Inverse Problems and Imaging
Volume 9, No. 2, 2015, 601–624
doi:10.3934/ipi.2015.9.601

PARALLEL MATRIX FACTORIZATION FOR LOW-RANK TENSOR COMPLETION

Yangyang Xu

Department of Computational and Applied Mathematics
Rice University
Houston, TX 77005, USA

Ruru Hao

School of Mathematical Sciences
Dalian University of Technology, Dalian, China

Wotao Yin

Department of Mathematics, University of California
Los Angeles, CA 90095, USA

Zhixun Su

School of Mathematical Sciences
Dalian University of Technology, Dalian, China

(Communicated by Weihong Guo)

Abstract. Higher-order low-rank tensors naturally arise in many applications including hyperspectral data recovery, video inpainting, seismic data reconstruction, and so on. We propose a new model to recover a low-rank tensor by simultaneously performing low-rank matrix factorizations to the all-mode matricizations of the underlying tensor. An alternating minimization algorithm is applied to solve the model, along with two adaptive rank-adjusting strategies when the exact rank is not known.

Phase transition plots reveal that our algorithm can recover a variety of synthetic low-rank tensors from significantly fewer samples than the compared methods, which include a matrix completion method applied to tensor recovery and two state-of-the-art tensor completion methods. Further tests on real-world data show similar advantages. Although our model is non-convex, our algorithm performs consistently throughout the tests and gives better results than the compared methods, some of which are based on convex models. In addition, subsequence convergence of our algorithm can be established in the sense that any limit point of the iterates satisfies the KKT condtions.

1. **Introduction.** *Tensor* is a generalization of *vector* and *matrix*. A vector is a first-order (also called one-way or one-mode) tensor, and a matrix is a second-order tensor. Higher-order tensor arises in many applications such as 3D image reconstruction [22], video inpainting [19], hyperspectral data recovery [14, 28], higher-order web link analysis [10], personalized web search [23], and seismic data reconstruction [11]. In this paper, we focus on the recovery of higher-order tensors that are (exactly or approximately) low-rank and have missing entries. We dub the problem as *low-rank tensor completion* (LRTC). The introduced model and algorithm can be

2010 *Mathematics Subject Classification.* Primary: 94A08, 94A12; Secondary: 90C90.

Key words and phrases. Higher-order tensor, low-rank matrix completion, low-rank tensor completion, alternating least squares, non-convex optimization.

extended in a rather straightforward way to recovering low-rank tensors from their linear measurements.

LRTC can be regarded as an extension of low-rank matrix completion [2]. To recover a low-rank tensor from its partially observed entries, one can unfold it into a matrix and apply a low-rank matrix completion algorithm such as FPCA [17], APGL [24], LMaFit [27], the alternating direction method [3, 31], the ℓ_q minimization method [13], and so on. However, this kind of method utilizes only one mode low-rankness of the underlying tensor. We are motivated and convinced by the results [16] that utilizing all mode low-ranknesses of the tensor gives much better performance.

Existing methods for LRTC in [16, 4] employ matrix nuclear-norm minimization and use the singular value decomposition (SVD) in their algorithms, which become very slow or even not applicable for large-scale problems. To tackle this difficulty, we apply *low-rank matrix factorization* to each mode unfolding of the tensor in order to enforce low-rankness and update the matrix factors alternatively, which is computationally much cheaper than SVD.

Our approach is non-convex, and the sizes of the matrix factors must be specified in the algorithm. Non-convexity makes it difficult for us to predict the performance of our approach in a theoretical way, and in general, the performance can vary to the choices of algorithm and starting point. We found cyclic updates of the unknown variables in the model to perform well enough. The sizes of the matrix factors dictate the rank of the recovered tensor. If they are fixed to values significantly different from the true rank, the recovery can overfit or underfit. On the other hand, during the run time of our algorithms, there are simple ways to adaptively adjust the factor sizes. In short, the all-mode matricizations, cyclic block minimization, and adaptive adjustment are the building blocks of our approach.

Before introducing our model and algorithm, we review some notation and tensor operations.

1.1. **Notation.** Following [9], we use bold lower-case letters $\mathbf{x}, \mathbf{y}, \ldots$ for vectors, bold upper-case letters $\mathbf{X}, \mathbf{Y}, \ldots$ for matrices, and bold caligraphic letters $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}, \ldots$ for tensors. The $(i_1, \ldots, i_N)$-th component of an N-way tensor $\boldsymbol{\mathcal{X}}$ is denoted as $x_{i_1 \ldots i_N}$. For $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, we define their inner product in the same way as that for matrices, i.e.,

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1 \ldots i_N} y_{i_1 \ldots i_N}.$$

The Frobenius norm of $\boldsymbol{\mathcal{X}}$ is defined as $\|\boldsymbol{\mathcal{X}}\|_F = \sqrt{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle}$.

A *fiber* of $\boldsymbol{\mathcal{X}}$ is a vector obtained by fixing all indices of $\boldsymbol{\mathcal{X}}$ except one, and a *slice* of $\boldsymbol{\mathcal{X}}$ is a matrix by fixing all indices of $\boldsymbol{\mathcal{X}}$ except two. For example, if $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{2\times 2\times 2}$ has two frontal slices (with the third index fixed)

$$\mathbf{X}(:,:,1) = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}, \quad \mathbf{X}(:,:,2) = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}, \tag{1}$$

then $[1,2]^\top$ is a mode-1 fiber (with all but the first indices fixed), $[1,3]^\top$ is a mode-2 fiber (with all but the second indices fixed), and $[1,5]^\top$ is a mode-3 fiber (with all but the third indices fixed). Its two horizontal (with the first index fixed) and two

lateral slices (with the second index fixed) are respectively

$$\mathbf{X}(1,:,:)=\begin{bmatrix}1 & 5\\ 3 & 7\end{bmatrix},\mathbf{X}(2,:,:)=\begin{bmatrix}2 & 6\\ 4 & 8\end{bmatrix},\ \text{and}\ \mathbf{X}(:,1,:)=\begin{bmatrix}1 & 5\\ 2 & 6\end{bmatrix},\mathbf{X}(:,2,:)=\begin{bmatrix}3 & 7\\ 4 & 8\end{bmatrix}.$$

The mode-n *matricization* (also called *unfolding*) of $\boldsymbol{\mathcal{X}}\in\mathbb{R}^{I_1\times\cdots\times I_N}$ is denoted as $\mathbf{X}_{(n)}\in\mathbb{R}^{I_n\times\Pi_{j\neq n}I_j}$, which is a matrix with columns being the mode-n fibers of $\boldsymbol{\mathcal{X}}$ in the lexicographical order. Take the tensor in (1) for example. Its mode-1 and mode-3 matricizations are respectively

$$X_{(1)}=\begin{bmatrix}1 & 3 & 5 & 7\\ 2 & 4 & 6 & 8\end{bmatrix},\ \text{and}\ X_{(3)}=\begin{bmatrix}1 & 2 & 3 & 4\\ 5 & 6 & 7 & 8\end{bmatrix}.$$

Relating to the matricization process, we define $\mathbf{unfold}_n(\boldsymbol{\mathcal{X}})=\mathbf{X}_{(n)}$ and $\mathbf{fold}_n$ to reverse the process, i.e., $\mathbf{fold}_n(\mathbf{unfold}_n(\boldsymbol{\mathcal{X}}))=\boldsymbol{\mathcal{X}}$. The n-rank of an N-way tensor $\boldsymbol{\mathcal{X}}$, denoted as $\text{rank}_n(\boldsymbol{\mathcal{X}})$, is the rank of $\mathbf{X}_{(n)}$, and we define the rank[1] of $\boldsymbol{\mathcal{X}}$ as an array: $\text{rank}(\boldsymbol{\mathcal{X}})=(\text{rank}(\mathbf{X}_{(1)}),\ldots,\text{rank}(\mathbf{X}_{(N)}))$. We say $\boldsymbol{\mathcal{X}}$ is (approximately) low-rank if $\mathbf{X}_{(n)}$ is (approximately) low-rank for all n.

1.2. **Problem formulation.** We aim at recovering an (approximately) low-rank tensor $\boldsymbol{\mathcal{M}}\in\mathbb{R}^{I_1\times\cdots\times I_N}$ from partial observations $\boldsymbol{\mathcal{B}}=\mathcal{P}_\Omega(\boldsymbol{\mathcal{M}})$, where Ω is the index set of observed entries, and $\mathcal{P}_\Omega$ keeps the entries in Ω and zeros out others. We apply low-rank matrix factorization to each mode unfolding of $\boldsymbol{\mathcal{M}}$ by finding matrices $\mathbf{X}_n\in\mathbb{R}^{I_n\times r_n},\mathbf{Y}_n\in\mathbb{R}^{r_n\times\Pi_{j\neq n}I_j}$ such that $\mathbf{M}_{(n)}\approx\mathbf{X}_n\mathbf{Y}_n$ for $n=1,\ldots,N$, where r_n is the estimated rank, either fixed or adaptively updated. Introducing one common variable $\boldsymbol{\mathcal{Z}}$ to relate these matrix factorizations, we solve the following model to recover $\boldsymbol{\mathcal{M}}$

$$\min_{\mathbf{X},\mathbf{Y},\boldsymbol{\mathcal{Z}}}\sum_{n=1}^{N}\frac{\alpha_n}{2}\|\mathbf{X}_n\mathbf{Y}_n-\mathbf{Z}_{(n)}\|_F^2,\ \text{subject to}\ \mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}})=\boldsymbol{\mathcal{B}},\tag{2}$$

where $\mathbf{X}=(\mathbf{X}_1,\ldots,\mathbf{X}_N)$ and $\mathbf{Y}=(\mathbf{Y}_1,\ldots,\mathbf{Y}_N)$. In the model, α_n, $n=1,\ldots,N$, are weights and satisfy $\sum_n\alpha_n=1$. The constraint $\mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}})=\boldsymbol{\mathcal{B}}$ enforces consistency with the observations and can be replaced with $\|\mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}})-\boldsymbol{\mathcal{B}}\|_F\leq\delta$ if $\boldsymbol{\mathcal{B}}$ is contaminated by noise with a known Frobenius norm equal to δ. In this paper, we do not assume the knowledge of δ and thus use (2) for both noiseless and noisy cases.

The ranks $r_1,\ldots,r_N$ in (2) must be specified, yet we do not assume the knowledge of their true values. To address this issue, we dynamically adjust the rank estimates in two schemes. One scheme starts from overestimated ranks and then decreases them by checking the singular values of the factor matrices in each mode. When a large gap between the $\hat{r}_n$th and $(\hat{r}_n+1)$th singular values of the factors is found, r_n is reduced to $\hat{r}_n$. The other scheme starts from underestimated ranks and then gradually increases them if the algorithm detects slow progress.

We try to solve (2) by cyclically updating $\mathbf{X}$, $\mathbf{Y}$, and $\boldsymbol{\mathcal{Z}}$. Although a global solution is not guaranteed, we demonstrate by numerical experiments that our algorithm can reliably recover a wide variety of low-rank tensors. In addition, we show that any limit point of the iterates satisfies the KKT conditions.

The details will be given in Section 3.

[1]Our definition relates to the Tucker decomposition [26]. Another popularly used definition is based on the CANDECOMP/PARAFAC (CP) decomposition [7].

1.3. **Related work.** Our model (2) can be regarded as an extension of the following model [27] from matrix completion to tensor completion

$$\min_{\mathbf{X},\mathbf{Y}} \frac{1}{2}\|\mathbf{XY}-\mathbf{Z}\|_F^2, \text{ subject to } \mathcal{P}_\Omega(\mathbf{Z})=\mathbf{B}, \tag{3}$$

where $\mathbf{B}=\mathcal{P}_\Omega(\mathbf{M})$ contains partially observed entries of the underlying (approximately) low-rank matrix $\mathbf{M}$. If $N=2$ in (2), i.e., the underlying tensor $\boldsymbol{\mathcal{M}}$ is two-way, then it is easy to see that (2) reduces to (3) by noting $\mathbf{unfold}_1(\boldsymbol{\mathcal{M}}) = \mathbf{unfold}_2(\boldsymbol{\mathcal{M}})^\top$. The problem (3) is solved in [27] by a successive over-relaxation (SOR) method, named as LMaFit. Although (3) is non-convex, extensive experiments on both synthetic and real-world data demonstrate that (3) solved by LMaFit performs significantly better than nuclear norm[2] based convex models such as

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_*, \text{ subject to } \mathcal{P}_\Omega(\mathbf{Z})=\mathbf{B}, \tag{4}$$

where $\|\mathbf{Z}\|_*$ denotes the nuclear norm of $\mathbf{Z}$, defined as the sum of its singular values.

