5-2021

Tensor low-rank representation for data recovery and clustering

Pan ZHOU
Singapore Management University, panzhou@smu.edu.sg

Canyi LU

Jiashi FENG

Zhouchen LIN

Shuicheng YAN

Citation

ZHOU, Pan; LU, Canyi; FENG, Jiashi; LIN, Zhouchen; and YAN, Shuicheng. Tensor low-rank representation for data recovery and clustering. (2021). *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 43, (5), 1718-1732.
Available at: https://ink.library.smu.edu.sg/sis_research/8997

Tensor Low-Rank Representation for Data Recovery and Clustering

Pan Zhou, Canyi Lu, *Member, IEEE*, Jiashi Feng, Zhouchen Lin, *Fellow, IEEE*, and Shuicheng Yan, *Fellow, IEEE*

Abstract—Multi-way or tensor data analysis has attracted increasing attention recently, with many important applications in practice. This article develops a tensor low-rank representation (TLRR) method, which is the first approach that can exactly recover the clean data of intrinsic low-rank structure and accurately cluster them as well, with provable performance guarantees. In particular, for tensor data with arbitrary sparse corruptions, TLRR can exactly recover the clean data under mild conditions; meanwhile TLRR can exactly verify their true origin tensor subspaces and hence cluster them accurately. TLRR objective function can be optimized via efficient convex programing with convergence guarantees. Besides, we provide two simple yet effective dictionary construction methods, the simple TLRR (S-TLRR) and robust TLRR (R-TLRR), to handle slightly and severely corrupted data respectively. Experimental results on two computer vision data analysis tasks, image/video recovery and face clustering, clearly demonstrate the superior performance, efficiency and robustness of our developed method over state-of-the-arts including the popular LRR and SSC methods.

Index Terms—Tensor low-rank representation, low-rank tensor recovery, tensor data clustering

1 Introduction

This paper studies the problem of recovering 3-way tensor data from noisy observations with corruption and clustering them as well. Formally, suppose one is provided with a 3-way noisy tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ which is composed of a clean low-rank tensor $\mathcal{L}_0$ and an additional sparse noise component $\mathcal{E}_0$:

$$\mathcal{X} = \mathcal{L}_0 + \mathcal{E}_0. \tag{1}$$

W.l.o.g., we assume samples are distributed along the second dimension (or mode) of $\mathcal{X}$, i.e., $\mathcal{X}(:,t,:)$ denotes the tth sample as shown in Fig. 1, and the samples $\mathcal{L}_0(:,t,:)$ are drawn from one of k independent linear tensor subspaces (see Section 5.2). We aim to exactly recover the low-rank tensor $\mathcal{L}_0$ from $\mathcal{X}$ and cluster n_2 samples $\mathcal{L}_0(:,t,:)$ into k clusters according to their affiliation with the k tensor subspaces. This problem is important for many applications, including image/video denoising [1], [3], data clustering [4], [5], saliency detection [6], and visual tracking [7].

- *P. Zhou and J. Feng are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077. E-mail: pzhou@u.nus.edu, elefjia@nus.edu.sg.*
- *C. Lu is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA. E-mail: canyilu@gmail.com.*
- *Z. Lin is with the Key Lab. of Machine Perception (MoE), School of EECS, Peking University, Haidian District, Beijing 100871, China. E-mail: zlin@pku.edu.cn.*
- *S. Yan is with YITU Tech, Shanghai 200051, China. E-mail: eleyans@nus.edu.sg.*

Manuscript received 17 Dec. 2018; revised 11 Sept. 2019; accepted 10 Nov. 2019. Date of publication 21 Nov. 2019; date of current version 1 Apr. 2021. (Corresponding author: Zhouchen Lin.) Recommended for acceptance by P. Favaro. Digital Object Identifier no. 10.1109/TPAMI.2019.2954874

This problem is well studied in the matrix domain. For instance, the matrix-based low-rank representation (LRR) algorithm [3], [4] clusters vector-valued samples into corresponding subspaces by seeking low-rank linear representations of all the samples w.r.t. a given dictionary. The representation coefficients encode subspace membership of the samples on which standard clustering methods can be applied to obtain sample clusters. But LRR and its variants [8], [9] are limited to 2-way data. Usually realistic data are in multi-way, e.g., videos, image collections and finance data. Naively flattening them into a big matrix and applying LRR-alike algorithms would lead to performance drop because the data intrinsic multi-way structure is destroyed [10], [11], [12], [13]. Recently, some tensor analysis based works, e.g., [13], [14], are proposed to recover the clean data $\mathcal{L}_0$ from $\mathcal{X}$. However, these methods do not learn the relations among samples and cannot cluster them either. Later, Fu et al. [15] integrated Tuker decomposition [16] with sparse coding [17] to simultaneously learn the low-rank spatial relations among samples and the linear relations of samples in the feature space for tensor data clustering. But their proposed TLRRSC method still requires reshaping 2-way data into vectors and thus suffers degraded performance [10], [11], [13]. Zhang et al. [18] proposed a low-rank tensor constrained multi-view subspace clustering method (LT-MSC), which performs LRR on the data matrix in each view with a unified low-rank constraint on the tensor consisting of all representation matrices. However, LT-MSC requires several kinds of features to construct the multi-view data. Besides, both TLRRSC and LT-MSC have no theoretical guarantee for data recovery and clustering. Though deep learning has achieved great success in many applications [19], [20], it cannot well solve the recovery and clustering problems of the high-dimensional but small-scale tensor

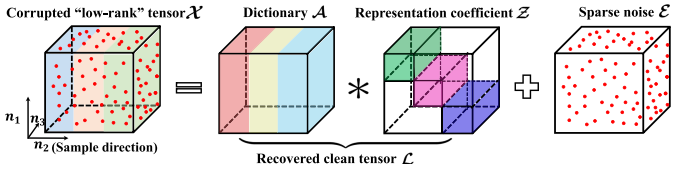


Fig. 1. Illustration of the proposed TLRR method for tensor data recovery and clustering. By exploiting intrinsic low-rank structure of the input tensor data $\mathcal{X}$, TLRR can effectively recover the underlying low-rank tensor $\mathcal{L}$ in presence of sparse noise $\mathcal{E}$, and cluster the samples in $\mathcal{X}$ (encoded by $\mathcal{Z}$ under dictionary $\mathcal{A}$).

data considered in this work. This is because it is hard to well train a network under this setting due to insufficient training samples. For example, one cannot train a network on a single corrupted image to denoise it.

In this work, we develop the first algorithmic solution to the problem (1) with both theoretical performance guarantees and practical efficacy. The developed Tensor Low-Rank Representation (TLRR) method can recover the clean tensor of low-rank structure and infer the clusters of samples. Concretely, based on the recently developed tensor nuclear norm [13], we propose to seek the following low-rank representation on the raw tensor data (see illustration in Fig. 1):

$$\min_{\mathcal{Z},\mathcal{E}} \ \|\mathcal{Z}\|_* + \lambda\|\mathcal{E}\|_1, \ \text{s.t.} \ \mathcal{X} = \mathcal{A} * \mathcal{Z} + \mathcal{E}, \tag{2}$$

where $\|\mathcal{Z}\|_*$ and $\|\mathcal{E}\|_1$ respectively denote the tensor nuclear norm [13] and the tensor ℓ_1 norm (see more details in Table 1); $*$ denotes the tensor product defined in Definition 1; each slice $\mathcal{A}(:,t,:)$ in $\mathcal{A} \in \mathbb{R}^{n_1 \times n_4 \times n_3}$ denotes a representation base and $\mathcal{Z}(:,t,:)$ gives the corresponding representation.

We prove that under mild conditions, TLRR *exactly* recovers the clean data $\mathcal{L}_0$ via $\mathcal{L}_0 = \mathcal{A} * \mathcal{Z}_\star$ or $\mathcal{L}_0 = \mathcal{X} - \mathcal{E}_\star$, where $(\mathcal{Z}_\star, \mathcal{E}_\star)$ is the minimizer to problem (2). This exact recovery result allows the tubal rank (see Definition 3) of $\mathcal{L}_0$ and the amount of noise to be as high as $\mathcal{O}(\min(n_1,n_2)/\log(n_3\max(n_1,n_2)))$ and $\mathcal{O}(n_1n_2n_3)$, respectively. That is, our TLRR can handle the challenging case where the number of noisy entries is at the same order as the total entry number of the observed tensor and the underlying clean data $\mathcal{L}_0$ have a high tubal rank. We also prove when the data are noise-free, the representation coefficient $\mathcal{Z}_\star$ has a tensor block-diagonal structure formed by nonzero entries (see $\mathcal{Z}$ in Fig. 1). It directly gives the tensor subspace (see Section 5.2) that a specific sample locates in and thus the clustering results over the samples in the tensor. Accordingly, clustering can benefit from the exact recovery guarantee, as if $\mathcal{L}_0$ can be exactly recovered, then the learned $\mathcal{Z}_\star$ would be tensor block-diagonal. Compared with the procedure of first applying data recovery methods, e.g., [13], [14], [21], to denoise data and then performing clustering on the recovered clean data, our TLRR can simultaneously achieve recovery and clustering of low-rank tensor data with theoretic guarantees and experimentally validated efficacy.

The dictionary $\mathcal{A}$ plays an important role in TLRR. We propose two simple yet effective dictionary construction approaches to enable the samples in $\mathcal{X}$ to be linearly represented by the constructed dictionary. When the observed tensor $\mathcal{X}$ is not grossly corrupted, we directly use $\mathcal{X}$ itself as the dictionary. This gives the first variant of TLRR called simple TLRR (S-TLRR). When corruptions are severe, i.e., sparse noise with large magnitude, we propose to take the estimated clean data by R-TPCA [13] as the dictionary $\mathcal{A}$, giving the second method called robust TLRR (R-TLRR). For both methods, the learned representation $\mathcal{Z}_\star$ indicates the similarity between samples and can be used for data clustering. Note, R-TLRR can also work on the data that are not severely corrupted, though it might be less computationally efficient than S-TLRR since it needs more efforts to construct the dictionary.

Finally, considering the high cost of directly solving our problem by standard convex optimization techniques, we propose to reformulate it into an equivalent one with reduced size, which can be solved efficiently by the alternating direction method of multipliers (ADMM) [22]. Accordingly, we significantly reduce the computational complexity from $\mathcal{O}((n_1+n_2)n_2^2n_3)$ to $\mathcal{O}(r_{\mathcal{A}}n_1n_2n_3)$ at each iteration, where the tubal rank $r_{\mathcal{A}}$ of dictionary $\mathcal{A}$ is usually much smaller than n_2.

To sum up, this paper makes the following contributions.

1) For the first time, we propose a tensor low-rank representation learning method for effective and robust low-rank tensor analysis with theoretical guarantees. TLRR can not only exactly recover the clean low-rank data but also reveal the data cluster structure. The latter feature is absent from existing low-rank tensor analysis methods.
2) We develop an efficient algorithm to optimize our problem. Instead of directly solving the original problem, we manage to simplify it and reduce the computational complexity from $\mathcal{O}((n_1+n_2)n_2^2n_3)$ to $\mathcal{O}(r_{\mathcal{A}}n_1n_2n_3)$.
3) We propose to use tensor linear representation for characterizing the linear relations among tensor samples, and theoretically show its advantages over vector linear representation when capturing complex data relations.
4) We provide theoretical performance guarantees for TLRR. Specifically, TLRR can exactly recover the clean data $\mathcal{L}_0$ with high probability, even in challenging situations where the tubal rank of $\mathcal{L}_0$ and the amount of noise are very high.
5) We also prove that for noise-free data, the minimizer to the TLRR model is tensor block-diagonal and

TABLE 1
Notational Convention in This Article

$\mathcal{B}$	A tensor.	b	A vector.
B	A matrix.	b	A scalar.
$\mathcal{I}$	The identity tensor.	$\mathcal{B}^*$	The conjugate transpose of $\mathcal{B}$.
$\mathcal{B}_{ijk}$	The (i,j,k)-th entry of $\mathcal{B}$.	$\mathcal{B}(i,j,:)$	The (i,j)-th tube of $\mathcal{B}$.
$\mathcal{B}(i,:,:)$	The i-th horizontal slice of $\mathcal{B}$.	$\bar{\mathcal{B}}$	The DFT of $\mathcal{B}$.
$\mathcal{B}(:,i,:)$	The i-th lateral slice of $\mathcal{B}$.	$\bar{B}^{(i)}$	$\bar{\mathcal{B}}(:,:,i)$.
$\mathcal{B}(:,:,i)$	The i-th frontal slice of $\mathcal{B}$.	$\|\mathcal{B}\|_1$	$\|\mathcal{B}\|_1 = \sum_{i,j,k}\|\mathcal{B}_{ijk}\|$.
$B^{(i)}$	$B^{(i)} = \mathcal{B}(:,:,i)$.	$\|\mathcal{B}\|_\infty$	$\|\mathcal{B}\|_\infty = \max_{i,j,k}\|\mathcal{B}_{ijk}\|$.
$\mathcal{B}_{(i)}$	$\mathcal{B}_{(i)} = \mathcal{B}(:,i,:)$.	$\|\mathcal{B}\|_F$	$\|\mathcal{B}\|_F = \sqrt{\sum_{i,j,k}\|\mathcal{B}_{ijk}\|^2}$.
rank(B)	The rank of matrix B.	$\|B\|_*$	Sum of singular values of B.
$\|B\|_1$	$\|B\|_1 = \sum_{i,j}\|B_{ij}\|$.	$\|B\|_F$	$\|B\|_F = \sqrt{\sum_{i,j}\|B_{ij}\|^2}$.
$\vec{\mathfrak{e}}_t^n$	$\vec{\mathfrak{e}}_t^n \in \mathbb{R}^{n\times 1\times n_3}$ whose $(t,1,1)$-th entry is 1 and the rest are $\mathbf{0}$.		
$\mathcal{P}_{\mathcal{U}}(\mathcal{B})$	The projection onto $\mathcal{U}$: $\mathcal{P}_{\mathcal{U}}(\mathcal{B}) = \mathcal{U} * \mathcal{U}^* * \mathcal{B}$.		
$\mathcal{U}_0 * \mathcal{S}_0 * \mathcal{V}_0^*$	Skinny t-SVD of clean data $\mathcal{L}_0$.		
$\mathcal{U}_{\mathcal{A}} * \mathcal{S}_{\mathcal{A}} * \mathcal{V}_{\mathcal{A}}^*$	Skinny t-SVD of clean data $\mathcal{A}$.		

indicates clustering structure of tensors straightforwardly, under a reasonable assumption on the tensor subspaces that samples locate in.

Extensive experimental results show that our method outperforms state-of-the-art subspace methods on the two important visual tasks, image/video recovery and face clustering.

