algorithms

MDPI

Article

Alternating Direction Method of Multipliers for Generalized Low-Rank Tensor Recovery

Jiarong Shi *, Qingyan Yin, Xiuyun Zheng and Wei Yang

School of Science, Xi'an University of Architecture and Technology, Xi'an 710055, China; qingyanyin@xauat.edu.cn (Q.Y.); xyzheng@xauat.edu.cn (X.Z.); yangweipyf@163.com (W.Y.)
* Correspondence: shijiarong@xauat.edu.cn; Tel.: +86-29-8220-5670; Fax: +86-29-82202330

Academic Editor: Stephan Chalup
Received: 23 November 2015; Accepted: 13 April 2016; Published: 19 April 2016

Abstract: Low-Rank Tensor Recovery (LRTR), the higher order generalization of Low-Rank Matrix Recovery (LRMR), is especially suitable for analyzing multi-linear data with gross corruptions, outliers and missing values, and it attracts broad attention in the fields of computer vision, machine learning and data mining. This paper considers a generalized model of LRTR and attempts to recover simultaneously the low-rank, the sparse, and the small disturbance components from partial entries of a given data tensor. Specifically, we first describe generalized LRTR as a tensor nuclear norm optimization problem that minimizes a weighted combination of the tensor nuclear norm, the l_1-norm and the Frobenius norm under linear constraints. Then, the technique of Alternating Direction Method of Multipliers (ADMM) is employed to solve the proposed minimization problem. Next, we discuss the weak convergence of the proposed iterative algorithm. Finally, experimental results on synthetic and real-world datasets validate the efficiency and effectiveness of the proposed method.

Keywords: low-rank tensor recovery; low-rank matrix recovery; nuclear norm minimization; alternating direction method of multipliers

1. Introduction

In the past decade, the low-rank property of some datasets has been explored skillfully to recover both the low-rank and the sparse components or complete the missing entries. The datasets to be analyzed are usually modeled by matrices and the corresponding recovery technique is named as Low-Rank Matrix Recovery (LRMR). LRMR has received a significant amount of attention in some fields of information science such as computer vision, machine learning, pattern recognition, data mining and linear system identification. There are several appealing types of LRMR including Matrix Completion (MC) [1], Robust Principal Component Analysis (RPCA) [2] and Low-Rank Representation (LRR) [3]. Mathematically, we customarily formulate LRMR as matrix nuclear norm minimization problems that can be effectively solved by several scalable methods. Diverse variants of LRMR derive from the aforementioned three models, for instance, MC with noise (or stable MC) [4], stable RPCA [5,6], incomplete RPCA [7], LRR with missing entries [8], and LRMR based on matrix factorization or tri-factorization [9,10].

The fields of image and signal processing usually require processing large amounts of multi-way data, such as video sequences, functional magnetic resonance imaging sequences and direct-sequence code-division multiple access (DS-CDMA) systems [11]. For the datasets with multi-linear structure, traditional matrix-based data analysis is prone to destroy the spatial and temporal structure, and can be affected by the curse of dimensionality. In contrast, tensor-based data representation can avoid or alleviate the above deficiencies to some extent. Furthermore, tensor decompositions can be used to obtain a low-rank approximation of an investigated data tensor. Two of the most popular tensor decompositions are the Tucker model and the PARAFAC (Parallel Factor Analysis) model, and they

can be thought of as the higher order generalizations of Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) [12].

Generally speaking, the traditional tensor decomposition models are very sensitive to missing entries, arbitrary outliers and gross corruptions. On the contrary, Low-Rank Tensor Recovery (LRTR) is very effective to recover simultaneously the low-rank component and the sparse noise, or complete the missing entries. As the higher order generalization of LRMR, LRTR is composed mainly of tensor completion [13] and Multi-linear RPCA (MRPCA) [14]. The task of tensor completion is to recover missing or unsampled entries according to the low-rank structure. The alternating least squares method based on the PARAFAC decomposition was originally proposed to complete missing entries [15]. Subsequently, the Tucker decomposition was applied to the problem of tensor completion [16–18]. The multi-linear rank is commonly used to describe the low-rank property, although it is hard to be properly estimated. The tensor nuclear norm generalizes the matrix nuclear norm and becomes a new standard for measuring the low-rank structure of a tensor. Hence, LRTR can usually be boiled down to a tensor nuclear norm minimization problem.

Liu *et al.* [13] established a tensor nuclear norm minimization model for tensor completion and proposed the Alternating Direction Method of Multipliers (ADMM) for efficiently solving this model. Gandy *et al.* [19] considered the low-n-rank tensor recovery problem with some linear constraints and developed the Douglas–Rachford splitting technique, a dual variant of the ADMM. To reduce the computational complexity of the tensor nuclear norm minimization problems, Liu *et al.* [20] introduced the matrix factorization idea into the minimization model and obtained a much smaller matrix nuclear norm minimization problem. Tan *et al.* [21] transformed tensor completion into a linear least squares problem and proposed a nonlinear Gauss–Seidel method to solve the corresponding optimization problem. In [14], Shi *et al.* extended RPCA to the case of tensors and presented the MRPCA model. MRPCA, also named robust low-rank tensor recovery [22], was described as a tensor nuclear norm minimization which can be solved efficiently by the ADMM.

This paper studies a generalized model of LRTR. In this model, the investigated data tensor is assumed to be the superposition of a low-rank component, a gross sparse tensor and a small dense error tensor. The Generalized Low-Rank Tensor Recovery (GLRTR) aims mainly to recover the low-rank and the sparse components from partially observed entries. For this purpose, we establish a tensor nuclear norm minimization model for GLRTR. In addition, the ADMM method is adopted to solve the proposed convex model.

The rest of this paper is organized as follows. Section 2 provides mathematical notations and introduces preliminaries on tensor algebra. In Section 3, we review related works on LRTR. We present a generalized LRTR model and develop an efficient iterative algorithm for solving the proposed model in Section 4. Section 5 discusses the weak convergence result on the proposed algorithm. We report the experimental results in Section 6. Finally, Section 7 draws the conclusions.

2. Notations and Preliminaries

This section will briefly introduce mathematical notations and review tensor algebra. We denote tensors by boldface Euclid Math One, e.g., $\boldsymbol{\mathcal{A}}$, matrices by boldface capital letters, e.g., $\mathbf{A}$, vectors by boldface letters, e.g., $\mathbf{a}$, and scalars by italic letters, e.g., a or A. A tensor with order N indicates that its entries can be expressed via N indices. For an Nth-order real tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, its $(i_1, i_2, \cdots, i_N)$ entry is denoted by $a_{i_1 i_2 \cdots i_N}$.

The n-mode matricization (unfolding) of tensor $\boldsymbol{\mathcal{A}}$, denoted by $\mathbf{A}_{(\mathrm{n})}$, is an $I_{\mathrm{n}} \times \prod_{\mathrm{j} \neq \mathrm{n}} I_{\mathrm{j}}$ matrix obtained by rearranging n-mode fibers to be the columns of the resulting matrix. Furthermore, we define an opposite operation "fold" as $\boldsymbol{fold}_{\mathrm{n}}(\mathbf{A}_{(\mathrm{n})}) = \boldsymbol{\mathcal{A}}$. Given another tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the inner product between $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ is defined as $\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} a_{i_1 i_2 \cdots i_N} b_{i_1 i_2 \cdots i_N}$. Then, the Frobenius norm (or l_2-norm) of tensor $\boldsymbol{\mathcal{A}}$ can be induced by the above inner product, that is, $\| \boldsymbol{\mathcal{A}} \|_{\mathrm{F}} = \sqrt{\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}} \rangle}$. Moreover, we also give other norm definitions of a tensor, which will be needed in the following sections. The l_1-norm

of tensor $\mathcal{A}$ is expressed as $||\mathcal{A}||_1 = \sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_N=1}^{I_N}|a_{i_1 i_2\cdots i_N}|$ and the tensor nuclear norm is $||\mathcal{A}||_* = \sum_{n=1}^{N} w_n ||\mathbf{A}_{(n)}||_*$, where $||\mathbf{A}_{(n)}||_*$, named the nuclear norm of $\mathbf{A}_{(n)}$, is the sum of singular values of $\mathbf{A}_{(n)}$, $w_n \geqslant 0$ and $\sum_{n=1}^{N} w_n = 1$. Due to the fact that the nuclear norm of one matrix is the tightest convex relaxation of the rank function on the unit ball in the spectral norm, it is easy to verify that the tensor nuclear norm $||\mathcal{A}||_*$ is a convex function with respect to $\mathcal{A}$.

The n-mode product of tensor $\mathcal{A}$ by a matrix $\mathbf{U} \in \mathbb{R}^{J_n \times I_n}$, denoted as $\mathcal{A} \times_n \mathbf{U}$, is an Nth-order tensor with the dimensionality of $I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n-1} \times \cdots \times I_N$, and its $(i_1, \cdots, i_{n-1}, j, i_{n+1} \cdots, i_N) - th$ entry is calculated as $\sum_{i_n=1}^{I_n} a_{i_1 i_2 \cdots i_N} u_{j i_n}$. The multi-linear rank of tensor $\mathcal{A}$ is stipulated as a vector $\Big(rank(\mathbf{A}_{(1)}), rank(\mathbf{A}_{(2)}), \ldots, rank(\mathbf{A}_{(N)})\Big)$, where $rank(\cdot)$ indicates the rank function of a matrix. Tensor $\mathcal{A}$ is low-rank if and only if $rank(\mathbf{A}_{(n)}) \ll I_n$ for some n. Then, the Tucker decomposition of $\mathcal{A}$ is defined as

$$\mathcal{A} \approx \mathcal{C} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_N \mathbf{U}^{(N)} \triangleq \mathcal{C} \times_{n=1}^{N} \mathbf{U}^{(n)}, \tag{1}$$

where the core tensor $\mathcal{C} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, mode matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{J_n \times I_n}$ and $J_n \leqslant I_n$. The multi-linear rank of a tensor has superiority over other definitions (e.g., the rank used in PARAFAC decomposition) because it is easy to compute. Refer to the survey [12] for further understanding of tensor algebra.

