COMPUTATION OF TENSOR DECOMPOSITIONS IN THE BHATTACHARYA–MESNER ALGEBRA WITH APPLICATIONS IN DATA SCIENCE

A dissertation

submitted by

Fan Tian

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in

Mathematics

TUFTS UNIVERSITY

May 2025

Adviser: Professor Misha E. Kilmer

Abstract

Tensors, as multidimensional generalizations of matrices, provide a natural representation for high-dimensional data structures such as video sequences, hyperspectral images, spatio-temporal datasets (e.g., fMRI), and dynamic networks. Initially explored within the psychometrics community in the 20th century, tensors have since gained popularity across a wide range of disciplines, including signal processing, statistics, machine learning, and data science. This thesis investigates the computational aspects and applications of a tensor decomposition method within the Bhattacharya-Mesner (BM) algebraic framework.

Given tensors $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}$ of size $m \times 1 \times n$, $m \times p \times 1$, and $1 \times p \times n$, respectively, their BM-product will result in a third-order tensor of dimension $m \times p \times n$ and BM-rank of 1 (Mesner and Bhattacharya, 1990). Thus, if an arbitrary $m \times p \times n$ third-order tensor can be written as a sum of a small number, relative to m, p, n, of such BM-rank 1 terms, this BM-decomposition (BMD) offers an implicitly compressed representation of the tensor.

In this thesis, we first show that grayscale surveillance video can be accurately captured by a low BM-rank decomposition and give methods for efficiently computing this decomposition. We further establish theoretical connections between rank-revealing matrix factorizations and tensor BMD. We will then introduce a novel BM-product tailored for color videos and propose an algorithm that successfully compresses data while preserving critical temporal features. Next, we extend the application of low BM-rank decomposition to tensor completion problems. The following study investigates the non-negative BM decomposition (NNBMD) for non-negative tensors and derives a multiplicative update algorithm for its computation. Finally, we address the efficient computation of tensor BMD in a streaming context, providing numerical experiments that demonstrate the proposed algorithm's ability to achieve high accuracy with significant computational savings.

To my family and friends.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Professor Misha Kilmer. I am truly grateful for her continuous support and invaluable guidance during my Ph.D. study. She has been a great role model for me by being a caring advisor, a knowledgeable researcher, and a passionate educator. I could not have asked for a better advisor.

I also want to thank Professor Abiy Tasissa for serving as my committee member and for his support along the way. I wish to express my gratitude to Dr. Lior Horesh for hosting me at IBM Research last summer, for serving on my thesis committee, and for great advice on my thesis. My sincere thanks also go to Professor Eric Miller for mentoring me on my projects and for serving on my committee. Last but not least, I would like to thank Professor Abani Patra for his support during my Ph.D. study, for mentoring me on my projects, and for serving as my committee member.

I am truly grateful to my collaborators Professor Mirjeta Pasha, Dr. Vasileios Kalantzis, Dr. Ben Huh, and Dr. Shashanka Ubaru for their encouragement and enormous help on the projects. I also want to thank Professor Elizabeth Newman, my academic sister, for her mentorship and friendship. Additionally, I wish to thank my friends at Tufts for making this department an incredibly fun place to work. Many thanks to the entire Tufts Math Department, for making this school such a welcoming place away from home, and for always hosting events that have connected everyone deeply.

Finally, I would like to thank my parents and my sister for their unconditional love and support. Thanks to my husband and my baby girl for making my life filled with joy and happiness. I love you!

Contents

List of Tables

List of Figures

Computation of tensor decompositions in the Bhattacharya–Mesner algebra with applications in data science

Chapter 1

Introduction

Multi-dimensional data arise in many real-world scenarios, such as EEG recordings or fMRI scans in medical imaging [14, 106], video sequences and hyperspectral imagery in computer vision [22, 44], and gene expression measurements in bioinformatics [54]. Traditional data analysis frameworks that rely on vectors or matrices are often inadequate for capturing the complex structure of such multi-way data, as they typically fail to preserve inter-dimensional relationships and dependencies. For example, images are inherently two-dimensional arrays, and a time series of images naturally forms a three-dimensional data structure. Unfolding these images into vectors and stacking them as matrix columns disrupts the spatial and temporal correlations intrinsic to the data. In contrast, *tensors*, as multi-dimensional generalizations of matrices, provide a more appropriate and expressive framework for modeling such data. For instance, the collection of images can be effectively modeled by a 3rd-order tensor as height × width × time.

While linear algebra is well-developed with profound theoretical foundations, efficient algorithms, and extensive computation analysis, multilinear algebra, or tensor algebra that is crucial for analyzing inherently high-dimensional real-world data is less studies. One of the challenges lies in extending linear algebraic theories and algorithms to higher-order tensors, as the added dimensionality increases complexity. Fundamental properties such as rank, inverse, and orthogonality do not generalize in a straightforward manner from matrices to tensors. A recent development of the tensor Bhattacharya-Mesner (BM) algebra marks a significant advancement in tensor methods. In 1990, Prabir Bhattacharya and Dale Mesner introduced an association scheme generalizing matrix multiplication to higher-order hypermatrices [68,69]. Yuval Filmus and Edinah Gnang further developed several key algebraic properties for hypermatrices including hypermatrix rank-nullity theorem, and spectral theorem,

to multidimensional settings [26, 27]. Despite growing theoretical interest in BM-algebra, its numerical aspects and applications remain under-explored. This thesis seeks to address this gap by introducing the tensor BM-decomposition (BMD) and developing efficient algorithms for its computation. Additionally, we highlight several application domains where the BMD exhibits distinct advantages over conventional tensor decomposition techniques.

This chapter provides the necessary background and notation used throughout the thesis. Section 1.1 introduces the fundamentals of tensor operations. Section 1.2 surveys several well-established tensor decomposition methods. Finally, Section 1.3 outlines the structure of the remainder of the thesis.

1.1 Tensor definition and preliminaries

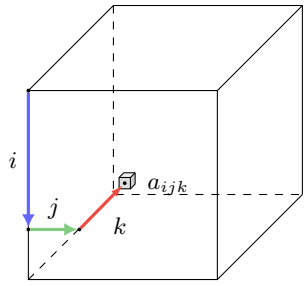
An order-d tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_k \times \cdots \times n_d}$ is a d-dimensional array. The $(i_1, \ldots, i_k, \ldots, i_d)$-th element of $\mathcal{X}$ is denoted by any of the following equivalent notation

$$\mathcal{X}(i_1, \ldots, i_k, \ldots, i_d) \equiv \mathcal{X}_{i_1,\ldots,i_k,\ldots,i_d} \equiv x_{i_1 \ldots i_k \ldots i_d}, \quad 1 \le i_k \le n_k,\ 1 \le k \le d.$$

An order-0 tensor represents a scalar and is denoted by a lowercase letter $a \in \mathbb{R}$. An order-1 tensor represents a vector, denoted by a bold lowercase letter $\mathbf{a} \in \mathbb{R}^n$. Matrices are tensors of order two and are denoted by a bold uppercase letter $\mathbf{A} \in \mathbb{R}^{m \times n}$. A third order tensor of size $m \times n \times p$ is denoted by a bold script letter $\boldsymbol{\mathcal{A}}$, which has three indices, i, j, and k as illustrated in Fig. 1.1. The element a_{ijk} is located at the i-th row, j-th column, and k-th depth of the cube in the figure.

Fibers of a higher-order tensor are the generalizations of the matrix rows and columns. A fiber is obtained from a tensor by fixing every index but one. For example, using MATLAB notation, a mode-k fiber of the d-th order tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times n_k \times n_{k+1} \times \cdots \times n_d}$ is denoted as

$$\mathcal{X}(i_1, \ldots, i_{k-1}, :, i_{k+1}, \ldots, i_d) \in \mathbb{R}^{1 \times \cdots \times 1 \times n_k \times 1 \times \cdots \times 1}, \quad 1 \le i_t \le n_t,\ 1 \le t \le d. \tag{1.1}$$



$\mathcal{A} \in \mathbb{R}^{m \times n \times p}$

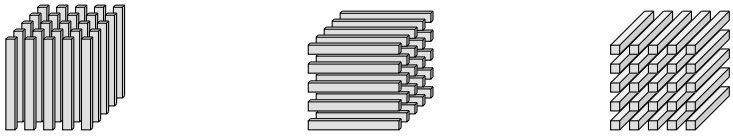
Figure 1.1: The (i, j, k)-th element of a third order tensor $\mathcal{A} \in \mathbb{R}^{m \times n \times p}$, denoted a_{ijk}.

Slices of a tensor are matrices oriented along two modes, which are obtained from a tensor by fixing every index but two. For example, a mode-(j, k) slice of the tensor $\mathcal{X}$ is denoted as

$$\mathcal{X}(i_1, \ldots, i_{j-1}, :, i_{j+1}, \ldots, i_{k-1}, :, i_{k+1}, \ldots, i_d) \in \mathbb{R}^{1 \times \cdots \times 1 \times n_j \times 1 \times \cdots \times 1 \times n_k \times 1 \times \cdots \times 1}, \qquad (1.2)$$

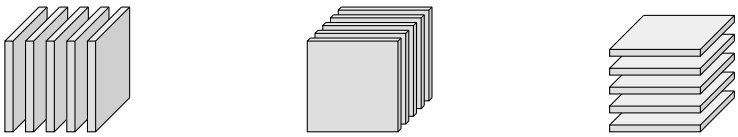
where $1 \leq i_t \leq n_t$, $1 \leq t \leq d$.

We illustrate the fibers and slices for third-order tensors in Fig. 1.2 and Fig. 1.3 respectively.



column fibers $\mathcal{X}_{:,j,k} \in \mathbb{R}^{n_1 \times 1 \times 1}$ | row fibers $\mathcal{X}_{i,:,k} \in \mathbb{R}^{1 \times n_2 \times 1}$ | tube fibers $\mathcal{X}_{i,j,:} \in \mathbb{R}^{1 \times 1 \times n_3}$

Figure 1.2: Fibers of a third-order tensor of size $n_1 \times n_2 \times n_3$ and corresponding indexing in MATLAB notation.



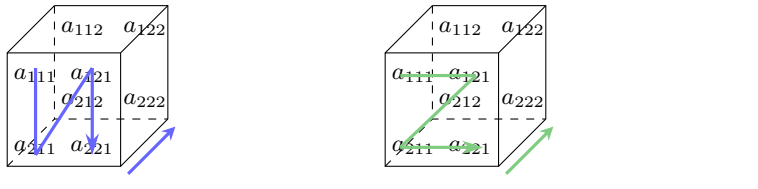
lateral slices $\mathcal{X}_{:,j,:} \in \mathbb{R}^{n_1 \times 1 \times n_3}$ | frontal slices $\mathcal{X}_{:,:,k} \in \mathbb{R}^{n_1 \times n_2 \times 1}$ | horizontal slices $\mathcal{X}_{i,:,:} \in \mathbb{R}^{1 \times n_2 \times n_3}$

Figure 1.3: Slices of a third-order tensor of size $n_1 \times n_2 \times n_3$ and corresponding indexing in MATLAB notation.

Definition 1.1.1 (Mode-k unfolding) *The mode-k unfolding of an order-d tensor* $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *is represented by the matrix* $\mathcal{X}_{(k)} \in \mathbb{R}^{n_k \times n_1 \cdots n_{k-1} n_{k+1} \cdots n_d}$ *and is defined as follows: the* $(i_1, \ldots i_d)$*-th element of the tensor* $\mathcal{X}$ *maps to the* (i_k, j)*-th element of the matrix* $\mathcal{X}_{(k)}$ *where*

$$j = 1 + \sum_{t \neq k}^{d} \left(\prod_{s \neq k}^{t-1} n_s \right) (i_t - 1).$$

Examples of the mode-k, $k = 1, 2, 3$, unfoldings of the tensor $\mathcal{A} \in \mathbb{R}^{2 \times 2 \times 2}$ are illustrated in Fig. 1.4.



$$\begin{pmatrix} a_{111} & a_{121} & a_{112} & a_{122} \\ a_{211} & a_{221} & a_{212} & a_{222} \end{pmatrix} \qquad \begin{pmatrix} a_{111} & a_{211} & a_{112} & a_{212} \\ a_{121} & a_{221} & a_{122} & a_{222} \end{pmatrix} \qquad \begin{pmatrix} a_{111} & a_{211} & a_{121} & a_{221} \\ a_{112} & a_{212} & a_{122} & a_{222} \end{pmatrix}$$

Mode-1 unfolding $\mathcal{A}_{(1)}$ Mode-2 unfolding $\mathcal{A}_{(2)}$ Mode-3 unfolding $\mathcal{A}_{(3)}$

Figure 1.4: Mode-k, $k = 1, 2, 3$, unfolding of the tensor $\mathcal{A} \in \mathbb{R}^{2 \times 2 \times 2}$.

Definition 1.1.2 (Mode-k product) *Let* $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *be an order-d tensor and let* $\mathbf{M} \in \mathbb{R}^{m \times n_k}$ *be a matrix. Then, the mode-k product of* $\mathcal{X}$ *by* $\mathbf{M}$*, is denoted as* $\mathcal{X} \times_k \mathbf{M}$ *is a tensor of size* $n_1 \times \cdots \times n_{k-1} \times m \times n_{k+1} \times \cdots \times n_d$ *whose entries are given by*

$$(\mathcal{X} \times_k \mathbf{M})_{i_1, \ldots, i_{k-1}, j, i_{k+1}, \ldots, i_d} = \sum_{i_k=1}^{n_k} \mathbf{M}_{j, i_k} \mathcal{X}_{i_1, \ldots, i_{k-1}, i_k, i_{k+1}, \ldots, i_d}$$

Definition 1.1.3 *The inner product of two tensors* $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *is the sum of the products of their elements, i.e.*

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1, \ldots, i_d} x_{i_1 \ldots i_d} y_{i_1 \ldots i_d}.$$

This defines the Frobenius norm [53] of a tensor as follows

$$\|\mathcal{X}\|_F = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle} = \left(\sum_{i_1, \ldots, i_d} x_{i_1 \ldots i_d}^2 \right)^{1/2}, \tag{1.3}$$

which is analogous to the matrix Frobenius norm.

Definition 1.1.4 (Kronecker product) *The Kronecker product of matrices* $\mathbf{A} \in \mathbb{R}^{m_1 \times n_1}$ *and* $\mathbf{B} \in \mathbb{R}^{m_2 \times n_2}$ *is denoted by* $\mathbf{A} \otimes \mathbf{B}$ *is a matrix of size* $m_1 m_2 \times n_1 n_2$ *and is defined as*

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1n_1}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2n_1}\mathbf{B} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m_1 1}\mathbf{B} & a_{m_1 2}\mathbf{B} & \cdots & a_{m_1 n_1}\mathbf{B} \end{pmatrix}.$$

Definition 1.1.5 (Khatri-Rao product [53]) *The Khatri-Rao product of two matrices* $\mathbf{A} \in \mathbb{R}^{m \times n}$ *and* $B \in \mathbb{R}^{p \times n}$ *which is denoted by* $\mathbf{A} \odot \mathbf{B}$*, is a matrix of size* $mp \times n$ *and is the Kronecker product of the corresponding columns of* $\mathbf{A}$ *and* $\mathbf{B}$*, i.e.,*

$$\mathbf{A} \odot \mathbf{B} = \begin{pmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_n \otimes \mathbf{b}_n \end{pmatrix},$$

where $\mathbf{a}_i, \mathbf{b}_i$ are the ith column of $\mathbf{A}$ and $\mathbf{B}$ respectively.

Definition 1.1.6 (Hadamard product) *The Hadamard product of two matrices* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$ *is denoted by* $\mathbf{A} * \mathbf{B} \in \mathbb{R}^{m \times n}$*, and is an element-wise product as follows*

$$\mathbf{A} * \mathbf{B} = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2n}b_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1}b_{m1} & a_{m2}b_{m2} & \cdots & a_{mn}b_{mn} \end{pmatrix}$$

Definition 1.1.7 *The outer product of* d *vectors* $\mathbf{a}^{(1)} \in \mathbb{R}^{n_1} \ldots, \mathbf{a}^{(d)} \in \mathbb{R}^{n_d}$ *is an order-*d *tensor* $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *whose elements are defined as*

$$\boldsymbol{\mathcal{X}}_{i_1,\ldots,i_d} = \mathbf{a}^{(1)}_{i_1} \ldots \mathbf{a}^{(1)}_{i_d}, \quad 1 \le i_t \le n_t,\ 1 \le t \le d.$$

Throughout the thesis, we will use MATLAB-derived **SQUEEZE** command to convert tensor slices into matrices. The operation $\mathbf{A} = \texttt{squeeze}(\boldsymbol{\mathcal{A}})$ turns a lateral slice $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{m \times 1 \times n}$ or a horizontal slice $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{1 \times m \times n}$ into a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ [47]. We also

use the **VEC** and **RESHAPE** operators. The `vec` operator maps a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ to a vector $\mathbf{a} \in \mathbb{R}^{mn\times 1}$ by column-major ordering [52]. In MATLAB notation, we have $\texttt{vec}(\mathbf{A}) \equiv \mathbf{A}(:)$. The $\texttt{reshape}(\mathbf{a}, [m, n])$ operation folds a given vector $\mathbf{a} \in \mathbb{R}^{mn\times 1}$ into a matrix $\mathbf{A}$ of size $m \times n$ by filling this matrix one column at a time.

Additionally, we define the **TVEC** and the **TFOLD** operations for unfolding and folding tensors. Given a tensor $\mathcal{X} \in \mathbb{R}^{m\times p\times n}$, we have

$$\mathbf{x} = \texttt{Tvec}(\mathcal{X}) = \begin{bmatrix} \mathbf{x}^{(1,1)} \\ \vdots \\ \mathbf{x}^{(i,j)} \\ \vdots \\ \mathbf{x}^{(m,p)} \end{bmatrix} = \begin{bmatrix} \texttt{squeeze}\left(\mathcal{X}_{1,1,:}\right) \\ \vdots \\ \texttt{squeeze}\left(\mathcal{X}_{i,j,:}\right) \\ \vdots \\ \texttt{squeeze}\left(\mathcal{X}_{m,p,:}\right) \end{bmatrix}.$$

for all $1 \le i \le m, 1 \le j \le p$. This operation is also equivalent to storing the three-dimensional array in sequential memory location using row-major ordering [52]. The `Tfold` operation is defined to be the inverse action of `Tvec`, i.e. $\texttt{Tfold}\left(\texttt{Tvec}(\mathcal{X})\right) = \mathcal{X}$. To make these definitions more concrete, we can think of the operation $\texttt{Tvec}(\mathcal{X})$ as stacking the tube fibers of $\mathcal{X}$ into a long vector $\mathbf{x}$, and the `Tfold` operator reverse the flattening back into a third-order tensor (referring to Fig. (1.5.a) for visual illustration).