2 Related Work

As aforementioned, LRR [3], [4] can cluster vector-valued samples $\boldsymbol{X}$ into corresponding subspaces through seeking low-rank linear representations $\boldsymbol{Z}$ w.r.t. a given dictionary $\boldsymbol{A}$:

$$\min_{\boldsymbol{Z}} \ \|\boldsymbol{Z}\|_* + \lambda\|\boldsymbol{E}\|_1, \ \text{s.t.} \ \boldsymbol{X} = \boldsymbol{A}\boldsymbol{Z} + \boldsymbol{E}. \tag{3}$$

Here $\|\cdot\|_*$ and $\|\cdot\|_1$ respectively denote the matrix nuclear and ℓ_1 norms. But LRR and its variants [8], [9] can only process 2-way data and cannot effectively handle common multi-way data (tensor), e.g., videos, image collections and finance data. Besides, if one reshapes tensor data into a big matrix and then applies the above matrix-based methods, the performance would severely drop due to destroying the low-rank structure and losing vital information [10], [11], [12], [13]. Thus, their application scope is limited.

Recently, Lu et al. [13] extended R-PCA [21] from the 2-way matrix to 3-way tensor data and considered the following robust tensor PCA (R-TPCA) problem:

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{E}}} \ \|\boldsymbol{\mathcal{L}}\|_* + \lambda\|\boldsymbol{\mathcal{E}}\|_1, \ \text{s.t.} \ \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{E}}.$$

Under certain conditions, R-TPCA can recover the clean data $\boldsymbol{\mathcal{L}}_0$ from noisy observation $\boldsymbol{\mathcal{X}}$. Comparatively, our TLRR can handle data sampled from a mixture of *multiple* tensor subspaces (see Theorem 3) thus has wider applications for realistic data analysis than R-TPCA. Moreover, TLRR can also cluster tensor data while R-TPCA cannot. Finally, TLRR provides different and more general theoretical recovery guarantees, and it degenerates to R-TPCA when taking an identity tensor as the dictionary $\boldsymbol{\mathcal{A}}$.

Besides the one used in this paper, there are several different definitions of tensor rank, leading to different low-rank tensor analysis methods. The CP rank [23], defined as the smallest number of factors in rank-one tensor decomposition, is generally NP-hard to compute [24], [25]. The Tucker rank [16] is defined on the unfolding matrices to depict the rank of a tensor. As minimizing the rank function is complex due to its combinational nature, Liu et al. [26] used the sum of the nuclear norm (SNN) $\sum_{i=1}^{k}\|\boldsymbol{X}_i\|_*$ to approximate the tensor rank in tensor completion problems. Later Huang et al. [14] and Goldfarb et al. [27] adopted SNN to recover low-rank tensor. But Romera et al. [28] proved that SNN is not a tight convex relaxation of $\sum_{i=1}^{k}\mathrm{rank}(\boldsymbol{X}_i)$. Song et al. [29] extended matrix CUR decomposition [30] to tensor CURT decomposition and proposed an algorithm to compute a low-rank CURT approximation to a tensor. But both this method and SNN [14] cannot learn the relations among samples, in contrast to our TLRR. Finally, if applying data recovery methods, e.g., R-TPCA [13] and SNN [14], to clustering, one needs to first apply them to denoise data and then cluster estimated clean data. Comparatively, our TLRR simultaneously recovers the clean data and clusters them. TLRR offers stronger clean data recovery capacity by considering more complex data structure. TLRR also directly guarantees tensor block-diagonal structures of its optimal solution, indicating the clustering structure of tensors straightforwardly (see Section 5.2).

Recently, Fu et al. [15] proposed a tensor low-rank representation and sparse coding based method (TLRRSC) for data clustering. Specifically, they arranged n samples of feature dimension n_k into a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_k}$, where $n = n_1 n_2\cdots n_{k-1}$. Then they adopted a low-rank Tucker decomposition on the spatial modes (modes 1 to $k-1$) to depict the low-rank spatial relations among samples, and used sparse coding [17] to capture the linear relations of samples in the feature space (mode k). However, TLRRSC still needs to reshape 2-way data into vectors. Zhang et al. [18] proposed a low-rank tensor constrained multi-view subspace clustering method. LT-MSC first extracts several kinds of features to construct the multi-view data, and then restrains the learned representation tensor constructed by the representation matrix for each view data to be low Tucker rank. Differently, our TLRR directly considers the multi-way structure of the raw tensor data and avoids ad hoc extracting features. Besides, our TLRR has theoretical guarantees for data recovery and clustering which are missing in both TLRRSC and LT-MSC.

3 Notations and Preliminaries

In this section, we summarize the notations, definitions and preliminaries throughout this paper, which can avoid re-introduction and help readers locate them.

3.1 Notations

Table 1 summarizes the notations used in this paper. $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{n\times n\times n_3}$ denotes the *identity tensor* whose first frontal slice is an $n \times n$ identity matrix, and the entries in the other frontal slices are all zeros. The *conjugate transpose* $\boldsymbol{\mathcal{B}}^* \in \mathbb{C}^{n_2\times n_1\times n_3}$ of tensor $\boldsymbol{\mathcal{B}} \in \mathbb{C}^{n_1\times n_2\times n_3}$ is obtained by conjugate transposing each frontal slice of $\boldsymbol{\mathcal{B}}$ and then reversing the order of transposed frontal slices 2 through n_3. The inner product of two tensors $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1\times n_2\times n_3}$ and $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{n_1\times n_2\times n_3}$ is defined as $\langle\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{C}}\rangle = \sum_{i=1}^{n_3}\langle\boldsymbol{B}^{(i)},\boldsymbol{C}^{(i)}\rangle$.

Now we introduce the Discrete Fourier Transformation (DFT) on a tensor used in Section 3.2. Let $\bar{\boldsymbol{\mathcal{B}}} \in \mathbb{C}^{n_1\times n_2\times n_3}$ be the DFT of $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1\times n_2\times n_3}$ along the 3rd dimension. Define the *DFT matrix*:

$$\boldsymbol{F}_{n_3} = [\boldsymbol{f}_1, \ldots, \boldsymbol{f}_i, \ldots, \boldsymbol{f}_{n_3}] \in \mathbb{R}^{n_3\times n_3}, \tag{4}$$

where $\boldsymbol{f}_i = [\omega^{0\times(i-1)}; \omega^{1\times(i-1)}; \cdots; \omega^{(n_3-1)\times(i-1)}] \in \mathbb{R}^{n_3}$ with $\omega = e^{-(2\pi\sqrt{-1}/n_3)}$. Then we have $\bar{\boldsymbol{\mathcal{B}}}(i,j,:) = \boldsymbol{F}_{n_3}\boldsymbol{\mathcal{B}}(i,j,:)$. Indeed, we can compute $\bar{\boldsymbol{\mathcal{B}}}$ directly by the Matlab command $\bar{\boldsymbol{\mathcal{B}}} = \mathtt{fft}(\boldsymbol{\mathcal{B}}, [\,], 3)$ and use the inverse DFT to obtain $\boldsymbol{\mathcal{B}} = \mathtt{ifft}(\bar{\boldsymbol{\mathcal{B}}}, [\,], 3)$. Then the *tensor spectral norm* of $\boldsymbol{\mathcal{B}}$ is defined as $\|\boldsymbol{\mathcal{B}}\| = \|\bar{\boldsymbol{\mathcal{B}}}\|$ [13]. For brevity, we define $\bar{\boldsymbol{B}} \in \mathbb{C}^{n_1 n_3\times n_2 n_3}$ as

$$\bar{\boldsymbol{B}} = \mathtt{bdiag}(\bar{\boldsymbol{\mathcal{B}}}) = \begin{bmatrix} \bar{\boldsymbol{B}}^{(1)} & & & \\ & \bar{\boldsymbol{B}}^{(2)} & & \\ & & \ddots & \\ & & & \bar{\boldsymbol{B}}^{(n_3)} \end{bmatrix} \in \mathbb{C}^{n_1 n_3\times n_2 n_3}, \tag{5}$$

where $\texttt{bdiag}(\cdot)$ unfolds the tensor $\bar{\mathcal{B}}$ to a block-diagonal matrix $\bar{B}$. Then we define the block circulant matrix $\texttt{bcirc}(\mathcal{B})$ of $\mathcal{B}$ as

$$\texttt{bcirc}(\mathcal{B}) = \begin{bmatrix} B^{(1)} & B^{(n_3)} & \cdots & B^{(2)} \\ B^{(2)} & B^{(1)} & \cdots & B^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ B^{(n_3)} & B^{(n_3-1)} & \cdots & B^{(1)} \end{bmatrix} \in \mathbb{R}^{n_1 n_3 \times n_2 n_3}.$$

Then we define the operator $\texttt{unfold}$ and its inverse operator $\texttt{fold}$ as

$$\texttt{unfold}(\mathcal{B}) = \begin{bmatrix} B^{(1)} \\ B^{(2)} \\ \vdots \\ B^{(n_3)} \end{bmatrix} \in \mathbb{R}^{n_1 n_3 \times n_2}, \ \texttt{fold}(\texttt{unfold}(\mathcal{B})) = \mathcal{B}.$$

Based on the definitions of $\bar{B}$, $\texttt{bcirc}(\mathcal{B})$ and $\texttt{unfold}(\mathcal{B})$, we can define the tensor rank and nuclear norm as introduced below.

3.2 Preliminaries

We start with explaining *t-product* for tensor product computation and then give necessary definitions for developing TLRR.

Definition 1 (t-product) [31]. *The t-product between two tensors $\mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{C} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$ is defined as $\mathcal{B} * \mathcal{C} = \texttt{fold}(\texttt{bcirc}(\mathcal{B}) \cdot \texttt{unfold}(\mathcal{C})) \in \mathbb{R}^{n_1 \times n_4 \times n_3}$.*

Indeed, t-product is equivalent to the matrix multiplication in the Fourier domain, i.e., $\mathcal{F} = \mathcal{B} * \mathcal{C}$ and $\bar{F} = \bar{B}\bar{C}$ are equivalent [31]. A tensor $\mathcal{P} \in \mathbb{R}^{n \times n \times n_3}$ is *orthogonal* if $\mathcal{P}^* * \mathcal{P} = \mathcal{P} * \mathcal{P}^* = \mathcal{I}$ holds. A tensor is $f-diagonal$ if its each frontal slice is diagonal. Then we can define t-SVD for tensor SVD as follows.

Definition 2 (t-SVD and skinny t-SVD) [31], [32], [33]. *For any $\mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, it can be factorized by t-SVD as $\mathcal{B} = \mathcal{U} * \mathcal{S} * \mathcal{V}^*$, where $\mathcal{U} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ and $\mathcal{V} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ are orthogonal, and $\mathcal{S} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is f-diagonal. Then the skinny t-SVD of $\mathcal{B}$ is $\mathcal{B} = \mathcal{U}_s * \mathcal{S}_s * \mathcal{V}_s^*$, where $\mathcal{U}_s = \mathcal{U}(:, 1:r, :)$, $\mathcal{S}_s = \mathcal{S}(1:r, 1:r, :)$, and $\mathcal{V}_s = \mathcal{V}(:, 1:r, :)$ in which r denotes the tensor tubal rank of $\mathcal{B}$ (see Definition 3).*

Based on t-SVD, we can describe the low-rank structure of a tensor by defining following two tensor ranks.

Definition 3 (Tensor average and tubal rank) [10], [13]. *For any $\mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, let $\boldsymbol{r} = (\text{rank}(\bar{B}^{(1)}); \cdots; \text{rank}(\bar{B}^{(n_3)}))$. The tensor* average rank *of $\mathcal{B}$ is defined as*

$$rank_a(\mathcal{B}) = \frac{1}{n_3} \sum\nolimits_{i=1}^{n_3} \boldsymbol{r}_i = \frac{1}{n_3} rank(\bar{B}).$$

The tensor tubal rank $\text{rank}_t(\mathcal{B})$ *is defined as the number of nonzero singular tubes of $\mathcal{S}$, i.e.,*

$$rank_t(\mathcal{B}) = \#\{i : \mathcal{S}(i,i,:) \neq 0\} = \max(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_3}),$$

where $\mathcal{S}$ is from the t-SVD of $\mathcal{B} = \mathcal{U} * \mathcal{S} * \mathcal{V}^*$.

To pursue a low-rank tensor, one can minimize the tensor rank defined as above which however is NP-hard. To alleviate the optimization difficulty, we instead minimize the tensor nuclear norm $\|\mathcal{B}\|_*$ defined below, which is a convex envelop of the tensor average rank within the unit ball of the tensor spectral norm [13].

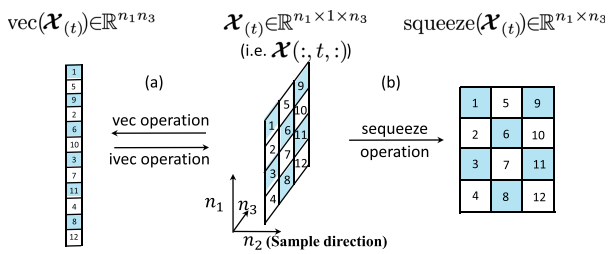


Fig. 2. Illustration of the vec, *ivec* and squeeze operations.

Definition 4 (Tensor nuclear norm) [13], [34], [35]. *The tensor nuclear norm $\|\mathcal{B}\|_*$ of a tensor $\mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is defined as the average of the nuclear norm of all the frontal slices of $\bar{\mathcal{B}}$, i.e.,*

$$\|\mathcal{B}\|_* = \frac{1}{n_3} \sum_{i=1}^{n_3} \|\bar{B}^{(i)}\|_* = \frac{1}{n_3} \|\bar{B}\|_*,$$

which is the convex envelop of the tensor average rank within the unit ball of the tensor spectral norm.

Next, we introduce the pseudo-inverse of a tensor.

Definition 5 (Tensor pseudo-inverse). *For an arbitrary tensor $\mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, its pseudo-inverse is defined as a tensor $\mathcal{B}^\dagger \in \mathbb{R}^{n_2 \times n_1 \times n_3}$ which satisfies (a) $\mathcal{B} * \mathcal{B}^\dagger * \mathcal{B} = \mathcal{B}$, (b) $\mathcal{B}^\dagger * \mathcal{B} * \mathcal{B}^\dagger = \mathcal{B}^\dagger$, (c) $\mathcal{B} * \mathcal{B}^\dagger = (\mathcal{B} * \mathcal{B}^\dagger)^*$, (d) $\mathcal{B}^\dagger * \mathcal{B} = (\mathcal{B}^\dagger * \mathcal{B})^*$.*

The pseudo-inverse of a tensor $\mathcal{B}$ can be computed frontal-slice-wisely after DFT. See more details in Section 4.1 in supplementary, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2019.2954874. Finally, we introduce a lemma that is essential for developing TLRR method and following theoretical analysis.

