

# A Randomized Algorithm for Tensor Singular Value Decomposition Using an Arbitrary Number of Passes

Salman Ahmadi-Asl<sup>a</sup>, Anh-Huy Phan<sup>a</sup>, Andrzej Cichocki<sup>b</sup>

<sup>a</sup>*Center for Artificial Intelligence Technology, Skolkovo Institute of Science and Technology, Moscow, Russia s.asl@skoltech.ru,*

<sup>b</sup>*Systems Research Institute of Polish Academy of Science, Warsaw, Poland,*

---

## Abstract

Efficient and fast computation of a tensor singular value decomposition (t-SVD) with a few passes over the underlying data tensor is crucial because of its many potential applications. The current/existing subspace randomized algorithms need  $(2q+2)$  passes over the data tensor to compute a t-SVD, where  $q$  is a non-negative integer number (power iteration parameter). In this paper, we propose an efficient and flexible randomized algorithm that can handle any number of passes  $q$ , which not necessary need be even. The flexibility of the proposed algorithm in using fewer passes naturally leads to lower computational and communication costs. This advantage makes it particularly appropriate when our task calls for several tensor decompositions or when the data tensors are huge. The proposed algorithm is a generalization of the methods developed for matrices to tensors. The expected/average error bound of the proposed algorithm is derived. Extensive numerical experiments on random and real-world data sets are conducted, and the proposed algorithm is compared with some baseline algorithms. The extensive computer simulation experiments demonstrate that the proposed algorithm is practical, efficient, and in general outperforms the state of the arts algorithms. We also demonstrate how to use the proposed method to develop a fast algorithm for the tensor completion problem.

**Keywords:** Tensor singular value decomposition, randomization, pass-efficient algorithms.

**2000 MSC:** 15A69, 46N40, 15A23

---

## 1. Introduction

Multidimensional arrays or tensors are natural extensions of matrices and vectors. It was in 1927 when the first definition of the tensor rank was presented and the corresponding tensor decomposition termed Canonical Polyadic Decomposition (CPD) was defined [1]. Later on, an alternative definition for the tensor rank (Tucker rank) was defined in 1963 [2, 3] and a new tensor decomposition called Tucker decomposition was proposed. The Tucker decomposition includes the CPD as a special case. Indeed, there is no unique definition for the tensor rank or the tensor decomposition. Other tensor decompositions include Tensor Train (TT) or Matrix Product State (MPS) [4, 5], Tensor Chain/Ring (TC) or MPS with periodic boundary conditions [6, 7], tensor SVD (t-SVD) [8, 9] and Hierarchical Tucker decomposition [10].

Tensors have been successfully applied in many machine learning and data analysis tasks such as data reconstruction, data compression, clustering, etc. See [11, 12] and the references therein for a comprehensive review of such applications. One of the main challenges in this work is developing fast algorithms for the computation of different types of tensor decomposition. For example, to solve the tensor completion problem [13], we usually need to compute tensor decompositions multiple times. When the data tensors are huge or many iterations are necessary for the convergence, these calculations become prohibitively expensive. Therefore, in order to be employed in real-time applications such as traffic data prediction, we need to build fast methods for various types of tensor decompositions.

The randomization framework has been proven to be an efficient technique, for low-rank matrix computation [14] and recently generalized to the tensors [15, 16, 17, 18]. It is known that randomized algorithms reduce the computational complexity of the deterministic counterparts and also their communication costs. The latter benefit is especially important when the data tensor is very large and stored on several machines. Here, the communication cost is the main concern and we need to access the data tensor as few times as possible. In the context of randomization, such methods are called randomized pass-efficient algorithms [19]. For example, in [20] a pass-efficient randomized algorithm is developed for the Tucker decomposition. The standard randomized subspace iteration method needs  $(2q + 2)$  passes/views over the data tensor [14], where  $q$  is a power iteration parameter. In [19], the author proposed new randomized algorithms for matrices, which do not have this limitation and, for any budget of passes, can compute a low-rank matrix approximation. In this paper, we generalize this idea to the

tensor case based on the tubal product (t-product) [8, 9]. The flexibility of using fewer passes leads to lower communication and computational costs and this is the key benefit of the proposed algorithm in this paper. For example, as will be shown in our experiments, for the image/video compression task, in Section 6, three passes provided quite good results, while the classical subspace randomized algorithm at least needs 4 passes, and this extra pass can be very expensive for data tensors stored out-of-core or when multiple low tubal rank approximations are required in our task, e.g., tensor completion. Simulations on synthetic and real-world datasets are provided to support the theoretical results. In particular, we present an application of the proposed algorithm for the image/video completion problem.

Our principal contributions can be summarized as follows:

- Developing a pass-efficient randomized algorithm for the computation of the t-SVD with an arbitrary number of passes
- Applying the proposed algorithm to reconstruct images/videos with missing pixels
- Extensive simulation results on synthetic benchmarks and real-world datasets

The remainder of this paper is organized as follows: The preliminary concepts and definitions are introduced in Section 2. Section 3 is devoted to introducing the t-SVD and its computational procedures. The proposed approach is outlined in Section 4. An application of the proposed algorithm to the tensor completion problem is presented in Section 5. Simulation results are presented in Section 6 and a conclusion is given in Section 7.

## 2. Preliminaries

Let us introduce the notations and main concepts that we need in the next sections. Tensors, matrices, and vectors are denoted by underlined bold upper case letters e.g.,  $\underline{\mathbf{X}}$ , bold upper case letters, e.g.,  $\mathbf{X}$ , and bold lower case letters, e.g.,  $\mathbf{x}$ , respectively. Slices are produced by fixing all but two modes of a tensor. For a third-order tensor  $\underline{\mathbf{X}}$ ,  $\underline{\mathbf{X}}(:, :, i)$ ,  $\underline{\mathbf{X}}(:, i, :)$  and  $\underline{\mathbf{X}}(i, :, :)$  are called frontal, lateral, and horizontal slices, respectively. Fibers are obtained by fixing all but one mode. For a third-order tensor  $\underline{\mathbf{X}}$ ,  $\underline{\mathbf{X}}(i, j, :)$  is called a tube. The Frobenius and infinity norms of tensors are denoted by  $\|\cdot\|_F$  and  $\|\cdot\|_\infty$ , respectively. The notation “conj”

denotes the complex conjugate of a complex number or the component-wise complex conjugate of a matrix. The mathematical expectation of a random tensor  $\underline{\mathbf{X}}$  is denoted by  $\mathbb{E}(\underline{\mathbf{X}})$ , and  $\lceil n \rceil$  means the nearest integer number greater than or equal to  $n$ . The element-wise or Hadamard product, is denoted by “ $\otimes$ ”. For a given data tensor  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  and the indicator set  $\underline{\Omega} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ , the projector  $\mathbf{P}_{\underline{\Omega}}$  is defined as follows

$$\mathbf{P}_{\underline{\Omega}}(\underline{\mathbf{X}}) = \begin{cases} \underline{\mathbf{X}}(\mathbf{i}) & \mathbf{i} \in \underline{\Omega}, \\ 0 & \mathbf{i} \notin \underline{\Omega}, \end{cases}$$

where  $\mathbf{i} = (i_1, i_2, \dots, i_N)$  is a multi-index and  $1 \leq i_n \leq I_n$ ,  $n = 1, 2, \dots, N$ . Throughout the paper, we focus only on real and third-order tensors, however, our results can be generalized to complex higher-order tensors straightforwardly.

Before presenting the t-SVD, we first need to present some basic definitions such as the t-product operation and f-diagonal tensors.

**Definition 1.** (t-product) Let  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$  and  $\underline{\mathbf{Y}} \in \mathbb{R}^{I_2 \times I_4 \times I_3}$ , then the t-product  $\underline{\mathbf{X}} * \underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_4 \times I_3}$  is defined as follows

$$\underline{\mathbf{C}} = \underline{\mathbf{X}} * \underline{\mathbf{Y}} = \text{fold}(\text{circ}(\underline{\mathbf{X}}) \text{unfold}(\underline{\mathbf{Y}})), \quad (1)$$

where

$$\text{circ}(\underline{\mathbf{X}}) = \begin{bmatrix} \underline{\mathbf{X}}(:, :, 1) & \underline{\mathbf{X}}(:, :, I_3) & \cdots & \underline{\mathbf{X}}(:, :, 2) \\ \underline{\mathbf{X}}(:, :, 2) & \underline{\mathbf{X}}(:, :, 1) & \cdots & \underline{\mathbf{X}}(:, :, 3) \\ \vdots & \vdots & \ddots & \vdots \\ \underline{\mathbf{X}}(:, :, I_3) & \underline{\mathbf{X}}(:, :, I_3 - 1) & \cdots & \underline{\mathbf{X}}(:, :, 1) \end{bmatrix},$$

and

$$\text{unfold}(\underline{\mathbf{Y}}) = \begin{bmatrix} \underline{\mathbf{Y}}(:, :, 1) \\ \underline{\mathbf{Y}}(:, :, 2) \\ \vdots \\ \underline{\mathbf{Y}}(:, :, I_3) \end{bmatrix}, \quad \underline{\mathbf{Y}} = \text{fold}(\text{unfold}(\underline{\mathbf{Y}})).$$

It is known that the t-product operation is indeed the circular convolution operator and can be computed using the Fast Fourier Transform (FFT). To this end, the FFT operator is applied to all tubes of two tensors  $\underline{\mathbf{X}}$ ,  $\underline{\mathbf{Y}}$ , and obtains new tensors  $\widehat{\underline{\mathbf{X}}}$ ,  $\widehat{\underline{\mathbf{Y}}}$ . Then, we multiply the frontal slices of the tensors  $\widehat{\underline{\mathbf{X}}}$ ,  $\widehat{\underline{\mathbf{Y}}}$  to get the new tensor  $\widehat{\underline{\mathbf{Z}}}$ . Finally, we apply the Inverse FFT (IFFT) operator to all tubes of the tensor  $\widehat{\underline{\mathbf{Z}}}$ . This procedure is summarized in Algorithm 1, which is the optimized version of the t-product, because it needs only the FFT of the  $\lceil \frac{I_3+1}{2} \rceil$  first frontal

slices suggested by the works [21, 22], while the original papers ([8, 9]) consider the FFT of all frontal slices. Note that  $\text{fft}(\underline{\mathbf{Z}}, \square, 3)$  is equivalent to computing the FFT of all tubes of the tensor  $\underline{\mathbf{Z}}$ . The t-product can be defined according to an arbitrary invertible transform [23]. For example, the unitary transform matrices were utilized in [24], instead of discrete Fourier transform matrices. It was also proposed in [25] to use non-invertible transforms instead of unitary matrices.

It can be proven that for a tensor  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ , we have

$$\|\underline{\mathbf{X}}\|_F^2 = \frac{1}{I_3} \sum_{i=1}^{I_3} \|\widehat{\underline{\mathbf{X}}}(:, :, i)\|_F^2, \quad (2)$$

where  $\widehat{\underline{\mathbf{X}}}(:, :, i)$  is the  $i$ -th frontal slice of the tensor  $\widehat{\underline{\mathbf{X}}} = \text{fft}(\underline{\mathbf{X}}, \square, 3)$ , see [22, 26].

---

**Algorithm 1:** t-product in the Fourier domain [8]

---

**Input :** Two data tensors  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ ,  $\underline{\mathbf{Y}} \in \mathbb{R}^{I_2 \times I_4 \times I_3}$   
**Output:** t-product  $\underline{\mathbf{C}} = \underline{\mathbf{X}} * \underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_4 \times I_3}$

- 1  $\widehat{\underline{\mathbf{X}}} = \text{fft}(\underline{\mathbf{X}}, \square, 3);$
- 2  $\widehat{\underline{\mathbf{Y}}} = \text{fft}(\underline{\mathbf{Y}}, \square, 3);$
- 3 **for**  $i = 1, 2, \dots, \lceil \frac{I_3+1}{2} \rceil$  **do**
- 4      $\widehat{\underline{\mathbf{C}}}(:, :, i) = \widehat{\underline{\mathbf{X}}}(:, :, i) \widehat{\underline{\mathbf{Y}}}(:, :, i);$
- 5 **end**
- 6 **for**  $i = \lceil \frac{I_3+1}{2} \rceil + 1, \dots, I_3$  **do**
- 7      $\widehat{\underline{\mathbf{C}}}(:, :, i) = \text{conj}(\widehat{\underline{\mathbf{C}}}(:, :, I_3 - i + 2));$
- 8 **end**
- 9  $\underline{\mathbf{C}} = \text{ifft}(\widehat{\underline{\mathbf{C}}}, \square, 3);$

---

**Definition 2.** (Transpose) Let  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$  be a given tensor. Then the transpose of the tensor  $\underline{\mathbf{X}}$  is denoted by  $\underline{\mathbf{X}}^T \in \mathbb{R}^{I_2 \times I_1 \times I_3}$ , which is constructed by transposing all its frontal slices and then reversing the order of transposed frontal slices 2 through  $I_3$ .

**Definition 3.** (Identity tensor) The tensor  $\underline{\mathbf{I}} \in \mathbb{R}^{I_1 \times I_1 \times I_3}$  is called identity if its first frontal slice is an identity matrix of size  $I_1 \times I_1$  and all other frontal slices are zero. It is easy to show  $\underline{\mathbf{I}} * \underline{\mathbf{X}} = \underline{\mathbf{X}}$  and  $\underline{\mathbf{X}} * \underline{\mathbf{I}} = \underline{\mathbf{X}}$  for all tensors of conforming sizes.

