# Online Robust Low-Rank Tensor Learning

**Ping Li**[1,2], **Jiashi Feng**[2], **Xiaojie Jin**[2], **Luming Zhang**[3], **Xianghua Xu**[1], **Shuicheng Yan**[4,2]
[1]School of Computer Science and Technology, Hangzhou Dianzi University
[2]Department of Electrical and Computer Engineering, National University of Singapore
[3]Department of Computer and Information, Hefei University of Technology
[4]Qihoo 360 Artificial Intelligence Institute
patriclouis.lee@gmail.com, elefjia@nus.edu.sg, xiaojie.jin@u.nus.edu,
zglumg@gmail.com, xhxu@hdu.edu.cn, eleyans@nus.edu.sg

## Abstract

The rapid increase of multidimensional data (*a.k.a.* tensor) like videos brings new challenges for low-rank data modeling approaches such as dynamic data size, complex high-order relations, and multiplicity of low-rank structures. Resolving these challenges require a new tensor analysis method that can perform tensor data analysis online, which however is still absent. In this paper, we propose an Online Robust Low-rank Tensor Modeling (ORLTM) approach to address these challenges. ORLTM dynamically explores the high-order correlations across all tensor modes for low-rank structure modeling. To analyze mixture data from multiple subspaces, ORLTM introduces a new dictionary learning component. ORLTM processes data streamingly and thus requires quite low memory cost that is independent of data size. This makes ORLTM quite suitable for processing large-scale tensor data. Empirical studies have validated the effectiveness of the proposed method on both synthetic data and one practical task, *i.e.*, video background subtraction. In addition, we provide theoretical analysis regarding computational complexity and memory cost, demonstrating the efficiency of ORLTM rigorously.

## 1 Introduction

The explosion of multidimensional data raises significant demand on efficient data analysis techniques. In recent years, low-rank tensor modeling methods have received increasing attention [Goldfarb and Qin, 2014; Lu *et al.*, 2016] as they can discover intrinsic structure of tensors and provide more transparent and consolidated knowledge of the data. However, many realistic multi-dimensional data (*e.g.*, surveillance videos) are usually generated in a streaming way from dynamic environments, which poses following new challenges: (1) how to develop scalable methods that can process large-scale dynamic tensors efficiently; (2) how to deal with data contamination such as noise or malicious outliers in the modeling process.

To handle noisy tensors, two *batch* robust tensor methods were developed: High-Order Robust Principal Component Analysis (HORPCA) [Goldfarb and Qin, 2014] and Tensor RPCA [Lu *et al.*, 2016]. They both generalize RPCA [Candès *et al.*, 2011] from matrix cases to tensors, *i.e.*, solving the tensor RPCA problem $\min_{\mathcal{Z},\mathcal{E}} \|\mathcal{Z}\|_*+\|\mathcal{E}\|_1$ with $\mathcal{X} = \mathcal{Z}+\mathcal{E}$, to learn the inherent low-rank structure. Although they can handle noisy data, those methods are not scalable to large-scale data due to their high memory cost which increases rapidly with the data size, and always seek a static low-rank component that cannot model dynamics of streaming data thus providing inferior performance. A proper method to analyze large-scale noisy tensor data is still absent.

In this work, we propose an Online Robust Low-rank Tensor Modeling (ORLTM) method for learning low-rank structures of tensors from streaming noisy tensor data. Different from batch approaches, ORLTM isolates the low-rank component into two factors in each mode $m$, *i.e.*, $\mathbf{Z}^m = \mathbf{W}^m\mathbf{R}^m$ where $\mathbf{W}^m$ models the low-rank subspace basis and $\mathbf{R}^m$ incorporates the corresponding tensor data representation coefficients *w.r.t.* $\mathbf{W}^m$. Here, the basis and data representations are decoupled. Thus, the basis size is typically small and can be stored with low memory cost while data representation can be updated online. In this way, ORLTM saves the memory cost dramatically without performance drop.

Existing tensor RPCA approaches generally assume data reside in a *single* low-rank subspace. This may not hold for many realistic tensor data with complex inherent structures, *e.g.*, those drawn from a union of multiple subspaces. Previous methods as well as the vanilla ORLTM are less competent in such circumstances. Hence, we further introduce a dictionary learning component to ORLTM. Dictionary provides a more flexible set of basis to represent the low-rank component of complex tensors. Instead of seeking a single set of basis, through learning proper dictionaries, ORLTM can capture sophisticated low-rank structures (*e.g.*, mixture of multiple low-rank components) for better tensor modeling.

ORLTM has wide applications in large-scale and dynamic tensor data analysis. In this paper, we empirically examined its performance on both synthetic data and video background subtraction, the results of which have well justified the effectiveness of ORLTM.

## 2 Related Work

Low-rank models are useful for robust data recovery. A typical one is Robust Principal Component Analysis (RPCA) [Candès *et al.*, 2011] that decomposes a given matrix into a low-rank component and a sparse component. Another is Low-Rank Representation (LRR) [Liu *et al.*, 2013] which considers the data drawn from a union of multiple subspaces. To this end, many variants have emerged, *e.g.*, [Liu and Yan, 2011] employed both observed and hidden data to enhance LRR; [Yin *et al.*, 2015] developed a dual graph regularized LRR. However, these methods require tremendous memory to store all the samples. Hence, some online learning methods were developed, *e.g.*, [Feng *et al.*, 2013] proposed an Online Robust PCA (ORPCA) via stochastic optimization; [Zhan *et al.*, 2016] studied online RPCA based on the recursive projected compressive sensing framework; [Shen *et al.*, 2016] designed an online LRR implementation with convergence guarantees. Nevertheless, they fail to exploit the low-dimensional tensor structures.

Robust tensor models can utilize information across all the modes. Two principled forms are CANDECOMP/PARAFAC (CP) decomposition and Tucker decomposition [Sun *et al.*, 2008]. For example, [Zhou *et al.*, 2016] proposed an accelerated CP method for dynamic tensors. Based on the latter, Higher-Order RPCA (HORPCA) was presented in [Goldfarb and Qin, 2014]. Moreover, [Zhang *et al.*, 2013] proposed a low-rank three-order tensor recovery method for rectification and alignment, while [Lu *et al.*, 2016] proposed one tensor RPCA (TRPCA) method for image denoising using the framework in [Kilmer and Martin, 2011]. Nonetheless, these methods need large memory for batch optimization and cannot handle samples sequentially. In addition, there are rare works concerning robust online tensor analysis, and the ever-known one is Online Stochastic Tensor Decomposition (OSTD) [Sobral *et al.*, 2015], which neglects the data with the mixture structure of multiple subspaces.

