SIAM J. Imaging Sciences
Vol. 17, No. 1, pp. 225–247

Robust Tensor CUR Decompositions: Rapid Low-Tucker-Rank Tensor Recovery with Sparse Corruptions*

HanQin Cai†, Zehan Chao‡, Longxiu Huang§, and Deanna Needell‡

Abstract. We study the tensor robust principal component analysis (TRPCA) problem, a tensorial extension of matrix robust principal component analysis, which aims to split the given tensor into an underlying low-rank component and a sparse outlier component. This work proposes a fast algorithm, called robust tensor CUR decompositions (RTCUR), for large-scale nonconvex TRPCA problems under the Tucker rank setting. RTCUR is developed within a framework of alternating projections that projects between the set of low-rank tensors and the set of sparse tensors. We utilize the recently developed tensor CUR decomposition to substantially reduce the computational complexity in each projection. In addition, we develop four variants of RTCUR for different application settings. We demonstrate the effectiveness and computational advantages of RTCUR against state-of-the-art methods on both synthetic and real-world datasets.

Key words. tensor CUR decomposition, robust tensor principal component analysis, low-rank tensor recovery, outlier detection

MSC codes. 68Q25, 68W25, 68W20, 68P20

DOI. 10.1137/23M1574282

1. Introduction.

In our real world, high-dimensional data, such as images, videos, and DNA microarrays, often reside approximately on low-dimensional manifolds [38]. This association between low-dimensional manifolds and high-dimensional data has led mathematicians to develop various dimension reduction methods, such as principal component analysis (PCA) [1] and nonnegative matrix factorization [35] under the low-rank assumption. It becomes increasingly important to study the low-rank structures for data in many fields, such as image processing [20, 37, 40], video processing [42, 62], text analysis [18, 45], and recommendation systems [54, 60]. In reality, many higher-order data are naturally represented by tensors [39, 44, 61], which are higher-order extensions of matrices. The tensor-related tasks, such as tensor compression and tensor recovery, usually involve finding a low-rank structure from a given tensor.

*Received by the editors May 22, 2023; accepted for publication (in revised form) September 18, 2023; published electronically January 25, 2024. This paper is an extension of work originally presented in ICCVW [8]. The authors contributed equally.

https://doi.org/10.1137/23M1574282

Funding: The work of the authors was partially supported by National Science Foundation grants DMS-2011140, DMS-2108479, and DMS-2304489 and an AMS Simons Travel grant.

†Department of Statistics and Data Science and Department of Computer Science, University of Central Florida, Orlando, FL 32816 USA (hqcai@ucf.edu).

‡Department of Mathematics, University of California, Los Angeles, Los Angeles, CA 90095 USA (zchao@math.ucla.edu, deanna@math.ucla.edu).

§Department of Computational Mathematics, Science, and Engineering and Department of Mathematics, Michigan State University, East Lansing, MI 48823 USA (huangl3@msu.edu).

As in the matrix setting but to an even greater extent, PCA is one of the most widely used methods for such dimension reduction tasks. However, standard PCA is oversensitive to extreme outliers [15]. To overcome this weakness, robust principal component analysis (RPCA) has been proposed to tolerate the sparse outliers in data analysis [16]. In particular, RPCA aims to reconstruct a low-rank matrix $\boldsymbol{L}^\star$ and a sparse outlier matrix $\boldsymbol{S}^\star$ from the corrupted observation

$$\boldsymbol{X} = \boldsymbol{L}^\star + \boldsymbol{S}^\star. \tag{1.1}$$

Existing studies on RPCA seek to find $\boldsymbol{L}^\star$ and $\boldsymbol{S}^\star$ by solving the following nonconvex problem [16], and we use $\boldsymbol{L}$ and $\boldsymbol{S}$ to denote the outcomes:

$$\begin{aligned} &\underset{\boldsymbol{L},\boldsymbol{S}}{\text{minimize}} \ \|\boldsymbol{X}-\boldsymbol{L}-\boldsymbol{S}\|_{\mathrm{F}} \\ &\text{subject to} \quad \boldsymbol{L} \text{ is low-rank and } \boldsymbol{S} \text{ is sparse.} \end{aligned} \tag{1.2}$$

The term *low-rank* refers to the constraint that the rank of $\boldsymbol{L}$ is much smaller than its size, and the term *sparse* refers to the restriction on the number of nonzero entries in $\boldsymbol{S}$ (for example, one possible restriction is to allow each column and row of $\boldsymbol{S}$ to contain at most 10% nonzero entries). This RPCA model has been widely studied [3, 5, 6, 7, 9, 11, 12, 28, 48] and applied to many applications, e.g., face modeling [57], feature identification [30], and video background subtraction [36]. However, the original RPCA method can only handle 2-mode arrays (i.e., matrices), while real-world data are often more naturally represented by higher-dimensional arrays (i.e., tensors). For instance, in the application of video background subtraction, a color video is automatically a 4-mode tensor (height, width, frame, and color). To apply RPCA to tensor data, one has to unfold the original tensor into a matrix along some specific mode(s). Although the upper bound of the unfolded matrix rank depends on the original tensor rank, the exact rank of the unfolded matrix remains unclear in some tensor rank settings. In addition, we seek methods that utilize the structural information of a tensor rather than ignoring the information. Therefore, it is important to generalize the standard RPCA to tensor settings. This task is called tensor robust principal component analysis (TRPCA) [43]. Moving from matrix PCA to the tensor setting could be challenging because some standard results known in the matrix case may not be generalized to tensors smoothly [21]. For example, the rank of a matrix is well-defined and uniquely defined, but researchers have proposed several definitions of tensor rank, such as Tucker rank [55], CP rank [17], and Tubal rank [59].

1.1. Notation and definitions.

A tensor is a multidimensional array, and its number of dimensions is called the *order* or *mode.* The space of real tensors of order n and of size $(d_1, \cdots, d_n)$ is denoted as $\mathbb{R}^{d_1 \times \cdots \times d_n}$. In this section, we first bring in the tensor-related notation and review some basic tensor properties which will be used throughout the rest of the paper. We denote tensors, matrices, vectors, and scalars in different typeface for clarity. More specifically, calligraphic capital letters (e.g., $\mathcal{X}$) are used for tensors, bold capital letters (e.g., $\boldsymbol{X}$) for matrices, bold lowercase letters (e.g., $\boldsymbol{x}$) for vectors, and regular letters (e.g., x) for scalars. We use $\boldsymbol{X}(I,:)$ and $\boldsymbol{X}(:,J)$ to denote the row and column submatrices of $\boldsymbol{X}$ with

index sets I and J, respectively. $\mathcal{X}(I_1, \cdots, I_n)$ denotes the subtensor of $\mathcal{X}$ with index sets I_k at the kth mode. A single element in a tensor is indexed as $\mathcal{X}_{i_1,\cdots,i_n}$. Moreover,

$$\|\mathcal{X}\|_\infty = \max_{i_1,\cdots,i_n} |\mathcal{X}_{i_1,\cdots,i_n}| \text{ and } \|\mathcal{X}\|_{\mathrm{F}} = \sqrt{\sum_{i_1,\cdots,i_n} \mathcal{X}^2_{i_1,\cdots,i_n}}$$

denote the max magnitude and Frobenius norm of a tensor, respectively. $\boldsymbol{X}^\dagger$ denotes the Moore–Penrose pseudoinverse of a matrix. The set of the first d natural numbers is denoted by $[d] := \{1, \cdots, d\}$.

Definition 1.1 (tensor matricization/unfolding). *An n-mode tensor $\mathcal{X}$ can be matricized, or reshaped into a matrix, in n ways by unfolding it along each of the n modes. The mode-k matricization/unfolding of tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_n}$ is the matrix denoted by*

$$\mathcal{X}_{(k)} \in \mathbb{R}^{d_k \times \prod_{j \neq k} d_j}, \tag{1.3}$$

whose columns are composed of all the vectors obtained from $\mathcal{X}$ by fixing all indices except for the kth dimension. The mapping $\mathcal{X} \mapsto \mathcal{X}_{(k)}$ is called the mode-k unfolding operator.

Definition 1.2 (mode-k product). *Let $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_n}$ and $\boldsymbol{A} \in \mathbb{R}^{J \times d_k}$. The kth-mode multiplication between $\mathcal{X}$ and $\boldsymbol{A}$ is denoted by $\mathcal{Y} = \mathcal{X} \times_k \boldsymbol{A}$, with*

$$\mathcal{Y}_{i_1,\cdots,i_{k-1},j,i_{k+1},\cdots,i_n} = \sum_{s=1}^{d_k} \mathcal{X}_{i_1,\cdots,i_{k-1},s,i_{k+1},\cdots,i_n} \boldsymbol{A}_{j,s}. \tag{1.4}$$

This can be written as a matrix product by noting that $\mathcal{Y}_{(k)} = \boldsymbol{A}\mathcal{X}_{(k)}$. If we have multiple tensor matrix products from different modes, then we use the notation $\mathcal{X} \times_{i=t}^{s} \boldsymbol{A}_i$ to denote the product $\mathcal{X} \times_t \boldsymbol{A}_t \times_{t+1} \cdots \times_s \boldsymbol{A}_s$. We also use "tensor-matrix product"to name this operation throughout our paper.

In tensor analysis, Tucker rank [29] (also known as multilinear rank) is one of the most essential tensor ranks related to subspace estimation, compression, and dimensionality reduction [51].

Definition 1.3 (tensor Tucker rank and Tucker decomposition). *The Tucker decomposition of tensor $\mathcal{X}$ is defined as an approximation of a core tensor $\mathcal{C}$ multiplied by n factor matrices $\boldsymbol{A}_k$ (whose columns are usually orthonormal) along each mode such that*

$$\mathcal{X} \approx \mathcal{C} \times_{i=1}^{n} \boldsymbol{A}_i. \tag{1.5}$$

If (1.5) becomes an equation and $\mathcal{C} \in \mathbb{R}^{r_1 \times \cdots \times r_n}$, then we say this decomposition is an exact Tucker decomposition of $\mathcal{X}$.

Note that higher-order singular value decomposition (HOSVD) [22] is a specific orthogonal Tucker decomposition which is popularly used in the literature.

1.2. Related work: Tensor CUR decompositions. Researchers have been actively studying CUR decompositions for matrices in recent years [24, 26]. For a matrix $\boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2}$, let $\boldsymbol{C}$ be a submatrix consisting of a subset of columns of $\boldsymbol{X}$ with column indices J, $\boldsymbol{R}$ be a

submatrix consisting of a subset of rows of $\boldsymbol{X}$ with row indices I, and $\boldsymbol{U} = \boldsymbol{X}(I, J)$. The theory of CUR decompositions states that $\boldsymbol{X} = \boldsymbol{C}\boldsymbol{U}^{\dagger}\boldsymbol{R}$ if $\operatorname{rank}(\boldsymbol{U}) = \operatorname{rank}(\boldsymbol{X})$. The first extension of CUR decompositions to tensors involved a single-mode unfolding of 3-mode tensors [46]. Later, [14] proposed a different variant of tensor CUR that accounts for all modes. Recently, [10] dubbed these decompositions with more descriptive monikers, namely, Fiber and Chidori CUR decompositions. In this paper, we will employ both Fiber CUR decomposition and Chidori CUR decomposition (see Figures 1 and 2 for illustration) to accelerate an essential step in the proposed algorithm. We state the Fiber CUR and Chidori CUR decomposition characterizations below for the reader's convenience.