**Definition 4.** (Orthogonal tensor) A tensor  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_1 \times I_3}$  is orthogonal if  $\underline{\mathbf{X}}^T * \underline{\mathbf{X}} = \underline{\mathbf{X}} * \underline{\mathbf{X}}^T = \underline{\mathbf{I}}$ .

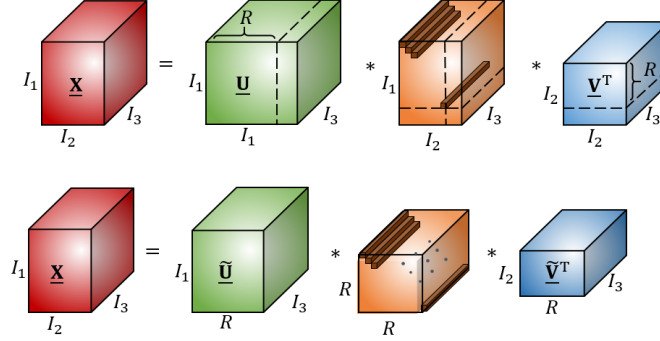


Figure 1: Illustration of (a) Tensor SVD (t-SVD) and (b) truncated t-SVD for a third-order tensor [34].

**Definition 5.** (f-diagonal tensor) If all frontal slices of a tensor are diagonal then the tensor is called an f-diagonal tensor.

**Definition 6.** (Random tensor) A tensor  $\underline{\Omega}$  is random if its first frontal slice  $\underline{\Omega}(:, :, 1)$  has independent and identically distributed (i.i.d) elements, while the other frontal slices are zero.

The MATLAB implementation of many operations in the t-product format can be found in the following useful toolbox:

<https://github.com/canyilu/Tensor-tensor-product-toolbox>.

### 3. Tensor SVD and Tensor QR decomposition

Tensor SVD (t-SVD) was originally proposed in ([8, 9, 27, 28]) to represent a third-order tensor as a product of three third-order tensors, where all frontal slices of the middle tensor are diagonal, (Figure 1, provides a graphical illustration on the t-SVD and its truncated version). For a generalization of the t-SVD to higher-order tensors, see [29]. This decomposition has found interesting applications such as completion, clustering, compression, and background initialization in video surveillance, see [30, 31, 32, 33].

The tubal rank is defined as the number of nonzero tubes of the middle tensor. It is interesting to note that, unlike other tensor decompositions, the truncated t-SVD provides the best approximation in the least-squares sense for any unitary invariant tensor norm [8].

Let  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ , then the t-SVD of the tensor  $\underline{\mathbf{X}}$ , admits the model  $\underline{\mathbf{X}} = \underline{\mathbf{U}} * \underline{\mathbf{S}} * \underline{\mathbf{V}}^T$ , where  $\underline{\mathbf{U}} \in \mathbb{R}^{I_1 \times R \times I_3}$ ,  $\underline{\mathbf{V}} \in \mathbb{R}^{I_2 \times R \times I_3}$  are orthogonal tensors and the tensor  $\underline{\mathbf{S}} \in \mathbb{R}^{R \times R \times I_3}$  is f-diagonal ([8, 9]). Computation of the t-SVD is performed in the Fourier domain and this is summarized in Algorithm 2. Algorithm 2 is a faster version of the original truncated t-SVD [9, 8] and was developed in [23, 22]. It requires only the SVD of the  $\lceil \frac{I_3+1}{2} \rceil$  first frontal slices [22]. The original t-SVD [8, 9] involves the SVD of all frontal slices in the Fourier domain. Similar to the t-product, instead of the discrete Fourier transform matrices, one can define the t-SVD according to an arbitrary unitary transform matrix. It is shown in [25] that this can provide a t-SVD with a lower tubal rank.

---

**Algorithm 2:** Truncated t-SVD [8, 9]

---

**Input :** A data tensor  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$  and a tubal rank  $R$ ;  
**Output:**  $\underline{\mathbf{U}}_R \in \mathbb{R}^{I_1 \times R \times I_3}$ ,  $\underline{\mathbf{S}}_R \in \mathbb{R}^{R \times R \times I_3}$ ,  $\underline{\mathbf{V}}_R \in \mathbb{R}^{I_2 \times R \times I_3}$ ;

```

1  $\hat{\underline{\mathbf{X}}} = \text{fft}(\underline{\mathbf{X}}, [], 3)$ ;
2 for  $i = 1, 2, \dots, \lceil \frac{I_3+1}{2} \rceil$  do
3    $[\underline{\mathbf{U}}, \underline{\mathbf{S}}, \underline{\mathbf{V}}] = \text{Truncated-SVD}(\hat{\underline{\mathbf{X}}}(:, :, i), R)$ ;
4    $\hat{\underline{\mathbf{U}}}(:, :, i) = \underline{\mathbf{U}}$ ;
5    $\hat{\underline{\mathbf{S}}}(:, :, i) = \underline{\mathbf{S}}$ ;
6    $\hat{\underline{\mathbf{V}}}(:, :, i) = \underline{\mathbf{V}}$ ;
7 end
8 for  $i = \lceil \frac{I_3+1}{2} \rceil + 1, \dots, I_3$  do
9    $\hat{\underline{\mathbf{U}}}(:, :, i) = \text{conj}(\hat{\underline{\mathbf{U}}}(:, :, I_3 - i + 2))$ ;
10   $\hat{\underline{\mathbf{S}}}(:, :, i) = \hat{\underline{\mathbf{S}}}(:, :, I_3 - i + 2)$ ;
11   $\hat{\underline{\mathbf{V}}}(:, :, i) = \text{conj}(\hat{\underline{\mathbf{V}}}(:, :, I_3 - i + 2))$ ;
12 end
13  $\underline{\mathbf{U}}_R = \text{ifft}(\hat{\underline{\mathbf{U}}}, [], 3)$ ,  $\underline{\mathbf{S}}_R = \text{ifft}(\hat{\underline{\mathbf{S}}}, [], 3)$ ,  $\underline{\mathbf{V}}_R = \text{ifft}(\hat{\underline{\mathbf{V}}}, [], 3)$ ;
```

---

The tensor QR (t-QR) decomposition can be defined similarly based on the t-product. To be precise, for the t-QR decomposition of a tensor  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ , i.e.,  $\underline{\mathbf{X}} = \underline{\mathbf{Q}} * \underline{\mathbf{R}}$ , we first compute the FFT of the tensor  $\underline{\mathbf{X}}$  as

$$\hat{\underline{\mathbf{X}}} = \text{fft}(\underline{\mathbf{X}}, [], 3), \quad (3)$$

and then the QR decompositions of all frontal slices of the tensor  $\hat{\underline{\mathbf{X}}}$  are computed as follows

$$\hat{\underline{\mathbf{X}}}(:, :, i) = \hat{\underline{\mathbf{Q}}}(:, :, i) \hat{\underline{\mathbf{R}}}(:, :, i). \quad (4)$$

Finally, the IFFT operator is applied to the tensors  $\hat{\underline{\mathbf{Q}}}$  and  $\hat{\underline{\mathbf{R}}}$  to compute the tensors  $\underline{\mathbf{Q}}$  and  $\underline{\mathbf{R}}$ . A given data tensor  $\underline{\mathbf{X}}$ , can be orthogonalized by applying the t-QR decomposition and taking the  $\underline{\mathbf{Q}}$  part. We use the notation  $\text{orth}(\underline{\mathbf{X}})$  to denote this operation. It is not difficult to see that the t-SVD and the t-QR decomposition for matrices ( $I_3 = 1$ ) are reduced to the classical matrix SVD and QR decomposition. Here,  $\text{orth}(\underline{\mathbf{X}})$  gives the orthogonal part of the matrix QR decomposition. In Matlab, this is equivalent to  $\underline{\mathbf{Q}} = \text{qr}(\underline{\mathbf{X}}, 0)$ . The tubal LU (t-LU) decomposition [35] can be defined in an analogous way by replacing LU decomposition with the QR decomposition in (4). We remark that the rank-revealing QR and LU decompositions and their randomized variants were extended to tensors based on the t-product in [35] and can also be used in this work.

#### 4. A Pass-Efficient randomized algorithm for computation of the t-SVD

It is obvious that Algorithm 2 is prohibitive for large-scale tensors because of the computation of multiple SVDs of some large-scale matrices. Let us first describe the idea of the random projection technique [26, 36] to tackle this problem. In the first stage of the random projection method, the size of a given data tensor  $\underline{\mathbf{X}}$  is reduced by multiplying the  $\underline{\mathbf{X}}$  with a random tensor  $\underline{\Omega} \in \mathbb{R}^{I_2 \times (R+P) \times I_3}$  as  $\underline{\mathbf{Y}} = \underline{\mathbf{X}} * \underline{\Omega}$ . Then, the tensor  $\underline{\mathbf{Y}}$  is orthogonalized through the t-QR decomposition of the tensor  $\underline{\mathbf{Y}}$  to provide the following low tubal rank approximation:

$$\underline{\mathbf{X}} \cong \underline{\mathbf{Q}} * \underline{\mathbf{B}}, \quad (5)$$

where  $\underline{\mathbf{B}} = \underline{\mathbf{Q}}^T * \underline{\mathbf{X}}$ , and  $\underline{\mathbf{Q}} \in \mathbb{R}^{I_1 \times R \times I_3}$ ,  $\underline{\mathbf{B}} \in \mathbb{R}^{R \times I_2 \times I_3}$ . Note  $R+P \ll I_2$ , where  $R$  is an estimation of the tubal rank and  $P$  is the *oversampling* parameter to better capture the action of the tensor  $\underline{\mathbf{X}}$  [26]. From the low tubal rank approximation (5), and the t-SVD of  $\underline{\mathbf{B}}$  as  $\underline{\mathbf{B}} = \underline{\hat{\mathbf{U}}} * \underline{\mathbf{S}} * \underline{\mathbf{V}}^T$ , we can recover the t-SVD of  $\underline{\mathbf{X}}$  as  $\underline{\mathbf{X}} = (\underline{\mathbf{Q}} * \underline{\hat{\mathbf{U}}}) * \underline{\mathbf{S}} * \underline{\mathbf{V}}^T$ . It is worth mentioning that the tensor  $\underline{\mathbf{B}}$  is smaller than the original data tensor  $\underline{\mathbf{X}}$  and requires less memory and computational resources. This approach is efficient when the singular values of the frontal slices decay very fast; otherwise, the power iteration should be utilized. Here, the original tensor  $\underline{\mathbf{X}}$  is replaced with  $\underline{\mathbf{Z}} = (\underline{\mathbf{X}} * \underline{\mathbf{X}}^T)^q * \underline{\mathbf{X}}$  and the above-mentioned procedure is applied to  $\underline{\mathbf{Z}}$ . Due to stability issues, the tensor  $\underline{\mathbf{Z}}$  should not be explicitly computed. Instead, it can be efficiently computed using the subspace iteration method [14, 26].

The basic randomized algorithm for the low-rank matrix approximation is summarized in Algorithm 3. The "for" loop (Lines 3-6), is the power iteration



technique and is used when the singular values of a matrix do not decay fast. It has been established that in practice,  $q = 1, 2, 3$  are sufficient to achieve good accuracy. In addition,  $P$  is the oversampling parameter, which helps to better capture the range of the matrix. These strategies have been generalized to develop fast algorithms for computation of the t-SVD. Algorithm 4, is an extension of Algorithm 3 to the case of tensors based on the t-product. For computational efficiency, one can replace the t-QR decomposition in Algorithm 4 with the t-LU decomposition [35, 37, 38], or alternatively use a combination of t-QR and t-LU decompositions [37, 39]. Similar to Algorithm 3, the power iteration procedure discussed earlier is performed in Algorithm 4, in Lines (3-6). For power iteration  $q$ , Algorithms 3 and 4 need to pass the data tensor  $(2q + 2)$  times. Indeed, for Algorithm 4 two passes in lines (2, &7) and  $2q$  passes in lines (3-6) are required. This means that the number of passes over the data tensor is always an even number. To the best of our knowledge, the only paper, which proposes a single-pass randomized algorithm for computing the t-SVD is [36], which is a generalization of those proposed in [40] from matrices to tensors. In [19], the authors resolved the drawback of the mentioned limitation of the randomized subspace algorithms for matrices and developed randomized algorithms, which are applicable for a budget of any number of passes. This algorithm is presented in Algorithm 5. Motivated by this efficient algorithm, we extend it to the tensor case based on the t-product. The proposed pass-efficient algorithm can compute a low tubal rank approximation of the underlying data tensor using an arbitrary number of passes and is presented in Algorithm 6. Besides, we exploit the proposed method to develop a fast algorithm to solve the tensor completion problem as a practical application.

Now, we describe the main procedure for computing the t-SVD in a pass-efficient way. Assuming that we afford a budget of  $v$  passes (the number of possible passes) for computing the t-SVD, it is clear that if  $v \geq 4$  is even, then we can use the classical randomized subspace Algorithm 4 with the power iteration parameter  $q = (v - 2)/2$ . So, how about making the amount of passes odd?

