This article has been accepted for publication in IEEE Transactions on Image Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIP.2023.3310331

Double Auto-weighted Tensor Robust Principal Component Analysis

Yulong Wang, Kit Ian Kou, Hong Chen, Yuan Yan Tang, *Life Fellow, IEEE,* and Luoqing Li

***Abstract*—Tensor Robust Principal Component Analysis (TRPCA), which aims to recover the low-rank and sparse components from their sum, has drawn intensive interest in recent years. Most existing TRPCA methods adopt the tensor nuclear norm (TNN) and the tensor ℓ_1 norm as the regularization terms for the low-rank and sparse components, respectively. However, TNN treats each singular value of the low-rank tensor $\mathcal{L}$ equally and the tensor ℓ_1 norm shrinks each entry of the sparse tensor $\mathcal{S}$ with the same strength. It has been shown that larger singular values generally correspond to prominent information of the data and should be less penalized. The same goes for large entries in $\mathcal{S}$ in terms of absolute values. In this paper, we propose a Double Auto-weighted TRPCA (DATRPCA) method. Instead of using predefined and manually set weights merely for the low-rank tensor as previous works, DATRPCA *automatically and adaptively* assigns smaller weights and applies lighter penalization to significant singular values of the low-rank tensor and large entries of the sparse tensor *simultaneously*. We have further developed an efficient algorithm to implement DATRPCA based on the Alternating Direction Method of Multipliers (ADMM) framework. In addition, we have also established the convergence analysis of the proposed algorithm. The results on both synthetic and real-world data demonstrate the effectiveness of DATRPCA for low-rank tensor recovery, color image recovery and background modelling.**

***Index Terms*—Tensor robust PCA, tensor nuclear norm, double weight learning, low-dimensional structure.**

I. Introduction

BY exploiting the low-dimensional structure of high-dimensional data, Principal Component Analysis (PCA) [1] has achieved great success in pattern recognition and machine learning. Nonetheless, it has been shown to be sensitive to outliers [2]. To improve the robustness of PCA against outliers, the Robust PCA (RPCA) approach is proposed with strong theoretical guarantee and high efficiency [3]. Consider a corrupted data matrix $\mathbf{X} = \mathbf{L} + \mathbf{S} \in \mathbb{R}^{d_1 \times d_2}$, where $\mathbf{L} \in \mathbb{R}^{d_1 \times d_2}$ is low-rank and $\mathbf{S} \in \mathbb{R}^{d_1 \times d_2}$ is sparse. RPCA attempts to recover

This work was supported in part by the Science and Technology Development Fund, Macau SAR FDCT/085/2018/A2 and in part by the National Natural Science Foundation of China under Grant Nos. 62076041, 62276111, and 12071166. (Corresponding author: Kit Ian Kou).

Y. Wang is with the College of Informatics, Huazhong Agricultural University, Wuhan 430070, China, and also with the Hubei Engineering Technology Research Center of Agricultural Big Data, Huazhong Agricultural University, Wuhan 430070, China. K. Kou is with the Faculty of Science and Technology, University of Macau, Macau 999078, China. H. Chen is with the College of Science, Huazhong Agricultural University, Wuhan 430070, China. Y. Y. Tang is with the Faculty of Science and Technology, University of Macau, Macau 999078, China. L. Li is with the Faculty of Mathematics and Statistics, Hubei University, Wuhan 430062, Hubei, P. R. China. (e-mail: wangyulong6251@gmail.com; kikou@umac.mo; chenh@mail.hzau.edu.cn; yytang@umac.mo; lilq@hubu.edu.cn).

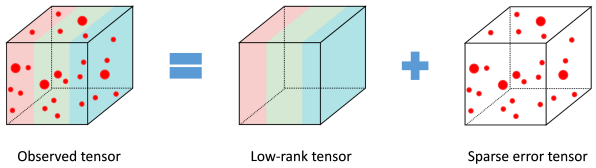


Figure 1: Illustration of TRPCA. The goal is to recover the low-rank and sparse tensors from their sum. The large red dots mean significant entries with large absolute values of the sparse tensor.

$\mathbf{L}$ and $\mathbf{S}$ by solving the following optimization problem

$$\min_{\mathbf{L},\mathbf{S}\in\mathbb{R}^{d_1\times d_2}} \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 \quad \text{s.t.} \quad \mathbf{X} = \mathbf{L} + \mathbf{S}, \tag{1}$$

where $\|\mathbf{L}\|_*$ denotes the nuclear norm (i.e., sum of singular values) of $\mathbf{L}$ and $\|\mathbf{S}\|_1$ represents the ℓ_1 norm (i.e., sum of absolute values of each entry) of $\mathbf{S}$, and λ is a positive parameter. It has been shown that the RPCA problem (1) can be solved efficiently in polynomial time [3].

Owing to its effectiveness and efficiency, RPCA has drawn dramatic interest and a number of variants have been proposed in the past decade [4]–[7]. However, a key limitation of RPCA methods is that they are confined to 2-way (matrix) data and cannot handle multi-way (tensor) data directly, which are ubiquitous in various applications. For instance, a color image is a 3-way data with row, column and color modes; a hyperspectral image can be indexed by two spatial variables and a spectral band variable; a grayscale video is also a 3-way data with row, column and temporal modes. To apply RPCA for multi-way data, one has to restructure the data into a matrix in advance, which may destroy the multi-way structure and lead to information and performance loss [8].

To extend RPCA to multi-way (tensor) data, a variety of tensor extensions of RPCA methods have been proposed in recent years [9]–[17]. Fig. 1 gives an intuitive illustration of the problem of the tensor RPCA. One of the main challenges of generalizing RPCA to tensor data lies in the definition of the tensor rank, which is more sophisticated than the matrix rank. Some popular definitions of tensor rank include the CP (CANDECOMP/PARAFAC) rank [18], the Tucker rank [19] and the tubal rank [20]–[22].

The CP rank is defined as the minimum number of rank one tensor decomposition [18]. Since the computation of the CP rank and its convex relaxation is intractable, it is challenging to apply it for RPCA [18]. As for the Tucker rank [19] of a k-way tensor $\mathcal{L}$, it is defined by a k-dimensional vector of which the i-th entry is the rank of the mode-i matricization $\mathbf{L}^{\{i\}}$ of

$\mathcal{L}$, i.e., $\text{rank}_{tc}(\mathcal{L}) = \left[\text{rank}\left(\mathbf{L}^{\{1\}}\right), \cdots, \text{rank}\left(\mathbf{L}^{\{k\}}\right)\right]$. Tucker rank based TRPCA methods [13], [23], [24] attempt to recover the low-Tucker-rank tensor $\mathcal{L}$ by minimizing the Sum of Nuclear Norms (SNN) [13] of $\mathcal{L}$, which is a convex surrogate of the Tucker rank. Nonetheless, since SNN is not the convex envelop of $\sum_{i=1}^{k} \text{rank}\left(\mathbf{L}^{\{i\}}\right)$, the solution of the SNN based TRPCA methods can be suboptimal [8]. Recently, Kilmer *et al.* [20], [21] propose the tensor-tensor product (or t-product for short) and the tubal rank as a decent alternative of the CP rank and the Tucker rank.

The three tensor ranks above attempt to capture the structural information of a tensor from different perspectives. But each of them has its own advantages and limitations. For example, the CP rank and Tucker rank can be used for any N-way tensor while the tubal rank is only defined for 3-way tensors. However, the CP rank is generally NP-hard to compute and its convex envelope is also intractable. Although the Tucker rank can be computed efficiently using the matrix Singular Value Decomposition (SVD), the rank tuple in the definition of the Tucker rank is difficult to be used in practice. In addition, the commonly used convex surrogate of the Tucker rank, i.e., Sum of Nuclear Norms (SNN) [13] is not the convex envelope. In contrast, the tubal rank is based on the T-SVD and can be computed efficiently using the Discrete Fourier Transform (DFT). Furthermore, the tubal rank induced Tensor Nuclear Norm (TNN) own the same tight recovery bound as the matrix cases. It has been shown that color images can be approximated by low-rank tensors using the definitions of the three tensor ranks above [8]. Since the frames of the background are highly correlated in general and thus can also be modeled as a low rank tensor [8]. Thus, the low-rank forms of the three tensor ranks are able to represent the natural color images and video backgrounds.

Based on the tubal rank, Lu *et al.* [8] define a new Tensor Nuclear Norm (TNN) and put forward a new Tensor RPCA (TRPCA) method with impressive performance for color image recovery and background modelling. To further improve TRPCA, Kong *et al.* [9] develop the tensor Schatten-p norm as a tighter regularizer in lieu of TNN for low-rank tensor recovery. To capture the spectral-spatial information of hyperspectral images, Chang *et al.* [10] propose a weighted low-rank tensor recovery model for hyperspectral image restoration. To improve the modelling power, Zhou and Cheung [25] come up with a Bayesian TRPCA approach by incorprating the low-tubal-rank prior and sparsity prior into the Bayesian framework. To improve the efficiency, Qiu *et al.* [11] put forward a fast nonconvex TRPCA approach with theoretical guarantee.

A. Paper Contributions

Despite their empirical success in various applications, existing TRPCA methods can be further improved upon. For example, most existing TRPCA methods penalize each singular value of the low-rank tensor $\mathcal{L}$ equally and overlook the differences of these singular values. In many practical applications, singular values generally have clear physical meanings and should be treated differently [26]. To alleviate such limitation, some variants of TRPCA methods have been developed [16], [27]. For example, Gao *et al.* [16] put forward an Enhanced TRPCA (ETRPCA) method by minimizing the weighted tensor Schatten p-norm of the low-rank tensor instead of the TNN. However, ETRPCA requires predefined and fixed weights for different singular values. Furthermore, the weights need to be set manually in advance. This restricts its capability and flexibility in practice. Jiang *et al.* [27] i develops the Partial Sum of the Tubal Nuclear Norm (PSTNN) as a surrogate of TNN to reduce the rank deficiency of TNN. Specially, minimizing PSTNN only shrinks the small singular values without any actions on the large ones. This implicitly assumes that the large singular values have nothing to do with the content of the tensor and such assumption may be too restrict in reality [16]. In addition, the methods aforementioned only consider the differences among singular values and ignore the differences among the entries of the sparse tensor. Concretely, they generally adopt the Tensor ℓ_1 Norm (TL1N) and regularize the entries of the sparse tensor equally. Such strategy may penalize the large important entries too heavily and lead to suboptimal results [28].

