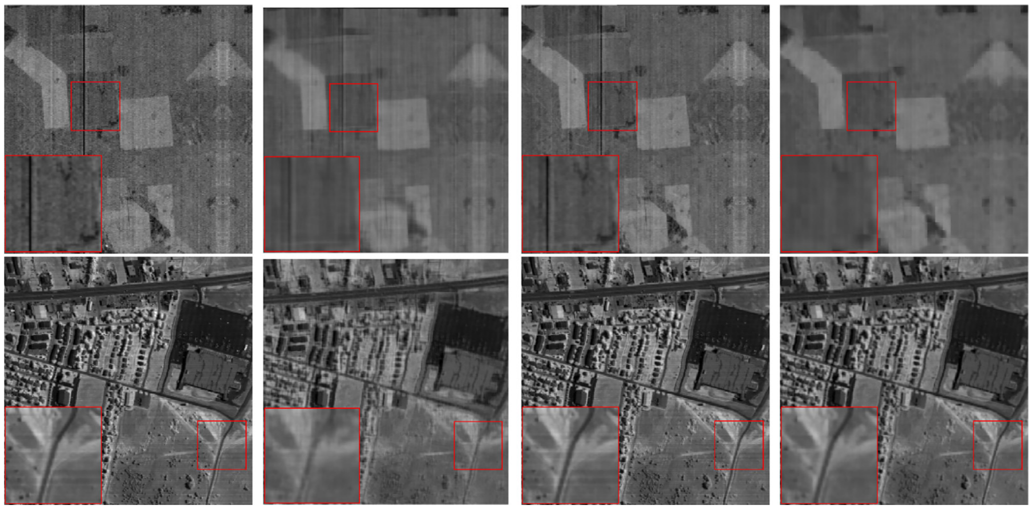

Fig. 9. Restored results on hyperspectral image stripes removal. From up to down: Hyperion and Urban. From left to right: the real data, the recovered results by SNN, TNN, and TTNN, respectively.

4.2.2. Hyperspectral images stripes removal

Two real-world hyperspectral images are used in our experiments to further test the performance of the proposed method including the *Hyperion*[4] and the *Urban*.[5] In our experiment, we only use subregions of size $256 \times 256 \times 10$ which are corrupted by stripe noise. Some representative destriping results are shown in Fig. 9. By comparing the destriping results, it is observed that SNN and TNN fail to remove heavy stripe noise. The superior performance of the proposed method can be easily observed in the labeled boxes, where our method removes stripe noise and preserves most of the details.

[4] http://remote-sensing.nci.org.au/.

[5] http://www.tec.army.mil/hypercube.

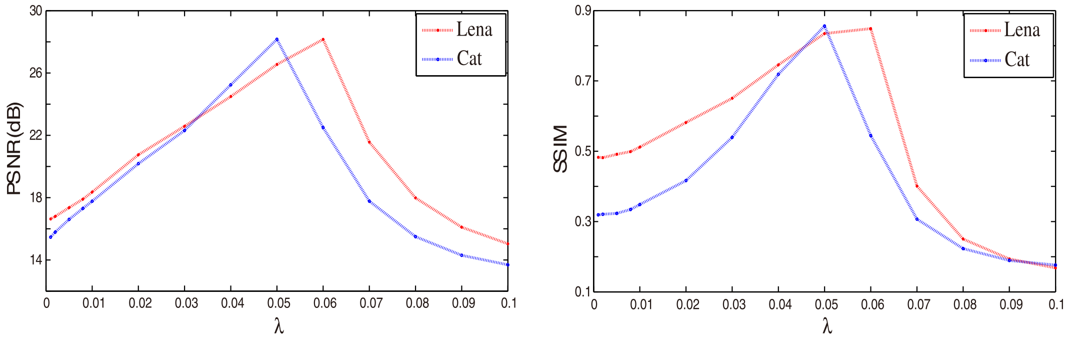

Fig. 10. The PSNR and SSIM values with respect to the regularization parameter λ.