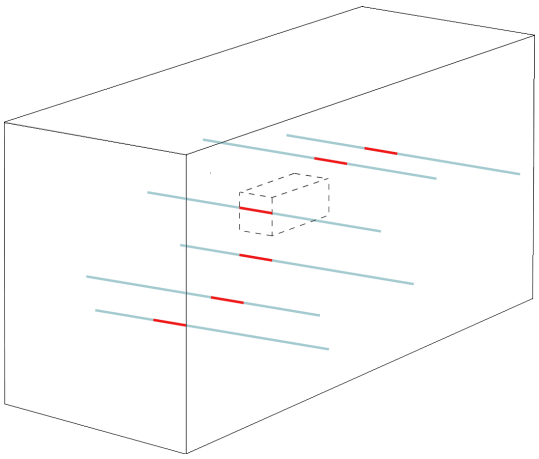

Figure 1. ([10]). *Illustration of the Fiber CUR decomposition of Theorem 1.4, in which J_i is not necessarily related to I_i. The lines correspond to rows of $\boldsymbol{C}_2$, and red indices correspond to rows of $\boldsymbol{U}_2$. Note that the lines may (but do not have to) pass through the core subtensor $\mathcal{R}$ outlined by dotted lines. For the figure's clarity, we do not show fibers in $\boldsymbol{C}_1$ and $\boldsymbol{C}_3$.*

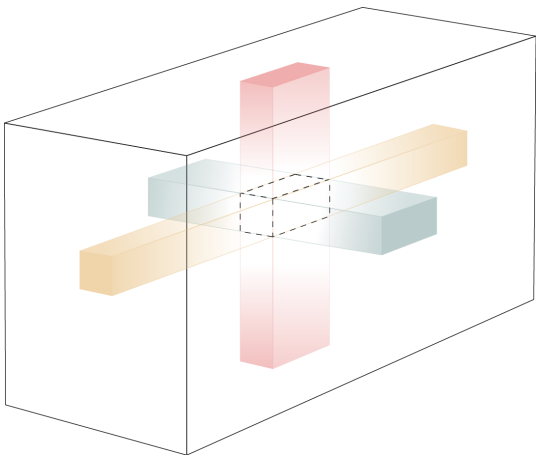

Figure 2. ([10]). *Illustration of Chidori CUR decomposition of a 3-mode tensor in the case when the indices I_i are each an interval and $J_i = \otimes_{j \neq i} I_j$ (see Theorem 1.4). The matrix $\boldsymbol{C}_1$ is obtained by unfolding the red subtensor along mode 1, $\boldsymbol{C}_2$ by unfolding the green subtensor along mode 2, and $\boldsymbol{C}_3$ by unfolding the yellow subtensor along mode 3. The dotted line shows the boundaries of $\mathcal{R}$. In this case, $\boldsymbol{U}_i = \mathcal{R}_{(i)}$ for all i.*

Theorem 1.4 (see [10, Theorem 3.3]). *Let $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_n}$ with Tucker rank $(r_1, \ldots, r_n)$. Let $I_i \subseteq [d_i]$ and $J_i \subseteq [\prod_{j \neq i} d_j]$. Set $\mathcal{R} = \mathcal{A}(I_1, \cdots, I_n)$, $\boldsymbol{C}_i = \mathcal{A}_{(i)}(:, J_i)$, and $\boldsymbol{U}_i = \boldsymbol{C}_i(I_i, :)$. Then the following statements are equivalent:*

(i) $\mathcal{A} = \mathcal{R} \times_{i=1}^{n} (\boldsymbol{C}_i \boldsymbol{U}_i^{\dagger})$;
(ii) $\operatorname{rank}(\boldsymbol{U}_i) = r_i$;
(iii) $\operatorname{rank}(\boldsymbol{C}_i) = r_i$ *for all* i, *and the Tucker rank of* $\mathcal{R}$ *is* $(r_1, \cdots, r_n)$.

Remark 1.5. In particular, when J_i are sampled independently from I_i, Theorem 1.4(i) is called Fiber CUR decomposition. When $J_i = \otimes_{j \neq i} I_j$, Theorem 1.4(i) is called Chidori CUR decomposition.

In addition, according to [27, Corollary 5.2], if one uniformly samples indices I_i and J_i with size $|I_i| = \mathcal{O}(r_i \log(d_i))$ and $|J_i| = \mathcal{O}\left(r_i \log(\prod_{j \neq i} d_j)\right)$, then $\operatorname{rank}(\boldsymbol{U}_i) = r_i$ holds for all i with high probability under some mild assumptions. Thus, the tensor CUR decomposition holds, and its computational complexity is dominated by computing the pseudoinverse of $\boldsymbol{U}_i$. Given the dimension of $\boldsymbol{U}_i$, the computational complexity of the pseudoinverse of $\boldsymbol{U}_i$ with Fiber sampling is $\mathcal{O}\left((n-1) r^3 \log^2 d\right)$; thus, Fiber CUR decomposition costs $\mathcal{O}\left(n r^3 \log^2 d\right)$.[1] The Chidori CUR decomposition has a slightly larger $|J_i|$, which is $\prod_{j \neq i} r_i \log(d_i) = \mathcal{O}((r \log d)^{n-1})$; thus, the decomposition costs $\mathcal{O}\left(r^{n+1} \log^n d\right)$. By contrast, the computational complexity of HOSVD is at least $\mathcal{O}(r d^n)$.

1.3. Related work: TRPCA. There is a long list of studies on RPCA [56] and low-rank tensor approximation [44], so we refer the reader to those two review articles for the aforementioned topics and focus on TRPCA works in this section. Consider a given tensor $\mathcal{X}$ that can represent a hypergraph network or a multidimensional observation [13]; the general assumption of TRPCA is that $\mathcal{X}$ can be decomposed as the sum of two tensors,

$$\mathcal{X} = \mathcal{L}^\star + \mathcal{S}^\star, \tag{1.6}$$

where $\mathcal{L}^\star \in \mathbb{R}^{d_1 \times \cdots \times d_n}$ is the underlying low-rank tensor and $\mathcal{S}^\star \in \mathbb{R}^{d_1 \times \cdots \times d_n}$ is the underlying sparse tensor. Compared to the exact low-rank tensor models, the TRPCA model contains an additional sparse tensor $\mathcal{S}$, which accounts for potential model outliers and hence is more stable with sparse noise. Different from the well-defined matrix rank, there exist various definitions of tensor decompositions that lead to various versions of tensor rank and to different versions of robust tensor decompositions. For example, [41, 43, 47] formulate TRPCA as a convex optimization model based on the tubal rank [59].

Based on the Tucker rank, we aim to solve the nonconvex optimization problem in this work:

$$\begin{aligned} &\underset{\mathcal{L}, \mathcal{S}}{\text{minimize}} \ \|\mathcal{X} - \mathcal{L} - \mathcal{S}\|_{\mathrm{F}} \\ &\text{subject to} \ \ \mathcal{L} \text{ is low-Tucker-rank and } \mathcal{S} \text{ is sparse.} \end{aligned} \tag{1.7}$$

Researchers have developed different optimization methods to solve (1.7) [13, 23, 31, 53]. For example, the work in [13] integrated the Riemannian gradient descent (RGD) and gradient

[1] For notational simplicity, we assume that the tensor has the same d and r along each mode when we discuss complexities. All log operators used in this paper stand for natural logarithms.

pruning methods to develop a linearly convergent algorithm for (1.7). This RGD algorithm will also serve as a guideline approach in our experiments. However, one of the major challenges in solving the Tucker rank–based TRPCA problem is the high computational cost for computing the Tucker decomposition. If $\mathcal{L}^\star$ is rank-$(r_1, \cdots, r_n)$, then the existing methods, e.g., [13, 25, 31, 32, 53], have computational complexity at least $\mathcal{O}(nd^n r)$—they are thus computationally challenging in large-scale problems. Thus, it is necessary to develop a highly efficient TRPCA algorithm for time-intensive applications.

1.4. Contributions. In this work, we consider the TRPCA problem under the Tucker rank setting. Our main contributions are threefold:

1. We provide theoretical evidence supporting the generality of the TRPCA model over the matrix robust PCA model obtained from the unfolded tensor (see section 2). That is, TRPCA requires a much weaker sparsity condition on the outlier component. Our theoretical finds will be empirically verified later in the numerical section.
2. We propose a novel nonconvex approach, coined robust tensor CUR decompositions (RTCUR), for large-scale[2] TRPCA problems (see section 3). RTCUR uses a framework of alternating projections and employs a novel modewise tensor decomposition [10] for fast low-rank tensor approximation. We present four variants of RTCUR with different sampling strategies (see subsection 3.4 for the details about sampling strategies). The computational complexity of RTCUR is as low as $\mathcal{O}(n^2dr^2\log^2 d)$ or $\mathcal{O}(ndr^n\log^n d)$ flops, depending on the sampling strategy, for an input n-mode tensor of size[3] $\mathbb{R}^{d\times\cdots\times d}$ with Tucker rank $(r,\ldots,r)$. Both computational complexities are substantially lower than the state-of-the-art TRPCA methods. For instance, two state-of-the-art methods [42, 43] based on tensor singular value decomposition have computational costs at least $\mathcal{O}(nd^n r)$ flops.
3. We verify the empirical advantages of RTCUR with synthetic datasets and three real-world applications (see section 4), including robust face modeling, video background subtraction, and network clustering. We show that RTCUR has not only speed efficiency but also superior robustness compared to the statesof the art. In particular, we provide certain outlier patterns that can be detected by RTCUR but that fail all matrix-based methods (see subsection 4.2). This further verifies our theoretical findings in section 2.

2. Characterizations of a sparse outlier tensor. The tensor RPCA problem can be also solved by using the RPCA method if we unfold the tensor into a matrix along a certain mode. To solve the tensor-to-matrix RPCA problem successfully, the unfolded outlier tensor must satisfy α-M-sparsity, a commonly used assumption in matrix RPCA problem. We state the definition of α-M-sparsity for matrix as follows.

Definition 2.1 (α-M-sparsity for matrix). *$S \in \mathbb{R}^{d_1\times d_2}$ is α-M-sparse if*

$$\|S\boldsymbol{e}_i\|_0 \le \alpha d_1 \quad \textit{and} \quad \|\boldsymbol{e}_j^\top S\|_0 \le \alpha d_2$$

for all $i = 1, \cdots, d_2$ and $j = 1, \cdots, d_1$.

[2] In our context, "large-scale" refers to large d.

[3] For notational simplicity, we assume that the tensor has the same d and r along each mode when we discuss complexities. All log operators used in this paper stand for natural logarithms.

However, when solving TRPCA directly with tensor-based methods, it is more natural to generalize α-M-sparsity to the tensor setting. We consider the following sparsity condition for outlier tensor [13].

Definition 2.2 (α-T-sparsity for tensor). *A tensor $\mathcal{S} \in \mathbb{R}^{d_1 \times \cdots \times d_n}$ is α-T-sparse if*

$$\|\mathcal{S} \times_j \boldsymbol{e}_{k_j,j}^\top\|_0 \le \alpha \prod_{i \ne j} d_i$$

for all $k_j = 1, \cdots, d_j$, where $\{\boldsymbol{e}_{k_j,j}\}_{k_j=1}^{d_j}$ is the standard basis of $\mathbb{R}^{d_j}$. Thus, $\mathcal{S} \times_j \boldsymbol{e}_{k_j,j}^\top$ corresponds to the k_jth slice of $\mathcal{S}$ along mode j.

We emphasize that Definitions 2.1 and 2.2 are not equivalent. In fact, from Figure 3, one can see that unfolding an outlier tensor along a certain mode can result in much worse sparsity in the unfolded matrix, with the exact same outlier pattern. Note that one can argue that unfolding alone a different mode may result in a not-worse M-sparsity. Although this is true, in practice, the user usually does not have prior knowledge of outlier patterns and thus cannot determine the most robust mode to unfold alone. Hence, we claim that Definition 2.2 is a much weaker condition than Definition 2.1. This is further verified by the following theorem.

Theorem 2.3. *Suppose that an n-order tensor $\mathcal{S} \in \mathbb{R}^{d \times \cdots \times d}$ is generated according to the Bernoulli distribution with expectation $\frac{\alpha}{2}$, i.e., $\mathcal{S}_{i_1,\cdots,i_n} \sim \texttt{Ber}(\frac{\alpha}{2})$, and that $\alpha d > \frac{2n \log d}{\log 4 - 1}$. Then $\mathcal{S}$ is α-T-sparse with probability at least $1 - nd^{1-nd^{n-2}}$ Moreover, by unfolding $\mathcal{S}$ into a matrix $M \in \mathbb{R}^{d^k \times d^{n-k}}$ along some particular modes with $k \in [1, n-1]$, M is α-M-sparse with probability at least $1 - d^{k-nd^{n-k-1}} - d^{n-k-nd^{k-1}}$.*

Remark 2.4. It is evident that for a large-scale tensor, i.e., $d \gg 1$ and $n > 2$, we have $1 - nd^{n-2} \ll \max\{k - nd^{n-k-1}, n - k - nd^{k-1}\}$. Thus, the condition of α-T-sparsity for $\mathcal{S}$ can be satisfied with much better probability than that of α-M-sparsity for the unfolders of $\mathcal{S}$.