## 3 Notations and Preliminaries

Tensors are denoted by calligraphic letters, *e.g.*, $\mathcal{A}$, matrices by boldface uppercase letters, *e.g.*, $\mathbf{A}$, vectors by boldface lowercase letters, *e.g.*, $\mathbf{a}$. The *order* or *mode* of a tensor equals the number of dimensions. For a third-order tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, its $(i,j,k)$-th *entry* is $x_{ijk}$, and its *fiber* is a column vector $\mathcal{X}(i,j,:)$; $\mathcal{X}(i,:,:)$ is the $i$-th *horizontal slice*; $\mathcal{X}(:,j,:)$ is the $j$-th *lateral slice*; $\mathcal{X}(:,:,k)$ or $\mathbf{X}_k$ is the $k$-th *frontal slice*. For the matrix, $\mathbf{A}_{ij}$ denotes the $(i,j)$-th entry, $\mathbf{A}^m$ without bracket denotes a traditional matrix.

**Tensor Unfolding** [Kolda and Bader, 2009] The mode-$m$ unfolding of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_M}$, $\mathbf{A}^{(m)}$, is obtained by arranging the mode-$m$ fibers in the matrix columns, *i.e.*, $unfold_m(\mathcal{A}) = \mathbf{A}^{(m)} \in \mathbb{R}^{n_{\bar{m}} \times n_m}$, where $n_{\bar{m}} = \prod_{j \neq m, j=1}^{M} n_j$ and $\prod_{j=1}^{M} n_j = n_1 \times n_2 \times \ldots \times n_M$.

**Tensor Folding** [Kolda and Bader, 2009] Folding the mode-$m$ unfolding of tensor $\mathcal{A}$ is defined as $fold(\mathbf{A}^{(m)}) = \mathcal{A}_m$, which returns the corresponding tensor $\mathcal{A}$ in mode $m$.

**Tensor Vectorization** [Kolda and Bader, 2009] The vectorization of $\mathcal{A}$ is denoted by $vec(\mathcal{A})$, which stacks the elements in a column vector. For mode-$m$ unfolding, $vec(\mathcal{A}_m)$ (*a.k.a.* $vec(\mathbf{A}^{(m)})$) is done by aligning the columns of mode-$m$ unfolding into a long column vector.

**Mode-*m* Product** [Goldfarb and Qin, 2014] The mode-$m$ product of tensor $\mathcal{A}$ and matrix $\mathbf{U}^m \in \mathbb{R}^{n_m \times n_c}$ on mode $m$ is $\mathcal{A} \times_m \mathbf{U}$, which is defined as $(\mathcal{A} \times_m \mathbf{U}^m)^{(m)} = \mathbf{A}^{(m)} \mathbf{U}^m$.

**Tensor Norms** [Goldfarb and Qin, 2014] The $\ell_1$-norm is $\|\mathcal{A}\|_1 = \|vec(\mathcal{A})\|_1 = \sum_{ijk} |a_{ijk}|$, and the Frobenius norm is $\|\mathcal{A}\|_F = \sqrt{(vec(A)^\top vec(A))} = \sqrt{(\sum_{ijk} a_{ijk}^2)}$. These norms can degenerate to the matrix or vector norms.

**Tensor Rank** [Goldfarb and Qin, 2014] The mode-$m$ rank of a tensor $\mathcal{A}$ denoted by $\text{rank}_m(\mathcal{A})$ is the column rank of $\mathbf{A}^{(m)}$, and the set of $M$ mode-$m$ ranks is called Tucker-rank. However, it is an *NP*-hard problem and one often uses its convex surrogate CTrank($\mathcal{A}$) in practice [Hillar and Lim, 2013; Yu *et al.*, 2015]. CTrank sums $M$ nuclear norms of the mode-$m$ unfoldings, *i.e.*, $\text{CTrank}(\mathcal{A}) := \sum_{m=1}^{M} \|\mathbf{A}^{(m)}\|_*$.

## 4 Online Robust Low-Rank Tensor Modeling

### 4.1 Formulation of ORLTM

We now introduce the objective formulation by starting with reviewing the batch-based High-Order RPCA (HORPCA) [Goldfarb and Qin, 2014]. This method factorizes the input tensor into a low-rank component plus a sparse noise component accounting for gross corruption or outliers, and its slice-wise model is formulated as

$$\min_{\mathcal{L},\mathcal{E}} \sum_{m=1}^{M} \|\mathbf{L}^{(m)}\|_* + \lambda_1 \|\mathcal{E}\|_1, \text{ s. t. } \mathcal{X} = \mathcal{L} + \mathcal{E}, \qquad (1)$$

where $\lambda_1 > 0$, $\mathcal{L} = fold(\mathbf{L}^{(m)})$, $\sum_{m=1}^{M} \|\mathbf{L}^{(m)}\|_* = \text{CTrank}(\mathcal{L})$, $\mathcal{X}, \mathcal{L}, \mathcal{E} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_M}$ are the observed noisy tensor, its low-rank component and the sparse component, respectively. On different mode-$m$ unfoldings $\mathbf{X}^{(m)}$, the problem (1) leads to different low-rank components and sparse components. Thus we adopt the variable-splitting technique and introduce a set of auxiliary tensor variables, *i.e.*, $\{\mathcal{L}_m\}_{m=1}^{M}$, $\{\mathcal{E}_m\}_{m=1}^{M}$, for $\mathcal{L}$ and $\mathcal{E}$ respectively. So the problem (1) can be reformulated as

$$\begin{aligned} \min_{\substack{\mathcal{L}_m, \mathcal{E}_m \\ m=1,2,\cdots,M}} & \sum_{m=1}^{M} \|\mathbf{L}^{(m)}\|_* + \lambda_1 \|\mathbf{E}^{(m)}\|_1, \\ \text{s. t.} \quad & \mathbf{X}^{(m)} = \mathbf{L}^{(m)} + \mathbf{E}^{(m)}, m = 1, \ldots, M, \\ & \mathcal{L}_m = fold(\mathbf{L}^{(m)}),\ \mathcal{E}_m = fold(\mathbf{E}^{(m)}), \end{aligned} \qquad (2)$$

where $\mathcal{X} = fold(\mathbf{X}^{(m)})$, and each unfolding $\mathbf{X}^{(m)}$ in all modes should return the common tensor $\mathcal{X}$ through the $fold(\cdot)$ operator. The tensors $\mathcal{L}$ and $\mathcal{E}$ can be approximated by average pooling, *i.e.*, $\mathcal{L} = \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}_m$, $\mathcal{E} = \frac{1}{M} \sum_{m=1}^{M} \mathcal{E}_m$.