The analysis of the existing TRPCA methods above motivate us to raise an interesting question: is it possible to devise an improved TRPCA method which penalizes *democratically* different singular values of $\mathcal{L}$ and distinct entries of $\mathcal{S}$ in an *automatic and adaptive* manner? This work attempts to provide an positive answer to this question. In summary, the contributions of this paper are as follows:

1. We propose a novel TRPCA method referred to as DATRPCA (Double Auto-weighted TRPCA), which takes into account the significance of different singular values of the low-rank tensor $\mathcal{L}$ and entries of the sparse tensor $\mathcal{S}$ *simultaneously*. This gives rise to the capability of DATRPCA in preserving prominent information of the tensor data well.
2. We put forward a new iterative weight learning strategy which *automatically and adaptively* assigns smaller weights and applies lighter penalization to significant singular values of the low-rank tensor and important entries of the sparse tensor simultaneously. This greatly facilitates the practical utility of the proposed method in reality.
3. We devise an efficient optimization algorithm for solving the DATRPCA problem. We have also proved the convergence of the proposed algorithm. The results show that the proposed algorithm outperforms TRPCA significantly.

B. Paper Organization

The remainder of the paper is arranged as below. Section II introduces some notations and preliminaries. In Section III, we describe the proposed method, as well as the optimization algorithm and convergence analysis. Section IV presents the experiments on both synthetic and real-world databases. Finally, Section V concludes. The proofs of theoretical results are provided in the supplementary material due to space limitation.

Table I: Key notations used in this paper.

Notation	Description
$\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$	tensor data
$\mathcal{L} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$	low-rank tensor
$\mathcal{S} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$	sparse tensor
$\mathcal{X}_{ijk}$ or $\mathcal{X}(i,j,k)$	(i,j,k)-th entry of $\mathcal{X}$
$\mathcal{X}(i,:,:)$	i-th horizontal slice of $\mathcal{X}$
$\mathcal{X}(:,j,:)$	j-th lateral slice of $\mathcal{X}$
$\mathcal{X}(:,:,k)$ or $\mathbf{X}^{(k)}$	k-th frontal slice of $\mathcal{X}$
DFT	Discrete Fourier Transform
$\bar{\mathcal{X}} \in \mathbb{C}^{d_1 \times d_2 \times d_3}$	DFT of $\mathcal{X}$ along the 3-rd dimension
$\bar{\mathbf{X}}^{(k)} \in \mathbb{C}^{d_1 \times d_2}$	k-th frontal slice of $\bar{\mathcal{X}}$
$\|\mathcal{X}\|_*$	Tensor Nuclear Norm (TNN)
$\|\mathcal{X}\|_1$	Tensor ℓ_1 Norm (TL1N)

II. Notations and Preliminaries

A. Notations

To begin with, we introduce the notations used throughout the paper for better readability. We denote scalars by lowercase letters, e.g., x, vectors by boldface lower-case letters, e.g., $\mathbf{x}$, matrices by boldface upper-case letters, e.g., $\mathbf{X}$, and tensors by boldface Euler script letters, e.g., $\mathcal{X}$, respectively. For a 3-order tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, we denote its (i,j,k)-th entry by $\mathcal{X}(i,j,k)$, i-th horizontal slice by $\mathcal{X}(i,:,:)$, j-th lateral slice by $\mathcal{X}(:,j,:)$, k-th frontal slice by $\mathcal{X}(:,:,k)$, respectively. For brevity, the frontal slice $\mathcal{X}(:,:,k)$ is often denoted as $\mathbf{X}^{(k)}$. Let $\bar{\mathcal{X}}$ denote the Discrete Fourier Transform (DFT) of $\mathcal{X}$ along the 3-rd dimension, i.e., $\bar{\mathcal{X}} = fft(\mathcal{X}, [], 3)$. The tensor ℓ_1 norm and Frobenius norm of $\mathcal{X}$ are defined as $\|\mathcal{X}\|_1 = \sum_{ijk} |\mathcal{X}_{ijk}|$ and $\|\mathcal{X}\|_F = \sqrt{\sum_{ijk} |\mathcal{X}_{ijk}|^2}$, respectively. Table I summarizes the key notations and their description used in the paper.

B. T-Product

To define the tensor-tensor product (or t-product), we first introduce some useful tensor operators.

Definition 1. *(**The fold and unfold operators** [20]) For any 3-order tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, we define*

$$\text{unfold}(\mathcal{X}) = \begin{bmatrix} \mathbf{X}^{(1)} \\ \mathbf{X}^{(2)} \\ \vdots \\ \mathbf{X}^{(d_3)} \end{bmatrix}, \quad \text{fold}(\text{unfold}(\mathcal{X})) = \mathcal{X}. \tag{2}$$

Definition 2. *(**The bcirc operator** [20]) The block circulant matrix of any 3-order tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is defined as*

$$\text{bcirc}(\mathcal{X}) = \begin{bmatrix} \mathbf{X}^{(1)} & \mathbf{X}^{(d_3)} & \cdots & \mathbf{X}^{(2)} \\ \mathbf{X}^{(2)} & \mathbf{X}^{(1)} & \cdots & \mathbf{X}^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}^{(d_3)} & \mathbf{X}^{(2)} & \cdots & \mathbf{X}^{(1)} \end{bmatrix} \tag{3}$$

where $\mathbf{X}^{(k)}$ denotes the k-th frontal slice of $\mathcal{X}$.

Definition 3. *(**T-product** [20]) The t-product of any two 3-order tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $\mathcal{Y} \in \mathbb{R}^{d_2 \times l \times d_3}$ is defined to be a tensor of size $d_1 \times l \times d_3$*

$$\mathcal{X} * \mathcal{Y} = \text{fold}(\text{bcirc}(\mathcal{X}) \cdot \text{unfold}(\mathcal{Y})). \tag{4}$$

In fact, the t-product $\mathcal{Z} = \mathcal{X} * \mathcal{Y}$ can be computed efficiently with the aid of fft [8]. Specifically, let $\bar{\mathcal{X}} = fft(\mathcal{X}, [], 3)$ and $\bar{\mathcal{Y}} = fft(\mathcal{Y}, [], 3)$. We can first calculate the k-th frontal slice of $\bar{\mathcal{Z}}$ as $\bar{\mathbf{Z}}^{(k)} = \bar{\mathbf{X}}^{(k)} \bar{\mathbf{Y}}^{(k)}$, where $\bar{\mathbf{X}}^{(k)}$ and $\bar{\mathbf{Y}}^{(k)}$ are the k-th frontal slice of $\bar{\mathcal{X}}$ and $\bar{\mathcal{Y}}$, respectively. Then we can obtain $\mathcal{Z} = ifft(\bar{\mathcal{Z}}, [], 3)$ using the inverse discrete fourier transform.

C. T-SVD and Tensor Nuclear Norm

Before introducing the Tensor-SVD (T-SVD) and Tensor Nuclear Norm (TNN), it is necessary to introduce some useful definitions associated with tensors.

Definition 4. *(**Identity tensor** [20]) The tensor can be factorized as The identity tensor $\mathcal{I} \in \mathbb{R}^{d \times d \times d_3}$ is the tensor with its first frontal slice being the $d \times d$ identity matrix, and other frontal slices being all zeros.*

Definition 5. *(**Orthogonal tensor** [20]) A tensor $\mathcal{Q} \in \mathbb{R}^{d \times d \times d_3}$ is orthogonal if it satisfies $\mathcal{Q}^* * \mathcal{Q} = \mathcal{Q} * \mathcal{Q}^* = \mathcal{I}$, where $\mathcal{Q}^* \in \mathbb{R}^{d_2 \times d_1 \times d_3}$ denotes the conjugate transpose of $\mathcal{Q}$.*

Definition 6. *(**F-diagonal tensor** [20]) A tensor is called f-diagonal if each of its frontal slices is a diagonal matrix.*

Definition 7. *(**Tensor Singular Value Decompositioni, T-SVD** [8]) The tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ can be factorized as*

$$\mathcal{X} = \mathcal{U} * \mathcal{\Sigma} * \mathcal{V}^*, \tag{5}$$

where $\mathcal{U} \in \mathbb{R}^{d_1 \times d_1 \times d_3}$, $\mathcal{V} \in \mathbb{R}^{d_2 \times d_2 \times d_3}$ are orthogonal, and $\mathcal{\Sigma} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is an f-diagonal tensor.

Definition 8. *(**Tensor Nuclear Norm, TNN** [8]) Let $\mathcal{X} = \mathcal{U} * \mathcal{\Sigma} * \mathcal{V}^*$ be the t-SVD of $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $d = \min(d_1, d_2)$. The tensor nuclear norm of $\mathcal{X}$ is defined as*

$$\|\mathcal{X}\|_* = \sum_{i=1}^{d} \mathcal{\Sigma}(i,i,1) = \frac{1}{d_3} \sum_{k=1}^{d_3} \sum_{i=1}^{d} \sigma_i \left(\bar{\mathbf{X}}^{(k)} \right), \tag{6}$$

where $\sigma_i \left(\bar{\mathbf{X}}^{(k)} \right)$ is the i-th singular value of $\bar{\mathbf{X}}^{(k)}$.

According to the definition above, we can compute the TNN efficiently. Concretely, we first calculate $\bar{\mathcal{X}} = fft(\mathcal{X}, [], 3)$. Then we compute the singular values of its frontal slice $\bar{\mathbf{X}}^{(k)}$. Finally, we can obtain the TNN of $\|\mathcal{X}\|_*$.

D. Tensor Robust Principal Component Analysis

With the definitions above, we are now ready to introduce the TRPCA model. Given the sum $\mathcal{X}$ of a low-rank tensor $\mathcal{L}_0$ and the sparse tensor $\mathcal{S}_0$, i.e., $\mathcal{X} = \mathcal{L}_0 + \mathcal{S}_0$, TRPCA aims to recover $\mathcal{L}_0$ and $\mathcal{S}_0$ from $\mathcal{X}$ simultaneously. To this end, it solves the following convex program

$$\min_{\mathcal{L}, \mathcal{S} \in \mathbb{R}^{d_1 \times d_2 \times d_3}} \|\mathcal{L}\|_* + \lambda \|\mathcal{S}\|_1 \quad \text{s.t.} \quad \mathcal{X} = \mathcal{L} + \mathcal{S}. \tag{7}$$

Note that TRPCA is a generalization of matrix RPCA for tensor data. Compared with RPCA, TRPCA can preserve

the natural multi-dimensional structure information of tensor data better. Due to its efficacy, TRPCA has shown promising performance in color image recovery, background modelling and so on.