Proof of Theorem 2.3. Since $\mathcal{S}_{i_1,\cdots,i_n} \sim \texttt{Ber}(\frac{\alpha}{2})$, we have $\mathbb{E}(\mathcal{S}_{i_1,\ldots,i_n}) = \frac{\alpha}{2}$. Let's first consider the sparsity of one slice of $\mathcal{S}$: We use $\mathcal{S}_{(j)[k]}$ to denote the kth slice along mode j. According to the multiplicative Chernoff bound, we have

$$\mathbb{P}\left(\sum_{i_1,\cdots,i_{j-1},i_{j+1},\cdots,i_n} [\mathcal{S}_{(j)[k]}]_{i_1,\cdots,i_{j-1},i_{j+1},\cdots,i_n} \ge \alpha d^{n-1} \right) < \left(\frac{e}{4}\right)^{\frac{\alpha d^{n-1}}{2}}. \tag{2.1}$$

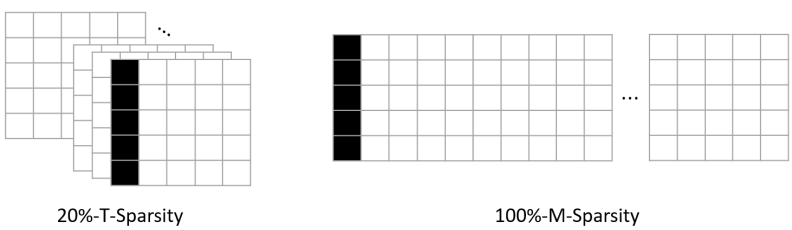

Figure 3. *T-sparsity versus M-sparsity. A black box represents an outlier entry, and a white box represents a good entry. The right-hand-side matrix is unfolded from the left-hand-side tensor.*

Taking the sparsity of all slices along all modes into account, the probability that $\mathcal{S}$ is α-sparsity can be bounded by

$$\mathbb{P}(\mathcal{S} \text{ is } \alpha\text{-T-sparse}) \geq \left(1 - \left(\frac{e}{4}\right)^{\frac{\alpha d^{n-1}}{2}}\right)^{nd} \geq 1 - nd\left(\frac{e}{4}\right)^{\frac{\alpha d^{n-1}}{2}}.$$

When $\alpha d > \frac{2n \log d}{\log 4 - 1}$, we thus have

$$\mathbb{P}(\mathcal{S} \text{ is } \alpha\text{-T-sparse}) \geq 1 - nd^{1-nd^{n-2}}. \tag{2.2}$$

Similarly, if we unfold $\mathcal{S}$ into a matrix $M \in \mathbb{R}^{d^k \times d^{n-k}}$, then we have that

$$\mathbb{P}(M \text{ is } \alpha\text{-M-sparse}) \geq 1 - d^{k-nd^{n-k-1}} - d^{n-k-nd^{k-1}} \tag{2.3}$$

provided that $\frac{2n \log d}{\log 4 - 1}$. ∎

Remark 2.5. The purpose of the sparsity condition is to ensure a well-defined robust tensor PCA problem; i.e., the low-rank and sparse tensors are separable. By the matrix rank-sparsity uncertainty principle [19], α-M-sparsity ensures that the matrix problem unfolded from the tensor is well-defined. However, with the newly developed tensor Tucker-rank-sparsity uncertainty principle [58, Proposition 2], the more relaxed α-T-sparsity is enough for well-defined robust tensor PCA problems. Moreover, to solve the problems with α-T-sparsity right, an algorithm based directly on tensor structures is needed like the one that will be proposed in the next section. The unfolded matrix-based algorithms will still require the more restricted α-M-sparsity condition and thus is more likely to fail.

3. Proposed approach.

In this section, we propose an efficient approach, called RTCUR, for the nonconvex TRPCA problem (1.7). RTCUR is developed in a framework of alternating projections: (I) First, we project $\mathcal{X} - \mathcal{L}^{(k)}$ onto the space of sparse tensors to update the estimate of outliers (i.e., $\mathcal{S}^{(k+1)}$); (II) then we project the less corrupted data $\mathcal{X} - \mathcal{S}^{(k+1)}$ onto the space of low-Tucker-rank tensors to update the estimate (i.e., $\mathcal{L}^{(k+1)}$). In our algorithm, the key to acceleration is using the tensor CUR decomposition for inexact low-Tucker-rank tensor approximation in step (II), which is proved to be much more efficient than the standard HOSVD [10] in terms of computational complexity. Consequently, in step (I), this inexact approximation allows us to estimate only the outliers in the smaller subtensors and submatrices involved in the tensor CUR decomposition. RTCUR is summarized in Algorithm 3.1. Notice that there are two variants of tensor CUR decompositions which will result in different J_i (see Remark 1.5), but the steps of Algorithm 3.1 will remain the same. Therefore, we will not distinguish the two decomposition methods in subsections 3.1 and 3.2 when discussing the details of steps (I) and (II). We will then show the computational complexity for Algorithm 3.1 with both Fiber and Chidori CUR decompositions in subsection 3.3.

3.1. Step (I): Update sparse component $\mathcal{S}$.

We consider the simple yet effective hard thresholding operator HT_ζ for outlier estimation. The operator is defined as

$$(\mathrm{HT}_\zeta(\mathcal{X}))_{i_1,\cdots,i_n} = \begin{cases} \mathcal{X}_{i_1,\cdots,i_n}, & |\mathcal{X}_{i_1,\cdots,i_n}| > \zeta, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Algorithm 3.1. RTCUR Decompositions.

Input: $\mathcal{X} = \mathcal{L}^\star + \mathcal{S}^\star \in \mathbb{R}^{d_1 \times \cdots \times d_n}$: observed tensor; $(r_1, \cdots, r_n)$: underlying Tucker rank of $\mathcal{L}^\star$; ε: targeted precision; $\zeta^{(0)}, \gamma$: thresholding parameters; $\{|I_i|\}_{i=1}^n, \{|J_i|\}_{i=1}^n$: cardinalities for sample indices. // J_i is defined differently for different sampling strategies. See subsection 3.4 for details about J_i and sampling strategies.
Initialization: $\mathcal{L}^{(0)} = \mathbf{0}, \mathcal{S}^{(0)} = \mathbf{0}, k = 0$
Uniformly sample the indices $\{I_i\}_{i=1}^n, \{J_i\}_{i=1}^n$
while $e^{(k)} > \varepsilon$ **do** // $e^{(k)}$ is defined in (3.7)
    (Optional) Resample the indices $\{I_i\}_{i=1}^n, \{J_i\}_{i=1}^n$
    // Step (I): Updating $\mathcal{S}$
    $\zeta^{(k+1)} = \gamma \cdot \zeta^{(k)}$
    $\mathcal{S}^{(k+1)} = \mathrm{HT}_{\zeta^{(k+1)}}(\mathcal{X} - \mathcal{L}^{(k)})$
    // Step (II): Updating $\mathcal{L}$
    $\mathcal{R}^{(k+1)} = (\mathcal{X} - \mathcal{S}^{(k+1)})(I_1, \cdots, I_n)$
    for $i = 1, \cdots, n$ **do**
        $\boldsymbol{C}_i^{(k+1)} = (\mathcal{X} - \mathcal{S}^{(k+1)})_{(i)}(:, J_i)$
        $\boldsymbol{U}_i^{(k+1)} = \mathrm{SVD}_{r_i}(\boldsymbol{C}_i^{(k+1)}(I_i, :))$
    end for
    $\mathcal{L}^{(k+1)} = \mathcal{R}^{(k+1)} \times_{i=1}^n \boldsymbol{C}_i^{(k+1)} \left(\boldsymbol{U}_i^{(k+1)}\right)^\dagger$
    $k = k + 1$
end while
Output: $\mathcal{R}^{(k)}, \boldsymbol{C}_i^{(k)}, \boldsymbol{U}_i^{(k)}$ for $i = 1, \cdots, n$: the estimates of the tensor CUR decomposition of $\mathcal{L}_\star$.

As shown in [6, 9, 48], with a properly chosen thresholding value, HT_ζ is effectively a projection operator onto the support of $\mathcal{S}^\star$. More specifically, we update

$$\mathcal{S}^{(k+1)} = \mathrm{HT}_{\zeta^{(k+1)}}(\mathcal{X} - \mathcal{L}^{(k)}). \tag{3.2}$$

If $\zeta^{(k+1)} = \|\mathcal{L}^\star - \mathcal{L}^{(k)}\|_\infty$ is chosen, then we have $\mathrm{supp}\,(\mathcal{S}^{(k+1)}) \subseteq \mathrm{supp}\,(\mathcal{S}^\star)$ and $\|\mathcal{S}^\star - \mathcal{S}^{(k+1)}\|_\infty \leq 2\|\mathcal{L}^\star - \mathcal{L}^{(k)}\|_\infty$. Empirically, we find that iteratively decaying thresholding values

$$\zeta^{(k+1)} = \gamma \cdot \zeta^{(k)} \tag{3.3}$$

provide superb performance with carefully tuned γ and $\zeta^{(0)}$. Note that a favorable choice of $\zeta^{(0)}$ is $\|\mathcal{L}^\star\|_\infty$, which can be easily estimated in many applications. The decay factor $\gamma \in (0, 1)$ should be tuned according to the level of difficulty of the TRPCA problem; e.g., those problems with higher rank, more dense outliers, or large condition numbers are considered to be harder. For successful reconstruction of $\mathcal{L}^\star$ and $\mathcal{S}^\star$, the harder problems require larger γ. When applying RTCUR on both synthetic and real-world data, we observe that $\gamma \in [0.6, 0.9]$ generally performs well. Since real-world data normally lead to more difficult problems, we fix $\gamma = 0.7$ for the synthetic experiment and $\gamma = 0.8$ for the real-world data studies in section 4.

3.2. Step (II): Update low-Tucker-rank component $\mathcal{L}$. SVD is the most popular method for low-rank approximation under matrix settings since SVD gives the best rank-r approximation of given matrix $\boldsymbol{X}$, both with respect to the operator norm and to the Frobenius norm [2]. Similarly, HOSVD has been the standard method for low-Tucker-rank approximation under tensor settings in many works [2, 22, 39, 52]. However, the computational complexity of HOSVD is at least $\mathcal{O}(rd^n)$; hence, computing HOSVD is very expensive when the problem scale is large. As highlighted in [10, sections 3.2 and 3.3], tensor CUR decomposition can serve as an effective low-Tucker-rank approximation method, even with perturbations. As such, we employ tensor CUR decomposition for accelerated inexact low-Tucker-rank tensor approximations. Namely, we update the estimate of the low-Tucker-rank component $\mathcal{L}$ by setting

$$\mathcal{L}^{(k+1)} = \mathcal{R}^{(k+1)} \times_{i=1}^{n} \boldsymbol{C}_i^{(k+1)} \left(\boldsymbol{U}_i^{(k+1)}\right)^{\dagger}, \tag{3.4}$$

where

$$\begin{aligned} \mathcal{R}^{(k+1)} &= (\mathcal{X} - \mathcal{S}^{(k+1)})(I_1, \cdots, I_n), \\ \boldsymbol{C}_i^{(k+1)} &= (\mathcal{X} - \mathcal{S}^{(k+1)})_{(i)}(:, J_i), \\ \boldsymbol{U}_i^{(k+1)} &= \mathrm{SVD}_{r_i}(\boldsymbol{C}_i^{(k+1)}(I_i, :)). \end{aligned} \tag{3.5}$$

3.3. Computational complexities. As mentioned in subsection 1.2, the complexity for computing a tensor CUR decomposition is much lower than HOSVD, and the dominating steps in RTCUR are the hard thresholding operator and the tensor/matrix multiplications. For both Fiber and Chidori CUR decompositions, only the sampled subtensors and submatrices are required when computing (3.5). Thus, we merely need to estimate the outliers on these subtensors and submatrices, and (3.2) should not be fully executed. Instead, we only compute

$$\begin{aligned} \mathcal{S}^{(k+1)}(I_1, \cdots, I_n) &= \mathrm{HT}_{\zeta^{(k+1)}}((\mathcal{X} - \mathcal{L}^{(k)})(I_1, \cdots, I_n)), \\ \mathcal{S}_{(i)}^{(k+1)}(:, J_i) &= \mathrm{HT}_{\zeta^{(k+1)}}((\mathcal{X} - \mathcal{L}^{(k)})_{(i)}(:, J_i)) \end{aligned} \tag{3.6}$$

for all i. Not only can we save the computational complexity on hard thresholding, but also much smaller subtensors of $\mathcal{L}^{(k)}$ need to be formed in (3.6). We can form the required subtensors from the saved tensor CUR components, which is much cheaper than forming and saving the whole $\mathcal{L}^{(k)}$.

