AN ONLINE TENSOR ROBUST PCA ALGORITHM FOR SEQUENTIAL 2D DATA

Zemin Zhang[⋆]*, Dehong Liu*[†]*, Shuchin Aeron*[⋆]*, Anthony Vetro*[†]

[⋆]Tufts University, Department of ECE, Medford, MA 02155
[†]Mitsubishi Electric Research Laboratories, Cambridge, MA 02139

ABSTRACT

Tensor robust principal component analysis (PCA) approaches have drawn considerable interests in many applications such as background subtraction, denoising, and outlier detection, *etc*. In this paper we propose an online tensor robust PCA where the multi-dimensional data (tensor) is revealed sequentially in online mode, and tensor PCA is updated based on the latest estimation and the newly collected data. Compared to the tensor robust PCA in batch mode, we significantly reduce the required memory and improve the computation efficiency. Application on fusing cloud-contaminated satellite images demonstrates that the proposed method shows superiority in both convergence speed and performance compared to the state-of-the-art approaches.

***Index Terms*—** Tensor Robust PCA, online learning, t-SVD, sequential data, cloud removal

1. INTRODUCTION

Principal Component Analysis (PCA) is commonly applied for dimensionality reduction in modern data analysis. However, when the data contains unintended artifacts such as gross corruptions or outliers, the classical PCA may fail. To solve this problem, a variety of robust PCA (RPCA) models have been developed. One of the effective models is known as principal component pursuit [1], which guarantees to recover a low-rank matrix and a sparse component from data under investigation with high probability given constraints on the rank and the sparsity level.

Recently, the online or recursive RPCA problem has been introduced, which aims to separate data samples in *online* mode, *i.e.*, with only the latest estimation and the newly collected data [2–4]. Unlike the standard RPCA methods which need to save all the data samples, these online RPCA algorithms significantly reduce the storage requirement and improve the computation efficiency.

For multidimensional data (tensor) of order greater than 2, it is general to embed the high dimensional data into a vector space by vectorizing the data points such that the conventional matrix-based approaches can still be used. Although this vectorization process works well in most cases, it restricts the effectiveness in extracting information from the multidimensional perspective. Alternatively, tensor algebraic approaches exhibit significant advantages in preserving multidimensional information when dealing with data of high order [**?**, 5–7]. However, it is very time-consuming for the tensor RPCA approaches to compute in batch mode since all of the high dimensional data needs to be stored and processed. To the best of our knowledge there is no existing online version tensor RPCA, and our work is to fill this gap.

To overcome the storage and computation efficiency problem, inspired by the matrix online robust PCA algorithm [2], we developed an online tensor RPCA algorithm. Our online tensor RPCA is based on the tensor Singular Value Decomposition (t-SVD) structure proposed in [8–10]. The key idea behind this tensor algebraic method is based on constructing group-rings along tensor fibers. Then 2-D data can be seen as vectors with elements from this group-ring. Subsequently, 3-D data can be seen as matrices with elements from this group-rings. Our work extends the batch tensor RPCA problem [5] and provide the benefit of sequential data collection, which reduces the required memory and increases efficiency.

The paper is organized as follows. In Section 2 we introduce the t-SVD structure which our algorithm lies in. In Section 3 we present our proposed online tensor RPCA algorithm. Experiment results using the proposed method as well as two state-of-the-art algorithms on multi-view cloud-contaminated images are reported in Section 4, followed by conclusion drawn in Section 5.

2. OVERVIEW OF TENSOR ALGEBRAIC FRAMEWORK

In this section, we briefly describe the tensor structure used in this paper, taking third-order tensor as an example. The development is adapted from [5, 9, 11, 12].

2.1. Notations and Definitions

For a third-order tensor $\boldsymbol{\mathcal{A}}$ of size $n_1 \times n_2 \times n_3$, $\boldsymbol{\mathcal{A}}(i,j,k)$ denotes the (i,j,k)-th element of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{A}}(i,j,:)$ denotes the (i,j)-th tubal-scalar. $\boldsymbol{\mathcal{A}}(i,:,:)$ is the i-th horizontal slice, $\boldsymbol{\mathcal{A}}(:,j,:)$ is the j-th lateral slice, and $\boldsymbol{\mathcal{A}}(:,:,k)$ or $\boldsymbol{\mathcal{A}}^{(k)}$ denotes the k-th frontal slice of $\boldsymbol{\mathcal{A}}$ respectively.

[Commutative Ring]. Let $\vec{v} \in \mathbb{R}^{1\times 1\times n_3}$ be a n_3-tuple, oriented into the board. As illustrated in Fig. 1, we define the multiplication operation (t-product) between two such n_3-tuples $\vec{u}$ and $\vec{v}$ via circular convolution which results in another n_3-tuple $\vec{w}$ represented by

$$\vec{w}(i) = \vec{u} * \vec{v} = \sum\nolimits_{k=0}^{n_3-1} \vec{u}(k)\vec{v}((i-k) \bmod(n_3)), \qquad (1)$$

where $i = 0, 1, ..., n_3 - 1$. Under the defined multiplication (t-product) and the usual addition, the set of n_3-tuples forms a commutative ring $\mathbb{R}(\mathbb{G}_{n_3})$ with the identity given by $\vec{e} = [1, 0, 0, ..., 0]$. In the following we will refer to these n_3-tuples oriented into the board as tubes or tubal-scalars.