We also define the **MAT** operation which flattens two tensors of specific sizes into a block-diagonal matrix (see Fig. 1.5.b). Given $\mathcal{A} \in \mathbb{R}^{m\times \ell\times n}$ and $\mathcal{B} \in \mathbb{R}^{\ell\times p\times n}$, $\texttt{Mat}(\mathcal{A}, \mathcal{B})$ yields $\mathbf{H} \in \mathbb{R}^{mpn\times mp\ell}$ defined as

$$\mathbf{H} = \texttt{Mat}(\mathcal{A}, \mathcal{B}) = \bigoplus_{i,j} \mathbf{H}^{(i,j)}, \quad \forall\, 1 \le i \le m, 1 \le j \le p,$$

where $\mathbf{H}^{(i,j)} \in \mathbb{R}^{n\times \ell}$ with its entries specified as $\mathbf{H}^{(i,j)}_{k,t} = \mathcal{A}_{i,t,k}\mathcal{B}_{t,j,k}$.

We use the MATLAB **PERMUTE** operator to rearrange the dimensions of an array. In particular, tensor transposes are defined based on the cyclic permutations of the indices of each entry. Details are given in Chapter. 2.

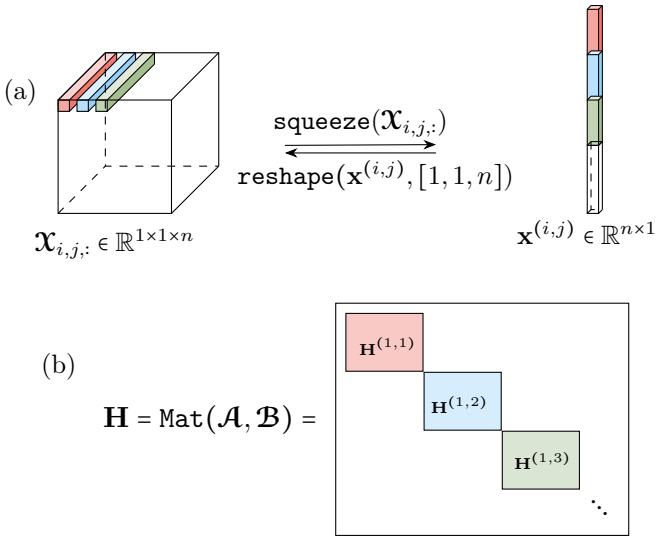


Figure 1.5: (a) Illustration of the tensor $\texttt{Tvec}(\mathcal{X})$ and $\texttt{Tfold}(\mathbf{x})$ operations. The vectorization and the reshaping of the first three tube fibers $\mathcal{X}_{1,1,:}, \mathcal{X}_{1,2,:}$, and $\mathcal{X}_{1,3,:}$ are shown in this figure, and the rest are omitted. (b) Illustration of the tensor Mat operation that flattens the two tensors $\mathcal{A} \in \mathbb{R}^{m\times\ell\times n}, \mathcal{B} \in \mathbb{R}^{\ell\times p\times n}$ into a block-diagonal matrix $\mathbf{H} \in \mathbb{R}^{mpn\times mp\ell}$.

1.2 Tensor decomposition methods

The study of tensors and their decompositions dates back to the work of Hitchcock in 1927 [39, 40]. Beginning in the 1970s, tensor decomposition methods gained traction within the fields of psychometrics and chemometrics [2, 35]. Since the early 2000s, the growing availability of computational resources and an improved understanding of multilinear algebra have expanded the adoption of tensor-based methods to other disciplines, including signal processing, neuroscience, and machine learning [53]. In this section, we review several of the most widely used tensor decomposition techniques, namely the CANDECOMP/PARAFAC (CP) decomposition, Tucker method, and the tensor singular value decomposition (t-SVDM) based on the $\star_{\mathbf{M}}$ product, in order to highlight their distinguishing features and contrast them with the proposed tensor Bhattacharya-Mesner decomposition (BMD).

1.2.1 CANDECOMP/PARAFAC decomposition

The CANDECOMP/PARAFAC (CP) decomposition, also known as the tensor rank decomposition, extends the classical concept of matrix factorization to higher-order

tensors. Originally introduced by Hitchcock in 1927 [39, 40], the approach was later developed and popularized under the names CANDECOMP (canonical decomposition) by Carroll and Chang [12], and PARAFAC (parallel factors) by Harshman [35]. The CP decomposition (CPD) expresses a given tensor as a sum of rank-one tensors, thereby providing a compact and interpretable representation of multi-way data.

We begin by defining a rank-one tensor as an outer product of vectors as follows

Definition 1.2.1 (Rank-one tensor) *Given a set of d vectors $\mathbf{a}^{(i)} \in \mathbb{R}^{n_i}$ for $i = 1, ..., d$, define a d-dimensional tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ as the outer product of these vectors*

$$\boldsymbol{\mathcal{A}} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(d)},$$

where $\circ$ represents the vector outer-product, and $\boldsymbol{\mathcal{A}}_{i_1, i_2, \ldots, i_d} = \mathbf{a}^{(1)}_{i_1} \mathbf{a}^{(2)}_{i_2} \cdots \mathbf{a}^{(d)}_{i_d}$ for $1 \leq i_k \leq n_d$ and $1 \leq k \leq d$. Then $\boldsymbol{\mathcal{A}}$ is a rank-one tensor.

Definition 1.2.2 (Tensor rank) *Suppose $\boldsymbol{\mathcal{A}}$ is a d-dimensional tensor. The rank of $\boldsymbol{\mathcal{A}}$ is the minimal number of rank-one tensors whose sum is equal to $\boldsymbol{\mathcal{A}}$; that is, if*

$$\boldsymbol{\mathcal{A}} \approx \sum_{t=1}^{r} \mathbf{a}^{(1)}_t \circ \mathbf{a}^{(2)}_t \circ \cdots \circ \mathbf{a}^{(d)}_t$$

and r is minimal, then rank$(\boldsymbol{\mathcal{A}}) = r$.

Next we define the higher-order CP decomposition as follows.

Definition 1.2.3 (CP decomposition) *Given $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, the tensor-rank-$r$ CP decomposition is the following*

$$\boldsymbol{\mathcal{X}} \approx \sum_{t=1}^{r} \lambda_t \cdot \mathbf{a}^{(1)}_t \circ \mathbf{a}^{(2)}_t \circ \cdots \circ \mathbf{a}^{(d)}_t$$

where $\mathbf{a}^{(k)}_t \in \mathbb{R}^{n_k}$, $1 \leq k \leq d$ are normalized vectors and $\lambda_t \in \mathbb{R}$ is the weight for $1 \leq t \leq r$.

For example, given a third-order tensor $\mathcal{X} \in \mathbb{R}^{m \times n \times p}$, the tensor rank-$r$ CP decomposition of $\mathcal{X}$ can be expressed as

$$\mathcal{X} \approx \sum_{t=1}^{r} \lambda_t \cdot \mathbf{a}_t \circ \mathbf{b}_t \circ \mathbf{c}_t$$

where $\mathbf{a}_t \in \mathbb{R}^m, \mathbf{b}_t \in \mathbb{R}^n$, and $\mathbf{c}_t \in \mathbb{R}^p$ are vectors (normalized by their ℓ^2 norms) for $t = 1, \ldots, r$.

The factor matrices of the CPD are matrices $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_t & \cdots & \mathbf{a}_r \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_t & \cdots & \mathbf{b}_r \end{bmatrix}$, and $\mathbf{C} = \begin{bmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_t & \cdots & \mathbf{c}_r \end{bmatrix}$, which are the collections of the factor vectors from the rank-one components. Then the CPD can be re-written in the unfolding form along each mode as follows,

$$\mathbf{X}_{(1)} \approx \mathbf{A}(\mathbf{C} \odot \mathbf{B})^\top,$$
$$\mathbf{X}_{(2)} \approx \mathbf{B}(\mathbf{C} \odot \mathbf{A})^\top,$$
$$\mathbf{X}_{(3)} \approx \mathbf{C}(\mathbf{B} \odot \mathbf{A})^\top.$$

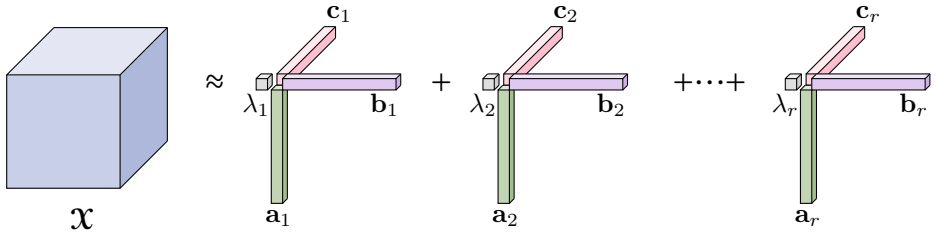


Figure 1.6: Illustration of the CP decomposition of a third-order tensor.

Although the definition of tensor rank parallels that of matrix rank, the computational complexity of determining tensor rank is substantially greater. While efficient algorithms such as Gaussian elimination exist for computing the rank of arbitrary matrices [96], the problem of determining the rank of a general tensor has been shown to be NP-hard [36, 38]. Consequently, in practical applications, it is common to employ a rank-k approximation rather than seeking an exact rank

decomposition [4, 53] by solve the following minimization problem

$$\min_{\mathcal{Y}} \|\mathcal{X} - \mathcal{Y}\|_F \text{ with } \mathcal{Y} = \sum_{t=1}^{r} \lambda_t \cdot \mathbf{a}_t \circ \mathbf{b}_t \circ \mathbf{c}_t, \tag{1.4}$$

where $\mathbf{a}_t \in \mathbb{R}^m, \mathbf{b}_t \in \mathbb{R}^n$, and $\mathbf{c}_t \in \mathbb{R}^p$ are normalized vectors for $t = 1, \ldots, r$.

There are many algorithms to solve Eq. (1.4). Among those, the most common approaches are iterative optimization techniques, including alternating least squares (CP-ALS), gradient descent, and quasi-Newton methods [4].

1.2.2 Tucker model

The Tucker decomposition method was introduced by Tucker in 1963 [97]. The decomposition is a higher-order extension of PCA, representing a tensor as a core tensor multiplied by factor matrices along each mode. Using the mode unfolding, the Tucker decomposition finds the best multi-linear rank approximation.

We first review the multi-linear rank for the order-3 case is defined as follows.

Definition 1.2.4 *Let $\mathcal{X} \in \mathbb{R}^{m \times n \times p}$. The multi-linear rank r is a tuple where $r_k = rank(\mathcal{X}_{(k)})$ for $1 \le k \le 3$ and $\mathcal{X}_{(k)}$ is the mode-k unfolding of $\mathcal{X}$.*

As shown in Fig. 1.7, the rank $r = (r_1, r_2, r_3)$ Tucker decomposition of $\mathcal{X}$ is then given as follows

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C},$$

where $\mathbf{A} \in \mathbb{R}^{m \times r_1}$, $\mathbf{B} \in \mathbb{R}^{n \times r_2}$, and $\mathbf{C} \in \mathbb{R}^{p \times r_3}$ are the factor matrices and can be thought of as the principal components in each mode. The tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is called the core tensor. The unfolding form of the Tucker decomposition can be written as

$$\mathcal{X}_{(1)} \approx \mathbf{A}\mathcal{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^\top,$$
$$\mathcal{X}_{(2)} \approx \mathbf{B}\mathcal{G}_{(2)}(\mathbf{C} \otimes \mathbf{A})^\top,$$
$$\mathcal{X}_{(3)} \approx \mathbf{C}\mathcal{G}_{(3)}(\mathbf{B} \otimes \mathbf{A})^\top.$$

The method for computing the Tucker decomposition of a given tensor was introduced by De Lathauwer, De Moor, and Vandewalle, and is called the higher-order SVD (HOSVD) [17].

Definition 1.2.5 *Given* $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$*, the HOSVD is*

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3,$$

where $\mathcal{G} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ *is the core and* $\mathbf{U}_k \in \mathbb{R}^{n_k \times n_k}$ *is the matrix containing the left-singular vectors of the matrix SVD of the mode-k unfolding of* $\mathcal{X}$*:*

$$\mathcal{X}_{(k)} = \mathbf{U}_k \cdot \mathbf{\Sigma}_k \cdot \mathbf{V}_k^\top, \quad \text{for } k = 1, 2, 3.$$

The core $\mathcal{G}$ *is formed as follows*

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}_1^\top \times_2 \mathbf{U}_2^\top \times_3 \mathbf{U}_3^\top.$$

A figure illustration of the third-order tensor HOSVD is shown in Fig. (1.7).

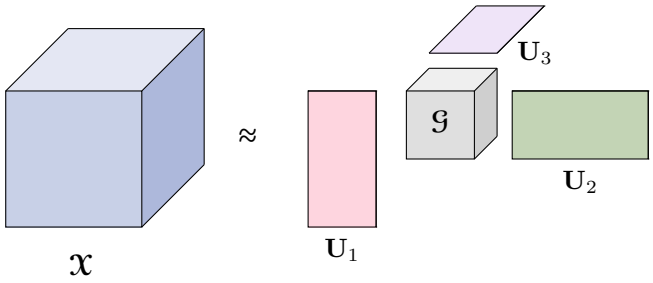


Figure 1.7: Illustration of the HOSVD of a third-order tensor.

While the matrix SVD provides the optimal low-rank approximation in the Frobenius norm, the truncated HOSVD (i.e., using truncated factor matrices) does not necessarily yield the optimal low multi-linear rank Tucker decomposition [17].

1.2.3 Tensor t-SVDM

The tensor t-SVDM is a tensor SVD method based on the $\star_{\mathbf{M}}$-product [45]. This multiplicative operation is defined between pairs of third-order tensors with compatible dimensions. It is a generalization the tensor t-product, which was first proposed by Kilmer and Martin in 2011, [48]. Over the past decade, the t-SVDM framework has gained increasing attentions attention and demonstrated considerable success across a range of applications, including dictionary learning [74, 90], facial recognition [33], data compression [47], and neural networks [73]. In what follows, we first review the definitions of these tensor-tensor product operations, specifically the t-product and the $\star_{\mathbf{M}}$-product.

Definition 1.2.6 (t-product) *Let $\mathcal{A}$ be an $m \times p \times n$ tensor and $\mathcal{B}$ be an $p \times \ell \times n$ tensor. The t-product of $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{C} = \mathcal{A} \star \mathcal{B}$, is an $m \times \ell \times n$ tensor*

$$\mathcal{C} = \texttt{fold}(\texttt{bcirc}(\mathcal{A})) \cdot \texttt{unfold}(\mathcal{B}).$$

Definition 1.2.7 ($\star_{\mathbf{M}}$-product) *Let $\mathbf{M}$ be any invertible $n \times n$ matrix, $\mathcal{A} \in \mathbb{R}^{m \times \ell \times n}$, and $\mathcal{B} \in \mathbb{R}^{\ell \times p \times n}$. Let $\hat{\mathcal{A}} := \mathcal{A} \times \mathbf{M}$ and $\hat{\mathcal{B}} := \mathcal{B} \times \mathbf{M}$. Compute $\hat{\mathcal{C}}_{:,:,i} = \hat{\mathcal{A}}_{:,:,i} \hat{\mathcal{B}}_{:,:,i}$ for all $i = 1, \ldots, n$, and define $\mathcal{C} = \hat{\mathcal{C}} \times_3 \mathbf{M}^{-1}$, then $\mathcal{C} = \mathcal{A} \star_{\mathbf{M}} \mathcal{B} \in \mathbb{R}^{m \times p \times n}$.*

The $\star_{\mathbf{M}}$-product allows us to define the tensor singular value decomposition (t-SVDM).

Definition 1.2.8 (t-SVDM) *For $\mathcal{X} \in \mathbb{R}^{m \times p \times n}$, the t-SVDM of $\mathcal{X}$ is given by*

$$\mathcal{X} = \mathcal{U} \star_{\mathbf{M}} \mathcal{S} \star_{\mathbf{M}} \mathcal{V}^{\top} = \sum_{i=1}^{r} \mathcal{U}_{:,i,:} \star_{\mathbf{M}} \mathcal{S}_{i,i,:} \star_{\mathbf{M}} \mathcal{V}^{\top}_{:,i,:},$$

where $\mathcal{U}$ and $\mathcal{V}$ are $\star_{\mathbf{M}}$-unitary tensors of sizes $m \times \ell \times n$ and $\ell \times p \times n$ respectively, and $\mathcal{S} \in \mathbb{R}^{m \times p \times n}$ is a tensor whose frontal slices are diagonal.

When $\mathbf{M}$ is the DFT matrix, this reduces to the t-product based t-SVD introduced in [48].

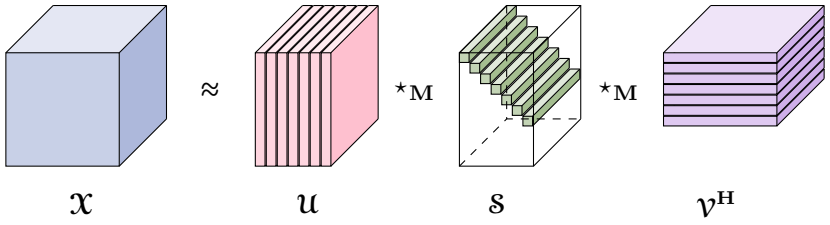


Figure 1.8: Illustration of the third-order tensor t-SVDM.