Within the field of low-rank matrix/tensor recovery, two proximal minimization problems are extensively employed:

$$\min_{\mathcal{X}} \lambda ||\mathcal{X}||_1 + \frac{1}{2}||\mathcal{X} - \mathcal{A}||_F^2, \tag{2}$$

$$\min_{\mathbf{X}} \lambda ||\mathbf{X}||_* + \frac{1}{2}||\mathbf{X} - \mathbf{A}||_F^2, \tag{3}$$

where $\mathcal{A}$ and $\mathbf{A}$ are a given data tensor and matrix respectively, $\lambda \geqslant 0$ is a regularization factor. To address the above two optimization problems, we first define an absolute thresholding operator $S_\lambda() : \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \to \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ as below: $(S_\lambda(\mathcal{A}))_{i_1 i_2 \cdots i_N} = \max\left(|a_{i_1 i_2 \cdots i_N}| - \lambda, 0\right)$. It has been proven that problems (2) and (3) have closed-form solutions denoted by $S_\lambda(\mathcal{A})$ [2] and $T_\lambda(\mathbf{A})$ [23] respectively, where $T_\lambda(\mathbf{A}) = \mathbf{U}S_\lambda(\mathbf{\Sigma})\mathbf{V}^T$ and $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is the singular value decomposition of $\mathbf{A}$.

3. Related Works

Tensor completion and MRPCA are two important and appealing applications of LRTR. In this section, we review the related works on the aforementioned two applications.

As the higher order generalization of matrix completion, tensor completion aims to recover all missing entries with the aid of the low-rank (or approximately low-rank) structure of a data tensor. Although low-rank tensor decompositions are practical in dealing with missing values [15–18], we have to estimate properly the rank of an incomplete tensor in advance. In the past few years, the matrix nuclear norm minimization model has been extended to the tensor case [13]. Given an incomplete data tensor $\mathcal{D} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a corresponding sampling index set $\mathbf{\Omega} \subset [I_1] \times [I_2] \times \cdots \times [I_N]$, we define a linear operator $P_\mathbf{\Omega}() : \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \to \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ as follows: if $(i_1, i_2, \cdots, i_N) \in \mathbf{\Omega}$, $(P_\mathbf{\Omega}(\mathcal{D}))_{i_1 i_2 \cdots i_N} = d_{i_1 i_2 \cdots i_N}$; otherwise, $(P_\mathbf{\Omega}(\mathcal{D}))_{i_1 i_2 \cdots i_N} = 0$, where $[I_n] = \{1, 2, \ldots, I_n\}$. Then, the original tensor nuclear norm minimization model for tensor completion [13] is described as:

$$\min_{\mathcal{X}} ||\mathcal{X}||_*, \text{ s.t. } P_\mathbf{\Omega}(\mathcal{X}) = P_\mathbf{\Omega}(\mathcal{D}). \tag{4}$$

To tackle problem (4), Liu *et al.* [13] developed three different algorithms and demonstrated experimentally that the ADMM is the most efficient algorithm in obtaining a high accuracy solution. If we further take the dense Gaussian noise into consideration, then the stable version of tensor completion is given as below [19]:

$$\min_{\mathcal{X}} ||\mathcal{X}||_*, \text{ s.t. } ||P_\mathbf{\Omega}(\mathcal{X}) - P_\mathbf{\Omega}(\mathcal{D})||_F \leqslant \eta, \tag{5}$$

or its corresponding unconstrained formulation:

$$\min_{\mathcal{X}} \lambda \| \mathcal{X} \|_* + \frac{1}{2} \| P_{\Omega}(\mathcal{X}) - P_{\Omega}(\mathcal{D}) \|_F^2, \tag{6}$$

where the regularized parameter $\lambda (\geqslant 0)$ is used to balance the low-rankness and the approximate error, and η is a known estimate of the noise level.

In RPCA, a data matrix is decomposed into the sum of a low-rank component and a sparse component, and it is possible to recover simultaneously the two components by principal component pursuit under some suitable assumptions [2]. Shi *et al.* [14] extended RPCA to the case of tensors and presented the framework of MRPCA, which regards the data tensor $\mathcal{D}$ as the sum of a low-rank tensor $\mathcal{A}$ and a sparse noise term $\mathcal{E}$. Mathematically, the low-rank and the sparse components can be simultaneously recovered by solving the following tensor nuclear norm minimization problem:

$$\min_{\mathcal{A},\mathcal{E}} \| \mathcal{A} \|_* + \lambda \| \mathcal{E} \|_1, \text{ s.t. } \mathcal{D} = \mathcal{A} + \mathcal{E}. \tag{7}$$

In [22], MRPCA is also called as robust low-rank tensor recovery.

4. Generalized Low-Rank Tensor Recovery

Both tensor completion and MRPCA do not consider dense Gaussian noise corruptions. In this section, we investigate the model of Generalized Low-Rank Tensor Recovery (GLRTR) and develop a corresponding iterative scheme.

4.1. Model of GLRTR

The datasets contaminated by Gaussian noise are very universal in practical engineering applications. In view of this, we assume the data tensor $\mathcal{D}$ to be the superposition of the low-rank component $\mathcal{A}$, the large sparse corruption $\mathcal{E}$ and the Gaussian noise $\mathcal{G}$. We also consider the case that some entries of $\mathcal{D}$ are missing. To recover simultaneously the above three terms, we establish a convex GLRTR model as follows:

$$\min_{\mathcal{A},\mathcal{E},\mathcal{G}} \| \mathcal{A} \|_* + \lambda \| \mathcal{E} \|_1 + \tau \| \mathcal{G} \|_F^2, \text{ s.t. } P_{\Omega}(\mathcal{D}) = P_{\Omega}(\mathcal{A} + \mathcal{E} + \mathcal{G}), \tag{8}$$

where the regularization coefficients λ and τ are nonnegative.

If we reinforce the constraints $\mathcal{G} = \mathcal{O}$(or equivalently $\tau \to +\infty$) and $\mathcal{E} = \mathcal{O}$ (or equivalently $\lambda \to +\infty$), then the GLRTR is transformed into the tensor completion model (4), where the zero tensor $\mathcal{O}$ has the same dimensionality as $\mathcal{D}$. If only the constraint $\mathcal{E} = \mathcal{O}$ is considered, then the GLRTR is equivalent to Equation (6). Furthermore, if we take $\mathcal{G} = \mathcal{O}$ and $\Omega = [I_1] \times [I_2] \times \cdots \times [I_N]$, then the model of GLRTR becomes the model of MRPCA. In summary, the proposed model is the generalization of the existing LRTR.

For the convenience of using the splitting method, we discard $\mathcal{A}$ and introduce $N + 1$ auxiliary tensor variables $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_N, \mathcal{X}$, where these auxiliary variables have the same dimensionality as $\mathcal{D}$. Let $\mathbf{M}_{n(n)}$ be the n-mode matricization of $\mathcal{M}_n$ for each $n \in [N]$ and $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_N\}$. Hence, we have the equivalent formulation of Equation (8):

$$\begin{aligned} &\min_{\mathcal{X},\mathcal{E},\mathcal{G},\mathcal{M}} \sum_{N=1}^{n} w_n \| \mathbf{M}_{n(n)} \|_* + \lambda \| \mathcal{E} \|_1 + \tau \| \mathcal{G} \|_F^2, \\ &\text{s.t. } \mathcal{X} = \mathcal{M}_n + \mathcal{G} + \mathcal{E}, P_{\Omega}(\mathcal{X}) = P_{\Omega}(\mathcal{D}), n = 1, 2, \ldots, N. \end{aligned} \tag{9}$$

As a matter of fact, the discarded low-rank tensor $\mathcal{A}$ can be represented by $\mathcal{A} = \sum_{N=1}^{n} \mathcal{M}_n / N$. Now, we explain the low-rankness of $\mathcal{A}$ from two viewpoints. One is $\| \mathcal{A} \|_* \leqslant \sum_{N=1}^{n} \| \mathcal{M}_n \|_* / N \leqslant \max_{1 \leqslant n \leqslant N} \| \mathcal{M}_n \|_*$. The other is that a better solution to Equation (9) will satisfy $\mathcal{M}_1 \approx \mathcal{M}_2 \approx \ldots \approx \mathcal{M}_N$. These viewpoints illustrate that $\mathcal{A}$ is approximately low-rank

along each mode. The aforementioned non-smooth minimization problem is distributed convex. Concretely speaking, the tensor variables can be split into several parts and the objective function is separable across this splitting.

4.2. Optimization Algorithm to GLRTR

As a special splitting method, the ADMM is very efficient to solve a distributed optimization problem with linear equality constraints. It takes the form of the decomposition–coordination procedure and blends the merits of dual decomposition and augmented Lagrangian methods. In this section, we will propose the method of ADMM to solve Equation (9).