[Remark]. Given two third-order tensors $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1\times n_2\times n_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_2\times n_4\times n_3}$, the result of the t-product of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ is a third-order tensor $\boldsymbol{\mathcal{C}}$ of size $n_1 \times n_4 \times n_3$ defined as

$$\boldsymbol{\mathcal{C}}(i,l,:) = \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}} = \sum\nolimits_{j=1}^{n_2} \boldsymbol{\mathcal{A}}(i,j,:) * \boldsymbol{\mathcal{B}}(j,l,:), \qquad (2)$$

where $i = 1, 2, ..., n_1$ and $l = 1, 2, ..., n_4$. This is consistent with the multiplication between matrices with the t-product '$*$' corresponding to the multiplication operation.

[Free-Module over the commutative ring]. Let $\mathbb{M}_{n_3}^{n_1}$ be a set

The work was done while Zemin Zhang was an intern with Mitsubishi Electric Research Laboratories.

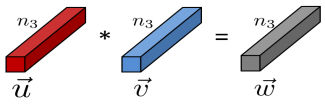

Fig. 1: Multiplication(t-product) over the commutative ring

of all 2-D lateral slices of size $n_1 \times 1 \times n_3$. Every element in $\mathbb{M}_{n_3}^{n_1}$ can be viewed as a vector of tubal-scalars. Since for any element $\vec{\mathcal{X}} \in \mathbb{M}_{n_3}^{n_1}$ and coefficient $v \in \mathbb{R}(\mathbb{G}_{n_3})$, $\vec{\mathcal{Y}} = \vec{\mathcal{X}} * v$ is also an element of $\mathbb{M}_{n_3}^{n_1}$, $\mathbb{M}_{n_3}^{n_1}$ is closed under the tubal-scalar multiplication.

Moreover, $\mathbb{M}_{n_3}^{n_1}$ is a free module of dimension n_3 over the commutative ring $\mathbb{R}(\mathbb{G}_{n_3})$ [8, 11]. Free module property means that one can construct a spanning basis $\{\vec{\mathcal{B}}_1, \vec{\mathcal{B}}_2, ..., \vec{\mathcal{B}}_{n_3}\}$ for this module using the relation between Fourier transform and circular convolution. Given the spanning basis, any element $\vec{\mathcal{X}} \in \mathbb{M}_{n_3}^{n_1}$ can be uniquely represented as a t-linear combination with some tubal-scalar coefficients $\vec{c}$

$$\vec{\mathcal{X}} = \sum_{i=1}^{n_3} \vec{\mathcal{B}}_i * \vec{c}_i, \tag{3}$$

as illustrate in Fig. 2.

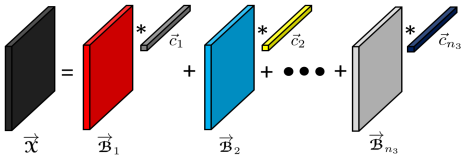

Fig. 2: An element of a free module generated by t-linear combination of spanning basis and coefficients.

2.2. Tensor-PCA and Tensor Singular Value Decomposition

Similar to the matrix PCA where we try to identify the lower-dimensional subspace which the data approximately lies in, we consider tensor PCA for high-order tensor data. In this paper, we focus on third-order tensors. Suppose the 2-D data samples come from a lower dimensional free submodule of the free module $\mathbb{M}_{n_3}^{n_1}$, where a free submodule is a subset of $\mathbb{M}_{n_3}^{n_1}$ with a spanning basis of dimension $d < n$. Our goal is to identify this free submodule which the 2-D data samples lie in.

[t-SVD]. Given n_2 2-D data samples $\vec{\mathcal{X}}_1, ..., \vec{\mathcal{X}}_{n_2}$ of size $n_1 \times n_3$, we arrange them as lateral slices to make a 3-D tensor $\mathcal{X}$ of size $n_1 \times n_2 \times n_3$. The t-SVD method is then used to compute the spanning basis (principal components) of this free submodule [12]. Specifically, t-SVD is defined as [1]

$$\mathcal{X} = \mathcal{U} * \mathcal{S} * \mathcal{V}^{\mathrm{T}}, \tag{4}$$

as shown in Fig. 3, where $\mathcal{U} \in \mathbb{R}^{n_1 \times d \times n_3}$ and $\mathcal{V} \in \mathbb{R}^{n_2 \times d \times n_3}$ are called orthogonal tensors which satisfy $\mathcal{U}^{\mathrm{T}} * \mathcal{U} = \mathcal{I}$ and $\mathcal{V}^{\mathrm{T}} * \mathcal{V} = \mathcal{I}$. $\mathcal{I}$ is the identity tensor whose frontal slices are all zeros except the first one an identity matrix. $\mathcal{S} \in \mathbb{R}^{d \times d \times n_3}$ is a tensor whose frontal slices are diagonal matrices. The tubal scalars $\mathcal{S}(i, i, :), i = 1, 2, ..., d$ into the board are called singular tubes and d is the tensor tubal-rank [5].