The tensor t-SVDM has several advantages over other tensor decomposition methods. Within the $\star_{\mathbf{M}}$ framework, it yields an optimal low-rank approximation of a tensor [47]. Additionally, the method is straightforward to implement and inherently parallelizable. Moreover, the $\star_{\mathbf{M}}$ framework preserves many matrix properties such as identity, inverse and orthogonality in high dimensional spaces. However, a limitation of the t-SVDM is its orientation dependence; that is, the quality of the approximation is influenced by the specific dimension to which the matrix $\mathbf{M}$ is applied.

This thesis focuses on operations and decompositions based on the Bhattacharya-Mesner (BM) product for several key reasons:

- The BM-product is orientation independent and naturally extends to tensors of arbitrary order.
- The BM-algebraic framework supports well-defined notions such as inverse, orthogonality, the spectral and rank-nullity theorems, Parseval's identity, and higher-order Fourier tensors [26, 27].
- A substantial portion of this work focuses on compressed representations and interpretable factor tensors derived from the BM decomposition (BMD). The results in [93] constitute one of the earliest computational contributions to tensor decomposition within the BM framework, including detailed implementation techniques, discussions of the algorithm, and convergence analysis.

1.3 Organization of the thesis

The remainder of this thesis is organized as follows. Chapter 2 reviews the Bhattacharya-Mesner (BM) product and its associated algebraic properties. Chapter 3 introduces the third-order tensor BM-decomposition (BMD) and proposes an alternating least squares algorithm (BMD-ALS) for solving the decomposition problem, with an application to grayscale video background/foreground separation. In Chapter 4, we extend the BMD to fourth-order tensors and introduce orientation-dependent variants of the BM-product. The effectiveness of the order-4 BMD is demonstrated through its application to color video background/foreground separation tasks. Chapter 5 presents a low BM-rank tensor completion method based on a slice-wise nuclear norm penalization, solved via an efficient alternating direction method of multipliers (ADMM) algorithm. Experimental results on real-world videos and hyperspectral images are provided, with performance compared to the HaLRTC method [62]. Chapter 6 addresses the non-negative BM-decomposition (NNBMD) of non-negative tensors. We propose a multiplicative update (MU) algorithm generalized from the case for computing the non-negative matrix factorization (NMF), and demonstrate its advantages in brain MRI image compression and facial recognition tasks. Finally, Chapter 7 considers the streaming BMD problem and introduces an iterative algorithm (*onlineBMD*), which achieves comparable accuracy to BMD-ALS with significantly reduced computation time. We further extend this approach to the multi-aspect streaming setting where the data is continuously streamed in all three dimensions.

Chapter 2

Tensor BM-algebra

2.1 Overview of the tensor BM-product

The tensor Bhattacharya-Mesner product (BMP) of order-3 tensors was first introduced as the association scheme on triples in [68,69]. The BMP naturally generalizes the matrix product to higher dimensions. In particular, the BMP of order-2 tensors corresponds to the usual matrix product.

Definition 2.1.1 (Matrix product) *For a conformable matrix pair* $\mathbf{A} \in \mathbb{R}^{m\times\ell}, \mathbf{B} \in \mathbb{R}^{\ell\times n}$*, the matrix product* $\mathbf{X} = \mathbf{AB} \in \mathbb{R}^{m\times n}$ *is given entry-wise by*

$$\mathbf{X}_{i,j} = \sum_{1\le t\le \ell} \mathbf{A}_{i,t}\mathbf{B}_{t,j}.$$

The BMP of third-order tensors is defined as follows.

Definition 2.1.2 *For a third-order conformable tensor triplet* $\mathcal{A} \in \mathbb{R}^{m\times\ell\times n}, \mathcal{B} \in \mathbb{R}^{m\times p\times\ell}$*, and* $\mathcal{C} \in \mathbb{R}^{\ell\times p\times n}$*, the BM-product* $\mathcal{X} = \mathtt{bmp}(\mathcal{A},\mathcal{B},\mathcal{C}) \in \mathbb{R}^{m\times p\times n}$ *is given entry-wise by*

$$\mathcal{X}_{i,j,k} = \sum_{1\le t\le \ell} \mathcal{A}_{i,t,k}\mathcal{B}_{i,j,t}\mathcal{C}_{t,j,k}. \tag{2.1}$$

We can easily extend the BMP to tensors of arbitrary order.

Definition 2.1.3 *For an* m*-tuple conformable order-*m *tensors* $\mathcal{A}^{(1)} \in \mathbb{R}^{n_1\times\ell\times n_3\times\cdots\times n_m}$, $\ldots$, $\mathcal{A}^{(j)} \in \mathbb{R}^{n_1\times\cdots\times n_j\times\ell\times n_{j+2}\times\cdots\times n_m}$, $\ldots$, $\mathcal{A}^{(m)} \in \mathbb{R}^{\ell\times n_2\times\cdots\times n_m}$*, the BM-product* $\mathcal{X} = \mathtt{bmp}\left(\mathcal{A}^{(1)},\ldots\mathcal{A}^{(j)},\ldots,\mathcal{A}^{(m)}\right) \in \mathbb{R}^{n_1\times\cdots\times n_j\times\cdots\times n_m}$ *is given entry-wise by*

$$\mathcal{X}_{i_1,\ldots,i_j,\ldots,i_m} = \sum_{1\le t\le \ell} \mathcal{A}^{(1)}_{i_1,t,i_3,\ldots,i_m}\cdots\mathcal{A}^{(j)}_{i_1,\ldots,i_j,t,i_{j+2},\ldots,i_m}\cdots\mathcal{A}^{(m)}_{t,i_2,\ldots,i_m}.$$

Unless otherwise specified, we will use BMP to refer to third-order tensor multiplication for the remainder of this thesis.

The BMP expresses the notion of the BM-inner product. In the third-order tensor case, the (i,j,k)-th entry of $\boldsymbol{\mathcal{X}}$ is the inner product of the (i,k)-th row-fiber of $\boldsymbol{\mathcal{A}}$, (i,j)-th tube-fiber of $\boldsymbol{\mathcal{B}}$, and the (j,k)-th column-fiber of $\boldsymbol{\mathcal{C}}$. A graphic illustration is given in Fig. (2.1).

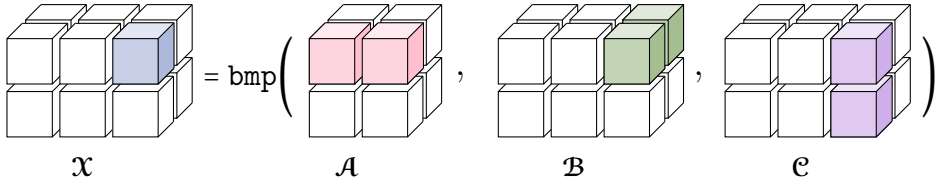


Figure 2.1: Illustration of the BM-inner product of a row-fiber of $\boldsymbol{\mathcal{A}}$, a tube-fiber of $\boldsymbol{\mathcal{B}}$, and a column-fiber of $\boldsymbol{\mathcal{C}}$

The BMP also conveniently expresses the notion of the BM-outer product. When $\ell = 1$, a BM-outer product corresponds to the BMP of the conformable order-2 tensor (matrix) slices, i.e. a lateral slice $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{m\times 1\times n}$, a frontal slice $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{m\times p\times 1}$, and a horizontal slice $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{1\times p\times n}$. Consequently, the BM-product $\boldsymbol{\mathcal{X}} = \texttt{bmp}\,(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}})$ given in Eq. (2.1) can be written equivalently as a sum of BM outer-products of matrix slices

$$\boldsymbol{\mathcal{X}} = \sum_{1\le t\le \ell} \texttt{bmp}\,(\boldsymbol{\mathcal{A}}_{:,t,:}, \boldsymbol{\mathcal{B}}_{:,:,t}, \boldsymbol{\mathcal{C}}_{t,:,:})\,. \tag{2.2}$$

A figure illustration is shown in Fig. (2.2).

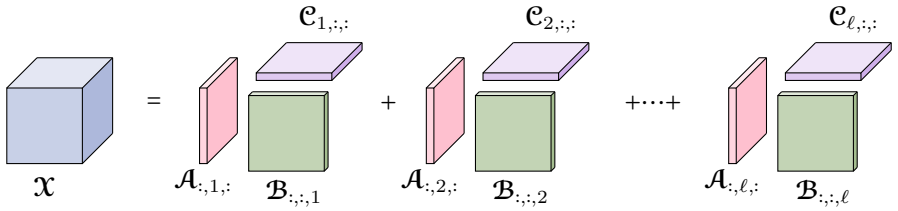


Figure 2.2: Illustration of the BM-product of a conformable tensor triplet $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}$, and $\boldsymbol{\mathcal{C}}$ as a sum of BM-outer products of matrix slices.

Remark 2.1.4 Every BM-product expresses a sum of BM-outer products and vice versa. ♢

Similar to how the matrix transpose corresponds to a permutation of the indices,

a tensor transpose under the BMP framework corresponds to cyclic permutations of the three indices of a given tensor.

Definition 2.1.5 (Tensor transpose [26]) *Suppose $\mathcal{X}$ is a third-order tensor of size $m \times p \times n$, then $\mathcal{X}$ has the following transpose operations which are given by cyclic permutations of the indices of each entry:*

$$\mathcal{X}^{\top}_{i,j,k} = \mathcal{X}_{k,i,j}, \quad \mathcal{X}^{\top} \in \mathbb{R}^{p\times n\times m}; \qquad \mathcal{X}^{\top^2}_{i,j,k} = \mathcal{X}_{j,k,i}, \quad \mathcal{X}^{\top^2} \in \mathbb{R}^{n\times m\times p}.$$

Note that in MATLAB, we can conveniently perform the transpose by the command $\mathcal{X}^{\top} = \texttt{permute}(\mathcal{X}, [2,3,1])$. Then $\mathcal{X}^{\top^2} = \texttt{permute}(\mathcal{X}, [3,1,2])$ and $\mathcal{X}^{\top^3} = \texttt{permute}(\mathcal{X}, [1,2,3]) = \mathcal{X}$.

When $\mathcal{X}$ is a BM-product of tensors $\mathcal{A}, \mathcal{B}$ and $\mathcal{C}$, then the transpose of the BM-product is a BM-product of transposes such that

$$\mathcal{X}^{\top} = \texttt{bmp}(\mathcal{A}, \mathcal{B}, \mathcal{C})^{\top} = \texttt{bmp}\left(\mathcal{B}^{\top}, \mathcal{C}^{\top}, \mathcal{A}^{\top}\right).$$

2.2 BM-algebraic properties

Within the BM-algebra framework, several matrix properties can be generalized directly to higher-order tensors. This section focuses on properties defined specifically for third-order tensors, as outlined in [24, 26, 27]. While these definitions are not directly used in this thesis, they are included to provide a more comprehensive description of the BM-algebra framework.

Definition 2.2.1 (Identity pair) *Given a third-order tensor $\mathcal{A} \in \mathbb{R}^{m\times p\times n}$, a tensor pair $(\mathcal{I}_1, \mathcal{I}_2) \in \mathbb{R}^{m\times n\times n} \times \mathbb{R}^{n\times p\times n}$ is an identity pair of $\mathcal{A}$ if*

$$\texttt{bmp}(\mathcal{I}_1, \mathcal{A}, \mathcal{I}_2) = \mathcal{A}.$$

By analogy to matrix inverse $\mathbf{A}^{-1}$ where for a matrix $\mathbf{A}, \mathbf{A}^{-1}$ is its inverse if $(\mathbf{MA})\mathbf{A}^{-1} = \mathbf{M}$ for any non-zero matrix $\mathbf{M}$.

Definition 2.2.2 (Inverse pair) *Given a third-order tensor* $\mathcal{X} \in \mathbb{R}^{m\times p\times n}$*, a tensor inverse pair* $(\mathcal{A}_1, \mathcal{A}_2) \in \mathbb{R}^{m\times n\times n} \times \mathbb{R}^{n\times p\times n}$ *and* $(\mathcal{B}_1, \mathcal{B}_2) \in \mathbb{R}^{m\times n\times n} \times \mathbb{R}^{n\times p\times n}$ *is an inverse pair of* $\mathcal{X}$ *if*

$$\texttt{bmp}(\mathcal{B}_1, \texttt{bmp}(\mathcal{A}_1, \mathcal{X}, \mathcal{A}_2), \mathcal{B}_2) = \mathcal{X}.$$

Definition 2.2.3 (Kronecker delta) *The Kronecker delta* $\mathbf{\Delta} \in \mathbb{R}^{n\times n\times n}$ *such that*

$$\mathbf{\Delta}_{i,j,k} = \begin{cases} 1, & \text{if } i = j = k \\ 0, & \text{otherwise} \end{cases}$$

When the tensor is of order-2, the Kronecker delta reduces to the identity matrix.

Orthogonality is a key concept in linear algebra. Gnang, Elgammal, and Retakh first introduced the notion of orthogonality for third-order tensors under the BM-algebra framework in [25]. One interpretation of matrix orthogonality is associated with the correlation constraints and arises from expressing the constraints for a given matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ such that

$$\Delta = \mathbf{Q}\mathbf{Q}^\top \Leftrightarrow \langle \mathbf{Q}_{i,:}, \mathbf{Q}^\top_{:,j} \rangle = \left(\sum_{1\leq t\leq \ell} q_{it} q_{jt} \right) = \delta_{ij},\ \forall\ 1 \leq i, j \leq n.$$

Definition 2.2.4 (Orthogonality) *Tensor orthogonality is associated with the correlation constraints and arises from expressing the constraints for a given tensor* $\mathbf{\Omega} \in \mathbb{R}^{n\times n\times n}$

$$\mathbf{\Delta} = \texttt{bmp}\left(\mathbf{\Omega}, \mathbf{\Omega}^{\top^2}, \mathbf{\Omega}^\top\right)$$
$$\Leftrightarrow \texttt{bmp}\left(\mathbf{\Omega}_{i,:,k}, \mathbf{\Omega}^{\top^2}_{i,j,:}, \mathbf{\Omega}^\top_{:,j,k}\right) = \left(\sum_{1\leq t\leq \ell} q_{itk} q_{jti} q_{ktj} \right) = \delta_{ijk},\ \forall\ 1 \leq i, j, k \leq n.$$

The BM-outer product in Eq.(2.2) induces a natural notion of the tensor BM-rank.

Definition 2.2.5 (BM-rank [27]) *The BM-rank of* $\mathcal{X} \in \mathbb{R}^{m\times p\times n}$ *is the minimum number of the BM-outer products of conformable matrix slices that sum up to* $\mathcal{X}$*.*

The BM-rank of a third-order tensor is equal to the BM-rank of its transpose.

Chapter 3

Tensor BM-Decomposition and Application to Video Data

Low rank tensor decomposition methods have provided domain-specific insight into large, multidimensional data sets as well as a means of compressing these data [53]. Many such methods have been proposed including the CANDECOMP/PARAFAC or canonical polyadic (CP) decomposition [46,70], the Tucker model or the higher-order SVD (HOSVD) method [17,98], tensor-train decomposition [76], t-SVD [48] and its more general form $\star_M$ tensor SVD (t-SVDM) [47]. The use of a specific method on a particular problem depends heavily on the underlying application (i.e., the properties of the data) as well as the processing objectives (compression, information extraction, etc.). Of interest here are the compression and the decomposition of video into stationary background and moving foreground components. In [44], regularized CP decomposition was used for the video background estimation.

We begin by introducing the tensor BM-decomposition (BMD) [93] based on the tensor BM-algebra framework. We develop an alternating least-squares (ALS) algorithm for solving the decomposition problem. Our examples in the video background and foreground separation show that the tensor BMD provides compressive separation results with high video quality. The fundamental difference between factoring a tensor using the BM-product representation and the tensor CP-decomposition lies in the order of the set of "factor tensors" into which a given third-order tensor (the case of interest here) is decomposed. The CP approach is based on outer products generated from triplets of vectors, as illustrated in Fig. (1.6), while a BM-decomposition (BMD) employs triplets of matrices, as we show in Fig. (2.2). Theoretical studies of third-order tensor spectral decomposition and singular value decomposition in terms

of BM-product have been discussed in [26] and [28] respectively. However, no numerical algorithms on tensor BM-decomposition methods have been proposed until recently. In 2022, we first presented an alternating least squares (ALS) algorithm to factor a third-order tensor into an unconstrained BMD [94], which served as motivation for the present work. Independently, in 2023, Luo et. al [65] proposed an unconstrained tensor factorization framework based on the third-order tensor BMP, which they rename as matrix outer-product (MOP). They, too, propose an ALS algorithm for an application in Bayesian inference, but do not address issues including starting guess or convergence.

This chapter begins by establishing the connections between rank-revealing matrix factorizations and the tensor BM-decomposition. We then address the unconstrained low BM-rank approximation problem and introduce an alternating least-squares algorithm (BMD-ALS) for computing the tensor BMD, along with its convergence analysis. Subsequently, we provide an overview of video background/foreground separation and discuss two commonly used initialization methods—Dynamic Mode Decomposition (DMD) and Spatio-temporal Slice-based SVD (SS-SVD). A generative spatiotemporal video model is proposed to illustrate the intrinsic low BM-rank structure of grayscale video tensors. A constrained variant of BMD-ALS is also presented to solve a regularized formulation that promotes separation into stationary and non-stationary components. Finally, this chapter concludes with a comprehensive numerical study demonstrating the effectiveness of the third-order BMD in compressible video background/foreground reconstruction.

3.1 Matrix-to-BMD Connections

In this section, we will show the connections between rank-revealing matrix factorizations and the tensor BMD, both on the unfolded data and on the frontal slices of the data. These results are proven to be useful in the following chapter to establish starting guesses for an iterative algorithm to solve the tensor BM-decomposition

problem. We emphasize using these results to establish a bound on the tensor BM-rank in this section.