We first construct the augmented Lagrangian function of the aforementioned convex optimization problem without considering the constraint $\mathrm{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{X}}) = \mathrm{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{D}})$:

$$\begin{aligned} \mathrm{L}_{\mu}(\boldsymbol{\mathcal{X}}, \quad \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}}) = & \sum_{\mathrm{n}=1}^{\mathrm{N}} \mathrm{w}_{\mathrm{n}} \left|\left| \mathbf{M}_{\mathrm{n}(\mathrm{n})} \right|\right|_{*} + \lambda \left|\left| \boldsymbol{\mathcal{E}} \right|\right|_{1} + \tau \left|\left| \boldsymbol{\mathcal{G}} \right|\right|_{\mathrm{F}}^{2} + \\ & \sum_{\mathrm{n}=1}^{\mathrm{N}} \left(\langle \boldsymbol{\mathcal{Y}}_{\mathrm{n}}, \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{M}}_{\mathrm{n}} - \boldsymbol{\mathcal{G}} - \boldsymbol{\mathcal{E}} \rangle + \frac{\mu}{2} \left|\left| \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{M}}_{\mathrm{n}} - \boldsymbol{\mathcal{G}} - \boldsymbol{\mathcal{E}} \right|\right|_{\mathrm{F}}^{2} \right), \end{aligned} \tag{10}$$

where μ is a positive numerical constant, $\boldsymbol{\mathcal{Y}}_{\mathrm{n}}$ are Lagrangian multiplier tensors and $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{\mathcal{Y}}_1, \boldsymbol{\mathcal{Y}}_2, \ldots, \boldsymbol{\mathcal{Y}}_{\mathrm{N}}\}$. There are totally five blocks of variables in $\mathrm{L}_{\mu}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}})$. The ADMM algorithm updates alternatively each block of variables while letting the other blocks of variables be fixed. If $\boldsymbol{\mathcal{Y}}$ is given, one block of variables in $\{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{E}}\}$ can be updated by minimizing $\mathrm{L}_{\mu}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}})$ with respect to one argument. On the contrary, if $\{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{E}}\}$ are given, $\boldsymbol{\mathcal{Y}}$ is updated by maximizing $\mathrm{L}_{\mu}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}})$ with respect to $\boldsymbol{\mathcal{Y}}$. The detailed iterative procedure is outlined as follows:

Computing $\boldsymbol{\mathcal{X}}$. Fix $\{\boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}}\}$ and minimize $\mathrm{L}_{\mu}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}})$ with respect to $\boldsymbol{\mathcal{X}}$:

$$\begin{aligned} \boldsymbol{\mathcal{X}}: \quad &= \arg\min_{\boldsymbol{\mathcal{X}}} \mathrm{L}_{\mu}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}}) \\ &= \arg\min_{\boldsymbol{\mathcal{X}}} \textstyle\sum_{\mathrm{n}=1}^{\mathrm{N}} \left|\left| \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{M}}_{\mathrm{n}} - \boldsymbol{\mathcal{G}} - \boldsymbol{\mathcal{E}} + \boldsymbol{\mathcal{Y}}_{\mathrm{n}}/\mu \right|\right|_{\mathrm{F}}^{2} \\ &= \widetilde{\boldsymbol{\mathcal{X}}}, \end{aligned} \tag{11}$$

where $\widetilde{\boldsymbol{\mathcal{X}}} = \sum_{\mathrm{n}=1}^{\mathrm{N}} (\boldsymbol{\mathcal{M}}_{\mathrm{n}} - \boldsymbol{\mathcal{Y}}_{\mathrm{n}}/\mu)/\mathrm{N} + \boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{E}}$. In consideration of the constraint $\mathrm{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{X}}) = \mathrm{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{D}})$, the final update formulation of $\boldsymbol{\mathcal{X}}$ is revised as

$$\boldsymbol{\mathcal{X}} := \mathrm{P}_{\overline{\boldsymbol{\Omega}}}(\widetilde{\boldsymbol{\mathcal{X}}}) + \mathrm{P}_{\boldsymbol{\Omega}}(\boldsymbol{\mathcal{D}}), \tag{12}$$

where $\overline{\boldsymbol{\Omega}}$ is the complement set of $\boldsymbol{\Omega}$.

Computing $\boldsymbol{\mathcal{M}}$. If $\boldsymbol{\mathcal{M}}_{\mathrm{n}}$ is unknown and other variables are fixed, we update $\boldsymbol{\mathcal{M}}_{\mathrm{n}}$ by minimizing $\mathrm{L}_{\mu}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}})$ with respect to $\boldsymbol{\mathcal{M}}_{\mathrm{n}}$:

$$\begin{aligned} \boldsymbol{\mathcal{M}}_{\mathrm{n}}: \quad &= \arg\min_{\boldsymbol{\mathcal{M}}_{\mathrm{n}}} \mathrm{L}_{\mu}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}}) \\ &= \arg\min_{\boldsymbol{\mathcal{M}}_{\mathrm{n}}} \mathrm{w}_{\mathrm{n}} \left|\left| \mathbf{M}_{\mathrm{n}(\mathrm{n})} \right|\right|_{*} + \frac{\mu}{2} \left|\left| (\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} - \boldsymbol{\mathcal{E}} + \boldsymbol{\mathcal{Y}}_{\mathrm{n}}/\mu) - \boldsymbol{\mathcal{M}}_{\mathrm{n}} \right|\right|_{\mathrm{F}}^{2} \\ &= \arg\min_{\boldsymbol{\mathcal{M}}_{\mathrm{n}}} \frac{\mathrm{w}_{\mathrm{n}}}{\mu} \left|\left| \mathbf{M}_{\mathrm{n}(\mathrm{n})} \right|\right|_{*} + \frac{1}{2} \left|\left| \left(\mathbf{Q}_{(\mathrm{n})} - \mathbf{M}_{\mathrm{n}(\mathrm{n})} \right) - \boldsymbol{\mathcal{M}}_{\mathrm{n}} \right|\right|_{\mathrm{F}}^{2} \\ &= \mathrm{fold}_{\mathrm{n}} \left(\mathrm{T}_{\mathrm{w}_{\mathrm{n}}/\mu} \left(\mathbf{Q}_{(\mathrm{n})} \right) \right), \end{aligned} \tag{13}$$

where $\boldsymbol{\mathcal{Q}} = \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} - \boldsymbol{\mathcal{E}} + \boldsymbol{\mathcal{Y}}_{\mathrm{n}}/\mu$ and $\mathbf{Q}_{(\mathrm{n})}$ is the n-mode matricization of $\boldsymbol{\mathcal{Q}}$.

Computing $\mathcal{E}$. If $\mathcal{E}$ is unknown and other variables are fixed, the calculation procedure of $\mathcal{E}$ is given as follows:

$$\begin{aligned}\mathcal{E}: &= \arg\min_{\mathcal{E}} \mathrm{L}_{\mu}(\mathcal{X},\mathcal{M},\mathcal{E},\mathcal{G},\mathcal{Y}) \\ &= \arg\min_{\mathcal{E}} \tfrac{\lambda}{\mu \mathrm{N}} ||\mathcal{E}||_1 + \tfrac{1}{2}||\mathcal{E} - \textstyle\sum_{\mathrm{n}=1}^{\mathrm{N}} (\mathcal{X} - \mathcal{M}_{\mathrm{n}} - \mathcal{G} + \mathcal{Y}_{\mathrm{n}}/\mu)/\mathrm{N}||_{\mathrm{F}}^2 \\ &= \mathrm{S}_{(\lambda/\mu \mathrm{N})}\left(\textstyle\sum_{\mathrm{n}=1}^{\mathrm{N}} (\mathcal{X} - \mathcal{M}_{\mathrm{n}} - \mathcal{G} + \mathcal{Y}_{\mathrm{n}}/\mu)/\mathrm{N}\right).\end{aligned} \tag{14}$$

Computing $\mathcal{G}$. The update formulation of $\mathcal{G}$ is calculated as

$$\begin{aligned}\mathcal{G}: &= \arg\min_{\mathcal{G}} \mathrm{L}_{\mu}(\mathcal{X},\mathcal{M},\mathcal{E},\mathcal{G},\mathcal{Y}) \\ &= \arg\min_{\mathcal{G}} \tau ||\mathcal{G}||_{\mathrm{F}}^2 + \tfrac{\mu \mathrm{N}}{2}||\mathcal{G} - \textstyle\sum_{\mathrm{n}=1}^{\mathrm{N}} (\mathcal{X} - \mathcal{M}_{\mathrm{n}} - \mathcal{E} + \mathcal{Y}_{\mathrm{n}}/\mu)/\mathrm{N}||_{\mathrm{F}}^2 \\ &= \tfrac{\mu}{2\tau + \mu \mathrm{N}} \sum_{\mathrm{n}=1}^{\mathrm{N}} (\mathcal{X} - \mathcal{M}_{\mathrm{n}} - \mathcal{E} + \mathcal{Y}_{\mathrm{n}}/\mu).\end{aligned} \tag{15}$$

Computing $\mathcal{Y}$. If $\{\mathcal{X},\mathcal{M},\mathcal{E},\mathcal{G}\}$ are fixed, we can update the Lagrangian multipliers $\mathcal{Y}$ as follows:

$$\mathcal{Y}_{\mathrm{n}} := \mathcal{Y}_{\mathrm{n}} + \mu(\mathcal{X} - \mathcal{M}_{\mathrm{n}} - \mathcal{G} - \mathcal{E}),\ \mathrm{n} = 1, 2, \ldots, \mathrm{N}. \tag{16}$$

Once $\{\mathcal{X},\mathcal{M},\mathcal{E},\mathcal{G},\mathcal{Y}\}$ are updated, we will increase the value of μ by multiplying it with a constant $\rho(> 1)$ during the procedure of iterations. The whole iterative procedure is outlined in Algorithm 1. The stopping condition of Algorithm 1 is set as $\max_{1 \leqslant \mathrm{n} \leqslant \mathrm{N}} ||\mathcal{X} - \mathcal{M}_{\mathrm{n}} - \mathcal{G} - \mathcal{E}||_{\mathrm{F}}^2 < \varepsilon$ or the maximum number of iterations is reached, where ε is a sufficiently small positive number.