Based on the relation between the circular convolution and the Discrete Fourier Transform (DFT), we can compute t-SVD via SVD in the Fourier domain. Let $\widehat{\mathcal{X}}$ be the DFT along the third dimension of tensor $\mathcal{X}$ represented by $\widehat{\mathcal{X}} = \mathrm{fft}(\mathcal{X}, [\,], 3)$. Given SVD in the

[1] For any $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{A}^{\mathrm{T}} \in \mathbb{R}^{n_2 \times n_1 \times n_3}$ is obtained by transposing each of the frontal slices and then reversing the order of transposed frontal slices 2 through n_3.

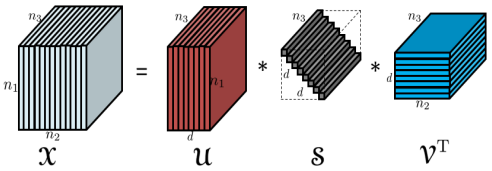

Fig. 3: t-SVD of an $n_1 \times n_2 \times n_3$ tensor.

Fourier domain $[\widehat{\mathcal{U}}(:,:,k), \widehat{\mathcal{S}}(:,:,k), \widehat{\mathcal{V}}(:,:,k)] = \mathtt{SVD}(\widehat{\mathcal{X}}(:,:,k))$, for $k = 1, ..., n_3$, we can compute t-SVD in (4) by

$$\mathcal{U} = \mathrm{ifft}(\widehat{\mathcal{U}}, [\,], 3), \mathcal{S} = \mathrm{ifft}(\widehat{\mathcal{S}}, [\,], 3), \mathcal{V} = \mathrm{ifft}(\widehat{\mathcal{V}}, [\,], 3). \tag{5}$$

Note that many properties of matrix SVD are retained in t-SVD, among which an important one is the optimality of truncated t-SVD for provably optimal dimension reduction [9].

3. ONLINE TENSOR ROBUST PCA

Now we consider the problem of recovering a tensor of low dimensional submodule from sparsely corrupted observations. Before going to the online version, we start with the batch setting. Suppose we have a third-order tensor $\mathcal{Z}$ which can be decomposed as,

$$\mathcal{Z} = \mathcal{X} + \mathcal{E}, \tag{6}$$

where $\mathcal{X}$ is a tensor with low tensor tubal rank and $\mathcal{E}$ is a sparse tensor. The problem of recovering $\mathcal{X}$ and $\mathcal{E}$ separately, termed tensor RPCA [5], can be formulated as an optimization problem [2]

$$\min_{\mathcal{X}, \mathcal{E}} \frac{1}{2}\|\mathcal{Z} - \mathcal{X} - \mathcal{E}\|_F^2 + \lambda_1 \|\mathcal{X}\|_{\mathrm{TNN}} + \lambda_2 \|\mathcal{E}\|_1, \tag{7}$$

where $\|\mathcal{X}\|_{\mathrm{TNN}} = \sum_{i,j} \widehat{\mathcal{S}}(i, i, j)$ denotes the tensor nuclear norm and it's a convex relaxation of the tensor tubal rank [5]; $\|\mathcal{E}\|_1 = \sum_{i,j,k} |\mathcal{E}(i, j, k)|$; and $\lambda_1, \lambda_2 > 0$.

Now we describe an implementation of tensor robust PCA that operates online. Suppose the 2-D data samples $\mathcal{Z}(:, i, :), i = 1, 2, ..., T$ are observed sequentially. Our goal is to estimate the spanning basis (principal components) of $\mathcal{X}$ on the fly, and separate the sparse tensor simultaneously. In order to proceed we introduce the following lemma without proof to save space.

Lemma 3.1. *For a third-order tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, suppose its tensor tubal rank is upper bounded by r, then we have*

$$\|\mathcal{X}\|_{TNN} = \inf_{\substack{\mathcal{L} \in \mathbb{R}^{n_1 \times r \times n_3} \\ \mathcal{R} \in \mathbb{R}^{n_2 \times r \times n_3}}} \left\{ \frac{n_3}{2}(\|\mathcal{L}\|_F^2 + \|\mathcal{R}\|_F^2) : \mathcal{X} = \mathcal{L} * \mathcal{R}^{\mathrm{T}} \right\}. \tag{8}$$