Lemma 1 [31]. *Suppose $\mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{C} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$ are two arbitrary tensors. Let $\mathcal{F} = \mathcal{B} * \mathcal{C}$. Then, we have*

1) $\|\mathcal{B}\|_F^2 = \frac{1}{n_3}\|\bar{B}\|_F^2$ *and* $\langle \mathcal{B}, \mathcal{C} \rangle = \frac{1}{n_3} \langle \bar{B}, \bar{C} \rangle$;
2) $\mathcal{F} = \mathcal{B} * \mathcal{C}$ *and* $\bar{F} = \bar{B}\bar{C}$ *are equivalent to each other.*

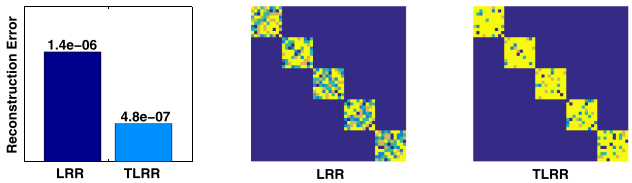
Lemma 1 is induced by tensor product definition and the orthogonality of $F_{n_3}/\sqrt{n_3}$, i.e. $F_{n_3}^* F_{n_3} = F_{n_3} F_{n_3}^* = n_3 I_{n_3}$.

4 Tensor Linear Representation

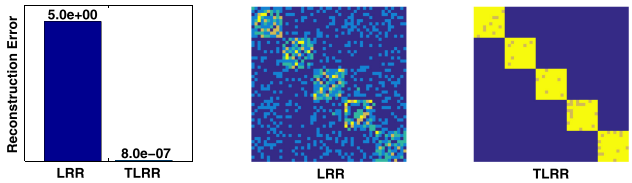
In this work, we propose a method for pursuing tensor low-rank linear representation. Before introducing the proposed method, here we explain the intuition behind the tensor linear representation $\mathcal{X} = \mathcal{A} * \mathcal{Z}$, which generalizes the data linear representation from the vector space to the tensor space. We introduce two operators: $\texttt{vec}(\cdot)$ and $ivec(\cdot)$. Let $\mathcal{X}_{(t)}$ be the tth lateral slice $\mathcal{X}(:,t,:)$ (see Table 1). As shown in Fig. 2, vec vectorizes each sample $\mathcal{X}_{(t)}$ in $\mathcal{X}$, while *ivec* is its inverse operation.

For any sample $\mathcal{X}_{(t)}$, if its vectorization $\boldsymbol{x}_{(t)} = \texttt{vec}(\mathcal{X}_{(t)})$ can be linearly represented by the bases in $A \in \mathbb{R}^{n_1 n_3 \times p'}$:

$$\boldsymbol{x}_{(t)} = A z_{(t)}, \ \forall t = 1, \ldots, n_2, \tag{6}$$



(a) Results of LRR and TLRR on the data with vector linear relations.



(b) Results of LRR and TLRR on the data with tensor linear relations.

Fig. 3. Comparison of vector linear representation and tensor linear representation. (a) Reports the reconstruction errors of LRR and TLRR which respectively are based on vector and tenslr linear representations, the learnt block-diagonal structures by LRR and TLRR, when the testing data have vector linear relations. (b) Reports the results under the same metrics on the data with tensor linear relations. *Best viewed in color pdf file.*

then one can always find two tensors $\mathcal{A}$ and $\mathcal{Z}$ such that tensor linear representation $\mathcal{X} = \mathcal{A} * \mathcal{Z}$ holds, as stated in Theorem 1.

Theorem 1. *If Eqn. (6) holds, then there exist two tensors* $\mathcal{A} \in \mathbb{R}^{n_1 \times p' \times n_3}$ *and* $\mathcal{Z} \in \mathbb{C}^{p' \times n_2 \times n_3}$ *such that*

$$\mathcal{X}_{(t)} = ivec(\boldsymbol{x}_{(t)}) = \mathcal{A} * \mathcal{Z}_{(t)}, \ (\forall t = 1, \ldots, n_2), \tag{7}$$

where $\mathcal{A}$ *can be found through* $\mathcal{A}_{(t)} = ivec(A(:,t))$ *and* $\mathcal{Z}$ *can be computed as* $\mathcal{Z}_{(t)} = \texttt{ifft}(\bar{\mathcal{Z}}_{(t)}, [\,], 3)$ *in which* $\bar{\mathcal{Z}}_{(t)}(:,t,j) = z_{(t)}$ $(j = 1, \cdots, n_3)$. *However, if there exists a* $\mathcal{Z}_{(t)}$ *such that Eqn. (7) holds, Eqn. (6) may not hold.*

The proof of Theorem 1 can be found in Section 5 in supplementary material, available online. Theorem 1 implies that if vectorized data samples can be linearly represented in the vector space, the tensor linear representation (7) for original tensor data also holds. One appealing advantage of tensor linear representation is that it gets rid of vectorizing tensor data for describing and analyzing their relationship. Thus using tensor linear representation effectively avoids destroying the multi-way structure and losing information [10], [11], [13]. Moreover, Theorem 1 also implies that the tensor linear representation can capture more complex data relations that cannot be depicted by vector linear representation. In this sense, tensor linear representation is more general and advantageous over vector linear representation for tensor data analysis.

Example 1. We give an example to demonstrate the above mentioned advantages of the tensor linear representation over vector linear representation. We first investigate whether tensor linear representation can well learn the linear relations and the vector subspace structure in the data with vector linear relations. Toward this goal, we generate a testing data matrix $X = [X_1, X_2, \ldots, X_k]$ with $X_i = A_i Z_i \in \mathbb{R}^{n_1 n_3 \times s}$ where $A_i \in \mathbb{R}^{n_1 n_3 \times p'}$ is a random orthogonal matrix and $Z_i \in \mathbb{R}^{p' \times s}$ is an i.i.d. $\mathcal{N}(0,1)$ matrix. In this way, the samples in X obey vector linear representation, and samples in X_i are drawn from the ith vector subspace spanned by A_i. Then we solve the LRR model Eqn. (3) to obtain its minimizer (Z^*, E^*) and report the reconstruction error $\|AZ^* - X\|_F$ in Fig. 3a to measure whether vector linear representation can learn the linear relations among data X. Next, we respectively use X and A to construct their tensor versions $\mathcal{X} \in \mathbb{R}^{n_1 \times ks \times n_3}$ and $\mathcal{A} \in \mathbb{R}^{n_1 \times kp' \times n_3}$. We also solve the TLRR model Eqn. (2) to obtain its optimizer $(\mathcal{Z}^*, \mathcal{E}^*)$ and report the reconstruction error $\|\mathcal{A} * \mathcal{Z}^* - \mathcal{X}\|_F$ in Fig. 3a. In the experiments, we set $n_1 = n_3 = 100$, $p' = s = 10$ and $k = 5$. By comparison, we can observe that the reconstruction error of TLRR is as small as that of LRR, demonstrating the representation capacity of tensor linear representation to the data with vector linear relations. Moreover, in Fig. 3a we also display the sample similarity matrices $\widetilde{Z} = \frac{1}{2}(|Z^*| + |Z^*|^T)$ in LRR and $\widehat{Z} = \frac{1}{2n_3}\sum_{i=1}^{n_3}(|\mathcal{Z}^*(:,:,i)| + |\mathcal{Z}^*(:,:,i)|^T)$ in TLRR, where each $\mathcal{Z}^*(:,:,i)$ learnt by TLRR captures the sample relations (see details in Section 5.1). From Fig. 3a, one can observe that tensor linear representation is capable of capturing the vector subspace structure in X and even learns better block-diagonal structure than LRR. On the contrary, we investigate the learning capacity of vector linear representation to the data with tensor linear relations. We generate $\mathcal{X} = [\mathcal{X}_1, \ldots, \mathcal{X}_k]$ with $\mathcal{X}_i = \mathcal{A}_i * \mathcal{Z}_i$, where $\mathcal{A}_i \in \mathbb{R}^{n_1 \times p' \times n_3}$ is a random orthogonal tensor and $\mathcal{Z}_i \in \mathbb{R}^{p' \times s \times n_3}$ are from i.i.d. $\mathcal{N}(0,1)$. Accordingly, the samples in $\mathcal{X}$ have tensor linear relations and samples in $\mathcal{X}_i$ lie in the ith tensor subspace (see Definition 6). Then by performing TLRR on $\mathcal{X}$, the learnt similarity matrix $\widehat{Z}$ has block-diagonal structure as shown in Fig. 3b which indicates the tensor subspace structures. Please refer to Section 5.2. Then, we vectorize $\mathcal{X}$ and $\mathcal{A}$ to solve LRR and report its reconstruction error and the learnt $\widetilde{Z}$ in Fig. 3b. Comparatively, vector linear representation has much larger reconstruction error than that of TLRR, which means it cannot learn the linear relationship of samples with tensor linear relations. Furthermore, the learnt structure of $\widehat{Z}$ in LRR also does not reveal the tensor subspace structure in $\mathcal{X}$. So these results well demonstrate the superiority of tensor linear representation on complex tensor data over vector linear representation.

5 TLRR for Data Clustering

In this section, we elaborate how TLRR formulated in Eqn. (2) can be applied for tensor data clustering. Specifically, we first introduce the TLRR based clustering algorithm in Section 5.1, and then theoretically analyze its clustering performance in Section 5.2. Finally, Section 5.3 interprets the TLRR based clustering algorithm from multi-view aspect, which can help understand the advantages of TLRR over matrix LRR [1], [4].

5.1 Algorithm for Clustering

Here we elaborate the TLRR based clustering algorithm which is used for clustering tensor data in this work. For clustering, here we take the data $\mathcal{X}$ or the recovered datum by other methods, e.g., R-TPCA [13], as the dictionary $\mathcal{A}$. In this case, each of the frontal slices $Z_\star^{(i)} \in \mathbb{R}^{n_2 \times n_2}$ $(i = 1, \ldots, n_3)$ of the learned representation $\mathcal{Z}_\star$ is a similarity matrix of samples. To apply the existing clustering tools, such as Ncut [36], we combine all $Z_\star^{(i)}$ into one affinity matrix $\widehat{Z}$:

$$\widehat{Z} = \tfrac{1}{2n_3}\textstyle\sum_{i=1}^{n_3}\left(|Z_\star^{(i)}| + |(Z_\star^{(i)})^*|\right). \tag{8}$$

Now we give our clustering algorithm below.

Algorithm 1. Data Clustering

Input: data $\mathcal{X}$, dictionary $\mathcal{A}$, and number k of clusters.
1. Obtain the minimizer $\mathcal{Z}_\star$ to problem (2).
2. Construct a similarity matrix $\widehat{Z}$ by (8).
3. Perform Ncut on $\widehat{Z}$ to cluster the samples into k clusters.

Output: clustering results.

In this work, we use Algorithm 1 to cluster tensor data and investigate its practical performance in Section 9.1. In the following subsection, we theoretically analyze its clustering performance.

5.2 Tensor Block-Diagonal Property

Here we analyze the performance of Algorithm 1 by analyzing the minimizer's structure of the TLRR model. We start with a simpler case where the tensor data are noise-free and the TLRR problem (2) degenerates to

$$\min_{\mathcal{Z}} \|\mathcal{Z}\|_*, \text{ s.t. } \mathcal{X} = \mathcal{A} * \mathcal{Z}. \tag{9}$$

If sparse corruptions are present, one can solve the original problem (2). Now we analyze the structure of the minimizer to noiseless TLRR Eqn. (9) for data clustering.

To analyze tensor data, here we develop certain new concepts and tools in tensor space which are consistent with those in the vector space. A *tensor space* is a set of tensors that is closed under finite tensor addition and scalar multiplication. Here the *tensor space* specially refers to the set $\mathbb{S} = \{\forall \mathcal{S} \in \mathbb{R}^{n_1 \times 1 \times n_3}\}$. A set of tensors $\{\mathcal{D}_{(1)}, \ldots, \mathcal{D}_{(p)}\} \subseteq \mathbb{S}$, where $\mathcal{D}_{(t)}$ is the tth lateral slice of $\mathcal{D} \in \mathbb{R}^{n_1 \times p \times n_3}$, is said to be *linearly independent* if there is not a nonzero $\mathcal{C} \in \mathbb{R}^{p \times 1 \times n_3}$ satisfying $\mathcal{D} * \mathcal{C} = 0$.

Definition 6 (Tensor subspace). *Given a set $\{\mathcal{D}_{(1)}, \ldots, \mathcal{D}_{(p)}\} \subseteq \mathbb{S}$ in which the elements $\mathcal{D}_{(i)}$ are linearly independent, the set $\mathcal{K} = \{\mathcal{Y} \,|\, \mathcal{Y} = \mathcal{D} * \mathcal{C}, \forall \mathcal{C} \in \mathbb{R}^{p \times 1 \times n_3}\}$ is called a tensor subspace of dimension $\dim(\mathcal{K}) = p$. Here $\mathcal{D}_{(1)}, \ldots, \mathcal{D}_{(p)}$ are the basis spanning $\mathcal{K}$.*

The defined tensor subspace includes the tensor 0 and is closed under tensor addition and scalar multiplication. Also when $n_3 = 1$, the tensor subspace reduces to the subspace defined on vectors. Next, we define the tensor direct sum, which also accords with the direct sum defined on the vector subspace when $n_3 = 1$.

Definition 7 (Tensor direct sum). *$\mathcal{K} = \oplus_{j=1}^{k} \mathcal{K}_j$ is called the direct sum of tensor subspaces $\{\mathcal{K}_1, \ldots, \mathcal{K}_k\}$, if for any $\mathcal{F} \in \mathbb{R}^{n_1 \times 1 \times n_3}$ in $\mathcal{K}$, there is a unique $\mathcal{F}_j \in \mathbb{R}^{n_1 \times 1 \times n_3}$ in $\mathcal{K}_j$ for $1 \le j \le k$ such that $\mathcal{F} = \sum_{j=1}^{k} \mathcal{F}_i$.*

If the tensor subspaces $\{\mathcal{K}_1, \cdots, \mathcal{K}_k\}$ obey the condition $\sum_{j=1}^{k} \mathcal{K}_j = \oplus_{j=1}^{k} \mathcal{K}_j$, they are said to be *independent*. Now we present our main results in Theorem 2 which guarantee the tensor block-diagonal structure of the learnt representation and indicate the clustering structure of tensors.

Theorem 2. *Suppose $\mathcal{X} = [\mathcal{X}_1, \ldots, \mathcal{X}_k]$, and the samples $(\mathcal{X}_j(:, i, :))$ in $\mathcal{X}_j \in \mathbb{R}^{n_1 \times m_j \times n_3}$ are drawn from the jth tensor subspace $\mathcal{K}_j$ whose p_j basis composes $\mathcal{A}_j \in \mathbb{R}^{n_1 \times p_j \times n_3}$. Then if $\{\mathcal{K}_1, \ldots, \mathcal{K}_k\}$ are independent, the minimizer $\mathcal{Z}_\star$ to problem (9) is tensor block-diagonal (see "$\mathcal{Z}$" in Fig. 1). Namely, each frontal slice $Z_\star^{(i)}$ of $\mathcal{Z}_\star$ has the following block-diagonal structure:*

$$Z_\star^{(i)} = \begin{bmatrix} (Z_\star^{(i)})_1 & 0 & \cdots & 0 \\ 0 & (Z_\star^{(i)})_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (Z_\star^{(i)})_k \end{bmatrix} \in \mathbb{R}^{(\sum_j p_j) \times (\sum_j m_j)},$$

where $(Z_\star^{(i)})_j \in \mathbb{R}^{p_j \times m_j}$ is a coefficient matrix. Then the DFT result $\bar{\mathcal{Z}}_\star$ of $Z_\star$ has the same block-diagonal structure as $\mathcal{Z}_\star$.