However, the constraints in (2) enforce the unfolding matrix to reside in a single low-rank subspace, failing to consider the situation where the data are with the mixture structure of several subspaces. Thus, this strict constraint hardly holds when data are drawn from a union of multiple subspaces, leading to degenerated performance. Therefore, to better discover the robust low-rank structure, we introduce a dictionary

$\mathbf{D}^{(m)} \in \mathbb{R}^{n_{\bar{m}} \times n_m}$. The low-rank property is preserved by enforcing nuclear norm on $\mathbf{Z}^m \in \mathbb{R}^{n_m \times n_{\bar{m}}}$, *i.e.*, $\|\mathbf{Z}^m\|_*$. As the inequality $\text{rank}(\mathbf{D}^{(m)}\mathbf{Z}^m) \leq \text{rank}(\mathbf{Z}^m)$ always holds, minimizing the nuclear norm of $\mathbf{Z}^m$ amounts to bounding the rank of the clean data $\mathbf{L}^{(m)}$ if it suffices to $\mathbf{L}^{(m)} \triangleq \mathbf{D}^{(m)}\mathbf{Z}^m$, hence a low-rank structure. Thus, we can reach the following function

$$\begin{aligned} \min_{\substack{\mathbf{Z}^m, \mathbf{E}^{(m)} \\ m=1,2,\cdots,M}} & \sum_{m=1}^{M} \|\mathbf{Z}^m\|_* + \lambda_1 \|\mathbf{E}^{(m)}\|, \\ \text{s. t.} \quad & \mathbf{X}^{(m)} = \mathbf{D}^{(m)}\mathbf{Z}^m + \mathbf{E}^{(m)}, m = 1, \ldots, M. \end{aligned} \tag{3}$$

The product of $\mathbf{D}^{(m)}$ and $\mathbf{Z}^m$ yields the unfolding $\mathbf{L}^{(m)}$. To dynamically handle streaming data, we relax the constrains for online optimization by treating them as quadratic penalty terms, leading to

$$\begin{aligned} \min_{\substack{\mathbf{Z}^m, \mathbf{E}^{(m)} \\ m=1,2,\cdots,M}} & \frac{1}{2} \sum_{m=1}^{M} \|\mathbf{X}^{(m)} - \mathbf{D}^{(m)}\mathbf{Z}^m - \mathbf{E}^{(m)}\|_F^2 \\ & + \lambda_1 \|\mathbf{E}^{(m)}\|_1 + \lambda_2 \|\mathbf{Z}^m\|_*, \end{aligned} \tag{4}$$

where $\lambda_2 > 0$ governs the role of the nuclear norm term.

To process samples on the fly, we propose to employ the bi-factor form $\mathbf{Z}^m = \mathbf{W}^m\mathbf{R}^m$ [Srebro *et al.*, 2004], where $\mathbf{W}^m \in \mathbb{R}^{n_m \times p}$ and $\mathbf{R}^m \in \mathbb{R}^{p \times n_{\bar{m}}}$ with $p \ll \min(n_m, n_{\bar{m}})$. Thus the rank of $\mathbf{Z}^m$ is upper bounded by $p$. As indicated by [Fazel *et al.*, 2001; Recht *et al.*, 2010], minimizing $\|\mathbf{Z}^m\|_*$ is equivalent to minimizing $\|\mathbf{W}^m\|_F^2$ and $\|\mathbf{R}^m\|_F^2$ simultaneously. So the unconstrained function in (4) can be reformulated as a non-convex optimization problem

$$\begin{aligned} \min_{\substack{\mathbf{W}^m, \mathbf{R}^m, \mathbf{E}^{(m)} \\ m=1,2,\ldots,M}} & \frac{1}{2} \sum_{m=1}^{M} \|\mathbf{X}^{(m)} - \mathbf{D}^{(m)}\mathbf{W}^m\mathbf{R}^m - \mathbf{E}^{(m)}\|_F^2 \\ & + \lambda_1 \|\mathbf{E}^{(m)}\|_1 + \frac{\lambda_2}{2}(\|\mathbf{W}^m\|_F^2 + \|\mathbf{R}^m\|_F^2), \end{aligned} \tag{5}$$

which is the objective function of ORLTM, and updating the entries in $\mathbf{Z}^m$ amounts to sequentially updating the rows of $\mathbf{W}^m$ and the columns of $\mathbf{R}^m$.

The objective function (5) reveals a fact that the sizes of unfolding matrices increase with $n_M$ and the dictionary $\mathbf{D}^{(m)}$ can only be partially accessed in each iteration. Besides, the row vectors of $\mathbf{W}^{(m)}$ are coupled together as being left multiplied by the dictionary. To tackle these issues, we introduce another set of auxiliary variables $\mathbf{B}^{(m)} = \mathbf{D}^{(m)}\mathbf{W}^m \in \mathbb{R}^{n_{\bar{m}} \times p}$, $m = 1, 2, \ldots, M$, to approximate the recovery part $(\mathbf{X}^{(m)} - \mathbf{E}^{(m)})$ by $\mathbf{B}^{(m)}\mathbf{R}^m$. This suggests that $\{\mathbf{B}^{(m)}\}_{m=1}^M$ can be treated as *Reinforced Basis Dictionaries* while $\{\mathbf{R}^m\}_{m=1}^M$ are the coefficients or low-dimensional representation. Compared with $\mathbf{D}^{(m)}$, the dictionary $\mathbf{B}^{(m)}$ is iteratively strengthened by respecting the two factored low-rank components of $\mathbf{Z}^m$. Therefore, we get an equivalent reformulation

$$\begin{aligned} \min_{\substack{\mathbf{B}^{(m)}, \mathbf{W}^m, \mathbf{R}^m, \mathbf{E}^{(m)} \\ m=1,2,\ldots,M}} & \sum_{m=1}^{M} \frac{1}{2} \|\mathbf{X}^{(m)} - \mathbf{B}^{(m)}\mathbf{R}^m - \mathbf{E}^{(m)}\|_F^2 \\ & + \lambda_1 \|\mathbf{E}^{(m)}\|_1 + \frac{\lambda_2}{2}(\|\mathbf{W}^m\|_F^2 + \|\mathbf{R}^m\|_F^2) \\ & + \frac{\lambda_3}{2} \|\mathbf{B}^{(m)} - \mathbf{D}^{(m)}\mathbf{W}^m\|_F^2, \end{aligned} \tag{6}$$

where $\lambda_3 > 0$ is used to control the reconstruction quality of the basis $\mathbf{B}^{(m)}$. This formulation is not only more appropriate for learning from streaming data, but also more informative since it explicitly models the basis of the union of multiple subspaces in all modes of tensor data. Thus, it achieves more accurate tensor subspace recovery and better tensor low-rank modeling.