III. Double Auto-weighted TRPCA

In this section, we propose a Double Auto-weighted TRPCA (DATRPCA) method for low-rank tensor recovery. Firstly, we introduce the motivation and objective function of DATRPCA. Secondly, we devise an efficient optimization algorithm to implement DATRPCA. Finally, we present the convergence analysis of the proposed algorithm.

As pointed out in the literature [16], [26], larger singular values of the low-rank tensor $\mathcal{L}$ generally correspond to important information of $\mathcal{L}$, such as major edge and texture information of images [26]. Thus, larger singular values should be shrunk less while smaller ones should be shrunk more. For this reason, we first introduce the definition of weighted tensor nuclear norm (WTNN).

Definition 9. *(Weighted Tensor Nuclear Norm, WTNN) Given any tensor $\mathcal{L} \in \mathbb{R}^{d_1\times d_2\times d_3}$ and the weight matrix $\mathbf{W} \in \mathbb{R}^{d_3\times d}$, $d = \min(d_1, d_2)$, the weighted tensor nuclear norm of $\mathcal{L}$ is defined as*

$$\|\mathcal{L}\|_{\mathbf{W},*} = \frac{1}{d_3}\sum_{k=1}^{d_3}\sum_{i=1}^{d} W_{ki}\sigma_i\left(\bar{\mathbf{L}}^{(k)}\right), \tag{8}$$

where $\bar{\mathcal{L}}$ is the result by applying DFT (Discrete Fourier Transform) on $\mathcal{L}$ along the third dimension, $\bar{\mathbf{L}}^{(k)}$ is the k-th frontal slice of $\bar{\mathcal{L}}$, and $\sigma_i\left(\bar{\mathbf{L}}^{(k)}\right)$ is the i-th singular value of $\bar{\mathbf{L}}^{(k)}$.

Remark 1. *It is worth pointing out that the definition of WTNN is different from the weighted tensor Schatten p-norm in previous work [16], which is defined as $\|\mathcal{L}\|_{\mathbf{w},S_p} = \left(\sum_{k=1}^{d_3}\sum_{i=1}^{d} w_i\sigma_i^p\left(\bar{\mathbf{L}}^{(k)}\right)\right)^{\frac{1}{p}}$. When $p = 1$, it reduces to*

$$\|\mathcal{L}\|_{\mathbf{w},S_1} = \sum_{k=1}^{d_3}\sum_{i=1}^{d} w_i\sigma_i\left(\bar{\mathbf{L}}^{(k)}\right). \tag{9}$$

Note that Eq. (9) assigns the same weights to the singular values $\sigma_i\left(\bar{\mathbf{L}}^{(k)}\right), k = 1,\cdots,d_3$ in the same location of different frontal slices of $\bar{\mathcal{L}}$. This implicitly assumes that they have the same significance. In comparison, the proposed WTNN does not rely on such assumption, making it have better flexibility.

Analogously, we can define the weighted tensor ℓ_1 norm (WTL1N) to take into account the significance of different entries of the sparse tensor $\mathcal{S}$.

Definition 10. *(Weighted Tensor ℓ_1 Norm, WTL1N) Given $\mathcal{S} \in \mathbb{R}^{d_1\times d_2\times d_3}$ and the weight tensor $\mathcal{W} \in \mathbb{R}^{d_1\times d_2\times d_3}$, the weighted tensor ℓ_1 norm of $\mathcal{X}$ is defined as*

$$\|\mathcal{S}\|_{\mathcal{W},1} = \sum_{i=1}^{d_1}\sum_{j=1}^{d_2}\sum_{k=1}^{d_3}\left|\mathcal{W}_{ijk}\mathcal{S}_{ijk}\right|. \tag{10}$$

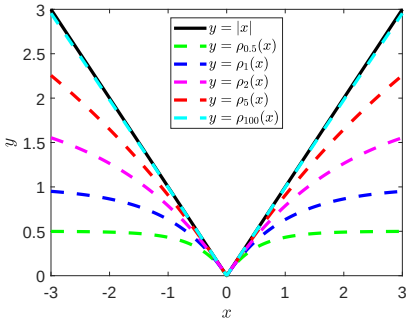


Figure 2: Comparison between the absolute function $|x|$ and $\rho_\gamma(x)$ with different values of γ.

By incorporating both WTNN and WTL1N into the TRPCA framework, we have the following double weighted TRPCA model

$$\min_{\mathcal{L},\mathcal{E}\in\mathbb{R}^{d_1\times d_2\times d_3}} \|\mathcal{L}\|_{\mathbf{W},*} + \lambda\|\mathcal{S}\|_{\mathcal{W},1} \quad \text{s.t.} \quad \mathcal{X} = \mathcal{L} + \mathcal{S}, \tag{11}$$

where $\mathbf{W} \in \mathbb{R}^{d_3\times d}$ is the weight matrix for $\mathcal{L}$ and $\mathcal{W} \in \mathbb{R}^{d_1\times d_2\times d_3}$ is a weight tensor for $\mathcal{S}$.

Compared to the TRPCA model (7), the DWTRPCA model (11) considers the significance of different singular values of $\mathcal{L}$ and entries of $\mathcal{S}$ simultaneously by assigning distinct weights to them. However, a critical issue is how to select the appropriate weight matrix $\mathbf{W}$ and weight tensor $\mathcal{W}$. A natural choice is to set them manually. Nevertheless, it is sophisticated and difficult to select appropriate weights for various data in a manual way. In addition, fixed weights overlook the information of the data itself and may lead to suboptimal solutions. For this reason, in the next section we develop a new weight learning strategy which iteratively learns the weight matrix $\mathbf{W}$ and the weight tensor $\mathcal{W}$ from data in an adaptive and automatic manner. This greatly facilitates the practical utility of the proposed method in reality.

A. Objective Function and Weight Learning

In this part, we develop a double auto-weighted learning method. Consider the exponential function

$$\rho_\gamma(x) = \gamma\left(1 - \exp\left(-|x|/\gamma\right)\right), \tag{12}$$

where γ is a positive parameter. Note that the Taylor expansion of $\rho_\gamma(x)$ is

$$\rho_\gamma(x) = \sum_{n=1}^{\infty}\frac{1}{n!}\frac{1}{\gamma^{n-1}}(-1)^{n+1}|x|^n = |x| + o(|x|/\gamma), \tag{13}$$

As $\gamma \to \infty$, $\rho_\gamma(x) \to |x|$. Note from Fig. 2 that $\rho_\gamma(x)$ increases slower than $|x|$ for large values. Therefore, if $\rho_\gamma(x)$ is adopted as a penalty function, large values of x will be less penalized compared with $|x|$. This is a desired property to construct better alternatives than the TNN and the tensor ℓ_1 norm.

Proposition 1. *There exists a convex conjugate function ψ of $\rho_\gamma(x)$ such that*

$$\rho_\gamma(x) = \min_{w\in\mathbb{R}_+}\left(w|x| + \psi(w)\right), \tag{14}$$

and for a fixed x, the minimum is reached at $w = \exp(-|x|/\gamma)$.

Remark 2. *Proposition 1 provides important clues to learn the weights $\mathbf{W}$ and $\boldsymbol{\mathcal{W}}$ from the data in an adaptive and automatic manner. Several prior works consider other alternative nonconvex regularizers such as ℓ_p $(0 < p < 1)$ [9] and log-sum [29]. However, as σ_{ik} increases to infinity, the exponential regularizer $\rho_\gamma(x)$ is upper bounded while the ℓ_p $(0 < p < 1)$ and log-sum regularizer also increase to infinity. In addition, when σ_{ik} increases to very large magnitude, the weight function of the exponential regularizer decreases much faster than those of ℓ_p $(0 < p < 1)$ and log-sum. Compared with ℓ_p $(0 < p < 1)$ and log-sum, $\rho_\gamma(x)$ penalizes less on large singular values, which usually contain prominent information of the data. In the supplementary material, we have further clarified the reason of choosing $\rho_\gamma(x)$ as the regularizer.*

Learning weights for $\boldsymbol{\mathcal{L}}$. First we construct an alternative low-rank regularization function for the low-rank tensor $\boldsymbol{\mathcal{L}}$ as follows.

$$R_L(\boldsymbol{\mathcal{L}}) = \frac{1}{d_3}\sum_{k=1}^{d_3}\sum_{i=1}^{d}\rho_{\gamma_k^L}\left(\sigma_i\left(\bar{\mathbf{L}}^{(k)}\right)\right), \tag{15}$$

where γ_k^L is a positive parameter for the k-th frontal slice $\bar{\mathbf{X}}^{(k)}$ of $\bar{\boldsymbol{\mathcal{X}}}$. According to Proposition 1, we have

$$R_L(\boldsymbol{\mathcal{L}}) = \min_{\mathbf{W}\in\mathbb{R}_+^{d_3\times d}} \frac{1}{d_3}\sum_{k=1}^{d_3}\sum_{i=1}^{d}\left(W_{ki}\sigma_i\left(\bar{\mathbf{L}}^{(k)}\right) + \psi(W_{ki})\right), \tag{16}$$

where the minimum is reached at $W_{ki} = \exp\left(-\sigma_i\left(\bar{\mathbf{L}}^{(k)}\right)/\gamma_k^L\right)$. Thus, given $\boldsymbol{\mathcal{L}}$ in the current iteration, the weight matrix $\mathbf{W}$ can be updated by

$$\mathbf{W} = F_L(\boldsymbol{\mathcal{L}}). \tag{17}$$

where the function $F_L : \mathbb{R}^{d_1\times d_2\times d_3} \to \mathbb{R}^{d_3\times d}$ is defined such that $(F_L(\boldsymbol{\mathcal{L}}))_{ki} = \exp\left(-\sigma_i\left(\bar{\mathbf{L}}^{(k)}\right)/\gamma_k^L\right)$. The parameter γ_k^L can be chosen in an adaptive way. In this work, we set it by

$$\gamma_k^L = c_L * \text{mean}\left(\sigma_1\left(\bar{\mathbf{L}}^{(k)}\right), \cdots, \sigma_d\left(\bar{\mathbf{L}}^{(k)}\right)\right), \tag{18}$$

where c_L is a positive constant and set as $c_L = 2$ in the experiments. Note that in Eq. (17) for each large singular value $\sigma_i\left(\bar{\mathbf{L}}^{(k)}\right)$, it will get small weight and will be less shrinked. As pointed out in recent works [16], [26], large singular values often correspond to salient information of the data. Thus, the weight learning strategy above can well preserve the salient content of the tensor data, such as a color image. Specifically, by Eq. (17) small singular values will be assigned large weights and more shrinked. In the extreme case, for a vanishing singular value, its weight reaches the maximum value 1.