Inspired by the idea presented in [19], we suggest the following procedures for computation of the t-SVD given  $v \geq 2$  passes:

- (If  $v$  is even) Construct an orthonormal tensor  $\underline{\mathbf{Q}}$  for the range of the tensor  $(\underline{\mathbf{X}} * \underline{\mathbf{X}}^T)^{(v-2)/2} * \underline{\mathbf{X}} * \underline{\mathbf{\Omega}}$ . Then, compute the truncated t-SVD of the tensor  $\underline{\mathbf{X}}^T * \underline{\mathbf{Q}}$  as

$$[\underline{\mathbf{V}}, \underline{\mathbf{S}}, \hat{\underline{\mathbf{U}}}] = \text{Trunacted t-SVD}(\underline{\mathbf{X}}^T * \underline{\mathbf{Q}})$$

and set  $\underline{\mathbf{U}} = \underline{\mathbf{Q}} * \hat{\underline{\mathbf{U}}}$ .

- (If  $v$  is odd) Construct an orthonormal tensor  $\underline{\mathbf{Q}}$  for the range of the tensor  $(\underline{\mathbf{X}}^T * \underline{\mathbf{X}})^{(v-1)/2} * \underline{\mathbf{\Omega}}$ . Then, compute the truncated t-SVD of the tensor  $\underline{\mathbf{X}}^T * \underline{\mathbf{Q}}$  as

$$[\underline{\mathbf{U}}, \underline{\mathbf{S}}, \hat{\underline{\mathbf{V}}}] = \text{Truncated t-SVD}(\underline{\mathbf{X}}^T * \underline{\mathbf{Q}})$$

and set  $\underline{\mathbf{V}} = \underline{\mathbf{Q}} * \hat{\underline{\mathbf{V}}}$ .

---

**Algorithm 3:** Classical randomized subspace method for computation of the Truncated SVD [39]

---

**Input :** A data matrix  $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ ; a matrix rank  $R$ ; Oversampling  $P$  and the power iteration  $q$ .

**Output:** Truncated t-SVD:  $\mathbf{X} \cong \mathbf{U}\mathbf{S}\mathbf{V}^T$

```

1  $\mathbf{\Omega} = \text{randn}(I_2, P + R)$ ;
2  $[\mathbf{Q}^{(1)}, \sim] = \text{orth}(\mathbf{X}\mathbf{\Omega})$ ;
3 for  $i = 1, 2, \dots, q$  do
4    $[\mathbf{Q}^{(2)}, \sim] = \text{orth}(\mathbf{X}\mathbf{Q}^{(1)})$ ;
5    $[\mathbf{Q}^{(1)}, \sim] = \text{orth}(\mathbf{X}^T \mathbf{Q}^{(2)})$ ;
6 end
7  $[\mathbf{Q}^{(2)}, \mathbf{R}] = \text{orth}(\mathbf{X}^T \mathbf{Q}^{(1)})$ ;
8  $[\hat{\mathbf{V}}, \mathbf{S}, \hat{\mathbf{U}}] = \text{Truncated SVD}(\mathbf{R}, R)$ ;
9  $\mathbf{V} = \mathbf{Q}^{(1)} \hat{\mathbf{V}}$ ;
10  $\mathbf{U} = \mathbf{Q}^{(2)} \hat{\mathbf{U}}$ ;
```

---

Note that the proposed algorithm 6 requires an estimation of the tubal rank as input but a fixed-precision algorithm is proposed in [41] that for a given approximation bound gives the tubal rank and corresponding low tubal rank approximation. The proposed randomized algorithm offers more flexibility in passing the underlying data tensor. The next theorem gives the expected/average upper bound of the approximated tensor computed by Algorithm 3 and we use it to derive similar results for Algorithm 5.

**Theorem 1.** [26] (Average Frobenius error for Algorithm 3). Let  $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$  and  $\mathbf{\Omega} \in \mathbb{R}^{I_2 \times (R+P)}$  be a given matrix and a Gaussian random matrix, respectively with the oversampling parameter  $P \geq 2$ . Suppose  $\mathbf{Q}$  is obtained from Algorithm

---

**Algorithm 4:** Classical randomized subspace method for computation of the Truncated t-SVD [26]

---

**Input** : A data tensor  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ ; a tubal rank  $R$ ; Oversampling  $P$  and the power iteration  $q$ .

**Output:** Truncated t-SVD:  $\underline{\mathbf{X}} \cong \underline{\mathbf{U}} * \underline{\mathbf{S}} * \underline{\mathbf{V}}^T$

```

1  $\underline{\mathbf{\Omega}} = \text{randn}(I_2, P + R, I_3)$ ;
2  $[\underline{\mathbf{Q}}^{(1)}, \sim] = \text{orth}(\underline{\mathbf{X}} * \underline{\mathbf{\Omega}})$ ;
3 for  $i = 1, 2, \dots, q$  do
4    $[\underline{\mathbf{Q}}^{(2)}, \sim] = \text{orth}(\underline{\mathbf{X}} * \underline{\mathbf{Q}}^{(1)})$ ;
5    $[\underline{\mathbf{Q}}^{(1)}, \sim] = \text{orth}(\underline{\mathbf{X}}^T * \underline{\mathbf{Q}}^{(2)})$ ;
6 end
7  $[\underline{\mathbf{Q}}^{(2)}, \underline{\mathbf{R}}] = \text{orth}(\underline{\mathbf{X}}^T * \underline{\mathbf{Q}}^{(1)})$ ;
8  $[\widehat{\underline{\mathbf{V}}}, \underline{\mathbf{S}}, \widehat{\underline{\mathbf{U}}}] = \text{Truncated t-SVD}(\underline{\mathbf{R}}, R)$ ;
9  $\underline{\mathbf{V}} = \underline{\mathbf{Q}}^{(1)} * \widehat{\underline{\mathbf{V}}}$ ;
10  $\underline{\mathbf{U}} = \underline{\mathbf{Q}}^{(2)} * \widehat{\underline{\mathbf{U}}}$ ;
```

---



---

**Algorithm 5:** Randomized truncated SVD with an arbitrary number of passes [19]

---

**Input** : A data matrix  $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ ; a matrix rank  $R$ ; Oversampling  $P$  and the budget of number of passes  $v$ .

**Output:** Truncated t-SVD:  $\mathbf{X} \cong \mathbf{U}\mathbf{S}\mathbf{V}^T$

```

1  $\mathbf{Q}^{(1)} = \text{randn}(I_2, P + R)$ ;
2 for  $i = 1, 2, \dots, v$  do
3   if  $i$  is odd then
4      $[\mathbf{Q}^{(2)}, \mathbf{R}^{(2)}] = \text{orth}(\mathbf{X}\mathbf{Q}^{(1)})$ ;
5   else
6      $[\mathbf{Q}^{(1)}, \mathbf{R}^{(1)}] = \text{orth}(\mathbf{X}^T \mathbf{Q}^{(2)})$ ;
7   end
8 end
9 if  $v$  is even then
10   $[\widehat{\mathbf{V}}, \mathbf{S}, \widehat{\mathbf{U}}] = \text{Truncated SVD}(\mathbf{R}^{(1)}, R)$ ;
11 else
12   $[\widehat{\mathbf{U}}, \mathbf{S}, \widehat{\mathbf{V}}] = \text{Truncated SVD}(\mathbf{R}^{(2)}, R)$ ;
13 end
14  $\mathbf{V} = \mathbf{Q}^{(1)} \widehat{\mathbf{V}}$ ;
15  $\mathbf{U} = \mathbf{Q}^{(2)} \widehat{\mathbf{U}}$ ;
```

---

---

**Algorithm 6:** Proposed pass-efficient randomized truncated t-SVD with an arbitrary number of passes

---

**Input** : A data tensor  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ ; a tubal rank  $R$ ; Oversampling  $P$  and the budget of number of passes  $v$ .

**Output:** Truncated t-SVD:  $\underline{\mathbf{X}} \cong \underline{\mathbf{U}} * \underline{\mathbf{S}} * \underline{\mathbf{V}}^T$

```

1  $\underline{\mathbf{Q}}^{(1)} = \text{randn}(I_2, P + R, I_3)$ ;
2 for  $i = 1, 2, \dots, v$  do
3   if  $i$  is odd then
4      $[\underline{\mathbf{Q}}^{(2)}, \underline{\mathbf{R}}^{(2)}] = \text{orth}(\underline{\mathbf{X}} * \underline{\mathbf{Q}}^{(1)})$ ;
5   else
6      $[\underline{\mathbf{Q}}^{(1)}, \underline{\mathbf{R}}^{(1)}] = \text{orth}(\underline{\mathbf{X}}^T * \underline{\mathbf{Q}}^{(2)})$ ;
7   end
8 end
9 if  $v$  is even then
10   $[\underline{\hat{\mathbf{V}}}, \underline{\hat{\mathbf{S}}}, \underline{\hat{\mathbf{U}}}] = \text{Truncate t-SVD}(\underline{\mathbf{R}}^{(1)}, R)$ ;
11 else
12   $[\underline{\hat{\mathbf{U}}}, \underline{\hat{\mathbf{S}}}, \underline{\hat{\mathbf{V}}}] = \text{Truncated t-SVD}(\underline{\mathbf{R}}^{(2)}, R)$ ;
13 end
14  $\underline{\mathbf{V}} = \underline{\mathbf{Q}}^{(1)} * \underline{\hat{\mathbf{V}}}$ ;
15  $\underline{\mathbf{U}} = \underline{\mathbf{Q}}^{(2)} * \underline{\hat{\mathbf{U}}}$ ;

```

---

3, then

$$\mathbb{E} (\|\mathbf{X} - \mathbf{Q}\mathbf{Q}^T\mathbf{X}\|_F^2) \leq \left(1 + \frac{R}{P-1}\tau_R^{4q}\right) \left(\sum_{j=R+1}^{\min\{I_1, I_2\}} \sigma_j^2\right),$$

where  $R$  is a matrix rank,  $q$  is the power iteration,  $\sigma_j$  is the  $j$ -th singular value of  $\mathbf{X}$ , and  $\tau_R = \sigma_{R+1}/\sigma_R \ll 1$  is the singular value gap.

It is not difficult to check that for an even number of passes, e.g.,  $v = 2q + 2$ , Algorithm 5 is equivalent to Algorithm 3, so Theorem 1 can be used for its error analysis. However, given an odd number of passes, Algorithm 5's error analysis is presented in Theorem 2. Using this, we can derive the average Frobenius norm error for Algorithm 6. Note that in [19], some average error bounds (in the spectral norm) have been established for Algorithm 5, but here we consider the Frobenius norm because it facilitates the derivation of the average error bound of the approximations achieved by the proposed Algorithm 6.

**Theorem 2.** (Average Frobenius error for Algorithm 5). Let  $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$  and  $\mathbf{Q}^{(1)} \in \mathbb{R}^{I_2 \times (R+P)}$  be a given matrix and a Gaussian random matrix respectively

with  $P \geq 2$  being the oversampling parameter. Suppose  $\mathbf{Q}$  is obtained from Algorithm 5, with an odd number of passes  $v$ , then

$$\mathbb{E} (\|\mathbf{X} - \mathbf{X}\mathbf{Q}\mathbf{Q}^T\|_F^2) \leq \left(1 + \frac{R}{P-1} \tau_R^{2(2v-1)}\right) \left(\sum_{j=R+1}^{\min\{I_1, I_2\}} \sigma_j^2\right),$$

where  $R$  is the matrix rank,  $q$  is the power iteration,  $\sigma_j$  is the  $j$ -th singular value of  $\mathbf{X}$ , and  $\tau_R = \sigma_{R+1}/\sigma_R \ll 1$  is the singular value gap.

*Proof.* See the Appendix.  $\square$

Theorem 3 provides the average error bound for Algorithm 4. Similar to the matrix case, Algorithm 6 is reduced to Algorithm 4 when an even number of passes is used hence Theorem 3 can be used for its error analysis. For the case of an odd number of passes, we present Theorem 4, and its proof is quite similar to the proof of Theorem 3.

**Theorem 3.** [26] Given an  $I_1 \times I_2 \times I_3$  tensor  $\underline{\mathbf{X}}$  and a Gaussian tensor  $\underline{\mathbf{\Omega}}$  of size  $I_2 \times (R + P) \times I_3$ , if  $\underline{\mathbf{Q}}$  is obtained from Algorithm 4, then

$$\mathbb{E} (\|\underline{\mathbf{X}} - \underline{\mathbf{Q}} * \underline{\mathbf{Q}}^T * \underline{\mathbf{X}}\|_F) \leq \left(\frac{1}{I_3} \sum_{i=1}^{I_3} \left(1 + \frac{R}{P-1} (\tilde{\tau}_R^{(i)})^{4q}\right) \left(\sum_{j>R} (\tilde{\sigma}_j^{(i)})^2\right)\right)^{1/2},$$

where  $R$  is a tubal rank,  $P \geq 2$  is an oversampling parameter,  $q$  is the power iteration,  $\tilde{\sigma}_j^{(i)}$  is the  $i$ -th component of  $\text{fft}(\underline{\mathbf{S}}(j, j, :), [], 3)$ , and the singular value gap  $\tilde{\tau}_R^{(i)} = \frac{\tilde{\sigma}_{R+1}^{(i)}}{\tilde{\sigma}_R^{(i)}} \ll 1$ .