The work [16] generalizes (4) to the tensor case, and to recover the (approximately) low-rank tensor $\boldsymbol{\mathcal{M}}$, it proposes to solve

$$\min_{\boldsymbol{\mathcal{Z}}} \sum_{n=1}^{N} \alpha_n \|\mathbf{Z}_{(n)}\|_*, \text{ subject to } \mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}})=\boldsymbol{\mathcal{B}}, \tag{5}$$

where $\alpha_n \geq 0, n=1,\ldots,N$ are preselected weights satisfying $\sum_n \alpha_n = 1$. Different from our model (2), the problem (5) is convex, and in [16], various methods are applied to solve it such as block coordinate descent method, proximal gradient method, and alternating direction method of multiplier (ADMM). The model (5) utilizes low-rankness of all mode unfoldings of the tensor, and as demonstrated in [16], it can significantly improve the solution quality over that obtained by solving (4), where the matrix $\mathbf{Z}$ corresponds to some mode unfolding of the tensor.

The recent work [18] proposes a more "square" convex model for recovering $\boldsymbol{\mathcal{M}}$ as follows:

$$\min_{\boldsymbol{\mathcal{Z}}} \|\hat{\mathbf{Z}}_{[j]}\|_*, \text{ subject to } \mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}})=\boldsymbol{\mathcal{B}}, \tag{6}$$

where $\hat{\boldsymbol{\mathcal{Z}}} \in \mathbb{R}^{I_{i_1}\times\ldots\times I_{i_N}}$ is a tensor by relabeling mode i_n of $\boldsymbol{\mathcal{Z}}$ to mode n for $n=1,\ldots,N$,

$$\hat{\mathbf{Z}}_{[j]} = \text{reshape}\left(\hat{\mathbf{Z}}_{(1)}, \prod_{n\leq j} I_{i_n}, \prod_{n>j} I_{i_n}\right),$$

and j and the permutation $(i_1,\ldots,i_N)$ are chosen to make $\prod_{n\leq j} I_{i_n}$ as close as to $\prod_{n>j} I_{i_n}$. The idea of reshaping a tensor into a "square" matrix has also appeared in [6] for tensor principal component analysis. As the order of $\boldsymbol{\mathcal{M}}$ is no more than three, (6) is the same as (4) with $\mathbf{Z}$ corresponding to some mode unfolding of the tensor, and it may not perform as well as (5). However, for a low-rank tensor of more than three orders, it is shown in [18] that (6) can exactly recover the tensor from far fewer observed entries than those required by (5).

There are some other models proposed recently for LRTC. For example, the one in [21] uses, as a regularization term, a tight convex relaxation of the average rank function $\frac{1}{N}\sum_n \text{rank}_n(\boldsymbol{\mathcal{M}})$ and applies the ADMM method to solve the problem.

[2]The matrix nuclear norm is the convex envelope of matrix rank function [20], and the nuclear norm minimization can promote the low-rank structure of the solution.

The work [12] directly constrains the solution in some low-rank manifold and employs the Riemannian optimization to solve the problem. Different from the above discussed models that use tensor n-rank, the model in [32] employs the so-called *tubal-rank* based on the recently proposed tensor singular value decomposition (t-SVD) [8]. For details about these models, we refer the readers to the papers where they are proposed.

1.4. **Organization.** The rest of the paper is organized as follows. Section 2 shows the phase transition of our proposed method and some existing ones. Section 3 gives our algorithm with two different rank-adjusting strategies, and the convergence result of the algorithm is given in section 4. In section 5, we compare the proposed method with some state-of-the-art methods for tensor completion on both synthetic and real-world data. Section 6 conludes the paper, and finally, section 7 shows all figures and tables of our numerical results.

2. Phase transition plots.

A phase transition plot uses greyscale colors to depict how likely a certain kind of low-rank tensors can be recovered by an algorithm for a range of different ranks and sample ratios. Phase transition plots are important means to compare the performance of different tensor recovery methods.

We compare our method (called TMac) to the following three methods on random tensors of different kinds. In section 5, we compare them on the real-world data including 3D images and videos.

- Matrix completion method for recovering low-rank tensors: we unfold the underlying N-way tensor $\boldsymbol{\mathcal{M}}$ along its Nth mode and apply LMaFit [27] to (3), where $\mathbf{Z}$ corresponds to $\mathbf{unfold}_N(\boldsymbol{\mathcal{M}})$. If the output is $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$, then we use $\mathbf{fold}_N(\tilde{\mathbf{X}}\tilde{\mathbf{Y}})$ to estimate $\boldsymbol{\mathcal{M}}$.
- Nuclear norm minimization method for tensor completion: we apply FaLRTC [16] to (5) and use the output $\boldsymbol{\mathcal{Z}}$ to estimate $\boldsymbol{\mathcal{M}}$.
- Square deal method: we apply FPCA [17] to (6) and use the output $\boldsymbol{\mathcal{Z}}$ to estimate $\boldsymbol{\mathcal{M}}$.

We call the above three methods as MatComp, FaLRTC, and SquareDeal, respectively. We chose these methods due to their popularity and code availability. LMaFit has been demonstrated superior over many other matrix completion solvers such as APGL [24], SVT [1], and FPCA [17]; (5) appears to be the first convex model for tensor completion, and FaLRTC is the first efficient and also reliable[3] solver of (5); the work [18] about SquareDeal appears the first work to give theoretical guarantee for low-rank higher-order tensor completion. We set the stopping tolerance to 10^{-5} for all algorithms except FPCA that uses 10^{-8} since 10^{-5} appears too loose for FPCA. Note that the tolerances used here are tighter than those in section 5 because we care more about the models' recoverability instead of algorithms' efficiency.

If the relative error

$$\text{relerr} = \frac{\|\boldsymbol{\mathcal{M}}^{rec} - \boldsymbol{\mathcal{M}}\|_F}{\|\boldsymbol{\mathcal{M}}\|_F} \leq 10^{-2},$$

the recovery was regarded as successful, where $\boldsymbol{\mathcal{M}}^{rec}$ denotes the recovered tensor.

[3] In [16], the ADMM is also coded up for solving (5) and claimed to give high accurate solutions. However, we found that it was not as reliable as FaLRTC. In addition, if the smoothing parameter μ for FaLRTC was set small, FaLRTC could also produce solutions of high accuracy.

2.1. **Gaussian random data.** Two Gaussian random datasets were tested. Each tensor in the first dataset was 3-way and had the form $\mathcal{M} = \mathcal{C} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3$, where $\mathcal{C}$ was generated by MATLAB command `randn(r,r,r)` and $\mathbf{A}_n$ by `randn(50,r)` for $n = 1, 2, 3$. We generated Ω uniformly at random. The rank r varies from 5 to 35 with increment 3 and the sample ratio

$$\mathrm{SR} = \frac{|\Omega|}{\Pi_n I_n}$$

from 10% to 90% with increment 5%. In the second dataset, each tensor was 4-way and had the form $\mathcal{M} = \mathcal{C} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3 \times_4 \mathbf{A}_4$, where $\mathcal{C}$ was generated by MATLAB command `randn(r,r,r,r)` and $\mathbf{A}_n$ by `randn(20,r)` for $n = 1, 2, 3, 4$. The rank r varies from 4 to 13 with increment 1 and SR from 10% to 90% with increment 5%. For each setting, 50 independent trials were run.

Figure 1 depicts the phase transition plots of TMac, MatComp, and FaLRTC for the 3-way dataset, and Figure 2 depicts the phase transition plots of TMac and SquareDeal for the 4-way dataset. Since LMaFit usually works better than FPCA for matrix completion, we also show the result by applying LMaFit to

$$\min_{\mathbf{X},\mathbf{Y},\mathcal{Z}} \|\mathbf{XY} - \hat{\mathbf{Z}}_{[j]}\|_F^2, \text{ subject to } \mathcal{P}_\Omega(\mathcal{Z}) = \mathcal{B}, \tag{7}$$

where $\hat{\mathbf{Z}}_{[j]}$ is the same as that in (6). From the figures, we see that TMac performed much better than all the other compared methods. Note that our figure (also in Figures 5 and 7) for SquareDeal looks different from that shown in [18], because we fixed the dimension of $\mathcal{M}$ and varied the rank r while [18] fixes rank($\mathcal{M}$) to an array of very small values and varies the dimension of $\mathcal{M}$. The solver and the stopping tolerance also affect the results. For example, the "square" model (7) solved by LMaFit gives much better results but still worse than those given by TMac.

In addition, Figure 3 depicts the phase transition plots of TMac utilizing 1, 2, 3, and 4 modes of matricization on the 4-way dataset. We see that TMac can recover more tensors as it uses more modes.

2.2. **Uniformly random data.** This section tests the recoverability of TMac, MatComp, FaLRTC, and SquareDeal on two datasets in which the tensor factors have uniformly random entries. In the first dataset, each tensor had the form $\mathcal{M} = \mathcal{C} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3$, where $\mathcal{C}$ was generated by MATLAB command `rand(r,r,r)-0.5` and $\mathbf{A}_n$ by `rand(50,r)-0.5`. Each tensor in the second dataset had the form $\mathcal{M} = \mathcal{C} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3 \times_4 \mathbf{A}_4$, where $\mathcal{C}$ was generated by MATLAB command `rand(r,r,r,r)-0.5` and $\mathbf{A}_n$ by `rand(20,r)-0.5`. Figure 4 shows the recoverability of each method on the first dataset and Figure 5 on the second dataset. We see that TMac with both rank-fixing and rank-increasing strategies performs significantly better than the other compared methods.

2.3. **Synthetic data with power-law decaying singular values.** This section tests the recoverability of TMac, MatComp, FaLRTC, and SquareDeal on two more difficult synthetic datasets. In the first dataset, each tensor had the form $\mathcal{M} = \mathcal{C} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3$, where $\mathcal{C}$ was generated by MATLAB command `rand(r,r,r)` and $\mathbf{A}_n$ by `orth(randn(50,r))*diag([1:r].^(-0.5))`. Note that the core tensor $\mathcal{C}$ has nonzero-mean entries and each factor matrix has power-law decaying singular values. This kind of low-rank tensor appears more difficult to recover compared to the previous random low-rank tensors. Each tensor in the second dataset had the form $\mathcal{M} = \mathcal{C} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3 \times_4 \mathbf{A}_4$, where $\mathcal{C}$ was generated by MATLAB

command `rand(r,r,r,r)` and $\mathbf{A}_n$ by `orth(randn(20,r))*diag([1:r].^(-0.5))`. For these two datasets, TMac with rank-decreasing strategy can never decrease r_n to the true rank and thus performs badly. Figure 6 shows the recoverability of each method on the first dataset and Figure 7 on the second dataset. Again, we see that TMac with both rank-fixing and rank-increasing strategies performs significantly better than the other compared methods.

3. Algorithm.

We apply the alternating least squares method to (2). Since the model needs an estimate of rank($\boldsymbol{\mathcal{M}}$), we provide two strategies to dynamically adjust the rank estimates.

3.1. Alternating minimization.

The model (2) is convex with respect to each block of the variables $\mathbf{X}, \mathbf{Y}$ and $\boldsymbol{\mathcal{Z}}$ while the other two are fixed. Hence, we cyclically update $\mathbf{X}, \mathbf{Y}$ and $\boldsymbol{\mathcal{Z}}$ one at a time. Let

$$f(\mathbf{X}, \mathbf{Y}, \boldsymbol{\mathcal{Z}}) = \sum_{n=1}^{N} \frac{\alpha_n}{2} \|\mathbf{X}_n \mathbf{Y}_n - \mathbf{Z}_{(n)}\|_F^2 \tag{8}$$

be the objective of (2). We perform the updates as

$$\mathbf{X}^{k+1} = \operatorname*{argmin}_{\mathbf{X}} f(\mathbf{X}, \mathbf{Y}^k, \boldsymbol{\mathcal{Z}}^k), \tag{9a}$$

$$\mathbf{Y}^{k+1} = \operatorname*{argmin}_{\mathbf{Y}} f(\mathbf{X}^{k+1}, \mathbf{Y}, \boldsymbol{\mathcal{Z}}^k), \tag{9b}$$

$$\boldsymbol{\mathcal{Z}}^{k+1} = \operatorname*{argmin}_{\mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}}) = \boldsymbol{\mathcal{B}}} f(\mathbf{X}^{k+1}, \mathbf{Y}^{k+1}, \boldsymbol{\mathcal{Z}}). \tag{9c}$$

Note that both (9a) and (9b) can be decomposed into N independent least squares problems, which can be solved in parallel. The updates in (9) can be explicitly written as

$$\mathbf{X}_n^{k+1} = \mathbf{Z}_{(n)}^k (\mathbf{Y}_n^k)^\top \left(\mathbf{Y}_n^k (\mathbf{Y}_n^k)^\top\right)^\dagger, \ n = 1, \ldots, N, \tag{10a}$$

$$\mathbf{Y}_n^{k+1} = \left((\mathbf{X}_n^{k+1})^\top \mathbf{X}_n^{k+1}\right)^\dagger (\mathbf{X}_n^{k+1})^\top \mathbf{Z}_{(n)}^k, \ n = 1, \ldots, N, \tag{10b}$$

$$\boldsymbol{\mathcal{Z}}^{k+1} = \mathcal{P}_{\Omega^c}\left(\sum_{n=1}^{N} \mathbf{fold}_n(\mathbf{X}_n^{k+1} \mathbf{Y}_n^{k+1})\right) + \boldsymbol{\mathcal{B}}, \tag{10c}$$

where $\mathbf{A}^\dagger$ denotes the Moore-Penrose pseudo-inverse of $\mathbf{A}$, Ω^c is the complement of Ω, and we have used the fact that $\mathcal{P}_{\Omega^c}(\boldsymbol{\mathcal{B}}) = \mathbf{0}$ in (10c).