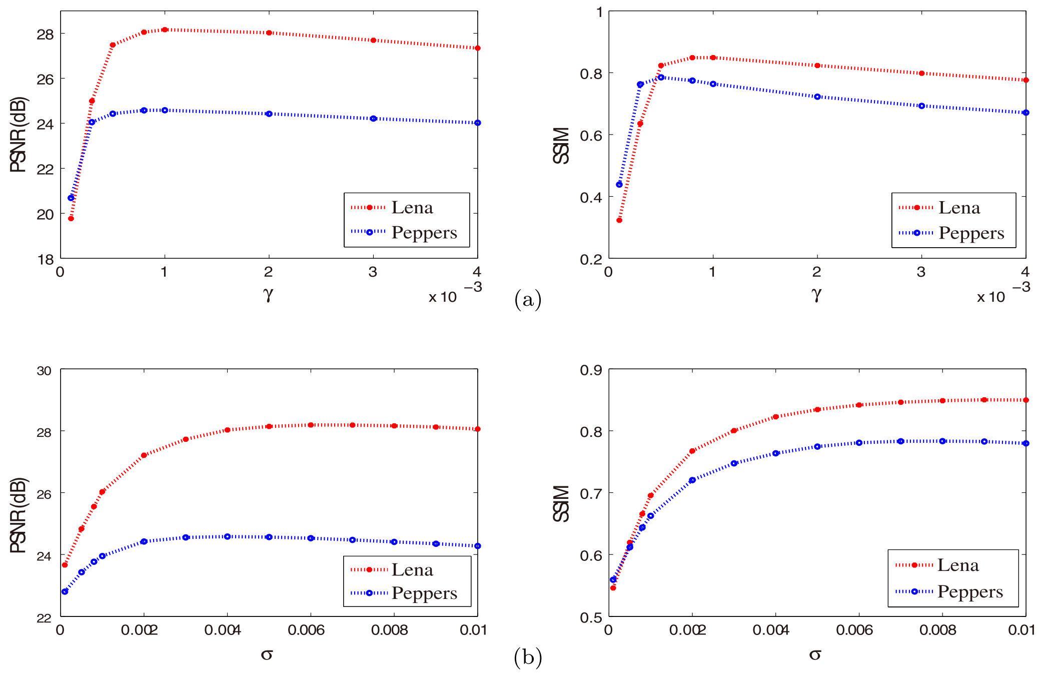

Fig. 11. The PSNR and SSIM values with respect to penalty parameters. (a) γ and (b)σ.

4.3. Discussions

Parameters selection. We study the influences of the regularization parameter λ and penalty parameters γ and σ. Taking color images (Cat, Lena, and Peppers) as examples, we test three color images corrupted by the sparse noise level 30%. The PSNR and SSIM values with respect to the regularization parameter λ are plotted in Fig. 10. It can be observed that restored images Cat and Lena achieve the highest PSNR and SSIM values with λ in 0.05 and 0.06 nearby, respectively. Fig. 11(a) and (b) show the PSNR and SSIM values with respect to penalty parameters γ and σ, respectively. From Fig. 11 (a), it is shown that PSNR and SSIM curves perform obvious improvement when γ is increased from 0 to 0.001. Moreover, we also observe that PSNR and SSIM curves are slowly declining when γ further goes increasing. In Fig. 11 (b), it also can be observed that PSNR and SSIM curves tend to be stable with respect to the parameter σ. Since our experiments involve various data and different sparse noise levels, we empirically set the optimal range of regularization parameter λ to [0.01,0.1] with increment of 0.01, and set the penalty parameter ranging as $\gamma \in [0.001, 0.003]$ and $\sigma \in [0.001, 0.01]$ with increment of 0.001 in this paper.

The effect of KA. Fig. 12 shows the recovered results by the proposed method with and without KA. We observe that the proposed method with KA achieves better visual performance compared with those without KA. The reason is that KA can effectively exploit the low-TT-rank structure hidden in the original data, by transforming it into a higher-order tensor. In fact, without KA, the TT rank for a third-order tensor only explores the correlation along the first and third modes, while KA helps TT to capture the intrinsic correlations among local structures. Thus, we use KA to enhance the performance of the proposed method.