## 4.2 Online Optimization for ORLTM

This section develops an efficient online optimization algorithm for optimizing the objective function in (6). This algorithm employs the stochastic optimization technique. First, two functions in mode $m$ are defined as

$$\begin{aligned} \tilde{\ell}(\mathbf{x}^m, \mathbf{B}^{(m)}, \mathbf{r}^m, \mathbf{e}^m) &\triangleq \frac{1}{2} \|\mathbf{x}^m - \mathbf{B}^{(m)}\mathbf{r}^m - \mathbf{e}^m\|_2^2 \\ &\quad + \lambda_1 \|\mathbf{e}^m\|_1 + \frac{\lambda_2}{2} \|\mathbf{r}^m\|_2^2, \\ \ell(\mathbf{x}^m, \mathbf{B}^{(m)}) &= \min_{\mathbf{r}^m, \mathbf{e}^m} \tilde{\ell}(\mathbf{x}^m, \mathbf{B}^{(m)}, \mathbf{r}^m, \mathbf{e}^m), \end{aligned} \tag{7}$$

$$\begin{aligned} \tilde{h}(\mathbf{D}^{(m)}, \mathbf{B}^{(m)}, \mathbf{W}^m) &\triangleq \sum_{i=1}^{N} \frac{\lambda_2}{2} \|\mathbf{w}_i^m\|_2^2 \\ &\quad + \frac{\lambda_3}{2} \|\mathbf{B}^{(m)} - \sum_{i=1}^{N} \mathbf{d}_i^m \mathbf{w}_i^m\|_F^2, \\ h(\mathbf{D}^{(m)}, \mathbf{B}^{(m)}) &= \min_{\mathbf{W}^m} \tilde{h}(\mathbf{D}^{(m)}, \mathbf{B}^{(m)}, \mathbf{W}^m), \end{aligned} \tag{8}$$

where $\mathbf{x}^m$, $\mathbf{e}^m$, $\mathbf{d}^m$, $\mathbf{r}^m$ are column vectors of matrices $\mathbf{X}^{(m)}, \mathbf{E}^{(m)}, \mathbf{D}^{(m)} \in \mathbb{R}^{n_{\bar{m}} \times n_m}$, $\mathbf{R}^m \in \mathbb{R}^{p \times n_{\bar{m}}}$, respectively, and $\mathbf{w}^m$ is the row vector of $\mathbf{W}^m \in \mathbb{R}^{n_m \times p}$. These formulations allow us to rewrite (6) as

$$\min_{\mathbf{B}^{(m)}} \min_{\substack{\mathbf{E}^{(m)}, \\ \mathbf{W}^m, \mathbf{R}^m}} \sum_{i=1}^{N} \tilde{\ell}(\mathbf{x}_i^m, \mathbf{B}_i^{(m)}, \mathbf{r}_i^m, \mathbf{e}_i^m) + \tilde{h}(\mathbf{D}^{(m)}, \mathbf{B}^{(m)}, \mathbf{W}^m), \tag{9}$$

which amounts to minimizing the empirical loss function:

$$f_N(\mathbf{B}^{(m)}) \triangleq \frac{1}{N} \sum_{i=1}^{N} \ell(\mathbf{x}_i^m, \mathbf{B}^{(m)}) + \frac{1}{N} h(\mathbf{D}^{(m)}, \mathbf{B}^{(m)}).$$

Now, we describe how to sequentially optimize the variables in (6), and we adopt the alternating method, *i.e.*, solving one variable while holding the rest. The whole algorithmic framework is given in Algorithm 1, which consists of following variable updatings at each iteration $t$.

**Computing $\mathbf{r}_t^m$, $\mathbf{e}_t^m$.** Given $\mathbf{B}_{t-1}^{(m)}$ in the previous iteration for mode-$m$ unfolding, we obtain the optimal $\{\mathbf{r}_t^m, \mathbf{e}_t^m\}$ by solving

$$\min_{\mathbf{r}^m, \mathbf{e}^m} \tilde{\ell}(\mathbf{x}_t^m, \mathbf{B}_{t-1}^{(m)}, \mathbf{r}^m, \mathbf{e}^m). \tag{10}$$

The above problem has a closed-form solution $\mathbf{r}_t^m$ when $\mathbf{e}^m$ is fixed, *i.e.*,

$$\mathbf{r}^m = (\mathbf{B}_{t-1}^{(m)\top}\mathbf{B}_{t-1}^{(m)} + \lambda_2\mathbf{I}_p)^{-1}\mathbf{B}_t^{(m)\top}(\mathbf{x}_t^m - \mathbf{e}), \quad (11)$$

and *w.r.t.* fixed $\mathbf{r}^m$, the local minimizer of $\mathbf{e}^m$ is given by the soft-thresholding operator [Hale *et al.*, 2008]:

$$\mathbf{e}^m = \mathcal{S}_{\lambda_1}[\mathbf{x}_t^m - \mathbf{B}_{t-1}^{(m)}\mathbf{r}^m]. \quad (12)$$

Both $\mathbf{r}_t^m$ and $\mathbf{e}_t^m$ can be optimized by the coordinate descent algorithm [Wright, 2015] in an efficient way.

**Computing $\mathbf{w}_t^m$.** Define an accumulation matrix $\mathbf{G}_{t-1}^{(m)} = \sum_{i=1}^{t-1}\mathbf{d}_i^m\mathbf{w}_i^m \in \mathbb{R}^{n_{\bar{m}}\times p}$, where $\mathbf{G}_0^m = 0$, and then we can figure $\mathbf{w}_t^m$ out by minimizing

$$\begin{aligned}&\tilde{\ell}_2(\mathbf{d}_t^m, \mathbf{B}_{t-1}^{(m)}, \mathbf{G}_{t-1}^{(m)}, \mathbf{w}^m)\\ &\triangleq \frac{\lambda_2}{2}\|\mathbf{w}^m\|_2^2 + \frac{\lambda_3}{2}\|\mathbf{B}_{t-1}^{(m)} - \mathbf{G}_{t-1}^{(m)} - \mathbf{d}_t^m\mathbf{w}^m\|_F^2,\end{aligned} \quad (13)$$

which leads to the solution in closed form:

$$\mathbf{w}_t^m = (\|\mathbf{d}_t^m\|_2^2 + \tfrac{\lambda_2}{\lambda_3})^{-1}\mathbf{d}_t^{m\top}(\mathbf{B}_{t-1}^{(m)} - \mathbf{G}_{t-1}^{(m)}). \quad (14)$$

While $\mathbf{w}_t^m$ depends on the entire dictionary $\mathbf{D}^{(m)}$, we only have access to the current atom $\mathbf{d}_t^m$, which thus desires a proper estimation to approximately solve each $\mathbf{w}_t^m$. The intuition behind the solution is that after observing the $t$-th atom $\mathbf{d}_t^m$, we only update $\mathbf{w}_t^m$ by keeping others, and each $\mathbf{w}_t^m$ is sequentially updated only once after revealing all the atoms. The strategy is essentially the one-pass block coordinate descent algorithm, and can be flexibly refined as a multiple-pass version in demanding practical applications.