3.1.1 Matrix based decomposition

Let $\mathcal{X} \in \mathbb{R}^{m\times p\times n}$. Construct a matrix $\mathbf{X} \in \mathbb{R}^{mn\times p}$ such that $\mathbf{X}_{:,j} = \texttt{vec}(\mathcal{X}_{:,j,:})$, i.e. $\mathbf{X} = \mathcal{X}_{(1)}$. We will show that a matrix rank-revealing decomposition of $\mathbf{X}$ can be rewritten as a tensor BMD:

Theorem 3.1.1 *Suppose* $\mathbf{X}$ *has the following rank-revealing decomposition* $\mathbf{X} = \mathbf{U}\mathbf{V}^\top$*, where* $\mathbf{U} \in \mathbb{R}^{mn\times\ell}$*,* $\mathbf{V}^\top \in \mathbb{R}^{\ell\times p}$*, and* ℓ *is the rank of* $\mathbf{X}$*. The matrix decomposition of* $\mathbf{X}$ *can be represented by a BMP of a tensor triplet that equals* $\mathcal{X}$*. Then* $\mathcal{X}$ *has BM-rank at most* ℓ*.*

Proof: Define $\mathcal{A}_{:,t,:} = \texttt{reshape}(\mathbf{U}_{:,t}, [m, n])$ for all t, $1 \le t \le \ell$, and $\mathcal{C}_{:,:,k} = \mathbf{V}^\top$ for all k, $1 \le k \le n$. Let $\mathcal{B} = \texttt{ones}([m, p, \ell])$. Then $\mathcal{X} = \texttt{bmp}(\mathcal{A}, \mathcal{B}, \mathcal{C})$, and the BM-rank of $\mathcal{X}$ is at most ℓ. □

In [27], the BM-rank of a generic tensor $\mathcal{X} \in \mathbb{R}^{m\times p\times n}$ is said to be bounded above by $\min\{m, p, n\}$. Theorem 3.1.1 implies that the BM rank may be smaller. Moreover, note that we can bound the BM-rank of $\mathcal{X}$ in terms of matrix ranks of the data arranged as one of three matrices, where the three matrices come from the mode-1 unfoldings of $\mathcal{X}, \mathcal{X}^\top$, and $\mathcal{X}^{\top^2}$:

Corollary 3.1.2 *Under the conditions of Theorem 3.1.1, the BM-rank of* $\mathcal{X}$ *is bounded above by the minimum matrix rank of the mode-*1 *unfoldings of* $\mathcal{X}, \mathcal{X}^\top$*, and* $\mathcal{X}^{\top^2}$*.*

So the matrix decomposition can be viewed as a BMD with a maximum rank of ℓ. However, the converse is generally not true: given a rank-ℓ BMD, the unfolding does not have to correspond to a low-rank matrix decomposition. We will use the following example of a $2 \times 2 \times 2$ Kronecker delta tensor $\mathbf{\Delta}$ to illustrate this.

The Kronecker delta tensor $\mathbf{\Delta}$ is

$$\mathbf{\Delta}(:,:,1) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}; \quad \mathbf{\Delta}(:,:,2) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix},$$

and has BM-rank 1 such that $\mathbf{\Delta} = \texttt{bmp}(\mathcal{A}, \mathcal{B}, \mathcal{C})$, where

$$\texttt{squeeze}(\mathcal{A}(:,1,:)) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}; \mathcal{B}(:,:,1) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}; \texttt{squeeze}(\mathcal{C}(1,:,:)) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

However, the unfoldings of the tensors $\mathbf{\Delta}$, $\mathbf{\Delta}^\top$ and $\mathbf{\Delta}^{\top^2}$, which are

$$\mathbf{\Delta}_{(1)} = \mathbf{\Delta}^\top_{(1)} = \mathbf{\Delta}^{\top^2}_{(1)} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

is a matrix of rank 2.

Thus, the BMD is not just a rearrangement of a low-rank matrix decomposition of the data set. If the data can be approximated by a tensor of small BM-rank relative to the dimensions, it offers the opportunity for a fundamentally new way of compressing the data.

3.1.2 Slice-wise based decomposition

The following operates on the tensor directly by factoring frontal slices, each of which can be done independently (in parallel).

Theorem 3.1.3 *Suppose $\mathcal{X}$ has the following slice-wise matrix decomposition*

$$\mathcal{X}_{:,:,k} = \mathbf{X}^{(k)}(\mathbf{Y}^{(k)})^\top, \quad \forall\, 1 \le k \le n$$

where $\mathbf{X}^{(k)} \in \mathbb{R}^{m\times\ell}, \mathbf{Y}^{(k)} \in \mathbb{R}^{p\times\ell}$. Then the slice-wise decomposition of $\mathcal{X}$ can be represented by a tensor BMD as follows:

Let $\mathcal{A}_{:,:,k} = \mathbf{X}^{(k)}$, and $\mathcal{C}_{:,:,k} = (\mathbf{Y}^{(k)})^\top$ for all k, $1 \le k \le n$. Let $\mathcal{B} = \texttt{ones}([m, p, \ell])$. Then $\mathcal{X} = \texttt{bmp}(\mathcal{A}, \mathcal{B}, \mathcal{C})$ with BM-rank at most ℓ.

Now suppose we compute the matrix SVD of each frontal slice $\mathcal{X}_{:,:,k} = \mathbf{U}^{(k)}\mathbf{\Sigma}^{(k)}(\mathbf{V}^{(k)})^\top$. Let $\ell = \max_k(r_k)$, where r_k is the matrix rank of the k-th frontal slice of $\mathcal{X}$. Then let $\mathbf{X}^{(k)} := \mathbf{U}^{(k)}_{:,1:\ell}\mathbf{\Sigma}^{(k)}_{1:\ell,1:\ell}$, and $\mathbf{Y}^{(k)} := (\mathbf{V}^{(k)})^\top_{1:\ell,:}$. We call this the **slicewise SVD** of $\mathcal{X}$. Because we can repeat the above slicewise SVD process for $\mathcal{X}^\top$, $\mathcal{X}^{\top^2}$ to get BMP

of different tensor triplets:

Corollary 3.1.4 *The BM-rank cannot exceed the smallest of the maximum slicewise matrix rank of the $n+m+p$ matrix slices given by*

$$\mathcal{X}_{:,:,k}, \quad \forall 1 \le k \le n; \qquad \texttt{squeeze}(\mathcal{X}_{i,:,:}), \quad \forall\ 1 \le i \le m; \qquad \texttt{squeeze}(\mathcal{X}_{:,j,:}), \quad \forall\ 1 \le j \le p$$

3.2 Low BM-rank Tensor Approximation

Given a third-order tensor $\mathcal{X} \in \mathbb{R}^{m \times p \times n}$ with BM-rank r, our goal is to compute a decomposition with BM-rank ℓ, $1 \le \ell \le r$, which best approximates $\mathcal{X}$ such that

$$\min_{\tilde{\mathcal{X}}} \|\mathcal{X} - \tilde{\mathcal{X}}\|_F^2 \text{ with } \tilde{\mathcal{X}} = \sum_{t=1}^{\ell} \texttt{bmp}\left(\mathcal{A}_{:,t,:}, \mathcal{B}_{:,:,t}, \mathcal{C}_{t,:,:}\right). \tag{3.1}$$

As we will show in the next section, in fact the BMD is not unique, a point which was not addressed in [65]. Thus, our ultimate goal will be to replace this problem with a regularized one which imposes desirable features in the context of our application. However, the regularized approach involves only straightforward modification to the unregularized subproblems below, so for the sake of readability, we present the unregularized version first.

Since the BM-product is a ternary multiplication of the $m \times \ell \times n, m \times p \times \ell$ and $\ell \times p \times n$ factor tensors $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$, respectively, finding a decomposition of $\mathcal{X}$ in terms of the factor tensors could be done by treating this as a nonlinear least-squares problem. The Jacobian of the residual vector, $\texttt{vec}(\mathcal{X} - \texttt{bmp}(\mathcal{A}, \mathcal{B}, \mathcal{C}))$ will have 3ℓ non-zero entries per row, mpn rows (more, in the regularized case) and $\ell(mn+mp+np)$ columns. Computing each search direction for a nonlinear least squares solver, therefore, would require a call to an iterative method for large tensors.

Keeping in mind issues of memory and data movement, and having a preference for parallelizability, we focus instead on deriving an alternating least-squares algorithm to solve the BMD problem, which we call BMD-ALS. We show the work involved per iteration can be decoupled into mp small problems that can be solved

independently of one another. The ALS algorithm has been widely used for computing the tensor CP decomposition [12, 53] and tensor block term decomposition [18]. Despite the limitations of the algorithm such as the slow convergence, the dependency on the starting guesses, the swamp effect [72] and more [60, 99], the simplicity of understanding and implementing the ALS algorithm with superior quality of results still marks it as today's "workhorse" algorithm for CP decomposition [53]. We will show in the following subsections that BMD-ALS is also straightforward to implement, with relatively small, dense, independent subproblems that can be solved in parallel. Moreover, we will show that the slicewise SVD serves as an excellent starting guess for BMD-ALS, particularly for our video application (see Sec. 3.3), and for which we can compute the initial error easily.

3.2.1 Phase I - Starting Guess

Two ways of obtaining starting guesses are outlined in Sec. 3.1: we can either take a suitable low rank matrix approximation from the unfolded data, and refold accordingly to get $\mathcal{A}^0, \mathcal{B}^0, \mathcal{C}^0$ as in Theorem. 3.1.1; or we can take a low-rank matrix approximation (with fixed allowable rank) to each frontal slice, and refold according to Theorem. 3.1.3 to get our starting guesses. The first choice will allow us to directly improve upon the DMD method discussed in Sec. 3.3.1. The 2nd choice will allow us a direct comparison to the SS-SVD approach given in Sec. 3.3.2.

3.2.2 Alternating Least-Squares (ALS) Algorithm

Given the data tensor $\mathcal{T} \in \mathbb{R}^{m\times p\times n}$, and the initial factor tensor triplet $\mathcal{A}^0 \in \mathbb{R}^{m\times \ell\times n}$, $\mathcal{B}^0 \in \mathbb{R}^{m\times p\times \ell}$ and $\mathcal{C}^0 \in \mathbb{R}^{\ell\times p\times n}$ obtained from phase I (Sec. 3.2.1), we solve the following ALS subproblems for iterations $k = 0, 1, 2, \ldots$

$$\begin{aligned}
\mathcal{B}^{k+1} &= \min_{\mathcal{B}\in\mathbb{R}^{m\times p\times \ell}} \left\| \mathcal{T} - \texttt{bmp}\left(\mathcal{A}^k, \mathcal{B}, \mathcal{C}^k\right)\right\|_F^2, \\
\left(\mathcal{C}^\top\right)^{k+1} &= \min_{\mathcal{C}^\top\in\mathbb{R}^{p\times n\times \ell}} \left\| \mathcal{T}^\top - \texttt{bmp}\left(\left(\mathcal{B}^\top\right)^{k+1}, \mathcal{C}^\top, \left(\mathcal{A}^\top\right)^{k}\right)\right\|_F^2, \\
\left(\mathcal{A}^{\top^2}\right)^{k+1} &= \min_{\mathcal{A}^{\top^2}\in\mathbb{R}^{n\times m\times \ell}} \left\| \mathcal{T}^{\top^2} - \texttt{bmp}\left(\left(\mathcal{C}^{\top^2}\right)^{k+1}, \mathcal{A}^{\top^2}, \left(\mathcal{B}^{\top^2}\right)^{k+1}\right)\right\|_F^2.
\end{aligned} \tag{3.3}$$

In each ALS subproblem, we are holding the first and the third factor tensors fixed and solving for the middle tensor. Taking the first subproblem in the ALS algorithm as an example, we show that the problem of updating $\mathcal{B}$ reduces to a linear least-squares problem as discussed in Phase II below (Sec. 3.2.3), and updating $\mathcal{C}$ and $\mathcal{A}$ factor tensors can be solved similarly using the same algorithm.

3.2.3 Phase II - Linear Least-Squares Problem

Taking the first subproblem for updating tensor $\mathcal{B} \in \mathbb{R}^{m\times p\times \ell}$ given in the ALS subproblems (Eq. 3.3) as an example, we will show in this section that this tensor least-squares problem can be reduced to a linear least-squares problem. Furthermore, the problem can be decoupled into many (mp for updating $\mathcal{B}$, pn for updating $\mathcal{C}$, and nm for updating $\mathcal{A}$) smaller sized independent subproblems making the computation massively parallelizable.

Holding the pair $\mathcal{A}$ and $\mathcal{C}$ fixed, we optimize for $\mathcal{B}$ by solving

$$\widehat{\mathcal{B}} = \min_{\mathcal{B}\in\mathbb{R}^{m\times p\times \ell}} \|\mathcal{T} - \texttt{bmp}(\mathcal{A},\mathcal{B},\mathcal{C})\|_F^2. \tag{3.4}$$

We will show in the following derivations that the tensor least-squares problem given in Eq. (3.4) can be equivalently written as a matrix least-squares problem.

By the definitions of the Frobenius norm and tensor BM-product, we have

$$\|\mathcal{T} - \texttt{bmp}\,(\mathcal{A},\mathcal{B},\mathcal{C})\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{n}\left|\mathcal{T}_{i,j,k} - \sum_{t=1}^{\ell}\mathcal{A}_{i,t,k}\mathcal{B}_{i,j,t}\mathcal{C}_{t,j,k}\right|^2.$$

Holding the indices i and j fixed, we have

$$\sum_{(i-1)p+j=1}^{mp}\sum_{k=1}^{n}\left|\mathcal{T}_{i,j,k} - \sum_{t=1}^{\ell}\mathcal{A}_{i,t,k}\mathcal{C}_{t,j,k}\mathcal{B}_{i,j,t}\right|^2$$
$$= \sum_{(i-1)p+j=1}^{mp}\sum_{k=1}^{n}\left|\mathbf{y}_{\mathcal{T}}^{(i,j)}(k,1) - \sum_{t=1}^{\ell}\mathbf{H}_{\mathcal{A},\mathcal{C}}^{(i,j)}(k,t)\mathbf{b}_t^{(i,j)}\right|^2 = \sum_{(i-1)p+j=1}^{mp}\left\|\mathbf{y}_{\mathcal{T}}^{(i,j)} - \mathbf{H}_{\mathcal{A},\mathcal{C}}^{(i,j)}\mathbf{b}^{(i,j)}\right\|_F^2,$$

where $\mathbf{y}_{\mathcal{T}}^{(i,j)} = \texttt{squeeze}(\mathcal{T}_{i,j,:}) \in \mathbb{R}^{n\times 1}$, $\mathbf{b}^{(i,j)} = \texttt{squeeze}(\mathcal{B}_{i,j,:}) \in \mathbb{R}^{\ell\times 1}$, and $\mathbf{H}_{\mathcal{A},\mathcal{C}}^{(i,j)} \in \mathbb{R}^{n\times\ell}$ with the (k,t)-th entry specified as $\mathbf{H}_{\mathcal{A},\mathcal{C}}^{(i,j)}(k,t) = \mathcal{A}_{i,t,k}\mathcal{C}_{t,j,k}$.

Let us stack the vectors $\mathbf{y}_{\mathcal{T}}^{(i,j)}$ and $\mathbf{b}^{(i,j)}$ as

$$\mathbf{y}_{\mathcal{T}} := \begin{bmatrix} \mathbf{y}_{\mathcal{T}}^{(1,1)} \\ \vdots \\ \mathbf{y}_{\mathcal{T}}^{(i,j)} \\ \vdots \\ \mathbf{y}_{\mathcal{T}}^{(m,p)} \end{bmatrix} = \texttt{Tvec}(\mathcal{T}) \in \mathbb{R}^{mpn \times 1}, \text{ and } \mathbf{b} := \begin{bmatrix} \mathbf{b}^{(1,1)} \\ \vdots \\ \mathbf{b}^{(i,j)} \\ \vdots \\ \mathbf{b}^{(m,p)} \end{bmatrix} = \texttt{Tvec}(\mathcal{B}) \in \mathbb{R}^{mp\ell \times 1}.$$

Moreover, we take direct-sums of $\mathbf{H}_{\mathcal{A},\mathcal{C}}^{(i,j)}$ and define $\mathbf{H}_{\mathcal{A},\mathcal{C}} := \underset{i,j}{\oplus} \mathbf{H}_{\mathcal{A},\mathcal{C}}^{(i,j)} = \texttt{Mat}(\mathcal{A}, \mathcal{C})$. Then the tensor least-squares problem given in Eq. (3.4) reduces to the following matrix least-squares problem

$$\widehat{\mathbf{b}} = \min_{\mathbf{b} \in \mathbb{R}^{mp\ell \times 1}} \left\| \mathbf{y}_{\mathcal{T}} - \mathbf{H}_{\mathcal{A},\mathcal{C}} \mathbf{b} \right\|_F^2. \tag{3.5}$$

Furthermore, Eq. (6.8) decouples into mp least-squares subproblems of size $n \times \ell$ such that

$$\widehat{\mathbf{b}}^{(i,j)} = \min_{\mathbf{b}^{(i,j)} \in \mathbb{R}^{\ell \times 1}} \left\| \mathbf{y}_{\mathcal{T}}^{(i,j)} - \mathbf{H}_{\mathcal{A},\mathcal{C}}^{(i,j)} \mathbf{b}^{(i,j)} \right\|_F^2. \tag{3.6}$$

We adopt the same flattening scheme described above to obtain the following equivalent linear least-squares subproblems for updating the factor $\boldsymbol{\mathcal{A}}$:

$$\widehat{\mathbf{c}} = \min_{\mathbf{c} \in \mathbb{R}^{pn\ell \times 1}} \left\| \mathbf{y}_{\mathcal{T}^\top} - \mathbf{H}_{\mathcal{B},\mathcal{A}} \mathbf{c} \right\|_F^2, \tag{3.7}$$

where $\mathbf{c} = \texttt{Tvec}\left(\mathcal{C}^\top\right)$, $\mathbf{y}_{\mathcal{T}^\top} = \texttt{Tvec}\left(\mathcal{T}^\top\right)$, and $\mathbf{H}_{\mathcal{B},\mathcal{A}} = \texttt{Mat}\left(\mathcal{B}^\top, \mathcal{A}^\top\right)$, and the decoupled problem is given by

$$\widehat{\mathbf{c}}^{(j,k)} = \min_{\mathbf{c}^{(j,k)} \in \mathbb{R}^{\ell \times 1}} \left\| \mathbf{y}_{\mathcal{T}^\top}^{(j,k)} - \mathbf{H}_{\mathcal{B}^\top,\mathcal{A}^\top}^{(j,k)} \mathbf{c}^{(j,k)} \right\|_F^2. \tag{3.8}$$

We update $\mathcal{C}$ by solving:

$$\widehat{\mathbf{a}} = \min_{\mathbf{a} \in \mathbb{R}^{nm\ell \times 1}} \left\| \mathbf{y}_{\mathcal{T}^{\top 2}} - \mathbf{H}_{\mathcal{C},\mathcal{B}} \mathbf{a} \right\|_F^2, \tag{3.9}$$

where $\mathbf{a} = \texttt{Tvec}\left(\mathcal{A}^{\top^2}\right)$, $\mathbf{y}_{\mathcal{T}^{\top^2}} = \texttt{Tvec}\left(\mathcal{T}^{\top^2}\right)$, and $\mathbf{H}_{\mathcal{C},\mathcal{B}} = \texttt{Mat}\left(\mathcal{C}^{\top^2}, \mathcal{B}^{\top^2}\right)$, and the decoupled problem is given by

$$\widehat{\mathbf{a}}^{(k,i)} = \min_{\mathbf{a}^{(k,i)} \in \mathbb{R}^{nm\ell \times 1}} \left\| \mathbf{y}_{\mathcal{T}^{\top^2}}^{(k,i)} - \mathbf{H}_{\mathcal{C},\mathcal{B}}^{(k,i)} \mathbf{a} \right\|_F^2 . \tag{3.10}$$

The implementation of the ALS algorithm for computing tensor BMD is illustrated in Algorithm. (1). The termination criteria for the algorithm is chosen to be either reaching the maximum number of iterations N or the relative change in successive iterations becomes sufficiently small, i.e. $\|\mathcal{T}^{n+1} - \mathcal{T}^n\|_F / \|\mathcal{T}^n\|_F < \epsilon$ for some tolerance parameter $\epsilon > 0$. A complete iterate requires to compute all three-factor tensors $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$.