Remark 3. *The parameter γ_k^L controls the weight values for each singular values. If the parameter γ_k^L is too large, the weights for different singular values are close and large. For example, when $\gamma_k^L \to \infty$, each weight tends to 1. If the parameter γ_k^L is too small, the weights for different singular values are also close and small. For instance, when $\gamma_k^L \to 0$, each weight equals to 0. Therefore, it is important to choose appropriate value of γ_k^L to get desired weights. Instead of choosing it manually, we set it in an adaptive way by Eq. (18). Such strategy can make the weight values in a reasonable interval and has been widely and commonly used in robust regression and robust sparse representation [30]. In addition, it is also user-friendly and make users get rid of tuning it for different data case by case.*

Learning weights for $\boldsymbol{\mathcal{S}}$. Based on $\rho_\gamma(x)$, we build the regularization term for the sparse tensor $\boldsymbol{\mathcal{S}}$ as

$$R_S(\boldsymbol{\mathcal{S}}) = \sum_{i=1}^{d_1}\sum_{j=1}^{d_2}\sum_{k=1}^{d_3}\rho_{\gamma^S}\left(\boldsymbol{\mathcal{S}}_{ijk}\right), \tag{19}$$

where γ^S is another positive parameter. According to Proposition 1, we have

$$R_S(\boldsymbol{\mathcal{S}}) = \min_{\boldsymbol{\mathcal{W}}\in\mathbb{R}_+^{d_1\times d_2\times d_3}} \sum_{i=1}^{d_1}\sum_{j=1}^{d_2}\sum_{k=1}^{d_3}\left(\boldsymbol{\mathcal{W}}_{ijk}|\boldsymbol{\mathcal{S}}_{ijk}| + \psi(\boldsymbol{\mathcal{W}}_{ijk})\right), \tag{20}$$

where the minimum is reached at $\boldsymbol{\mathcal{W}}_{ijk} = \exp\left(-|\boldsymbol{\mathcal{S}}_{ijk}|/\gamma^S\right)$. Therefore, given $\boldsymbol{\mathcal{S}}$ in the current iteration, the weight tensor $\boldsymbol{\mathcal{W}}$ can be updated by

$$\boldsymbol{\mathcal{W}} = G_S(\boldsymbol{\mathcal{S}}), \tag{21}$$

where the function $G_S : \mathbb{R}^{d_1\times d_2\times d_3} \to \mathbb{R}^{d_1\times d_2\times d_3}$ is defined such that $(G_S(\boldsymbol{\mathcal{S}}))_{ijk} = \exp\left(-|\boldsymbol{\mathcal{S}}_{ijk}|/\gamma^S\right)$. The parameter γ^S can also be chosen adaptively by

$$\gamma^S = c_S * \text{mean}\left(|\boldsymbol{\mathcal{S}}_{ijk}|,\ (i,j,k) \in [d_1]\times[d_2]\times[d_3]\right), \tag{22}$$

where c_S is a positive constant and set as $c_S = 2$ in the experiments. Here $[d_1]\times[d_2]\times[d_3] := \{(i,j,k)|i = 1,\cdots,d_1, j = 1,\cdots,d_2, k = 1,\cdots,d_3\}$.

Objective function. By incorporating $R_L(\boldsymbol{\mathcal{L}})$ and $R_S(\boldsymbol{\mathcal{S}})$ into the TRPCA framework, we have the following model

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{E}}\in\mathbb{R}^{d_1\times d_2\times d_3}} R_L(\boldsymbol{\mathcal{L}}) + \lambda R_S(\boldsymbol{\mathcal{S}}) \ \text{ s.t. } \ \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}}. \tag{23}$$

According to Proposition 1, the model (23) can be reformulated as the following weighted version

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{E}},\mathbf{W},\boldsymbol{\mathcal{W}}} \|\boldsymbol{\mathcal{L}}\|_{\mathbf{W},*} + \lambda\|\boldsymbol{\mathcal{S}}\|_{\boldsymbol{\mathcal{W}},1} + \Psi_M(\mathbf{W}) + \Psi_T(\boldsymbol{\mathcal{W}}) \ \text{ s.t. } \ \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}}. \tag{24}$$

where $\Psi_M(\mathbf{W}) \in \mathbb{R}$ and $\Psi_T(\boldsymbol{\mathcal{W}})$ are defined such that $(\Psi_M(\mathbf{W}))_{ki} = \psi(W_{ki})$ and $(\Psi_T(\boldsymbol{\mathcal{W}}))_{ijk} = \psi(\boldsymbol{\mathcal{W}}_{ijk})$. Since both $\mathbf{W}$ and $\boldsymbol{\mathcal{W}}$ can be learned *automatically* and *adaptively*, the model (24) is referred to as Double Auto-weighted TRPCA (DATRPCA).

B. Optimization

In this part, we develop an efficient optimization algorithm to implement the proposed DATRPCA method based on the ADMM framework. It is worth pointing out that unlike previous works using predefined and fixed weights, both of the weight matrix $\mathbf{W}$ and the weight tensor $\boldsymbol{\mathcal{W}}$ are variables, which are also updated in the iterations. This makes the convergence analysis of the proposed algorithm more challenging. The Lagrangian function of the DATRPCA model (24) is

$$\begin{aligned} L(\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}},\mathbf{W},\boldsymbol{\mathcal{W}},\boldsymbol{\mathcal{Y}},\mu) &= \|\boldsymbol{\mathcal{L}}\|_{\mathbf{W},*} + \lambda\|\boldsymbol{\mathcal{S}}\|_{\boldsymbol{\mathcal{W}},1} + \Psi_M(\mathbf{W}) \\ &+ \Psi_T(\boldsymbol{\mathcal{W}}) + \frac{\mu}{2}\|\boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{Y}}/\mu\|_F^2 - \frac{\mu}{2}\|\boldsymbol{\mathcal{Y}}/\mu\|_F^2, \end{aligned} \tag{25}$$

where $\mathcal{Y} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ denotes the Lagrangian multiplier and μ is a positive parameter. The variables can be updated alternatively by fixing others in each iteration. Let $\mathcal{S}_t$, $\mathcal{Y}_t$, $\mathbf{W}_t$, $\mathcal{W}_t$ and μ_t be the value of $\mathcal{S}$, $\mathcal{Y}$, $\mathbf{W}$, $\mathcal{W}$ and μ in the t-th iteration, respectively.

Update $\mathcal{L}$: By fixing other variables, we first update $\mathcal{L}$ by minimizing $L(\mathcal{L}, \mathcal{S}_t, \mathbf{W}_t, \mathcal{W}_t, \mathcal{Y}_t, \mu_t)$, which is equivalent to

$$\mathcal{L}_{t+1} = \underset{\mathcal{L} \in \mathbb{R}^{d_1 \times d_2 \times d_3}}{\arg\min} \frac{1}{2} \|\mathcal{L} - (\mathcal{X} - \mathcal{S}_t - \mathcal{Y}_t/\mu_t)\|_F^2 + \frac{1}{\mu_t} \|\mathcal{L}\|_{\mathbf{W}_t,*}. \tag{26}$$

The problem above has a closed-form optimal solution. To derive the solution, we first introduce the following lemma about the solution to the WNNM (weighted nuclear norm minimization) problem of matrix data [26].

Lemma 1. *[26] Given a data matrix $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$ and a weight vector $\mathbf{w} = [w_1, \cdots, w_d]^T \in \mathbb{R}^d$ where $d = \min(d_1, d_2)$, let $\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}$ be the SVD of $\mathbf{M}$. Consider the WNNM problem*

$$\mathbf{prox}_{\|\cdot\|_{\mathbf{w},*}}(\mathbf{M}) := \underset{\mathbf{L} \in \mathbb{R}^{d_1 \times d_2}}{\arg\min} \frac{1}{2} \|\mathbf{L} - \mathbf{M}\|_F^2 + \|\mathbf{L}\|_{\mathbf{w},*}, \tag{27}$$

where $\mathbf{prox}_{\|\cdot\|_{\mathbf{w},*}}(\mathbf{M})$ denotes the proximal operator w.r.t. the weighted nuclear norm $\|\cdot\|_{\mathbf{w},*}$, $\|\mathbf{L}\|_{\mathbf{w},*} = \sum_{i=1}^{d} w_i \sigma_i(\mathbf{L})$ and $\sigma_i(\mathbf{L})$ is the i-th singular value of $\mathbf{L}$. If the weights satisfy $0 \le w_1 \le \cdots \le w_d$, the global solution to (27) is

$$\mathbf{L}^* = \mathbf{prox}_{\|\cdot\|_{\mathbf{w},*}}(\mathbf{M}) = \mathbf{U}\mathbf{P}_{\mathbf{w}}(\Sigma)\mathbf{V}^T,$$

where $\mathbf{P}_{\mathbf{w}}(\Sigma)$ is a diagonal matrix and $(\mathbf{P}_{\mathbf{w}}(\Sigma))_{ii} = (\Sigma_{ii} - w_i)_+$. Here $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ otherwise.

Now we generalize this lemma to the tensor data.

Theorem 1. *Given $\mathcal{M} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, $d = \min(d_1, d_2)$, and a weight matrix $\mathbf{W} \in \mathbb{R}^{d_3 \times d}$ where $d = \min(d_1, d_2)$, let $\mathcal{M} = \mathcal{U} * \mathcal{\Sigma} * \mathcal{V}^T$ be the T-SVD of $\mathcal{M}$. Consider the WTNN minimization problem*

$$\mathbf{prox}_{\|\cdot\|_{\mathbf{W},*}}(\mathcal{M}) = \underset{\mathcal{L} \in \mathbb{R}^{d_1 \times d_2 \times d_3}}{\arg\min} \frac{1}{2} \|\mathcal{L} - \mathcal{M}\|_F^2 + \|\mathcal{L}\|_{\mathbf{W},*}, \tag{28}$$

where $\mathbf{prox}_{\|\cdot\|_{\mathbf{W},*}}(\mathcal{M})$ denotes the proximal operator w.r.t. the WTNN $\|\cdot\|_{\mathbf{W},*}$. If the weights satisfy $0 \le W_{k1} \le W_{k2} \le \cdots \le W_{kd}$ for $k = 1, \cdots, d_3$, the global solution to the problem (28) is

$$\mathcal{L}^* = \mathbf{prox}_{\|\cdot\|_{\mathbf{W},*}}(\mathcal{M}) = \mathcal{U} * ifft\left(\mathcal{P}_{\mathbf{W}}(\bar{\Sigma}), [], 3\right) * \mathcal{V}^T,$$

where the k-th frontal slice $\bar{\Sigma}^{(k)}$ of $\bar{\mathbf{\Sigma}}$ is the singular value matrix of $\bar{\mathbf{M}}^{(k)}$ and $\mathcal{P}_{\mathbf{W}}(\bar{\Sigma})$ is a tensor such that its k-th frontal slice is $\mathbf{P}_{\mathbf{w}_k}\left(\bar{\Sigma}^{(k)}\right)$ for $k = 1, \cdots, d_3$ and $\mathbf{w}_k$ is the k-th row of the weight matrix $\mathbf{W}$.