**Computing $\mathbf{B}_t^{(m)}$.** The reinforced basis dictionary can be updated by minimizing the following surrogate function

$$\begin{aligned}h_t(\mathbf{B}^{(m)}) \triangleq &\frac{1}{t}\sum_{i=1}^{t}\tilde{\ell}(\mathbf{x}_i^m, \mathbf{B}^{(m)}, \mathbf{r}_i^m, \mathbf{e}_i^m)\\ &+\frac{\lambda_2}{2t}\sum_{i=1}^{t}\|\mathbf{w}_i^m\|_2^2 + \frac{\lambda_3}{2t}\|\mathbf{B}^{(m)} - \mathbf{G}_t^{(m)}\|_F^2.\end{aligned} \quad (15)$$

One key to obtain efficient dictionary updates is that the surrogate $h_t(\cdot)$ can be summarized by a few sufficient statistics updated iteratively, *i.e.*, it is possible to describe $h_t(\cdot)$ without explicitly storing previous samples [Mensch *et al.*, 2016]. Actually, we can define two accumulation matrices:

$$\mathbf{\Phi}_t^m = \sum_{i=1}^{t}\mathbf{r}_i^m\mathbf{r}_i^{m\top},\ \mathbf{\Theta}_t^{(m)} = \sum_{i=1}^{t}(\mathbf{x}_i^m - \mathbf{e}_i^m)\mathbf{r}_i^{m\top}.$$

Then, the solution to (15) is expressed by

$$\mathbf{B}_t^{(m)} = (\mathbf{\Theta}_t^{(m)} + \lambda_3\mathbf{G}_t^{(m)})(\mathbf{\Phi}_t^m + \lambda_3\mathbf{I}_p)^{-1}, \quad (16)$$

where $\mathbf{\Phi}_t^m \in \mathbb{R}^{p\times p}$ and $\mathbf{\Theta}_t^{(m)} \in \mathbb{R}^{n_{\bar{m}}\times p}$ are independent of the sample size $N$. This makes our approach potentially scalable to large data size.

In summary, the pipeline of our ORLTM method is described in Algorithm 1. For an $M$-th order tensor $\mathcal{X}$, $n_M$ indicates the number of *unit sub-tensor* samples, *i.e.*, one ($M$-1)-th order sub-tensor. If the number of *unit* sub-tensor samples is $n$, *i.e.*, the entire tensor size, then the number of sub-tensor samples is $N = \frac{n}{n_M}$. At each iteration, ORLTM admits one sub-tensor, either *single* unit sub-tensor sample ($n_M = 1$) or *multiple* unit sub-tensor samples.

**Algorithm 1** Online Robust Low-Rank Tensor Modeling

**Input:** Observed $M$-th order tensor $\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times\ldots\times n}$ and pre-given dictionary $\mathcal{D}$, tradeoff parameters $\{\lambda_1, \lambda_2, \lambda_3\}$, target rank $p$, sub-tensor size $n_M$.
**Output:** $\mathcal{L}, \mathcal{B}, \mathcal{E}, \{\mathbf{W}^m, \mathbf{R}^m\}_{m=1}^M$.

**Initialize:** Set all entries of $\{\mathcal{L}, \mathcal{E}\} \in \mathbb{R}^{n_1\times n_2\times\ldots\times n}$, $\mathbf{W}^m \in \mathbb{R}^{n_m\times p}$, $\mathbf{R}^m \in \mathbb{R}^{p\times n_m}$, $\mathbf{G}_0^{(m)}$, $\mathbf{\Theta}_0^{(m)}$, $\mathbf{\Phi}_0^m$ to zero, initialized $\mathcal{B} \in \mathbb{R}^{n_1\times n_2\times\ldots\times p}$, $N = \frac{n}{n_M}$.
**for** $t = 1$ to $N$ **do**
  **for** $m = 1$ to $M$ **do**
    Access the $t$-th sample $\mathbf{x}_t^m$ by $vec(\mathbf{X}_t^{(m)})$.
    Reveal the $t$-th atom $\mathbf{d}_t^m$ by $vec(\mathbf{D}_t^{(m)})$.
    Obtain the coefficients $\mathbf{r}_t^m$ and sparse vectors $\mathbf{e}_t^m$ by optimizing $\tilde{\ell}(\mathbf{x}_t^m, \mathbf{B}_{t-1}^{(m)}, \mathbf{r}^m, \mathbf{e}^m)$ and utilize coordinate descent algorithm for Eqs. (11)(12) to derive the solutions.
    Compute the coefficients $\mathbf{w}_t^m$ by minimizing $\tilde{\ell}_2(\mathbf{d}_t^m, \mathbf{B}_{t-1}^{(m)}, \mathbf{G}_{t-1}^m, \mathbf{w}^m)$, leading to Eq. (14).
    Update the accumulation matrices as
    $\mathbf{G}_t^{(m)} \leftarrow \mathbf{G}_{t-1}^{(m)} + \mathbf{d}_t^m\mathbf{w}_t^m$,
    $\mathbf{\Theta}_t^{(m)} \leftarrow \mathbf{\Theta}_{t-1}^{(m)} + (\mathbf{x}_t^m - \mathbf{e}_t^m)\mathbf{r}_t^{m\top}$,
    $\mathbf{\Phi}_t^m \leftarrow \mathbf{\Phi}_{t-1}^m + \mathbf{r}_t^m\mathbf{r}_t^{m\top}$.
    Update the dictionary $\mathbf{B}_t^{(m)}$ by Eq. (16).
    Update low-rank and sparse components by $\mathbf{L}_t^{(m)} \leftarrow \mathbf{B}_t^{(m)}\mathbf{r}_t^m$ and $\mathbf{E}_t^{(m)} \leftarrow \mathbf{e}_t^m$, respectively.
  **end for**
**end for**
Obtain low-rank tensor $\mathcal{L}$, sparse tensor $\mathcal{E}$, reinforced dictionary tensor $\mathcal{B}$ by average pooling on mode-$m$ foldings, *i.e.*, $\mathcal{L} = \frac{1}{M}\sum_{m=1}^M fold(\mathbf{L}^{(m)})$, $\mathcal{E} = \frac{1}{M}\sum_{m=1}^M fold(\mathbf{E}^{(m)})$, $\mathcal{B} = \frac{1}{M}\sum_{m=1}^M fold(\mathbf{B}^{(m)})$.

### 4.3 Complexity Analysis

**Computational Complexity.** There are four variables in each iteration. First, it is cheap to compute $\{\mathbf{r}_t^m, \mathbf{e}_t^m\}$ as one may utilize the block coordinate descent method in [Richtárik and Takáč, 2014] which enjoys linear convergence. Second, $\mathbf{w}_t^m$ needs $\mathcal{O}(n_{\bar{m}}p)$ to conduct matrix-vector multiplication, where the rank $p \ll \min(n_{\bar{m}}, n_m)$. Besides, the sub-tensor size $n_M$ is usually quite small. Third, it requires $\mathcal{O}(n_{\bar{m}}p)$ to update accumulation matrices $\mathbf{G}_t^{(m)}$, $\mathbf{\Theta}_t^{(m)}$, $\mathbf{\Phi}_t^{(m)}$. Fourth, computing $\mathbf{B}_t^m$ costs $\mathcal{O}(n_{\bar{m}}p^2)$. Therefore, the total computational cost is limited.