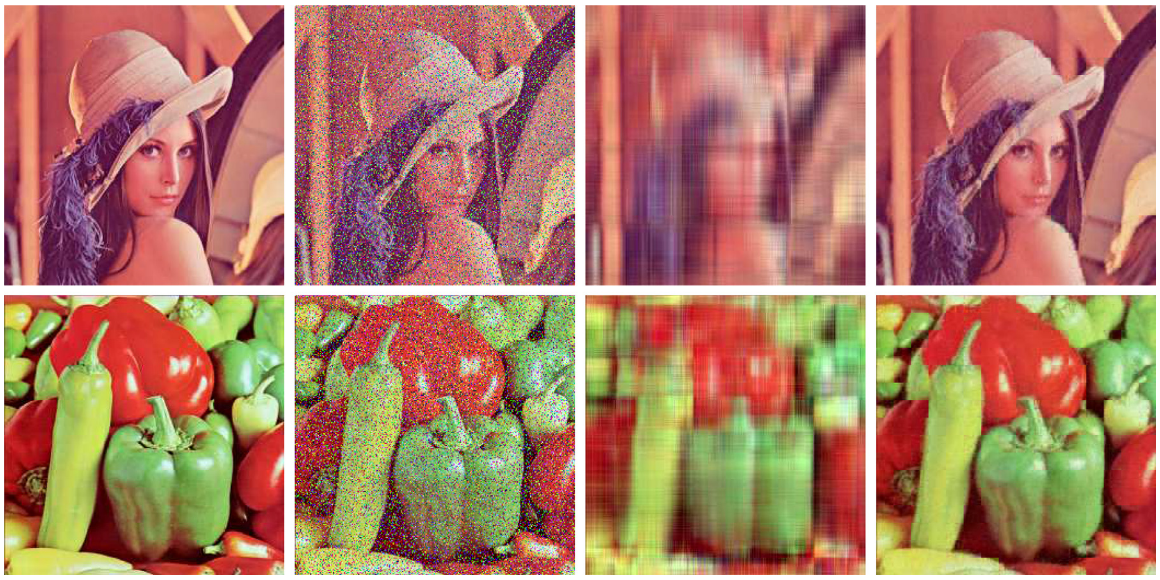

Fig. 12. The effect of KA. From left to right: the original data, the observed data, the recovered results by TTNN without and with KA, respectively.

5. Conclusion

TRPCA focuses on efficiently recovering low-rank and sparse components from the observed high-dimensional data. The key of TRPCA is to characterize the low-rankness of tensors. In this paper, we introduced TTNN into the TRPCA problem by taking into full consideration the global correlation of the high-dimensional data. The ADMM have been designed to solve the proposed model. In the simulations with various high-dimensional data sets, TTNN showed better results than SNN and TNN, which are two of the state-of-the-art TRPCA methods. For the real-world data experiments, e.g., the rain streaks removal of videos and the stripe noise removal of hyperspectral images, our method obtained visually more satisfied restoration results

Acknowledgments

The research is supported by National Natural Science Foundation of China (61876203, 61772003, 11901450), the Fundamental Research Funds for the Central Universities (31020180QD126), and National Postdoctoral Program for Innovative Talents (BX20180252). We would like to thank Canyi Lu for providing the codes of the compared methods in [18].

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.amc.2019.124783.