Using the above lemma, we re-write (7) as

$$\begin{aligned} \min_{\mathcal{L}, \mathcal{R}, \mathcal{E}} \; & \frac{1}{2}\|\mathcal{Z} - \mathcal{L} * \mathcal{R}^{\mathrm{T}} - \mathcal{E}\|_F^2 + \frac{n_3 \lambda_1}{2}(\|\mathcal{L}\|_F^2 + \|\mathcal{R}\|_F^2) \\ & + \lambda_2 \|\mathcal{E}\|_1 \;\; \text{s.t.} \;\; \mathcal{X} = \mathcal{L} * \mathcal{R}^{\mathrm{T}}, \end{aligned} \tag{9}$$

where $\mathcal{L} \in \mathbb{R}^{n_1 \times r \times n_3}$, $\mathcal{R} \in \mathbb{R}^{n_2 \times r \times n_3}$. For sequentially observed data $\{\vec{\mathcal{Z}}_1, \vec{\mathcal{Z}}_2, ..., \vec{\mathcal{Z}}_T\} \in \mathbb{R}^{n_1 \times 1 \times n_3}$, we define the loss function

[2] In [5] the authors use $\|.\|_{1,1,2}$ norm as the complexity of the "tubal-sparsity" of tensors in the third-dimension. Here we consider a general case of sparsity of $\mathcal{S}$, therefore $\|.\|_1$ norm is utilized.

for each sample based on (9) as

$$\ell(\overrightarrow{\mathcal{Z}}_i, \mathcal{L}) = \min_{\overrightarrow{\mathcal{R}}, \overrightarrow{\mathcal{E}}} \frac{1}{2}\|\overrightarrow{\mathcal{Z}}_i - \mathcal{L} * \overrightarrow{\mathcal{R}}^{\mathrm{T}} - \overrightarrow{\mathcal{E}}\|_F^2 + \frac{n_3\lambda}{2}\|\overrightarrow{\mathcal{R}}\|_F^2 + \lambda_2\|\overrightarrow{\mathcal{E}}\|_1. \tag{10}$$

We summarize our online tensor RPCA approach in Algorithm 1. For simplicity, we denote the output of $\mathrm{fft}(\mathcal{A}, [\,], 3)$ with wide-hat symbol $\widehat{\mathcal{A}} = [\widehat{\mathcal{A}}^{(1)}, \widehat{\mathcal{A}}^{(2)}, \cdots, \widehat{\mathcal{A}}^{(n_3)}]$, and let bar symbol $\overline{\mathcal{A}} \in \mathbb{R}^{n_1 n_3 \times n_2 n_3}$ be the block diagonal matrix defined by

$$\overline{\mathcal{A}} = \mathrm{blkdiag}(\widehat{\mathcal{A}}^{(1)}, \widehat{\mathcal{A}}^{(2)}, \cdots, \widehat{\mathcal{A}}^{(n_3)}). \tag{11}$$

The key idea of our online tensor RPCA algorithm is that at each time round t, we minimizes the loss function over $\overrightarrow{\mathcal{Z}}_t$ given the previous estimation $\mathcal{L}_{t-1}$, to get the optimal $\overrightarrow{\mathcal{R}}_t$ and $\overrightarrow{\mathcal{E}}_t$. Then we alternately use the latest estimated components to update the spanning basis $\mathcal{L}_t$ via minimizing the cumulative loss.

Specifically, $\overrightarrow{\mathcal{R}}_t$ and $\overrightarrow{\mathcal{E}}_t$ are optimized in Step 3 with detailed description in Algorithm 2. In the data projection step in Algorithm 2, $S_\lambda[\cdot]$ is a soft-thresholding operator defined by [13],

$$S_\lambda[x] = \begin{cases} x - \lambda, & \text{if } x > \lambda \\ x + \lambda, & \text{if } x < -\lambda \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