**Memory Cost.** Lower memory cost is a very promising advantage of ORLTM. On the one hand, we need $\mathcal{O}(n_{\bar{m}}p)$ to load $\mathbf{B}_t^{(m)}$ and $\mathbf{x}_t^m$ to obtain $\{\mathbf{r}_t^m, \mathbf{e}_t^m\}$, and also $\mathcal{O}(n_{\bar{m}}p)$ to exploit $\mathbf{B}_t^{(m)}$, $\mathbf{G}_t^{(m)}$ for calculating $\mathbf{w}_t^m$. On the other hand, because the history information of $h_t(\mathbf{B}_t^{(m)})$ in (15) has been recorded by the accumulation matrices $\mathbf{G}_t^{(m)}$, $\mathbf{\Theta}_t^{(m)}$, $\mathbf{\Phi}_t^{(m)}$, it only costs at most $\mathcal{O}(n_{\bar{m}}p)$. Hence, ORLTM only desires $\mathcal{O}(n_{\bar{m}}p)$ memory cost for every iteration.

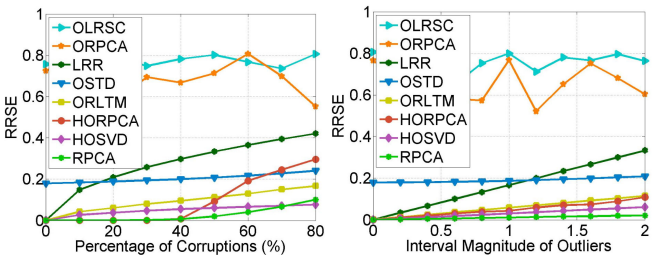



Figure 1: RRSE on different corrupted entries on synthetic data. Left: RRSE vs Corruption Percentage ($H = 1$); Right: RRSE vs Outlier Intervals ($2|H|$ with $\eta = 50\%$).

# 5 Experiments

## 5.1 Synthetic Data

To generate synthetic data with low-rank structure, we construct an authentic tensor $\mathcal{S}$ of size $50 \times 50 \times 30$ by *rank*-3 factor matrices as in [Sobral *et al.*, 2015], *e.g.*, $\mathbf{S}^m \in \mathbb{R}^{50\times 3}$ (here $m$ denotes mode). Each factor matrix consists of three components $[\sin(\frac{2\pi m i_m}{50}), \cos(\frac{2\pi m i_m}{50}), \text{sgn}(\sin(0.5\pi i_m))]$, where $i_m$ denotes the column element index in mode $m$, and the product of two factor matrices yields a frontal slice of *mode*-3. We randomly corrupt entries of the tensor by small noise from distribution $N(0, 0.01)$ and outliers from uniform distribution $U(-|H|, |H|)$ ($H$ is interval bound magnitude). For evaluation, we use Root Relative Square Error (RRSE) as the measure criterion calculated by $\frac{\|\hat{\mathcal{S}}-\mathcal{S}\|_F}{\|\mathcal{S}\|_F}$, where $\hat{\mathcal{S}}$ is the estimated low-rank tensor. In all, we evaluate four batch methods, *i.e.*, RPCA [Candès *et al.*, 2011], LRR [Liu *et al.*, 2013; 2017], HOSVD [Goldfarb and Qin, 2014], and HORPCA [Goldfarb and Qin, 2014], and also four online methods ($n_M$=1), *i.e.*, ORPCA [Feng *et al.*, 2013], OLRSC [Shen *et al.*, 2016], OSTD [Sobral *et al.*, 2015], and our ORLTM approach. Note that batch approaches have the hindsight knowledge of all the streaming data and provides performance upper bound for online counterparts. For compared methods, we use the default parameters in their papers, and the target rank $p$ is set to the ground truth 3. For ORLTM, $\lambda_1 = 0.5/\sqrt{\log(n_1 n_2)}$, $\lambda_2 = 1$, $\lambda_3 = \lambda_1\sqrt{\log(t)}$. The comparison results with a varying corruption percentage $\eta$ from 0 to $80\%$ and different outlier intervals $2|H|$ in $[0 : 0.2 : 2]$ are plotted in Fig. 1. The figure shows that ORLTM indeed outperforms all online methods no matter how the corruption percentage and the outlier interval vary, and is comparable to batch approaches. Besides, our method is surprisingly better than LRR consistently and better than RPCA when the corruption exceeds $55\%$, which well justifies the superiority of ORLTM by exploiting the low-rank structure in all tensor modes.

In addition, we provide quantitative analysis with different sizes ($n_3$) for online methods in Table 1. As seen from the table, the performance of compared methods is improved by increasing tensor size, which demonstrates that more robust tensor recovery could be achieved by taking in more samples sequentially. Meanwhile, our method still performs the best no matter how the tense size varies.

Table 1: RRSE of Online Methods with Different Tensor Sizes ($\eta = 50\%$, $H = 1$, $n_1 = 50$, $n_2 = 50$).

| $n_3$ | ORPCA | OLRSC | OSTD | **ORLTM** |
|---|---|---|---|---|
| 100 | 0.5456 | 0.7104 | 0.1183 | **0.0667** |
| 500 | 0.6571 | 0.5685 | 0.0580 | **0.0324** |
| 1000 | 0.5220 | 0.5352 | 0.0430 | **0.0234** |
| 5000 | 0.4316 | 0.4786 | 0.0224 | **0.0130** |