Algorithm 1 Tensor BMD-ALS

procedure $[\mathcal{A}^N, \mathcal{B}^N, \mathcal{C}^N]$=BMD-ALS$(\mathcal{T}, \mathcal{A}^0, \mathcal{B}^0, \mathcal{C}^0, N, \epsilon)$
  for $n = 0, 1, 2, \ldots, N$ **do**
    Update the (i,j)-th tube fiber $\mathcal{B}_{i,j,:}^{n+1}$ by solving Eq. 3.6.
    Update the (j,k)-th tube fiber $(\mathcal{C}^{\top})_{j,k,:}^{n+1}$ by solving Eq. 3.8.
    Update the (k,i)-th tube fiber $(\mathcal{A}^{\top^2})_{k,i,:}^{n+1}$ by solving Eq. (3.10).
    $\mathcal{T}^{n+1} = \texttt{bmp}(\mathcal{A}^{n+1}, \mathcal{B}^{n+1}, \mathcal{C}^{n+1})$
  end for
  if $\|\mathcal{T}^{n+1} - \mathcal{T}^n\|_F / \|\mathcal{T}^n\|_F < \epsilon$ **then**
    $n \leftarrow N$
  end if
end procedure

3.2.3.1 Computational Complexity

The cost of implementing SVD for a matrix of size $m \times n$ with $m \geq n$ is $\mathcal{O}(mn^2 + n^2)$ flops. Moreover, the cost of solving a least-squares problem with the matrix of size $m \times n$, $m \geq n$, has the same time complexity as computing the SVD of the matrix [29]. In phase I, the first choice of obtaining a starting guess is to compute a DMD of the vectorized video matrix. The total cost for DMD is dominated by taking SVD of the data matrix of size $mn \times (p-1)$ where $mn \geq (p-1)$ and solving a least-squares problem with the coefficient matrix of size $mn \times \ell$. Hence, the total cost

is $\mathcal{O}\left((mn+1)((p-1)^2+\ell^2)\right)$ flops for the DMD choice. For the second choice, the total cost is $\mathcal{O}\left(n(m+1)p^2\right)$ flops for computing the n frontal slice matrix SVDs, assuming $m \geq p$. In phase II, the total cost of solving the mp smaller least-squares problem is $\mathcal{O}\left(mp(n+1)\ell^2\right)$ flops. The least-squares solutions are obtained using e.g., MATLAB's `lsqminnorm` function for the minimum norm solution.

We also emphasize that both phase I (with the slice-wise SVD choice) and phase II computations can be done in parallel since we can compute matrix SVDs simultaneously on the individual frontal slices of the input tensor, and the second phase of the algorithm is also parallelizable since the mp smaller least-squares problems can be solved independently and concurrently. So parallel computation methods can potentially improve the execution time significantly, though further discussion is beyond the scope of this study and hence will not be discussed further in the current work.

3.2.4 ALS Convergence Analysis

The alternating least-squares algorithm has been widely used for computing the tensor CP decomposition [53] and the block term decomposition [18, 72] among others. The local and global convergence of the ALS algorithm for the tensor CP decomposition has been studied in [60, 99, 105] based on its connection to the block nonlinear Gauss–Seidel (GS) method. In this section, we will show that the ALS algorithm for computing low BM-rank tensor approximation is also closely connected to the nonlinear GS method, and hence several convergence results follow directly from its framework.

Recall the nonlinear GS method solves the following minimization problem

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &\text{subject to } \mathbf{x} \in X = X_1 \times X_2 \times \cdots \times X_M \subset \mathbb{R}^{N\times 1}, \end{aligned}$$

where f is a continuously differentiable function from $\mathbb{R}^{N\times 1}$ to $\mathbb{R}$ and X is a Cartesian product of closed, nonempty and convex subsets $X_i \subset \mathbb{R}^{N_i\times 1}$, for $i = 1, \ldots, M$ with

$\sum_{i=1}^{M} N_i = N$. If the vector $\mathbf{x} \in \mathbb{R}^N$ is partitioned into M component vectors $\mathbf{x}_i \in \mathbb{R}^{N_i \times 1}$, then we can consider f a function from $\mathbb{R}^{N_1 \times 1} \times \mathbb{R}^{N_2 \times 1} \times \cdots \times \mathbb{R}^{N_M \times 1}$ to $\mathbb{R}$ with $f(\mathbf{x}) = f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M)$. The solution to the nonlinear optimization function can then be found by the block Gauss-Seidel method via the following iteration in a cyclic order,

$$\mathbf{x}_i^{k+1} = \min_{\mathbf{y}_i \in X_i} f\left(\mathbf{x}_1^{k+1}, \ldots, \mathbf{x}_{i-1}^{k+1}, \mathbf{y}_i, \mathbf{x}_{i+1}^{k}, \ldots, \mathbf{x}_M^{k}\right),$$

which updates the components of $\mathbf{x}$. The iterative technique starts from a given initial guess $\mathbf{x}^0 = \left(\mathbf{x}_1^0, \mathbf{x}_2^0, \ldots, \mathbf{x}_M^0\right)$ and generates a sequence $\{\mathbf{x}^k\} = \left\{\left(\mathbf{x}_1^k, \mathbf{x}_2^k, \ldots, \mathbf{x}_M^k\right)\right\}$.

The connection between the nonlinear block Gauss-Seidel method and the ALS-BMD algorithm is made evident by noting the cost function we want to minimize,

$$\left\|\mathcal{X} - \hat{\mathcal{X}}\right\|_F^2 = \sum_{i,j,k} \left(\mathcal{X}_{i,j,k} - \sum_{t=1}^{\ell} \mathcal{A}_{i,t,k} \mathcal{B}_{i,j,t} \mathcal{C}_{t,j,k}\right)^2 = f(\mathcal{A}, \mathcal{B}, \mathcal{C}),$$

is a function $f : \mathbb{R}^{(mn+mp+pn)\ell \times 1} \to \mathbb{R}$. By letting $\mathbf{v} = [\mathbf{a}; \mathbf{b}; \mathbf{c}] \in \mathbb{R}^{(mn+mp+pn)\ell \times 1}$ where $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ are the vectorized factor tensors, then we can see that

$$f(\mathbf{v}) = f(\mathcal{A}, \mathcal{B}, \mathcal{C}).$$

The BMD problem therefore can be reformulated into the following problem

$$\begin{aligned} &\min f(\mathbf{v}) \\ &\text{subject to } \mathbf{v} \in \mathbb{R}^{mn\ell \times 1} \times \mathbb{R}^{mp\ell \times 1} \times \mathbb{R}^{pn\ell \times 1}. \end{aligned}$$

The ALS algorithm updates the components of $\mathbf{v}$ by

$$\begin{aligned} \mathbf{b}^{k+1} &= \min_{\mathbf{y} \in \mathbb{R}^{mp\ell \times 1}} f\left(\mathbf{a}^k, \mathbf{y}, \mathbf{c}^k\right), \\ \mathbf{c}^{k+1} &= \min_{\mathbf{y} \in \mathbb{R}^{pn\ell \times 1}} f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{y}\right), \\ \mathbf{a}^{k+1} &= \min_{\mathbf{y} \in \mathbb{R}^{mn\ell \times 1}} f\left(\mathbf{y}, \mathbf{b}^{k+1}, \mathbf{c}^{k+1}\right). \end{aligned}$$

This is exactly the nonlinear block Gauss-Seidel method.

By Proposition 2.1 in [105], let $\mathbf{v}^k$ denote the k-th solution vector, i.e. $\mathbf{v}^k = \left(\mathbf{a}^k, \mathbf{b}^k, \mathbf{c}^k\right)$. When the normal equations matrix in each of the linear least-squares subproblems given in Eq. (6.3), Eq. (6.5), and Eq. (6.6) in the main text is positive definite, i.e. $\mathbf{H}^\top\mathbf{H} \succ 0$ for any $\mathbf{H} \in \{\mathbf{H}_{\mathcal{AC}}, \mathbf{H}_{\mathcal{BA}}, \mathbf{H}_{\mathcal{CB}}\}$, then each least-squares subproblem has a unique solution and the whole sequence $\{\mathbf{v}^k\}$ generated by ALS converges to a limit point. In particular,

$$f\left(\mathbf{v}^{k+1}\right) \leq f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^{k+1}\right) \leq f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^k\right) \leq f\left(\mathbf{v}^k\right) \leq \cdots \leq f\left(\mathbf{v}^0\right),$$

shows that ALS monotonically reduces the cost function. Since f is bounded below, $\{f(\mathbf{v}^k)\}$ has a limit point $f^\star \geq 0$. Moreover, by Theorem 3.1 in [105], if $\{\mathbf{v}^k\}$ is bounded, then the limit point of the sequence is a stationary point of the problem.

3.3 Background on Video Surveillance Separation and Modeling

The task of video background and foreground separation is an important computer vision application [10, 23, 59, 95]. Background subtraction is often used in this task in order to detect moving foreground objects. Hence, accurate modeling of the video background under complex, diverse, and cluttered conditions is of paramount importance for most of the background/foreground separation methods. One of the more promising research directions in the field focuses on utilizing matrix decomposition methods. By vectorizing video frames into vectors and stacking them column-wise into a matrix, decomposition methods such as the robust principal component analysis (RPCA) [10], and the dynamic mode decomposition (DMD) [55] method can be used to separate the video data matrix into a stationary background and moving foreground components which are usually assumed to be low-rank and sparse respectively.

However, flattening three-dimensional video data into a matrix presents disadvantages. For instance, vectorization destroys the intrinsic spatial structure within

frames. Moreover, complex disturbances within the background can be difficult to identify and remove after flattening, which would potentially lead to a lower background reconstruction quality [61]. To overcome these difficulties for matrix-based methods, an increasing number of tensor decomposition methods have been proposed dealing with background subtraction [41, 44, 61, 88]. Similar to matrix-based methods, the tensor-based techniques model the background video with a low tensor rank component and obtain the sparse foreground by subtracting the background from the video data. Although for most of the aforementioned methods the low-rank property of the background suggests that it is also possible to compress the stationary background for more efficient data storage, the step of background subtraction does not guarantee a compressed representation of the foreground video.

3.3.1 Dynamic Mode Decomposition (DMD)

Though initially introduced in the fluid dynamics community for extracting spatiotemporal patterns [55], DMD has also been effective in the video context for separating the stationary background from foreground motions by differentiating between the DMD modes with near-zero frequency and remaining modes with frequencies bounded away from the origin [30]. Better visual results and superior computational efficiency compared to the Robust-PCA algorithm of the DMD method have drawn great attention in the computer vision community with new algorithms developed based on DMD for improved accuracy and efficiency. A few examples include the compressed DMD [19], randomized DMD [20], DMD via dictionary learning [34], and multi-resolution DMD for object tracking with varying motion rates [56].

More recently, the connection between the general DMD method and the CP decomposition has been studied [82]. When multiple experiments were conducted, data matrices generated from the experiments were collected and ordered as frontal slices of a third-order tensor. Then performing the DMD on each frontal slice of the data tensor decomposes it into a sum of vector outer products of the DMD vector triplets consisting of the DMD modes, DMD eigenfunctions, and the corresponding eigenvalues. It is easily shown that this decomposition is equivalent to taking a CP

decomposition of the third-order tensor when the experimental data are collected from distinct sources with a single exponential growth or decay and/or oscillatory dynamics.

More specifically, for the video background/foreground separation application [55], the DMD stationary background video sequence is reconstructed by taking

$$\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} = b_p \boldsymbol{\varphi}_p e^{\omega_p \mathbf{t}} \in \mathbb{R}^{mn \times 1}, \tag{3.11}$$

where ω_p, $p \in \{1, 2, ..., \ell\}$, is the Fourier mode satisfying $\|\omega_p\| \approx 0$. The vector $\boldsymbol{\varphi}_p$ is the associated p-th DMD mode, and b_p is the initial amplitude of the corresponding mode. The vector $\mathbf{t} = [t_1, t_2, \ldots, t_p]$ contains the times at which the frames were collected.

In the present work, we can view the DMD results obtained from decomposing a single data matrix as a tensor BM-decomposition by Theorem 3.1.1. This is particularly meaningful to our video application since the DMD method for video background/foreground separation developed originally is applied to a single data matrix flattened from a third-order video tensor [30]. Detailed derivations on rewriting the DMD results into order-3 tensor BMP were provided in the Supplementary material in [93], where we will also show that due to the nonlinearity of the foreground motion in general, the BMD factor tensor triplet does not directly convert back to DMD modes and eigenvalues.

3.3.2 Spatiotemporal Slice-based SVD (SS-SVD)

In the recent study by Kajo et al. [41], the slice-wise SVD discussed previously was applied to the spatiotemporal slices of an input video tensor to extract the background information. The authors refer to this method as spatiotemporal slice-based SVD (SS-SVD). Since it has a connection to our initialization step in the BMD-ALS algorithm, we will briefly describe their method here.

Given a video of p frames with size $m \times n$, we order the frames as lateral slices to form a third-order tensor $\boldsymbol{\mathcal{X}}$ of size $m \times p \times n$. Each frontal slice, $\boldsymbol{\mathcal{X}}_{:,:,k}, 1 \leq k \leq n$, or

horizontal slice $\mathcal{X}_{i,:,:}, 1 \le i \le m$ is called a spatiotemporal slice containing both space and time information. Here we construct the SS-SVD using frontal spatiotemporal slices. Specifically, the SS-SVD method applies a low-rank approximation to each frontal spatiotemporal slice using truncated SVDs with a target matrix rank ℓ:

$$\mathcal{X}_{:,:,k} \approx \sum_{t=1}^{\ell} \mathbf{u}_t^{(k)} \sigma_t^{(k)} \left(\mathbf{v}_t^{(k)}\right)^\top, 1 \le k \le n. \tag{3.12}$$

When $t = 1$, the rank-1 matrix reconstruction corresponds to the largest singular value of each slice which the authors of [41] argue captures mainly the dominant background scene across slices*. We denote this set of tensor slices as $\hat{\mathcal{X}}^{\text{bg}}_{:,:,k}$ for all $k = 1, \ldots, n$ and is given by

$$\hat{\mathcal{X}}^{\text{bg}}_{:,:,k} = \mathbf{u}_1^{(k)} \sigma_1^{(k)} \left(\mathbf{v}_1^{(k)}\right)^\top. \tag{3.13}$$

The foreground in [41] is taken to be the difference between the given video data and the reconstructed background, i.e. $\hat{\mathcal{X}}^{\text{fg}} = \mathcal{X} - \hat{\mathcal{X}}^{\text{bg}}$, *which is not guaranteed to be a compressed representation.*

However, the approximation (3.12), which we will call the **Slicewise SVD**, by our results in Section. 3.1, *can be interpreted as a BMD of BM-rank at most* ℓ.

Theorem 3.3.1 *The tensor* $\hat{\mathcal{X}}$ *defined with*

$$\hat{\mathcal{X}}_{:,:,k} = \mathbf{W}^{(k)} (\mathbf{V}^{(k)})^T,$$

where $\mathbf{u}_t^{(k)} \sigma_t^{(k)}$ *from (3.12) are the* ℓ *columns of* $\mathbf{W}^{(k)}$ *and* $\mathbf{v}_t^{(k)}$ *are the* ℓ *columns of* $\mathbf{V}^{(k)}$ *has BM-rank at most* ℓ *and the error in the approximation is*

$$\|\mathcal{X} - \hat{\mathcal{X}}\|_F^2 = \sum_{k=1}^{n} \sum_{i=\ell+1}^{r} (\sigma_i^{(k)})^2.$$

In the next section, using our generative model for insight, we will explain the utility of this result.

*We note that if the data is non-negative, the first rank-1 triples will be non-negative by the Perrone-Frobenius theorem.

3.4 Generative Model for Surveillance Video

In this section, we consider a BM-product based model of one or two moving objects on a static background. Since in our model the stationary background image and the moving foreground object can be incorporated into separate BM-rank 1 components, a low BM-rank tensor decomposition separates the video data into compressed representations of both the background and the foreground simultaneously.