Algorithm 1. Solving GLRTR by ADMM

Input: Data tensor $\mathcal{D}$, sampling index set Ω, regularization parameters λ and τ.
Initialize: $\mathcal{M},\mathcal{E},\mathcal{G},\mathcal{Y},\mu,\mu_{\max}$ and ρ.
Output: $\mathcal{X},\mathcal{E},\mathcal{G}$ and $\mathcal{M}$.
While not converged **do**

1. Update $\mathcal{X}$ according to (12).
2. Update $\mathcal{M}_{\mathrm{n}}$ according to (13), $\mathrm{n} = 1, 2, \ldots, \mathrm{N}$.
3. Update $\mathcal{E}$ according to (14).
4. Update $\mathcal{G}$ according to (15).
5. Update $\mathcal{Y}_{\mathrm{n}}$ according to (16), $\mathrm{n} = 1, 2, \ldots, \mathrm{N}$.
6. Update μ as $\mu := \min(\rho\mu, \mu_{\max})$.

End while

In our implementation, $\mathcal{M},\mathcal{E},\mathcal{G}$ and $\mathcal{Y}$ are initialized to zeros tensors. The other parameters are set as follows: $\mathrm{w}_1 = \mathrm{w}_2 = \ldots = \mathrm{w}_{\mathrm{N}} = 1/\mathrm{N}, \mu = 10^{-4}, \rho = 1.1,\ \mu_{\max} = 10^{10}$ and $\varepsilon = 10^{-8}$. Furthermore, the maximum number of iterations is set to 100.

5. Convergence Analysis

Because the number of block variables in Equation (9) is more than two, it is difficult for us to prove the convergence of Algorithm 1. Nevertheless, the experimental results in the next section demonstrate this algorithm has good convergence behavior. This section will discuss the weak convergence result on our ADMM algorithm.

We consider a special case of Algorithm 1, that is, there is no missing entries. In this case, it holds that $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{D}}$. Thus, Equation (9) is transformed into

$$\begin{array}{c} \min\limits_{\boldsymbol{\mathcal{M}},\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{G}}} \mathrm{w_n} \,||\, \mathbf{M}_{\mathrm{n(n)}} \,||_* + \boldsymbol{\lambda} \,||\, \boldsymbol{\mathcal{E}} \,||_{\mathbf{1}} + \boldsymbol{\tau} \,||\, \boldsymbol{\mathcal{G}} \,||_{\mathrm{F}}^{\mathbf{2}} \\ \mathrm{s.t.}\ \boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{M}}_{\mathrm{n}} + \boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{E}}, \mathrm{n} = \mathbf{1}, \mathbf{2}, \ldots, \mathrm{N}. \end{array} \tag{17}$$

We can design a corresponding ADMM algorithm by revising Algorithm 1, namely, the update formulation of $\boldsymbol{\mathcal{X}}$ is replaced by $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{D}}$. In fact, the revised algorithm is an inexact version of ADMM. Subsequently, we give the iterative formulations of exact ADMM for Equation (17):

$$\begin{cases} \boldsymbol{\mathcal{M}} := \arg\min\limits_{\boldsymbol{\mathcal{M}}} \mathrm{L}_{\boldsymbol{\mu}}(\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}}), \\ (\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}) := \arg\min\limits_{\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{G}}} \mathrm{L}_{\boldsymbol{\mu}}(\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}}), \\ \boldsymbol{\mathcal{Y}}_{\mathrm{n}} := \boldsymbol{\mathcal{Y}}_{\mathrm{n}} + \boldsymbol{\mu}\,(\boldsymbol{\mathcal{D}} - \boldsymbol{\mathcal{M}}_{\mathrm{n}} - \boldsymbol{\mathcal{G}} - \boldsymbol{\mathcal{E}}), \mathrm{n} = \mathbf{1}, \mathbf{2}, \ldots, \mathrm{N}. \end{cases} \tag{18}$$

In this circumstance, we have the following statement:

Theorem 1. *Let* $\left\{\boldsymbol{\mathcal{M}}^{(\mathrm{k})}, \boldsymbol{\mathcal{E}}^{(\mathrm{k})}, \boldsymbol{\mathcal{G}}^{(\mathrm{k})}, \boldsymbol{\mathcal{Y}}^{(\mathrm{k})}\right\}$ *be the sequence produced by (18). If* $\mathrm{L}_{\mathbf{0}}(\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}})$ *has a saddle point with respect to* $\{\boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Y}}\}$*, then* $\left\{\boldsymbol{\mathcal{M}}^{(\mathrm{k})}, \boldsymbol{\mathcal{E}}^{(\mathrm{k})}, \boldsymbol{\mathcal{G}}^{(\mathrm{k})}\right\}$ *satisfy that the iterative sequence of objective function in Equation (17) converges to the minimum value.*

Proof. By matricizing each tensor along the one-mode, the constraints of Equation (17) can be rewritten as a linear equation:

$$\begin{pmatrix} \mathbf{D}_{(\mathbf{1})} \\ \mathbf{D}_{(\mathbf{1})} \\ \vdots \\ \mathbf{D}_{(\mathbf{1})} \end{pmatrix} = \begin{pmatrix} \mathbf{M}_{\mathbf{1}(\mathbf{1})} \\ \mathbf{M}_{\mathbf{2}(\mathbf{1})} \\ \vdots \\ \mathbf{M}_{\mathrm{N}(\mathbf{1})} \end{pmatrix} + \begin{pmatrix} \mathbf{G}_{(\mathbf{1})} \\ \mathbf{G}_{(\mathbf{1})} \\ \vdots \\ \mathbf{G}_{(\mathbf{1})} \end{pmatrix} + \begin{pmatrix} \mathbf{E}_{(\mathbf{1})} \\ \mathbf{E}_{(\mathbf{1})} \\ \vdots \\ \mathbf{E}_{(\mathbf{1})} \end{pmatrix}, \tag{19}$$

or equivalently,

$$\begin{pmatrix} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{pmatrix} \mathbf{D}_{(\mathbf{1})} = \begin{pmatrix} \mathbf{M}_{\mathbf{1}(\mathbf{1})} \\ \mathbf{M}_{\mathbf{2}(\mathbf{1})} \\ \vdots \\ \mathbf{M}_{\mathrm{N}(\mathbf{1})} \end{pmatrix} + \begin{pmatrix} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{pmatrix} \mathbf{G}_{(\mathbf{1})} + \begin{pmatrix} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{pmatrix} \mathbf{E}_{(\mathbf{1})}, \tag{20}$$

where $\mathbf{I}$ is an identity matrix of size $\mathrm{I}_1 \times \mathrm{I}_1$. If we integrate $\mathbf{G}_{(\mathbf{1})}$ and $\mathbf{E}_{(\mathbf{1})}$ into one block of variables, then Equation (20) is re-expressed as below:

$$\begin{pmatrix} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{pmatrix} \mathbf{D}_{(\mathbf{1})} = \begin{pmatrix} \mathbf{M}_{\mathbf{1}(\mathbf{1})} \\ \mathbf{M}_{\mathbf{2}(\mathbf{1})} \\ \vdots \\ \mathbf{M}_{\mathrm{N}(\mathbf{1})} \end{pmatrix} + \begin{pmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & \mathbf{I} \\ \vdots & \vdots \\ \mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{G}_{(\mathbf{1})} \\ \mathbf{E}_{(\mathbf{1})} \end{pmatrix}. \tag{21}$$