**Theorem 4.** Given an  $I_1 \times I_2 \times I_3$  tensor  $\underline{\mathbf{X}}$  and an  $I_2 \times (R + P) \times I_3$  Gaussian tensor  $\underline{\mathbf{\Omega}}$ , if  $\underline{\mathbf{Q}}$  is obtained from Algorithm 6, then

$$\mathbb{E} (\|\underline{\mathbf{X}} - \underline{\mathbf{X}} * \underline{\mathbf{Q}} * \underline{\mathbf{Q}}^T\|_F) \leq \left(\frac{1}{I_3} \sum_{i=1}^{I_3} \left(1 + \frac{R}{P-1} (\tilde{\tau}_R^{(i)})^{2(2v-1)}\right) \left(\sum_{j>R} (\tilde{\sigma}_j^{(i)})^2\right)\right)^{1/2},$$

where  $R$  is the tubal rank,  $P \geq 2$  is the oversampling parameter, with an odd number of passes  $v$ ,  $\tilde{\sigma}_j^{(i)}$  is the  $i$ -th component of  $\text{fft}(\underline{\mathbf{S}}(j, j, :), [], 3)$ , and the singular value gap  $\tilde{\tau}_R^{(i)} = \frac{\tilde{\sigma}_{R+1}^{(i)}}{\tilde{\sigma}_R^{(i)}} \ll 1$ .

*Proof.* See the Appendix.  $\square$

## 5. An Application to tensor completion

The problem of recovering a data tensor from only a part of its components is known as *tensor completion* [13]. Let  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  be a given tensor with missing elements, where the indicator set  $\underline{\Omega}$ , stores the location of known (observed) elements. It is generally known that we may effectively recover the underlying original tensor from its incomplete version under the low-rank property assumption [13]. The following is a common tensor decomposition approach to solve the tensor completion problem [42, 13]

$$\begin{aligned} \min_{\underline{\mathbf{X}}} \quad & \|\mathbf{P}_{\underline{\Omega}}(\underline{\mathbf{X}}) - \mathbf{P}_{\underline{\Omega}}(\underline{\mathbf{M}})\|_F^2, \\ \text{s.t.} \quad & \text{Rank}(\underline{\mathbf{X}}) = R, \end{aligned} \quad (6)$$

where  $\underline{\mathbf{M}}$  is the exact data tensor. As described in [43], using an auxiliary variable  $\underline{\mathbf{C}}$ , the optimization problem (6) can be solved more conveniently by the following reformulation

$$\begin{aligned} \min_{\underline{\mathbf{X}}, \underline{\mathbf{C}}} \quad & \|\underline{\mathbf{X}} - \underline{\mathbf{C}}\|_F^2, \\ \text{s.t.} \quad & \text{Rank}(\underline{\mathbf{X}}) = R, \\ & \mathbf{P}_{\underline{\Omega}}(\underline{\mathbf{C}}) = \mathbf{P}_{\underline{\Omega}}(\underline{\mathbf{M}}) \end{aligned} \quad (7)$$

and we can alternatively solve the minimization problem (6) over variables  $\underline{\mathbf{X}}$  and  $\underline{\mathbf{C}}$ . Thus, the solution to the minimization problem (6) can be approximated by the following iterative procedures

$$\underline{\mathbf{X}}^{(n)} \leftarrow \mathcal{L}(\underline{\mathbf{C}}^{(n)}), \quad (8)$$

$$\underline{\mathbf{C}}^{(n+1)} \leftarrow \underline{\Omega} \circledast \underline{\mathbf{M}} + (\underline{\mathbf{1}} - \underline{\Omega}) \circledast \underline{\mathbf{X}}^{(n)}, \quad (9)$$

where  $\mathcal{L}$  is an operator to compute a low-rank tensor approximation of the data tensor  $\underline{\mathbf{C}}^{(n)}$  and  $\underline{\mathbf{1}}$  is a tensor whose all components are equal to one. Note that equation (8) solves the minimization problem (7) over  $\underline{\mathbf{X}}$  for fixed variable  $\underline{\mathbf{C}}$ . Also, Equation (9) solves the minimization problem (7) over  $\underline{\mathbf{C}}$  for fixed variable  $\underline{\mathbf{X}}$ . The algorithm consists of two main steps, *low-rank tensor approximation* (8)

and *Masking computation* (9). It starts from the initial incomplete data tensor  $\underline{\mathbf{X}}^{(0)}$  with the corresponding observation index set  $\underline{\Omega}$  and sequentially improves the approximate solution till some stopping criterion is satisfied or the maximum number of iterations is reached. We do not need to compute the term  $\underline{\Omega} \circledast \underline{\mathbf{M}}$  at each iteration because it is just the initial data tensor  $\underline{\mathbf{X}}^{(0)}$ . The filtering/smoothing procedure is a known technique in signal processing community to improve the image quality. Indeed, in the above procedure, we exploit this idea and smooths out the tensor  $\underline{\mathbf{C}}^{(n+1)}$  before applying the low tensor rank approximation operator  $\mathcal{L}$  to get better results. The first step is computationally expensive steps especially when a large number of iterations is required for convergence or the data tensor is quite large. Here, we use our randomized pass-efficient Algorithm 6 instead of the deterministic algorithms. The experimental results show that this algorithm provides promising results with lower computational cost.

## 6. Simulations

In this section, we test the proposed randomized algorithm on synthetic and real-world datasets. All numerical simulations were performed on a laptop computer with 2.60 GHz Intel(R) Core(TM) i7-5600U processor and 8GB memory. The Peak Signal-to-Noise Ratio (PSNR) and relative error have been utilized to evaluate the performance of the proposed algorithm. The PSNR of two images  $\underline{\mathbf{X}}$  and  $\underline{\mathbf{Y}}$  is defined as

$$\text{PSNR} = 10 \log_{10} \left( \frac{\|\underline{\mathbf{X}}\|_{\infty}}{\|\underline{\mathbf{X}} - \underline{\mathbf{Y}}\|_F} \right) \quad \text{dB}.$$

The relative error is also defined as

$$e(\tilde{\underline{\mathbf{X}}}) = \frac{\|\underline{\mathbf{X}} - \tilde{\underline{\mathbf{X}}}\|_F}{\|\underline{\mathbf{X}}\|_F},$$

where  $\underline{\mathbf{X}}$  is the original tensor and  $\tilde{\underline{\mathbf{X}}}$  is the approximated tensor. We compare the proposed algorithm with the baseline methods: Truncated t-SVD (Algorithm 2) and randomized t-SVD (Algorithm 4).

**Example 1.** In this experiment, we generate a tensor  $\underline{\mathbf{X}} \in \mathbb{R}^{500 \times 500 \times 500}$  with exact tubal rank 15. We set the oversampling parameter  $P = 5$  and apply Algorithm 6 with the tubal rank  $R = 10$  with different numbers of passes over the data tensor  $\underline{\mathbf{X}}$ . In Figure 2 (right), we report the relative error versus the number of

Table 1: Comparing the running time and relative errors achieved by the proposed algorithm, Algorithm 2 and Algorithm 4 for Example 1. The results are for the tubal rank  $R = 10$ .

Algorithms	Running Time (Seconds)	Relative error
Truncated t-SVD [8, 9]	25.52	<b>3.4e-15</b>
Randomized t-SVD [26]	17.65	5.2e-15
Proposed algorithm	<b>8.23</b>	7.1e-15

passes. The results show that with  $v = 2, 3$  passes, we can achieve quite good results and for a larger number of passes, the relative error is smoothly decreased, while the computational complexity is also higher. In Figure 2 (left), we report the running time of Algorithm 6 for different numbers of passes. Please note that the benefit of Algorithm 6 compared with Algorithm 4 is that it does not necessarily need to pass the original data tensor  $\underline{\mathbf{X}}$  four times, and with only two passes, we can achieve reasonable results. This is one benefit of the proposed method over method 4, which lacks this flexibility in terms of the number of passes. In Table 1, we compare the outcomes obtained by the proposed method with those obtained by the truncated t-SVD (Algorithm 2) and the randomized t-SVD (Algorithm 4). The outcome clearly demonstrates that we can get about the same accuracy for the aforementioned data tensor in much less time. It is intriguing that the suggested approach only requires two runs, but Algorithm 4 requires at least four passes (for power iteration  $q = 1$ ). This means that the proposed algorithm even requires less running time than the randomized Algorithm 4.

The output of Algorithm 6 for two and three runs was insignificant since the artificial data tensor utilized in this case was noiseless. Examples 2 and 3 show that the outcomes of two and three passes are important for the real-world datasets (images and videos).

To further examine the proposed approach, the results for different tubal ranks are also reported in Figure 3. We see that the proposed algorithm is still efficient and robust for other tubal ranks.

Additionally, we examined the robustness of the proposed algorithm for situations that the true rank is not selected. To this end, we first considered a larger tubal rank  $R = 15$ , where the algorithm provided an approximation with a relative error of  $2.1231e - 15$  while for a smaller tubal rank  $R = 5$ , the algorithm gave an approximation with a relative error of 0.5619 which is close to the best approximation computed by the t-SVD. To solve the problem of selecting a tubal rank lower than the true tubal rank, one can gradually increase the tubal rank until



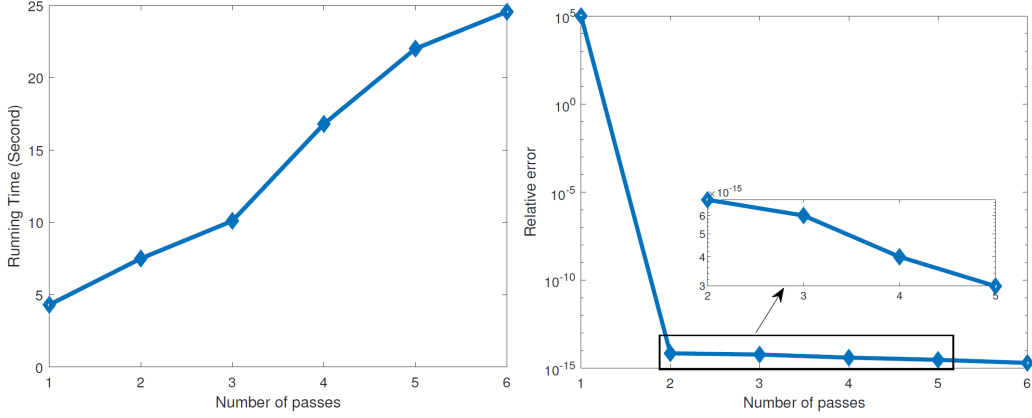


Figure 2: Running time and relative error of the approximations achieved by the proposed algorithm for a synthetic data tensor of size  $500 \times 500 \times 500$  and the tubal rank  $R = 15$  using different numbers of passes for Example 1.

a satisfying approximation is achieved. Indeed, we combined this technique with our proposed algorithm and by gradually increasing the tubal rank from  $R = 5$  to  $R = 10$ , the approximation with a relative error of  $5.4582e - 15$  was achieved. As the proposed algorithm is very fast especially for relatively small tubal ranks, it is also applicable for the exact tubal rank estimation task. These simulations convinced us that it can be efficiently used in different applications.

**Example 2.** In this example, we apply the proposed algorithm to compress color images. To this end, we consider the “Kodim03” and “Kodim23” color images included in the Kodak dataset<sup>1</sup>. We set the oversampling parameter  $P = 6$  and apply Algorithm 6 to the mentioned images with the tubal rank  $R = 40$  for different numbers of passes. The reconstructed images and corresponding results including relative error, PSNR, and running time achieved by the proposed algorithm are reported in Figure 4. As can be seen, the higher the number of passes, the better performance of the images and the higher the computational cost. Moreover, we compare the running time and the PSNR achieved by the proposed algorithm with the Truncated t-SVD (Algorithm 2) and the randomized t-SVD (Algorithm 4) and they are shown in Figure 2. These results indicate that the proposed algorithm can

<sup>1</sup><http://www.cs.albany.edu/~xypan/research/snr/Kodak.html>

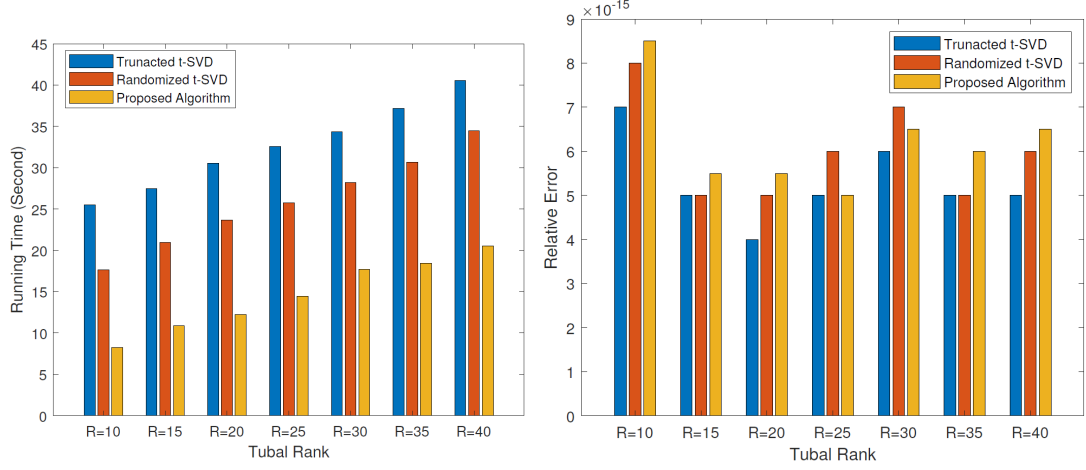


Figure 3: Running time and relative error of the approximations achieved by the truncated t-SVD, the randomized t-SVD and the proposed algorithm for a synthetic data tensor of size  $500 \times 500 \times 500$  for different tubal ranks for Example 1.

provide approximately the same reconstruction (PSNR) as the baseline methods but with less execution time.