We defer the proof of Theorem 2 to Section 6 in supplementary, available online. The subspace independence assumption here is not strict. Indeed, its vector version is a common assumption in sparse and low-rank data analysis [1], [2], [3]. Fig. 8 in Section 9.1 displays the block-diagonal structure learnt by TLRR on real face data, verifying the validity of the tensor subspace assumption and Theorem 2 as well.

Theorem 2 is useful for clustering, since the block-diagonal structure of $\mathcal{Z}_\star$ directly indicates the tensor subspace membership of a specific sample, according to which one can easily obtain the clusters. Specifically, for the tth sample $\mathcal{X}_{(t)}$, we have

$$\mathcal{X}_{(t)} = \mathcal{A} * \mathcal{Z}_{\star(t)} = \sum_{j=1}^{k} \mathcal{A}_j * \mathcal{Z}_{\star(t)}^{j}, \ \forall t = 1, \ldots, n_2, \tag{10}$$

where $\mathcal{Z}_{\star(t)}^{j} = \mathcal{Z}_\star(1 + \sum_{i=1}^{j-1} p_i : \sum_{i=1}^{j} p_i, t, :)$. By Theorem 2, if $\mathcal{X}_{(t)}$ is drawn from the sth subspace $\mathcal{K}_s$, $\mathcal{Z}_{\star(t)}^{s}$ will have nonzero entries while $\mathcal{Z}_{\star(t)}^{j}$ $(j \neq s)$ will be 0. Then according to the nonzero entries in $\mathcal{Z}_{\star(t)}$, we can cluster $\mathcal{X}_{(t)}$ accurately.

If we take the raw datum $\mathcal{X}$ itself as the dictionary, the learned representation $\mathcal{Z}_\star$ is also block-diagonal, as the samples are from independent tensor subspaces. Besides, by replacing $\mathcal{A}$ with $\mathcal{X}$ in Eqn. (10), we have $\mathcal{X}_{(t)} = \sum_{j=1}^{n_2} \mathcal{X}_{(j)} * \mathcal{Z}_\star(j, t, :)$. So the coefficient $\mathcal{Z}_\star(j, t, :)$ depicts the similarity between samples $\mathcal{X}_{(t)}$ and $\mathcal{X}_{(j)}$. To perform clustering (e.g. using Ncut) on the learnt representation $\mathcal{Z}_\star$, following Eqn. (8) we combine the two vectors $\mathcal{Z}_\star(t, j, :)$ and $\mathcal{Z}_\star(j, t, :)$ into a value $\widehat{Z}(t, j) = \frac{1}{n_3} \sum_{i=1}^{n_3} (|\mathcal{Z}_\star(t, j, i)| + |\mathcal{Z}_\star(j, t, i)|)$, which measures the similarity between the samples $\mathcal{X}_{(t)}$ and $\mathcal{X}_{(j)}$. Also, if $\mathcal{X}_{(t)}$ and $\mathcal{X}_{(j)}$ are drawn from different tensor subspaces, $\widehat{Z}(t, j)$ would be zero. Then, clustering can be performed on the similarity matrix $\widehat{Z}$.

For corrupted data, our Theorem 3 in Section 6.2 shows that TLRR can exactly recover the clean low-rank data under mild conditions. Then if the conditions in Theorem 2 are satisfied, the minimizer $\mathcal{Z}_\star$ will be tensor block-diagonal. We will discuss this later in more details in Section 6.2.

5.3 Multi-View Interpretation on TLRR

To better understand TLRR, we give another explanation from the multi-view aspect. By tensor nuclear norm definition $\|\mathcal{Z}\|_* = \frac{1}{n_3} \|\bar{Z}\|_*$ and the block-diagonal structure of $\bar{X}$, $\bar{A}$ and $\bar{Z}$ (see their definitions in Eqn. (5)), we rewrite Eqn. (9) into its equivalent form:

$$\min_{\bar{Z}^{(i)}} \|\bar{Z}^{(i)}\|_*, \text{ s.t. } \bar{X}^{(i)} = \bar{A}^{(i)} \bar{Z}^{(i)}, \ (i = 1, \ldots, n_3). \tag{11}$$

On the other hand, when performing DFT on $\mathcal{X}$, the tth column of $\bar{X}^{(i)}$ is from the tth sample $\mathcal{X}_{(i)}$. This is because we have

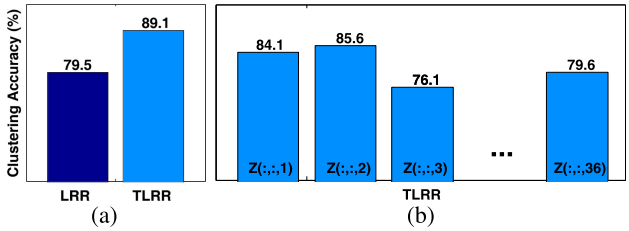


Fig. 4. Comparison of clustering accuracies of LRR and TLRR on the FRGC 2.0 dataset [37].

$$\bar{X}^{(i)} = \left[M^1 \boldsymbol{f}_i, M^2 \boldsymbol{f}_i, \ldots, M^t \boldsymbol{f}_i, \ldots, M^p \boldsymbol{f}_i \right].$$

Here $\boldsymbol{f}_i$ is the ith column of the DFT matrix F_{n_3} defined in Eqn. (4), and $M^t = \text{squeeze}(\mathcal{X}_{(t)}) \in \mathbb{R}^{n_1 \times n_3}$ where the operation squeeze transforms the tth sample $\mathcal{X}_{(t)}$ into a matrix (see Fig. 2). So in this sense, $\bar{X}^{(i)}$ can be viewed as the new features of samples in $\mathcal{X}$ under the ith Fourier basis $\boldsymbol{f}_i$. Similarly, $\bar{A}^{(i)}$ can be viewed as a new dictionary under the Fourier basis $\boldsymbol{f}_i$. Thus, from a multi-view aspect, the frontal slice $\bar{Z}^{(i)}$ $(i = 1, \ldots, n_3)$ learnt by model Eqn. (11) can be regarded as n_3 different representation matrices under n_3 different views and may better depict the relations among samples. Thus, the tth lateral $\bar{\mathcal{Z}}_{(t)}$ is the representation tensor of the tth sample $\mathcal{X}_{(t)}$. Conducting inverse DFT on $\bar{\mathcal{Z}}_{(t)}$ gives $\mathcal{Z}_{(t)} = \frac{1}{n_3} F_{n_3}^* \text{squeeze}(\bar{\mathcal{Z}}_{(t)})$. This combines all the representation information under n_3 different views. Such a mechanism also makes the advantage of TLRR stand out, as compared with matrix LRR [1], [4] which only learns the representation under a single view, TLRR aims at seeking n_3 representations under n_3 different views and may better reveal the relationship among samples. This advantage is actually testified by Example 1 in Section 4 : TLRR can learn more accurate subspace structures in the simulation data with vector or tensor linear relations, and thus it can better capture sample relations. Here we further demonstrate the advantage of TLRR on a realistic face dataset, namely FRGC 2.0 [37]. It consists of 60 classes and each class has 20 images of size 32×36. Here we use the raw data itself as the dictionary and then apply LRR and TLRR. In LRR we need to vectorize each image. See more experimental details in Section 9.1. From Fig. 4a, one can observe that for clustering accuracy, TLRR makes 9.6 percent improvement over LRR. This is because compared with LRR considering data from one single view, TLRR considers the data under n_3 $(= 36)$ different views and seeks n_3 representations which can better capture sample relations. This can be illustrated by observing Fig. 4a and 4b: compared with each single similarity matrix $\mathcal{Z}(:,:,i)$ in Fig. 4b, the uniform similarity matrix $\widehat{Z} = \frac{1}{2n_3} \sum_{i=1}^{n_3} (|\mathcal{Z}(:,:,i)| + |\mathcal{Z}(:,:,i)|^T)$ in Fig. 4a can achieve better clustering results, as the representations obtained from different views could provide complementary information and boost the clustering performance.

6 TLRR for Exact Data Recovery

Here we provide the theoretical performance guarantees for TLRR on data recovery. We prove that under mild conditions TLRR in Eqn. (2) can exactly recover the intrinsic low-rank data $\mathcal{L}_0$ lying in multiple tensor subspaces, even in presence of gross noise $\mathcal{E}_0$.

6.1 Incoherence Condition for TLRR

In this subsection, we introduce the incoherence condition of TLRR for exactly data recovery. Intuitively, exactly separating $\mathcal{X}$ into the low-rank term $\mathcal{L}_0$ plus the sparse term $\mathcal{E}_0$ requires $\mathcal{L}_0$ to be not too sparse. Similar consideration is made for the matrix case in [3], [21]. To formally characterize this intuition, we present the incoherence condition for low-rank tensors. Let $r = \text{rank}_t(\mathcal{L}_0)$. We define row incoherence parameter μ_1, column incoherence parameter μ_2, and total incoherence parameter μ_3 as follows.

$$\mu_1(\mathcal{L}_0) = \frac{n_2 n_3}{r} \max_{j=1,\cdots,n_2} \|\mathcal{V}_0^* * \boldsymbol{e}_j^{n_2}\|_F^2,$$
$$\mu_2(\mathcal{L}_0) = \frac{n_1 n_3}{r} \max_{i=1,\cdots,n_1} \|\mathcal{U}_0^* * \boldsymbol{e}_i^{n_1}\|_F^2,$$
$$\mu_3(\mathcal{L}_0) = \frac{n_1 n_2 n_3^2}{r} \left(\|\mathcal{U}_0 * \mathcal{V}_0^*\|_\infty\right)^2,$$

where the standard basis $\boldsymbol{e}_j^{n_2}$ and $\boldsymbol{e}_i^{n_1}$ are defined in Table 1. A small value of $\mu(\mathcal{L}_0) = \max(\mu_1(\mathcal{L}_0), \mu_2(\mathcal{L}_0), \mu_3(\mathcal{L}_0))$ implies the low-rank term $\mathcal{L}_0$ is not sparse. Such a condition is also critical for other tensor analysis methods, e.g., R-TPCA [13].

As TLRR involves an extra dictionary $\mathcal{A}$, we need to develop a new necessary incoherence condition that cannot be straight-forwardly generalized from the matrix case or the one used in R-TPCA [13]. Let $\text{rank}_t(\mathcal{A}) = r_{\mathcal{A}}$. Then we define a new row incoherence parameter related to $\mathcal{A}$:

$$\mu_1^{\mathcal{A}}(\mathcal{L}_0) = \mu_1(\mathcal{L}_0) \max_{i=1,\cdots,n_1} \|\mathcal{U}_{\mathcal{A}}^* * \boldsymbol{e}_i^{n_1}\|_F^2 = \frac{r_{\mathcal{A}}}{n_1 n_3} \mu_1(\mathcal{L}_0) \mu_2(\mathcal{A}).$$

The *incoherence condition* of TLRR for exact recovery only requires $\max(\mu_2(\mathcal{L}_0), \mu_1^{\mathcal{A}}(\mathcal{L}_0))$ to be sufficiently small and does not rely on $\mu_3(\mathcal{L}_0)$. So it is a looser condition compared with the exact recovery condition in R-TPCA [13].

6.2 Exact Recovery Performance Guarantee

We summarize the exact recovery results of TLRR in Theorem 3. Let $n_{(1)} = \max(n_1, n_2)$ and $n_{(2)} = \min(n_1, n_2)$. Recall $\mathcal{P}_{\mathcal{U}}(\cdot)$ is a projection operator defined in Table 1.

Theorem 3. *Assume the support set Ω of $\mathcal{E}_0$ is uniformly distributed, $\mathcal{P}_{\mathcal{U}_{\mathcal{A}}}(\mathcal{U}_0) = \mathcal{U}_0$, and $\mathcal{A}$ obeys that the ranks of $\bar{A}^{(i)}$ $(i=1, \ldots, n_3)$ are equal. Let $\mu^{\mathcal{A}} = \max(\mu_2(\mathcal{L}_0), \mu_1^{\mathcal{A}}(\mathcal{L}_0))$. If*

$$rank_t(\mathcal{L}_0) \leq \frac{\rho_r n_{(2)}}{\mu^{\mathcal{A}} \log(n_{(1)} n_3)} \quad and \quad |\Omega| \leq \rho_s n_1 n_2 n_3,$$

where ρ_r and ρ_s are constants, then with probability at least $1 - n_{(1)}^{-10}$, $(\mathcal{Z}_\star, \mathcal{E}_\star)$, where $\mathcal{Z}_\star = \mathcal{A}^\dagger * \mathcal{L}_0$ and $\mathcal{E}_\star = \mathcal{E}_0$, is the unique optimal solution to problem (2) with $\lambda = 1/\sqrt{n_{(1)} n_3}$.

Theorem 3 shows that TLRR can exactly recover the low-rank clean data $\mathcal{L}_0$ even when the amount of noise is at the same order as the entry number of the observed tensor and the tubal rank is as high as $\mathcal{O}(n_{(2)}/\log(n_{(1)} n_3))$. Also the exact recovery does not require previously knowing the location of noise (the corrupted elements). This is very useful for data denoising in practice where noise is difficult to

detect. This theorem also implies that applying TLRR to recover clean data can also benefit subsequent data clustering: if the clean data $\mathcal{L}_0$ is exactly recovered, by Theorem 2 in Section 5.2 the optimal solution $\mathcal{Z}_\star$ would be tensor block-diagonal. Thus getting data clusters becomes straightforward.

The condition $\mathcal{P}_{\mathcal{U}_\mathcal{A}}(\mathcal{U}_0) = \mathcal{U}_0$ in Theorem 3 is necessary and reasonable—each authentic sample in $\mathcal{L}_0$ that is drawn from a particular subspace should be linearly representable by the bases in $\mathcal{A}$. The condition on the ranks of $\bar{A}^{(i)}$ $(i = 1, \ldots, n_3)$ is indispensable for exact recovery, as the equality $\mathcal{L}_0 = \mathcal{A} * \mathcal{Z}_\star = \mathcal{A} * \mathcal{A}^\dagger * \mathcal{L}_0$ implies $\mathcal{A} * \mathcal{A}^\dagger = \mathcal{I}$, requiring that the ranks of $\bar{A}^{(i)}$ $(i = 1, \ldots, n_3)$ are equal. This condition is also necessary for R-TPCA, as R-TPCA can be viewed as a special case of TLRR by choosing the identity tensor $\mathcal{I}$ as the dictionary, naturally satisfying the condition. Also if $\mathcal{A} = \mathcal{I}$, Theorem 3 still holds and it gives guarantees on exact recovery for R-TPCA.[1] Theorem 3 is also applicable when $n_3 = 1$. Thus performance guarantees for matrix LRR [3], [4] can be derived as a special case of our results.