By recalling the definition of WTNN in Definition 9, we have $\frac{1}{\mu_t}\|\mathcal{L}\|_{\mathbf{W}_t,*} = \|\mathcal{L}\|_{\mu_t^{-1}\mathbf{W}_t,*}$. Therefore, according to Theorem 1, the global optimum of the subproblem (26) with respect to (w.r.t.) $\mathcal{L}$ is

$$\mathcal{L}_{t+1} = \mathbf{prox}_{\|\cdot\|_{\mu_t^{-1}\mathbf{W}_t,*}}(\mathcal{M}_t) \tag{29}$$

where $\mathcal{M}_t = \mathcal{X} - \mathcal{S}_t - \mathcal{Y}_t/\mu_t$.

Update $\mathcal{S}$: By fixing other variables, we can update $\mathcal{S}$ by minimizing $L(\mathcal{L}_{t+1}, \mathcal{S}, \mathbf{W}_t, \mathcal{W}_t, \mathcal{Y}_t, \mu_t)$, which is equivalent to

$$\mathcal{S}_{t+1} = \underset{\mathcal{S} \in \mathbb{R}^{d_1 \times d_2 \times d_3}}{\arg\min} \frac{1}{2} \|\mathcal{S} - (\mathcal{X} - \mathcal{L}_{t+1} - \mathcal{Y}_t/\mu_t)\|_F^2 + \frac{\lambda}{\mu_t} \|\mathcal{S}\|_{\mathcal{W}_t,1}. \tag{30}$$

Algorithm 1 Double Auto-weighted Tensor Robust Principal Component Analysis (DATRPCA)

Input: Data tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, and the parameter λ.
Initialization: $\mathcal{L}_0 = 0$, $\mathcal{S}_0 = 0$, $\mathcal{Y}_0 = 0$, $\mathbf{W}_0 = \mathbf{1}_{d_3 \times d}$, $\mathcal{W}_0 = \mathbf{1}_{d_1 \times d_2 \times d_3}$, $\mu_0 = 10^{-2}$, $\rho = 1.1$, $\epsilon = 10^{-6}$ and $t = 0$.
while not converged **do**

1: Update $\mathcal{L}$ by Eq. (26).
2: Update $\mathcal{S}$ by Eq. (30).
3: Update the weights $\mathbf{W}$ and $\mathcal{W}$ by by Eq. (32).
4: Update the multipliers $\mathcal{Y}_{t+1} = \mathcal{Y}_t + \mu_t(\mathcal{L}_{t+1} + \mathcal{S}_{t+1} - \mathcal{X})$.
5: Update the parameter μ by $\mu_{t+1} = \rho\mu_t$.
6: Check the convergence condition:

$$\|\mathcal{L}_{t+1} - \mathcal{L}_t\|_\infty < \epsilon, \quad \|\mathcal{S}_{t+1} - \mathcal{S}_t\|_\infty < \epsilon,$$
$$\|\mathcal{X} - \mathcal{L}_{t+1} - \mathcal{S}_{t+1}\|_\infty < \epsilon.$$

end while
Output: $\mathcal{L} = \mathcal{L}_{t+1}$, $\mathcal{S} = \mathcal{S}_{t+1}$.

The problem above also has a closed-form solution. For ease of statement, we define the soft-thresholding operator $\mathbf{ST}$ such that $(\mathbf{ST}(\mathcal{T}, \mathcal{W}))_{ijk} = \text{sign}(\mathcal{T}_{ijk})(|\mathcal{T}_{ijk}| - \mathcal{W}_{ijk})_+$ where $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ otherwise. Then the closed-form solution to (30) is

$$\mathcal{S}_{t+1} = \mathbf{ST}\left(\mathcal{T}_t, \frac{\lambda}{\mu_t}\mathcal{W}_t\right), \tag{31}$$

where $\mathcal{T}_t = \mathcal{X} - \mathcal{L}_{t+1} - \mathcal{Y}_t/\mu_t$.

Update $\mathbf{W}$ and $\mathcal{W}$: According to Proposition 1, Eq. (17) and Eq. (21), the weights $\mathbf{W}$ and $\mathcal{W}$ can be updated in an adaptive way by

$$\mathbf{W}_{t+1} = F_L(\mathcal{L}_{t+1}), \quad \mathcal{W}_{t+1} = G_S(\mathcal{S}_{t+1}). \tag{32}$$

Update $\mathcal{Y}$ and μ: The Lagrangian multiplier tensor $\mathcal{Y}$ and the parameter μ are updated by

$$\mathcal{Y}_{t+1} = \mathcal{Y}_t + \mu_t(\mathcal{L}_{t+1} + \mathcal{S}_{t+1} - \mathcal{X}), \quad \mu_{t+1} = \rho\mu_t,$$

where $\rho = 1.1$. Algorithm 1 summarizes the optimization procedure for the DATRPCA model (24). Note that $\mathbf{1}_{d_3 \times d}$ and $\mathbf{1}_{d_1 \times d_2 \times d_3}$ denote a matrix of size $d_3 \times d$ and a tensor of size $d_1 \times d_2 \times d_3$ with all entries as ones, respectively.

C. Convergence Analysis

Before proving the convergence of Algorithm 1, we first some useful lemmas.

Lemma 2. *The sequences $\{\mathcal{Y}_t\}_{t=1}^{\infty}$, $\{\mathbf{W}_t\}_{t=1}^{\infty}$, and $\{\mathcal{W}_t\}_{t=1}^{\infty}$ generated by Algorithm 1 are bounded.*

Lemma 3. *The sequences $\{\mathcal{L}_t\}_{t=1}^{\infty}$ and $\{\mathcal{S}_t\}_{t=1}^{\infty}$ generated by Algorithm 1 are bounded.*

With the results above, now we turn to proving the convergence of the proposed algorithm.

Theorem 2. *The sequences $\{\mathcal{L}_t\}_{t=1}^{\infty}$, $\{\mathcal{S}_t\}_{t=1}^{\infty}$ generated by Algorithm 1 satisfy*

(1) $\lim_{t\to\infty} \|\mathcal{L}_t + \mathcal{S}_t - \mathcal{X}\|_F = 0$.
(2) $\lim_{t\to\infty} \|\mathcal{L}_{t+1} - \mathcal{L}_t\|_F = 0$.
(3) $\lim_{t\to\infty} \|\mathcal{S}_{t+1} - \mathcal{S}_t\|_F = 0$.

This article has been accepted for publication in IEEE Transactions on Image Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIP.2023.3310331

Table II: Correct recovery for random problems of varying sizes. Best results are marked bold.

$r = \text{rank}_t(\mathcal{L}_0) = 0.1n$, $m = \|\mathcal{S}_0\| = 0.1n^3$						
n	Algorithm	r	m	$\text{rank}_t(\hat{\mathcal{L}})$	$\frac{\|\hat{\mathcal{L}}-\mathcal{L}_0\|_F}{\|\mathcal{L}_0\|_F}$	$\frac{\|\hat{\mathcal{S}}-\mathcal{S}_0\|_F}{\|\mathcal{S}_0\|_F}$
100	RPCA	10	1e5	100	7.12e-01	7.11e-03
	TRPCA	10	1e5	10	2.27e-07	9.75e-10
	ETRPCA	10	1e5	10	1.40e-07	5.70e-10
	DATRPCA	10	1e5	10	**7.95e-08**	**4.97e-10**
200	RPCA	20	8e5	200	7.11e-01	2.52e-03
	TRPCA	20	8e5	20	7.52e-07	9.84e-10
	ETRPCA	20	8e5	20	4.66e-07	5.13e-10
	DATRPCA	20	8e5	20	**4.94e-08**	**8.95e-11**

$r = \text{rank}_t(\mathcal{L}_0) = 0.1n$, $m = \|\mathcal{S}_0\| = 0.2n^3$						
n	Algorithm	r	m	$\text{rank}_t(\hat{\mathcal{L}})$	$\frac{\|\hat{\mathcal{L}}-\mathcal{L}_0\|_F}{\|\mathcal{L}_0\|_F}$	$\frac{\|\hat{\mathcal{S}}-\mathcal{S}_0\|_F}{\|\mathcal{S}_0\|_F}$
100	RPCA	10	1e5	100	7.95e-01	5.60e-03
	TRPCA	10	1e5	10	5.45e-07	2.95e-09
	ETRPCA	10	1e5	10	3.16e-07	1.69e-09
	DATRPCA	10	1e5	10	**1.80e-07**	**9.32e-10**
200	RPCA	20	8e5	200	7.98e-01	2.83e-03
	TRPCA	20	8e5	20	6.38e-07	1.72e-09
	ETRPCA	20	8e5	20	5.77e-07	1.54e-09
	DATRPCA	20	8e5	20	**3.53e-07**	**9.17e-10**

IV. Experiments

In this section, we evaluate the performance of DATRPCA on both synthetic and real data. For the experiments on synthetic data, we aim to recover the low-tubal-rank tensor in the presence of sparse random noise. As for the experiments on real data, we apply DATRPCA for color image recovery and background modeling with video data.

A. Exact Recovery from Varying Fractions of Error

In this part, we aim to assess the performance of DATRPCA against varying fractions of noise.