**Example 3.** In this experiment, we examine Algorithm 6 for compressing video datasets. We have used the “Foreman” and “News” videos from [44] in this test. Both videos are third-order tensors of size  $144 \times 176 \times 300$ . We set the oversampling parameter  $P = 5$  and apply the proposed algorithm for computing t-SVD with the tubal rank  $R = 20$ . For this tubal rank, we achieve the compression ratio 3.7271. The PSNR of some random samples of the frames for the mentioned two videos and different numbers of passes are reported in Figure 6. Besides, the PSNR of all frames of both videos are shown in Figure 7 (upper). The corresponding running times can be seen in Figure 7 (bottom). Here again, the same results as the previous two simulations are achieved and using more passes over the video dataset, we achieve better results with a higher computational cost. A comparison between the mean of the PSNR of all reconstructed frames by the proposed algorithm and the truncated t-SVD (Algorithm 2) and the randomized t-SVD (Algorithm 4) are made in Table 3. The running time and PSNRs of the reconstructed images obtained by the proposed algorithm and the baselines for different tubal ranks are also compared in 5. Here again, we see that the proposed algorithm provides almost the same reconstruction as the baseline but with less

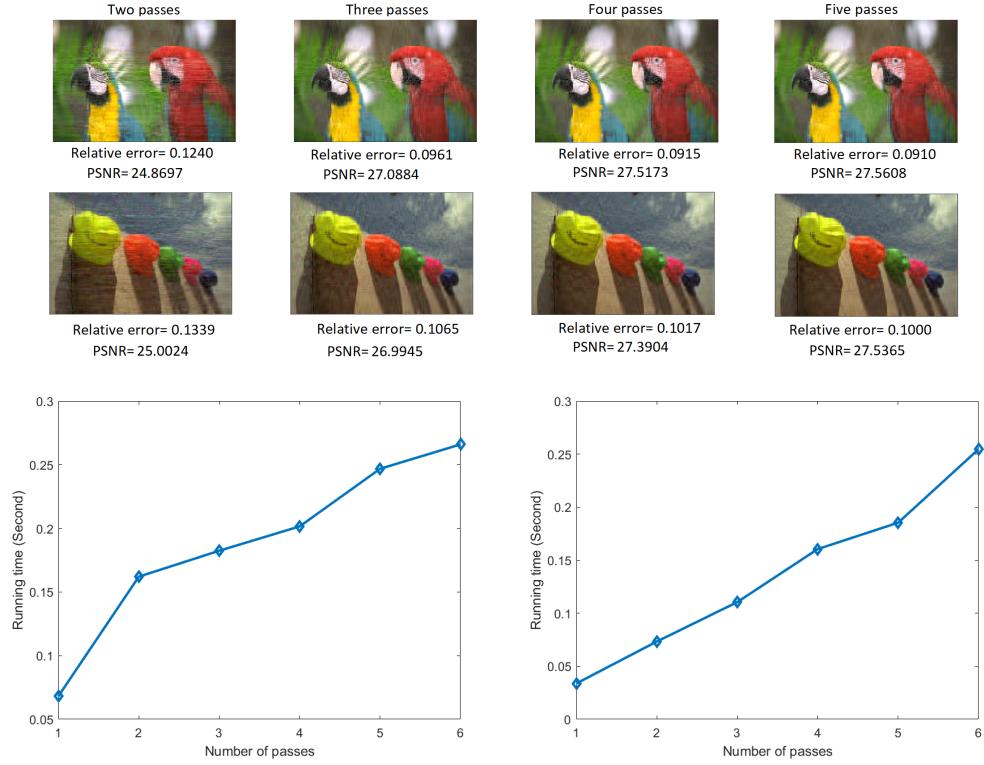


Figure 4: (Upper) The reconstruction of the “Kodim23” and the “Kodim03” images using the proposed algorithm for the tubal rank  $R = 20$  and different numbers of passes (Bottom) The running time of the proposed algorithm for computation of the truncated t-SVD of the “Kodim23” image (left) and the “Kodim03” image (right) with the tubal rank  $R = 40$  and using different numbers of passes for Example 2. .

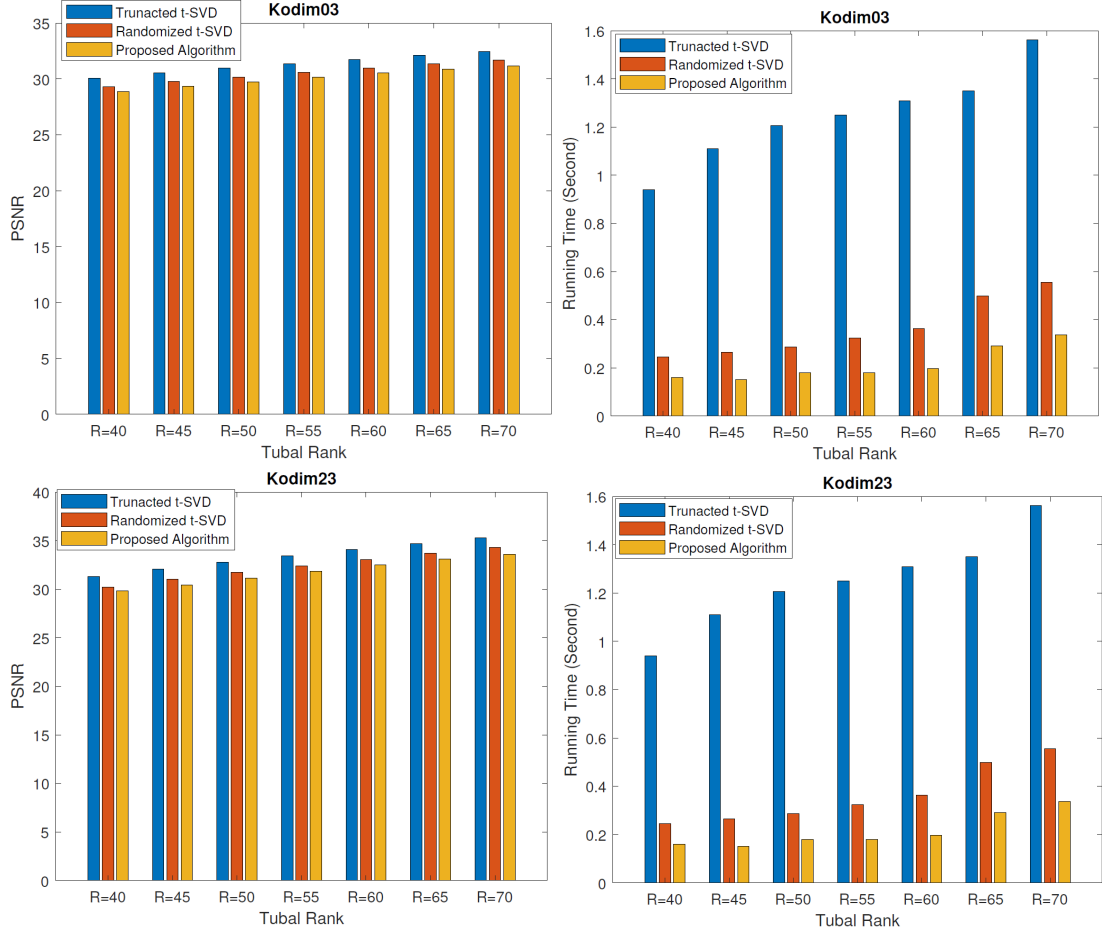


Figure 5: The running time and PSNRs of the reconstructed images achieved by the truncated t-SVD, the randomized t-SVD and the proposed algorithm using different tubal ranks for Example 2.

Table 2: Comparing the running time and the PSNR achieved by the proposed algorithm, Algorithm 2 and Algorithm 4 for Example 2. The results are for the tubal rank  $R = 40$ .

Kodim23 Image		
Algorithms	Running Time (Seconds)	PSNR
Truncated t-SVD [8, 9]	0.45	<b>27.87</b>
Randomized t-SVD [26]	0.23	27.51
Proposed algorithm	<b>0.18</b>	27.38
Kodim03 Image		
Algorithms	Running Time (Seconds)	PSNR
Truncated t-SVD [8, 9]	0.23	<b>27.67</b>
Randomized t-SVD [26]	0.19	27.39
Proposed algorithm	<b>0.10</b>	27.23

running time.

**Example 4.** In this simulation we investigate the effectiveness of Algorithm 6 for the tensor completion task described in Section 5. In our simulations, we make use of both images and videos. First, we consider four color images taken from the well-known Kodak dataset (“Kodim03”, “Kodim15”, “Kodim16”, “Kodim23”). The size of these images is  $512 \times 768 \times 3$ , and we remove 80% of the pixels randomly. Then apply the completion procedure described in Section 5 with the tubal rank  $R = 30$  and use Algorithm 6 with two passes ( $v = 2$ ) and the oversampling parameter  $P = 10$ . The reconstructed images and their PSNRs are displayed in Figure 9. The results show the good performance of the proposed algorithm for the image completion task. A comparison between the proposed algorithm and Truncated t-SVD (Algorithm 2) and randomized t-SVD (Algorithm 4) for the low tubal rank approximation is made and the results are shown in Table 4. The experimental results clearly illustrate that the proposed algorithm can provide the recovered images with approximately the same accuracy but much faster.

To compare the pass-efficient tensor-based (Algorithm 6) and the pass-efficient matrix-based algorithms (Algorithm 5), we consider the pepper image depicted in Figure 10 (first left) that is of size  $256 \times 256 \times 3$  and we remove some pixels of the image in a structure way shown in the second left image. Then, we apply the completion procedure described in Section where the proposed pass-efficient tensor-based algorithm (Algorithm 6) was used for the operator  $\mathcal{L}$ . Also, we reshaped the image to a matrix and then apply the iterative procedures (8)-9 to it, in which the pass-efficient matrix-based algorithm (Algorithm 5) is used or the

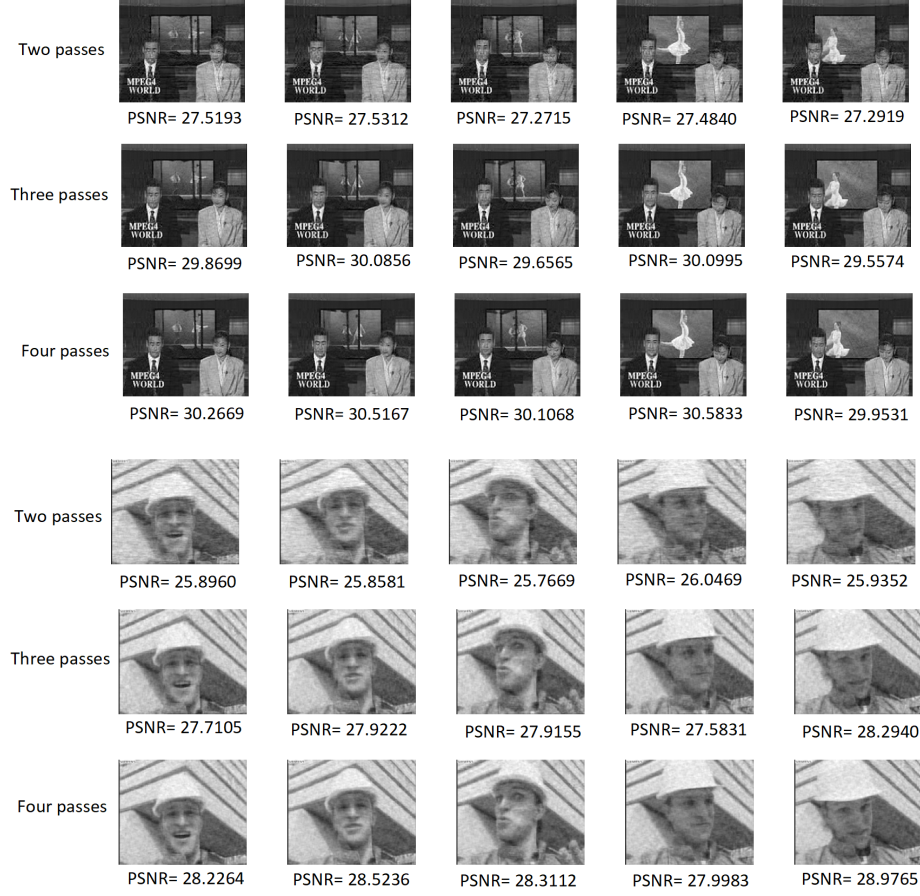


Figure 6: Reconstruction of some random frames for the “News” and the “Foreman” videos using the proposed algorithm. The tubal rank  $R = 20$  and different numbers of passes for Example 3.