In particular, for $\mathcal{X} \in \mathbb{R}^{d \times \cdots \times d}$, $r_1 = \cdots = r_n = r$ and $|I_1| = \cdots = |I_n| = \mathcal{O}(r \log d)$, computing $\mathcal{L}^{(k)}(I_1, \cdots, I_n)$ requires n tensor-matrix product operations, so the complexity for computing $\mathcal{L}^{(k)}(I_1, \cdots, I_n)$ is $\mathcal{O}(n(r \log d)^{n+1})$ flops for both Fiber and Chidori CUR decompositions. The complexity for computing $\mathcal{L}_{(i)}^{(k)}(:, J_i)$ with Fiber CUR is different from the complexity of computing $\mathcal{L}_{(i)}^{(k)}(:, J_i)$ with Chidori CUR. With Fiber CUR, we compute each fiber in $\mathcal{L}_{(i)}^{(k)}(:, J_i)$ independently, and each fiber takes n tensor-matrix product operations. The first $n-1$ operations transform the n-mode core tensor $\mathcal{L}^{(k)}(I_1, \cdots, I_n)$ into a 1-mode tensor, which is a vector of length $\mathcal{O}(r \log d)$, and the last operation transforms this vector into another vector of length d. Since there are $J_i = \mathcal{O}(nr \log d)$ fibers in total, the complexity for

computing $\mathcal{L}_{(i)}^{(k)}(:, J_i)$ with Fiber CUR decomposition is $\mathcal{O}(nr \log d((r \log d)^n + dr \log d))$ flops. With Chidori CUR, we compute $\mathcal{L}_{(i)}^{(k)}(:, J_i)$ as a complete unit using n tensor-matrix product operations. The first $n-1$ operations on the core tensor do not change its size, and the last operation changes the size of the ith mode to d. Therefore, the complexity for computing $\mathcal{L}_{(i)}^{(k)}(:, J_i)$ with Chidori CUR decomposition is $\mathcal{O}(n(r \log d)^{n+1} + d(r \log d)^n)$ flops.

Moreover, for time-saving purposes, we may avoid computing the Frobenius norm of the full tensor when computing the relative error for the stopping criterion. In RTCUR, we adjust the relative error formula to be

$$e^{(k)} = \frac{\|\mathcal{E}^{(k)}(I_1, \cdots, I_n)\|_{\mathrm{F}} + \sum_{i=1}^{n} \|\mathcal{E}_{(i)}^{(k)}(:, J_i)\|_{\mathrm{F}}}{\|\mathcal{X}(I_1, \cdots, I_n)\|_{\mathrm{F}} + \sum_{i=1}^{n} \|\mathcal{X}_{(i)}(:, J_i)\|_{\mathrm{F}}}, \tag{3.7}$$

where $\mathcal{E}^{(k)} = \mathcal{X} - \mathcal{L}^{(k)} - \mathcal{S}^{(k)}$ so that it does not use any extra subtensor or fiber but only those we already have. We hereby summarize the computational complexity for each step from Algorithm 3.1 in Table 3.1.

If we assume that the tensor size d is comparable with or greater than $\mathcal{O}((r \log d)^{n-1})$, then we can conclude that the total computational complexity is $\mathcal{O}(n^2 dr^2 \log^2 d)$ flops for RTCUR with Fiber CUR decomposition and $\mathcal{O}(ndr^n \log^n d)$ flops for RTCUR with Chidori CUR decomposition. Otherwise, the computational complexity would be $\mathcal{O}(n^2 r^{n+1} \log^{n+1} d)$ flops for RTCUR with Fiber CUR, and the complexity for RTCUR with Chidori CUR remains unchanged. For all tensors tested in section 4, the first case holds. Therefore, in Table 3.1, we highlighted $\mathcal{O}(n^2 dr^2 \log^2 d)$ and $\mathcal{O}(ndr^n \log^n d)$ as the dominating terms.

3.4. Four variants of RTCUR. In subsection 1.2, we discussed two versions of tensor CUR decomposition: Fiber CUR decomposition and Chidori CUR decomposition. Each of the decomposition methods could derive two slightly different RTCUR algorithms depending on if we fix sample indices through all iterations (see Algorithm 3.1). As a result of this, we obtain four variants in total. We give different suffixes for each variant of the RTCUR algorithm: RTCUR-FF, RTCUR-FC, RTCUR-RF, and RTCUR-RC. We will showcase experimental results for all variants in section 4. The first letter in the suffix indicates whether we fix the sample indices through all iterations: "F" stands for "fix," where the variant uses fixed

Table 3.1

Computational complexity for each step from Algorithm 3.1. The complexity for computing $\mathcal{S}(I_1, \cdots, I_n)$ and $\mathcal{R}$ are the same as their size; the complexity for $\boldsymbol{C}_i \boldsymbol{U}_i^\dagger$ is introduced in subsection 1.2; the complexity for computing $\mathcal{L}$ and the error term is introduced in subsection 3.3. The dominating terms are highlighted in bold.

Computational Complexity	Fiber Sampling	Chidori Sampling
Sparse subtensor $\mathcal{S}(I_1, \cdots, I_n)$ or $\mathcal{R}$	$\mathcal{O}(r^n \log^n d)$	$\mathcal{O}(r^n \log^n d)$
All $\mathcal{S}_{(i)}(:, J_i)$ or $\boldsymbol{C}_i$ for n modes	$\mathcal{O}(n^2 rd \log d)$	$\mathcal{O}(nr^{n-1} d \log^{n-1} d)$
All $\boldsymbol{U}_i^\dagger$ for n modes	$\mathcal{O}(n^2 r^3 \log^2 d)$	$\mathcal{O}(nr^{n+1} \log^n d)$
All $\boldsymbol{C}_i \boldsymbol{U}_i^\dagger$ for n modes	$\mathcal{O}(\boldsymbol{n^2 r^2 d \log^2 d})$	$\mathcal{O}(\boldsymbol{nr^n d \log^n d})$
Low-rank subtensor $\mathcal{L}(I_1, \cdots, I_n)$	$\mathcal{O}(nr^{n+1} \log^{n+1} d)$	$\mathcal{O}(nr^{n+1} \log^{n+1} d)$
All $\mathcal{L}_{(i)}(:, J_i)$ for n modes	$\mathcal{O}(n^2 r^{n+1} \log^{n+1} d + \boldsymbol{n^2 r^2 d \log^2 d})$	$\mathcal{O}(n^2 r^{n+1} \log^{n+1} d + \boldsymbol{nr^n d \log^n d})$
Error term $\mathcal{E}^{(k)}(I_1, \cdots, I_n)$ and $\mathcal{E}_{(i)}^{(k)}(:, J_i)$	$\mathcal{O}(r^n \log^n d + n^2 dr \log d)$	$\mathcal{O}(dr^{n-1} \log^{n-1}(d))$

Algorithm 3.2. Conversion from CUR to HOSVD.

Input: $\mathcal{R}, \boldsymbol{C}_i, \boldsymbol{U}_i$: CUR decomposition of the tensor $\mathcal{A}$
$[\boldsymbol{Q}_i, \boldsymbol{R}_i] = \mathrm{qr}\left(\boldsymbol{C}_i \boldsymbol{U}_i^{\dagger}\right)$ for $i = 1, \cdots, n$
$\mathcal{T}_1 = \mathcal{R} \times_1 \boldsymbol{R}_1 \times_2 \cdots \times_n \boldsymbol{R}_n$
Compute HOSVD of $\mathcal{T}_1$ to find $\mathcal{T}_1 = \mathcal{T} \times_1 \boldsymbol{V}_1 \times_2 \cdots \times_n \boldsymbol{V}_n$
Output: $[\![\mathcal{T}; \boldsymbol{Q}_1 \boldsymbol{V}_1, \cdots, \boldsymbol{Q}_n \boldsymbol{V}_n]\!]$: HOSVD decomposition of $\mathcal{A}$

sample indices through all iterations, and "R" stands for "resampling," where the variant resamples $\{I_i\}_{i=1}^n$ and $\{J_i\}_{i=1}^n$ in each iteration. The second letter indicates which type of CUR decomposition we use in RTCUR. "F" represents that RTCUR is derived from Fiber CUR decomposition, and "C" stands for Chidori CUR. For Fiber CUR, the amount of fibers to be sampled refers to [27, Corollary 5.2], i.e., $|I_i| = \upsilon r_i \log(d_i)$, $|J_i| = \upsilon r_i \log(\prod_{j \neq i} d_j)$, and J_i is sampled independently from I_i. Here, υ denotes the sampling constant, a hyperparameter that will be tuned in the experiments. For Chidori CUR, $|I_i| = \upsilon r_i \log(d_i)$ and $J_i = \otimes_{j \neq i} I_j$. Of these four variants, RTCUR-FF requires minimal data accessibility and runs slightly faster than other variants. The resampling variants access more data and take some extra computing; for example, the denominator of (3.7) has to be recomputed per iteration. However, accessing more redundant data means that resampling variants have better chances of correcting any "unlucky" sampling over the iterations. Thus, we expect resampling variants to have superior outlier tolerance over fixed sampling variants, and the fixed sampling variants have an efficiency advantage over the resampling variants under specific conditions (e.g., when reaccessing the data is expansive).

The difference between Chidori variants and Fiber variants has similar properties: If we choose the same υ and let $|I_i| = \upsilon r_i \log(d_i)$ for both Chidori and Fiber CUR described in subsection 1.2, then the Chidori variants generally access more tensor entries compared to the Fiber variants. Therefore, with the same sampling constant υ, Chidori variants require more computing time in each iteration. Nevertheless, Chidori variants can tolerate more dense outliers with these extra entries than Fiber variants. We will further investigate their computational efficiency and practical performance in section 4.

Remark 3.1. The tensor CUR decomposition represented in Theorem 1.4(i) is also in Tucker decomposition form. We can efficiently convert the tensor CUR decomposition to HOSVD with Algorithm 3.2 [10]. In contrast, converting HOSVD to a tensor CUR decomposition is not as straightforward.

4. Numerical experiments.

This section presents a set of numerical experiments that compare the empirical performance of RTCUR with several state-of-the-art robust matrix/tensor PCA algorithms, including RGD [13], the alternating direction method of multipliers (ADMM) [43], accelerated alternating projections (AAP) [6], and iterative robust CUR (IRCUR) [9]. RGD and ADMM are designed for the TRPCA task, while AAP and IRCUR are designed for the traditional matrix RPCA task. Note that RGD is Tucker-rank based and that ADMM is tubal-rank based.

In each subsection, we evaluate the performance of all four proposed variants: RTCUR-FF, RTCUR-RF, RTCUR-FC, and RTCUR-RC. However, for the network clustering experiment, we only use fixed sampling (RTCUR-FF and RTCUR-FC). As the coauthorship network is highly sparse, the resampling variants may diminish the core tensor with a high probability.