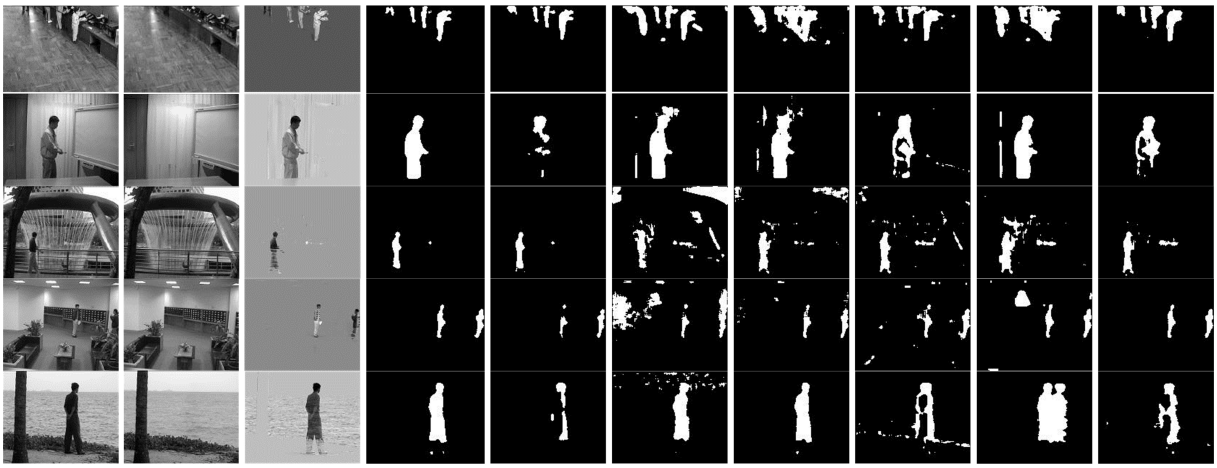
## 5.2 Video Background Subtraction

An input video is treated as a long third-order tensor, of which each frontal slice denotes a frame and multiple frames constitute a sub-tensor. We model the background (BG, *i.e.*, static information) by the low-rank component $\mathcal{L}$ while the foreground (FG, *i.e.*, moving objects) is pushed to the sparse component $\mathcal{E}$. We test on the I2R database [Li *et al.*, 2004] which involves various indoor and outdoor environments. We use *eight* video sequences containing over 15,000 frames with sizes ranging from $120\times160$ to $256\times320$. Each sequence provides 20 ground-truth *Foreground Mask* (FM) of frames. We compare ORLTM with online methods including ORPCA [Feng *et al.*, 2013], OLRSC [Shen *et al.*, 2016], OSTD [Sobral *et al.*, 2015], and batch methods including RPCA [Candès *et al.*, 2011], LRR [Liu *et al.*, 2013], HORPCA [Goldfarb and Qin, 2014], for which parameters are set as indicated in their papers. For our method, parameters are fixed as $\lambda_1 = \alpha/\sqrt{\log(n_1 n_2)}$ ($\alpha$ is 0.02 for *Curtain*, *Lobby*, *Watersurface*; and 0.1 for the rest), $\lambda_2 = 1$, $\lambda_3 = \lambda_1\sqrt{\log(t)}$ ($t$ is the iteration number), and each sub-tensor size is set to 1. For all methods, the target rank $p$ is empirically defined as 10, and the foreground masks are obtained by thresholding the sparse component, *i.e.*, $\text{FM}_{ij} = 1$ if $\mathbf{E}_{ij}^2/2 \geq (std(vec(\mathbf{E})))^2$, and zero otherwise, where $std(\cdot)$ denotes the standard deviation. For online methods, all frames are passed *two* epochs. For batch methods, they are applied to each *small*-batch (500 frames) per time considering limited memory resource. To initialize $\mathbf{B}^{(m)}$ in ORLTM, we adopt the Bilateral Random Projections (BRP) [Zhou and Tao, 2012]. Concretely, given dense matrix $\mathbf{\Gamma} \in \mathbb{R}^{d\times c}$, its low-rank approximation is built by $\tilde{\mathbf{\Gamma}} = \mathbf{\Gamma}\mathbf{Y}_1(\mathbf{Y}_2^\top\mathbf{\Gamma}\mathbf{Y}_1)^{-1}\mathbf{Y}_2^\top\mathbf{\Gamma}$, where $\mathbf{Y}_1 \in \mathbb{R}^{c\times r}$, $\mathbf{Y}_2 \in \mathbb{R}^{d\times r}$ are random matrices, and the rank $r \leq \min(d, c)$. In tests, we found LRR performs very poorly using the data set itself as the dictionary, so we use its mean matrix.

For evaluation, we employ *F-score* [Brutzer *et al.*, 2011] computed by comparing each sequence with its available ground-truth frames. The results are recorded in Table 2. From the table, we observe that ORLTM has the most encouraging overall performance across all scenes, about $10\%$ higher than second best RPCA (batch method). Especially, our method performs much better on *Bootstrap*, *Curtain*, *Fountain*, *Lobby*, and *Hall*, compared to the rest, and it also does well on the other sequences as ORLTM can achieve almost the same *F-score* to the best ones. The superior performance of ORLTM can be attributed to two facts: one is dictionary learning in ORLTM can better model video background despite in noisy and interrupted environments; the other is that learning the low-

Table 2: *F-score* (%) comparisons for background subtraction. Best in boldface and second best underlined.

| Methods | Bootstrap | Campus | Curtain | Fountain | Lobby | Shopmall | Watersurface | Hall | Average |
|---|---|---|---|---|---|---|---|---|---|
| RPCA | 63.86 | 59.88 | 66.33 | 79.28 | 72.07 | **75.87** | 44.39 | 53.61 | 64.41 |
| LRR | 55.02 | 32.75 | 84.92 | 62.87 | 35.47 | 70.30 | 61.14 | 55.60 | 57.26 |
| HORPCA | 60.27 | 32.58 | 72.79 | 31.31 | 53.12 | 74.95 | 43.39 | 58.02 | 53.30 |
| ORPCA | 55.48 | 37.26 | 82.29 | 56.76 | 62.20 | 49.53 | **89.86** | 56.53 | 61.24 |
| OLRSC | 44.75 | 23.01 | 77.76 | 26.96 | 15.54 | 21.49 | 79.16 | 32.53 | 40.15 |
| OSTD | 58.06 | **60.00** | 56.11 | 71.18 | 40.08 | 74.47 | 48.36 | 61.95 | 58.78 |
| **ORLTM** | **64.33** | 59.20 | **88.85** | **82.76** | **77.78** | 74.40 | 89.59 | **65.02** | **75.24** |



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 2: Sample frame masks for video background subtraction in a variety of scenes. (a) Input frame; (b) ORLTM (Background); (c) ORLTM (Foreground); (d) ORLTM; (e) OSTD; (f) OLRSC; (g) ORPCA; (h) HORPCA; (i) LRR; (j) RPCA.

Table 3: Speed comparison (*fps*). The value in the bracket denotes the acceleration rate of ORLTM compared with others.

| Video | $n_1 \times n_2 \times n_3$ | RPCA | HORPCA | **ORLTM** |
|---|---|---|---|---|
| Bootstrap | 120×160×3055 | 0.546 (12.16) | 0.352 (18.87) | **6.642** |
| Campus | 128×160×1439 | 2.220 (4.82) | 0.358 (29.87) | **10.694** |
| Curtain | 128×160×2964 | 2.740 (1.96) | 0.404 (13.31) | **5.379** |
| Fountain | 128×160× 523 | 2.556 (6.31) | 0.346 (46.65) | **16.140** |
| Lobby | 128×160×1551 | 2.621 (3.45) | 0.254 (35.60) | **9.042** |
| Shopmall | 256×320×1286 | 0.321 (8.50) | 0.101 (27.01) | **2.728** |
| Watersurface | 128×160× 641 | 3.456 (2.67) | 0.412 (22.37) | **9.217** |
| Hall | 144×176×3584 | 1.567 (2.76) | 0.289 (14.94) | **4.318** |

rank structures of tensor in all modes can sufficiently exploit the underlying information of video sequences. Note that batch methods process 500 frames per time, which might slightly degenerate its performance.