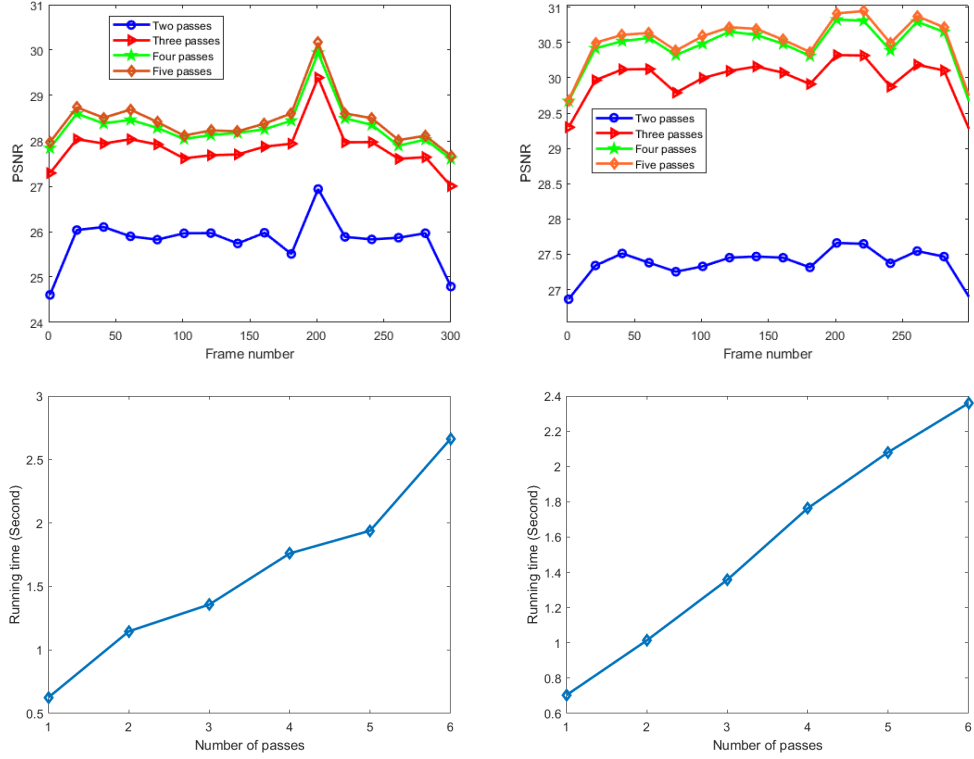


Figure 7: Example 3. (Upper) The results of the proposed algorithm for video compression using different numbers of passes and the tubal rank  $R = 20$  (the left figures are for the “Foreman” video and the right is for the “News” video). (Bottom). The running time of the proposed algorithm for different numbers of passes and the tubal rank  $R = 20$ . The left figure is for the “Foreman” video and the right figure is for the “News” video.

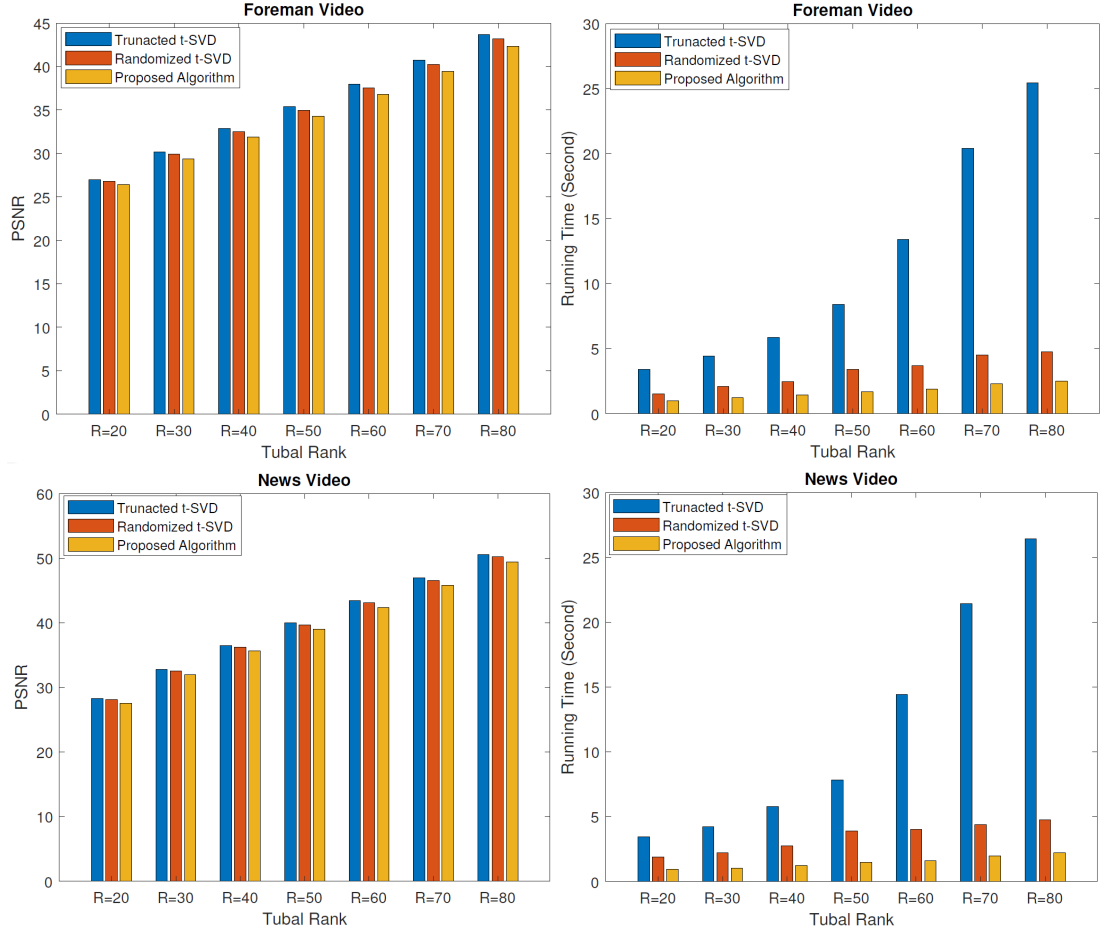


Figure 8: The running time and the mean of the PSNRs of the all reconstructed frames of the videos achieved by the truncated t-SVD, the randomized t-SVD and the proposed algorithm using different tubal ranks 3.



Table 3: Comparing the running time and the mean of the PSNR of all frames achieved by the proposed algorithm, Algorithm 2 and Algorithm 4 for Example 3. The results are for the tubal rank  $R = 25$ .

News dataset		
Algorithms	Running Time (Seconds)	PSNR
Truncated t-SVD [8, 9]	2.64	<b>30.45</b>
Randomized t-SVD [26]	2.02	30.10
Proposed algorithm	<b>1.10</b>	29.65
Foreman dataset		
Algorithms	Running Time (Seconds)	PSNR
Truncated t-SVD [8, 9]	2.67	<b>28.25</b>
Randomized t-SVD [26]	2.13	28.31
Proposed algorithm	<b>1.05</b>	28.09

operator  $\mathcal{L}$ . These reconstructed images using two algorithms are visualized in Figure 10. This indicates that due to the tensor structure-preserving property, the proposed algorithm has overall better reconstruction quality. The proposed algorithm for a given tubal rank  $R$ , provides close to optimal approximation, which is achieved by the truncated t-SVD. However, in some image restoration methods, like matrix factorization-based techniques, the rank ( $R$ ) of a matrix or a tensor is a crucial parameter. This parameter represents the number of significant components or features used to model the image. Choosing an appropriate rank is essential, and it can significantly impact the quality of the restored image. If the user provides an incorrect value for  $R$  (either too large or too small), it can negatively affect the quality of the restored image. A too small value might result in a loss of details, while a too large value may overfit the data, leading to artifacts and noise into the image.

We used the idea known in signal processing, where one starts with a small tubal rank and gradually increases it until the quality of the image is not improved significantly[45]. We should highlight that depending on the specific characteristics of the image and the degradation process, the initial tubal ranks may be different. For example, in our experiments reported in the paper, for images of size  $512 \times 768 \times 3$  and 80% of pixels removed, we started by the tubal rank of 25 and gradually increased the tubal rank for which the tubal rank of 30 provided the best reconstruction. Also, for the images with 90% of pixels missing, we started with the tubal rank of  $R = 15$  and again gradually increased the tubal rank. Here the tubal rank  $R = 21$  provided the best recovery results. Please note that since

the proposed randomized algorithm is fast, running it for several tubal ranks is not a concern and this is one of the advantages of the proposed algorithm.

We next considered three videos “Akiyo”, “Foreman”, and “News”[44], which are third-order tensors of size  $176 \times 144 \times 300$  (collocation of 300 frames or black and white images). We remove 70% of the pixels of the mentioned videos randomly. With the same procedure described for the images, we used our proposed pass-efficient algorithm in the completion stage (8), with two passes ( $v = 2$ ), the oversampling parameter  $P = 10$ , and the tubal rank  $R = 15$  to reconstruct the incomplete videos. The PSNR of all reconstructed frames of the “Akiyo” video are shown in Figure 11 (upper). Also, the original, the observed, and the reconstruction of some frames are displayed in Figure 11 (bottom). The obtained results for the “News” and the “Foreman” videos are reported in Figures 12 and 13, respectively. Similar results as the images were achieved for the case of videos and the proposed algorithms provided almost the same recovery performance as the baseline methods by in a faster time. These results are skipped in order to prevent duplication. The clearly indicates the efficiency of the proposed algorithm at delivering satisfactory results faster.

## 7. Conclusion and future works

In this paper, a pass-efficient randomized algorithm was proposed for the computation of the tensor SVD (t-SVD). Contrary to the classical randomized subspace algorithms, which need an even number of passes over the data tensor, it can find a low tubal rank approximation of a third-order tensor using an arbitrary number of passes. We applied the proposed algorithm for developing a fast completion method to reconstruct images and videos with missing pixels. The suggested approach is for real-valued tensors of third order, but it may be easily extended to higher-order complex tensors. The thorough simulation results demonstrated the feasibility and efficiency of the proposed algorithm. In future work, we plan to extend the block Krylov subspace algorithms to the tensor case based on the t-product and also propose efficient single-pass algorithms for the computation of the t-SVD.

## 8. Acknowledgement

The authors would like to thank the editor and two reviewers for their constructive comments, which have greatly improved the quality of the paper. The



Figure 9: Comparing the original, the observed (with 80% missing pixels) and the reconstructed images using the completion algorithm based on the proposed algorithm using two passes and the tubal rank 30 for Example 4.

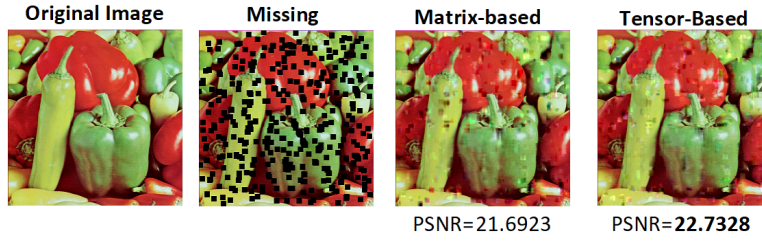


Figure 10: Comparing the matrix and tensor based completion algorithms. For the matrix based method the matrix rank 25 and for the tensor based approach the tubal rank 25 were used for Example 4.

Table 4: Comparison of the running time and the PSNR achieved by the proposed algorithm, Algorithm 2 and Algorithm 4 for Example 4.

Kodim03		
Algorithms	Running Time (Seconds)	PSNR
Truncated t-SVD [8, 9]	17.34	<b>28.01</b>
Randomized t-SVD [26]	10.03	27.93
Proposed algorithm	<b>5.1</b>	27.88
Kodim15		
Algorithms	Running Time (Seconds)	PSNR
Truncated t-SVD [8, 9]	20.13	<b>25.98</b>
Randomized t-SVD [26]	9.84	25.75
Proposed algorithm	<b>4.57</b>	25.60
Kodim16		
Algorithms	Running Time (Seconds)	PSNR
Truncated t-SVD [8, 9]	19.62	<b>27.56</b>
Randomized t-SVD [26]	11.05	27.34
Proposed algorithm	<b>4.89</b>	27.17
Kodim23		
Algorithms	Running Time (Seconds)	PSNR
Truncated t-SVD [8, 9]	19.45	<b>27.89</b>
Randomized t-SVD [26]	11.43	27.75
Proposed algorithm	<b>5.21</b>	27.69

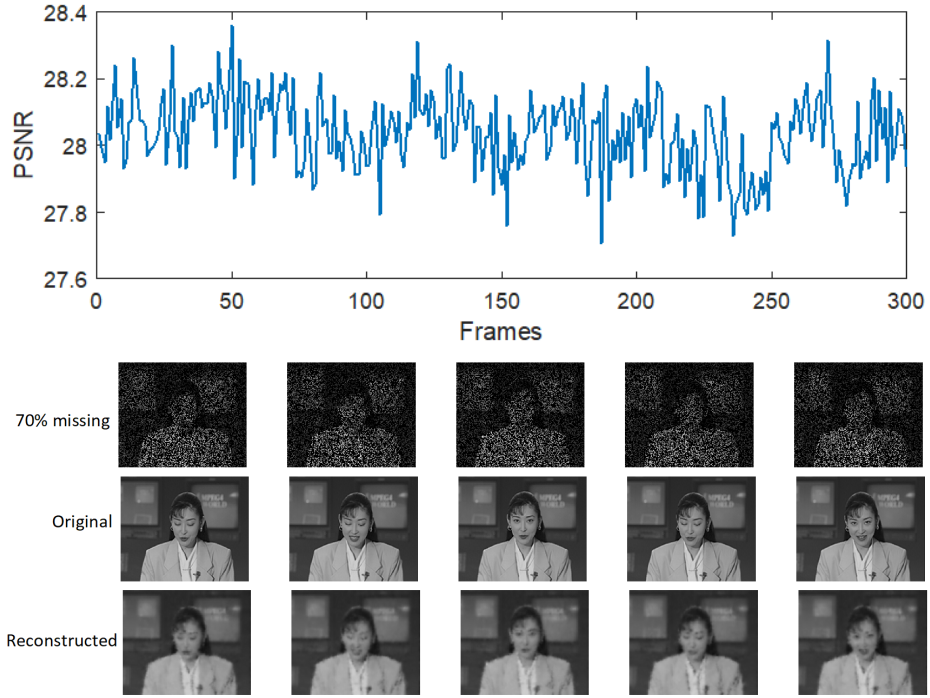


Figure 11: (Upper) The PSNR of all reconstructed frames of the “Akiyo” video using the completion procedure combined by the proposed algorithm. The tubal rank  $R = 15$  and two passes (Bottom) Visualization of some random samples of the original, the observed (70% missing pixels) and the reconstructed frames for Example 4.