References

[1] X.J. Zhang, A nonconvex relaxation approach to low-rank tensor completion, IEEE Trans. Neural Netws. Learn. Syst. (2018), doi:10.1109/TNNLS.2018.2872583.
[2] Q.B. Zhao, L.Q. Zhang, A. Cichocki, Bayesian CP factorization of incomplete tensors with automatic rank determination, IEEE Trans. Pattern Anal. 37 (9) (2015) 1751–1763.
[3] X. Fu, K.J. Huang, B. Yang, W.K. Ma, N.D. Sidiropoulos, Robust volume minimization-based matrix factorization for remote sensing and document clustering, IEEE Trans. Signal Process. 64 (23) (2016) 6254–6268.
[4] J.H. Yang, X.L. Zhao, J.J. Mei, S. Wang, T.H. Ma, T.Z. Huang, Total variation and high-order total variation adaptive model for restoring blurred images with cauchy noise, Comput. Math. Appl. 77 (5) (2019) 1255–1272.
[5] H.Y. Lu, S.Q. Li, Q.G. Liu, M.H. Zhang, Mf-lrtc: multi-filters guided low-rank tensor coding for image restoration research article, Neurocomputing 303 (2018) 88–102.
[6] R.V. Oliveira, J.S. Pereira, The role of diffusion magnetic resonance imaging in Parkinson's disease and in the differential diagnosis with atypical parkinsonism, Radiol. Bras. 50 (4) (2017) 250–257.
[7] Y. Chang, L.X. Yan, T. Wu, S. Zhong, Remote sensing image stripe noise removal: from image decomposition perspective, IEEE Trans. Geosci. Remote Sens. 54 (12) (2016) 7018–7031.
[8] Y. Chen, T.Z. Huang, X.L. Zhao, Destriping of multispectral remote sensing image using low-rank tensor decomposition, IEEE J. STARS 11 (12) (2018) 4950–4967.
[9] T.Y. Ji, T.Z. Huang, X.L. Zhao, T.H. Ma, G. Liu, Tensor completion using total variation and low-rank matrix factorization, Inf. Sci. 326 (2016) 243–257.
[10] T.X. Jiang, T.Z. Huang, X.L. Zhao, L.J. Deng, Y. Wang, A novel video rain streak removal method using directional gradient priors, IEEE Trans. Image Process. 28 (4) (2019) 2089–2102.