The rest of this section is structured as follows. In subsection 4.1, we present two synthetic experiments. Specifically, subsection 4.1.1 examines the empirical relationship between outlier tolerance and sample size for RTCUR, while subsection 4.1.2 demonstrates the speed advantage of RTCUR over the state of the art. In subsections 4.2 and 4.3, we apply RTCUR to two real-world problems, namely, face modeling and color video background subtraction. In subsection 4.4, we apply RTCUR on network clustering applications and analyze the obtained results.

We obtain the codes for all compared algorithms from the authors' websites and hand-tune the parameters for their best performance. For RTCUR, we sample $|I_i| = \upsilon r_i \log(d_i)$ (and $|J_i| = \upsilon r_i \log(\prod_{j \neq i} d_j)$ for Fiber variants) for all i, and υ is called the *sampling constant* throughout this section. All the tests are executed from Matlab R2020a on an Ubuntu workstation with an Intel i9-9940X CPU and 128 GB RAM. The relevant codes are available at https://github.com/huangl3/RTCUR.

4.1. Synthetic examples. For the synthetic experiments, we use $d := d_1 = \cdots = d_n$ and $r := r_1 = \cdots = r_n$. The observed tensor $\mathcal{X}$ is composed as $\mathcal{X} = \mathcal{L}^\star + \mathcal{S}^\star$. To generate n-mode $\mathcal{L}^\star \in \mathbb{R}^{d \times \cdots \times d}$ with Tucker rank $(r, \cdots, r)$, we take $\mathcal{L}^\star = \mathcal{Y} \times_1 Y_1 \times_2 \cdots \times_n Y_n$, where $\mathcal{Y} \in \mathbb{R}^{r \times \cdots \times r}$ and $\{Y_i \in \mathbb{R}^{d \times r}\}_{i=1}^n$ are Gaussian random tensors and matrices with standard normal entries. To generate the sparse outlier tensor $\mathcal{S}^\star$, we uniformly sample $\lfloor \alpha d^n \rfloor$ entries to be the support of $\mathcal{S}^\star$, and the values of the nonzero entries are uniformly sampled from the interval $[-\mathbb{E}(|\mathcal{L}^\star_{i_1,\cdots,i_n}|), \mathbb{E}(|\mathcal{L}^\star_{i_1,\cdots,i_n}|)]$.

4.1.1. Phase transition. We study the empirical relation between the outlier corruption rate α and sampling constant υ for all four variants of RTCUR using $300 \times 300 \times 300$ (i.e., $n = 3$ and $d = 300$) problems with Tucker rank (r, r, r), where $r = 3, 5$, or 10. We set the thresholding parameters to $\zeta^{(0)} = \|\mathcal{L}\|_\infty$ and $\gamma = 0.7$ and use the stopping condition $e^{(k)} < 10^{-5}$. A test example is considered successfully solved if $\|\mathcal{L}^\star - \mathcal{L}^{(k)}\|_{\mathrm{F}} / \|\mathcal{L}^\star\|_{\mathrm{F}} \leq 10^{-3}$. For each pair of α and υ, we generate 10 test examples.

We summarize the experimental results in Figure 4, where a white pixel means that all 10 test examples are successfully solved under the corresponding problem parameter setting and a black pixel means that all 10 test cases fail. On observation, one has that the Chidori variants RTCUR-FC and RTCUR-RC can recover the low-rank tensor with a higher outlier rate than the Fiber variants with the same sampling constant υ. This expected behavior can be attributed to the fact that Chidori sampling accesses more data from the tensor with the same υ, leading to more stable performance compared to Fiber sampling. Furthermore, RTCUR-RF also outperforms RTCUR-FF, as resampling the fibers allows access to more tensor entries throughout the process. This phenomenon holds true for the Chidori variants as well, where RTCUR-FC and RTCUR-RC exhibit very similar phase transitions. Additionally, we observe that smaller values of r tolerate more outliers, as larger values of r make the TRPCA task more complex. Increasing υ improves outlier tolerance, but at the cost of larger subtensors and more fibers to be sampled, resulting in longer computational time.

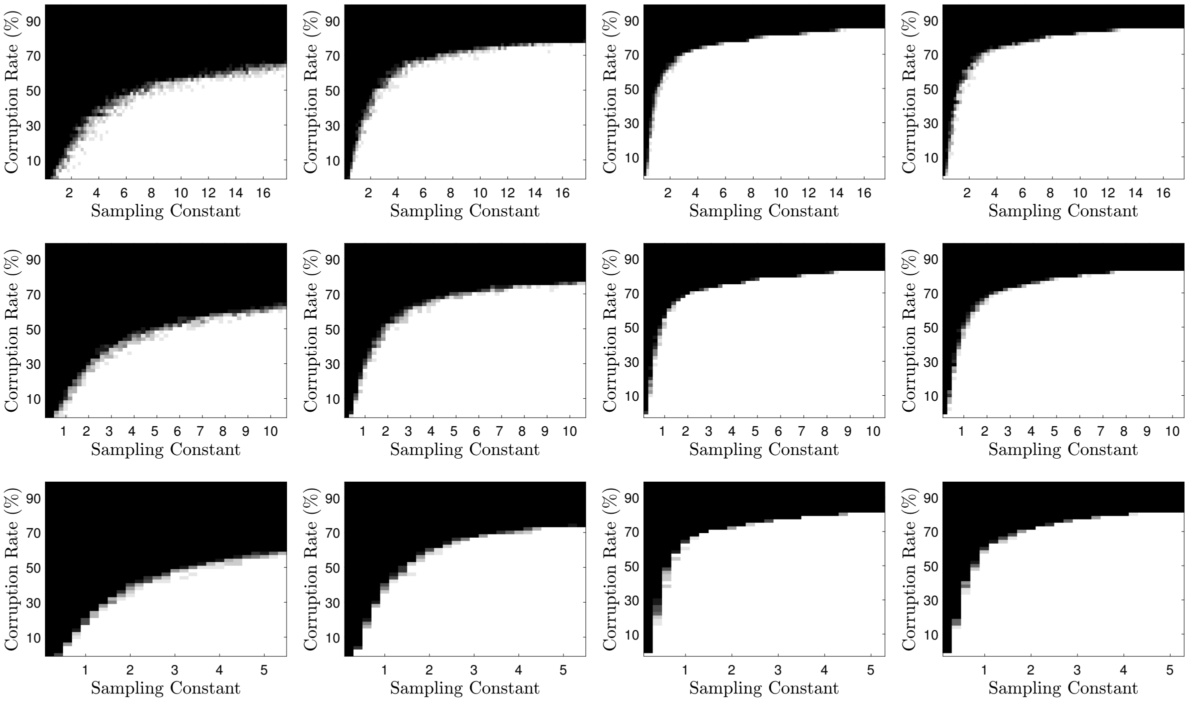

Figure 4. *Empirical phase transition in corruption rate α and sampling constant υ. Left to Right: RTCUR-FF, RTCUR-RF, RTCUR-FC, RTCUR-RC.* ***Top****: $r = 3$.* ***Middle****: $r = 5$.* ***Bottom****: $r = 10$.*

station with an Intel i9-9940X CPU and 128GB RAM. The relevant codes are available at https://github.com/huangl3/RTCUR.

4.1.2. Computational efficiency. In this section, we present a comparative analysis of the computational efficiency of RTCUR and the state-of-the-art tensor/matrix RPCA algorithms mentioned in section 4. To apply the matrix RPCA algorithms, we first unfold an n-mode $d \times \cdots \times d$ tensor to a $d \times d^{n-1}$ matrix and then solve the matrix RPCA problem by setting the matrix rank as r. It is worth noting that we do not apply ADMM to this fixed-Tucker-rank experiment, as the target tensor rank for ADMM is tubal-rank instead of Tucker-rank [43]. For all tests, we set the underlying low-Tucker-rank component as rank $[3, 3, 3]$ and add 20% corruption. We set parameters $\upsilon = 3$, $\zeta^{(0)} = \|\mathcal{L}\|_\infty$, $\gamma = 0.7$ for all four variants of RTCUR. The reported runtime is averaged over 20 trials.

In Figure 5, we investigate the 3-mode TRPCA problem with varying dimension d and compare the total runtime (all methods halt when relative error $e^{(k)} < 10^{-5}$). It is evident that all variants of RTCUR are significantly faster than the compared algorithms when d is large. Next, we evaluate the convergence behavior of the tested algorithms in Figure 6. We exclude RGD as well because running RGD on this experiment is too expensive. We find that all the tested algorithms converge linearly and that variants of RTCUR run the fastest. Moreover, as discussed in subsection 3.4, the Fiber sampling has a runtime advantage over Chidori sampling with the same sampling constant, and fixed sampling runs slightly faster than resampling in all tests.

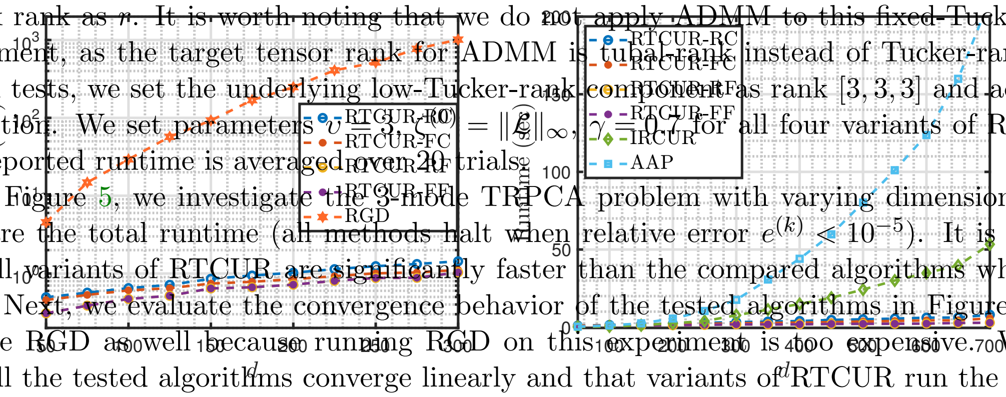

Figure 5. *Runtime vs. dimension comparison among variants of RTCUR, RGD, IRCUR and AAP on tensors with size $d \times d \times d$ and Tucker rank $(3, 3, 3)$. The RGD method proceeds relatively slowly for larger tensors, so we only test the RGD runtime for tensors with a size smaller than 300 for each mode.*

4.1. Synthetic examples. For the synthetic experiments, we use $d := d_1 = \cdots = d_n$ and $r := r_1 = \cdots = r_n$. The observed tensor $\mathcal{X}$ is composed as $\mathcal{X} = \mathcal{L}^\star + \mathcal{S}^\star$. To generate

4.2. Robust face modeling. In this section, we apply the four variants of RTCUR and compare them with the aforementioned tensor/matrix RPCA algorithms on the robust face

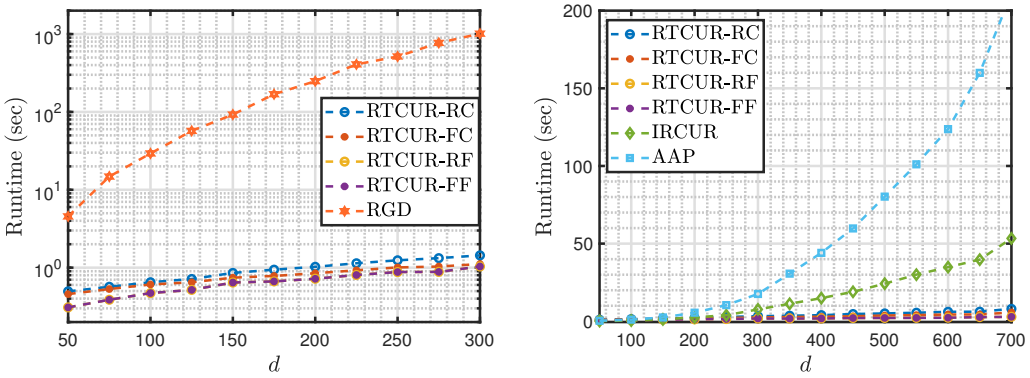

Figure 5. *Runtime-versus-dimension comparison among variants of RTCUR, RGD, IRCUR, and AAP on tensors with size $d \times d \times d$ and Tucker rank $(3,3,3)$. The RGD method proceeds relatively slowly for larger tensors, so we only test the RGD runtime for tensors with a size smaller than 300 for each mode.*