We denote $\mathbf{M} = \left\{\mathbf{M}_{\mathbf{1}(\mathbf{1})}, \mathbf{M}_{\mathbf{2}(\mathbf{1})}, \cdots, \mathbf{M}_{\mathrm{N}(\mathbf{1})}\right\}$ and partition all variables in Equation (17) into two parts in form of matrices: $\mathbf{M}$ and $\left(\mathbf{E}_{(\mathbf{1})}, \mathbf{G}_{(\mathbf{1})}\right)$. Essentially, $\mathbf{M}, \mathbf{E}_{(\mathbf{1})}$ and $\mathbf{G}_{(\mathbf{1})}$ are the matricizations of $\boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{E}}$ and $\boldsymbol{\mathcal{G}}$, respectively. Let $\mathrm{g}(\mathbf{M}) = \sum\limits_{\mathrm{n}=\mathbf{1}}^{\mathrm{N}} \mathrm{w_n} \,||\, \mathbf{M}_{\mathrm{n(n)}} \,||_*$, $\mathrm{h}(\mathbf{E}_{(\mathbf{1})}, \mathbf{G}_{(\mathbf{1})}) = \boldsymbol{\lambda} \,||\, \mathbf{E}_{(\mathbf{1})} \,||_{\mathbf{1}} + \boldsymbol{\tau} \,||\, \mathbf{G}_{(\mathbf{1})} \,||_{\mathrm{F}}^{\mathbf{2}}$. Then, the objective function in Equation (17) can be expressed as $\mathrm{g}(\mathbf{M}) + \mathrm{h}(\mathbf{E}_{(\mathbf{1})}, \mathbf{G}_{(\mathbf{1})})$. It is obvious that $\mathrm{g}(\mathbf{M})$ and $\mathrm{h}(\mathbf{E}_{(\mathbf{1})}, \mathbf{G}_{(\mathbf{1})})$ are two closed, proper and convex functions. According to the basic

convergence result given in [24], we have $\lim_{k\to+\infty} g(\mathbf{M}^{(k)})+h(\mathbf{E}_{(1)}^{(k)},\mathbf{G}_{(1)}^{(k)}) = f^*$, where f^* is the minimum value of $g(\mathbf{M}) + h(\mathbf{E}_{(1)},\mathbf{G}_{(1)})$ under the linear Constraint (21). This ends the proof. □

In the following, we discuss the detailed iterative procedure of $\left\{\mathbf{M},\mathbf{E}_{(1)},\mathbf{G}_{(1)}\right\}$ or $\{\boldsymbol{\mathcal{M}},\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{G}}\}$ in Equation (18). It is easy to verify that the update formulation of $\boldsymbol{\mathcal{M}}$ in Equation (18) is equivalent to:

$$\begin{aligned}&(\mathbf{M}_{1(1)},\mathbf{M}_{2(1)},\cdots,\mathbf{M}_{N(1)}) := \arg\min_{\mathbf{M}} L_\mu(\boldsymbol{\mathcal{D}},\boldsymbol{\mathcal{M}},\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{G}},\boldsymbol{\mathcal{Y}})\\ &= \arg\min_{(\mathbf{M}_{1(1)},\mathbf{M}_{2(1)},\cdots,\mathbf{M}_{N(1)})} \sum_{n=1}^{N}\left(w_n||\mathbf{M}_{n(n)}||_* + \frac{\mu}{2}||\boldsymbol{\mathcal{D}}-\boldsymbol{\mathcal{M}}_n-\boldsymbol{\mathcal{G}}-\boldsymbol{\mathcal{E}}+\frac{1}{\mu}\boldsymbol{\mathcal{Y}}_n||_F^2\right).\end{aligned} \tag{22}$$

The block of variables $\mathbf{M}_{n(1)}$ can be solved in parallel because the objective function in Equation (22) is separable with respect to $\mathbf{M}_{1(1)},\mathbf{M}_{2(1)},\cdots,\mathbf{M}_{N(1)}$. Hence, the iterative formulation of $\boldsymbol{\mathcal{M}}_n$ is similar to that of Equation (13).

For fixed $\mathbf{M}$ and $\boldsymbol{\mathcal{Y}}$, we can get the optimal block of variables $(\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{G}})$ or $(\mathbf{E}_{(1)},\mathbf{G}_{(1)})$ by minimizing $f(\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{G}})$, where $f(\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{G}}) = \lambda||\boldsymbol{\mathcal{E}}||_1 + \tau||\boldsymbol{\mathcal{G}}||_F^2 + \frac{\mu}{2}\sum_{n=1}^{N}||(\boldsymbol{\mathcal{D}}-\boldsymbol{\mathcal{M}}_n+\boldsymbol{\mathcal{Y}}_n/\mu)-(\boldsymbol{\mathcal{G}}+\boldsymbol{\mathcal{E}})||_F^2$. The partial derivative of $f(\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{G}})$ with respect to $\boldsymbol{\mathcal{G}}$ is

$$\frac{\partial f}{\partial \boldsymbol{\mathcal{G}}} = 2\tau\boldsymbol{\mathcal{G}} + \mu\sum_{n=1}^{N}((\boldsymbol{\mathcal{G}}+\boldsymbol{\mathcal{E}})-(\boldsymbol{\mathcal{D}}-\boldsymbol{\mathcal{M}}_n+\boldsymbol{\mathcal{Y}}_n/\mu)). \tag{23}$$

By letting $\frac{\partial f}{\partial \boldsymbol{\mathcal{G}}} = \mathbf{0}$, we have

$$\boldsymbol{\mathcal{G}} := \delta\boldsymbol{\mathcal{T}} - \delta N\boldsymbol{\mathcal{E}}, \tag{24}$$

where $\delta = \frac{\mu}{2\tau+\mu N}$, $\boldsymbol{\mathcal{T}} = \sum_{n=1}^{N}(\boldsymbol{\mathcal{D}}-\boldsymbol{\mathcal{M}}_n+\boldsymbol{\mathcal{Y}}_n/\mu)$. By substituting Equation (24) into $f(\boldsymbol{\mathcal{E}},\boldsymbol{\mathcal{G}})$, we have the update formulation of $\boldsymbol{\mathcal{E}}$:

$$\begin{aligned}&\boldsymbol{\mathcal{E}} := \arg\min_{\boldsymbol{\mathcal{E}}} \lambda||\boldsymbol{\mathcal{E}}||_1 + \tau||\delta\boldsymbol{\mathcal{T}}-\delta N\boldsymbol{\mathcal{E}}||_F^2 + \frac{\mu}{2}\sum_{n=1}^{N}||(\boldsymbol{\mathcal{D}}-\boldsymbol{\mathcal{M}}_n+\boldsymbol{\mathcal{Y}}_n/\mu)-\boldsymbol{\mathcal{E}}-\delta\boldsymbol{\mathcal{T}}+\delta N\boldsymbol{\mathcal{E}}||_F^2\\ &= \arg\min_{\boldsymbol{\mathcal{E}}} \lambda||\boldsymbol{\mathcal{E}}||_1 + \tau\delta^2N^2||\boldsymbol{\mathcal{E}}-\tfrac{1}{N}\boldsymbol{\mathcal{T}}||_F^2 + \frac{\mu(1-\delta N)^2N}{2}||\boldsymbol{\mathcal{E}}-\frac{1}{(1-\delta N)N}\sum_{n=1}^{N}(\boldsymbol{\mathcal{D}}-\boldsymbol{\mathcal{M}}_n+\boldsymbol{\mathcal{Y}}_n/\mu-\delta\boldsymbol{\mathcal{T}})||_F^2\\ &= \arg\min_{\boldsymbol{\mathcal{E}}} \lambda||\boldsymbol{\mathcal{E}}||_1 + \tau\delta^2N^2||\tfrac{1}{N}\boldsymbol{\mathcal{T}}-\boldsymbol{\mathcal{E}}||_F^2 + \frac{\mu(1-\delta N)^2N}{2}||\boldsymbol{\mathcal{E}}-\tfrac{1}{N}\boldsymbol{\mathcal{T}}||_F^2\\ &= \arg\min_{\boldsymbol{\mathcal{E}}} \lambda||\boldsymbol{\mathcal{E}}||_1 + \frac{\tau\delta^2N^2+\mu(1-\delta N)^2N}{2}||\boldsymbol{\mathcal{E}}-\tfrac{1}{N}\boldsymbol{\mathcal{T}}||_F^2\\ &= S_{(\lambda/(\tau\delta^2N^2+\mu(1-\delta N)^2N))}\left(\tfrac{1}{N}\boldsymbol{\mathcal{T}}\right).\end{aligned} \tag{25}$$

A standard extension of the classic ADMM is to use varying parameter μ for each iteration. The goal of this extension is to improve the convergence and reduce the dependence on the initial choice of μ. In the context of the method of multipliers, this approach is proven to be superlinearly convergent if $\mu^{(k)} \to +\infty$ [25], which inspires us to adopt the non-decreasing sequence $\left\{\mu^{(k)}\right\}$. Furthermore, large values of μ result in a large penalty on violations of primal feasibility and are thus inclined to produce small primal residuals. However, it is difficult to prove the convergence of ADMM [24]. A commonly-used choice for $\left\{\mu^{(k)}\right\}$ is $\mu^{(k+1)} := \min(\rho\mu^{(k)},\mu_{max})$, where $\rho > 1$ and μ_{max} is the upper limit of μ. Due to the fact that $\mu^{(k)}$ is fixed after a finite number of iterations, the corresponding ADMM is convergent according to Theorem 1.

6. Experimental Results

We perform experiments on synthetic data and two real-world video sequences, and validate the feasibility and effectiveness of the proposed method. The experimental results of GLRTR are compared with that of MRPCA, where the missing values in MRCPA are replaced by zeros.

6.1. Synthetic Data

In this subsection, we synthesize data tensors with missing entries. First, we generate an Nth-order low-rank tensor as follows: $\mathcal{A} = \mathcal{C} \times_{n=1}^{N} \mathbf{U}^{(n)}$, with the core tensor $\mathcal{C} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ and mode matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times J_n} (J_n < I_n)$. The entries of $\mathcal{C}$ and $\mathbf{U}^{(n)}$ are independently drawn from the standard normal distribution. Then, we generate randomly a dense noise tensor $\mathcal{G} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ whose entries also obey the standard normal distribution. Next, we construct a sparse noise tensor $P_{\Omega'}(\mathcal{E})$, where $\mathcal{E} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is produced by a uniform distribution on the interval $(-a, a)$ and the index set Ω' is produced by uniformly sampling on $[I_1] \times [I_2] \times \cdots \times [I_N]$ with probability p′%. Finally, the generation of the sampling index set Ω is similar to Ω' and the corresponding sampling rate is set to be p%. Therefore, an incomplete data tensor is synthesized as $\mathcal{D}_\Omega = P_\Omega(\mathcal{A} + \mathcal{G} + \mathcal{E})$.