Data generation. We first generate a low-tubal-rank tensor $\mathcal{L}_0 = \mathcal{P} * \mathcal{Q} \in \mathbb{R}^{n\times n\times n}$ where the entries of $\mathcal{P} \in \mathbb{R}^{n\times r\times n}$ and $\mathcal{Q} \in \mathbb{R}^{r\times n\times n}$ are sampled from the normal distribution $\mathcal{N}(0, \frac{1}{n})$ independently. Then we generate a sparse tensor $\mathcal{S}_0 \in \mathbb{R}^{n\times n\times n}$ by randomly and uniformly selecting m entries of $\mathcal{S}_0$ and set their values as independent Bernoulli ± 1. The rest entries of $\mathcal{S}_0$ are set as 0s.

Results. Table II presents the recovery results of comparison methods with varying fraction of error and different data dimension n. It can be seen from the results that RPCA does not estimate the tubal rank correctly in most cases while the other three TRPCA methods do. As the percent of noisy entries increases, the relative recovery errors $\frac{\|\hat{\mathcal{L}}-\mathcal{L}_0\|_F}{\|\mathcal{L}_0\|_F}$ and $\frac{\|\hat{\mathcal{S}}-\mathcal{S}_0\|_F}{\|\mathcal{S}_0\|_F}$ of each method increase as expected. However, DATRPCA outperforms other competing algorithms with varying fraction of noise, especially with more noisy entries.

To have a better understanding of the proposed method, we show the learned weight matrix $\mathbf{W}$ and tensor $\mathcal{W}$ for the low-rank tensor $\mathcal{L}$ and sparse tensor $\mathcal{S}$, respectively. In this experiment, we set $n = 100$, $r = 0.1n = 10$ and $m = 0.1n^3 = 1e5$. Recall from the definition of WTNN in Definition 9 that for the k-th frontal slice $\bar{\mathcal{L}}^{(k)}$ of $\bar{\mathcal{L}}$, the k-th row $\mathbf{W}(k,:)$ of $\mathbf{W}$ assigns the weights to the singular values of $\bar{\mathcal{L}}^{(k)}$. Ideally, for larger singular values of $\bar{\mathcal{L}}^{(k)}$, we expect to assign smaller weights and shrunk them less.

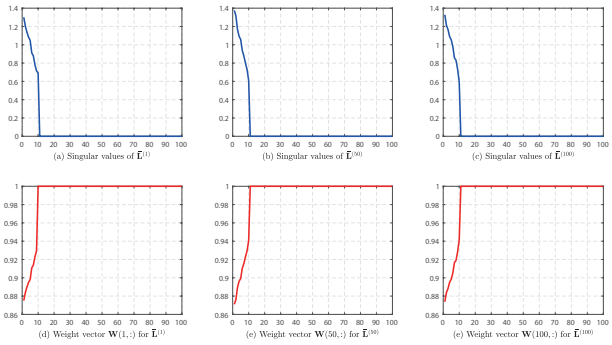


Figure 3: (a)-(c) Singular values of the 1-st, 50-th and 100-th front slice $\bar{\mathbf{L}}_0^{(1)}$, $\bar{\mathbf{L}}_0^{(50)}$ and $\bar{\mathbf{L}}_0^{(100)}$ of $\bar{\mathcal{L}}_0$; (d)-(f) the 1-st, 50-th and 100-th row of the weight matrix $\mathbf{W}$ learned by DATRPCA.

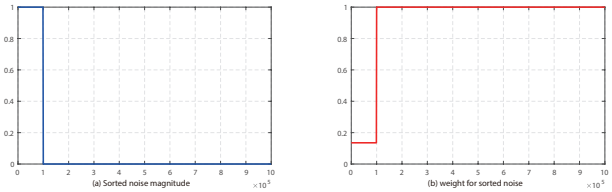


Figure 4: (a) The absolute values of vectorized version $\mathbf{s}_0$ of the sparse Bernoulli noise tensor $\mathcal{S}_0$ in descending order; (b) the vectorized version $\mathbf{w}$ of the weight tensor $\mathcal{W}$ learned by DATRPCA using the same order as $\mathbf{s}_0$.

Fig. 3 shows the curves of singular values of the 1-st, 50-th and 100-th front slice $\bar{\mathbf{L}}_0^{(1)}$, $\bar{\mathbf{L}}_0^{(50)}$ and $\bar{\mathbf{L}}_0^{(100)}$ of $\bar{\mathcal{L}}_0$ and the corresponding weight vectors $\mathbf{W}(1,:)$, $\mathbf{W}(50,:)$, $\mathbf{W}(100,:)$ learned by DATRPCA, respectively. Note from the results that DATRPCA can adaptively assign smaller weights to larger singular values of the three frontal slices of $\bar{\mathcal{L}}$. In fact, the same phenomenon can be observed for other frontal slices, which are not shown here due to space limitation. As for the sparse noise tensor $\mathcal{S}_0$, we also expect that DATRPCA can assign smaller weights to those entries of $\mathcal{S}_0$ with large absolute values and shrink them less. To show the results intuitively, we first construct a vector $\mathbf{s}_0 \in \mathbb{R}^{n^3}$ consisting of the absolute values of $\mathcal{S}_0$ in descending order. Then we build a weight vector $\mathbf{w} \in \mathbb{R}^{n^3}$ containing entries of the weight tensor $\mathcal{W}$ learned by DATRPCA in the same order as $\mathbf{s}_0$. Such arrangement is to ensure that $|\mathcal{S}_0(i,j,k)|$ and $\mathcal{W}(i,j,k)$ are in the same location of $\mathbf{s}_0$ and $\mathbf{w}$, respectively. Fig. 4 shows the curves of $\mathbf{s}_0$ and $\mathbf{w}$. Note that DATRPCA can also adaptively assign smaller weights to significant entries of $\mathcal{S}_0$ with larger absolute values. Actually, we have also repeated the experiments with different sparsity of $\mathcal{S}_0$. The results are shown in supplementary material.

B. Phase Transition in Tubal Rank and Tubal Sparsity

In the previous part, we have assessed the recovery performance of the proposed method with the fixed relative tubal rank $\rho_r := \frac{r}{n}$ of $\mathcal{L}_0$ as 0.1, and the fixed relative sparsity $\rho_s := \frac{m}{n^3}$ of $\mathcal{S}_0$ as 0.1 or 0.2. To further validate the efficacy

This article has been accepted for publication in IEEE Transactions on Image Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIP.2023.3310331

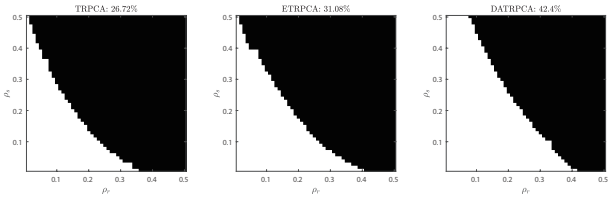


Figure 5: Correct recovery for varying rank ($r = \rho_r n$) and sparsity ($m = \rho_s n^3$). The number after each method is the percent of the white area, which means the successful recovery rate.

of the proposed method, in this part we compare it with other competing algorithms with varying relative tubal rank ρ_r and relative sparsity ρ_s.

Data generation. We first generate a tensor $\mathcal{L}_0 = \mathcal{P} * \mathcal{Q} \in \mathbb{R}^{n\times n\times n_3}$ of tubal rank r, where the entries of $\mathcal{P} \in \mathbb{R}^{n\times r\times n_3}$ and $\mathcal{Q} \in \mathbb{R}^{r\times n\times n_3}$ are sampled from the normal distribution $\mathcal{N}(0, \frac{1}{n})$ independently. In this experiment, we set $n = 100$ and $n_3 = 50$ [8]. Then we generate the sparse error tensor $\mathcal{S}_0$ by setting its entries to be 0 with probability $1 - \rho_s$ and ± 1 each with probability $\rho_s/2$. The recovery is deemed correct and successful if the recovered tensor $\hat{\mathcal{L}}$ satisfies $\frac{\|\hat{\mathcal{L}}-\mathcal{L}_0\|_F}{\|\mathcal{L}_0\|_F} \leq 10^{-3}$. In the experiment, we set $\rho_r,\ \rho_s \in [0.01 : 0.01 : 0.5]$.

Results. Fig. 5 shows the recovery results for each pair (ρ_r, ρ_s). Here black and white correspond to correct and fail recovery, respectively. From the results, we make the following conclusions.

- Firstly, the three TRPCA methods can yield correct recovery results when $\mathcal{L}_0$ has relatively low tubal rank and $\mathcal{S}_0$ is relatively sparse. This is as expected because the recovery problem is relatively easier in this case.
- Secondly, the correct recovery region of DATRPCA is larger than other competing algorithms. As shown in Fig. 5, DATRPCA can tolerate large tubal rank and large noise rate. The results demonstrate its superiority over other comparison methods for low-rank tensor recovery.

C. Application to Color Image Recovery

In this section, we apply DATRPCA to color image recovery in the presence of random noise. This is motivated by the fact that color images can be modeled as 3-order low-rank tensors with row, column and color modes [8].

Datasets. Three benchmark color image databases are used to evaluate the performance of the proposed method. They are the Berkeley Segmentation Dataset (BSD) [31], the Kodak dataset [32], and the ZheJiang University (ZJU) dataset used in [33], respectively. These databases have been widely used in the literature of computer vision [34]. Fig. 6(a) shows some sample images from the BSD dataset. In the experiments, we randomly set a fraction of pixels in each color image to random values in $[0, 255]$. To make the problems more challenging, we make all the 3 channels of the images are corrupted at the same positions and the positions of the corrupted pixels are agnostic to the recovery algorithms. See Fig. 6(b) for several noisy sample images on the BSD dataset.

Table III: Comparison of the PSNR values on the 6 images in Fig. 6. Best results are marked bold.

Image	1	2	3	4	5	6
RPCA	36.94	24.51	27.48	23.41	23.95	26.32
SNN	39.18	26.43	29.23	25.23	26.29	27.44
TRPCA	40.50	28.30	31.08	25.45	27.34	27.58
ETRPCA	41.15	28.54	31.75	26.26	27.76	28.27
TRPCA-Lp	41.24	28.83	32.16	26.77	28.37	28.75
DATRPCA	**43.69**	**30.20**	**34.46**	**29.34**	**30.12**	**31.42**

Table IV: Comparison of the SSIM values on the 6 images in Fig. 6. Best results are marked bold.