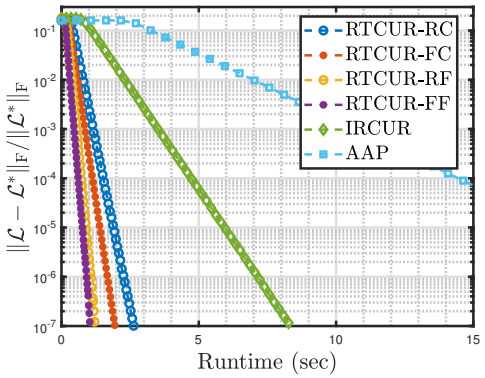

Figure 6. *Runtime-versus-relative error comparison among RTCUR-F, RTCUR-R, AAP, and IRCUR on tensors with size $500 \times 500 \times 500$ and Tucker rank $(3,3,3)$.*

modeling task using data from the UT Dallas database [50]. The dataset consists of a face speech video of approximately 5 seconds with a resolution of 360 × 540. We extract 10 nonsuccessive frames and mark a monochromatic block on different color channels for 10 distinct frames per color. This process results in a total of 40 color frames, including the original 10 unmarked frames. As a monochromatic frame usually does not have a low-rank structure, we vectorize each color channel of each frame into a vector and construct a (height · width) × 3 × frames tensor. The targeted Tucker rank is set as $\boldsymbol{r} = (3,3,3)$ for all videos. For those matrix algorithms, including AAP and IRCUR, we unfold the tensor to a (height · width) × (3 · frames) matrix, and rank 3 is used. We set RTCUR parameters $\upsilon = 2$, $\zeta^{(0)} = 255$, $\gamma = 0.7$ in this experiment.

Figure 7 presents the test examples and visual results; Table 2.1 summarizes the runtime for each method applied on this task. One can see that the matrix-based methods fail to detect the monochromatic outlier blocks since they lose the structural connection between color channels after matricization, albeit they spend less time on this task. In contrast, all variants of RTCUR successfully detect the outlier blocks. The other two TPRCA methods, ADMM

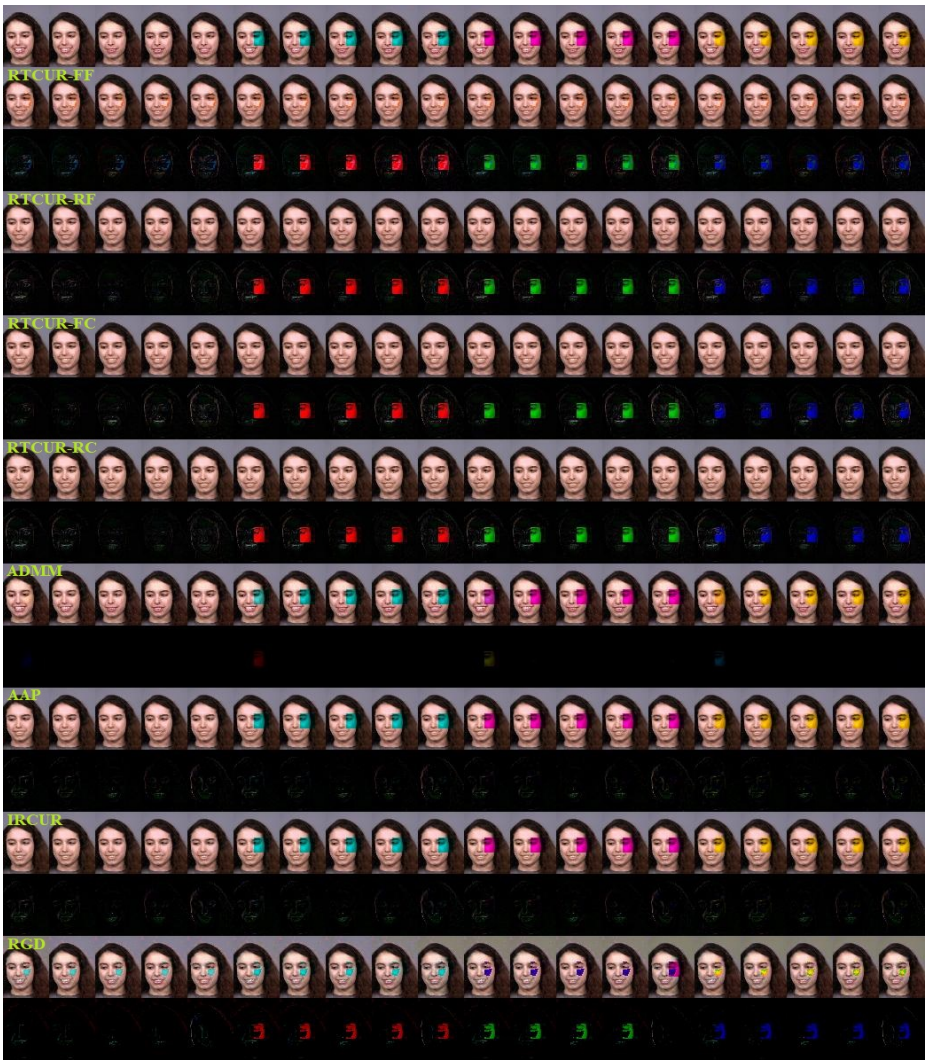

Figure 7. *Visual results for robust face modeling. The top row contains the corrupted faces, the second and third rows are the recovered faces and detected outliers outputted by RTCUR-FF, the fourth and fifth rows are results from RTCUR-RF, the sixth and seventh rows are results from RTCUR-FC, the eighth and ninth rows are results from RTCUR-RC, the tenth and eleventh rows are results from ADMM, the twelfth and thirteenth rows are results from AAP, the fourteenth and fifteenth rows are results from IRCUR, and the sixteenth and seventeenth rows are results from RGD.*

and RGD, partially detect the outlier blocks. Since ADMM is based on tubal decomposition, it is not a surprise to see different performances in this experiment. The empirical results in this section verify our claim in Remark 2.5.

4.3. Color video background subtraction. We apply the four variants of RTCUR and the aforementioned tensor/matrix RPCA algorithms on the color video background subtraction task. We obtain five color video datasets from various sources: *Shoppingmall* [34], *Highway* [4], *Crossroad* [4], *Port* [4], and *Parking-lot* [49]. Similar to subsection 4.2, we vectorize each frame

Table 2.1

Runtime comparison (in seconds) for the face modeling task. The matrix RPCA approaches (AAP and IRCUR) meet the termination condition earlier with the unfolded tensor, but they failed to detect the artificial noise in this task (see Figure 7).

Method	Runtime	Method	Runtime
RTCUR-FF	2.247	ADMM	30.61
RTCUR-RF	2.289	AAP	1.754
RTCUR-FC	2.319	IRCUR	1.307
RTCUR-RC	2.701	RGD	1430.8

Table 3.2

Video information and runtime comparison (in seconds) for color video background subtraction task.

	Video size	RTCUR-FF	RTCUR-RF	RTCUR-FC	RTCUR-RC	ADMM	AAP	IRCUR
Shoppingmall	$256 \times 320 \times 3 \times 1250$	**3.53**	5.83	10.68	10.75	783.67	50.38	15.71
Highway	$240 \times 320 \times 3 \times 440$	**3.15**	5.47	6.80	7.55	168.55	18.10	3.87
Crossroad	$350 \times 640 \times 3 \times 600$	**6.15**	13.33	8.46	12.01	1099.3	97.85	35.47
Port	$480 \times 640 \times 3 \times 1000$	**11.04**	18.34	26.63	27.93	2934.3	154.30	71.64
Parking-lot	$360 \times 640 \times 3 \times 400$	**3.79**	4.52	6.62	8.14	854.50	34.70	17.38

and construct a $(\text{height} \cdot \text{width}) \times 3 \times \text{frame}$ data tensor. The tensor is unfolded to a $(\text{height} \cdot \text{width}) \times (3 \cdot \text{frame})$ matrix for matrix RPCA methods. We use Tucker rank $(3, 3, 3)$ for tensors methods and rank 3 for matrix methods. We exclude RGD from this experiment because the disk space required for RGD exceeds our server limit. Among the tested videos, *Shoppingmall, Highway,* and *Parking-lot* are normal-speed videos with almost static backgrounds. *Crossroad* and *Port* are outdoor time-lapse videos; hence, their background colors change slightly between different frames. We observe that all tested algorithms perform very similarly for videos with static backgrounds and produce visually desirable output. On the other hand, the color of the extracted background varies slightly among different algorithms on time-lapse videos. Since the color of the background keeps changing slightly for the time-lapse videos, we cannot determine the ground-truth color of the background; hence, we do not rank the performance of different algorithms. The runtime for results along with video size information are summarized in Table 3.2. By comparing the runtime of four variants of RTCUR, we can observe that the experiment result generally agrees with the analysis on computational efficiency in subsection 3.4. All RTCUR variants accomplish the background subtraction task faster than the guideline methods. In addition, we provide some selected visual results in Figure 8.

4.4. Network clustering. In this section, we apply our RTCUR algorithm and the TR-PCA algorithm RGD from [13] for the community detection task on the coauthorship network data from [33] and compare their results and efficiency. This dataset contains 3248 papers with a list of authors for each paper (3607 authors in total) and hence could naturally serve as the adjacency matrix for the weighted coauthorship graph. The original paper for this dataset tests a number of community detection algorithms, including network spectral clustering, profile likelihood, the pseudolikelihood approach, etc., on a selected subset with 236 authors and