For given data tensor $\mathcal{D}_\Omega$ with missing values, its low-rank component recovered by some method is denoted by $\hat{\mathcal{A}}$. The Relative Error (RE) is employed to evaluate the recovery performance of the low-rank structure and its definition is given as follows: $\text{RE} = ||\hat{\mathcal{A}} - \mathcal{A}||_F / ||\mathcal{A}||_F$. Small relative error means good recovery performance. The experiments are carried out on $50 \times 50 \times 50$ tensors and $20 \times 20 \times 20 \times 20$ tensors, respectively. Furthermore, we set a = 500 and $J_1 = J_2 = \cdots = J_N = r$.

For convenience of comparison, we design three groups of experiments. In the first group of experiments, we only consider the case that there are no missing entries, that is, $\Omega = [I_1] \times [I_2] \times \cdots \times [I_N]$ or p = 100. The values of $||\mathcal{E}||_F / ||\mathcal{A}||_F$ and $||\mathcal{G}||_F / ||\mathcal{A}||_F$ are adopted to indicate the Inverse Signal-to-Noise Ratio (ISNR) with respect to the sparse and the Gaussian noise respectively. Three different degrees of sparsity are taken into account, that is, p′ = 5, 10 and 15. In addition, we take $\lambda = 0.025, \tau = 0.02, r \in \{3, 5\}$ for 3rd-order tensors and $\lambda = 0.03, \tau = 0.01, r \in \{2, 4\}$ for 4th-order tensors. For given parameters, we repeat the experiments ten times and report the average results. As a low-rank approximation method for tensors, the Higher-Order SVD (HOSVD) truncation [12] to rank-$(r_1, r_2, \ldots, r_N)$ is not suitable for gross corruptions owing to the fact that its relative error reaches up to 97% or even 100%. Hence, we do not compare this method with our GLRTR in subsequent experiments. The experimental results are shown in Tables 1 and 2 respectively.

Table 1. Comparison of experimental results on $50 \times 50 \times 50$ tensors.

(r, p′)	$\|\|\mathcal{E}\|\|_F / \|\|\mathcal{A}\|\|_F$ (%)	$\|\|\mathcal{G}\|\|_F / \|\|\mathcal{A}\|\|_F$ (%)	RE of MRPCA (%)	RE of GLRTR (%)
(3, 5)	1253	18.33	9.71 ± 0.98	6.60 ± 0.69
(3, 10)	1769	19.20	8.04 ± 0.84	7.99 ± 0.84
(3, 15)	2218	20.05	9.84 ± 1.97	8.75 ± 1.87
(5, 5)	569	8.79	7.39 ± 0.79	3.74 ± 0.42
(5, 10)	855	9.40	8.55 ± 0.97	4.47 ± 0.52
(5, 15)	1040	9.16	9.13 ± 0.52	5.05 ± 0.31

Table 2. Comparison of experimental results on $20 \times 20 \times 20 \times 20$ tensors.

(r, p′)	$\|\|\mathcal{E}\|\|_F / \|\|\mathcal{A}\|\|_F$ (%)	$\|\|\mathcal{G}\|\|_F / \|\|\mathcal{A}\|\|_F$ (%)	RE of MRPCA (%)	RE of GLRTR (%)
(2, 5)	1722	24.05	29.24 ± 7.95	13.90 ± 4.68
(2, 10)	2823	37.72	48.07 ± 18.47	23.98 ± 9.66
(2, 15)	3052	27.17	44.71 ± 13.66	23.10 ± 8.52
(4, 5)	448	6.97	12.00 ± 1.60	8.96 ± 1.15
(4, 10)	563	6.04	15.12 ± 1.98	6.26 ± 0.87
(4, 15)	758	6.58	24.22 ± 2.61	12.16 ± 2.90

From the above two tables, we have the following observations: (I) Although the values of ISNR on sparse noise are very large, both MRPCA and GLRTR remove efficiently sparse noise to some extent. Meanwhile, large values of ISNR on Gaussian noise are disadvantageous for recovering the low-rank components; (II) GLRTR has better recovery performance than MRPCA. In an average sense, the relative error of GLRTR is 2.68% smaller than that of MRPCA for $50 \times 50 \times 50$ tensors, and 14.17% for

$20 \times 20 \times 20 \times 20$ tensors; (III) For 3rd-order tensors, GLRTR removes effectively Gaussian noise, and, on average, its relative error is 5.96% smaller than the value of ISNR on Gaussian noise; for 4th-order tensors, GLRTR effectively removes Gaussian noise only in the case $r = 2$. In summary, GLRTR is more effective than MRPCA in recovering the low-rank components.

The second group of experiments considers four different sampling rates for $\boldsymbol{\Omega}$ and one fixed degree of sparsity for $\boldsymbol{\Omega}'$, that is, $p \in \{\mathbf{30, 50, 70, 90}\}$ and $p' = 5$. We set $\boldsymbol{\tau} = 0.02$ for both 3rd-order and 4th-order tensors, and choose the superior tradeoff parameter $\boldsymbol{\lambda}$ for each p. The comparisons of experimental results between MRPCA and GLRTR are shown in Tables 3 and 4 respectively. We can see from these two tables that MRPCA is very sensitive to the sampling rate p%, and it hardly recovers the low-rank components. In contrast, GLRTR achieves better recovery performance for 3rd-order tensors. As for 4th-order tensors, it also has smaller relative error when the sampling rate p% is relatively large. These observations show that GLRTR is more robust to missing values than MRPCA.

Table 3. Comparison of experimental results on incomplete $50 \times 50 \times 50$ tensors for $r = 5$.

(p, λ)	(30,0.07)	(50, 0.05)	(70, 0.03)	(90,0.03)
RE of MRPCA (%)	87.81 ± 0.27	77.02 ± 0.46	60.64 ± 1.77	11.49 ± 1.73
RE of GLRTR (%)	11.58 ± 1.72	6.62 ± 0.82	5.51 ± 0.83	4.27 ± 0.47

Table 4. Comparison of experimental results on incomplete $20 \times 20 \times 20 \times 20$ tensors for $r = 4$.

(p, λ)	(30, 0.03)	(50, 0.035)	(70, 0.03)	(90, 0.03)
RE of MRPCA (%)	88.38 ± 0.25	78.94 ± 0.34	65.06 ± 0.59	40.03 ± 1.34
RE of GLRTR (%)	62.51 ± 1.72	20.09 ± 2.63	9.14 ± 0.12	8.58 ± 1.11

We will evaluate the sensitivity of GLRTR to the choice of $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$ in the last group of experiments. For convenience of designing experiments, we only perform experiments on $50 \times 50 \times 50$ tensors and consider the case that $p = 100$. The values of $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$ are set according to the following manner: we vary the value of one parameter while letting the other be fixed. In the first case, the parameter $\boldsymbol{\tau}$ is chosen as 0.01. Under this circumstance, the relative errors versus different $\boldsymbol{\lambda}$ of MRPCA and GLRTR are shown in Figure 1. We take $\boldsymbol{\lambda} = 0.01$ in the second case and the relative errors *versus* different $\boldsymbol{\tau}$ of GLRTR are shown in Figure 2.

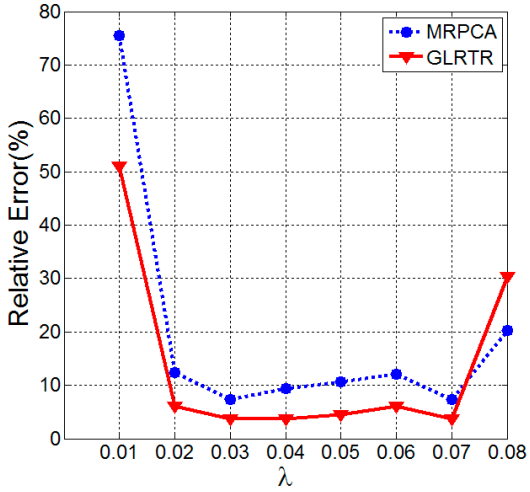

Figure 1. Comparison of relative errors between Multi-linear Robust Principal Component Analysis (MRPCA) and Generalized Low-Rank Tensor Recovery (GLRTR) with varying $\boldsymbol{\lambda}$.

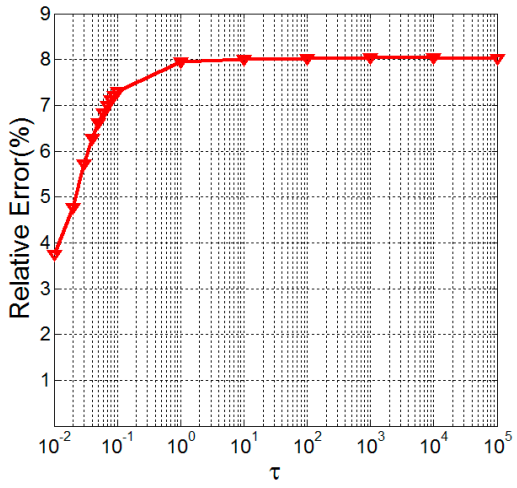

Figure 2. Relative errors of Generalized Low-Rank Tensor Recovery (GLRTR) with varying $\boldsymbol{\tau}$.