Image	1	2	3	4	5	6
RPCA	0.9887	0.8465	0.9353	0.9510	0.9322	0.9730
SNN	0.9933	0.8984	0.9529	0.9651	0.9592	0.9792
TRPCA	0.9949	0.9225	0.9642	0.9597	0.9635	0.9747
ETRPCA	0.9954	0.9256	0.9679	0.9658	0.9666	0.9787
TRPCA-Lp	0.9954	0.9268	0.9690	0.9636	0.9680	0.9780
DATRPCA	**0.9969**	**0.9460**	**0.9800**	**0.9782**	**0.9784**	**0.9870**

Comparison methods. We compare the proposed DATRPCA method with RPCA [3], SNN [13], TRPCA [8], ETRPCA [16], and TRPCA-Lp [9]. Since RPCA is originally developed for matrix data, it is applied on each color channel independently and the results are combined to yield the final recovered image. The parameter is set to be $\lambda = 1/\sqrt{\max(d_1, d_2)}$ as suggested by the corresponding authors [3]. For SNN, the parameter is set as $[15, 15, 1.5]$ to perform well in most cases [8]. For TRPCA, ETRPCA and TRPCA-Lp, the parameter is set as $\lambda = 1/\sqrt{3\max(d_1, d_2)}$ according to the authors [8], [9], [16]. For fair comparison, we also set $\lambda = 1/\sqrt{3\max(d_1, d_2)}$ for DATRPCA. To evaluate the recovery performance, we adopt two widely used image quality indexes, i.e., Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity (SSIM) [35].

Results. Fig. 6 shows several sample images and the recovered results. Their PSNR and SSIM values are listed in Tables III and IV, respectively. Fig. 7 presents the PSNR and SSIM values of the recovered images by different comparison methods on 50 images of the BSD dataset. To make the results more convincing, we have also applied competing algorithms on the whole BSD dataset. The orginal BSD dataset contains three subsets: BSD_train, BSD_val and BSD_test. The number of images in these subsets are 200, 100 and 200, respectively. Tables V and VI compare the recovery performance of competing algorithms in terms of average PSNR and SSIM values on the whole BSD dataset with 10% and 20% corrupted ratios, respectively. Due to space limitation, the recovery results on the Kodak dataset are present in the supplementary material.

Based on the results, we can draw the following conclusions.

- Firstly, the three TRPCA methods generally achieve much better recovery performance than the matrix RPCA in most cases. This is due to the fact that the matrix RPCA processes each channel of a color image independently and separately. While TRPCA methods process the color image in a holistic manner and exploit the correlation

This article has been accepted for publication in IEEE Transactions on Image Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIP.2023.3310331

Figure 6: Recovery performance comparison on 6 sample images of the BSD dataset. From top row to bottom row: (a) Original clean image; (b) observed noisy image; (c)-(h) recovered images by RPCA, SNN TRPCA, ETRPCA, TRPCA-Lp and DATRPCA, respectively. The demarcated area is enlarged in the top right corner for better visualization. The figure is better seen by zooming on a computer screen.

information among multiple channels to further improve the recovery performance.

- Secondly, DATRPCA outperforms other competing algorithms in terms of both visual quality and quantitative evaluation indexes in most cases. For instance, for the second column in Fig. 6, DATRPCA reconstructs more image details and color information of the bird head compared with other methods. With 10% corruption ratio on the BSD dataset, DATRPCA achieves at least 1.5dB improvement in terms of average PSNR over other methods as shown in Table V.
- Thirdly, as shown in Table VI, with 20% corruption ratio, DATRPCA is the only method which achieves the average PSNR value over 28dB and the average SSIM value higher than 0.9 among all the competing algorithms on the BSD dataset. This further validates the robustness of DATRPCA for color image recovery against heavy random pixel corruption.

Ablation Study. The proposed DATRPCA method is a joint learning method which takes advantage of the prior information of the low-rank tensor $\mathcal{L}$ and the sparse tensor $\mathcal{S}$ simultaneously. To this end, DATRPCA harness the newly designed regularization functions $R_L(\mathcal{L})$ and $R_S(\mathcal{S})$ in lieu of $\|\mathcal{L}\|_*$ and $\|\mathcal{S}\|_1$ in TRPCA, respectively. A natural question aries: whether $R_L(\mathcal{L})$ or $R_S(\mathcal{S})$ alone is effective for improving the recovery performance? To answer this question, we conduct ablation studies to evaluate performances of two model variants: (a) DATRPCA-L that uses $R_L(\mathcal{L})$ and $\|\mathcal{S}\|_1$; (b) DATRPCA-S that uses $\|\mathcal{L}\|_*$ and $R_S(\mathcal{S})$.

Table VII reports the average PNSR and SSIM values of four methods on the three datasets with varying corruption ratio. From the results, we make the following conclusions:

This article has been accepted for publication in IEEE Transactions on Image Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIP.2023.3310331

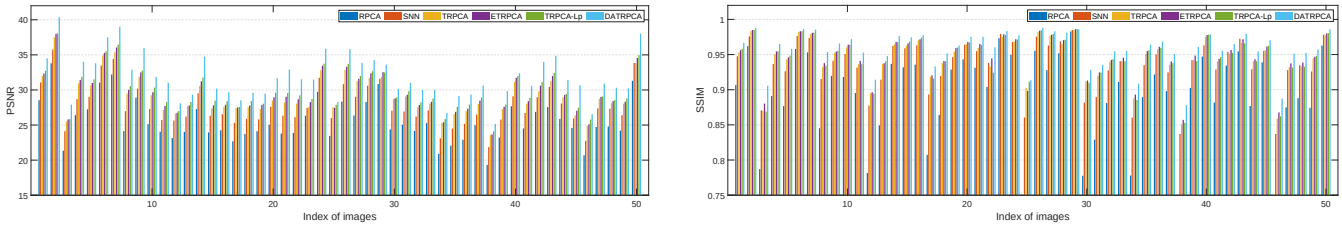


Figure 7: Comparison of the PSNR values (left) and SSIM values (right) obtained by different methods on 50 images of the BSD dataset with 10% corruption ratios. Better viewed by zooming on a computer screen.

Table V: Average PSNR and SSIM values on the images of the BSD dataset with 10% corruption ratio. Best results are marked bold. BSD_train, BSD_val and BSD_test are three subsets of the whole BSD dataset. The number in each parentheses means the number of images in each subset of the BSD dataset.

Methods	BSD_train (200)		BSD_val (100)		BSD_test (200)		All (500)	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
RPCA	25.53	0.8947	25.01	0.8765	25.10	0.8929	25.25	0.8903
SNN	27.63	0.9354	27.19	0.9251	27.19	0.9335	27.37	0.9326
TRPCA	29.12	0.9454	28.73	0.9349	28.63	0.9418	28.85	0.9419
ETRPCA	29.50	0.9491	29.07	0.9389	29.03	0.9461	29.23	0.9459
TRPCA-Lp	29.90	0.9473	30.01	0.9422	29.45	0.9436	29.74	0.9448
DATRPCA	**31.64**	**0.9626**	**31.08**	**0.9538**	**31.21**	**0.9607**	**31.36**	**0.9601**

Table VI: Average PSNR and SSIM values on the images of the BSD dataset with 20% corruption ratio.

Methods	BSD_train (200)		BSD_val (100)		BSD_test (200)		All (500)	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
RPCA	24.52	0.8625	24.02	0.8383	24.07	0.8581	24.24	0.8559
SNN	26.14	0.8977	25.67	0.8787	25.67	0.8927	25.85	0.8919
TRPCA	26.63	0.8823	26.08	0.8557	26.10	0.8739	26.31	0.8736
ETRPCA	27.00	0.8914	26.44	0.8667	26.49	0.8842	26.69	0.8836
TRPCA-Lp	26.66	0.8681	26.62	0.8541	26.14	0.8591	26.44	0.8617
DATRPCA	**28.81**	**0.9339**	**28.24**	**0.9192**	**28.35**	**0.9307**	**28.51**	**0.9297**

Table VII: Ablation Study. SSIM values of DATRPCA on different datasets with varying corruption ratios.

Methods	Regularization		BSD		Kodak		ZJU	
	$R_L(\mathcal{L})$	$R_S(\mathcal{S})$	10%	20%	10%	20%	10%	20%
TRPCA			0.9419	0.8736	0.9466	0.8890	0.9735	0.9353
DATRPCA-L	✓		0.9504	0.8773	0.9550	0.8925	0.9790	0.9368
DATRPCA-S		✓	0.9446	0.9054	0.9476	0.9144	0.9771	0.9560
DATRPCA	✓	✓	**0.9601**	**0.9297**	**0.9627**	**0.9361**	**0.9821**	**0.9672**

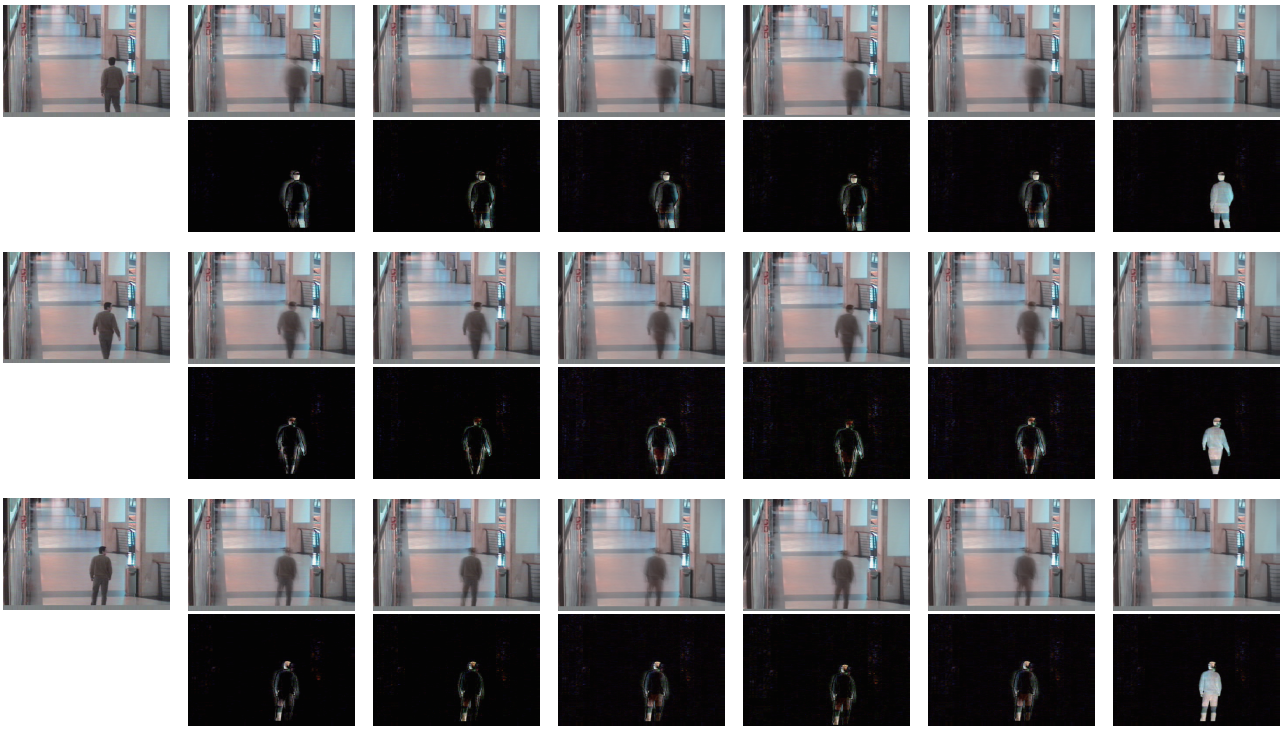
- Firstly, both DATRPCA-L and DATRPCA-S outperform TRPCA in terms of average PSNR and SSIM values on all the three datasets. This indicates that both $R_L(\mathcal{L})$ and $R_S(\mathcal{S})$ alone are effective in improving the recovery performance of TRPCA.
- Secondly, DATRPCA has better performance than DATRPCA-L and DATRPCA-S in various cases. This means that it is valuable by integrating $R_L(\mathcal{L})$ and $R_S(\mathcal{S})$ into a joint framework.
- Thirdly, DATRPCA-L outperforms DATRPCA-S with 10% corruption while the results are opposite with 20% corruption. This indicates that DATRPCA-S has better robustness than DATRPCA-L against gross corruption. This is as expected because DATRPCA-S adaptively assigns smaller weights on large entries of $\mathcal{S}$, giving rise to its strong robustness against heavy noises.