No matter how $\mathbf{X}_n$ is computed, only the products $\mathbf{X}_n \mathbf{Y}_n, n = 1, \ldots, N$, affect $\boldsymbol{\mathcal{Z}}$ and thus the recovery $\boldsymbol{\mathcal{M}}$. Hence, we shall update $\mathbf{X}$ in the following more efficient way

$$\mathbf{X}_n^{k+1} = \mathbf{Z}_{(n)}^k (\mathbf{Y}_n^k)^\top, \ n = 1, \ldots, N, \tag{11}$$

which together with (10b) gives the same products $\mathbf{X}_n^{k+1} \mathbf{Y}_n^{k+1}, \forall n$, as those by (10a) and (10b) according to the following lemma, which is similar to Lemma 2.1 in [27]. We give a proof here for completeness.

Lemma 3.1. *For any two matrices* $\mathbf{B}, \mathbf{C}$*, it holds that*

$$\begin{aligned} &\left(\mathbf{C}\mathbf{B}^\top (\mathbf{B}\mathbf{B}^\top)^\dagger\right) \left(\left(\mathbf{C}\mathbf{B}^\top (\mathbf{B}\mathbf{B}^\top)^\dagger\right)^\top \left(\mathbf{C}\mathbf{B}^\top (\mathbf{B}\mathbf{B}^\top)^\dagger\right)\right)^\dagger \left(\mathbf{C}\mathbf{B}^\top (\mathbf{B}\mathbf{B}^\top)^\dagger\right)^\top \mathbf{C} \\ = \ &\left(\mathbf{C}\mathbf{B}^\top\right) \left(\left(\mathbf{C}\mathbf{B}^\top\right)^\top \left(\mathbf{C}\mathbf{B}^\top\right)\right)^\dagger \left(\mathbf{C}\mathbf{B}^\top\right)^\top \mathbf{C}. \end{aligned} \tag{12}$$

Proof. Let $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the compact SVD of $\mathbf{B}$, i.e., $\mathbf{U}^\top\mathbf{U} = \mathbf{I}, \mathbf{V}^\top\mathbf{V} = \mathbf{I}$, and $\mathbf{\Sigma}$ is a diagonal matrix with all positive singular values on its diagonal. It is not difficult to verify that $\mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger = \mathbf{C}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top$. Then

$$\begin{aligned}
&\text{first line of (12)}\\
&= \mathbf{C}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top\big(\mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{V}^\top\mathbf{C}^\top\mathbf{C}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top\big)^\dagger\mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{V}^\top\mathbf{C}^\top\mathbf{C}\\
&= \mathbf{C}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\big(\mathbf{C}^\top\mathbf{C}\big)^\dagger\mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{V}^\top\mathbf{C}^\top\mathbf{C}\\
&= \mathbf{C}\mathbf{V}\mathbf{V}^\top\big(\mathbf{C}^\top\mathbf{C}\big)^\dagger\mathbf{V}\mathbf{V}^\top\mathbf{C}^\top\mathbf{C},
\end{aligned}$$

where we have used $(\mathbf{G}\mathbf{H})^\dagger = \mathbf{H}^\dagger\mathbf{G}^\dagger$ for any $\mathbf{G}, \mathbf{H}$ of appropriate sizes in the second equality. On the other hand,

$$\begin{aligned}
&\text{second line of (12)}\\
&= \mathbf{C}\mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top\big(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{C}^\top\mathbf{C}\mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top\big)^\dagger\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{C}^\top\mathbf{C}\\
&= \mathbf{C}\mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{V}^\top\big(\mathbf{C}^\top\mathbf{C}\big)^\dagger\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{C}^\top\mathbf{C}\\
&= \mathbf{C}\mathbf{V}\mathbf{V}^\top\big(\mathbf{C}^\top\mathbf{C}\big)^\dagger\mathbf{V}\mathbf{V}^\top\mathbf{C}^\top\mathbf{C}.
\end{aligned}$$

Hence, we have the desired result (12). □

3.2. **Rank-adjusting schemes.** The problem (2) requires one to specify the ranks $r_1, \ldots, r_N$. If they are fixed, then a good estimate is important for (2) to perform well. Too small r_n's can cause underfitting and a large recovery error whereas too large r_n's can cause overfitting and large deviation to the underlying tensor $\boldsymbol{\mathcal{M}}$. Since we do not assume the knowledge of rank($\boldsymbol{\mathcal{M}}$), we provide two schemes to dynamically adjust the rank estimates $r_1, \ldots, r_N$. In our algorithm, we use parameter ξ_n to determine which one of the two scheme to apply. If $\xi_n = -1$, the rank-decreasing scheme is applied to r_n; if $\xi_n = 1$, the rank-increasing scheme is applied to r_n; otherwise, r_n is fixed to its initial value.

3.2.1. *Rank-decreasing scheme.* This scheme starts from an input overestimated rank , i.e., $r_n > \text{rank}_n(\boldsymbol{\mathcal{M}})$. Following [27, 13], we calculate the eigenvalues of $\mathbf{X}_n^\top\mathbf{X}_n$ after each iteration, which are assumed to be ordered as $\lambda_1^n \geq \lambda_2^n \geq \ldots \geq \lambda_{r_n}^n$. Then we compute the quotients $\bar{\lambda}_i^n = \lambda_i^n/\lambda_{i+1}^n, i = 1, \ldots, r_n - 1$. Suppose

$$\hat{r}_n = \underset{1\leq i\leq r_n-1}{\text{argmax}}\ \bar{\lambda}_i.$$

If

$$\text{gap}_n = \frac{(r_n - 1)\bar{\lambda}_{\hat{r}_n}}{\sum_{i\neq\hat{r}_n}\bar{\lambda}_i} \geq 10, \tag{13}$$

which means a "big" gap between $\lambda_{\hat{r}_n}$ and $\lambda_{\hat{r}_n+1}$, then we reduce r_n to $\hat{r}_n$. Assume that the SVD of $\mathbf{X}_n\mathbf{Y}_n$ is $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. Then we update $\mathbf{X}_n$ to $\mathbf{U}_{\hat{r}_n}\mathbf{\Sigma}_{\hat{r}_n}$ and $\mathbf{Y}_n$ to $\mathbf{V}_{\hat{r}_n}^\top$, where $\mathbf{U}_{\hat{r}_n}$ is a submatrix of $\mathbf{U}$ containing $\hat{r}_n$ columns corresponding to the largest $\hat{r}_n$ singular values, and $\mathbf{\Sigma}_{\hat{r}_n}$ and $\mathbf{V}_{\hat{r}_n}$ are obtained accordingly.

We observe in our numerical experiments that this rank-adjusting scheme generally works well for exactly low-rank tensors. Because, for these tensors, gap_n can be very large and easy to identify, the true rank is typically obtained after just one rank adjustment. For approximately low-rank tensors, however, a large gap may or may not exist, and when it does not, the rank overestimates will not decrease. For these tensors, the rank-increasing scheme below works better.

3.2.2. *Rank-increasing scheme.* This scheme starts an underestimated rank, i.e., $r_n \leq \text{rank}_n(\boldsymbol{\mathcal{M}})$. Following [27], we increase r_n to $\min(r_n + \Delta r_n, r_n^{\max})$ at iteration $k+1$ if

$$\left| 1 - \frac{\left\| \boldsymbol{\mathcal{B}} - \mathcal{P}_\Omega\left(\mathbf{fold}_n(\mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1})\right)\right\|_F}{\left\| \boldsymbol{\mathcal{B}} - \mathcal{P}_\Omega\left(\mathbf{fold}_n(\mathbf{X}_n^{k}\mathbf{Y}_n^{k})\right)\right\|_F} \right| \leq 10^{-2}, \tag{14}$$

which means "slow" progress in the r_n dimensional space along the n-th mode. Here, Δr_n is a positive integer, and $r_n^{\max}$ is the maximal rank estimate. Let the economy QR factorization of $(\mathbf{Y}_n^{k+1})^\top$ be $\mathbf{QR}$. We augment $\mathbf{Q} \leftarrow [\mathbf{Q}, \hat{\mathbf{Q}}]$ where $\hat{\mathbf{Q}}$ has Δr_n randomly generated columns and then orthonormalize $\mathbf{Q}$. Next, we update $\mathbf{Y}_n^{k+1}$ to $\mathbf{Q}^\top$ and $\mathbf{X}_n^{k+1} \leftarrow [\mathbf{X}_n^{k+1}, \mathbf{0}]$, where $\mathbf{0}$ is an $I_n \times \Delta r_n$ *zero* matrix[4].

Numerically, this scheme works well not only for exactly low-rank tensors but also for approximately low-rank ones. However, for exactly low-rank tensors, this scheme causes the algorithm to run longer than the rank-decreasing scheme. Figure 8 shows the performance of our algorithm equipped with the two rank-adjusting schemes. As in section 2.1, we randomly generated $\boldsymbol{\mathcal{M}}$ of size $50 \times 50 \times 50$ with $\text{rank}_n(\boldsymbol{\mathcal{M}}) = r, \forall n$, and we started TMac with 25% overestimated ranks for rank-decreasing scheme and 25% underestimated ranks for rank-increasing scheme. From Figure 8, we see that TMac with the rank-decreasing scheme is better when r is small while TMac with the rank-increasing scheme becomes better when r is large. We can also see from the figure that after r_n is adjusted to match $\text{rank}_n(\boldsymbol{\mathcal{M}})$, our algorithm converges linearly.

In general, TMac with the rank-increasing scheme works no worse than it does with the rank-decreasing scheme in terms of solution quality no matter the underlying tensor is exactly low-rank or not. Numerically, we observed that if the rank is relatively low, TMac with the rank-decreasing scheme can adjust the estimated rank to the true one within just a few iterations and converges fast, while TMac with the rank-increasing scheme may need more time to get a comparable solution. However, if the underlying tensor is approximately low-rank, or it is exactly low-rank but the user concerns more on solution quality, TMac with the rank-increasing scheme is always preferred, and small Δr_n's usually give better solutions at the cost of more running time.

3.3. **Pseudocode.** The above discussions are distilled in Algorithm 1. After the algorithm terminates with output $(\mathbf{X}, \mathbf{Y}, \boldsymbol{\mathcal{Z}})$, we use $\sum_{n=1}^N \alpha_n \mathbf{fold}_n(\mathbf{X}_n\mathbf{Y}_n)$ to estimate the tensor $\boldsymbol{\mathcal{M}}$, which is usually better than $\boldsymbol{\mathcal{Z}}$ when the underlying $\boldsymbol{\mathcal{M}}$ is only approximately low-rank or the observations are contaminated by noise, or both.

Remark 1. In Algorithm 1, we can have different rank-adjusting schemes, i.e., different ξ_n, for different modes. For simplicity, we set $\xi_n = -1$ or $\xi_n = 1$ uniformly for all n in our experiments.

4. **Convergence analysis.** Introducing Lagrangian multiplier $\boldsymbol{\mathcal{W}}$ for the constraint $\mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}}) = \boldsymbol{\mathcal{B}}$, we write the Lagrangian function of (2)

$$L(\mathbf{X}, \mathbf{Y}, \boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{W}}) = f(\mathbf{X}, \mathbf{Y}, \boldsymbol{\mathcal{Z}}) - \langle \boldsymbol{\mathcal{W}}, \mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}}) - \boldsymbol{\mathcal{B}} \rangle. \tag{15}$$

[4]Since we update the variables in the order of $\mathbf{X}, \mathbf{Y}, \boldsymbol{\mathcal{Z}}$, appending any matrix of appropriate size after $\mathbf{X}_n$ does not make any difference.