3.4.1 Generative Spatiotemporal Model With Low BM-rank

Suppose we have a static background image $\mathbf{X} \in \mathbb{R}^{m\times n}$, and an object (of constant intensity α and constant rectangular size $r_1 \times r_2$ for the time being) moving across the background over p time steps. At time k, $1 \le k \le p$, the object is located at $\mathcal{I}_k \times \mathcal{J}_k := [i_k, i_k + r_1] \times [j_k, j_k + r_2]$ with $1 \le i_k \le m - r_1$ and $1 \le j_k \le n - r_2$. Consider a binary image of the same size as the background image denoted $\mathbf{M}_{\mathcal{I}_k,\mathcal{J}_k}$, where

$$\mathbf{M}_{\mathcal{I}_k,\mathcal{J}_k}[i,j] = \begin{cases} \alpha, \text{ if } (i,j) \in \mathcal{I}_k \times \mathcal{J}_k \\ 0, \text{ otherwise} \end{cases} . \tag{3.14}$$

Next, define a binary vector $\mathbf{b}^{(k)} \in \mathbb{R}^m$ such that $\mathbf{b}_i^{(k)} = 1$ when $i \in \mathcal{I}_k$ and is 0 otherwise. Similarly, define $\mathbf{c}^{(k)} \in \mathbb{R}^n$ such that $\mathbf{c}_j^{(k)} = 1$ when $j \in \mathcal{J}_k$ and is 0 otherwise. Finally, let $\mathbf{1}_{m\times n} \in \mathbb{R}^{m\times n}$ be a matrix of all-ones. Then the rectangle image at time k can be expressed as

$$\mathbf{M}_{\mathcal{I}_k,\mathcal{J}_k} = \alpha \, \mathrm{diag}(\mathbf{b}^{(k)}) \cdot \mathbf{1}_{m\times n} \cdot \mathrm{diag}(\mathbf{c}^{(k)}), \tag{3.15}$$

where $\mathrm{diag}(\mathbf{v})$ means the square, diagonal matrix with the diagonal entries provided by the vector argument $\mathbf{v}$. Hence, the k-th video frame that captures both the background image and the moving object, denoted as $\mathbf{T}^{(k)}$, can be expressed as

$$\begin{aligned} \mathbf{T}^{(k)} &:= \mathbf{X} - \mathrm{diag}(\mathbf{b}^{(k)}) \cdot \mathbf{X} \cdot \mathrm{diag}(\mathbf{c}^{(k)}) + \mathbf{M}_{\mathcal{I}_k,\mathcal{J}_k} \\ &= \mathrm{diag}(\mathbf{1}_m)\mathbf{X}\,\mathrm{diag}(\mathbf{1}_n) + \mathrm{diag}(\mathbf{b}^{(k)})\,(-\mathbf{X} + \alpha\mathbf{1}_{m\times n})\,\mathrm{diag}(\mathbf{c}^{(k)}). \end{aligned} \tag{3.17}$$

The term $-\operatorname{diag}(\mathbf{b}^{(k)})\mathbf{X}\operatorname{diag}(\mathbf{c}^{(k)})$ "zeros out" the entries in the stationary image where the object is living in this frame, and the term $\mathbf{M}_{\mathcal{I}_k,\mathcal{J}_k}$ puts the constant value rectangle over those pixels. Both are necessary to ensure that the object retains its constant value.

Now we show this simple video can be modeled using a low-rank BMD. Specifically,

1. Define the $m \times 2 \times n$ tensor $\boldsymbol{\mathcal{A}}$ with $\boldsymbol{\mathcal{A}}_{:,1,:} = \mathbf{X}$ and $\boldsymbol{\mathcal{A}}_{:,2,:} = (-\mathbf{X} + \alpha \mathbf{1}_{m\times n})$,
2. Define the $m \times p \times 2$ tensor $\boldsymbol{\mathcal{B}}$ with $\boldsymbol{\mathcal{B}}_{:,:,1} = \mathbf{1}_{m\times p}$ and $\boldsymbol{\mathcal{B}}_{:,k,2} = \mathbf{b}^{(k)}$,
3. Define the $2 \times p \times n$ tensor $\boldsymbol{\mathcal{C}}$ with $\boldsymbol{\mathcal{C}}_{1,:,:} = \mathbf{1}_{p\times n}$ and $\boldsymbol{\mathcal{C}}_{2,k,:} = \mathbf{c}^{(k)}$.

Then the video tensor $\boldsymbol{\mathcal{X}} = \texttt{bmp}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}})$ is a sum of two BM-rank 1 tensors: the stationary background tensor and the foreground moving object tensor, i.e.

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}^{\text{bg}} + \boldsymbol{\mathcal{X}}^{\text{fg}} := \texttt{bmp}(\boldsymbol{\mathcal{A}}_{:,1,:}, \boldsymbol{\mathcal{B}}_{:,:,1}, \boldsymbol{\mathcal{C}}_{1,:,:}) + \texttt{bmp}(\boldsymbol{\mathcal{A}}_{:,2,:}, \boldsymbol{\mathcal{B}}_{:,:,2}, \boldsymbol{\mathcal{C}}_{2,:,:}). \tag{3.18}$$

The k-th lateral slice of tensor $\boldsymbol{\mathcal{X}}$ is given exactly by Eq. (3.17), i.e. $\texttt{squeeze}(\boldsymbol{\mathcal{X}}_{:,k,:}) = \mathbf{T}^{(k)}$.

We can augment this generative spatiotemporal video model in straightforward ways. For example, if there is a second object that is moving either in the same horizontal or same vertical direction as the first, we can augment the columns of $\boldsymbol{\mathcal{B}}_{:,2,:}$ (vertical) or $\boldsymbol{\mathcal{C}}_{2,:,:}$ (horizontal) to capture the other object's motion without increasing the BM rank (see Figure 3.1). If the two objects are moving with different trajectories, and possibly with different intensities, we can model the video with at most three BM-rank 1 terms (see Figure 3.2).

The point is that our generative model illustrates why we might reasonably expect to be able to capture the foreground and the background with the compressive power of a low BM-rank approximation. Moreover, in one of the numerical experiments, we will use this generative model to illustrate superiority of the our low BM rank model vs. a state-of-the-art DMD-based approach.

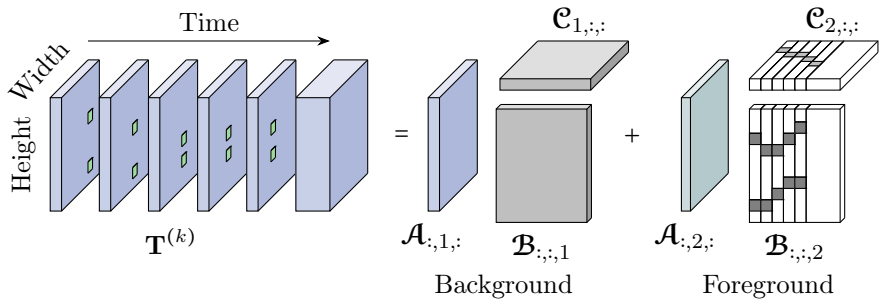


Figure 3.1: Illustration of the generative spatiotemporal video model with two constant valued, same intensity, rectangular objects moving in the same horizontal direction, on a constant background.

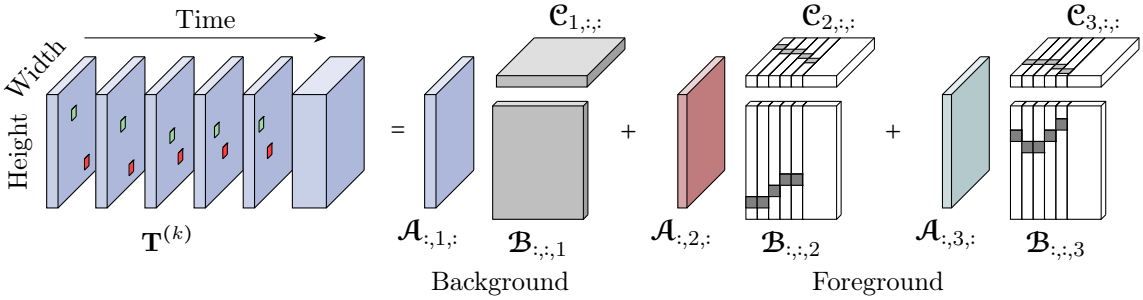


Figure 3.2: Illustration of the generative spatiotemporal video model with two constant valued, different intensity, rectangular objects moving in different directions, on a constant background. Three BM-rank 1 terms are sufficient to capture the foreground objects' motion and the background

3.4.2 SS-SVD - BMD Connection for Our Generative Model

We are now in a position, using the generative model, to explain why the frontal slices of the surveillance video will have low (matrix) rank and connect our BMD-based model to the SS-SVD. Consider our generative model in which only a single pixel-size object is traversing from left to right in the i-th row of the background image $\mathbf{X}$. This video tensor $\mathcal{X}$ will have a BM-rank of two. Each frame is a *lateral* slice $\mathcal{X}_{:,j,:}$. Then it is easy to see that the rank of each *frontal* slice $\mathcal{X}_{:,:,k}$ is either 1 or 2, because $\mathcal{X}_{:,j,k}$ is either $\mathbf{X}_{:,k}$ (if no object is present at time j on the k-th column of the background image) or $\mathbf{X}_{:,k} + (-\mathbf{X}_{i,k} + \alpha)\mathbf{e}_i$ (the single pixel object is present), where $\mathbf{e}_i$ is the i-th standard basis vector. Thus, indeed, a constant multiple of the left singular vector $\mathbf{u}_1^{(k)}$ will accurately approximate the background column pixel $\mathbf{X}_{:,k}$. However, by orthogonality and the fact that $c_k\mathbf{u}_1^k \approx \mathbf{X}_{:,k}$ for some constant

$c_k \neq 0$, $\mathbf{u}_2^{(k)}$ is approximately some multiple of $(-\mathbf{X}_{i,k} + \alpha)\mathbf{e}_i - \kappa_k \mathbf{X}_{:,k}$ where κ_k is also a non-zero constant. In sum, for $t = 1$, we approximate $\mathcal{X}^{\text{bg}}$ in (3.18) and the $t = 2$ term approximates $\mathcal{X}^{\text{fg}}$ in (3.18). So both the foreground and the background have compressed representations in the BMD form. We will utilize this observation in our choice of starting guess for our algorithm.

3.5 Regularized ALS

Grouping the slice-wise singular values with the corresponding left singular vectors as in Theorem 3.3.1 was arbitrarily made in order to preserve the scaling constants of the left-singular vectors. This illustrates a non-uniqueness property of the BMD. The following lemma discusses the general case.

Lemma 3.5.1 *Let* $\mathbf{A}$ *be* $m \times n$, $\mathbf{B}$ *be* $m \times p$, *and* $\mathbf{C}$ *be* $p \times n$. *Then define two tensor triplets* $(\mathcal{A}, \mathcal{B}, \mathcal{C}), (\tilde{\mathcal{A}}, \tilde{\mathcal{B}}, \tilde{\mathcal{C}}) \in \mathbb{R}^{m\times 1\times n} \times \mathbb{R}^{m\times p\times 1} \times \mathbb{R}^{1\times p\times n}$ *such that*

$$\mathcal{A}_{:,1,:} = \mathbf{A}, \qquad \mathcal{B}_{:,:,1} = \mathbf{B}, \qquad \mathcal{C}_{1,:,:} = \mathbf{C},$$
$$\tilde{\mathcal{A}}_{:,1,:} = \mathbf{D}_1\mathbf{A}, \qquad \tilde{\mathbf{B}}_{:,:,1} = \mathbf{D}_1^{-1}\mathbf{B}\mathbf{D}_2^{-1}, \qquad \tilde{\mathcal{C}}_{1,:,:} = \mathbf{D}_2\mathbf{C},$$

where $\mathbf{D}_1$ *and* $\mathbf{D}_2$ *are invertible* $m \times m$ *and* $p \times p$ *diagonal matrices respectively. Then*

$$\texttt{bmp}(\mathcal{A}, \mathcal{B}, \mathcal{C}) = \texttt{bmp}(\tilde{\mathcal{A}}, \tilde{\mathcal{B}}, \tilde{\mathcal{C}}),$$

showing that the BMD factors are non-unique.

To address the non-uniqueness, we might wish to add regularization. Recall our generative model suggests that one of the lateral slices of $\mathcal{A}$ should give information about the background scene, and the subsequent SS-SVD analysis suggests that it should correspond to the largest singular value. The $\mathcal{B}$ and $\mathcal{C}$ tensors each have information about one spatial dimension and time. So, the simplest regularization model would be

$$\min_{\mathcal{A},\mathcal{B},\mathcal{C}} \|\mathcal{T} - \texttt{bmp}\,(\mathcal{A}, \mathcal{B}, \mathcal{C})\|_F^2 + \frac{1}{2}\left(\|\mathbf{L}_a(\mathcal{A})\|_F^2 + \|\mathbf{L}_b(\mathcal{B})\|_F^2 + \|\mathbf{L}_c(\mathcal{C})\|_F^2\right) \tag{3.19}$$

where $\mathbf{L}_a, \mathbf{L}_b$, and $\mathbf{L}_c$ are linear operators acting on the vectorized factor tensors $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}$, and $\boldsymbol{\mathcal{C}}$ respectively. More specifically, for all $1 \le i \le m, 1 \le j \le p, 1 \le k \le n$, $\mathbf{L}_a = \underset{k,i}{\oplus}\mathbf{L}$ where $\mathbf{L} = \text{diag}([\lambda_1, \lambda_2, \ldots, \lambda_\ell])$, $\mathbf{L}_b = \underset{i,j}{\oplus}\beta\mathbf{I}_{\ell\times\ell}$ and $\mathbf{L}_c = \underset{j,k}{\oplus}\gamma\mathbf{I}_{\ell\times\ell}$, where λ_t for $1 \le t \le \ell$, β, and γ are positive constants. Here, we set λ_1 to be at least an order of magnitude smaller than λ_2. This is consistent with the fact that the dominant feature is the background (requiring the least penalty).

Importantly, this choice keeps the subproblems decoupled: the only change required is to add a constraint term to each of Eq. (6.8), Eq. (3.7), and Eq. (7.1) respectively, in the form of either $\kappa\|\mathbf{v}\|_2^2$ for $\mathbf{v} = \mathbf{b}^{(i,j)}$ or $\mathbf{v} = \mathbf{c}^{(i,j)}$ (Eq. 3.6, Eq. 3.8 respectively) or $0.5\|\mathbf{L}\mathbf{a}^{(k,i)}\|_2$ (to Eq. 3.10).

Let $\Psi(\boldsymbol{\omega})$ denote the objective function given in Eq. (3.19) where $\boldsymbol{\omega}$ is the solution vector, i.e. $\boldsymbol{\omega} = (\mathbf{a}, \mathbf{b}, \mathbf{c})$. Then the sequence $\{\boldsymbol{\omega}^k\}_{k\in\mathbb{N}}$ generated by the regularized alternating least-squares algorithm converges to a critical point of Ψ. We provide the proof in the next subsection.

3.5.1 Regularized ALS Convergence Analysis

Convergence analysis for the ALS algorithm with Tikhonov regularization has been studied for the CP decomposition [43]. We note that a similar analysis can be derived for the regularized ALS algorithm for BM-decomposition.

Given a third order data tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{m\times p\times n}$, our goal is to find a tensor BM-decomposition $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{m\times \ell\times n}$, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{m\times p\times \ell}$, and $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{\ell\times p\times n}$ from the constrained problem

$$\min_{\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{C}}} \|\boldsymbol{\mathcal{X}} - \text{bmp}\,(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}})\|_F^2 + \frac{1}{2}\left(\|\mathbf{L}_a\,(\boldsymbol{\mathcal{A}})\|_F^2 + \|\mathbf{L}_b\,(\boldsymbol{\mathcal{B}})\|_F^2 + \|\mathbf{L}_c\,(\boldsymbol{\mathcal{C}})\|_F^2\right), \tag{3.20}$$

where $\mathbf{L}_a$, $\mathbf{L}_b$, and $\mathbf{L}_c$ are linear operators acting on the vectorized factor tensors $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, and $\boldsymbol{\mathcal{C}}$ respectively. More specifically, for all $1 \le i \le m, 1 \le j \le p, 1 \le k \le n$, $\mathbf{L}_a = \underset{k,i}{\oplus}\mathbf{L}$ where $\mathbf{L} = \text{diag}([\lambda_1, \lambda_2, \ldots, \lambda_\ell])$, $\mathbf{L}_b = \underset{i,j}{\oplus}\beta\mathbf{I}_{\ell\times\ell}$ and $\mathbf{L}_c = \underset{j,k}{\oplus}\gamma\mathbf{I}_{\ell\times\ell}$.

Then the regularized ALS subproblems are given by

$$\hat{\mathcal{B}}[i,j,:] = \min_{\mathbf{b}^{(i,j)}\in\mathbb{R}^{\ell\times 1}} \left\|\mathbf{y}_{\mathcal{T}}^{(i,j)} - \mathbf{H}_{\mathcal{AC}}^{(i,j)}\mathbf{b}^{(i,j)}\right\|_F^2 + \frac{1}{2}\left\|\mathbf{L}_b^{(i,j)}\mathbf{b}^{(i,j)}\right\|_F^2,$$

$$\hat{\mathcal{C}}^{\top}[j,k,:] = \min_{\mathbf{c}^{(i,j)}\in\mathbb{R}^{\ell\times 1}} \left\|\mathbf{y}_{\mathcal{T}^{\top}}^{(j,k)} - \mathbf{H}_{\mathcal{BA}}^{(j,k)}\mathbf{c}^{(j,k)}\right\|_F^2 + \frac{1}{2}\left\|\mathbf{L}_c^{(j,k)}\mathbf{c}^{(j,k)}\right\|_F^2,$$

$$\hat{\mathcal{A}}^{\top^2}[k,i,:] = \min_{\mathbf{a}^{(i,j)}\in\mathbb{R}^{\ell\times 1}} \left\|\mathbf{y}_{\mathcal{T}^{\top^2}}^{(k,i)} - \mathbf{H}_{\mathcal{CB}}^{(k,i)}\mathbf{a}^{(i,j)}\right\|_F^2 + \frac{1}{2}\left\|\mathbf{L}_a^{(k,i)}\mathbf{a}^{(k,i)}\right\|_F^2,$$

for all $1 \le i \le m$, $1 \le j \le p$, and $1 \le k \le n$. Let Ψ represents the objective function in Eq. 3.20, then $\Psi : \mathbb{R}^{\ell(mn+mp+np)} \to \mathbb{R}^{+}$, where

$$\Psi(\mathcal{A},\mathcal{B},\mathcal{C}) = f(\mathcal{A},\mathcal{B},\mathcal{C}) + \frac{1}{2}\left(\|\mathbf{L}_a(\mathcal{A})\|_F^2 + \|\mathbf{L}_b(\mathcal{B})\|_F^2 + \|\mathbf{L}_c(\mathcal{C})\|_F^2\right) \tag{3.21}$$

We will first show that the objective functions in the subproblems are μ-strongly convex.