To update the spanning basis $\mathcal{L}_t$, we have

$$\begin{aligned}
\mathcal{L}_t &= \operatorname*{argmin}_{\mathcal{L}} \sum_{i=1}^{t} \Big(\frac{1}{2}\|\overrightarrow{\mathcal{Z}}_i - \mathcal{L} * \overrightarrow{\mathcal{R}}_i^{\mathrm{T}} - \overrightarrow{\mathcal{E}}_i\|_F^2 \\
&\qquad + \frac{n_3\lambda_1}{2}\|\overrightarrow{\mathcal{R}}_i\|_F^2 + \lambda_2\|\overrightarrow{\mathcal{E}}_i\|_1 \Big) + \frac{n_3\lambda_1}{2}\|\mathcal{L}\|_F^2 \\
&= \operatorname*{argmin}_{\mathcal{L}} \frac{1}{2}\|\mathcal{Z}_t - \mathcal{E}_t - \mathcal{L} * \mathcal{R}_t^{\mathrm{T}}\|_F^2 + \frac{n_3\lambda_1}{2}\|\mathcal{L}\|_F^2 \\
&= \operatorname*{argmin}_{\mathcal{L}} \frac{1}{2}\|\overline{\mathcal{Z}}_t - \overline{\mathcal{E}}_t - \overline{\mathcal{L}}\overline{\mathcal{R}}_t^{\mathrm{T}}\|_F^2 + \frac{n_3\lambda_1}{2}\|\overline{\mathcal{L}}\|_F^2 \\
&= \operatorname*{argmin}_{\mathcal{L}} \frac{1}{2}\mathrm{tr}\big((\overline{\mathcal{Z}}_t - \overline{\mathcal{E}}_t - \overline{\mathcal{L}}\overline{\mathcal{R}}_t^{\mathrm{T}})^{\mathrm{T}}(\overline{\mathcal{Z}}_t - \overline{\mathcal{E}}_t - \overline{\mathcal{L}}\overline{\mathcal{R}}_t^{\mathrm{T}})\big) \\
&\qquad + \frac{n_3\lambda_1}{2}\mathrm{tr}(\overline{\mathcal{L}}^{\mathrm{T}}\overline{\mathcal{L}}) \\
&= \operatorname*{argmin}_{\mathcal{L}} \frac{1}{2}\mathrm{tr}\big(\overline{\mathcal{L}}(\overline{\mathcal{R}}_t^{\mathrm{T}}\overline{\mathcal{R}}_t + n_3\lambda_1\mathbf{I})\overline{\mathcal{L}}^{\mathrm{T}}\big) - \mathrm{tr}(\overline{\mathcal{L}}^{\mathrm{T}}(\overline{\mathcal{Z}}_t - \overline{\mathcal{E}}_t)\overline{\mathcal{R}}_t).
\end{aligned}$$

Let $\mathcal{A}_t = \mathcal{A}_{t-1} + \overrightarrow{\mathcal{R}}_t * \overrightarrow{\mathcal{R}}_t^{\mathrm{T}}$ and $\mathcal{B}_t = \mathcal{B}_{t-1} + (\overrightarrow{\mathcal{Z}}_t - \overrightarrow{\mathcal{E}}_t) * \overrightarrow{\mathcal{R}}_t^{\mathrm{T}}$, as indicated in Step 4 of Algorithm 1, where $\overrightarrow{\mathcal{R}}_t \in \mathbb{R}^{r\times 1\times n_3}, \overrightarrow{\mathcal{E}}_t \in \mathbb{R}^{n_1\times 1\times n_3}$. We update $\mathcal{A}_t, \mathcal{B}_t$ each time new data comes and save the updated ones such that we can update the spanning basis $\mathcal{L}$ in the Fourier domain with block-coordinate descent [2], as indicated in Step 5 of Algorithm 1 with details in Algorithm 3. Note that our algorithm needs a prior information about estimated upper bound of the rank of the overall data samples.

As regarding to the storage needed, for the batch tensor robust PCA all the data samples up to time T, *i.e.*, the total number of entries in $\{\mathcal{Z}_i\}_{i=1}^T$, are required. Therefore the storage requirement for the batch tensor robust PCA is $n_1 n_3 T$. While for online tensor robust PCA, we need to save $\mathcal{L}_{t-1} \in \mathbb{R}^{n_1\times r\times n_3}$, $\mathcal{R}_T \in \mathbb{R}^{T\times r\times n_3}$ ($\mathcal{A}_T$ can be computed through $\mathcal{R}_T$), $\mathcal{B}_T \in \mathbb{R}^{n_1\times r\times n_3}$, and the total storage requirement is $n_3 rT + n_1 n_3 r$, which is much smaller than that of the batch tensor robust PCA when $r << T$.

Algorithm 1 Online Tensor Robust PCA

Input :Sequentially observed data $\mathcal{Z}_T = \{\overrightarrow{\mathcal{Z}}_1, ..., \overrightarrow{\mathcal{Z}}_T\} \in \mathbb{R}^{n_1\times 1\times n_3}$. $\lambda_1, \lambda_2 > 0$, number of rounds T.

Initial: $\mathcal{L}_0 \in \mathbb{R}^{n_1\times r\times n_3}, \overrightarrow{\mathcal{R}}_0 \in \mathbb{R}^{r\times 1\times n_3}, \overrightarrow{\mathcal{E}}_0 \in \mathbb{R}^{n_1\times 1\times n_3}$.

1: **for** $t = 1, 2, ..., T$ **do**

2: Reveal data sample $\overrightarrow{\mathcal{Z}}_t$.