Algorithm 1: Low-rank **T**ensor Completion by Parallel **M**atrix F**ac**torization (TMac)

Input: Ω, $\boldsymbol{\mathcal{B}} = \mathcal{P}_\Omega(\boldsymbol{\mathcal{M}})$, and $\alpha_n \geq 0, n = 1, \ldots, N$ with $\sum_{n=1}^N \alpha_n = 1$.
Parameters: $r_n, \Delta r_n, r_n^{\max}, \xi_n, n = 1, \ldots, N$.
Initialization: $(\mathbf{X}^0, \mathbf{Y}^0, \boldsymbol{\mathcal{Z}}^0)$ with $\mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}}^0) = \boldsymbol{\mathcal{B}}$.
for $k = 0, 1, \ldots$ **do**
  $\mathbf{X}^{k+1} \leftarrow$ (11), $\mathbf{Y}^{k+1} \leftarrow$ (10b), and $\boldsymbol{\mathcal{Z}}^{k+1} \leftarrow$ (10c).
  if *stopping criterion is satisfied* **then**
    Output $(\mathbf{X}^{k+1}, \mathbf{Y}^{k+1}, \boldsymbol{\mathcal{Z}}^{k+1})$.
  for $n = 1, \ldots, N$ **do**
    if $\xi_n = -1$ **then**
      Apply rank-decreasing scheme to $\mathbf{X}_n^{k+1}$ and $\mathbf{Y}_n^{k+1}$ in section 3.2.1.
    else if $\xi_n = 1$ **then**
      Apply rank-increasing scheme to $\mathbf{X}_n^{k+1}$ and $\mathbf{Y}_n^{k+1}$ in section 3.2.2.

Letting $\boldsymbol{\mathcal{T}} = (\mathbf{X}, \mathbf{Y}, \boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{W}})$ and $\nabla_{\boldsymbol{\mathcal{T}}} L = 0$, we have the KKT conditions

$$(\mathbf{X}_n \mathbf{Y}_n - \mathbf{Z}_{(n)})\mathbf{Y}_n^\top = 0, \ n = 1, \ldots, N, \tag{16a}$$

$$\mathbf{X}_n^\top (\mathbf{X}_n \mathbf{Y}_n - \mathbf{Z}_{(n)}) = 0, \ n = 1, \ldots, N, \tag{16b}$$

$$\boldsymbol{\mathcal{Z}} - \sum_{n=1}^N \alpha_n \cdot \mathbf{fold}_n(\mathbf{X}_n \mathbf{Y}_n) - \mathcal{P}_\Omega(\boldsymbol{\mathcal{W}}) = 0, \tag{16c}$$

$$\mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}}) - \boldsymbol{\mathcal{B}} = 0. \tag{16d}$$

Our main result is summarized in the following theorem.

Theorem 4.1. *Suppose* $\{(\mathbf{X}^k, \mathbf{Y}^k, \boldsymbol{\mathcal{Z}}^k)\}$ *is a sequence generated by Algorithm 1 with fixed* r_n*'s and fixed positive* α_n*'s. Let* $\boldsymbol{\mathcal{W}}^k = \boldsymbol{\mathcal{B}} - \mathcal{P}_\Omega\big(\sum_n \alpha_n \cdot \mathbf{fold}_n(\mathbf{X}_n^k \mathbf{Y}_n^k)\big)$. *Then any limit point of* $\{\boldsymbol{\mathcal{T}}^k\}$ *satisfies the KKT conditions in* (16).

Our proof of Theorem 4.1 mainly follows [27]. The literature has work that analyzes the convergence of alternating minimization method for non-convex problems such as [5, 25, 29, 30]. However, to the best of our knowledge, none of them implies our convergence result.

Before giving the proof, we establish some important lemmas that are used to establish our main result. We begin with the following lemma, which is similar to Lemma 3.1 of [27].

Lemma 4.2. *For any matrices* $\mathbf{A}, \mathbf{B}$ *and* $\mathbf{C}$ *of appropriate sizes, letting* $\tilde{\mathbf{A}} = \mathbf{C}\mathbf{B}^\top, \tilde{\mathbf{B}} = (\tilde{\mathbf{A}}^\top \tilde{\mathbf{A}})^\dagger \tilde{\mathbf{A}}^\top \mathbf{C}$, *then we have*

$$\|\mathbf{A}\mathbf{B} - \mathbf{C}\|_F^2 - \|\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{C}\|_F^2 = \|\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{A}\mathbf{B}\|_F^2. \tag{17}$$

Proof. From Lemma 3.1, it suffices to prove (17) by letting $\tilde{\mathbf{A}} = \mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger, \tilde{\mathbf{B}} = (\tilde{\mathbf{A}}^\top \tilde{\mathbf{A}})^\dagger \tilde{\mathbf{A}}^\top \mathbf{C}$. Assume the compact SVDs of $\tilde{\mathbf{A}}$ and $\mathbf{B}$ are $\tilde{\mathbf{A}} = \mathbf{U}_{\tilde{a}} \boldsymbol{\Sigma}_{\tilde{a}} \mathbf{V}_{\tilde{a}}^\top$ and $\mathbf{B} = \mathbf{U}_b \boldsymbol{\Sigma}_b \mathbf{V}_b^\top$. Noting $\mathbf{B} = \mathbf{B}\mathbf{V}_b\mathbf{V}_b^\top$ and $\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger \mathbf{B} = \mathbf{V}_b\mathbf{V}_b^\top$, we have

$$\tilde{\mathbf{A}}\mathbf{B} - \mathbf{A}\mathbf{B} = (\mathbf{C} - \mathbf{A}\mathbf{B})\mathbf{V}_b\mathbf{V}_b^\top. \tag{18}$$

Similarly, noting $\tilde{\mathbf{A}} = \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top\tilde{\mathbf{A}}$ and $\tilde{\mathbf{A}}(\tilde{\mathbf{A}}^\top\tilde{\mathbf{A}})^\dagger\tilde{\mathbf{A}}^\top = \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top$, we have

$$\begin{aligned}\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \tilde{\mathbf{A}}\mathbf{B} =& \tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top\tilde{\mathbf{A}}\mathbf{B}\\ =& \tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top(\tilde{\mathbf{A}}\mathbf{B} - \mathbf{A}\mathbf{B} + \mathbf{A}\mathbf{B})\\ =& \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top\mathbf{C} - \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top\left[(\mathbf{C} - \mathbf{A}\mathbf{B})\mathbf{V}_b\mathbf{V}_b^\top - \mathbf{A}\mathbf{B}\right]\\ =& \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top(\mathbf{C} - \mathbf{A}\mathbf{B})(\mathbf{I} - \mathbf{V}_b\mathbf{V}_b^\top)\end{aligned} \tag{19}$$

where the third equality is from (18). Summing (18) and (19) gives

$$\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{A}\mathbf{B} = (\mathbf{I} - \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top)(\mathbf{C} - \mathbf{A}\mathbf{B})\mathbf{V}_b\mathbf{V}_b^\top + \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top(\mathbf{C} - \mathbf{A}\mathbf{B}). \tag{20}$$

Since $(\mathbf{I} - \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top)(\mathbf{C} - \mathbf{A}\mathbf{B})$ is orthogonal to $\mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top(\mathbf{C} - \mathbf{A}\mathbf{B})$, we have

$$\left\|\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{A}\mathbf{B}\right\|_F^2 = \left\|(\mathbf{I} - \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top)(\mathbf{C} - \mathbf{A}\mathbf{B})\mathbf{V}_b\mathbf{V}_b^\top\right\|_F^2 + \left\|\mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top(\mathbf{C} - \mathbf{A}\mathbf{B})\right\|_F^2. \tag{21}$$

In addition, note

$$\langle(\mathbf{I} - \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top)(\mathbf{C} - \mathbf{A}\mathbf{B})\mathbf{V}_b\mathbf{V}_b^\top, \mathbf{C} - \mathbf{A}\mathbf{B}\rangle = \|(\mathbf{I} - \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top)(\mathbf{C} - \mathbf{A}\mathbf{B})\mathbf{V}_b\mathbf{V}_b^\top\|_F^2,$$

and

$$\langle\mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top(\mathbf{C} - \mathbf{A}\mathbf{B}), \mathbf{C} - \mathbf{A}\mathbf{B}\rangle = \|\mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top(\mathbf{C} - \mathbf{A}\mathbf{B})\|_F^2.$$

Hence,

$$\langle\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{A}\mathbf{B}, \mathbf{C} - \mathbf{A}\mathbf{B}\rangle = \left\|(\mathbf{I} - \mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top)(\mathbf{C} - \mathbf{A}\mathbf{B})\mathbf{V}_b\mathbf{V}_b^\top\right\|_F^2 + \left\|\mathbf{U}_{\tilde{a}}\mathbf{U}_{\tilde{a}}^\top(\mathbf{C} - \mathbf{A}\mathbf{B})\right\|_F^2,$$

and thus $\|\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{A}\mathbf{B}\|_F^2 = \langle\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{A}\mathbf{B}, \mathbf{C} - \mathbf{A}\mathbf{B}\rangle$. Then (17) can be shown by noting

$$\|\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{C}\|_F^2 = \|\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{A}\mathbf{B}\|_F^2 + 2\langle\tilde{\mathbf{A}}\tilde{\mathbf{B}} - \mathbf{A}\mathbf{B}, \mathbf{A}\mathbf{B} - \mathbf{C}\rangle + \|\mathbf{A}\mathbf{B} - \mathbf{C}\|_F^2.$$

This completes the proof. □

We also need the following lemma.

Lemma 4.3. *For any two matrices* $\mathbf{B}$ *and* $\mathbf{C}$ *of appropriate sizes, it holds that*

$$\mathcal{R}_c\left(\mathbf{C}\mathbf{B}^\top\right) = \mathcal{R}_c\left(\mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger\right), \tag{22a}$$

$$\begin{aligned}&\mathcal{R}_r\left(\left[(\mathbf{C}\mathbf{B}^\top)^\top(\mathbf{C}\mathbf{B}^\top)\right]^\dagger(\mathbf{C}\mathbf{B}^\top)^\top\mathbf{C}\right)\\ &= \mathcal{R}_r\left(\left[(\mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger)^\top(\mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger)\right]^\dagger\left[\mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger\right]^\top\mathbf{C}\right),\end{aligned} \tag{22b}$$

where $\mathcal{R}_c(\mathbf{A})$ *and* $\mathcal{R}_r(\mathbf{A})$ *denote the column and row space of* $\mathbf{A}$*, respectively.*

Proof. Following the proof of Lemma 3.1, we assume the compact SVD of $\mathbf{B}$ to be $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. Then

$$\mathbf{C}\mathbf{B}^\top = \mathbf{C}\mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top \text{ and } \mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger = \mathbf{C}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top.$$

For any vector $\mathbf{x}$ of appropriate size, there must be another vector $\mathbf{y}$ such that $\mathbf{U}^\top\mathbf{y} = \mathbf{\Sigma}^2\mathbf{U}^\top\mathbf{x}$ or equivalently $\mathbf{\Sigma}^{-1}\mathbf{U}^\top\mathbf{y} = \mathbf{\Sigma}\mathbf{U}^\top\mathbf{x}$. Hence $\mathbf{C}\mathbf{B}^\top\mathbf{x} = \mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger\mathbf{y}$, which indicates $\mathcal{R}_c\left(\mathbf{C}\mathbf{B}^\top\right) \subset \mathcal{R}_c\left(\mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger\right)$. In the same way, one can show the reverse inclusion, and hence $\mathcal{R}_c\left(\mathbf{C}\mathbf{B}^\top\right) = \mathcal{R}_c\left(\mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger\right)$. The result (22b) can be shown similarly by noting that

$$\left[(\mathbf{C}\mathbf{B}^\top)^\top(\mathbf{C}\mathbf{B}^\top)\right]^\dagger(\mathbf{C}\mathbf{B}^\top)^\top\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top(\mathbf{C}^\top\mathbf{C})^\dagger\mathbf{V}\mathbf{V}^\top\mathbf{C}^\top\mathbf{C},$$

$$\left[(\mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger)^\top(\mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger)\right]^\dagger\left[\mathbf{C}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^\dagger\right]^\top\mathbf{C} = \mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{V}^\top(\mathbf{C}^\top\mathbf{C})^\dagger\mathbf{V}\mathbf{V}^\top\mathbf{C}^\top\mathbf{C}.$$

This completes the proof. □

According to Lemma 4.3, it is not difficult to get the following corollary.