To examine the efficiency, we also report the *Frames Per Second (fps)* of our method and two batch methods in Table 3. As shown in the table, ORLTM is very efficient, *e.g.*, it is over 12 times faster than RPCA on *Bootstrap* and up to 46 times faster than HORPCA on *Fountain*. This is because ORLTM as an *online* method only needs constantly small memory and its computational cost is low in each iteration. On the contrary, batch methods have to store all frames in the memory in advance while they require expensive singular value decompositions on batch frames. All tests were run with 3.06 GHz Core X5675 processor and 24GB RAM.

In addition, we display the foreground masks of some randomly selected frames in Fig. 2, showing ORLTM outperforms others in most scenes, *e.g.*, curtain swinging (*row* 2), running fountain (*row* 3), illumination change (*row* 4). This demonstrates our method can well model background and foreground in a rich variety of scenes.

## 6 Conclusion

We present an Online Robust Low-rank Tensor Modeling (ORLTM) method to learn low-rank structures of streaming noisy tensor data robustly. By developing the equivalent bi-factor formulation of the nuclear norm, it can process samples sequentially, thus being scalable to large-scale tensor data. The objective function is solved by stochastic optimization, and in each iteration it saves memory cost by a factor of $n$ compared to batch methods. Synthetic studies suggest ORLTM outperforms well-established online methods and its performance is comparable to batch ones when the data are corrupted by up to 80% gross noise with large magnitude. In video background subtraction, ORLTM gains the highest average *F-score* over 10% higher than the state of the arts, and is significantly faster than batch methods, up to 46 times faster than HORPCA.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 61502131, 61572169, 61472266, Zhejiang Provincial Natural Science Foundation of China under Grant LQ15F020012, and China Scholarship Council. The work of Jiashi Feng was supported in part by National University of Singapore Startup Grant R-263-000-C08-133 and Ministry of Education of Singapore AcRF Tier One Grant R-263-000-C21-112.

## References


[Brutzer *et al.*, 2011] Sebastian Brutzer, Benjamin Höferlin, and Gunther Heidemann. Evaluation of background subtraction techniques for video surveillance. In *CVPR*, pages 1937–1944, 2011.

[Candès *et al.*, 2011] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):No.11, 2011.

[Fazel *et al.*, 2001] Maryam Fazel, Haitham Hindi, and Stephen P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the American Control Conference*, volume 6, pages 4734–4739, 2001.

[Feng *et al.*, 2013] Jiashi Feng, Huan Xu, and Shuicheng Yan. Online robust pca via stochastic optimization. In *NIPS*, pages 404–412, 2013.

[Goldfarb and Qin, 2014] Donald Goldfarb and Zhiwei Qin. Robust low-rank tensor recovery: models and algorithms. *SIAM Journal on Matrix Analysis and Applications*, 35(1):225–253, 2014.

[Hale *et al.*, 2008] Elaine T Hale, Wotao Yin, and Yin Zhang. Fixed-point continuation for $\ell_1$-minimization: Methodology and convergence. *SIAM Journal on Optimization*, 19(3):1107–1130, 2008.

[Hillar and Lim, 2013] Christopher J. Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM*, 60(6):No.45, 2013.

[Kilmer and Martin, 2011] Misha E Kilmer and Carla D Martin. Factorization strategies for third-order tensors. *Linear Algebra and its Applications*, 435(3):641–658, 2011.

[Kolda and Bader, 2009] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[Li *et al.*, 2004] Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, 2004.

[Liu and Yan, 2011] Guangcan Liu and Shuicheng Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *ICCV*, pages 1615–1622, 2011.

[Liu *et al.*, 2013] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013.

[Liu *et al.*, 2017] Guangcan Liu, Qingshan Liu, and Ping Li. Blessing of dimensionality: recovering mixture data via dictionary pursuit. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):47–60, 2017.

[Lu *et al.*, 2016] Canyi Lu, Jiashi Feng, Yudong Chen, Wei Liu, Zhouchen Lin, and Shuicheng Yan. Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization. In *CVPR*, pages 5249–5257, 2016.

[Mensch *et al.*, 2016] Arthur Mensch, Julien Mairal, Bertrand Thirion, and Gaël Varoquaux. Dictionary learning for massive matrix factorization. In *ICML*, volume 48, pages 1737–1746, 2016.

[Recht *et al.*, 2010] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52:471–501, 2010.

[Richtárik and Takáč, 2014] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

[Shen *et al.*, 2016] Jie Shen, Ping Li, and Huan Xu. Online low-rank subspace clustering by basis dictionary pursuit. In *ICML*, volume 48, pages 622–631, 2016.

[Sobral *et al.*, 2015] Andrews Sobral, Sajid Javed, Soon Ki Jung, Thierry Bouwmans, and El-hadi Zahzah. Online stochastic tensor decomposition for background subtraction in multispectral video sequences. In *ICCV Workshop*, pages 106–113, 2015.

[Srebro *et al.*, 2004] Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, pages 1329–1336, 2004.

[Sun *et al.*, 2008] Jimeng Sun, Dacheng Tao, Spiros Papadimitriou, Philip S Yu, and Christos Faloutsos. Incremental tensor analysis: Theory and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(3):11, 2008.

[Wright, 2015] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[Yin *et al.*, 2015] Ming Yin, Junbin Gao, Zhouchen Lin, Qinfeng Shi, and Yi Guo. Dual graph regularized latent low-rank representation for subspace clustering. *IEEE Transactions on Image Processing*, 24(12):4918–4933, 2015.

[Yu *et al.*, 2015] Rose Yu, Dehua Cheng, and Yan Liu. Accelerated online low-rank tensor learning for multivariate spatio-temporal streams. In *ICML*, pages 238–247, 2015.

[Zhan *et al.*, 2016] Jinchun Zhan, Brian Lois, Han Guo, and Namrata Vaswani. Online (and offline) robust pca: novel algorithms and performance guarantees. In *AISTATS*, pages 1488–1496, 2016.

[Zhang *et al.*, 2013] Xiaoqin Zhang, Di Wang, Zhengyuan Zhou, and Yi Ma. Simultaneous rectification and alignment via robust recovery of low-rank tensors. In *NIPS*, pages 1637–1645, 2013.

[Zhou and Tao, 2012] Tianyi Zhou and Dacheng Tao. Bilateral random projections. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 1286–1290, 2012.

[Zhou *et al.*, 2016] Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson. Accelerating online cp decompositions for higher order tensors. In *SIGKDD*, pages 1375–1384, 2016.