It can be seen from Figure 1 that the relative errors of MRPCA and GLRTR are about 9.90% and 4.62%, respectively, if $0.02 \leqslant \boldsymbol{\lambda} \leqslant 0.07$, which means the latter has better recovery performance than the former. Furthermore, their relative errors are relatively stable when $\boldsymbol{\lambda}$ lies within a certain interval. Figure 2 illustrates that the relative error has the tendency to increase monotonically, and it becomes almost stationary when $\boldsymbol{\tau} \geqslant 1$. At this moment, the relative errors lie in the interval (0.037, 0.080). This group of experiments implies that, for our synthetic data, GLRTR is not very sensitive to the choice of $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$.

6.2. Influence of Noise and Sampling Rate on the Relative Error

This subsection will evaluate the influence of noise and sampling rate on the relative error. For this purpose, we design four groups of experiments and use the synthetic data generated in the same manner as in the previous subsection. For the sake of convenience, we only carry out experiments on $50 \times 50 \times 50$ tensors.

The first group of experiments aims to investigate the influence of noise on the recovery performance. In the data generation process, we only change the manner for generating $\mathcal{G}$, that is, each entry of $\mathcal{G}$ is drawn independently from the normal distribution with mean zero and standard deviation b. Let r = 5,p′ = 10,p = 100,a = 50i and b = 0.2j, where i,j ∈ {0,1,...,10}. For different combinations of a and b, the relative errors of GLRTR are shown in Figure 3. We can draw two conclusions from this figure. For given b, the relative error is relatively stable with the increasing of a, which means the relative error is not very sensitive to the magnitude of sparse noise. The relative error monotonically increases with the increasing of Gaussian noise level, which validates that large Gaussian noise is disadvantageous for recovering the low-rank component.

Next, we study the influence of sampling rate p% on the relative error for different r. Set p′ = 10,a = 500,b = 1,r = 1 + 2i,i = 1,2,...,14 and vary the value of p% from 30 to 100 in steps of size 10. For fixed r and p, we obtain the low-rank component according to GLRTR and then plot the relative errors in Figure 4. From the 3-D colored surface in Figure 4, we can see that both r and p have significant influence on the relative error. This observation indicates that small r or large p is conducive to the recovery of low-rank term.

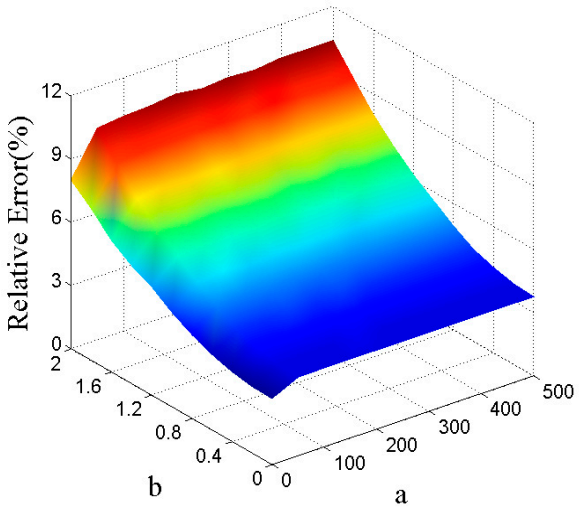

Figure 3. Relative errors for different a and b.

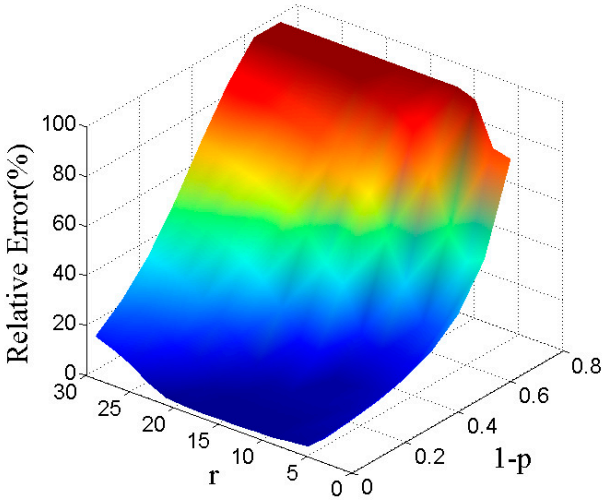

Figure 4. Relative errors for different r and p.

The third group of experiments will validate the robustness of GLRTR to sparse noise. Concretely speaking, we investigate the recovery performance under different ISNR on sparse noise without consideration of Gaussian noise. Set r $\in$ { 3,5 },p′ = 10,p = 100,b = 0, a = $2^{i} \times 500$, i = $-15, -14, \ldots, 10$. The experimental results are shown in Figure 5, where the horizontal and the vertical coordinates represent the ISNR on sparse noise and the relative error, respectively. This figure illustrates that the relative error is less than 4.5% for synthetic 3rd-order tensors, which verifies experimentally that our method is very robust to sparse noise.

In the last group of experiments, we discuss the performance of GLRTR in removing the small dense noise for given large sparse noise. We also propose a combination strategy: GLRTR + HOSVD, that is, GLRTR is followed by HOSVD. The goal of this new method is to improve the denoising performance of GLRTR. Let r $\in$ { 3,5 },p′ = 10,p = 100,a = 500 and b = 0.1i,i = $0, 1, \ldots, 50$. Different values for b lead to different ISNR on Gaussian noise. We draw four curves to reflect the relationship between the relative error and ISNR on Gaussian noise, as shown in Figure 6, where the black dashed line is a reference line. We have two observations from this figure. When ISNR on Gaussian noise is larger than 3.5%, GLRTR not only successfully separates the sparse noise to some extent but also effectively removes the Gaussian noise. The GLRTR+HOSVD method has better denoising performance than GLRTR in the presence of large Gaussian noise.

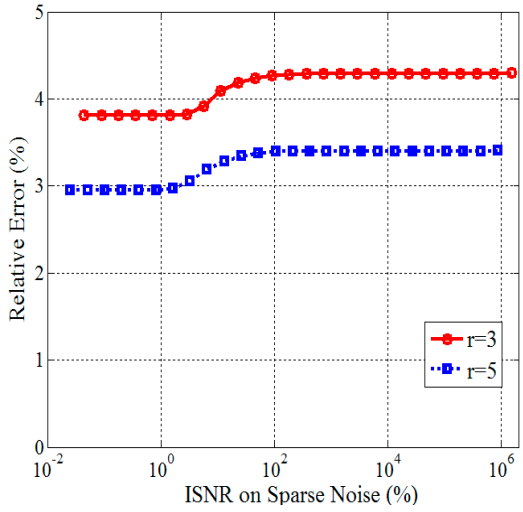

Figure 5. Relative errors *versus* different Inverse Signal-to-Noise Ratio (ISNR) on sparse noise.

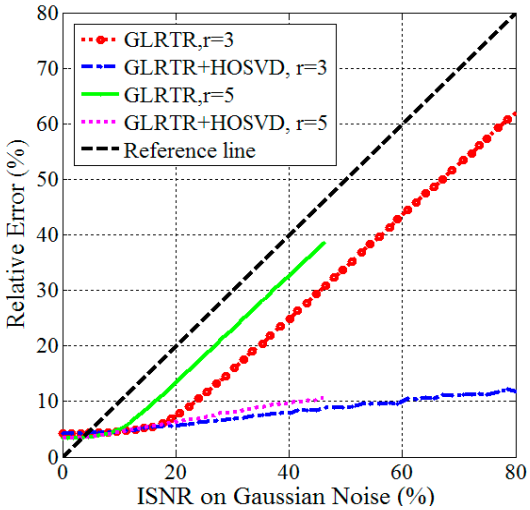

Figure 6. Relative errors *versus* different Inverse Signal-to-Noise Ratio (ISNR) on Gaussian noise.

6.3. Applications in Background Modeling

In this subsection, we test our method on two real-world surveillance videos for object detection and background subtraction: Lobby and Bootstrap datasets [26]. For convenience of computation, we only consider the first 200 frames for each dataset and transform the color images into the gray-level images. The resolutions of each image in the Lobby and Bootstrap datasets are 128 $\times$ 160 and 120 $\times$ 160, respectively. We add Gaussian noise with mean zero and standard deviation 5 to each image. Hence, we obtain two data tensors of order 3 and their sizes are 128 $\times$ 160 $\times$ 200 and 120 $\times$ 160 $\times$ 200, respectively. For two given tensors, we execute random sampling on them with a probability of 50%.

Considering the fact that MRPCA fails in recovering the low-rank components on the synthetic data with missing values, we only implement the method of GLRTR on the video datasets. Two tradeoff parameters are set as follows: $\boldsymbol{\lambda}$ = 0.0072 and $\boldsymbol{\tau}$ = 0.001. We can obtain the low-rank, the sparse and the completed components from the incomplete data tensors according to the proposed method. Actually, the low-rank terms are the backgrounds and the sparse noise terms correspond to the foregrounds. The experimental results are partially shown in Figures 7 and 8 respectively, where the missing entries in incomplete images are shown in white. From Figures 7 and 8 we can see that

GLRTR can recover efficiently the low-rank images and the sparse noise images. Moreover, we observe from the recovered images that a large proportion of missing entries are effectively completed.