Corollary 1. *For any matrices* $\mathbf{A}, \mathbf{B}$ *and* $\mathbf{C}$ *of appropriate sizes, let* $\tilde{\mathbf{A}} = \mathbf{C}\mathbf{B}^\top, \tilde{\mathbf{B}} = (\tilde{\mathbf{A}}^\top\tilde{\mathbf{A}})^\dagger\tilde{\mathbf{A}}^\top\mathbf{C}$. *If the compact SVDs of* $\tilde{\mathbf{A}}$ *and* $\mathbf{B}$ *are* $\tilde{\mathbf{A}} = \mathbf{U}_{\tilde{a}}\mathbf{\Sigma}_{\tilde{a}}\mathbf{V}_{\tilde{a}}^\top$ *and* $\mathbf{B} = \mathbf{U}_b\mathbf{\Sigma}_b\mathbf{V}_b^\top$, *then we have* (21).

We are now ready to prove Theorem 4.1.

Proof of Theorem 4.1. Let $\bar{\boldsymbol{\mathcal{T}}} = (\bar{\mathbf{X}}, \bar{\mathbf{Y}}, \bar{\boldsymbol{\mathcal{Z}}}, \bar{\boldsymbol{\mathcal{W}}})$ be a limit point of $\{\boldsymbol{\mathcal{T}}^k\}$, and thus there is a subsequence $\{\boldsymbol{\mathcal{T}}^k\}_{k\in\mathcal{K}}$ converging to $\bar{\boldsymbol{\mathcal{T}}}$. According to (9c), we have $\mathcal{P}_\Omega(\boldsymbol{\mathcal{Z}}^k) = \boldsymbol{\mathcal{B}}$, and $\mathcal{P}_{\Omega^c}(\boldsymbol{\mathcal{Z}}^k) = \mathcal{P}_{\Omega^c}(\sum_n \alpha_n \cdot \mathbf{fold}_n(\mathbf{X}_n^k\mathbf{Y}_n^k))$. Hence, (16c) and (16d) hold at $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{T}}^k$ for all k and thus at $\bar{\boldsymbol{\mathcal{T}}}$.

From Lemma 4.2, it follows that

$$f(\mathbf{X}^k, \mathbf{Y}^k, \boldsymbol{\mathcal{Z}}^k) - f(\mathbf{X}^{k+1}, \mathbf{Y}^{k+1}, \boldsymbol{\mathcal{Z}}^k) = \sum_{n=1}^N \frac{\alpha_n}{2}\|\mathbf{X}^k\mathbf{Y}^k - \mathbf{X}^{k+1}\mathbf{Y}^{k+1}\|_F^2. \tag{23}$$

In addition, it is not difficult to verify

$$f(\mathbf{X}^{k+1}, \mathbf{Y}^{k+1}, \boldsymbol{\mathcal{Z}}^k) - f(\mathbf{X}^{k+1}, \mathbf{Y}^{k+1}, \boldsymbol{\mathcal{Z}}^{k+1}) = \frac{1}{2}\|\boldsymbol{\mathcal{Z}}^k - \boldsymbol{\mathcal{Z}}^{k+1}\|_F^2. \tag{24}$$

Summing up (23) and (24) and observing that f is lower bounded by *zero*, we have

$$\sum_{k=0}^\infty \left(\sum_{n=1}^N \frac{\alpha_n}{2}\|\mathbf{X}^k\mathbf{Y}^k - \mathbf{X}^{k+1}\mathbf{Y}^{k+1}\|_F^2 + \frac{1}{2}\|\boldsymbol{\mathcal{Z}}^k - \boldsymbol{\mathcal{Z}}^{k+1}\|_F^2\right) < \infty,$$

and thus

$$\lim_{k\to\infty} \|\mathbf{X}_n^k\mathbf{Y}_n^k - \mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1}\|_F^2 = 0, \text{ and } \lim_{k\to\infty} \|\boldsymbol{\mathcal{Z}}^k - \boldsymbol{\mathcal{Z}}^{k+1}\|_F^2 = 0. \tag{25}$$

For each n and k, let the compact SVDs of $\mathbf{X}_n^k$ and $\mathbf{Y}_n^k$ be $\mathbf{X}_n^k = \mathbf{U}_{x_n^k}\mathbf{\Sigma}_{x_n^k}\mathbf{V}_{x_n^k}^\top$ and $\mathbf{Y}_n^k = \mathbf{U}_{y_n^k}\mathbf{\Sigma}_{y_n^k}\mathbf{V}_{y_n^k}^\top$. Letting $\mathbf{A} = \mathbf{X}_n^k, \mathbf{B} = \mathbf{Y}_n^k, \tilde{\mathbf{A}} = \mathbf{X}_n^{k+1}, \tilde{\mathbf{B}} = \mathbf{Y}_n^{k+1}$, and $\mathbf{C} = \mathbf{Z}_{(n)}^k$ in (21), we have

$$\begin{aligned}&\left\|\mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1} - \mathbf{X}_n^k\mathbf{Y}_n^k\right\|_F^2\\ &= \left\|(\mathbf{I} - \mathbf{U}_{x_n^{k+1}}\mathbf{U}_{x_n^{k+1}}^\top)(\mathbf{Z}_{(n)}^k - \mathbf{X}_n^k\mathbf{Y}_n^k)\mathbf{V}_{y_n^k}\mathbf{V}_{y_n^k}^\top\right\|_F^2 + \left\|\mathbf{U}_{x_n^{k+1}}\mathbf{U}_{x_n^{k+1}}^\top(\mathbf{Z}_{(n)}^k - \mathbf{X}_n^k\mathbf{Y}_n^k)\right\|_F^2,\end{aligned}$$

which together with (25) gives

$$\lim_{k\to\infty} \left(\mathbf{Z}_{(n)}^k - \mathbf{X}_n^k\mathbf{Y}_n^k\right)\mathbf{V}_{y_n^k}\mathbf{V}_{y_n^k}^\top = 0, \tag{26a}$$

$$\lim_{k\to\infty} \mathbf{U}_{x_n^{k+1}}\mathbf{U}_{x_n^{k+1}}^\top\left(\mathbf{Z}_{(n)}^k - \mathbf{X}_n^k\mathbf{Y}_n^k\right) = 0. \tag{26b}$$

Since $(\mathbf{Y}_n^k)^\top = \mathbf{V}_{y_n^k}\mathbf{V}_{y_n^k}^\top(\mathbf{Y}_n^k)^\top$ and $\{\mathbf{Y}_n^k\}_{k\in\mathcal{K}}$ is bounded, then right multiplying $(\mathbf{Y}_n^k)^\top$ for $k \in \mathcal{K}$ on both sides of (26a) yields

$$\left(\bar{\mathbf{Z}}_{(n)} - \bar{\mathbf{X}}_n\bar{\mathbf{Y}}_n\right)\bar{\mathbf{Y}}_n^\top = \lim_{\substack{k\to\infty\\ k\in\mathcal{K}}} \left(\mathbf{Z}_{(n)}^k - \mathbf{X}_n^k\mathbf{Y}_n^k\right)(\mathbf{Y}_n^k)^\top = 0,$$

which indicates that (16a) is satisfied at $\bar{\boldsymbol{\mathcal{T}}}$. From (25) and (26b), we have

$$\lim_{k\to\infty} \mathbf{U}_{x_n^k}\mathbf{U}_{x_n^k}^\top\left(\mathbf{Z}_{(n)}^k - \mathbf{X}_n^k\mathbf{Y}_n^k\right) = 0,$$

which together with the boundedness of $\{\mathbf{X}_n^k\}_{k\in\mathcal{K}}$ and $\mathbf{X}_n^k = \mathbf{U}_{x_n^k}\mathbf{U}_{x_n^k}^\top\mathbf{X}_n^k$ gives

$$\bar{\mathbf{X}}_n^\top\left(\bar{\mathbf{Z}}_{(n)} - \bar{\mathbf{X}}_n\bar{\mathbf{Y}}_n\right) = \lim_{\substack{k\to\infty\\ k\in\mathcal{K}}} (\mathbf{X}_n^k)^\top\left(\mathbf{Z}_{(n)}^k - \mathbf{X}_n^k\mathbf{Y}_n^k\right) = 0,$$

and thus (16b) is satisfied at $\bar{\boldsymbol{\mathcal{T}}}$. This completes the proof.

5. **Numerical experiments.** This section tests Algorithm 1, TMac, for solving (2). To demonstrate its effectiveness, we compared it with MatComp and FaLRTC (see section 2) on real-world data.

5.1. **Dynamic weights and stopping rules.** The parameters $\alpha_1, \ldots, \alpha_N$ in (2) were uniformly set to $\frac{1}{N}$ at the beginning of TMac. During the iterations, we either fixed them or dynamically updated them according to the fitting error

$$\mathbf{fit}_n(\mathbf{X}_n\mathbf{Y}_n) = \|\mathcal{P}_\Omega\big(\mathbf{fold}_n(\mathbf{X}_n\mathbf{Y}_n) - \boldsymbol{\mathcal{B}}\big)\|_F.$$

The smaller $\mathbf{fit}_n(\mathbf{X}_n\mathbf{Y}_n)$ is, the larger α_n should be. Specifically, if the current iterate is $(\mathbf{X}^k, \mathbf{Y}^k, \boldsymbol{\mathcal{Z}}^k)$, we set

$$\alpha_n^k = \frac{\big[\mathbf{fit}_n(\mathbf{X}_n^k\mathbf{Y}_n^k)\big]^{-1}}{\sum_{i=1}^N \big[\mathbf{fit}_i(\mathbf{X}_i^k\mathbf{Y}_i^k)\big]^{-1}},\ n = 1, \ldots, N. \tag{27}$$

As demonstrated below, dynamic updating α_n's can improve the recovery quality for tensors that have better low-rankness in one mode than others. TMac was terminated if one of the following conditions was satisfied for some k

$$\frac{\Big|\sum_{n=1}^N \mathbf{fit}_n(\mathbf{X}_n^k\mathbf{Y}_n^k) - \sum_{n=1}^N \mathbf{fit}_n(\mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1})\Big|}{1 + \sum_{n=1}^N \mathbf{fit}_n(\mathbf{X}_n^k\mathbf{Y}_n^k)} \le tol, \tag{28}$$

$$\frac{\sum_{n=1}^N \alpha_n^k \cdot \mathbf{fit}_n(\mathbf{X}_n^{k+1}\mathbf{Y}_n^{k+1})}{\|\boldsymbol{\mathcal{B}}\|_F} \le tol, \tag{29}$$

where tol is a small positive value specified below. The condition (28) checks the relative change of the overall fitting, and (29) is satisfied if the weighted fitting is good enough.

5.2. **MRI data.** This section compares TMac, MatComp, and FaLRTC on a $181 \times 217 \times 181$ brain MRI data, which has been used in [16]. The data is approximately low-rank: for its three mode unfodings, the numbers of singular values larger than 0.1% of the largest one are 28, 33, and 29, respectively. One slice of the data is shown in Figure 9. We tested all three methods on both noiseless and noisy data[5]. Specifically, we added scaled Gaussian noise to the original data to have

$$\boldsymbol{\mathcal{M}}^{nois} = \boldsymbol{\mathcal{M}} + \sigma\frac{\|\boldsymbol{\mathcal{M}}\|_\infty}{\|\boldsymbol{\mathcal{N}}\|_\infty}\boldsymbol{\mathcal{N}},$$

and made noisy observations $\boldsymbol{\mathcal{B}} = \mathcal{P}_\Omega(\boldsymbol{\mathcal{M}}^{nois})$, where $\|\boldsymbol{\mathcal{M}}\|_\infty$ denotes the maximum absolute value of $\boldsymbol{\mathcal{M}}$, and the entries of $\boldsymbol{\mathcal{N}}$ follow idendically independent standard Gaussian distribution.