3: Project the new sample(See Algorithm 2):

$$\{\overrightarrow{\mathcal{R}}_t, \overrightarrow{\mathcal{E}}_t\} = \operatorname*{argmin}_{\substack{\overrightarrow{\mathcal{R}}\in\mathbb{R}^{r\times 1\times n_3} \\ \overrightarrow{\mathcal{E}}\in\mathbb{R}^{n_1\times 1\times n_3}}} \frac{1}{2}\|\overrightarrow{\mathcal{Z}}_t - \mathcal{L}_{t-1} * \overrightarrow{\mathcal{R}} - \overrightarrow{\mathcal{E}}\|_F^2 + \frac{\lambda_1}{2}\|\overrightarrow{\mathcal{R}}\|_F^2 + \lambda_2\|\overrightarrow{\mathcal{E}}\|_1$$

4: Compute $\mathcal{A}_t = \mathcal{A}_{t-1} + \overrightarrow{\mathcal{R}}_t * \overrightarrow{\mathcal{R}}_t^{\mathrm{T}}$, $\mathcal{B}_t = \mathcal{B}_{t-1} + (\overrightarrow{\mathcal{Z}}_t - \overrightarrow{\mathcal{E}}_t) * \overrightarrow{\mathcal{R}}_t^{\mathrm{T}}$, and then block diagonal matrices $\overline{\mathcal{A}}_t$ and $\overline{\mathcal{B}}_t$ using $\widehat{\mathcal{A}}_t = \mathrm{fft}(\mathcal{A}_t, [\,], 3)$, $\widehat{\mathcal{B}}_t = \mathrm{fft}(\mathcal{B}_t, [\,], 3)$ according to (11) .

5: Update spanning tensor basis $\mathcal{L}_t$ (See Algorithm 3):

$$\overline{\mathcal{L}}_t \triangleq \operatorname*{argmin}_{\overline{\mathcal{L}}} \frac{1}{2}\mathrm{tr}[\overline{\mathcal{L}}^{\mathrm{T}}(\overline{\mathcal{A}}_t + n_3\lambda_1\mathbf{I})\overline{\mathcal{L}}] - \mathrm{tr}(\overline{\mathcal{L}}^{\mathrm{T}}\overline{\mathcal{B}}_t).$$

6: Organize the block diagonal $\overline{\mathcal{L}}_t$ into the tensor form $\widehat{\mathcal{L}}_t$ and let $\mathcal{L}_t = \mathtt{ifft}(\widehat{\mathcal{L}}_t, [\,], 3)$.

7: **end for**

8: $\mathcal{R}_T(t, :, :) = \overrightarrow{\mathcal{R}}_t^{\mathrm{T}}, \mathcal{E}_T(:, t, :) = \overrightarrow{\mathcal{E}}_t, t = 1, 2, ..., T$.

9: **Return** $\mathcal{X}_T = \mathcal{L}_T * \mathcal{R}_T^{\mathrm{T}}$ and $\mathcal{E}_T$.

Output: Low tubal rank tensor $\mathcal{X}_T$ and sparse tensor $\mathcal{E}_T$.

4. EXPERIMENTAL RESULTS

We consider the cloud removal problem on satellite images. A total of 24 cloud contaminated images captured by Landsat 7 ETM+ and Landsat 8 OLI near Harz, Germany over a period of time are used in our numerical experiments [14]. Each image is of size 598×1070. The image backgrounds change slightly and the variability is mainly caused by the clouds and their shadows. Two example images are shown in Fig. 4. Considering the fact that the images are captured sequentially with restricted onboard storage, we use tensor online RPCA to perform cloud removal process such that clear images can be generated. Each time we receive an image from the satellite of this area, we directly estimate the spanning basis without saving all the past images but only the latest estimation and the newly collected data.

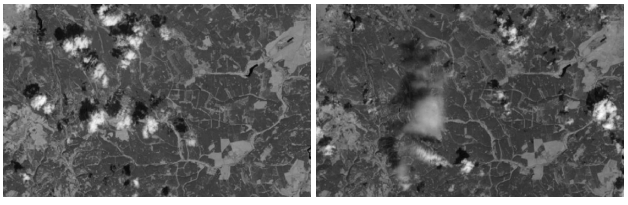

Fig. 4: Example images of the same location taken by Landsat 7 ETM+ and Landsat 8 OLI on different days. A total of 24 images are used in our experiments.

To perform the proposed algorithm, images are normalized to have intensity of $[0, 1]$ and reshaped as a lateral slice $\vec{z}_i$ of size $598 \times 1 \times 1070$. In the experiments we set $r = 3$ as a upper bound of the tensor tubal rank since the background is barely changing without clouds and shadows and $\lambda_1 = \lambda_2 = 1/\sqrt{598}$. The results are shown