The proof of Theorem 3 is given in Section 7 of supplementary, available online. It is worthy mentioning that the proof is carefully and elaborately conducted based on the interaction between both the original and Fourier domains. This can be intuitively interpreted from the fact that for the TLRR model (2), its equivalent form, namely

$$\min_{\mathcal{Z},\mathcal{E}} \frac{1}{n_3}\left(\|\bar{\mathcal{Z}}\|_* + \lambda\|\texttt{bcirc}(\mathcal{E})\|_1\right), \text{ s.t. } \mathcal{X} = \mathcal{A} * \mathcal{Z} + \mathcal{E}, \tag{12}$$

is a mixed model, as the low-rank term is performed on the Fourier domain while the sparse regularization is in the original domain. Interpreting the tensor nuclear norm of $\mathcal{Z}$ as a matrix nuclear norm of $\bar{Z}$ in the Fourier domain allows us to use some properties of matrix nuclear norm in the proof. But the analysis of the sparse term is still in the original domain, as the ℓ_1-norm has no equivalent form in the Fourier domain. So this requires us to finish the proof based on the interaction between both domains. Compared with matrix LRR [3], [4], our proof is more challenging, as 1) our proof stretches across two domains while the proof of LRR only involves the original domain; 2) the structures of $\bar{Z}$ and $\texttt{bcirc}(\mathcal{E})$ are more complicated than those in LRR (3), thus making the proof of TLRR harder than those in LRR. For instance, our proofs (e.g., in Lemma 9 of supplementary, available online) require to upper bound the norms of random tensors, which involves block circulant matrices and the Fourier transformation and needs to carefully consider their properties and structures. Note, these two factors also make the clustering performance analysis of TLRR in Section 5 and the optimization algorithm design and analysis in Section 7 more challenging than LRR.

6.3 Comparison Between TLRR and R-TPCA

Since both TLRR and R-TPCA [13] are based on the recently proposed tensor tubal rank and t-SVD [31] and can be used for data recovery, here we compare their recovery ability by analyzing their incoherence parameters.

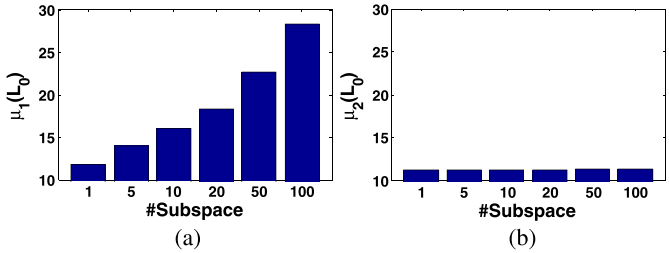


Fig. 5. Illustration of effects of the tensor subspace number k on $\mu_1(\mathcal{L}_0)$ and $\mu_2(\mathcal{L}_0)$. We produce a random tensor $\mathcal{L}_0 \in \mathbb{R}^{1000\times1000\times10}$ with $\text{rank}_\text{t}(\mathcal{L}_0) = 100$. We increase the subspace number k and set the sample number as $100/k$ per subspace (see case (a) in Section 9.2.1 for details of producing testing data).

We find with the increasing number of tensor subspaces in data, $\mu_1(\mathcal{L}_0)$ increases notably while $\mu_2(\mathcal{L}_0)$ is nearly constant. This observation coincides with the matrix case in [3]. We explain this below. Assume $\mathcal{L}_0$ obeys the assumptions on $\mathcal{X}$ in Theorem 2. Then $\mathcal{V}_0$ is tensor block-diagonal:

$$\mathcal{V}_0 = \begin{bmatrix} \mathcal{V}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathcal{V}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathcal{V}_k \end{bmatrix},$$

where $\mathcal{V}_i$ $(i = 1, \ldots, k)$ is of size $p_i \times m_i \times n_3$. When the tensor subspace number k is large, $\mathcal{V}_0$ would be very sparse and $\|\mathcal{V}_0^* * \mathfrak{e}_j^{n_2}\|_F^2 = \|\mathcal{V}_0(:, j, :)\|_F^2$ would be close to 1, leading to a large incoherence parameter $\mu_1(\mathcal{L}_0) \approx n_2 n_3/r$. In contrast, $\mathcal{U}_0$ is generally not block-diagonal and is indeed not sparse. Actually, when the rank of $\mathcal{L}_0$ is fixed, $\mathcal{U}_0$ is invariant to the tensor subspace number. This is because $\mathcal{U}_0$ just needs to span the tensor subspaces that samples lie in and only depends on the tensor rank of $\mathcal{L}_0$. So $\mu_2(\mathcal{L}_0)$ is nearly constant. This observation is also verified by empirical studies shown in Fig. 5. It also explains why T-RPCA is less successful when dealing with more tensor subspaces. For exact recovery, R-TPCA requires

$$\text{rank}_\text{t}(\mathcal{L}_0) \leq \frac{\rho_r' n_{(2)}}{\mu(\mathcal{L}_0)\left(\log\left(n_{(1)} n_3\right)\right)^2}.$$

Here ρ_r' is a constant, but $\mu(\mathcal{L}_0)$ $(\geq \mu_1(\mathcal{L}_0))$ is usually large.

Conversely, with the help of $\mathcal{A}$, the incoherence parameter $\mu_1^\mathcal{A}(\mathcal{L}_0)$ can be small. Though $\mu_1(\mathcal{L}_0)$ may become large when the tensor subspace number increases, $\mu_2(\mathcal{A})$ is constant and $r_\mathcal{A}/(n_1 n_3)$ remains small. Thus, Theorem 3 guarantees stronger data recovery power of TLRR over R-TPCA. Besides, R-TPCA requires an extra condition: $\mu_3(\mathcal{L}_0)$ should be sufficiently small. Such a restrictive condition is not necessary for TLRR. Finally, TLRR can not only exactly recover the clean data $\mathcal{L}_0$ but also learn the relationship among samples in $\mathcal{L}_0$, which benefit clustering and also other tasks, e.g., classification and metric learning.

7 Optimization Technique

Here we propose an efficient algorithm to solve (2), and then analyze its convergence behavior and computational complexity.

1. We derive the guarantees by adopting the least square proof framework instead of Golfing proof scheme [38] in R-TPCA due to the different dual certificates of TLRR and R-TPCA.

7.1 Optimization Algorithms

ADMM [22] is a straightforward optimization approach to solve problem (2). But when the sample number n_2 is large, directly applying ADMM is highly computationally expensive, as it requires to compute n_3 SVD over $n_2 \times n_2$ matrices at each iteration. To reduce the computational cost, we provide a new and equivalent reformulation for Eqn. (2). Assume $\mathcal{U}_{\mathcal{A}} * \mathcal{S}_{\mathcal{A}} * \mathcal{V}_{\mathcal{A}}^*$ is the skinny t-SVD of $\mathcal{A}$ (ref. Definition 2). Then, we respectively replace $\mathcal{A}$ and $\mathcal{Z}$ in Eqn. (2) with $\mathcal{D} = \mathcal{U}_{\mathcal{A}} * \mathcal{S}_{\mathcal{A}} \in \mathbb{R}^{n_1 \times r_{\mathcal{A}} \times n_3}$ and $\mathcal{V}_{\mathcal{A}} * \mathcal{Z}'$, where $r_{\mathcal{A}} = \text{rank}_{\text{t}}(\mathcal{A})$ and $\mathcal{Z}' \in \mathbb{R}^{r_{\mathcal{A}} \times n_2 \times n_3}$ is a variable needed to be optimized. This gives the following equivalent formulation:

$$\min_{\mathcal{Z}', \mathcal{E}} \ \|\mathcal{Z}'\|_* + \lambda \|\mathcal{E}\|_1, \ \text{s.t.} \ \mathcal{X} = \mathcal{D} * \mathcal{Z}' + \mathcal{E}. \tag{13}$$

Using such reformulation, we only need to compute SVD for matrices with a significantly reduced size of $r_{\mathcal{A}} \times n_2$ at each iteration. So we first solve Eqn. (13) to obtain its minimizer $(\mathcal{Z}'_*, \mathcal{E}_*)$ and then recover the minimizer $(\mathcal{V}_{\mathcal{A}} * \mathcal{Z}'_*, \mathcal{E}_*)$ to problem (2). The optimality of such a solution is guaranteed by Theorem 4.

Theorem 4. *Assume the pair $(\mathcal{Z}'_*, \mathcal{E}_*)$ is an optimal solution to problem (13). Then, the pair $(\mathcal{V}_{\mathcal{A}} * \mathcal{Z}'_*, \mathcal{E}_*)$ is the minimizer to problem (2).*

The proof of Theorem 4 can be found in Section 8 of supplementary, available online. To apply ADMM on the size-reduced problem (13), we first introduce one auxiliary variables $\mathcal{J}$ to decouple the variables from the objective and the constraint. Then one can update variables more easily. Problem (13) can be rewritten as

$$\min_{\mathcal{J}, \mathcal{Z}', \mathcal{E}} \ \|\mathcal{Z}'\|_* + \lambda \|\mathcal{E}\|_1, \ \text{s.t.} \ \mathcal{Z}' = \mathcal{J}, \mathcal{X} = \mathcal{D} * \mathcal{J} + \mathcal{E}. \tag{14}$$

To tackle the hard constraints, we resort to augmented Lagrangian multiplier method and solve the following problem instead:

$$H(\mathcal{J}, \mathcal{Z}', \mathcal{E}, \mathcal{Y}^1, \mathcal{Y}^2) = \|\mathcal{Z}'\|_* + \lambda \|\mathcal{E}\|_1 + \langle \mathcal{Y}^1, \mathcal{Z}' - \mathcal{J} \rangle + \frac{\beta}{2} \|\mathcal{Z}' - \mathcal{J}\|_F^2 + \langle \mathcal{Y}^2, \mathcal{X} - \mathcal{D} * \mathcal{J} - \mathcal{E} \rangle + \frac{\beta}{2} \|\mathcal{X} - \mathcal{D} * \mathcal{J} - \mathcal{E}\|_F^2,$$

where $\mathcal{Y}^1$ and $\mathcal{Y}^2$ are the Lagrange multipliers introduced for the two constraints respectively, and β is an auto-adjusted penalty parameter. Then we solve the problem through alternately updating two blocks, namely $\mathcal{J}$ and $(\mathcal{Z}', \mathcal{E})$, in each iteration to minimize $H(\mathcal{J}, \mathcal{Z}', \mathcal{E}, \mathcal{Y}^1, \mathcal{Y}^2)$ with other variables fixed. Algorithm 2 summarizes the whole optimization procedure. Both problems (15) for updating $\mathcal{J}_{k+1}$ and (16) for updating the block $(\mathcal{Z}'_{k+1}, \mathcal{E}_{k+1})$ have closed form solutions. Note, problem (16) can be split into subproblems for $\mathcal{Z}'$ and $\mathcal{E}$ as these two variables are independent in this problem. Accordingly, we update the variable $\mathcal{Z}'$ and $\mathcal{E}$ independently. See detailed optimization of $\mathcal{J}_{k+1}$ and $(\mathcal{Z}'_{k+1}, \mathcal{E}_{k+1})$ in Section 3 in supplementary, available online.

7.2 Convergence and Complexity Analysis

Since problem (14) is a convex problem which involves two blocks of variables, $\mathcal{J}$ and $(\mathcal{Z}', \mathcal{E})$, and only includes linear constraints, convergence analysis results in [22] guarantee the solution of Algorithm 2 would converge to the global optimum.

Algorithm 2. Tensor LRR (TLRR)

Input: Input $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, dictionary $\mathcal{A} \in \mathbb{R}^{n_1 \times n_4 \times n_3}$.
Initialize: $\mathcal{D}$=$\mathcal{U}_{\mathcal{A}} * \mathcal{S}_{\mathcal{A}}$ with skinny t-SVD $\mathcal{U}_{\mathcal{A}} * \mathcal{S}_{\mathcal{A}} * \mathcal{V}_{\mathcal{A}}^*$ of $\mathcal{A}$, $\mathcal{J}_0$=$\mathcal{Z}'_0$=$\mathcal{Y}^1_0$=$\mathbf{0}$, $\mathcal{E}_0$=$\mathcal{Y}^2_0$=$\mathbf{0}$, λ=$1/\sqrt{n_3 \max(n_1, n_2)}$, γ=1.1, β_0=$1e-5$, $\beta_{\max}$=$1e+8$, ϵ=$1e-8$, and k=0.
While not converged **do**

1. Fix $\mathcal{Z}'_k$ and $\mathcal{E}_k$. Update $\mathcal{J}_{k+1}$ by solving

$$\mathcal{J}_{k+1} = \arg\min_{\mathcal{J}} \left\| \mathcal{Z}'_k + \frac{\mathcal{Y}^1_k}{\beta_k} - \mathcal{J} \right\|_F^2 + \left\| \mathcal{X} - \mathcal{E}_k + \frac{\mathcal{Y}^2_k}{\beta_k} - \mathcal{D} * \mathcal{J} \right\|_F^2. \tag{15}$$

2. Fix $\mathcal{J}_{k+1}$. Update the block $(\mathcal{Z}', \mathcal{E})$ by solving

$$(\mathcal{Z}'_{k+1}, \mathcal{E}_{k+1}) = \arg\min_{\mathcal{Z}', \mathcal{E}} \|\mathcal{Z}'\|_* + \lambda \|\mathcal{E}\|_1 + \frac{\beta_k}{2} \left\| \mathcal{Z}' - \mathcal{J}_{k+1} + \frac{\mathcal{Y}^1_k}{\beta_k} \right\|_F^2 + \frac{\beta_k}{2} \left\| \mathcal{E} - \mathcal{X} + \mathcal{D} * \mathcal{J}_{k+1} - \frac{\mathcal{Y}^2_k}{\beta_k} \right\|_F^2. \tag{16}$$

3. Update Lagrange multipliers with $\mathcal{G}_{k+1}$=$\mathcal{D} * \mathcal{J}_{k+1}$+$\mathcal{E}_{k+1}$:

$$\mathcal{Y}^1_{k+1} = \mathcal{Y}^1_k + \beta_k (\mathcal{Z}'_{k+1} - \mathcal{J}_{k+1}), \quad \mathcal{Y}^2_{k+1} = \mathcal{Y}^2_k + \beta_k (\mathcal{X} - \mathcal{G}_{k+1}).$$

4. $\beta_{k+1} = \min(\gamma \beta_k, \beta_{\max})$.
5. Check the convergence conditions:

$$\max(\|\mathcal{J}_{k+1} - \mathcal{J}_k\|_\infty, \|\mathcal{Z}'_{k+1} - \mathcal{Z}'_k\|_\infty, \|\mathcal{E}_{k+1} - \mathcal{E}_k\|_\infty) \le \epsilon,$$
$$\max(\|\mathcal{J}_{k+1} - \mathcal{Z}'_{k+1}\|_\infty, \|\mathcal{X} - \mathcal{D} * \mathcal{J}_{k+1} - \mathcal{E}_{k+1}\|_\infty) \le \epsilon.$$

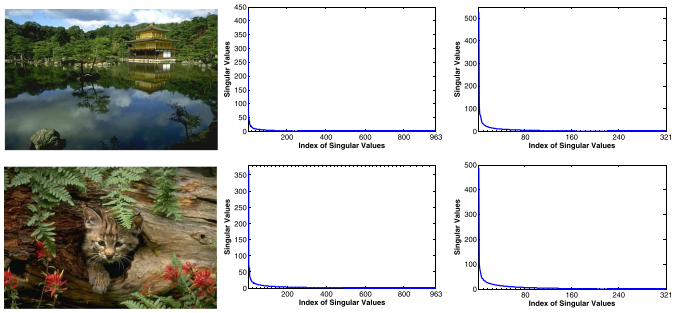
6. $k = k + 1$.

end while
Output: $\mathcal{Z}_* = \mathcal{V}_{\mathcal{A}} * \mathcal{Z}'_{k+1}$, $\mathcal{E}_* = \mathcal{E}_{k+1}$, $\mathcal{L}_* = \mathcal{X} - \mathcal{E}_*$.