We ran all the algorithms to maximum 1000 iterations. The stopping tolerance was set to $tol = 10^{-3}$ for TMac and LMaFit. For FaLRTC, $tol = 10^{-4}$ was set since we found 10^{-3} was too loose. Both TMac and LMaFit used the rank-increasing strategy. For TMac, we initialized $r_n = 5$ and set $\Delta r_n = 3, r_n^{\max} = 50, \forall n$, and for LMaFit, we set initial rank $K = 5$, increment $\kappa = 3$, and maximal rank $K^{\max} = 50$. We tested TMac with fixed parameters $\alpha_n = \frac{1}{3}, n = 1, 2, 3$, and also dynamically

[5]For noisy case, it could be better to relax the equality constraints in (2), (3), and (5) to include some information on the noise level. However, we did not assume such information, and these methods could still work reasonably.

updated ones by (27) starting from $\alpha_n^0 = \frac{1}{3}, n = 1, 2, 3$. The smoothing parameter for FaLRTC was set to its default value $\mu = 0.5$ and weight parameters set to $\alpha_n = \frac{1}{3}, n = 1, 2, 3$. Table 1 shows the average relative errors and running times of five independent trials for each setting of σ and SR. Figure 9 shows one noisy masked slice and the corresponding recovered slices by different methods with the setting of $\sigma = 0.05$ and SR = 10%. From the results, we see that TMac consistently reached lower relative errors than those by FaLRTC and cost less time. MatComp used the least time and could achieve low relative error as SR is large. However, for low SR's (e.g., SR=10%), it performed extremely bad, and even we ran it to more iterations, say 5000, it still performed much worse than TMac and FaLRTC. In addition, TMac using fixed α_n's worked similarly well as that using dynamically updated ones, which should be because the data has similar low-rankness along each mode.

5.3. **Hyperspectral data.** This section compares TMac, MatComp, and FaLRTC on a $205 \times 246 \times 96$ hyperspectral data, one slice of which is shown in Figure 10. This data is also approximately low-rank: for its three mode unfoldings, the numbers of singular values larger than 1% of the largest one are 19,19, and 4, respectively. However, its low-rank property is not as good as that of the above MRI data. Its numbers of singular values larger than 0.1% of the largest one are 198, 210, and 18. Hence, its mode-3 unfolding has better low-rankness, and we assigned larger weight to the third mode. For FaLRTC, we set $\alpha_1 = \alpha_2 = 0.25, \alpha_3 = 0.5$. For TMac, we tested it with fixed weights $\alpha_1 = \alpha_2 = 0.25, \alpha_3 = 0.5$ and also with dynamically updated ones by (27) starting from $\alpha_1^0 = \alpha_2^0 = 0.25, \alpha_3^0 = 0.5$. All other parameters of the three methods were set as the same as those used in the previous test.

For each setting of σ and SR, we made 5 independent runs. Table 2 reports the average results of each tested method, and Figure 10 shows one noisy slice with 90% missing values and 5% Gaussian noise, and the corresponding recovered slices by the compared methods. From the results, we see again that TMac outperformed FaLRTC in both solution quality and running time, and MatComp gave the largest relative errors at the cost of least time. In addition, TMac with dynamically updated α_n's worked better than that with fixed α_n's as SR was relatively large (e.g., SR=30%, 50%), and this should be because the data has much better low-rankness along the third mode than the other two. As SR was low (e.g., SR=10%), TMac with dynamically updated α_n's performed even worse. This could be explained by the case that all slices might have missing values at common locations (i.e., some mode-3 fibers were entirely missing) as SR was low, and in this case, the third mode unfolding had some entire columns missing. In general, it is impossible for any matrix completion solver to recover an entire missing column or row of a matrix, and thus putting more weight on the third mode could worsen the recovery. That also explains why MatComp gave much larger relative errors than those by FaLRTC but the slice recovered by MatComp looks better than that by FaLRTC in Figure 10. Note that there are lots of black points on the slice given by MatComp, and these black points correspond to missing columns of the third mode unfolding. Therefore, we do not recommend to dynamically update α_n's in TMac when SR is low or some fibers are entirely missing.

5.4. **Video inpainting.** In this section, we compared TMac, MatComp, and FaLRTC on both grayscale and color videos. The grayscale video[6] has 200 frames with each one of size 130×160, and the color video[7] has 150 frames with each one of size 144×176. We treated the grayscale video as a $130 \times 160 \times 200$ tensor and the color video as three $144 \times 176 \times 150$ tensors[8], one for each channel. For the grayscale video, the numbers of singular values larger than 1% of the largest one for each mode unfolding are 79, 84, and 35. Hence, its rank is not low, and it is relatively difficult to recover this video. The color video has lower rank. For each of its three channel tensors, the numbers of singular values larger than 1% of the largest one are about[9] 50, 50, and 24. During each run, the three channel tensors had the same index set of observed entries, which is the case in practice, and we recovered each channel independently. We set $\alpha_n = \frac{1}{3}, n = 1, 2, 3$ for FaLRTC and TMac, while the latter was tested with both fixed α_n's and dynamically updated one by (27). All the other parameters of the test methods were set as the same as those in the previous test. The average results of 5 independent runs were reported in Table 3 for the grayscale video and Table 4 for the color video. Figure 11 shows one frame of recovered grayscale video by each method and Figure 12 one frame of recovered color video. From the tables, we see that the comparisons are similar to those for the previous hyperspectral data recovery.

6. **Discussions.** We have proposed a new method for low-rank tensor completion. Our model utilizes low-rank matrix factorizations to all-mode unfoldings of the tensor. Synthetic data tests demonstrate that our model can recover significantly more low-rank tensors than two nuclear norm based models and one model that performs low-rank matrix factorization to only one mode unfolding. In addition, numerical results on 3D images and videos show that our method consistently produces the best solutions among all compared methods and outperforms the nuclear norm minimization method in both solution quality and running time.

Numerically, we have observed that our algorithm converges fast (e.g., linear convergence in Figure 8). Papers [15, 27] demonstrate that the SOR technique can significantly accelerate the progress of alternating least squares. However, we did not observe any acceleration applying the same technique to our algorithm. In the future, we will explore the reason and try to develop other techniques to accelerate our algorithm. We also plan to incorporate the objective term in (7) to enrich (2), if the underlying low-rank tensor has more than three orders.

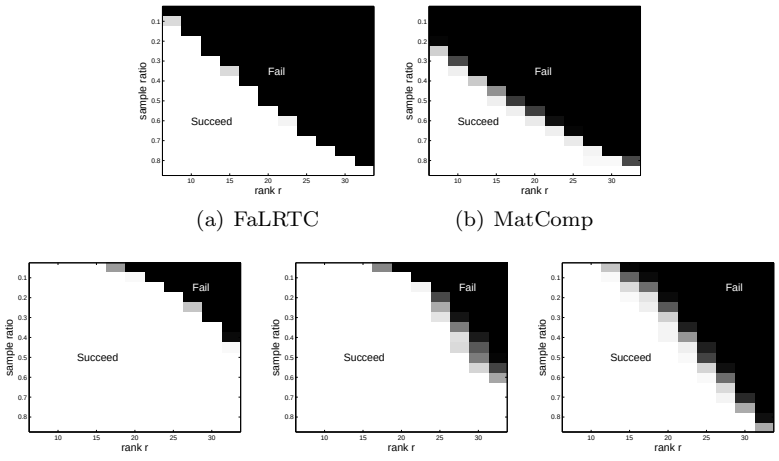
7. **Figures and tables.** We give all figures and tables of our numerical results in this section.

[6] http://cvxr.com/tfocs/demos/rpca/

[7] http://media.xiph.org/video/derf/ The original video has 500 frames, and we used its first 150 frames in our test.

[8] One can also treat the color video as a 4th-order tensor. However, we found that recovering a 4th-order tensor cost much more time than recovering three 3rd-order tensors and made no quality improvement.

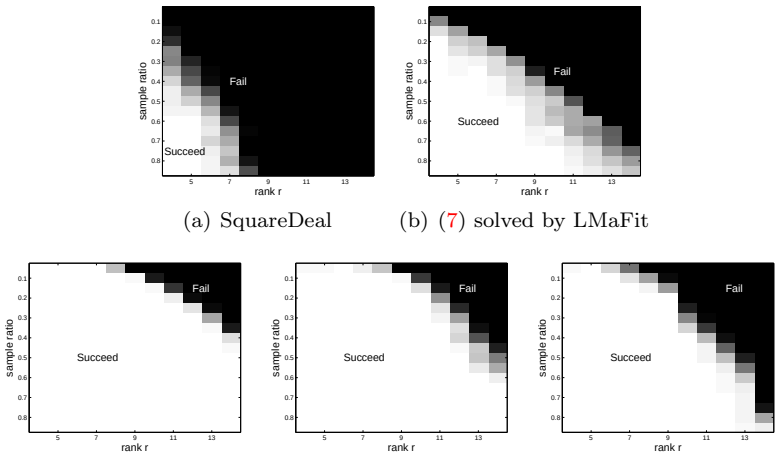
[9] More precisely, the numbers are $(51, 53, 24)$, $(48, 51, 24)$, and $(49, 52, 24)$ respectively for three channel tensors.



(a) FaLRTC (b) MatComp

(c) TMac-fix (d) TMac-inc (e) TMac-dec

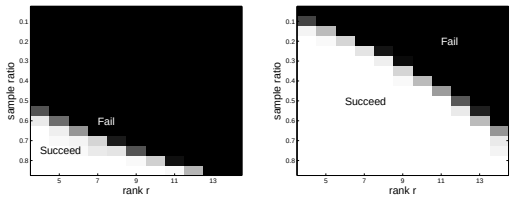
FIGURE 1. Phase transition plots for different methods on **3-way low-rank tensors whose factors have Gaussian random entries**. (a) FaLRTC: the tensor completion method in [16] that solves (5). (b) MatComp: the matrix completion solver LMaFit in [27] that solves (3) with $\mathbf{Z}$ corresponding to $\mathbf{M}_{(N)}$. (c) TMac-fix: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{3}$ and r_n fixed to r, $\forall n$. (d) TMac-inc: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{3}, \forall n$ and using rank-increasing strategy starting from $r_n = \text{round}(0.75r), \forall n$. (e) TMac-dec: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{3}, \forall n$ and using rank-decreasing strategy starting from $r_n = \text{round}(1.25r), \forall n$.



(a) SquareDeal (b) (7) solved by LMaFit

(c) TMac-fix (d) TMac-inc (e) TMac-dec

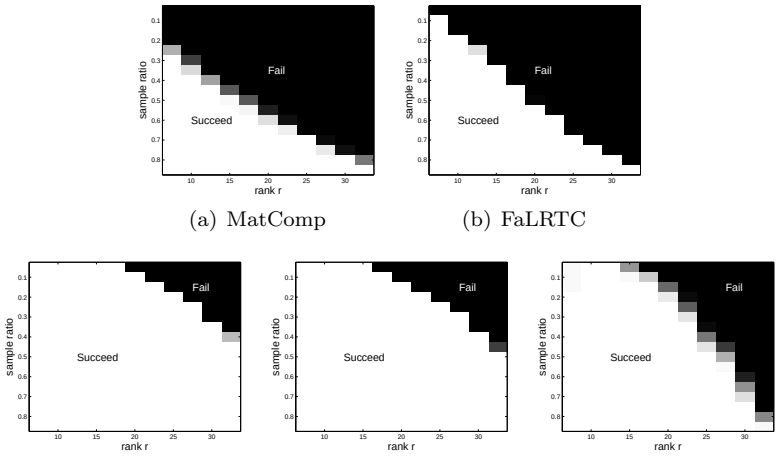
FIGURE 2. Phase transition plots for different methods on **4-way low-rank tensors whose factors have Gaussian random entries**. (a) SquareDeal: the matrix completion solver FPCA [17] solves the model (6) proposed in [18]. (b) the matrix completion solver LMaFit [27] solves (7), which is a non-convex variant of (6). (c) TMac-fix: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{4}$ and r_n fixed to r, $\forall n$. (d) TMac-inc: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{4}, \forall n$ and using rank-increasing strategy starting from $r_n = \text{round}(0.75r), \forall n$. (e) TMac-dec: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{4}, \forall n$ and using rank-decreasing strategy starting from $r_n = \text{round}(1.25r), \forall n$.