D. Application to Background Modeling

This part devotes to evaluating the effectiveness of DATRPCA for background modeling, which attempts to split the foreground objects from the background using video data. Since the frames of the background are highly correlated in general, it can be modeled as a low-rank tensor [8]. Meanwhile, since the foreground objects generally occupy only a fraction of pixels, they can be modeled as sparse error tensor. Thus, background modeling can also be cast as a TRPCA problem.

Datasets. In this experiment, we consider the CAVIAR1 video from the Scene Background Initialization (SBI) database [36]. The CAVIAR1 video has $N = 610$ color frames and each frame has the size $H \times W \times 3$ where the image height $H = 256$ and the image width $W = 384$. For efficiency, we adopt the

This article has been accepted for publication in IEEE Transactions on Image Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIP.2023.3310331



(a) Original (b) RPCA (c) SNN (d) TRPCA (e) ETRPCA (f) TRPCA-Lp (g) DATRPCA

Figure 8: Background modeling results of video sequences. (a) Original frames; (b)-(g) low rank and sparse components learned by RPCA, SNN, TRPCA, ETRPCA, TRPCA-Lp and DATRPCA, respectively. The figure is better seen by zooming on a computer screen.

first $k = 100$ frames. To apply RPCA, the video is reshaped into a $(3HW) \times k$ matrix. For the other tensor RPCA methods, each video is reshaped into a $(HW) \times 3 \times k$ tensor [8].

Parameter settings. For RPCA, the parameter is set to be $\lambda = 1/\sqrt{\max(3HW, k)}$ as per the authors [3]. For SNN, The parameter λ of SNN is set to $\lambda = [10; 0.1; 1] \times 20$ [8]. For TRPCA, ETRPCA, TRPCA-Lp and DATRPCA, the parameter is set as $\lambda = 1/\sqrt{k \max(HW, 3)}$ according to the authors [8], [16].

Results. Fig. 8 shows the three original frames and background modeling results of different methods. The frame number is 60, 70 and 80, respectively. Note from Fig. 8 that DATRPCA can yield clean background and coherent foreground masks simultaneously in most cases. In contrast, the background images obtained by other competing methods are blurry and have severe ghosting effects. The reason lies in the fact DATRPCA can adaptively assign lighter penalization on significant singular values and sparse entries simultaneously. This gives rise to the capability of DATRPCA in preserving prominent information of the background images.

V. Conclusion

This paper addresses the Tensor RPCA problem, which aims to recover the low rank tensor $\mathcal{L}$ and the sparse tensor $\mathcal{S}$ from their sum. To preserve the significant information of $\mathcal{L}$ and $\mathcal{S}$, we propose a Double Auto-weighted tensor RPCA (DATRPCA) method. A key advantage of DATRPCA is its ability to learn the weights for the singular values of $\mathcal{L}$ and the entries of $\mathcal{S}$ simultaneously from the data in an *adaptive* manner. The results show that DATRPCA can *adaptively and automatically* assigns smaller weights and applies lighter penalization to significant singular values of $\mathcal{L}$ and important entries of $\mathcal{S}$. We have also devised an efficient optimization algorithm for DATRPCA and established the convergence analysis of the proposed algorithm. Experiments on real-world data validate the efficacy of our method and its superiority over state of the art.

References

[1] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educat. Psych.*, vol. 24, pp. 417–441, 1933.
[2] P. J. Huber, *Robust Statistics*. New York: Wiley, 1981.
[3] E. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis," *J. ACM*, vol. 58, no. 3, p. 11, 2011.
[4] H. Xu, C. Caramanis, and S. Sanghavi, "Robust PCA via outlier pursuit," in *Proc. NIPS*, 2010, p. 23.
[5] J. Feng, H. Xu, and S. Yan, "Online robust PCA via stochastic optimization," in *Proc. NIPS*, 2013, p. 26.
[6] Q. Zhou, D. Meng, Z. Xu, W. Zuo, and L. Zhang, "Robust principal component analysis with complex noise," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 55–63.
[7] T. Oh, Y. Tai, J. Bazin, H. Kim, and I. Kweon, "Partial sum minimization of singular values in robust pca: Algorithm and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 744–758, 2016.
[8] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis with a new tensor nuclear norm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 925–938, Apr. 2020.
[9] H. Kong, X. Xie, and Z. Lin, "t-schatten-p norm for low-rank tensor recovery," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 6, pp. 1405–1419, 2018.

[10] Y. Chang, L. Yan, X. Zhao, H. Fang, Z. Zhang, and S. Zhong, "Weighted low-rank tensor recovery for hyperspectral image restoration," *IEEE Transactions on Cybernetics*, vol. 50, no. 11, pp. 4558–4572, 2020.

[11] H. Qiu, Y. Wang, S. Tang, D. Meng, and Q. Yao, "Fast and provable nonconvex tensor RPCA," in *International Conference on Machine Learning*, 2022, pp. 18 211–18 249.

[12] H. Kong, C. Lu, and Z. Lin, "Tensor Q-rank: New data dependent definition of tensor rank," *Machine Learning*, vol. 110, no. 7, pp. 1867–1900, 2021.

[13] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 208–220, 2013.

[14] C. Lu, "Transforms based tensor robust PCA: Corrupted low-rank tensors recovery via convex optimization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 1145–1152.

[15] F. Zhang, J. Wang, W. Wang, and C. Xu, "Low-tubal-rank plus sparse tensor recovery with prior subspace information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3492–3507, 2021.

[16] Q. Gao, P. Zhang, W. Xia, D. Xie, X. Gao, and D. Tao, "Enhanced tensor rpca and its application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 6, pp. 2133–2140, 2021.

[17] T. Jiang, M. Ng, X. Zhao, and T. Huang, "Framelet representation of tensor nuclear norm for third-order tensor completion," *IEEE Trans. Image Process.*, vol. 29, pp. 7233–7244, 2020.

[18] T. G. Kolda and B. W. Bader, "Tensor completion for estimating missing values in visual data," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

[19] L. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[20] M. Kilmer and C. Martin, "Factorization strategies for third-order tensors," *Linear Algebra Appl.*, vol. 435, no. 3, pp. 641–658, 2011.

[21] M. Kilmer, K. Braman, N. Hao, and R. C. Hoover, "Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 148–172, 2013.

[22] W. Xu, X. Zhao, T. Ji, and et al., "Laplace function based nonconvex surrogate for low-rank tensor completion," *Signal Process. Image Commun.*, vol. 73, pp. 62–69, 2019.

[23] D. Goldfarb and Z. Qin, "Robust low-rank tensor recovery: Models and algorithms," *SIAM J. Matrix Anal. Appl.*, vol. 35, no. 1, pp. 225–253, 2014.

[24] C. Mu, B. Huang, J. Wright, and D. Goldfarb, "Square deal: Lower bounds and improved relaxations for tensor recovery," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 73–81.

[25] Y. Zhou and Y. Cheung, "Bayesian low-tubal-rank robust tensor factorization with multi-rank determination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 62–76, 2021.

[26] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, "Weighted nuclear norm minimization and its applications to low level vision," *Int. J. Comput. Vis.*, vol. 121, no. 2, pp. 183–208, 2017.

[27] T. Jiang, T. Huang, X. Zhao, and L. Deng, "Multi-dimensional imaging data recovery via minimizing the partial sum of tubal nuclear norm," *J. Comput. Appl. Math.,*, vol. 372, no. 112680, 2020.

[28] E. Candès, M. Wakin, and S. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *J. Fourier Anal. Appl.*, vol. 14, no. 5, pp. 877–905, 2008.

[29] L. Chen, X. Jiang, X. Liu, and Z. Zhou, "Robust low-rank tensor recovery via nonconvex singular value minimization," *IEEE Trans. Image Process.*, vol. 29, pp. 9044–9059, 2020.

[30] R. He, W. Zheng, T. Tan, and Z. Sun, "Half-quadratic-based iterative minimization for robust sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 261–275, Feb. 2014.

[31] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. Int. Conf. Comput. Vis.*, 2001, pp. 416–423.

[32] E. Kodak, "Kodak lossless true color image suite (photocd pcd0992)," *URL http://r0k. us/graphics/kodak*, vol. 6, 1993.

[33] Y. Hu, D. Zhang, J. Ye, X. Li, and X. He, "Fast and accurate matrix completion via truncated nuclear norm regularization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 9, pp. 2117–2130, 2013.

[34] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," *Arxiv preprint arXiv:1803.04189*, 2018.

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[36] L. Maddalena and A. Petrosino, "Towards benchmarking scene background initialization," in *Proc. Int. Conf. Image Anal. Process.*, 2014, pp. 55–63.