Algorithm 2 Projecting data samples

Input: $\mathcal{L}_{t-1} \in \mathbb{R}^{n_1 \times r \times n_3}$, $\overrightarrow{\mathcal{Z}}_t \in \mathbb{R}^{n_1 \times 1 \times n_3}$, $\lambda_1, \lambda_2 > 0$.
Initial: $\overrightarrow{\mathcal{E}}_t = 0$, $\hat{\boldsymbol{r}}_t \in \mathbb{C}^{r \times 1 \times n_3}$.
$\hat{\boldsymbol{z}}_t = \texttt{fft}(\overrightarrow{\mathcal{Z}}_t, [\,], 3)$, $\widehat{\mathcal{L}}_{t-1} = \texttt{fft}(\mathcal{L}_{t-1}, [\,], 3)$.
while not converge **do**
  for $i = 1, 2, ..., n_3$ **do**
    $\hat{\boldsymbol{r}}_t^{(i)} = ((\widehat{\mathcal{L}}_{t-1}^{(i)})^{\mathrm{T}} \widehat{\mathcal{L}}_{t-1}^{(i)} + \lambda_1 \mathbf{I})^{-1} (\widehat{\mathcal{L}}_{t-1}^{(i)})^{\mathrm{T}} (\hat{\boldsymbol{z}}_t^{(i)} - \hat{\boldsymbol{e}}_t^{(i)})$.
  end for
  $\overrightarrow{\mathcal{R}}_t = \texttt{ifft}(\hat{\boldsymbol{r}}_t, [\,], 3)$.
  $\overrightarrow{\mathcal{E}}_t = S_{\lambda_2}[\overrightarrow{\mathcal{Z}}_t - \mathcal{L}_{t-1}^{\mathrm{T}} * \overrightarrow{\mathcal{R}}_t]$.
end while
Output: $\overrightarrow{\mathcal{R}}_t$ and $\overrightarrow{\mathcal{E}}_t$.

Algorithm 3 Update spanning tensor basis

Input: $\mathcal{L}_{t-1} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$,$\mathcal{A}_t \in \mathbb{R}^{r \times r \times n_3}$,$\mathcal{B}_t \in \mathbb{R}^{n_1 \times r \times n_3}$

$\mathcal{L}_t = \mathcal{L}_{t-1}$, $\widehat{\mathcal{L}}_t = \texttt{fft}(\mathcal{L}_t, [\,], 3)$.
$\widehat{\mathcal{B}}_t = \texttt{fft}(\mathcal{B}_t, [\,], 3)$.
$\mathcal{C}_t = \mathcal{A}_t + \lambda_1 \mathcal{I}$, $\widehat{\mathcal{C}}_t = \texttt{fft}(\mathcal{C}_t, [\,], 3)$.
for $j = 1, 2, ..., r$ **do**
  for $k = 1, 2, ..., n_3$ **do**
    $\widehat{\mathcal{L}}_t(:, j, k) = \frac{\widehat{\mathcal{B}}_t(:,j,k) - \widehat{\mathcal{L}}_t(:,:,k) * \widehat{\mathcal{C}}_t(:,j,k)}{\widehat{\mathcal{C}}_t(j,j,k)} + \widehat{\mathcal{L}}_t(:, j, k)$.
  end for
end for
Output: $\mathcal{L}_t = \texttt{ifft}(\widehat{\mathcal{L}}_t, [\,], 3)$.

in Fig. 5, in which the top row and the bottom row represents the low tubal rank parts and the sparse parts respectively. It is clear that the cloud and shadow contaminated areas are well recovered using our online tensor RPCA method.

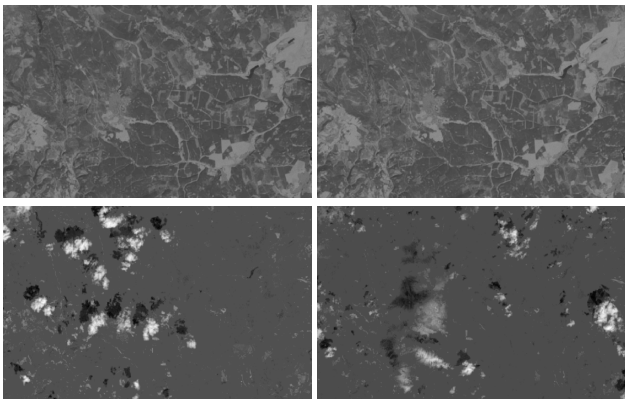

Fig. 5: Tensor online Robust PCA results on the images shown in Fig. 4. The top row images show the low tubal rank components and the bottom row images show the sparse components.

In order to further evaluate our algorithm, we compare the tensor online robust PCA to the matrix online robust PCA [2] and the Grasta streaming [3]. Here we synthesize a total number of 500 cloud-free images of the same location but different time as ground truth tensor $\mathcal{G}$. The images are of size 100×100 and intensity normalized to $[0, 1]$. We randomly add zero-mean Gaussian noise with standard deviation 10 to sparse locations of the images and use them as our input data samples $\mathcal{Z}$. The sparsity level varies from 0.02 to 0.2. We run over all the possible ranks using the tensor online robust PCA and the matrix online robust PCA, then choose the ranks which give the best performance on the two algorithms. In tensor online robust PCA, $\lambda_1 = \lambda_2 = 1/\sqrt{100} = 0.1$. For matrix online robust PCA and the Grasta streaming, each time the data image is vectorized into a long vector as input. At each round t we compute the relative square error up to time t defined as

$$\textbf{RSE}(t) = \|\mathcal{X}_t - \mathcal{G}_t\|_F / \|\mathcal{G}_t\|_F, \tag{13}$$

where $\mathcal{X}_t$ and $\mathcal{G}_t$ are the low rank component and ground truth we get up to time t, respectively. The performance is compared in Fig. 6. We can tell that our tensor online RPCA has a smaller relative square error comparing to the matrix online robust PCA and Grasta streaming on all sparsity levels.