(a) 1 mode (b) 2 modes

(c) 3 modes (d) 4 modes

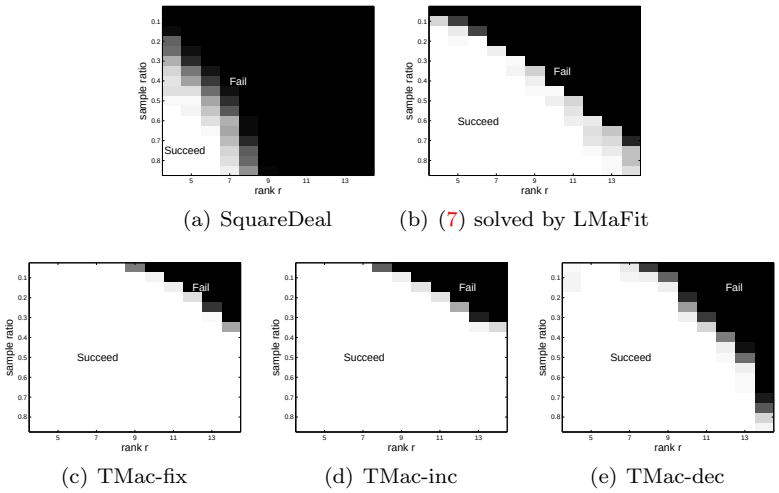
FIGURE 3. Phase transition plots for model (2) utilizing different numbers of mode matricization on **4-way low-rank tensors whose factors have Gaussian random entries**. (a) 1 mode: Algorithm 1 solves (2) with $\alpha_1 = 1, \alpha_n = 0, n \geq 2$ and each r_n fixed to r; (b) 2 modes: Algorithm 1 solves (2) with $\alpha_1 = \alpha_2 = 0.5, \alpha_3 = \alpha_4 = 0$, and each r_n fixed to r; (c) 3 modes: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{3}, n \leq 3, \alpha_4 = 0$, and each r_n fixed to r; (d) 4 modes: Algorithm 1 solves (2) with $\alpha_n = 0.25, \forall n$ and each r_n fixed to r.



(a) MatComp (b) FaLRTC

(c) TMac-fix (d) TMac-inc (e) TMac-dec

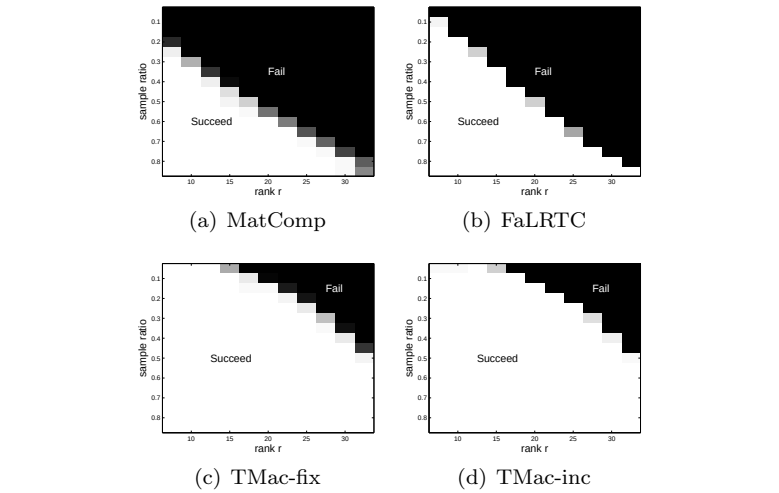
FIGURE 4. Phase transition plots for different methods on **3-way low-rank tensors whose factors have uniformly random entries**. (a) FaLRTC: the tensor completion method in [16] that solves (5). (b) MatComp: the matrix completion solver LMaFit in [27] that solves (3) with $\mathbf{Z}$ corresponding to $\mathbf{M}_{(N)}$. (c) TMac-fix: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{3}$ and r_n fixed to r, $\forall n$. (d) TMac-inc: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{3}, \forall n$ and using rank-increasing strategy starting from $r_n = \text{round}(0.75r), \forall n$. (e) TMac-dec: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{3}, \forall n$ and using rank-decreasing strategy starting from $r_n = \text{round}(1.25r), \forall n$.



(a) SquareDeal (b) (7) solved by LMaFit

(c) TMac-fix (d) TMac-inc (e) TMac-dec

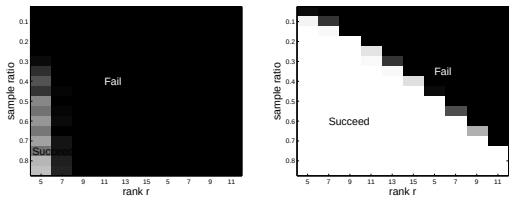
FIGURE 5. Phase transition plots for different methods on **4-way low-rank tensors whose factors have uniformly random entries**. (a) SquareDeal: the matrix completion solver FPCA [17] solves the model (6) proposed in [18]. (b) the matrix completion solver LMaFit [27] solves (7), which is a non-convex variant of (6). (c) TMac-fix: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{4}$ and r_n fixed to r, $\forall n$. (d) TMac-inc: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{4}, \forall n$ and using rank-increasing strategy starting from $r_n = \text{round}(0.75r), \forall n$. (e) TMac-dec: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{4}, \forall n$ and using rank-decreasing strategy starting from $r_n = \text{round}(1.25r), \forall n$.



(a) MatComp (b) FaLRTC

(c) TMac-fix (d) TMac-inc

FIGURE 6. Phase transition plots for different methods on **3-way random low-rank tensors whose factors have power-law decaying singular values**. (a) FaLRTC: the tensor completion method in [16] that solves (5). (b) MatComp: the matrix completion solver LMaFit in [27] that solves (3) with $\mathbf{Z}$ corresponding to $\mathbf{M}_{(N)}$. (c) TMac-fix: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{3}$ and r_n fixed to r, $\forall n$. (d) TMac-inc: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{3}, \forall n$ and using rank-increasing strategy starting from $r_n = \text{round}(0.75r), \forall n$.



(a) SquareDeal (b) (7) solved by LMaFit

(c) TMac-fix (d) TMac-inc

FIGURE 7. Phase transition plots for different methods on **4-way random low-rank tensors whose factors have power-law decaying singular values**. (a) SquareDeal: the matrix completion solver FPCA [17] solves the model (6) proposed in [18]. (b) the matrix completion solver LMaFit [27] solves (7), which is a non-convex variant of (6). (c) TMac-fix: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{4}$ and r_n fixed to r, $\forall n$. (d) TMac-inc: Algorithm 1 solves (2) with $\alpha_n = \frac{1}{4}, \forall n$ and using rank-increasing strategy starting from $r_n = \text{round}(0.75r), \forall n$.

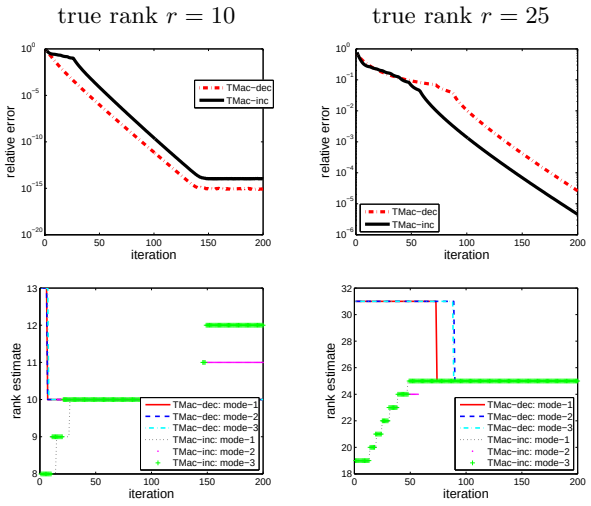


FIGURE 8. Convergence behavior of Algorithm 1 with two rank-adjusting strategies on Gaussian randomly generated $50 \times 50 \times 50$ tensors that have each mode rank to be r.

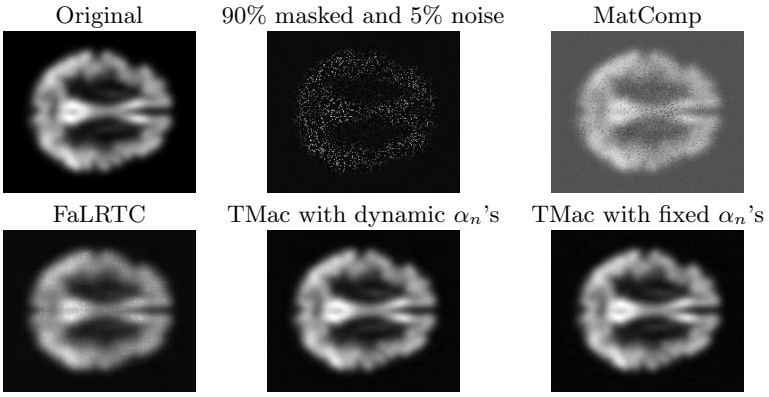


FIGURE 9. Brain MRI images: one original slice, the corresponding slice with 90% pixels missing and 5% Gaussian noise, and the recovered slices by different tensor completion methods.

TABLE 1. Brain MRI images: average results of 5 independent runs by different tensor completion methods for different settings of noise level σ's and sample ratio SR's.

	TMac with dynamic α_n's		TMac with fixed α_n's		MatComp		FaLRTC	
SR	relerr	time	relerr	time	relerr	time	relerr	time
noise level $\sigma = 0$								
10%	1.54e-03	1.79e+02	1.56e-03	1.82e+02	2.39e-01	2.40e+01	8.67e-02	2.65e+02
30%	1.14e-03	9.89e+01	1.15e-03	9.28e+01	2.13e-02	1.98e+01	9.59e-03	2.47e+02
50%	8.55e-04	8.11e+01	1.00e-03	7.63e+01	3.19e-03	1.84e+01	7.34e-03	2.01e+02
noise level $\sigma = 0.05$								
10%	2.15e-02	1.39e+02	2.15e-02	1.43e+02	2.55e-01	3.00e+01	1.15e-01	2.96e+02
30%	1.67e-02	9.04e+01	1.66e-02	8.71e+01	8.10e-02	3.12e+01	3.86e-02	1.43e+02
50%	1.62e-02	8.11e+01	1.61e-02	7.84e+01	4.35e-02	2.26e+01	3.66e-02	1.36e+02
noise level $\sigma = 0.10$								
10%	4.34e-02	1.26e+02	4.34e-02	1.30e+02	3.00e-01	3.33e+01	1.48e-01	2.46e+02
30%	3.37e-02	7.69e+01	3.33e-02	7.81e+01	1.66e-01	3.16e+01	7.19e-02	1.05e+02
50%	3.25e-02	7.22e+01	3.33e-02	7.81e+01	8.61e-02	2.12e+01	7.17e-02	1.01e+02

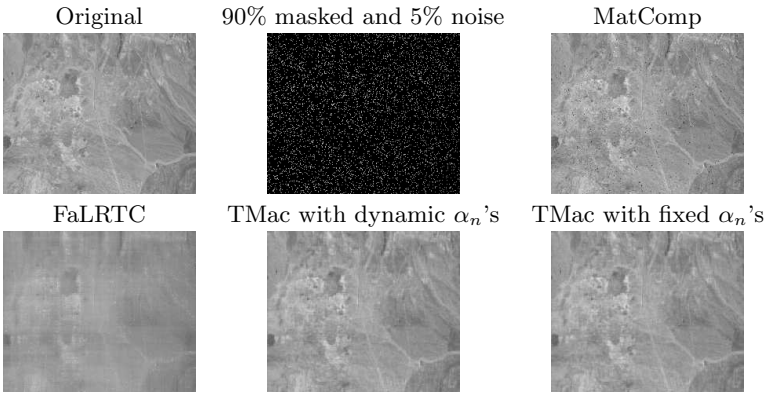


FIGURE 10. Hyperspectral images: one original slice, the corresponding slice with 90% pixels missing and 5% Gaussian noise, and the recovered slices by different tensor completion methods.

Table 2. Hyperspectral images: average results of 5 independent runs by different tensor completion methods for different settings of noise level σ's and sample ratio SR's.