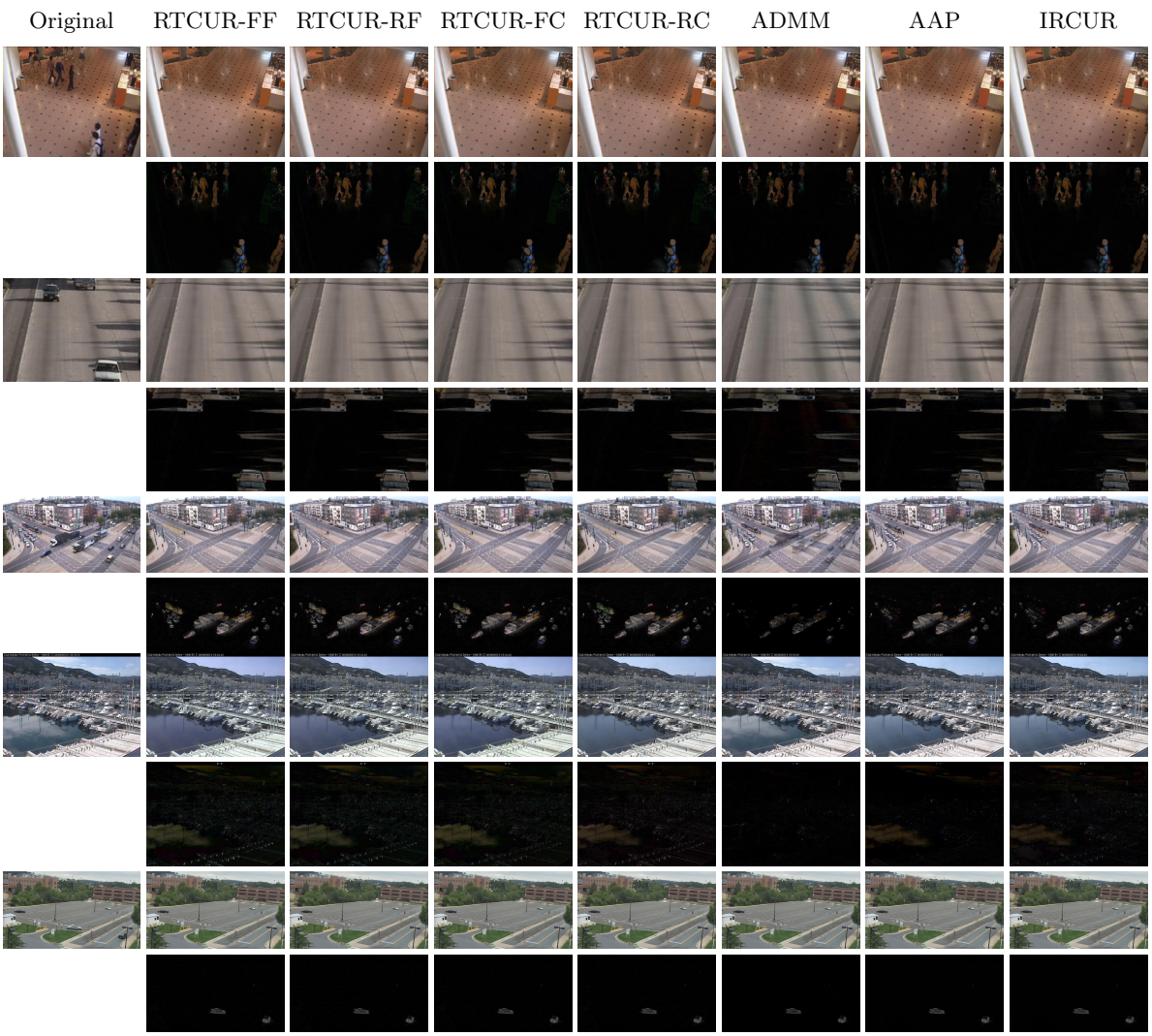

Figure 8. *Visual results for color video background subtraction. The first two rows are separated backgrounds and foregrounds corresponding to a frame from Shoppingmall, the third and fourth rows are separated backgrounds and foregrounds corresponding to a frame from Highway, the fifth and sixth rows are separated backgrounds and foregrounds corresponding to a frame from Crossroad, the seventh and eighth rows correspond to a frame from Port and the last two rows correspond to a frame from Parkinglot except the first column which is the original frame.*

542 papers. To obtain this subset, we generate an undirected graph for the 3607 authors and put an edge between two authors if they have coauthored two or more papers. We then take the largest connected component, and all the nodes in this component are the subset of authors we are interested in. Since we aim to explore the higher-order interactions among this coauthorship network, we convert this connected component into a 3-mode adjacency tensor $\mathcal{T}$ and apply TRPCA algorithms to obtain the low-rank component from $\mathcal{T}$. We construct the adjacency tensor $\mathcal{T}$ with the following rules:

Here $\mathfrak{S}(S)$ denotes all permutations of the set S. Therefore the adjacency tensor $\mathcal{T}$ is symmetric. Now we apply RTCUR with different sampling constants as well as the TRPCA algorithm ReGD [15] to learn the low-rank component $\mathcal{L}$ with Tucker rank $(4, 4, 4)$, which is used to infer the communities in this network. Then we apply the SCORE algorithm [33] as the clustering algorithm on the low-rank tensor $\mathcal{L}$. We use SCORE instead of other traditional clustering algorithms such as spectral clustering because SCORE could mitigate the influence of node heterogeneity [33]. We plot the results from each TRPCA algorithm in Figure 9.

- For any two connected authors (i, j), which means that author i and j have worked together for at least two papers, we set $\mathcal{T}_{\mathfrak{G}(i,i,j)} = \mathcal{T}_{\mathfrak{G}(i,j,j)} = 1$.
- For any three pairwisely connected authors (i, j, k), we set $\mathcal{T}_{\mathfrak{G}(i,j,k)} = 1$. Notice that these three authors may not appear in one paper at the same time, but each pair of them have worked together for at least two papers.

Here $\mathfrak{G}(S)$ denotes all permutations of the set S. Therefore, the adjacency tensor $\mathcal{T}$ is symmetric. Now we apply RTCUR with different sampling constants as well as the TRPCA algorithm RGD [13] to learn the low-rank component $\mathcal{L}$ with Tucker rank $(4, 4, 4)$, which is used to infer the communities in this network. Then we apply the SCORE algorithm [33] as the clustering algorithm on the low-rank tensor $\mathcal{L}$. We use SCORE instead of other traditional clustering algorithms, such as spectral clustering, because SCORE could mitigate the influence of node heterogeneity [33]. We plot the results from each TRPCA algorithm in Figure 9.

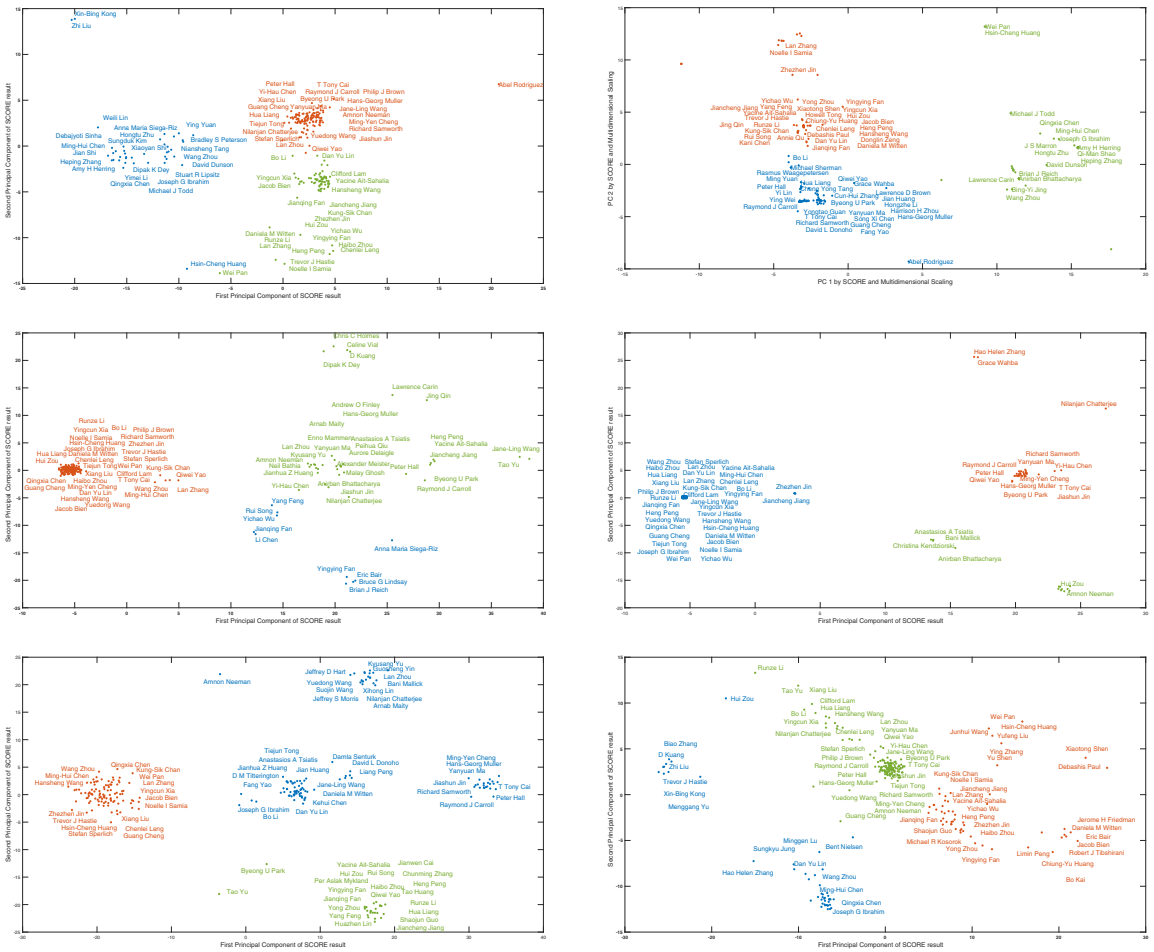

Figure 9. *Three communities detected in the "High-Dimensional Data Analysis" coauthorship network with SCORE [33], RGD [13], and RTCUR. Top left: Result from SCORE on original tensor. Top right: Result from RGD and SCORE. Middle left: Result from RTCUR-FF and SCORE, with* $\upsilon = 6$. *Middle right: Result from RTCUR-FC and SCORE, with* $\upsilon = 2$. *Bottom left: Result from RTCUR-FC and SCORE, with* $\upsilon = 6$. *Bottom right: Result from RTCUR-FC and SCORE, with* $\upsilon = 11$. *All TRPCA methods are applied on the adjacency tensor* $\mathcal{T}$ *with Tucker rank* $= (4, 4, 4)$.

The clustering of the "High-Dimensional Data Analysis" coauthorship network is an unsupervised task, which means that the ground truth of labeling an author with a certain community does not exist. Therefore, we do not focus on qualitatively evaluating each result, but we present the new findings from higher-order interactions among coauthors and analyze the results from different choices of parameters. Previous studies on this coauthorship network generally provide three clusters with names: the "Carroll–Hall" group, the "North Carolina" community, and the "Fan and Others" group [13, 33]. Among them, the "Carroll–Hall" group generally includes researchers in nonparametric and semiparametric statistics, functional estimation, and high-dimensional statistics; the "North Carolina" group generally includes researchers from Duke University, the University of North Carolina, and North Carolina State University; and the "Fan and Others" group includes primarily the researchers collaborating closely with Jianqing Fan or his coauthors and other researchers who do not obviously belong to the first two groups [13]. For conciseness, we will make use of the same name of each group as in previous studies on this coauthorship network.

The top two plots of Figure 9 are existing results from [13]. With SCORE as the clustering method, the original tensor and the RGD output both successfully reveal the two groups: the "Carroll–Hall" group and a "North Carolina" community, with a slightly different clustering result for researchers who do not obviously belong to one group, such as Debajyoti Sinha, Michael J Todd, and Abel Rodriguez. One can observe that Fiber RTCUR detected the "Carroll–Hall" group. However, Fiber RTCUR labels most authors not having a strong connection with Peter Hall, Raymond Carroll, and Jianqing Fan as the "North Carolina" community. Similarly, Chidori RTCUR with $v = 2$ generates the center of the "Carroll–Hall" group as one cluster and categorizes most authors with a lower number of coauthorships and not coauthored with kernel members into the "Fan and Others" group. We infer that the tendency to cluster most members into one group is due to insufficient sampling. The coauthorship tensor is very sparse, with only about 2% of entries being nonzero, so the feature of each node may not be sufficiently extracted from Fiber sampling or Chidori sampling with small v. From the middle two plots, we can observe that most authors in the largest group have very close first and second principal components in the two-dimensional embedding, providing the evidence that the algorithm ignored some nonzero entries for nodes with fewer numbers of connections during the sampling process.

Note that the sampling constant of Fiber sampling should be $r \log d$ times the constant of Chidori sampling in order to access the same amount of data from the original tensor, where d denotes the number of authors in this experiment. So we only test the Chidori sampling with a larger sampling constant on the coauthorship tensor $\mathcal{L}$ for efficiency. The result is shown in the bottom: $v = 6$ for the bottom left and $v = 11$ for the bottom right of Figure 9. Both settings generate the "Carroll–Hall" group with authors having strong ties to Peter Hall and Raymond Carroll, such as Richard Samworth, Hans-Georg Muller, Anastasios Tsiatis, Yuanyuan Ma, Yuedong Wang, Lan Zhou, etc. The "Fan and Others" group is also successfully detected, including coauthors of high-degree node Jianqing Fan, such as Hua Liang, Yingying Fan, Haibo Zhou, Yong Zhou, Jiancheng Jiang, Qiwei Yao, etc. The sizes of the three clusters generated from these two settings are more balanced than the result from RTCUR with smaller v. Therefore, we can conclude that, in this real-world network clustering task, different choices of sampling constant provide the same core members of each group, and the group size is more balanced with a larger amount of sampling data at the

Table 4.3
Runtime comparison (in seconds) of TRPCA algorithms: RGD and RTCUR.