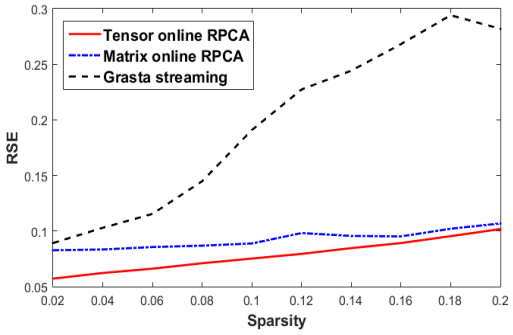

Fig. 6: Comparison of tensor online robust PCA, matrix online robust PCA, and Grasta streaming version.

Figure 7 shows the convergence speed of the three algorithms when the sparsity level of the Gaussian noise is equal to 0.2. Our tensor online robust PCA converges the fastest among the three. Consequently, when the total number of data samples are small, our proposed algorithm is more efficient and accurate than the other two methods.

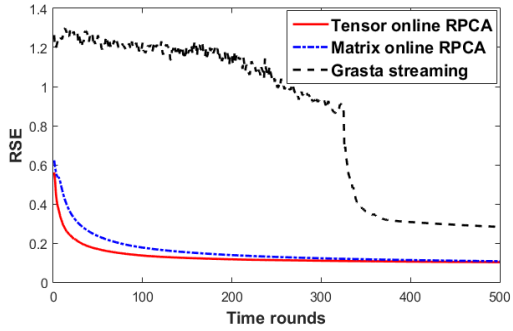

Fig. 7: Convergence speed comparison when the sparsity of Gaussian noise is 0.2.

5. CONCLUSION AND FUTURE WORK

In this paper, a novel tensor online robust PCA method has been introduced. Comparing to the batch mode PCA methods, it is more efficient and faster since it is not necessary to store all the data samples. In addition, our algorithm is different from the matrix based online learning algorithms in treating high-order data as a "lateral slice" from the multidimensional perspective. Experimental results on cloud removal of the satellite images demonstrate that the proposed tensor online RPCA algorithm outperforms significantly over the state-of-the-art algorithms in both convergence speed and performance. In future work it is of interest to perform theoretical analysis on the performance and convergence of this algorithm and employ other tensor algebra to it.

6. REFERENCES

[1] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright, "Robust principal component analysis?," *J. ACM*, vol. 58, no. 3, pp. 11:1–11:37, June 2011.

[2] Jiashi Feng, Huan Xu, and Shuicheng Yan, "Online robust PCA via stochastic optimization," in *Advances in Neural Information Processing Systems 26*, pp. 404–412. Curran Associates, Inc., 2013.

[3] Jun He, Laura Balzano, and Arthur Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1568–1575.

[4] Brian Lois and Namrata Vaswani, "A correctness result for online robust PCA," *CoRR*, vol. abs/1409.3959, 2014.

[5] Zemin Zhang, Gregory Ely, Shuchin Aeron, Ning Hao, and Misha Elena Kilmer, "Novel methods for multilinear data completion and de-noising based on tensor-SVD," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3842–3849.

[6] Donald Goldfarb and Zhiwei (Tony) Qin, "Robust low-rank tensor recovery: Models and algorithms," *SIAM J. Matrix Analysis Applications*, vol. 35, no. 1, pp. 225–253, 2014.

[7] Wenfei Cao, Yao Wang, Jian Sun, Deyu Meng, Can Yang, Andrzej Cichocki, and Zongben Xu, "A novel tensor robust PCA approach for background subtraction from compressive measurements," *CoRR*, vol. abs/1503.01868, 2015.

[8] Karen Braman, "Third-order tensors as linear operators on a space of matrices," *Linear Algebra Appl.*, vol. 433, no. 7, pp. 1241–1253, 2010.

[9] Misha E. Kilmer and Carla D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra and its Applications*, vol. 435, no. 3, pp. 641–658, 8 2011.

[10] Carmeliza Navasca, Michael Opperman, Timothy Penderghest, and Christino Tamon, "Tensors as module homomorphisms over group rings," *CoRR*, vol. abs/1005.1894, 2010.

[11] Eric Kernfeld, Shuchin Aeron, and Misha Elena Kilmer, "Clustering multi-way data: a novel algebraic approach," *CoRR*, vol. abs/1412.7056, 2014.

[12] Misha E. Kilmer, Karen Braman, Ning Hao, and Randy C. Hoover, "Third-order tensors as operators on matrices: a theoretical and computational framework with applications in imaging," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 148–172, 2013.

[13] David L. Donoho, "De-noising by soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, 1995.

[14] NASA Landsat Program, "Landsat ETM+ scene," available at `http://glcf.umd.edu/data/`.