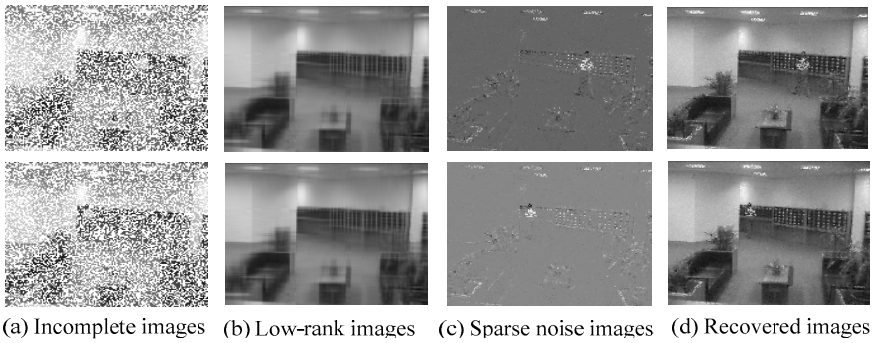

Figure 7. Background modeling from lobby video. (**a**) draws the images with missing entries, (**b**) plots the recovered background, *i.e.*, the low-rank images, (**c**) shows the recovered foreground, *i.e.*, the sparse noised images, and (**d**) displays the recovered images.

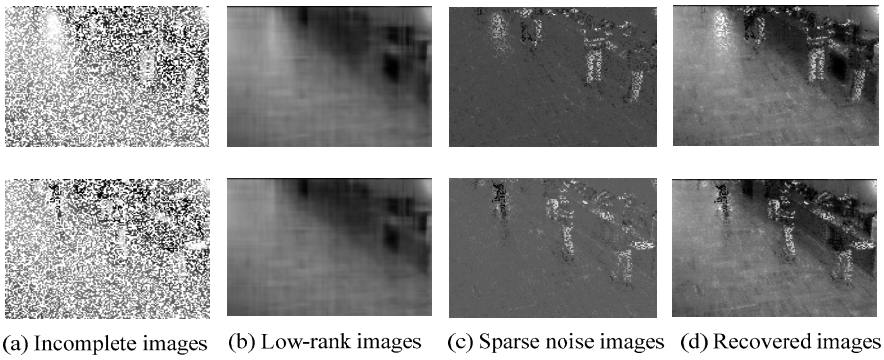

Figure 8. Background modeling from bootstrap video. (**a**) draws the images with missing entries, (**b**) plots the recovered background, *i.e.*, the low-rank images, (**c**) shows the recovered foreground, *i.e.*, the sparse noised images, and (**d**) diplays the recovered images.

To evaluate the completion performance of GLRTR, we define the Relative Approximate Error (RAE) as RAE $= \ ||\hat{\boldsymbol{\mathcal{D}}} - \boldsymbol{\mathcal{D}}||_{\mathrm{F}}/||\boldsymbol{\mathcal{D}}||_{\mathrm{F}}$, where $\boldsymbol{\mathcal{D}}$ is the original video tensor without Gaussian noise corruptions and missing entries, and $\hat{\boldsymbol{\mathcal{D}}}$ is the approximated term of $\boldsymbol{\mathcal{D}}$. The RAE of the Lobby dataset is 8.94% and that of the Bootstrap dataset is 20.11%. These results demonstrate GLRTR can complete approximately the missing entries to a certain degree. There are two reasons for that the Bootstrap dataset has relatively large RAE: one is its more complex foreground and the other is that the entries of the foreground can not be recovered when they are missing. In summary, GLRTR is robust to gross corruption, Gaussian noise and missing values.

7. Conclusions

In this paper, we investigate a generalized model of LRTR in which large sparse corruption, missing entries and Gaussian noise are taken into account. For the generalized LRTR, we establish an optimization problem that minimizes the weighted combination of the tensor nuclear norm, the l_1 norm and the Frobenius norm. To address this minimization problem, we present an iterative scheme based on the technique of ADMM. The experimental results on synthetic data and real-world video datasets illustrate that the proposed method is efficient and feasible in recovering the low-rank components and completing missing entries. In the future, we will consider the theoretical conditions for exact recoverability and other scalable algorithms.

Acknowledgments: This work is partially supported by the National Natural Science Foundation of China (No. 61403298, No. 11526161 and No. 11401357), and the Natural Science Basic Research Plan in Shaanxi Province of China (No. 2014JQ8323).

Author Contributions: Jiarong Shi constructed the model, developed the algorithm and wrote the manuscript. Qingyan Yin designed the experiments. Xiuyun Zheng and Wei Yang implemented all experiments. All four authors were involved in organizing and refining the manuscript. All authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Candès, E.J.; Recht, B. Exact matrix completion via convex optimization. *Found. Comput. Math.* **2009**, *9*, 717–772. [CrossRef]
2. Candès, E.J.; Li, X.; Ma, Y.; Wright, J. Robust principal component analysis? *J. ACM.* **2011**, *58*, 37. [CrossRef]
3. Liu, G.; Lin, Z.; Yan, S.; Sun, J.; Yu, Y.; Ma, Y. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 171–184. [CrossRef] [PubMed]
4. Candès, E.J.; Plan, Y. Matrix completion with noise. *P. IEEE.* **2010**, *98*, 925–936. [CrossRef]
5. Zhou, Z.; Li, X.; Wright, J.; Candès, E.J.; Ma, Y. Stable principal component pursuit. In Proceedings of the 2010 IEEE International Symposium on Information Theory Proceedings (ISIT), Austin, TX, USA, 13–18 June 2010.
6. Xu, H.; Caramanis, C.; Sanghavi, S. Robust PCA via outlier pursuit. *IEEE Trans. Inf. Theory.* **2012**, *58*, 3047–3064. [CrossRef]
7. Shi, J.; Zheng, X.; Yong, L. Incomplete robust principal component analysis. *ICIC Express Letters, Part B Appl.* **2014**, *5*, 1531–1538.
8. Shi, J.; Yang, W.; Yong, L.; Zheng, X. Low-rank representation for incomplete data. *Math. Probl. Eng.* **2014**, *10*. [CrossRef]
9. Liu, Y.; Jiao, L.C.; Shang, F.; Yin, F.; Liu, F. An efficient matrix bi-factorization alternative optimization method for low-rank matrix recovery and completion. *Neural Netw.* **2013**, *48*, 8–18. [CrossRef] [PubMed]
10. Liu, Y.; Jiao, L.C.; Shang, F. A fast tri-factorization method for low-rank matrix recovery and completion. *Pattern Recogn.* **2013**, *46*, 163–173. [CrossRef]
11. De Lathauwer, L.; Castaing, J. Tensor-based techniques for the blind separation of DS–CDMA signals. *Signal Process.* **2007**, *87*, 322–336. [CrossRef]
12. Kolda, T.G.; Bader, B.W. Tensor decompositions and applications. *SIAM Rev.* **2009**, *51*, 455–500. [CrossRef]
13. Liu, J.; Musialski, P.; Wonka, P.; Ye, J. Tensor completion for estimating missing values in visual data. *IEEE Trans. Patt. Anal. Mach. Intel.* **2013**, *35*, 208–220. [CrossRef] [PubMed]
14. Shi, J.; Zhou, S.; Zheng, X. Multilinear robust principal component analysis. *Acta Electronica Sinica.* **2014**, *42*, 1480–1486.
15. Tomasi, G.; Bro, R. PARAFAC and missing values. *Chemom. Intell. Lab. Syst.* **2005**, *75*, 163–180. [CrossRef]
16. Shi, J.; Jiao, L.C.; Shang, F. Tensor completion algorithm and its applications in face recognition. *Pattern Recognit. Artif. Intell.* **2011**, *24*, 255–261.
17. Kressner, D.; Steinlechner, M.; Vandereycken, B. Low-rank tensor completion by Riemannian optimization. *BIT Numer. Math.* **2014**, *54*, 447–468. [CrossRef]
18. Shi, J.; Yang, W.; Yong, L.; Zheng, X. Low-rank tensor completion via Tucker decompositions. *J. Comput. Inf. Syst.* **2015**, *11*, 3759–3768.
19. Gandy, S.; Recht, B.; Yamada, I. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inv. Probl.* **2011**, *27*, 19. [CrossRef]
20. Liu, Y.; Shang, F. An efficient matrix factorization method for tensor completion. *IEEE Signal Process. Lett.* **2013**, *20*, 307–310. [CrossRef]
21. Tan, H.; Cheng, B.; Wang, W.; Zhang, Y.J.; Ran, B. Tensor completion via a multi-linear low-n-rank factorization model. *Neurocomputing.* **2014**, *133*, 161–169. [CrossRef]
22. Goldfarb, D.; Qin, Z. Robust low-rank tensor recovery: Models and algorithms. *SIAM J. Matrix Anal. Appl.* **2014**, *35*, 225–253. [CrossRef]
23. Cai, J.F.; Candès, E.J.; Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM J. Optimiz.* **2010**, *20*, 1956–1982. [CrossRef]

24. Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **2011**, *3*, 1–122. [CrossRef]
25. Rockafellar, R.T. Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.* **1976**, *14*, 877–898. [CrossRef]
26. Databases of Lobby and Bootstrap. Available online: http://perception.i2r.a-star.edu.sg/bk_ model/bk_index.html (accessed on 1 November 2015).

© 2016 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).