Method	Runtime
RGD [13]	120.5
RTCUR-FF with $\upsilon = 6$	3.526
RTCUR-FC with $\upsilon = 2$	0.571
RTCUR-FC with $\upsilon = 6$	4.319
RTCUR-FC with $\upsilon = 11$	7.177

cost of computation efficiency. Table 4.3 shows the runtime for each algorithm and sampling method.

5. Conclusion. This paper presents a highly efficient algorithm, RTCUR, for large-scale TRPCA problems. RTCUR is developed by introducing a novel inexact low-Tucker-rank tensor approximation via modewise tensor decomposition. The structure of this approximation significantly reduces the computational complexity, resulting in a computational complexity for each iteration of $\mathcal{O}(n^2 d(r \log d)^2)$ with Fiber CUR and $\mathcal{O}(nd(r \log d)^n)$ with Chidori CUR. This is much lower than the minimum of $\mathcal{O}(rd^n)$ required by HOSVD-based algorithms. Numerical experiments on synthetic and real-world datasets also demonstrate the efficiency advantage of RTCUR against other state-of-the-art tensor/matrix RPCA algorithms. Additionally, the fixed sampling variants of RTCUR only require partial information from the input tensor for the TRPCA task.

REFERENCES

[1] H. Abdi and L. J. Williams, *Principal component analysis*, Wiley Interdiscip. Rev. Comput. Stat., 2 (2010), pp. 433–459.

[2] G. Bergqvist and E. G. Larsson, *The higher-order singular value decomposition: Theory and an application [lecture notes]*, IEEE Signal Process. Mag., 27 (2010), pp. 151–154.

[3] T. Bouwmans, S. Javed, H. Zhang, Z. Lin, and R. Otazo, *On the applications of robust PCA in image and video processing*, Proc. IEEE, 106 (2018), pp. 1427–1457.

[4] T. Bouwmans, L. Maddalena, and A. Petrosino, *Scene background initialization: A taxonomy*, Pattern Recognit. Lett., 96 (2017), pp. 3–11.

[5] H. Cai, J.-F. Cai, T. Wang, and G. Yin, *Accelerated structured alternating projections for robust spectrally sparse signal recovery*, IEEE Trans. Signal Process., 69 (2021), pp. 809–821.

[6] H. Cai, J.-F. Cai, and K. Wei, *Accelerated alternating projections for robust principal component analysis*, J. Mach. Learn. Res., 20 (2019), pp. 685–717.

[7] H. Cai, J.-F. Cai, and J. You, *Structured gradient descent for fast robust low-rank Hankel matrix completion*, SIAM J. Sci. Comput., 45 (2023), pp. 1172–1198.

[8] H. Cai, Z. Chao, L. Huang, and D. Needell, *Fast robust tensor principal component analysis via Fiber CUR decomposition*, in Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, IEEE, New York, 2021, pp. 189–197.

[9] H. Cai, K. Hamm, L. Huang, J. Li, and T. Wang, *Rapid robust principal component analysis: CUR accelerated inexact low rank estimation*, IEEE Signal Process. Lett., 28 (2020), pp. 116–120.

[10] H. Cai, K. Hamm, L. Huang, and D. Needell, *Mode-wise tensor decompositions: Multi-dimensional generalizations of CUR decompositions*, J. Mach. Learn. Res., 22 (2021), pp. 1–36.

[11] H. Cai, K. Hamm, L. Huang, and D. Needell, *Robust CUR decomposition: Theory and imaging applications*, SIAM J. Imaging Sci., 14 (2021), pp. 1472–1503.

[12] H. Cai, J. Liu, and W. Yin, *Learned Robust PCA: A Scalable Deep Unfolding Approach for High-Dimensional Outlier Detection*, preprint, arXiv:2110.05649, 2021.

[13] J.-F. Cai, J. Li, and D. Xia, *Generalized low-rank plus sparse tensor estimation by fast Riemannian optimization*, J. Amer. Statist. Assoc., (2022), https://doi.org/10.1080/01621459.2022.2063131.
[14] C. F. Caiafa and A. Cichocki, *Generalizing the column–row matrix decomposition to multi-way arrays*, Linear Algebra Appl., 433 (2010), pp. 557–573.
[15] E. Candes and J. Romberg, *Sparsity and incoherence in compressive sampling*, Inverse Problems, 23 (2007), 969.
[16] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, J. ACM, 58 (2011), pp. 1–37.
[17] J. D. Carroll and J.-J. Chang, *Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition*, Psychometrika, 35 (1970), pp. 283–319.
[18] J. Chambua, Z. Niu, A. Yousif, and J. Mbelwa, *Tensor factorization method based on review text semantic similarity for rating prediction*, Expert Syst. Appl., 114 (2018), pp. 629–638.
[19] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, *Rank-sparsity incoherence for matrix decomposition*, SIAM J. Optim., 21 (2011), pp. 572–596.
[20] Z. Chao, L. Huang, and D. Needell, *HOSVD-based algorithm for weighted tensor completion*, J. Imaging, 7 (2021), p. 110.
[21] J. Chen and Y. Saad, *On the tensor SVD and the optimal low rank orthogonal approximation of tensors*, SIAM J. Matrix Anal. Appl., 30 (2009), pp. 1709–1734.
[22] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-1 and rank-$(r_1, \dots, r_n)$ approximation of higher-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342.
[23] H. Dong, T. Tong, C. Ma, and Y. Chi, *Fast and provable tensor robust principal component analysis via scaled gradient descent*, Inf. Inference, 12 (2023), iaad019.
[24] S. A. Goreĭnov, N. L. Zamarashkin, and E. E. Tyrtyshnikov, *Pseudo-skeleton approximations*, Dokl. Akad. Nauk, 343 (1995), pp. 151–152.
[25] Q. Gu, H. Gui, and J. Han, *Robust tensor decomposition with gross corruption*, Adv. Neural Inform. Process. Syst., 27 (2014), pp. 1422–1430.
[26] K. Hamm and L. Huang, *Perspectives on CUR decompositions*, Appl. Comput. Harmon. Anal., 48 (2020), pp. 1088–1099.
[27] K. Hamm and L. Huang, *Stability of sampling for CUR decompositions*, Found. Data Sci., 2 (2020), 83.
[28] K. Hamm, M. Meskini, and H. Cai, *Riemannian CUR decompositions for robust principal component analysis*, in Topological, Algebraic and Geometric Learning Workshops 2022, PMLR, Cambridge, MA, 2022, pp. 152–160.
[29] F. L. Hitchcock, *Multiple invariants and generalized rank of a p-way matrix or tensor*, J. Math. Phys., 7 (1928), pp. 39–79.
[30] Y. Hu, J.-X. Liu, Y.-L. Gao, and J. Shang, *DSTPCA: Double-sparse constrained tensor principal component analysis method for feature selection*, IEEE/ACM Trans. Comput. Biol. Bioinform., 18 (2021), pp. 1481–1491.
[31] Y. Hu and D. B. Work, *Robust tensor recovery with fiber outliers for traffic events*, ACM Trans. Knowledge Discovery Data, 15 (2020), pp. 1–27.
[32] B. Huang, C. Mu, D. Goldfarb, and J. Wright, *Provable low-rank tensor recovery*, Optim. Online, 4252 (2014), pp. 455–500.
[33] P. Ji and J. Jin, *Coauthorship and citation networks for statisticians*, Ann. Appl. Stat., 10 (2016), pp. 1779–1812.
[34] E. Lopez-Rubio, *Foreground detection in video sequences with probabilistic self-organizing maps*, http://www.lcc.uma.es/∼ezeqlr/fsom/fsom.html.
[35] D. D. Lee and H. S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401 (1999), pp. 788–791.
[36] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian, *Statistical modeling of complex backgrounds for foreground object detection*, IEEE Trans. Image Process., 13 (2004), pp. 1459–1472.
[37] X. Li, D. Xu, H. Zhou, and L. Li, *Tucker tensor regression and neuroimaging analysis*, Statist. Biosci., 10 (2018), pp. 520–545.
[38] Z. Lin and H. Zhang, *Low-Rank Models in Visual Analysis*, Elsevier, New York, 2017, pp. 1–2.
[39] J. Liu, P. Musialski, P. Wonka, and J. Ye, *Tensor completion for estimating missing values in visual data*, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2012), pp. 208–220.

[40] T. Liu, M. Yuan, and H. Zhao, *Characterizing spatiotemporal transcriptome of the human brain via low-rank tensor decomposition*, Statist. Biosci., (2022), pp. 1–29.
[41] Y. Liu, L. Chen, and C. Zhu, *Improved robust tensor principal component analysis via low-rank core matrix*, IEEE J. Sel. Top. Signal Process., 12 (2018), pp. 1378–1389.
[42] 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 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, New York, 2016, pp. 5249–5257.
[43] 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., 42 (2019), pp. 925–938.
[44] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, *A survey of multilinear subspace learning for tensor data*, Pattern Recognit., 44 (2011), pp. 1540–1551.
[45] Y. Luo, Y. Xin, E. Hochberg, R. Joshi, O. Uzuner, and P. Szolovits, *Subgraph augmented non-negative tensor factorization (SANTF) for modeling clinical narrative text*, J. Amer. Med. Inform. Assoc., 22 (2015), pp. 1009–1019.
[46] M. W. Mahoney, M. Maggioni, and P. Drineas, *Tensor-CUR decompositions for tensor-based data*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 957–987.
[47] M. M. Salut and D. Anderson, *Randomized Tensor Robust Principal Component Analysis*, TechRxiv, 2022.
[48] P. Netrapalli, N. U. N, S. Sanghavi, A. Anandkumar, and P. Jain, *Non-convex robust PCA*, in Advances in Neural Information Processing Systems, Vol. 27, MIT Press, Cambridge, MA, 2014.
[49] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai, *A large-scale benchmark dataset for event recognition in surveillance video*, in CVPR 2011, IEEE, New York, 2011, pp. 3153–3160.
[50] A. J. O'Toole, J. Harms, S. L. Snow, D. R. Hurst, M. R. Pappas, J. H. Ayyad, and H. Abdi, *A video database of moving faces and people*, IEEE Trans. Pattern Anal. Mach. Intell., 27 (2005), pp. 812–816.
[51] S. Rabanser, O. Shchur, and S. Günnemann, *Introduction to Tensor Decompositions and Their Applications in Machine Learning*, preprint, arXiv:1711.10781, 2017.
[52] A. Rajwade, A. Rangarajan, and A. Banerjee, *Image denoising using the higher order singular value decomposition*, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2012), pp. 849–862.
[53] S. E. Sofuoglu and S. Aviyente, *A two-stage approach to robust tensor decomposition*, in 2018 IEEE Statistical Signal Processing Workshop (SSP), IEEE, New York, 2018, pp. 831–835.
[54] T. Song, Z. Peng, S. Wang, W. Fu, X. Hong, and P. S. Yu, *Based cross-domain recommendation through joint tensor factorization*, in International Conference on Database Systems for Advanced Applications, Springer-Verlag, Berlin, 2017, pp. 525–540.
[55] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
[56] N. Vaswani and P. Narayanamurthy, *Static and dynamic robust PCA and matrix completion: A review*, Proc. IEEE, 106 (2018), pp. 1359–1379.
[57] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, *Robust face recognition via sparse representation*, IEEE Trans. Pattern Anal. Mach. Intell., 31 (2008), pp. 210–227.
[58] D. Xia, M. Yuan, and C.-H. Zhang, *Statistically optimal and computationally efficient low rank tensor completion from noisy entries*, Ann. Statist., 49 (2021), pp. 76–99.
[59] 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 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, New York, 2014, pp. 3842–3849.
[60] X. Zheng, W. Ding, Z. Lin, and C. Chen, *Topic tensor factorization for recommender system*, Inform. Sci., 372 (2016), pp. 276–293.
[61] H. Zhou, L. Li, and H. Zhu, *Tensor regression with applications in neuroimaging data analysis*, J. Amer. Statist. Assoc., 108 (2013), pp. 540–552.
[62] P. Zhou, C. Lu, Z. Lin, and C. Zhang, *Tensor factorization for low-rank tensor completion*, IEEE Trans. Image Process., 27 (2017), pp. 1152–1163.