Definition 3.5.2 *A differentiable function $h : dom(h) \to \mathbb{R}$, where $dom(h) \subseteq \mathbb{R}^n$ is called μ-strongly convex if there exists a constant $\mu > 0$ such that*

$$h(y) \ge h(x) + \nabla h(x)^{\top}(y-x) + \frac{\mu}{2}\|y-x\|_2^2, \forall\, x, y \in dom(h).$$

Theorem 3.5.3 *The objective function*

$$f\left(\mathbf{a},\mathbf{b}^k,\mathbf{c}^k\right) + \frac{1}{2}\|\mathbf{L}_a\mathbf{a}\|_F^2$$

is μ-strongly convex.

Proof: The least-squares problem $f\left(\mathbf{a},\mathbf{b}^k,\mathbf{c}^k\right) = \left\|\mathbf{y}_{\mathcal{T}^{\top^2}} - \mathbf{H}_{\mathcal{CB}}\mathbf{a}\right\|_F^2$ is convex. The regularization term $\frac{1}{2}\|\mathbf{L}_a\mathbf{a}\|_F^2$ has the following first derivative

$$\frac{\partial}{\partial\mathbf{a}}\left\{\frac{1}{2}\|\mathbf{L}_a\mathbf{a}\|_F^2\right\} = \mathbf{L}_a^{\top}\mathbf{L}_a\mathbf{a}$$

and second derivative

$$\frac{\partial^2}{\partial \mathbf{a} \partial \mathbf{a}^\top} \left\{ \frac{1}{2} \left\| \mathbf{L}_a \mathbf{a} \right\|_F^2 \right\} = \frac{\partial}{\partial \mathbf{a}} \left\{ \mathbf{L}_a^\top \mathbf{L}_a \mathbf{a} \right\} = \mathbf{L}_a^\top \mathbf{L}_a,$$

which is positive-semidefinite, since $\mathbf{L}_a$ is a direct sum of mn diagonal matrices with non-negative entries. So there exists a constant $\mu > 0$ such that $\frac{\partial^2}{\partial \mathbf{a}^2} \left\{ \frac{1}{2} \left\| \mathbf{L}_a \mathbf{a} \right\|_F^2 \right\} \succcurlyeq \mu \mathbf{I}$.

A linear combination of a convex and a μ-strongly convex function is also μ-strongly convex. Hence, the objective function of the subproblem with respect to $\boldsymbol{\mathcal{A}}$ is μ-strongly convex. □

Similarly, the objective function of the subproblems with respect to $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$ are also μ-strongly convex.

Lemma 3.5.4 *Suppose* $\mathbf{a}^{k+1}$ *is obtained by solving the subproblem have decrease in the objective function after a single update of* $\mathbf{a}$*, i.e.*

$$\Psi\left(\mathbf{a}^k\right) - \Psi\left(\mathbf{a}^{k+1}\right) \geq \frac{\mu}{2} \left\| \mathbf{a}^k - \mathbf{a}^{k+1} \right\|_2^2$$

for some constant $\mu > 0$*.*

Proof: By the first order optimality, we have

$$\nabla_a \Psi\left(\mathbf{a}^{k+1}\right) = \nabla_a f\left(\mathbf{a}^{k+1}\right) + \mathbf{L}_a^\top \mathbf{L}_a \mathbf{a} = 0.$$

Then the strongly convexity of the objective function yields

$$\Psi\left(\mathbf{a}^k\right) - \Psi\left(\mathbf{a}^{k+1}\right) \geq \nabla_a \Psi\left(\mathbf{a}^{k+1}\right)^\top \left(\mathbf{a}^k - \mathbf{a}^{k+1}\right) + \frac{\mu}{2} \left\| \mathbf{a}^k - \mathbf{a}^{k+1} \right\|_F^2 \geq \frac{\mu}{2} \left\| \mathbf{a}^k - \mathbf{a}^{k+1} \right\|_2^2. \quad \square$$

Similar results hold for $\mathbf{b}$ and $\mathbf{c}$.

Next we want to show that Ψ decreases monotonically at each iteration.

Theorem 3.5.5 *(Sufficiently decrease property) Let* Ψ *represent the objective function in Eq. 3.21, let* $\boldsymbol{\omega}^k = \left(\mathbf{a}^k, \mathbf{b}^k, \mathbf{c}^k\right)$*, then we have*

$$\Psi\left(\boldsymbol{\omega}^k\right) - \Psi\left(\boldsymbol{\omega}^{k+1}\right) \geq \rho \left\| \boldsymbol{\omega}^k - \boldsymbol{\omega}^{k+1} \right\|_2^2$$

for some constant $\rho > 0$. In addition, we have

$$\sum_{k=0}^{\infty} \left\| \boldsymbol{\omega}^k - \boldsymbol{\omega}^{k+1} \right\|_2^2 < \infty.$$

Proof: By the previous result in 2, we have

$$\Psi\left(\boldsymbol{\omega}^k\right) - \Psi\left(\boldsymbol{\omega}^{k+1}\right) \geq \frac{1}{2}\left\{\mu_1 \left\| \mathbf{a}^k - \mathbf{a}^{k+1} \right\|_2^2 + \mu_2 \left\| \mathbf{b}^k - \mathbf{b}^{k+1} \right\|_2^2 + \mu_3 \left\| \mathbf{c}^k - \mathbf{c}^{k+1} \right\|_2^2\right\},$$

Then

$$\lim_{n\to\infty} \sum_{k=0}^{n-1} \left\| \boldsymbol{\omega}^k - \boldsymbol{\omega}^{k+1} \right\|_2^2 \leq \lim_{n\to\infty} \frac{1}{\rho}\left(\Psi\left(\boldsymbol{\omega}^0\right) - \Psi\left(\boldsymbol{\omega}^n\right)\right) < \infty$$

$$\Longrightarrow \sum_{k=0}^{\infty} \left\| \boldsymbol{\omega}^k - \boldsymbol{\omega}^{k+1} \right\|_2^2 < \infty. \qquad \square$$

Remark 3.5.6 The sequence $\left\{\boldsymbol{\omega}^k\right\}_{k\in\mathbb{N}}$ is bounded, since the regularization terms in the objective function Ψ bounds the blocks $\mathbf{a}, \mathbf{b}$, and $\mathbf{c}$, and $\Psi\left(\boldsymbol{\omega}^k\right)$ is non-increasing. $\diamond$

Theorem 3.5.7 *Let $\left\{\boldsymbol{\omega}^k\right\}_{k\in\mathbb{N}}$ be the sequence generated by our regularized ALS algorithm, then there exists a constant $\nu > 0$ such that for any $k \in \mathbb{N}$, there is a vector $\eta^{k+1} \in \partial\Psi\left(\boldsymbol{\omega}^{k+1}\right)$ such that*

$$\left\|\eta^{k+1}\right\| \leq \nu \left\| \boldsymbol{\omega}^k - \boldsymbol{\omega}^{k+1} \right\|$$

Proof: Let k be a positive integer. By the first-order optimality condition of the subproblems, we have

$$\begin{aligned}
&\nabla_b f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^k\right) + \mathbf{L}_b^\top \mathbf{L}_b \mathbf{b}^{k+1} = 0, \\
&\nabla_c f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^{k+1}\right) + \mathbf{L}_c^\top \mathbf{L}_c \mathbf{c}^{k+1} = 0, \\
&\nabla_a f\left(\mathbf{a}^{k+1}, \mathbf{b}^{k+1}, \mathbf{c}^{k+1}\right) + \mathbf{L}_a^\top \mathbf{L}_a \mathbf{a}^{k+1} = 0.
\end{aligned}$$

Let us define $\eta_2^{k+1} := \nabla_b f\left(\boldsymbol{\omega}^{k+1}\right) - \nabla_b f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^k\right)$, then

$$\eta_2^{k+1} = \nabla_b f\left(\boldsymbol{\omega}^{k+1}\right) + \mathbf{L}_b^\top \mathbf{L}_b \mathbf{b}^{k+1} = \nabla_b \Psi\left(\boldsymbol{\omega}^{k+1}\right).$$

Similarly, define $\eta_3^{k+1} := \nabla_c f\left(\boldsymbol{\omega}^{k+1}\right) - \nabla_c f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^{k+1}\right)$ implies $\eta_1^{k+1} = \nabla_c \Psi\left(\boldsymbol{\omega}^{k+1}\right)$, and define $\eta_3^{k+1} := \nabla_a f\left(\boldsymbol{\omega}^{k+1}\right) - \nabla_a f\left(\mathbf{a}^{k+1}, \mathbf{b}^{k+1}, \mathbf{c}^{k+1}\right)$ implies $\eta_2^{k+1} = \nabla_b \Psi\left(\boldsymbol{\omega}^{k+1}\right)$. Hence the vector $\eta^{k+1} = \left(\eta_1^{k+1}, \eta_2^{k+1}, \eta_3^{k+1}\right) \in \partial\Psi\left(\boldsymbol{\omega}^{k+1}\right)$.

Moreover, since the sequence $\left\{\boldsymbol{\omega}^k\right\}_{k\in\mathbb{N}}$ is bounded, and the objective function f is twice continuously differentiable, then by the mean value theorem, ∇f is Lipschitz continuous. Hence, there exists a constant $C_2 > 0$ such that

$$
\begin{aligned}
\left\|\eta_2^{k+1}\right\| &= \left\|\nabla_b f\left(\boldsymbol{\omega}^{k+1}\right) - \nabla_b f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^k\right)\right\| \\
&= \left\|\nabla_b f\left(\boldsymbol{\omega}^{k+1}\right) - \nabla_b f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^k\right) + \nabla_b f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^k\right) - \nabla_b f\left(\mathbf{a}^k, \mathbf{b}^k, \mathbf{c}^k\right)\right\| \\
&\leq \left\|\nabla_b f\left(\boldsymbol{\omega}^{k+1}\right) - \nabla_b f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^k\right)\right\| + \left\|\nabla_b f\left(\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^k\right) - \nabla_b f\left(\mathbf{a}^k, \mathbf{b}^k, \mathbf{c}^k\right)\right\| \\
&\leq C_1 \left\|\boldsymbol{\omega}^{k+1} - (\mathbf{a}^k, \mathbf{b}^{k+1}, \mathbf{c}^k)\right\| + C_2 \left\|\boldsymbol{\omega}^k - \boldsymbol{\omega}^{k+1}\right\| \\
&\leq P_2 \left\|\boldsymbol{\omega}^k - \boldsymbol{\omega}^{k+1}\right\|
\end{aligned}
$$

for some constant $P_2 \geq C_1 + C_2$

Similarly, there exists cosntants $P_1, P_3 > 0$ such that $\left\|\eta_1^{k+1}\right\| \leq P_1 \left\|\boldsymbol{\omega}^k - \boldsymbol{\omega}^{k+1}\right\|$ and $\left\|\eta_3^{k+1}\right\| \leq P_3 \left\|\boldsymbol{\omega}^k - \boldsymbol{\omega}^{k+1}\right\|$.

Setting $\nu = \max\{P_1, P_2, P_3\}$ gives $\left\|\eta^{k+1}\right\| \leq \nu \left\|\boldsymbol{\omega}^k - \boldsymbol{\omega}^{k+1}\right\|$. □

Theorem 3.5.8 *Let $\left\{\boldsymbol{\omega}^k\right\}_{k\in\mathbb{N}}$ be the sequence generated by our regularized ALS algorithm, then $\left\{\boldsymbol{\omega}^k\right\}_{k\in\mathbb{N}}$ converges to the critical point of Ψ.*

Proof: By Lemma 2.3 in [3], the squared distance function $g(x, y) = \|x - y\|^2$ for $x, y \in \mathbb{R}^m$ is semi-algebraic. Moreover, the univariate function $f(x) = |x|$ is semi-algebraic [43]. Since the addition and the composition of semi-algebraic functions are semi-algebraic, the objective function given in Eq. 3.21 is semi-algebraic. Hence the objective function satisfies the Kurdyka-Łojasiewicz property. Then by Theorem. 3.5.7 and 3.5.5, results follow from Theorem 2.9 in [3]. □

3.6 Numerical Results

In this section, we illustrate the performance of (grayscale) video background and foreground separation with the following methods (1) SS-SVD (2) regularized BMD-ALS with spatiotemporal slice-wise SVD initial guess (BMD-ALS$_{SVD}$) (3) Dynamic mode decomposition (DMD) [30] (4) regularized BMD-ALS with DMD initial guess (BMD-ALS$_{DMD}$) on six video datasets:

1. First is the "Simulated Video" based on the BM-rank 3 generative spatiotemporal video model discussed in Sec. 3.4. We use a 50×50 cloud image [92] and simulate 2 rectangular objects intensities 85 and 15 respectively and are moving over 30 time steps across the background.

2. The second, "Car" video is from a surveillance video of moving vehicles on a highway. The video data is available in MATLAB and can be loaded using the command `VideoReader('traffic.mj2')`. This video consists of 120 grayscale frames each of size 120×160. The camera is pointing at an entrance of a highway where the cars are traveling from the top right corner to the bottom of the image as frames progress.

3. The third video, referred to as the "Escalator" video, is taken from a surveillance video of escalators and people. The data was originally used in [111] which is now available in the author's github repository [109]. This video consists of 200 frames in grayscale and each frame is of size 130×160. The camera is pointing directly from above at three parallel escalators. The staircase of the escalators is moving periodically while people either walk across the platforms above the escalator, stand on the escalators or walk down the escalator on the right.

4. The last three videos are from the the Scene Background Initialization (SBI) dataset [7, 67]. The three videos include "Hall and Monitor", "Human Body", and "IBM Test". The video frames are re-scaled to sizes 100×147, 100×134, and 100×134 respectively. For videos "Hall and Monitor" and "Human Body", the

first 150 frames are included for the following experiments. The total number of frames of the "IBM Test" video is 90.

Specific (grayscale) video frames of each of the six video datasets are displayed in Fig. (3.3). In Chapter. 4, we will demonstrate a way to extend the third-order tensor BMD to fourth-order color video tensors and show the background/foreground separation results for the color videos available including the "Simulated Video" and the SBI video data tensors.

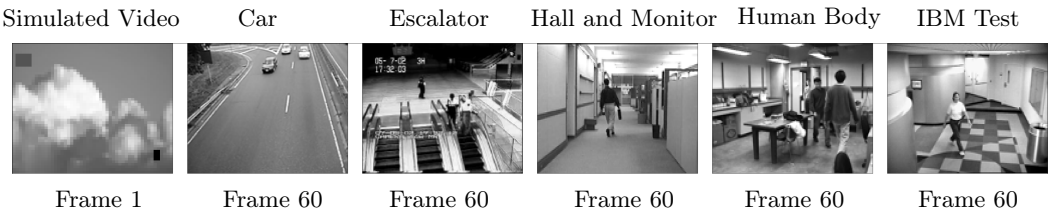


Figure 3.3: Video frames selected to display from the six testing videos.

The choices of these videos are based on the following considerations. The first video is a simulated data tensor which we use to check the generative video model results discussed in Sec. 3.4. The second and the third video tensors differ by the level of the background image complexities. While the highway road is a simpler image, the escalator scene represents a more complex background image which a higher tensor BM-rank is expected to be used in order to achieve a similar level of reconstruction accuracy as the "Car" video. The last three videos taken from the SBI datasets contain foreground objects of different sizes comparing to the size of the image.

For all numerical experiments in the third-order grayscale video decomposition case, we found that following regularization parameters in the alternating least-squares algorithm provide sufficiently good reconstruction results. For penalizing the $\boldsymbol{\mathcal{A}}$ factor tensor, we set $\lambda_1 = 0.01$ and $\lambda_t = 1, \forall\, 2 \leq t \leq \ell$, where $\mathbf{L}_a = \underset{k,i}{\oplus}\mathbf{L}$ with $\mathbf{L} = \text{diag}([\lambda_1, \lambda_2, \ldots, \lambda_\ell])$, $1 \leq k \leq p$ and $1 \leq i \leq m$. We found in experiments that these regularization parameters are robust as long as the first parameter λ_1 is approximately 100 times smaller than other parameters $\lambda_2, \ldots, \lambda_\ell$. For penalizing the $\boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$ factor tensors, we set $\beta = \gamma = 1$ so that $\mathbf{L}_b = \underset{i,j}{\oplus}\mathbf{I}_{\ell\times\ell}$ and $\mathbf{L}_c = \underset{j,k}{\oplus}\mathbf{I}_{\ell\times\ell}$, for

all $1 \le i \le m$, $1 \le j \le n$, and $1 \le k \le p$.

Moreover, we set the termination criteria for the alternating least-squares algorithm to be either when a total number of 100 iterations is reached and or the relative error of consecutive iterates, i.e. $\|\mathcal{X}^{k+1} - \mathcal{X}^k\|_F / \|\mathcal{X}^k\|_F$, is less than 10^{-5}. For the following numerical experiments, all video frames displayed are re-scaled to the pixel range of $[0, 255]$ on grayscale. For both of the DMD method and the BMD-ALS$_{DMD}$ method, since the resulting dynamic modes may be complex, we followed the code included in [55] and the real parts of the solutions were taken for the video reconstructions.