	TMac with dynamic α_n's		TMac with fixed α_n's		MatComp		FaLRTC	
SR	relerr	time	relerr	time	relerr	time	relerr	time
noise level $\sigma = 0$								
10%	4.00e-02	8.88e+01	3.91e-02	8.28e+01	3.15e-01	1.22e+01	6.48e-02	1.70e+02
30%	2.40e-02	5.16e+01	3.08e-02	5.55e+01	6.25e-02	1.86e+01	3.10e-02	1.56e+02
50%	6.35e-03	4.54e+01	2.73e-02	5.71e+01	1.71e-02	2.68e+01	1.68e-02	1.64e+02
noise level $\sigma = 0.05$								
10%	4.14e-02	9.40e+01	4.12e-02	8.91e+01	3.16e-01	1.34e+01	6.65e-02	1.61e+02
30%	2.98e-02	5.76e+01	3.43e-02	6.08e+01	7.65e-02	2.71e+01	3.52e-02	1.53e+02
50%	2.25e-02	5.50e+01	3.01e-02	5.93e+01	4.14e-02	3.87e+01	2.45e-02	1.38e+02
noise level $\sigma = 0.10$								
10%	4.52e-02	9.18e+01	4.53e-02	9.04e+01	3.25e-01	1.47e+01	6.99e-02	1.60e+02
30%	3.53e-02	5.63e+01	3.77e-02	6.31e+01	1.00e-01	4.07e+01	4.35e-02	1.35e+02
50%	3.23e-02	5.49e+01	3.41e-02	5.10e+01	8.03e-02	4.08e+01	3.62e-02	1.23e+02

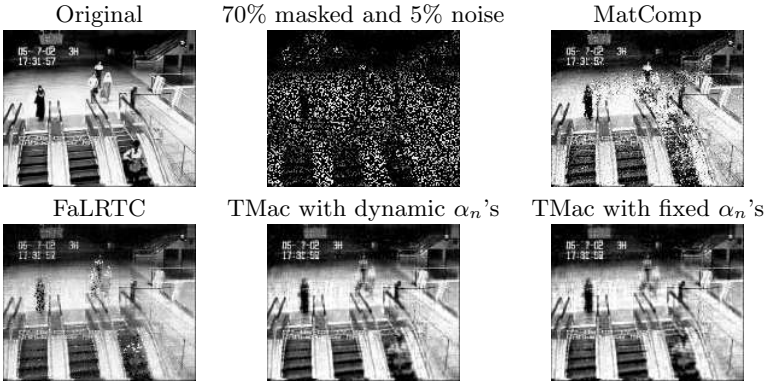


Figure 11. Grayscale video: one original frame, the corresponding frame with 70% pixels missing and 5% Gaussian noise, and the recovered frames by different tensor completion methods.

Table 3. Grayscale video: average results of 5 independent runs by different tensor completion methods for different settings of noise level σ's and sample ratio SR's.

	TMac with dynamic α_n's		TMac with fixed α_n's		MatComp		FaLRTC	
SR	relerr	time	relerr	time	relerr	time	relerr	time
noise level $\sigma = 0$								
10%	1.45e-01	9.23e+01	1.41e-01	9.77e+01	2.51e-01	1.88e+01	2.35e-01	1.22e+02
30%	9.85e-02	5.37e+01	1.01e-01	5.77e+01	1.41e-01	2.24e+01	1.28e-01	6.55e+01
50%	7.78e-02	4.75e+01	8.51e-02	5.27e+01	8.86e-02	1.80e+01	8.07e-02	7.38e+01
noise level $\sigma = 0.05$								
10%	1.45e-01	9.79e+01	1.41e-01	9.49e+01	2.47e-01	2.13e+01	2.36e-01	1.36e+02
30%	9.89e-02	5.62e+01	1.01e-01	5.59e+01	1.40e-01	1.99e+01	1.29e-01	6.95e+01
50%	7.80e-02	5.03e+01	8.54e-02	5.45e+01	8.90e-02	1.66e+01	8.27e-02	7.35e+01
noise level $\sigma = 0.10$								
10%	1.46e-01	9.68e+01	1.43e-01	5.45e+01	2.47e-01	1.93e+01	2.38e-01	1.42e+02
30%	1.00e-01	5.72e+01	1.02e-01	5.64e+01	1.57e-01	2.24e+01	1.32e-01	6.83e+01
50%	8.00e-02	5.29e+01	8.64e-02	5.02e+01	9.12e-02	1.81e+01	8.81e-02	6.79e+01

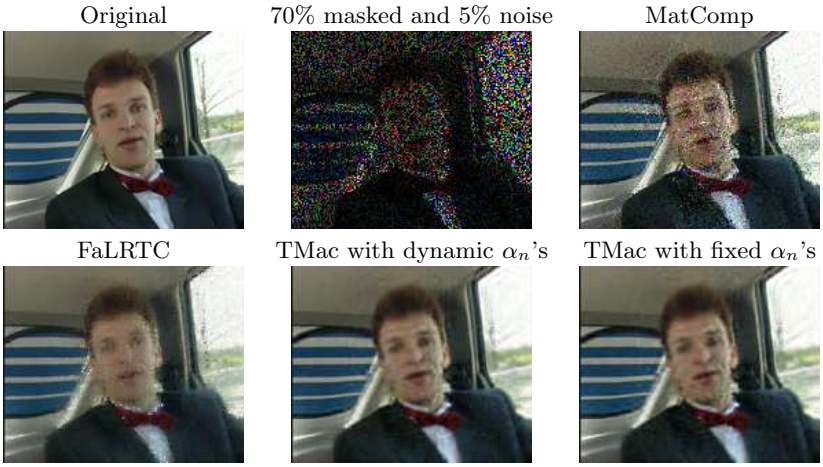


Figure 12. Color video: one original frame, the corresponding frame with 70% pixels missing and 5% Gaussian noise, and the recovered frames by different tensor completion methods.

Table 4. Color video: average results of 5 independent runs by different tensor completion methods for different settings of noise level σ's and sample ratio SR's.

	TMac with dynamic α_n's		TMac with fixed α_n's		MatComp		FaLRTC	
SR	relerr	time	relerr	time	relerr	time	relerr	time
noise level $\sigma = 0$								
10%	8.27e-02	2.64e+02	8.17e-02	2.65e+02	2.42e-01	4.78e+01	1.82e-01	3.19e+02
30%	5.75e-02	1.57e+02	5.81e-02	1.56e+02	1.04e-01	1.09e+02	8.40e-02	2.70e+02
50%	4.47e-02	1.46e+02	4.85e-02	1.52e+02	5.63e-02	5.41e+01	5.02e-02	2.68e+02
noise level $\sigma = 0.05$								
10%	8.38e-02	2.73e+02	8.22e-02	2.52e+02	2.43e-01	4.92e+01	1.83e-01	3.14e+02
30%	5.76e-02	1.52e+02	5.86e-02	1.58e+02	1.12e-01	8.68e+01	8.65e-02	2.39e+02
50%	4.56e-02	1.39e+02	4.90e-02	1.50e+02	6.00e-02	5.14e+01	5.38e-02	2.29e+02
noise level $\sigma = 0.10$								
10%	8.75e-02	2.46e+02	8.70e-02	2.43e+02	2.50e-01	4.77e+01	1.86e-01	2.89e+02
30%	5.94e-02	1.52e+02	6.06e-02	1.49e+02	1.38e-01	8.44e+01	9.27e-02	2.01e+02
50%	4.77e-02	1.40e+02	5.07e-02	1.47e+02	7.13e-02	4.92e+01	6.23e-02	2.01e+02

Acknowledgments. The authors thank two anonymous referees and the associate editor for their very valuable comments. Y. Xu is supported by NSF grant ECCS-1028790. R. Hao and Z. Su are supported by NSFC grants 61173103 and U0935004. R. Hao's visit to UCLA is supported by China Scholarship Council. W. Yin is partially supported by NSF grants DMS-0748839 and DMS-1317602, and ARO/ARL MURI grant FA9550-10-1-0567.

REFERENCES

[1] J. Cai, E. Candes and Z. Shen, A singular value thresholding algorithm for matrix completion, *SIAM J. Optim.,* **20** (2010), 1956–1982.

[2] E. Candès and B. Recht, Exact matrix completion via convex optimization, *Foundations of Computational Mathematics,* **9** (2009), 717–772.

[3] C. Chen, B. He and X. Yuan, Matrix completion via an alternating direction method, *IMA Journal of Numerical Analysis,* **32** (2012), 227–245.
[4] S. Gandy, B. Recht and I. Yamada, Tensor completion and low-n-rank tensor recovery via convex optimization, *Inverse Problems*, **27** (2011), 025010, 1–19.
[5] L. Grippo and M. Sciandrone, On the convergence of the block nonlinear Gauss-Seidel method under convex constraints, *Oper. Res. Lett.,* **26** (2000), 127–136.
[6] B. Jiang, S. Ma and S. Zhang, Tensor principal component analysis via convex optimization, *Mathematical Programming,* (2014), 1–35.
[7] H. A. L. Kiers, Towards a standardized notation and terminology in multiway analysis, *Journal of Chemometrics,* **14** (2000), 105–122.
[8] M. Kilmer, K. Braman, N. Hao and R. Hoover, Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging, *SIAM Journal on Matrix Analysis and Applications,* **34** (2013), 148–172.
[9] T. G. Kolda and B. W. Bader, Tensor decompositions and applications, *SIAM review,* **51** (2009), 455–500.
[10] T. G. Kolda, B. W. Bader and J. P. Kenny, *Higher-order web link analysis using multilinear algebra*, in *Data Mining,* Fifth IEEE International Conference on, IEEE, 2005.
[11] N. Kreimer and M. D. Sacchi, A tensor higher-order singular value decomposition for prestack seismic data noise reduction and interpolation, *Geophysics,* **77** (2012), V113–V122.
[12] D. Kressner, M. Steinlechner and B. Vandereycken, Low-rank tensor completion by riemannian optimization, *BIT,* **54** (2014), 447–468.
[13] M. Lai, Y. Xu and W. Yin, Improved iteratively reweighted least squares for unconstrained smoothed ℓ_q minimization, *SIAM Journal on Numerical Analysis,* **51** (2013), 927–957.
[14] N. Li and B. Li, Tensor completion for on-board compression of hyperspectral images, in *2010 17th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2010, 517–520.
[15] Q. Ling, Y. Xu, W. Yin and Z. Wen, Decentralized low-rank matrix completion, in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 2012, 2925–2928.
[16] J. Liu, P. Musialski, P. Wonka and J. Ye, Tensor completion for estimating missing values in visual data, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2012), 208–220.
[17] S. Ma, D. Goldfarb and L. Chen, Fixed point and bregman iterative methods for matrix rank minimization, *Mathematical Programming,* **128** (2011), 321–353.
[18] C. Mu, B. Huang, J. Wright and D. Goldfarb, Square deal: Lower bounds and improved relaxations for tensor recovery, preprint, `arXiv:1307.5870`, (2013).
[19] K. A. Patwardhan, G. Sapiro and M. Bertalmío, Video inpainting under constrained camera motion, *IEEE Transactions on Image Processing,* **16** (2007), 545–553.
[20] B. Recht, M. Fazel and P. A. Parrilo, Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization, *SIAM Review,* **52** (2010), 471–501.
[21] B. Romera-Paredes and M. Pontil, A new convex relaxation for tensor completion, preprint, `arXiv:1307.4653`, (2013).
[22] A. C. Sauve, A. O. Hero III, W. Leslie Rogers, S. J. Wilderman and N. H. Clinthorne, 3d image reconstruction for a compton spect camera model, *Nuclear Science, IEEE Transactions on*, **46** (1999), 2075–2084.
[23] J. Sun, H. Zeng, H. Liu, Y. Lu and Z. Chen, Cubesvd: a novel approach to personalized web search, in *Proceedings of the 14th international conference on World Wide Web,* ACM, 2005, 382–390.
[24] K. C. Toh and S. Yun, An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems, *Pacific Journal of Optimization,* **6** (2010), 615–640.
[25] P. Tseng, Convergence of a block coordinate descent method for nondifferentiable minimization, *Journal of Optimization Theory and Applications*, **109** (2001), 475–494.
[26] L. R. Tucker, Some mathematical notes on three-mode factor analysis, *Psychometrika,* **31** (1966), 279–311.
[27] Z. Wen, W. Yin and Y. Zhang, Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm, *Mathematical Programming Computation,* **4** (2012), 333–361.
[28] Z. Xing, M. Zhou, A. Castrodad, G. Sapiro and L. Carin, Dictionary learning for noisy and incomplete hyperspectral images, *SIAM Journal on Imaging Sciences,* **5** (2012), 33–56.

[29] Y. Xu and W. Yin, A block coordinate descent method for regularized multi-convex optimization with applications to nonnegative tensor factorization and completion, *SIAM Journal on Imaging Sciences*, **6** (2013), 1758–1789.
[30] Y. Xu and W. Yin, A globally convergent algorithm for nonconvex optimization based on block coordinate update, arXiv:1410.1386, (2014).
[31] Y. Xu, W. Yin, Z. Wen and Y. Zhang, An alternating direction algorithm for matrix completion with nonnegative factors, *Journal of Frontiers of Mathematics in China, Special Issue on Computational Mathematics*, **7** (2011), 365–384.
[32] Z. Zhang, G. Ely, S. Aeron, N. Hao and M. Kilmer, Novel factorization strategies for higher order tensors: Implications for compression and recovery of multi-linear data, preprint, arXiv:1307.0805v3, (2013).

Received December 2013; revised October 2014.

E-mail address: yangyang.xu@rice.edu
E-mail address: haoruru.math@gmail.com
E-mail address: wotaoyin@math.ucla.edu
E-mail address: zxsu@dlut.edu.cn