At each iteration, when updating $\mathcal{J}_{k+1}$, computing its closed form solution to problem (15) costs $\mathcal{O}(r_{\mathcal{A}}(n_1 + n_2) n_3 \log(n_3). + r_{\mathcal{A}} n_1 n_2 n_3)$. The major cost of computing the closed form solution $(\mathcal{Z}'_{k+1}, \mathcal{E}_{k+1})$ to problem (22) includes n_3 SVD on $r_{\mathcal{A}} \times n_2$ matrices of cost $\mathcal{O}(r_{\mathcal{A}}^2 n_2 n_3)$ and tensor product of cost $\mathcal{O}(r_{\mathcal{A}} n_1 n_2 n_3 + r_{\mathcal{A}}(n_1 + n_2) n_3 \log(n_3))$. So the cost of Algorithm 2 is $\mathcal{O}(r_{\mathcal{A}} n_1 n_2 n_3 + r_{\mathcal{A}}(n_1 + n_2) n_3 \log(n_3))$ for each iteration. Compared with directly solving problem (2) whose iteration cost is $\mathcal{O}((n_1 + n_2) n_2^2 n_3)$, the reformulation Eqn. (13) reduces the cost significantly. Compared with R-TPCA [13] which is a counterpart of TLRR but without a dictionary, the convergence speed of TLRR is usually slower, since 1) TLRR has to update three variables ($\mathcal{J}$, $\mathcal{Z}'$ and $\mathcal{E}$) due to the dictionary instead of two variables (the clean data and the noise) in R-TPCA, and 2) empirically ADMM with more variables needs more iterations to achieve a certain optimization accuracy ϵ. But experiment results in Section 9 show that TLRR only runs a little slower than R-TPCA but provides much better clustering and recovery results.

8 Dictionary Construction

A qualified dictionary $\mathcal{A}$ is necessary in TLRR, as the bases in $\mathcal{A}$ should be able to linearly represent each authentic sample in the clean data $\mathcal{L}_0$. In this way, the clean data can be exactly recovered and the learnt relationship among



(a) (b) (c)

Fig. 6. Illustration of the low tubal rank property of the images in Berkeley Segmentation dataset. (a) Two randomly selected images. (b) Plots the singular values of $\bar{X}$ obtained by conducting linear transformation on the DFT result $\bar{\mathcal{X}}$ of image tensor $\mathcal{X}$. (c) Displays $\sum_{i=1}^{n_3} \boldsymbol{s}_i$, where $\boldsymbol{s}_i$ is the singular value vector (in a descending order) of the ith frontal slice $\bar{X}^{(i)}$ of $\bar{\mathcal{X}}$.

samples is accurate. Here we provide two different approaches to construct the dictionary $\mathcal{A}$.

The first one uses the raw tensor data $\mathcal{X}$ as the dictionary $\mathcal{A}$, which is similar to the strategy used by LRR [1]. We call this method *S-TLRR* with "S" denoting "simple". In this case, the learned representation $\mathcal{Z}_\star$ indicates the similarity between samples and can be used for clustering (see details in Section 5.2).

However, when the data are heavily corrupted, taking the contaminated data as the dictionary would harm the performance—the learned relationship among samples using a grossly corrupted dictionary would be far less accurate. This is also reflected by Theorem 3 that shows TLRR requires $\mathcal{P}_{\mathcal{U}_\mathcal{A}}(\mathcal{U}_0) = \mathcal{U}_0$ for exact recovery, i.e., the dictionary $\mathcal{A}$ and the clean samples should share their tensor subspaces. In this challenging scenario, we propose to use the estimation $\mathcal{L}'$ for clean data $\mathcal{L}_0$ from R-TPCA as the dictionary. Though R-TPCA hardly recovers $\mathcal{L}_0$ for a large number of subspaces (see Section 6.3), it can remove noise from $\mathcal{X}$ to some extent and provide a less noisy dictionary $\mathcal{L}'$ than $\mathcal{X}$. This method is termed as *Robust-TLRR* or *R-TLRR* in short.

Algorithm 3. Dictionary Construction

Input: Tensor data $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

1. Utilize R-TPCA to estimate $\mathcal{L}'$ with regularization parameter $\lambda = 1/\sqrt{n_3 \max(n_1, n_2)}$.
2. Estimate the tubal rank $r_{\mathcal{L}'}$ of $\mathcal{L}'$.
3. Truncate $\mathcal{L}'$ to obtain an approximation tensor $\mathcal{L}''$ which obeys $\text{rank}_\text{t}(\mathcal{L}'') \le r_{\mathcal{L}'}$:

$$\mathcal{L}'' = \arg\min_{\mathcal{L}} \|\mathcal{L} - \mathcal{L}'\|_F^2, \quad \text{s.t. } \text{rank}_\text{t}(\mathcal{L}'') \le r_{\mathcal{L}'}. \tag{17}$$

Output: dictionary $\mathcal{A} = \mathcal{L}''$.

Algorithm 3 describes the steps of using R-TPCA [13] to construct the dictionary $\mathcal{A}$. We first use R-TPCA to denoise data $\mathcal{X}$ for obtaining an estimation $\mathcal{L}'$ to the clean data $\mathcal{L}_0$, and then take a rank-truncated $\mathcal{L}'$ as the dictionary. As for Steps 2 and 3, we adopt the following strategies.

Step 2 For each frontal slice $(\bar{L}')^{(i)}$ of $\bar{\mathcal{L}}'$, we first compute the nonzero singular values $\{\sigma_1^i, \ldots, \sigma_{n_i}^i\}$ (in a descending order) of $(\bar{L}')^{(i)}$, and then let $\boldsymbol{c}_j^i = \sigma_j^i / \sigma_{j+1}^i$ $(1 \le j \le n_i - 1)$, $j_*^i = \arg\max_j c_j^i$ and $c_*^i = c_{j_*^i}^i$. If $(n_i - 1) c_*^i / \sum_{j \ne j_*^i} c_j^i < 10$ (need not truncation), then let $j_*^i = n_i$. Finally, we obtain the estimated tubal rank $r_{\mathcal{L}'} = \max(j_*^1, \cdots, j_*^{n_3})$.

Step 3 By Theorem 2.3.1 in [12], problem (17) in Algorithm 3 has closed form solution: $\mathcal{L}'' = \sum_{j=1}^{r_{\mathcal{L}'}} \mathcal{U}'(:, j, :) * \mathcal{S}'(j, j, :) * (\mathcal{V}'(:, j, :))^*$, where $\mathcal{U}' * \mathcal{S}' * (\mathcal{V}')^*$ is the t-SVD of $\mathcal{L}'$.

As Step 2 may reduce the estimated tubal rank of $\mathcal{L}'$, together with Step 3 the tubal rank $r_\mathcal{A}$ of the dictionary and consequently the coherence parameter $\mu_1^\mathcal{A}(\mathcal{L}_0)$ would be reduced. This will benefit R-TLRR for exactly recovering $\mathcal{L}_0$ with relatively higher rank (see conditions in Theorem 3).

The above dictionary construction method has another appealing property. In applications, we find that the constructed dictionary $\mathcal{A} = \mathcal{L}''$ usually satisfies the exact recovery condition in Theorem 3—the ranks of $\bar{A}^{(i)}$ $(i = 1, \ldots, n_3)$ are equal. The reason is that after Steps 2 and 3 in Algorithm 3, all frontal slices $(\bar{L}'')^{(i)}$ $(i = 1, \ldots, n_3)$ of the estimated data $\mathcal{L}''$ have the same rank since the ranks of $(\bar{L}'')^{(i)}$ $(i = 1, \ldots, n_3)$ are usually larger than the computed truncation rank $r_{\mathcal{L}'}$. To verify this, we randomly select two images (Fig. 6a) from the Berkeley Segmentation dataset [39] and plot the singular values of $\bar{X}$ in Fig. 6b. We observe that most of these singular values are very close to 0 and much smaller than the leading singular values. Also, by Definition 3, the tensor tubal rank $\text{rank}_\text{t}(\mathcal{X}) = \max(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_3})$, where $\boldsymbol{r}_i$ is the rank of $\bar{X}^{(i)}$. So we compute the singular value vector $\boldsymbol{s}_i$ (elements are in a descending order) of $\bar{X}^{(i)}$ and plot $\boldsymbol{v} = \sum_{i=1}^{n_3} \boldsymbol{s}_i$ in Fig. 6c. We find that the tubal rank of these images is indeed very low, since most values in $\boldsymbol{v}$ are almost zero. So by the truncation operation in Steps 2 and 3, the estimated tubal rank $r_{\mathcal{L}'}$ would be much smaller than the rank of all frontal slices $(\bar{L}'')^{(i)}$ $(i = 1, \ldots, n_3)$. Thus, the constructed dictionary $\mathcal{A} = \mathcal{L}''$ can obey the condition that the ranks of $\bar{A}^{(i)}$ $(i = 1, \ldots, n_3)$ are equal to each other. Accordingly, the exactly recovery performance can be guaranteed.

(a) Extended YaleB (b) FRGC 2.0 (c) FRDUE

Dataset	#Class	#Per class	#Total image	Size	Difficulty
YaleB	28	30	840	80×60	ill.
FRGC 2.0	60	20	1200	32×36	ill. and exp.
FRDUE	153	≈ 20	3059	22×25	def. and pos.

(d) Descriptions of the three datasets. ("ill.", "exp.", "def." and "pos." are short for "illumination", "expression", "deformation" and "pose", respectively.

Fig. 7. Experimental settings of the three testing datasets.

9 Experiments

We compare our S-TLRR and R-TLRR with state-of-the-arts for data recovery on both synthetic and real data. In all the experiments, we fix the regularization parameters of R-PCA [21], LRR [1] and R-LRR [3] (R-LRR uses the estimated data by R-PCA as its dictionary) as $1/\sqrt{\max(n_1', n_2')}$, where $n_1' \times n_2'$ is the data matrix size processed by them. For R-TPCA [13], S-TLRR and R-TLRR, the parameter takes

TABLE 2
Clustering Results (ACC, NMI, and PUR) and the Algorithm Running Time (in Seconds) on the Three Testing Databases

Dataset	Metric	LSA	LSR_1	LSR_2	SSC	EnSC	R-PCA	LRR	R-LRR	TLRRSC	R-TPCA	S-TLRR	R-TLRR
Extended YaleB	ACC	0.461	0.819	0.813	0.821	0.828	0.708	0.753	0.803	0.662	0.720	**0.845**	**0.873**
	NMI	0.660	0.882	0.876	0.885	0.890	0.820	0.871	0.888	0.787	0.832	**0.897**	**0.927**
	PUR	0.535	0.843	0.837	0.841	0.850	0.728	0.796	0.825	0.673	0.744	**0.857**	**0.877**
	Time	722.8	**0.22**	**0.26**	2027.4	42.3	1728.5	274.1	1813.6	728.2	200.7	555.3	697.6
FRGC 2.0	ACC	0.540	0.865	0.863	0.861	0.870	0.735	0.795	0.830	0.602	0.812	**0.891**	**0.911**
	NMI	0.772	0.927	0.921	0.924	0.933	0.873	0.901	0.922	0.841	0.906	**0.947**	**0.963**
	PUR	0.614	0.868	0.858	0.865	0.870	0.765	0.825	0.855	0.657	0.838	**0.910**	**0.929**
	Time	574.5	**0.24**	**0.25**	676.3	32.6	1611.4	304.7	1734.6	901.4	49.8	141.6	162.4
FRDUE (100 classes)	ACC	0.553	0.763	0.744	0.796	0.800	0.626	0.707	0.753	0.673	0.658	**0.825**	**0.846**
	NMI	0.769	0.918	0.911	0.921	0.920	0.821	0.884	0.920	0.858	0.837	**0.927**	**0.943**
	PUR	0.541	0.821	0.806	0.831	0.836	0.716	0.773	0.792	0.726	0.740	**0.852**	**0.874**
	Time	1350.2	**0.96**	**0.88**	844.9	16.5	354.06	190.0	397.1	568.2	36.6	117.9	147.2
FRDUE (all classes)	ACC	0.490	0.740	0.712	0.773	0.779	0.632	0.702	0.721	0.635	0.695	**0.785**	**0.818**
	NMI	0.678	0.900	0.895	0.905	0.907	0.814	0.830	0.894	0.817	0.880	**0.914**	**0.932**
	PUR	0.513	0.801	0.781	0.816	**0.829**	0.725	0.770	0.775	0.704	0.752	0.821	**0.852**
	Time	2926.6	**1.03**	**0.98**	1011.5	21.0	472.9	305.3	569.6	755.4	49.2	183.9	205.2

$1/\sqrt{\max(n_1, n_2)n_3}$. These parameter settings are provided by the authors [3], [13], [21] and our Theorem 3. We manually tune the parameters of other compared methods. The code is available at https://panzhous.github.io/.

9.1 Application to Data Clustering

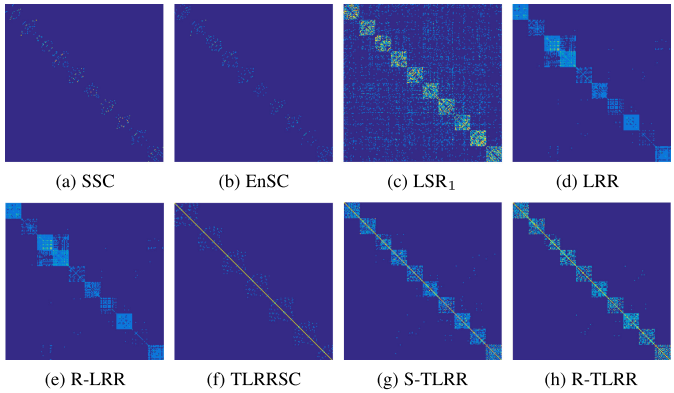
We first apply S-TLRR and R-TLRR for data clustering. See their detailed clustering steps in Algorithm 1. We evaluate them on face data, as most authentic face images approximately lie in a union of low-rank linear subspaces [40], [41], [42], [43]. Table 7 describes the testing datasets: Extended YaleB (uncropped) [44], FRGC 2.0 [37] and FRDUE.[2] For FRDUE, we respectively use its first 100 and all classes for testing. Here we compare our methods with other sparse and low-rank based methods, including LSA [45], LSR_1 [46], LSR_2 [46], SSC [2], EnSC [47], R-PCA [21], LRR [1], R-LRR [3], TLRRSC [15] and R-TPCA [13]. For our methods, we organize the images along the 2nd dimension. Such organization can well capture the linear representation relations among samples (see Section 5.3). We use three metrics: accuracy (ACC) [2], normalized mutual information (NMI) [48] and purity (PUR) [49], to evaluate the clustering performance. We run all the experiments for 20 times and report the average performance.