[11] Y.T. Wang, X.L. Zhao, T.X. Jiang, L.J. Deng, Y.T. Zhang, A total variation and group sparsity based tensor optimization model for video rain streak removal, Signal Process. Image Commun. 73 (2019) 96–108.
[12] Y.B. Zheng, T.Z. Huang, T.Y. Ji, X.L. Zhao, T.X. Jiang, T.H. Ma, Low-rank tensor completion via smooth matrix factorization, Appl. Math. Model. 70 (2019) 677–695.
[13] Z. Jia, M. Ng, G. Song, Robust quaternion matrix completion with applications to image inpainting, Numer. Linear Algebra (2019), doi:10.1002/NLA.2245.
[14] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.
[15] E.J. Candés, X.D. Li, Y. Ma, J. Wright, Robust principal component analysis? J. ACM 58 (3) (2011) 1–37.
[16] W.F. Cao, Y. Wang, J. Sun, D.Y. Meng, C. Yang, A. Cichocki, Z.B. Xu, Total variation regularized tensor RPCA for background subtraction from compressive measurements, IEEE Trans. Image Process. 25 (9) (2016) 4075–4090.
[17] B. Huang, C. Mu, D. Goldfarb, J. Wright, Provable models for robust low-rank tensor completion, Pac. J. Optim. 11 (2) (2015) 339–364.
[18] C.Y. Lu, J.S. Feng, Y.D. Chen, W. Liu, Z.C. Lin, S.C. Yan, Tensor robust principal component analysis: exact recovery of corrupted low-rank tensors via convex optimization, in: Proceedings of the CVPR, 2016, pp. 5249–5257.
[19] Q. Zhao, D.Y. Meng, Z.B. Xu, C.Q. Gao, A block coordinate descent approach for sparse principal component analysis, Neurocomputing 153 (2015) 180–190.
[20] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500.
[21] H.C. Tan, B. Cheng, W.H. Wang, Y.J. Zhang, B. Ran, Tensor completion via a multi-linear low-n-rank factorization model, Neurocomputing 133 (2014) 161–169.
[22] J. Liu, P. Musialski, P. Wonka, J. Ye, Tensor completion for estimating missing values in visual data., IEEE Trans. Pattern Anal. 35 (1) (2013) 208–220.
[23] J.A. Bengua, H.N. Phien, H.D. Tuan, M.N. Do, Efficient tensor completion for color image and video recovery: low-rank tensor train, IEEE Trans. Image Process. 26 (5) (2017) 2466–2479.
[24] Z.M. Zhang, G. Ely, S. Aeron, N. Hao, M. Kilmer, Novel methods for multilinear data completion and de-noising based on tensor-SVD, in: Proceedings of the CVPR, 44, 2014, pp. 3842–3849.
[25] M.E. Kilmer, K. Braman, N. Hao, R.C. Hoover, Third-order tensors as operators on matrices: a theoretical and computational framework with applications in imaging, SIAM J. Matrix Anal. A 34 (1) (2013) 148–172.
[26] O. Semerci, N. Hao, M.E. Kilmer, E.L. Miller, Tensor-based formulation and nuclear norm regularization for multienergy computed tomography, IEEE Trans. Image Process. 23 (4) (2014) 1678–1693.
[27] Z.M. Zhang, S. Aeron, Exact tensor completion using T-SVD, IEEE Trans. Signal Process. 65 (6) (2017) 1511–1526.
[28] N. Lee, A. Cichocki, Estimating a few extreme singular values and vectors for large-scale matrices in tensor train format, SIAM J. Matrix Anal. A 36 (3) (2015) 994–1014.
[29] H. Rauhut, R. Schneider, S. Željka, Tensor completion in hierarchical tensor representations, Compressed Sensing and Its Applications, Springer, 2015, pp. 419–450.
[30] I.V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. 33 (5) (2011) 2295–2317.
[31] J.A. Bengua, H.D. Tuan, H.N. Phien, M.N. Do, Concatenated image completion via tensor augmentation and completion, in: Proceedings of the ICSPCS, 2016, pp. 1–7.
[32] M. Fazel, P.T. Kei, D. Sun, P. Tseng, Hankel matrix rank minimization with applications to system identification and realization, SIAM J. Matrix Anal. A 34 (3) (2013) 946–977.
[33] M. Ding, T.Z. Huang, S. Wang, J.J. Mei, X.L. Zhao, Total variation with overlapping group sparsity for deblurring images under cauchy noise, Appl. Math. Comput. 341 (2019) 128–147.
[34] M.L.N. Goncalves, On the pointwise iteration-complexity of a dynamic regularized ADMM with over-relaxation stepsize, Appl. Math. Comput. 336 (2018) 315–325.
[35] X.L. Zhao, W. Wang, T.Y. Zeng, T.Z. Huang, M.K. Ng, Total variation structured total least squares method for image restoration, SIAM J. Sci. Comput. 35 (6) (2013) 1304–1320.
[36] D.P. Bertsekas, A. Nedic, A.E. Ozdaglar, Convex Analysis and Optimization, Athena Scientific, 2013.
[37] T.H. Ma, T.Z. Huang, X.L. Zhao, Group-based image decomposition using 3-D cartoon and texture priors, Inf. Sci. 328 (2016) 510–527.
[38] X.L. Zhao, F. Wang, M.K. Ng, A new convex optimization model for multiplicative noise and blur removal, SIAM J. Imaging Sci. 7 (1) (2014) 456–475.
[39] C. Li, C.L. Wang, J. Wang, Convergence analysis of the augmented Lagrange multiplier algorithm for a class of matrix compressive recovery, Appl. Math. Lett. 59 (2016) 12–17.
[40] S. Ma, D. Goldfarb, L. Chen, Fixed point and Bregman iterative methods for matrix rank minimization, Math. Programm. 128 (1–2) (2009) 321–353.
[41] B. Huang, C. Mu, D. Goldfarb, J. Wright, Provable models for robust low-rank tensor completion, Pac. J. Optim. 11 (2) (2015) 339–364.
[42] D.L. Chen, Y.Q. Chen, D.Y. Xue, Fractional-order total variation image denoising based on proximity algorithm, Appl. Math. Comput. 257 (2015) 537–545.
[43] R.W. Liu, L. Shi, W. Huang, J. Xu, S.C. Yu, D. Wang, Generalized total variation-based MRI Rician denoising model with spatially adaptive regularization parameters, Magn. Reson. Imaging 32 (6) (2014) 702–720.