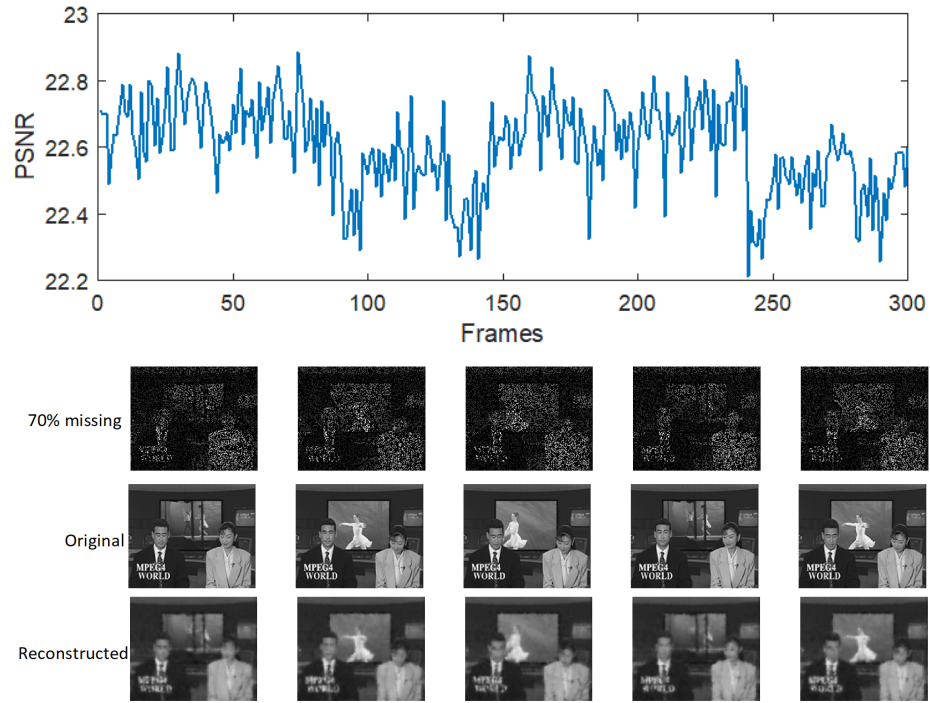


Figure 12: (Upper) The PSNR of all reconstructed frames of the “News” video using the completion procedure combined by the proposed algorithm. The tubal rank  $R = 15$  and two passes (Bottom) Visualization of some random samples of the original, the observed (70% missing pixels) and the reconstructed frames for Example 4.

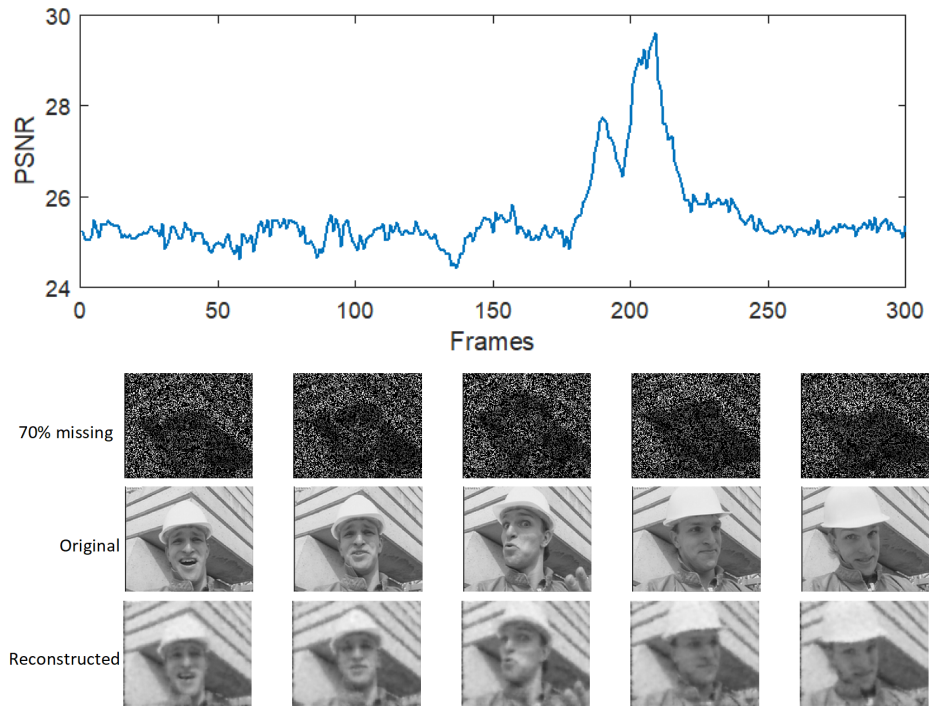


Figure 13: (Upper) The PSNR of all reconstructed frames of the “Foreman” video using the completion procedure combined by the proposed algorithm. The tubal rank  $R = 15$  and two passes (Bottom) Visualization of some random samples of the original, the observed (70% missing pixels) and the reconstructed frames for Example 4.

work was partially supported by the Ministry of Education and Science (grant 075.10.2021.068).

## 9. Conflict of Interest Statement

The author declares that he has no conflict of interest with anything.

## 10. Data Availability

Data openly available in public repositories. The Kodak dataset is accessible at <https://www.kaggle.com/datasets/sherylmehta/kodak-dataset>

## 11. Appendix

The proof of Theorem 2 is almost the same as the proof of Theorem 1 presented in [26]. In fact, they worked on the matrix  $(\mathbf{X}\mathbf{X}^T)^v \mathbf{X}$  and here we consider  $\mathbf{Y} = (\mathbf{X}^T \mathbf{X})^v$ . However, due to some tricky modifications, we provide the details.

Let  $\mathbf{X} = [\mathbf{U}_1, \mathbf{U}_2] \begin{bmatrix} \Sigma_1 & \mathbf{0} \\ \mathbf{0} & \Sigma_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \end{bmatrix}$  and define  $\Omega_1 = \mathbf{V}_1^T \Omega$ ,  $\Omega_2 = \mathbf{V}_2^T \Omega$ , where  $\Sigma_1$  and  $\Sigma_2$  are square. Now, from straightforward computations we have

$$\begin{aligned} \mathbf{Y} = (\mathbf{X}^T \mathbf{X})^v \Omega &= [\mathbf{V}_1 \quad \mathbf{V}_2] \begin{bmatrix} \Sigma_1^{2v} & \mathbf{0} \\ \mathbf{0} & \Sigma_2^{2v} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \end{bmatrix} \Omega = \\ &= [\mathbf{V}_1 \quad \mathbf{V}_2] \begin{bmatrix} \Sigma_1^{2v} & \mathbf{0} \\ \mathbf{0} & \Sigma_2^{2v} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \end{bmatrix}, \end{aligned}$$

and consider the orthogonal projector  $\mathbf{P}_V = \mathbf{V}\mathbf{V}^T$ .

The next propositions are used in our subsequent analysis.

**Proposition 5.** [14] Suppose  $\mathbf{U}$  is unitary. Then  $\mathbf{U}^T \mathbf{P}_M \mathbf{U} = \mathbf{P}_{\mathbf{U}^T \mathbf{M}}$ .

**Proposition 6.** [14] Suppose  $\text{range}(\mathbf{N}) \subset \text{range}(\mathbf{M})$ . Then, for each matrix  $\mathbf{A}$ , it holds that  $\|\mathbf{P}_N \mathbf{A}\| \leq \|\mathbf{P}_M \mathbf{A}\|$  and that  $\|(\mathbf{I} - \mathbf{P}_M) \mathbf{A}\| \leq \|(\mathbf{I} - \mathbf{P}_N) \mathbf{A}\|$ .

To prove Theorem 2, we need to first prove the following theorem.

**Theorem 7.** Let  $\mathbf{X} \in \mathbb{R}^{I \times J}$  with the SVD  $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$  and  $v \geq 0$  be a fixed parameter. Choose  $\Omega_1$  and  $\Omega_2$  as above and assume that  $\Omega_1$  is of full rank. Compute the  $\mathbf{Q}$  (an orthogonal matrix which forms a basis for the range



of  $\mathbf{Y} = (\mathbf{X}^T \mathbf{X})^v \mathbf{\Omega}$  using Algorithm 5 for an odd number of passes  $v$ , then the approximation error satisfies

$$\|\mathbf{X}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\|_F^2 \leq \|\mathbf{\Sigma}_2\|_F^2 + \tau^{2(2v-1)} \|\mathbf{\Sigma}_2 \mathbf{\Omega}_2 \mathbf{\Omega}_1^\dagger\|_F^2. \quad (10)$$

*Proof.* Define matrices  $\mathbf{Z}$  and  $\mathbf{F}$  as

$$\mathbf{Z} = \mathbf{V}^T \mathbf{Y} \mathbf{\Omega}_1^\dagger \mathbf{\Sigma}_1^{-2v} = \begin{bmatrix} \mathbf{I} \\ \mathbf{F} \end{bmatrix}, \quad \mathbf{F} \equiv \mathbf{\Sigma}_2^{2v} \mathbf{\Omega}_2 \mathbf{\Omega}_1^\dagger \mathbf{\Sigma}_1^{-2v}. \quad (11)$$

From the construction of  $\mathbf{Z}$ , we have

$$\text{Range}(\mathbf{Z}) \subset \text{Range}(\mathbf{Y}) \subset \text{Range}(\mathbf{V}^T \mathbf{Y}) \subset \text{Range}(\mathbf{V}^T \mathbf{Q}). \quad (12)$$

It is not difficult to see

$$\begin{aligned} \|\mathbf{X}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\|_F^2 &= \|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T(\mathbf{I} - \mathbf{P}_\mathbf{Q})\|_F^2 = \\ &= \|\mathbf{\Sigma}\mathbf{V}^T(\mathbf{I} - \mathbf{P}_\mathbf{Q})\|_F^2, \end{aligned} \quad (13)$$

and from (12)-13 together with Propositions 5 and 6, we have

$$\begin{aligned} \|\mathbf{\Sigma}\mathbf{V}^T(\mathbf{I} - \mathbf{P}_\mathbf{Q})\|_F^2 &= \|\mathbf{\Sigma}\mathbf{V}^T(\mathbf{I} - \mathbf{P}_\mathbf{Q})\mathbf{V}\mathbf{\Sigma}^T\|_F \leq \\ &= \|\mathbf{\Sigma}(\mathbf{I} - \mathbf{P}_{\mathbf{V}^T \mathbf{Q}})\mathbf{\Sigma}^T\|_F = \|(\mathbf{I} - \mathbf{P}_\mathbf{Z})\mathbf{\Sigma}^T\|_F^2. \end{aligned} \quad (14)$$

Similar to [14, 26], we can prove

$$(\mathbf{I} - \mathbf{P}_\mathbf{Z})\mathbf{\Sigma}^T = \begin{bmatrix} (\mathbf{I} + \mathbf{F}^T \mathbf{F})^{-1} \mathbf{F}^T \mathbf{F} \mathbf{\Sigma}_1 \\ (\mathbf{I} - \mathbf{F}(\mathbf{I} + \mathbf{F}^T \mathbf{F})^{-1}) \mathbf{F}^T \mathbf{\Sigma}_2 \end{bmatrix},$$

and so we come at

$$\begin{aligned} \|(\mathbf{I} - \mathbf{P}_\mathbf{Z})\mathbf{\Sigma}^T\|_F^2 &= \|(\mathbf{I} + \mathbf{F}^T \mathbf{F})^{-1} \mathbf{F}^T \mathbf{F} \mathbf{\Sigma}_1\|_F^2 + \\ &+ \|(\mathbf{I} - \mathbf{F}(\mathbf{I} + \mathbf{F}^T \mathbf{F})^{-1}) \mathbf{F}^T \mathbf{\Sigma}_2\|_F^2. \end{aligned} \quad (15)$$