From the clustering results in Table 2, one can observe that R-TLRR always achieves the best clustering performance and S-TLRR also outperforms others in most cases. On the widely used ACC metric, R-TLRR respectively improves by 4.5, 4.1, 4.6 and 3.9 percent over the runner-up on the four testing cases (top-down). These results clearly prove the superior performance and robustness of our methods. These results come from the following advantages of our method. 1) S-TLRR and R-TLRR effectively exploit the multi-dimensional structure of the tensor data. In contrast, the matricization based methods directly unfold the tensor data along certain mode which would destroy the multi-way low-rank data structure and lead to degraded performance [12], [13]. 2) Unlike R-TPCA that assumes data are from a single tensor subspace, our methods consider the mixture structure in data (more consistent with reality) and learn more accurate relations among samples. Also, the results show that the robust versions, i.e., R-TLRR and R-LRR, usually outperform their counterparts, i.e., S-TLRR and LRR, since directly using the corrupted sample data as the dictionary would lead to inaccurate representation. TLRRSC [15] does not perform well, for the following three reasons. 1) It reshaps the 2-way face images into vectors and destroys the intrinsic multi-way data structure. 2) Randomly arranging the samples in the first $(k-1)$ modes may give non-low-rank tensor structure. 3) The Frobenius norm in TLRRSC can deal with Gaussian noise but cannot well handle the complex noise in face images [21]. Table 2 also reports the algorithm running time. Note, the computational time of R-TLRR contains the time cost for dictionary construction and solving the TLRR problem (13). Besides, R-TLRR is always much faster than its counter-part R-LRR, and S-TLRR runs faster than LRR in most cases. Though LSR_1, LSR_2, and EnSC are also efficient, tuning their critical regularization parameters requires significant additional effort.

We further display the block-diagonal structures learned by some compared methods in Fig. 8. For expediently plotting the representation tensors in our methods, we simply plot $\widehat{\mathcal{Z}}$ defined in Eqn. (8). Due to space limit, we only display the coefficients among the first 10 classes in FRGC 2.0, e.g., $\widehat{\mathcal{Z}}(1{:}200, 1{:}200)$ in our methods. We can observe better grouping effects of S-TLRR and R-TLRR over others. It also helps explain the superiority of our methods and reasonability of the tensor subspace assumption—the block-diagonal structures learned by our methods coincide with Theorem 2.

To verify that TLRR can clean and cluster data simultaneously, we use FRGC 2.0 for evaluation. Here we use its original images of sizes 72×64. For each image, we respectively randomly set $0 \sim 35$ percent of pixels to random values in [0,255] and then apply these compared methods. From Fig. 9, we find under different noise ratios, R-TLRR

2. http://cswww.essex.ac.uk/mv/allfaces/



(a) SSC (b) EnSC (c) LSR_1 (d) LRR

(e) R-LRR (f) TLRRSC (g) S-TLRR (h) R-TLRR

Fig. 8. Comparison of block-diagonal structures learned by the compared methods. *Best viewed in color pdf file.*

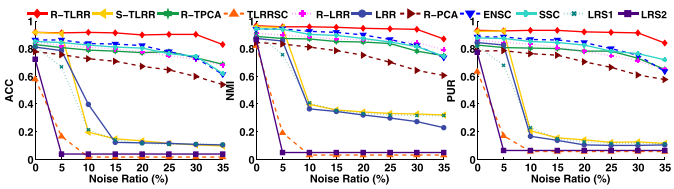


Fig. 9. Clustering results (ACC, NMI, and PUR) on noisy FRGC 2.0. *Best viewed in* $\times 2$ *sized color pdf file.*

always achieves the best results, while S-TLRR cannot work well with high noise ratio. This is because R-TLRR uses a more qualified dictionary and Theorem 3 guarantees the exact recovery of TLRR when the noise ratio is relatively low and the dictionary is sufficiently good. We also compute the error $\texttt{err} = \|\mathcal{L}_\star^0 - \mathcal{L}_\star\|_F/(n_1n_2n_3)$ for performance measure, where $\mathcal{L}_\star^0$ is the recovered data by R-TLRR without corruption and $\mathcal{L}_\star$ denotes the estimated clean data by R-TLRR with sparse noise ($5 \sim 35$ percent). From Fig. 10, we can see err is very small, implying that R-TLRR can recover the low-rank clean data from noisy cases. So the learned representations under different noise ratios are similar, providing almost the same performance.

9.2 Application to Data Recovery

9.2.1 Application to Synthetic Data Recovery

Here we evaluate TLRR and compare it with R-TPCA. R-TLRR uses the recovered data by R-TPCA as the dictionary $\mathcal{A}$ which (approximatively) meets the condition $\mathcal{P}_{\mathcal{U}_0}(\mathcal{U}_\mathcal{A}) = \mathcal{U}_\mathcal{A}$, while S-TLRR does not. So we verify Theorem 3 on R-TLRR.

We generate $\mathcal{L}_0$ and $\mathcal{E}_0$ as follows. We produce 6 random low-rank tensors $\{\mathcal{L}_1, \ldots, \mathcal{L}_6\}$ with $\mathcal{L}_i = \mathcal{B}_i * \mathcal{C}_i$, where the entries of $\mathcal{B}_i \in \mathbb{R}^{n_1 \times r_i \times n_3}$ and $\mathcal{C}_i \in \mathbb{R}^{r_i \times m_i \times n_3}$ are from i.i.d. $\mathcal{N}(0,1)$. So $\mathcal{L}_i$ is a low-rank tensor of m_i samples and tubal rank r_i. Then let $\mathcal{L}_0 = [\mathcal{L}_1, \ldots, \mathcal{L}_6] \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, where $n_2 = \sum_{i=1}^6 m_i$ and $\text{rank}_\text{t}(\mathcal{L}_0) = \sum_{i=1}^6 r_i$. As for $\mathcal{E}_0$, its support set Ω is chosen uniformly at random. We test two kinds of noise: (a) similar to [3], we normalize the values of entries in $\mathcal{L}_0$ such that $\|\mathcal{L}_0\|_\infty = 1$ and i.i.d. produce the noise in $\mathcal{E}_0$ as ± 1 with probability 0.5; (b) $\mathcal{L}_0$ is not normalized and the noise in $\mathcal{E}_0$ is also drawn from i.i.d. $\mathcal{N}(0,1)$. For simplicity, we set $n_1 = 240$, $n_3 = 20$, $m_i = 500$ and $r_i = r$ which varies from 1 to 24. The fraction $\rho = |\Omega|/(n_1n_2n_3)$ ranges from 2.5 to 65 percent with increment 2.5 percent. Similar to [3], for each pair (ρ, r), we simulate 20 test instances and declare a trial successful if the recovered $\mathcal{L}_\star$ obeys $\|\mathcal{L}_\star - \mathcal{L}_0\|_F/\|\mathcal{L}_0\|_F \leq 0.05$.

(a) Corrupted images (b) Recovered images in (a) by R-TLRR

Noise	5%	10%	15%	20%	25%	30%	35%
err	2.1^{-6}	3.0^{-6}	4.0^{-6}	5.0^{-6}	6.1^{-6}	7.5^{-6}	3.9^{-5}

(c) Numerical recovery performance on all images.

Fig. 10. Recovery performance of R-TLRR on FRGC 2.0. (a) Images of 0, 10, 20, and 30 percent corruptions (from left to right). (b) Recovered images in (a) by R-TLRR (from left to right).

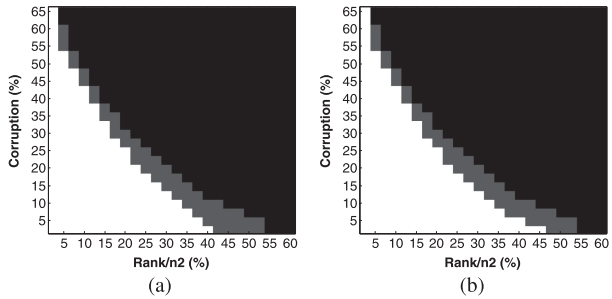


Fig. 11. Comparison between R-TPCA and R-TLRR. White Region: Both R-TPCA and R-TLRR succeed. Gray Regions: R-TLRR succeeds while R-TPCA fails. Black Regions: Both R-TPCA and R-TLRR fail. (a) Normalize $\mathcal{L}_0$ such that $\|\mathcal{L}_0\|_\infty = 1$ and i.i.d. produce ± 1 noise in $\mathcal{E}_0$ with probability 0.5. (b) Produce i.i.d. $\mathcal{N}(0,1)$ noise in $\mathcal{E}_0$.

Fig. 11 reports the experimental results. In both cases, R-TLRR exactly recovers the clean data (gray and block areas) when the tubal rank $\text{rank}_\text{t}(\mathcal{L}_0)$ is relatively low and the ratio ρ of noise is small, as implied by Theorem 3. Also, R-TLRR outperforms R-TPCA, as there are some cases (gray areas) that R-TLRR succeeds while R-TPCA fails. This coincides with the conclusion in Section 6.3 that R-TLRR has a stronger recovery ability than R-TPCA. The results also show validity of the dictionary built by T-RPCA.

9.2.2 Application to Image/Video Denoising

Here we evaluate the denoising performance of R-TLRR. As analyzed in Section 9.2.1, S-TLRR uses corrupted data as its dictionary and cannot be applied for exact recovery. We use the Berkeley segmentation dataset [39] and YUV video sequences[3] for testing. Berkeley dataset contains 200 color images of various natural scenes. The YUV dataset includes 26 videos. See details in Section 2 of supplementary, available online. In the experiments, we organize the color images along the channel direction to form a $w \times n \times h$ tensor with images size $w \times h$ and channel number n. Similarly, for videos, $w \times h$ and n respectively represent the frame size and frame number. We use the peak signal-to-noise ratio (PSNR) to evaluate the denoising performance:

$$\text{PSNR} = 10 \log_{10}\left(n_1n_2n_3\|\mathcal{L}_0\|_\infty^2/\|\mathcal{L}_\star - \mathcal{L}_0\|_F^2\right),$$

where $\mathcal{L}_\star$ is the recovered tensor of $\mathcal{L}_0 \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

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

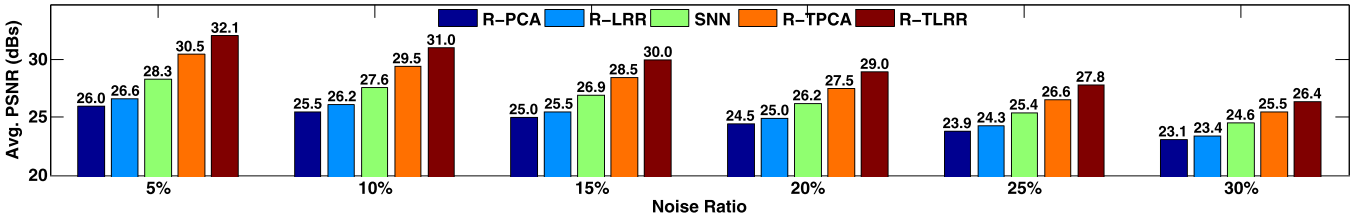


Fig. 12. Comparison of the image denoising performance. We apply the compared methods to recover the 200 images corrupted by 5 $\sim$ 30 percent noise in Berkeley dataset and report the average PSNR values on the 200 images. *Best viewed in* $\times 2$ *sized color pdf file.*

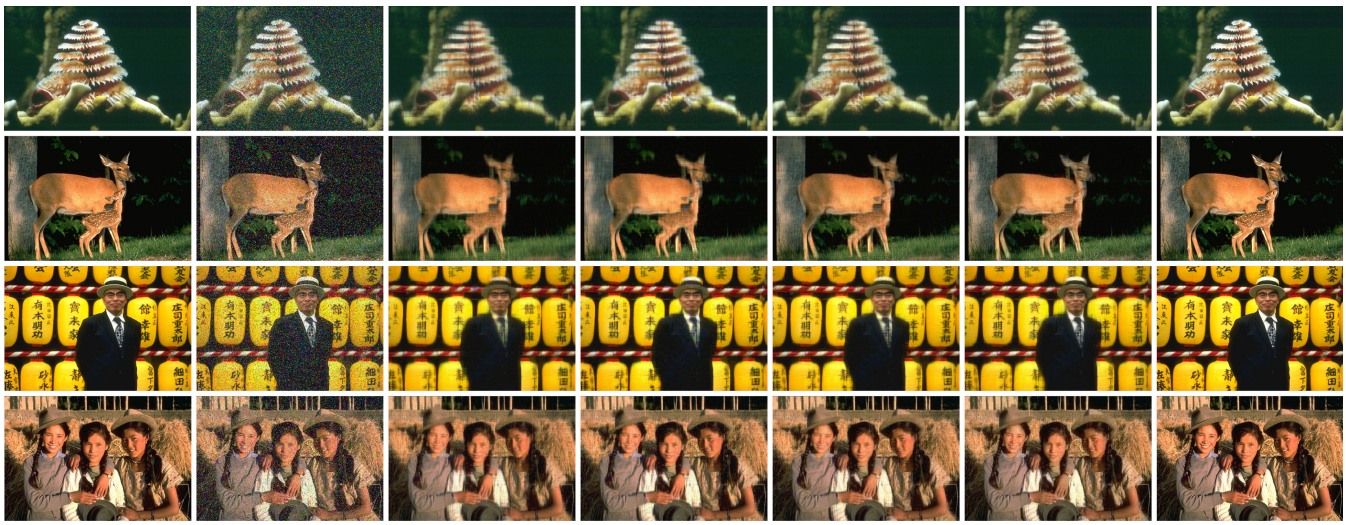
Our method and other low-rank based methods can be applicable for image/video denoising. Many works [3], [13], [21] validated that image and video data can be well approximated by low-rank matrices/tensors. Moreover, from the previous result in Fig. 6, the images in Berkeley dataset have low tubal rank structure. See details at the end of Section 8. So low-rank based methods are applicable to them. The low tubal rank structure of the testing video data is also notable but is not revealed here due to the similar results as images.

Image Denoising. For each testing image, we randomly set 5 $\sim$ 30 percent of pixels to random values in [0,255] and then apply these compared methods to recover it. The corrupted locations are unknown for these compared methods.