Now, we bound two terms of (15). For the first term, consider

$$\begin{aligned} \|(\mathbf{I} + \mathbf{F}^T \mathbf{F})^{-1} \mathbf{F}^T \mathbf{F} \mathbf{\Sigma}_1\|_F^2 &\leq \|\mathbf{F}(\mathbf{I} + \mathbf{F}^T \mathbf{F})^{-1}\|_2 \|\mathbf{F} \mathbf{\Sigma}_1\|_F \\ &\leq \|\mathbf{F} \mathbf{\Sigma}_1\|_F, \end{aligned} \quad (16)$$

and for the second term, we get

$$\|(\mathbf{I} - \mathbf{F}(\mathbf{I} + \mathbf{F}^T \mathbf{F})^{-1}) \mathbf{F}^T \mathbf{\Sigma}_2\|_F^2 \leq \|\mathbf{\Sigma}_2\|_F^2, \quad (17)$$

because  $\mathbf{I} - \mathbf{F}(\mathbf{I} + \mathbf{F}^T \mathbf{F})^{-1} \mathbf{F}^T \preceq \mathbf{I}$ , (see [14] for the proof). From (13), (15), (16) and (17), we have

$$\|\mathbf{X}(\mathbf{I} - \mathbf{P}_{\mathbf{Q}})\|_F^2 \leq \|\mathbf{X}(\mathbf{I} - \mathbf{P}_{\mathbf{Z}})\|_F^2 \leq \|\Sigma_2\|_F^2 + \|\mathbf{F}\Sigma_1\|_F^2. \quad (18)$$

It is seen that  $\mathbf{F}\Sigma_1 = \Sigma_2^{2v-1}(\Sigma_2\Omega_2\Omega_1^\dagger)\Sigma_1^{-2v+1}$  and with some straightforward computations we get

$$\|\mathbf{F}\Sigma_1\|_F \leq \|\Sigma_2^{2v-1}\|_2 \|\Sigma_1^{-(2v-1)}\|_2 \|\Sigma_2\Omega_2\Omega_1^\dagger\|_F \leq \tau^{(2v-1)} \|\Sigma_2\Omega_2\Omega_1^\dagger\|_F. \quad (19)$$

From (17) and (19), we can conclude the identity (10).  $\square$

**Proof of Theorem 2.** Combining Theorem 7 and Theorem 8, the desired result can be achieved.

**Theorem 8.** (average Frobenius error) [14]. Suppose that  $\mathbf{A}$  is a real  $m \times n$  matrix with singular values  $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots$ . Choose a target rank  $k \geq 2$  and an oversampling parameter  $p \geq 2$ , where  $k + p \geq \min\{m, n\}$ . Draw an  $n \times (k + p)$  standard Gaussian matrix  $\mathbf{\Gamma}$ , and construct the sample matrix  $\mathbf{Y} = \mathbf{A}\mathbf{\Gamma}$ . Then the expected approximation error  $\mathbb{E}(\|\mathbf{I} - \mathbf{P}_{\mathbf{Y}}\mathbf{A}\|_F) \leq (1 + \frac{k}{p-1})^{1/2} (\sum_{j>k} \sigma_j^2)^{1/2}$ .

**Proof of Theorem 4.** From the linearity of the expectation operator and relation (2) we have

$$\begin{aligned} & \mathbb{E} (\|\underline{\mathbf{X}} - \underline{\mathbf{X}} * \underline{\mathbf{Q}} * \underline{\mathbf{Q}}^T\|_F^2) \leq \\ & \frac{1}{I_3} \left( \sum_{i=1}^{I_3} \mathbb{E} \|\hat{\mathbf{X}}^{(i)} - \hat{\mathbf{X}}^{(i)} \hat{\mathbf{Q}}^{(i)} \hat{\mathbf{Q}}^{(i)T}\|_F^2 \right), \end{aligned} \quad (20)$$

where  $\hat{\mathbf{X}}^{(i)} = \underline{\hat{\mathbf{X}}}(:, :, i)$  and  $\hat{\mathbf{Q}}^{(i)} = \underline{\hat{\mathbf{Q}}}(:, :, i)$ . We can now use Theorem 2 to bound each term of summation (20) as follows

$$\begin{aligned} & \mathbb{E} (\|\hat{\mathbf{X}}^{(i)} - \hat{\mathbf{X}}^{(i)} \hat{\mathbf{Q}}^{(i)} \hat{\mathbf{Q}}^{(i)T}\|_F^2) \leq \\ & \frac{1}{I_3} \left( 1 + \frac{R}{P-1} (\tau_R^{(i)})^{2(2q-1)} \right) \left( \sum_{j>R} (\hat{\sigma}_j^{(i)})^2 \right), \end{aligned}$$

and so we have

$$\begin{aligned} & \mathbb{E} (\|\underline{\mathbf{X}} - \underline{\mathbf{X}} * \underline{\mathbf{Q}} * \underline{\mathbf{Q}}^T\|_F^2) \leq \\ & \sum_{i_3=1}^{I_3} \frac{1}{I_3} \left( 1 + \frac{R}{P-1} (\tau_R^{(i)})^{2(2q-1)} \right) \left( \sum_{j>R} (\hat{\sigma}_j^{(i)})^2 \right). \end{aligned}$$

It suffices to use the Holder's identity as follows

$$\mathbb{E} \left( \|\underline{\mathbf{X}} - \underline{\mathbf{X}} * \underline{\mathbf{Q}} * \underline{\mathbf{Q}}^T\|_F \right) \leq \left( \mathbb{E} \left( \|\underline{\mathbf{X}} - \underline{\mathbf{X}} * \underline{\mathbf{Q}} * \underline{\mathbf{Q}}^T\|_F^2 \right) \right)^{1/2},$$

to get the desired result.

## References

- [1] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, *Journal of Mathematics and Physics* 6 (1-4) (1927) 164–189.
- [2] L. R. Tucker, Implications of factor analysis of three-way matrices for measurement of change, *Problems in measuring change* 15 (1963) 122–137.
- [3] L. R. Tucker, et al., The extension of factor analysis to three-dimensional matrices, *Contributions to mathematical psychology* 110119 (1964).
- [4] I. V. Oseledets, Tensor-train decomposition, *SIAM Journal on Scientific Computing* 33 (5) (2011) 2295–2317.
- [5] S. Östlund, S. Rommer, Thermodynamic limit of density matrix renormalization, *Physical review letters* 75 (19) (1995) 3537.
- [6] M. Espig, K. K. Naraparaju, J. Schneider, A note on tensor chain approximation, *Computing and Visualization in Science* 15 (6) (2012) 331–344.
- [7] Q. Zhao, G. Zhou, S. Xie, L. Zhang, A. Cichocki, Tensor ring decomposition, *arXiv preprint arXiv:1606.05535* (2016).
- [8] M. E. Kilmer, C. D. Martin, Factorization strategies for third-order tensors, *Linear Algebra and its Applications* 435 (3) (2011) 641–658.
- [9] M. E. Kilmer, K. Braman, N. Hao, R. C. Hoover, Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging, *SIAM Journal on Matrix Analysis and Applications* 34 (1) (2013) 148–172.
- [10] W. Hackbusch, S. Kühn, A new scheme for the tensor representation, *Journal of Fourier analysis and applications* 15 (5) (2009) 706–722.
- [11] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, *SIAM review* 51 (3) (2009) 455–500.

- [12] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, D. P. Mandic, et al., Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives, *Foundations and Trends® in Machine Learning* 9 (6) (2017) 431–673.
- [13] Q. Song, H. Ge, J. Caverlee, X. Hu, Tensor completion algorithms in big data analytics, *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13 (1) (2019) 1–48.
- [14] N. Halko, P.-G. Martinsson, J. A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, *SIAM review* 53 (2) (2011) 217–288.
- [15] O. A. Malik, S. Becker, Low-rank Tucker decomposition of large tensors using tensorsketch, in: *Advances in Neural Information Processing Systems*, 2018, pp. 10117–10127.
- [16] M. Che, Y. Wei, Randomized algorithms for the approximations of Tucker and the tensor train decompositions, *Advances in Computational Mathematics* (2018) 1–34.
- [17] O. A. Malik, S. Becker, A sampling-based method for tensor ring decomposition, in: *International Conference on Machine Learning*, PMLR, 2021, pp. 7400–7411.
- [18] S. Ahmadi-Asl, S. Abukhovich, M. G. Asante-Mensah, A. Cichocki, A. H. Phan, T. Tanaka, I. Oseledets, Randomized algorithms for computation of tucker decomposition and higher order svd (hosvd), *IEEE Access* 9 (2021) 28684–28706.
- [19] E. K. Bjarkason, Pass-efficient randomized algorithms for low-rank matrix approximation using any number of views, *SIAM Journal on Scientific Computing* 41 (4) (2019) A2355–A2383.
- [20] Y. Sun, Y. Guo, C. Luo, J. Tropp, M. Udell, Low-rank tucker approximation of a tensor from streaming data, *SIAM Journal on Mathematics of Data Science* 2 (4) (2020) 1123–1150.
- [21] N. Hao, M. E. Kilmer, K. Braman, R. C. Hoover, Facial recognition using tensor-tensor decompositions, *SIAM Journal on Imaging Sciences* 6 (1) (2013) 437–463.

- [22] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, S. Yan, Tensor robust principal component analysis with a new tensor nuclear norm, *IEEE transactions on pattern analysis and machine intelligence* 42 (4) (2019) 925–938.
- [23] E. Kernfeld, M. Kilmer, S. Aeron, Tensor–tensor products with invertible linear transforms, *Linear Algebra and its Applications* 485 (2015) 545–570.
- [24] G. Song, M. K. Ng, X. Zhang, Robust tensor completion using transformed tensor singular value decomposition, *Numerical Linear Algebra with Applications* 27 (3) (2020) e2299.
- [25] T.-X. Jiang, M. K. Ng, X.-L. Zhao, T.-Z. Huang, Framelet representation of tensor nuclear norm for third-order tensor completion, *IEEE Transactions on Image Processing* 29 (2020) 7233–7244.
- [26] J. Zhang, A. K. Saibaba, M. E. Kilmer, S. Aeron, A randomized tensor singular value decomposition based on the t-product, *Numerical Linear Algebra with Applications* 25 (5) (2018) e2179.
- [27] K. Braman, Third-order tensors as linear operators on a space of matrices, *Linear Algebra and its Applications* 433 (7) (2010) 1241–1253.
- [28] D. F. Gleich, C. Greif, J. M. Varah, The power and arnoldi methods in an algebra of circulants, *Numerical Linear Algebra with Applications* 20 (5) (2013) 809–831.
- [29] C. D. Martin, R. Shafer, B. LaRue, An order-p tensor factorization with applications in imaging, *SIAM Journal on Scientific Computing* 35 (1) (2013) A474–A490.
- [30] A. Sobral, E.-h. Zahzah, Matrix and tensor completion algorithms for background model initialization: A comparative evaluation, *Pattern Recognition Letters* 96 (2017) 22–33.
- [31] T.-X. Jiang, T.-Z. Huang, X.-L. Zhao, L.-J. Deng, A novel nonconvex approach to recover the low-tubal-rank tensor data: When t-svd meets pssv, *arXiv preprint arXiv:1712.05870* (2017).
- [32] Q. Jiang, M. Ng, Robust low-tubal-rank tensor completion via convex optimization., in: *IJCAI, 2019*, pp. 2649–2655.

- [33] Y. He, G. K. Atia, Robust low-tubal-rank tensor completion based on tensor factorization and maximum correntopy criterion, arXiv preprint arXiv:2010.11740 (2020).
- [34] S. Ahmadi-Asl, C. F. Caiafa, A. Cichocki, A. H. Phan, T. Tanaka, I. Osleledets, J. Wang, Cross tensor approximation methods for compression and dimensionality reduction, IEEE Access 9 (2021) 150809–150838.
- [35] Y. Zhu, Y. Wei, Tensor LU and QR decompositions and their randomized algorithms, Computational Mathematics and Computer Modeling with Applications (CMCMA) 1 (1) (2022) 1–16.
- [36] L. Qi, G. Yu, T-singular values and t-sketching for third order tensors, arXiv preprint arXiv:2103.00976 (2021).
- [37] H. Li, G. C. Linderman, A. Szlam, K. P. Stanton, Y. Kluger, M. Tygert, Algorithm 971: An implementation of a randomized algorithm for principal component analysis, ACM Transactions on Mathematical Software (TOMS) 43 (3) (2017) 28.
- [38] N. B. Erichson, S. Voronin, S. L. Brunton, J. N. Kutz, Randomized matrix decompositions using R, arXiv preprint arXiv:1608.02148 (2016).
- [39] S. Voronin, P.-G. Martinsson, Rsvdpack: Subroutines for computing partial singular value decompositions via randomized sampling on single core, multi core, and GPU architectures, arXiv preprint arXiv:1502.05366 2 (2015) 16.
- [40] C. Battaglino, G. Ballard, T. G. Kolda, A practical randomized CP tensor decomposition, SIAM Journal on Matrix Analysis and Applications 39 (2) (2018) 876–901.
- [41] S. Ahmadi-Asl, An efficient randomized fixed-precision algorithm for tensor singular value decomposition, Communications on Applied Mathematics and Computation (2022) 1–20.
- [42] E. J. Candès, B. Recht, Exact matrix completion via convex optimization, Foundations of Computational mathematics 9 (6) (2009) 717–772.
- [43] S. Ahmadi-Asl, M. G. Asante-Mensah, A. Cichocki, A. H. Phan, I. Osleledets, J. Wang, Fast cross tensor approximation for image and video completion, Signal Processing (2023) 109121.

- [44] <http://trace.eas.asu.edu/yuv/>.
- [45] Q. Zhao, L. Zhang, A. Cichocki, Bayesian cp factorization of incomplete tensors with automatic rank determination, *IEEE transactions on pattern analysis and machine intelligence* 37 (9) (2015) 1751–1763.