Fig. 12 reports the average PSNR values (on top of each bar) achieved by the compared methods. R-TLRR always performs the best and it improves the average PSNR values by about 1.5 dBs over the runner-up under different noise ratios. Besides, R-TLRR mostly outperforms others on every image. For instance, when the noise ratio is 20 percent, R-TLRR makes at least 2.0, 1.5, 1.0 and 0.5 dBs improvements than the second best on 41, 105, 174, and 194 images, respectively. See more details in Section 2 in supplementary, available online.

Fig. 13 displays the denoising results with their PSNR values when the noise ratio is 20 percent. R-TLRR performs much better the others. It preserves more details. For instance, it well recovers the contours of hydrophyte leaves and the spots of deers. R-TLRR improves by at least 2.4 dB over the second best R-TPCA on the testing images. As R-PCA and LRR recover the R, G and B channels separately and do not fully utilize the structure information, their performance is worse than the tensor based methods. Matricization based methods, e.g., SNN may destroy the data structure and lose optimality of the low-rank property [11], [12]. Conversely, our R-TLRR avoids the low-rank structure information loss [12], [13], hence giving better performance. R-TLRR also outperforms R-TPCA, according with Theorem 3 in Section 6.3: when given a qualified dictionary, R-TLRR has a stronger recovery guarantee than R-TPCA. This also demonstrates that the dictionary pursued by R-TPCA is qualified.

Video Denoising. We also randomly set 5 $\sim$ 30 percent of pixels in each video sequence to random values in [0,255]. From the denoising results in Fig. 14, one can see that our R-TLRR always outperforms other methods and it respectively improves the average PSNR values on the 26 videos by about 1.9, 1.8, 1.7, 1.5, 1.3 and 0.8 dBs over the second best method, i.e., R-TPCA, for the six noise ratios. Moreover, tensor based methods, i.e., SNN, R-TPCA and our method, outperform matrix based methods, i.e., RPCA and LRR. Table 3 reports the PSNR values under noise ratio 20 percent. R-TLRR achieves the best denoising results on all testing videos.



(a) Original image (b) Corrupted image (c) R-PCA (d) R-LRR (e) SNN (f) R-TPCA (g) R-TLRR

Image	R-PCA	R-LRR	SNN	R-TPCA	R-TLRR	Image	R-PCA	R-LRR	SNN	R-TPCA	R-TLRR
Hydrophyte	23.9	25.0	25.8	26.6	**29.5**	Deers	24.1	24.8	25.4	26.0	**28.5**
Man	22.2	23.7	23.6	23.7	**26.1**	Girls	22.5	23.4	24.6	25.1	**27.7**

(h) PSNR values achieved by the compared methods on the above four images.

Fig. 13. Examples of image denoising under noise ratio 20 percent. (a) Original image. (b) Corrupted image. (c)-(g) are the recovered results by the compared methods. (h) PSNR values on the above six images. *Best viewed in* $\times 2$ *sized color pdf file.*

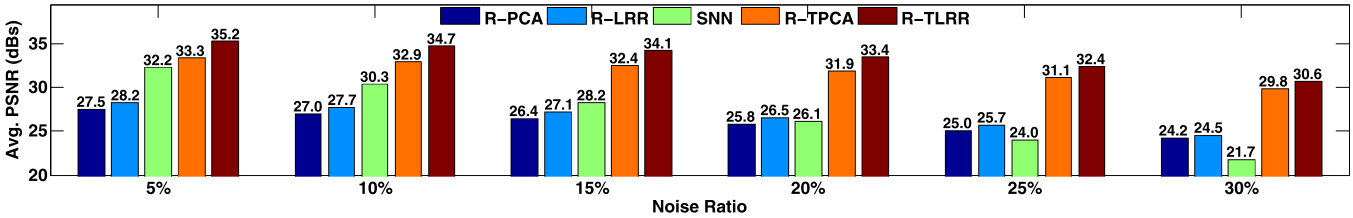


Fig. 14. Comparison of the video denoising performance. We apply the compared methods to recover the 26 videos corrupted by 5 $\sim$ 30 percent noise in YUV dataset and report the average PSNR values on the 26 videos. *Best viewed in* $\times 2$ *sized color pdf file.*

10 Conclusion

In this paper, we proposed a new tensor low-rank representation method for tensor data denoising and clustering. TLRR takes advantage of the multi-dimensional structure in tensor data and directly performs the low-rank representation on raw tensor data. In this way, it avoids destroying the tensor structure like other matrix based methods (e.g., LRR) and better preserves the low-rank structure. Unlike R-TPCA, TLRR considers the mixture structure in data, more consistent with practical data distribution, and thus obtains better performance. We further prove the tensor block-diagonal property of the optimal solution to the TLRR problem and analyze the exact recovery ability of TLRR theoretically. By comparison, when equipped with a qualified dictionary, TLRR has stronger recovery power than R-TPCA. Finally, we develop two variants of TLRR, i.e., S-TLRR and R-TLRR with different dictionary construction strategies. Extensive data clustering and recovery experiments testify the superiority of our methods.

Acknowledgments

Jiashi Feng was partially supported by NUS IDS R-263- 000-C67-646, ECRA R-263-000-C87-133, MOE Tier-II R-263-000-D17-112 and AI.SG R-263-000-D97-490. Z. Lin is supported by NSF China (grant no.s 61625301 and 61731018), Major Scientific Research Project of Zhejiang Lab (grant no. 2019KB0AC01), Zhejiang Lab (grant no. 2019KB0AB02), and Beijing Academy of Artificial Intelligence.

TABLE 3
PSNR Values on the YUV Video Sequences

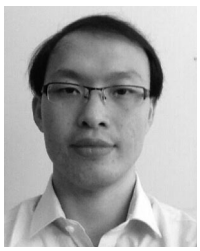
ID	Video name	R-PCA	R-LRR	SNN	R-TPCA	R-TLRR
1	Akiyo	26.2	26.6	30.24	34.8	**36.5**
2	Big Buck Bunny	33.2	33.3	20.9	34.9	**36.5**
3	Bridge (close)	25.8	26.3	29.7	36.2	**37.6**
4	Bridge (far)	30.7	30.8	35.6	43.3	**44.1**
5	Bus	23.2	24.2	17.1	23.3	**23.9**
6	Carphone	24.6	25.9	28.3	30.7	**32.4**
7	Claire	26.9	28.5	32.4	36.0	**37.6**
8	Coastguard	25.0	25.7	25.4	27.7	**28.5**
9	Container	23.0	23.9	27.8	36.4	**38.3**
10	Elephants Dream	33.8	34.0	22.2	37.7	**40.7**
11	Flower	22.6	22.7	22.3	23.6	**24.7**
12	Foreman	25.1	25.7	26.2	29.6	**31.4**
13	Grandma	25.2	26.5	31.3	37.2	**38.8**
14	Hall	23.9	24.7	28.1	31.6	**33.2**
15	Highway	27.7	27.7	28.7	31.0	**31.8**
16	Miss America	27.3	29.2	31.3	34.1	**36.1**
17	Mobile	20.4	20.7	21.2	26.1	**27.7**
18	Mother Daughter	28.0	28.5	31.1	34.4	**36.0**
19	News	22.7	24.2	26.9	30.3	**32.6**
20	Paris	21.4	22.6	17.8	26.9	**29.0**
21	Salesman	23.6	24.6	27.8	32.6	**34.4**
22	Silent	27.0	27.8	29.9	32.6	**34.1**
23	Stefan	22.2	23.3	18.4	23.4	**23.8**
24	Suzie	28.7	29.6	29.6	30.9	**32.8**
25	Tempete	23.6	24.3	17.9	27.6	**28.9**
26	Waterfall	27.7	28.3	20.7	35.6	**37.4**

References

[1] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 663–670.
[2] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, Nov. 2013.
[3] G. Liu, Q. Liu, and P. Li, "Blessing of dimensionality: Recovering mixture data via dictionary pursuit," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, Jan. 2016.
[4] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, Jan. 2013.
[5] Y. Cui, C. Zheng, and J. Yang, "Identifying subspace gene clusters from microarray data using low-rank representation," *PloS One*, vol. 8, no. 3, 2013, Art. no. e59377.
[6] C. Lang, G. Liu, J. Yu, and S. Yan, "Saliency detection by multitask sparsity pursuit," *IEEE Trans. Image Process.*, vol. 21, no. 3, pp. 1327–1338, Mar. 2012.
[7] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Low-rank sparse learning for robust visual tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 470–484.
[8] G. Liu and S. Yan, "Latent low-rank representation for subspace segmentation and feature extraction," in *Proc. IEEE Conf. Comput. Vis.*, 2011, pp. 1615–1622.
[9] M. Yin, J. Gao, and Z. Lin, "Laplacian regularized low-rank representation and its applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 3, pp. 504–517, Mar. 2016.
[10] 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 J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 148–172, 2013.
[11] 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.*, 2013, pp. 73–81.
[12] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer, "Novel methods for multilinear data completion and de-noising based on tensor-SVD," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3842–3849.
[13] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5249–5257.
[14] B. Huang, C. Mu, D. Goldfarb, and J. Wright, "Provable low-rank tensor recovery," *Optim. Online*, vol. 4252, 2014, Art. no. 2.
[15] Y. Fu, J. Gao, D. Tien, Z. Lin, and H. Xia, "Tensor LRR and sparse coding-based subspace clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 10, pp. 2120–2133, Oct. 2016.
[16] L. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.
[17] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," in *Proc. Conf. Neutral Inf. Process. Syst.*, 2007, pp. 801–808.
[18] C. Zhang, H. Fu, S. Liu, G. Liu, and X. Cao, "Low-rank tensor constrained multiview subspace clustering," in *Proc. IEEE Conf. Comput. Vis.*, 2016, pp. 1582–1590.
[19] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[21] E. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *J. ACM*, vol. 58, no. 3, 2011, Art. no. 11.
[22] Z. Lin, R. Liu, and Z. Su, "Linearized alternating direction method with adaptive penalty for low-rank representation," in *Proc. Conf. Neutral Inf. Process. Syst.*, 2011, pp. 612–620.
[23] H. Kiers, "Towards a standardized notation and terminology in multiway analysis," *J. Chemometrics*, vol. 14, no. 3, pp. 105–122, 2000.
[24] C. Hillar and L. Lim, "Most tensor problems are NP-hard," *J. ACM*, vol. 60, no. 6, 2013, Art. no. 45.
[25] J. Landsberg, *Tensors: Geometry and Applications*, Providence, RI, USA: American Mathematical Society, 2012.
[26] 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, Jan. 2013.
[27] 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.
[28] B. Romera-Paredes and M. Pontil, "A new convex relaxation for tensor completion," *Proc. Conf. Neutral Inf. Process. Syst.*, pp. 2967–2975, 2013.
[29] Z. Song, D. P. Woodruff, and P. Zhong, "Relative error tensor low rank approximation," in *Proc. Conf. Symp. Discrete Algorithms*, pp. 2772–2789, 2019.
[30] M. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *Proc. Nat. Acad. Sci.*, vol. 106, no. 3, pp. 697–702, 2009.
[31] M. Kilmer and C. Martin, "Factorization strategies for third-order tensors," *Linear Algebra Appl.*, vol. 435, no. 3, pp. 641–658, 2011.
[32] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.
[33] P. Zhou and J. Feng, "Outlier-robust tensor PCA," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1–9.
[34] 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. 35, no. 1, pp. 208–220, Jan. 2018.
[35] P. Zhou, C. Lu, Z. Lin, and C. Zhang, "Tensor factorization for low-rank tensor completion," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1152–1163, Mar. 2017.
[36] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[37] P. J. Phillips *et al.*, "Overview of the face recognition grand challenge," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 947–954.
[38] D. Gross, "Recovering low-rank matrices from few coefficients in any basis," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1548–1566, Mar. 2011.
[39] 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. IEEE Int. Conf. Comput. Vis.*, 2001, pp. 416–423.
[40] P. Zhou, Z. Lin, and C. Zhang, "Integrated low-rank-based discriminative feature learning for recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 5, pp. 1080–1093, May 2016.
[41] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.
[42] P. Zhou, C. Fang, Z. Lin, C. Zhang, and E. Chang, "Dictionary learning with structured noise," *Neurocomputing*, vol. 273, pp. 414–423, 2017.
[43] P. Zhou, C. Zhang, and Z. Lin, "Bilevel model based discriminative dictionary learning for recognition," *IEEE Trans. Image Process.*, vol. 26, no. 3, pp. 1173–1187, Mar. 2017.
[44] A. Georghiades, P. Belhumeur, and D. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 643–660, Jun. 2001.
[45] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 94–106.
[46] C. Lu, H. Min, Z. Zhao, L. Zhu, D. Huang, and S. Yan, "Robust and efficient subspace segmentation via least squares regression," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 347–360.
[47] C. You, C. Li, D. Robinson, and R. Vidal, "Oracle based active set algorithm for scalable elastic net subspace clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3928–3937.
[48] N. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, 2010.
[49] C. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

Pan Zhou received master's degree in computer science from Peking University, in 2016. Now, he is working toward the PhD degree in the Department of Electrical and Computer Engineering (ECE), National University of Singapore, Singapore. His research interests include computer vision, machine learning, and optimization. He was the winner of the Microsoft Research Asia Fellowship in 2018.



Canyi Lu received the PhD degree from the National University of Singapore, in 2017. He is currently a postdoctoral research associate with Carnegie Mellon University. His current research interests include computer vision, machine learning, pattern recognition, and optimization. He was the winner of the Microsoft Research Asia Fellowship in 2014. He is a member of the IEEE.

Jiashi Feng received the PhD degree from the National University of Singapore (NUS), in 2014. He was a postdoctoral research follow with the University of California at Berkeley, Berkeley. He joined NUS as a faculty member, where he is currently an assistant professor with the Department of Electrical and Computer Engineering. His research areas include computer vision, machine learning, robust learning, and deep learning.

Zhouchen Lin (M'00-SM'08-F'18) is currently a professor with the School of Electronics Engineering and Computer Science, Peking University. His research interests include computer vision, image processing, machine learning, pattern recognition, and numerical optimization. He is an area chair of CVPR 2014/2016/2019/2020, ICCV 2015, NIPS 2015/2018/2019, ICML 2020, AAAI 2019/2020, and IJCAI 2020, and a senior program committee member of AAAI 2016/2017/2018, and IJCAI 2016/2018. He is an associate editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence and the International Journal of Computer Vision. He is a fellow of the IAPR and the IEEE.

Shuicheng Yan is a chief technology officer of YITU Tech Company, and also the dean's chair associate professor with the National University of Singapore. His research areas include machine learning, computer vision and multimedia, and he has authored/co-authored hundreds of technical papers over a wide range of research topics, with a Google Scholar citation more than 20,000 times and H-index 66. He is ISI Highly-cited researcher of 2014, 2015, and 2016, respectively. His team received seven times winner or honorable-mention prizes in PASCAL VOC and ILSVRC competitions, along with more than 10 times best (student) paper prizes. He is also an IAPR fellow and a fellow of the IEEE.