In Fig. (3.4), we compare the relative error (RE[†]) of video reconstruction results of the BMD-ALS algorithm with both the SS-SVD and the DMD initial guesses for different BM-rank ℓ, $\ell = 2, \ldots, 10$. The BM-decomposition is computed on the entire video tensor. As we can see in Fig. (3.4), The BMD-ALS algorithm with both starting guesses performed similarly well for different BM-ranks on real video tensors. As for the simulated video constructed with three BM-rank 1 terms, a BM-rank 2 approximating by the BMD-ALS$_{SVD}$ algorithm outperforms the BMD-ALS$_{DMD}$ algorithm by a factor of 10. Additionally, we also observe that the approximation relative error for the simulated video is small even when ℓ is larger than the number of BM-rank 1 terms of the video. Overall, as we can see in the plot that, as ℓ increases, the amount of the improvement of the approximation quality decreases since the differences between RE become smaller for larger ℓ. This result suggests that surveillance type videos inherently have a low BM-rank. Based on Fig. (3.4), we determined the target BM-ranks for the real videos having a similar level of reconstruction accuracy with RE approximately 0.04. In Table. 3.1, we summarize the target BM-ranks for each video as well as the compression ratio (CR[‡]) for the BMD method.

Next we illustrate in detail the quality of separating the background and foreground using the BM-decomposition method, and compare with the methods of the

[†]Relative Error: RE = $\frac{\|\mathcal{X} - \hat{\mathcal{X}}\|_F}{\|\mathcal{X}\|_F}$.

[‡]Compression Ratio: CR = $\frac{\text{Uncompressed size}}{\text{Compressed size}}$.

SS-SVD and DMD. We note that for the DMD method, video sequences are usually processed on segments. Choosing a proper video segment size can reduce the processing time and potentially improve the video reconstruction accuracy. For the real-video streams, the sequences are broken into segments of 30 frames, and the rank truncation parameter is taken as $\ell = 30 - 1 = 29$ as suggested in [55]. For the simulated video, we found that choosing a segment size of 10 with rank $\ell = 9$ yields a much better result. All other methods including SS-SVD, BMD-ALS$_{SVD}$, and BMD-ALS$_{DMD}$ are applied to the entire video sequence.

We also note that both the SS-SVD and the DMD methods are algorithms for the background initialization application in computer vision [23]. These methods aim to model a clear background image without foreground objects from a video sequence. Obtaining the foreground objects are usually done by subtracting the background image from the overall video sequence. For the SS-SVD method, we followed the suggestions given in [41] and selected the first rank-1 component from the slice-wise SVDs for the video stationary background sequence reconstruction. However, it is also possible to include more components to improve the reconstructed background quality as the authors discussed in [41]. Additionally, the first image frame of the sequence is used to represent the approximated background, since the stationary background image sequence consists of the same image repeated p times. As for the DMD method, we take $t_1 = 0$ in Eq. (3.11) to represent the background image. Detailed derivations and discussions for the DMD method are referred to the original article [55].

For our method, we model both the background image sequence and the foreground objects' motion using a few BM-rank 1 terms. In particular, the background video sequence is reconstructed by $\boldsymbol{\mathcal{X}}^{\text{bg}}_{\text{BMD}} = \texttt{bmp}\,(\boldsymbol{\mathcal{A}}_{:,1,:}, \boldsymbol{\mathcal{B}}_{:,:,1}, \boldsymbol{\mathcal{C}}_{1,:,:})$, and the foreground video sequence is reconstructed by $\boldsymbol{\mathcal{X}}^{\text{fg}}_{\text{BMD}} = \sum_{t=2}^{\ell} \texttt{bmp}\,(\boldsymbol{\mathcal{A}}_{:,t,:}, \boldsymbol{\mathcal{B}}_{:,:,t}, \boldsymbol{\mathcal{C}}_{t,:,:})$, where ℓ is the target BM-rank.

In Table. 3.2, we summarize the video background reconstruction comparisons of the four methods: SS-SVD, BMD-ALS$_{SVD}$, DMD, and BMD-ALS$_{DMD}$, for the

simulated video, and the three SBI video sequences where the ground-truth background images are available. The following metrics for the background evaluations were measured: Average Gray-level Error (AGE), Percentage of Error Pixels (pEPs), Percentage of Clustered Error Pixels (pCEPs), Peak-Signal-to-Noise-Ratio (PSNR), and Multi-Scale Structural Similarity Index (MS-SSIM). For the metics AGE, pEPs, and pCEPs, the lower the values, the better is the background estimate. For the PSNR and MS-SSIM values, the higher the values, the better is the background estimate. Detailed descriptions of the these metrics and MATLAB implementations are available in [6]. For the SS-SVD and the DMD methods, the first frame of the reconstructed background sequence are compared with the ground-truth background images. For the BMD method with both SS-SVD and DMD initial guesses, the metrics are measured against every frame in the reconstructed background video sequence, and the average values over all frames are taken and summarized in the table. As we can see from Table. (3.2), the DMD-ALS$_{SVD}$ method performs well for the "Simulated", the "Hall and Monitor", and the "Human Body" video sequences.

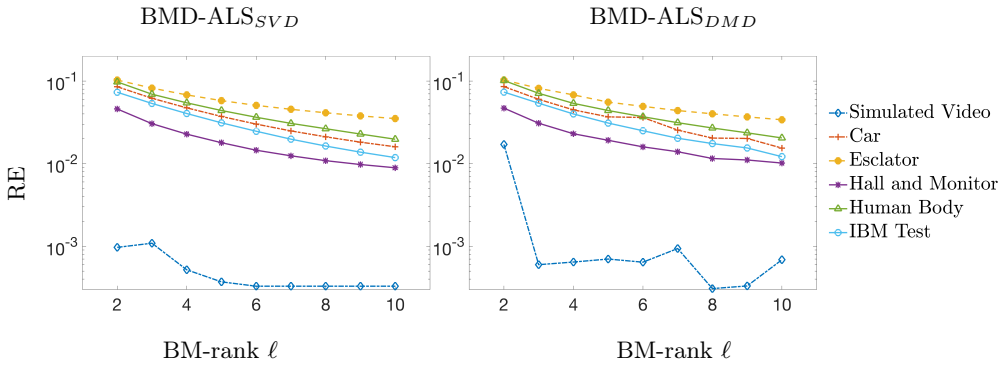


Figure 3.4: Comparison of the BMD-ALS$_{SVD}$ and the BMD-ALS$_{DMD}$ methods with change in BM-rank for the six testing videos.

In Fig. (3.5), the first video background and foreground frames are displayed for the simulated video. From left to right are respectively: the ground–truth frames, and frames reconstructed by the four methods: SS-SVD, DMD, BMD-ALS$_{SVD}$, and BMD-ALS$_{DMD}$. As we can see from the images, the BMD-ALS method with both initial guesses obtained great separation results of the stationary background and the moving foreground objects compared to the ground-truth frames. In particular,

the BMD-ALS$_{SVD}$ results are almost identical to the ground-truth frames.

Furthermore, as discussed in Sec. 3.4, for the simulated video with two objects of different intensities and are traveling with different trajectories, the first BM-rank 1 term of the generative video model consists of the background image captured by $\boldsymbol{\mathcal{A}}_{:,1,:}$. The foreground component, denoted $\boldsymbol{\mathcal{X}}^{\text{fg}} = \texttt{bmp}(\boldsymbol{\mathcal{A}}_{:,2,:}, \boldsymbol{\mathcal{B}}_{:,:,2}, \boldsymbol{\mathcal{C}}_{2,:,:}) + \texttt{bmp}(\boldsymbol{\mathcal{A}}_{:,3,:}, \boldsymbol{\mathcal{B}}_{:,:,3}, \boldsymbol{\mathcal{C}}_{3,:,:})$, consists of the vertical locations of the objects captured by columns of the slice $\boldsymbol{\mathcal{B}}_{:,:,2}$ for the first object, and $\boldsymbol{\mathcal{B}}_{:,:,3}$ for the second object respectively. The horizontal positions of the two objects are modeled by rows of the slices $\texttt{squeeze}(\boldsymbol{\mathcal{C}}_{2,:,:})$ and $\texttt{squeeze}(\boldsymbol{\mathcal{C}}_{3,:,:})$. In Fig. (3.6.a), we demonstrated the ground-truth factor tensor slices of the BM-rank 3 generative video model in the first row. Additionally, the BM-rank 2 decomposition of the simulated video by both the BMD-ALS$_{SVD}$, and the BMD-ALS$_{DMD}$ methods are shown in the second and the third rows of Fig. (3.6.a) respectively. We note that in the tensor BMD-ALS$_{SVD}$ results, the background slice $\boldsymbol{\mathcal{A}}_{:,1,:}$ have inverted pixel intensities compared to the ground-truth image and hence we multiplied $\boldsymbol{\mathcal{A}}$ by -1 for display purposes. To make the overall BM-product positive, the factor tensor $\boldsymbol{\mathcal{C}}$ is multiplied by -1.

Overall, as we can see in Fig. (3.6.a), the BMD-ALS results with both initial guesses recover the stationary background image captured in $\boldsymbol{\mathcal{A}}_{:,1,:}$. However, regarding the foreground motion, the decomposition with the DMD initial guess is less accurate for obtaining information depicting the trajectory of the motion comparing with using the SS-SVD initial guess. Moreover, the object trajectories in a BM-rank 2 decomposition using the BMD-ALS$_{SVD}$ method is well aligned with the ground-truth that was constructed by a sum of three BM-rank 1 terms. This observation suggests that this simulated video could potentially be modeled exactly by a BM-rank 2 model, but further investigations are needed. Finally, the ALS convergence plots are shown in Fig. (3.6.b). As we can see in the plots, the BMD-ALS$_{SVD}$ algorithm outperforms the BMD-ALS$_{DMD}$ algorithm with a much smaller relative error at convergence.

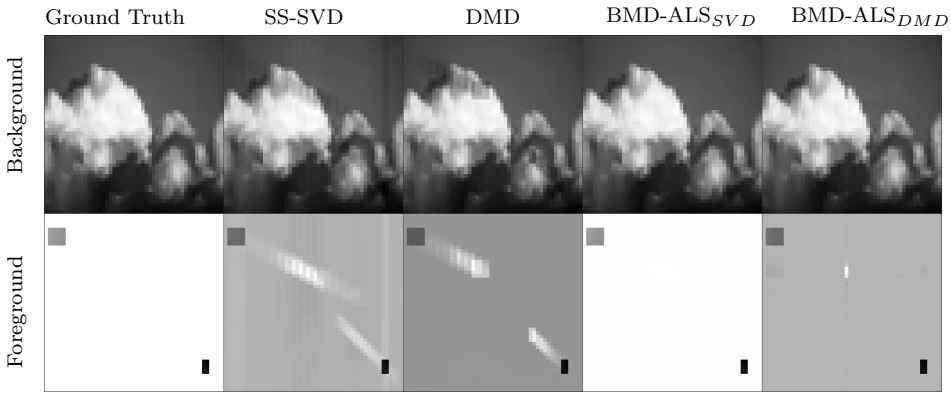


Figure 3.5: Video background/foreground separation results to the simulated BM-rank 3 video with ℓ = 2 approximations. Comparison of the ground-truth, the SS-SVD method, the BMD-ALS method with slice-wise SVD initial guess (BMD-ALS$_{SVD}$), the DMD algorithm, and the BMD-ALS method with the DMD initial guess (BMD-ALS$_{DMD}$). The first frame is selected to display.

Fig. (3.7) shows the ground-truth background images, the first frames of the SS-SVD and DMD reconstructed background, and the 60th frames of the BMD reconstructed background video sequences. We compare qualitatively the four methods. As we can see in the figure, all four methods perform comparably well for approximating the stationary road background in the "Car" video sequence. Although for the DMD method, the foreground car motion trajectories are visible in the reconstructed background image. For both the"Escalator" and the "Human Body" video sequences, vertical artifacts appeared for the SS-SVD, BMD-ALS$_{SVD}$ and BMD-ALS$_{DMD}$ methods. As for the DMD method, we observe similar foreground motions present in the background, which is also the case for the other two videos "Hall and Monitor" and "IBM Test". Additionally, in the "IBM Test" video, we observe a rectangular artifact appeared at the top left corner for the BMD-ALS$_{DMD}$ result. In our experiments, we noticed this artifact could be eliminated if we increase the BM-rank. Overall, all four methods provide visually appealing stationary background reconstructions. This result aligns with the significance of application of the SS-SVD method and the DMD method in background subtraction for surveillance video. In Fig. (3.8), the 60-th frame of each foreground video sequence is displayed for the four methods of interest. Both the SS-SVD and the DMD foreground frames are obtained

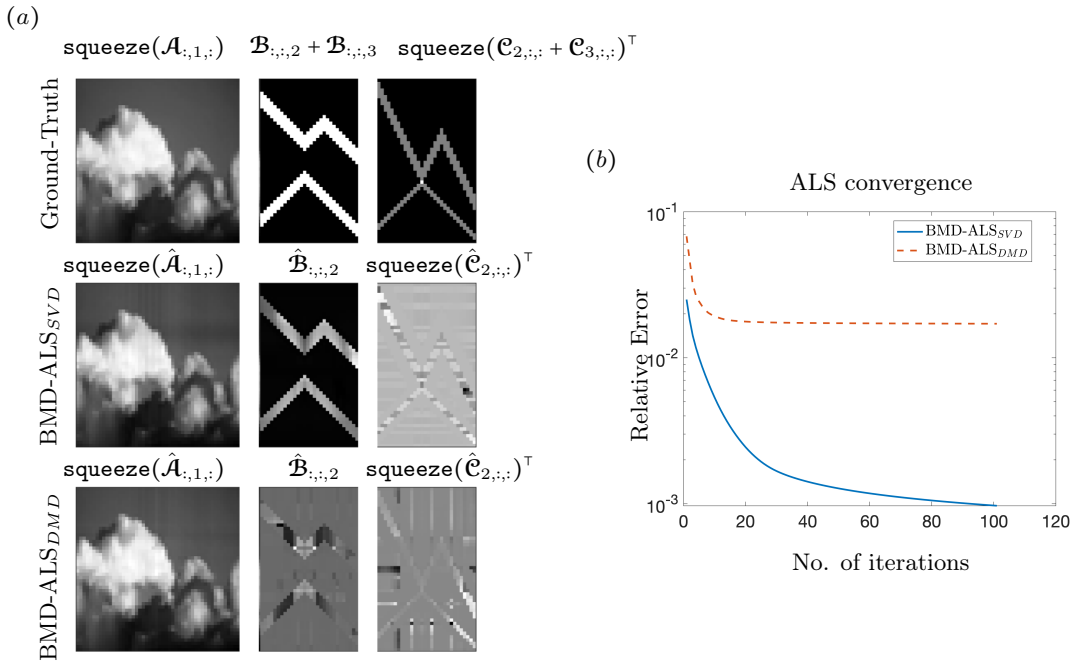


Figure 3.6: Comparison of the BM-rank 2 approximation factor tensors and ALS convergence to the simulated video with spatiotemporal slice-wise SVD initial guess (BMD-ALS$_{SVD}$) and the DMD initial guess (BMD-ALS$_{DMD}$). (a) The ground-truth factor tensor slices with $\texttt{squeeze}(\mathcal{A}_{:,1,:})$ capturing the stationary background image, $\mathcal{B}_{:,:,2}$ depicting the vertical trajectory of the objects' motion and $\texttt{squeeze}(\mathcal{C}_{2,:,:})^\top$ depicting the horizontal trajectory of the objects' motion. (b) The ALS convergence comparisons for the two methods. The relative error is computed for the overall video tensor $\frac{\|\mathcal{X}-\hat{\mathcal{X}}\|_F}{\|\mathcal{X}\|_F}$.

by subtracting the background frame shown in Fig. (3.7) from the original video. For the BMD method, we show the 60-th foreground frame from the reconstructed sequence $\mathcal{X}^{\text{fg}}_{\text{BMD}}$.Overall, we observe that the background subtraction methods have the drawback of unable to remove the background information completely when the background is not perfectly approximated. For instance, in the "Human Boday" and "IBM Test" video sequences, the office background and the checkerboard floor are also visible in the foreground frames as shown in Fig. (3.8). However, in the BMD-ALS$_{SVD}$ reconstructed foreground frames, we can see that the background information are clearly removed. Despite the artifacts, the people and their cloths are well approximated with great details comparable to the ground-truth. Additionally, we see that when the object size is small relative to the frame size, the foreground frames are approximated with better visual quality for all four methods, such as the

frames of the "Car", "Escalator" and "IBM Test" video sequences shown in Fig. (3.7). For the videos "Hall and Monitor" and "Human Body", we observe more artifacts in the BMD results when the foreground objects are relatively large comparing to the frame size. A larger BM-rank is needed to recover better foreground results. Moreover, we notice that between the two different initializations of SS-SVD and DMD, the BMD-ALS algorithm with SS-SVD initial guess often outperforms the DMD initial guess making it a more suitable choice for the video background/foreground separation application. In particular, in the next section we will demonstrate the impressive fourth-order color BMD results using only the SS-SVD initial guess.

Video sequence		Simulated	Car	Escalator	Hall and Monitor	Human Body	IBM Test
Height × # Frames × Width		$50 \times 30 \times 50$	$120 \times 120 \times 160$	$130 \times 200 \times 160$	$100 \times 150 \times 147$	$100 \times 150 \times 134$	$100 \times 90 \times 134$
BM-rank ℓ		2	5	8	3	6	4
CR		0.1467	0.1146	0.1515	0.0704	0.1448	0.1143
RE	BMD-ALS$_{SVD}$	**0.0010**	0.0374	0.0413	**0.0304**	**0.0365**	0.0405
	BMD-ALS$_{DMD}$	0.0111	**0.0364**	**0.0401**	0.0311	0.0372	**0.0402**

Table 3.1: Comparison of the video reconstruction results for the BMD-ALS algorithm with both the SS-SVD and the DMD initializations.

Video Sequence	Methods	Background Evaluation Metrics				
		AGE	pEPs	pCEPs	MS-SSIM	PSNR
Simulated	SS-SVD	4.1104	0.0332	0.0120	0.9800	30.1105
	DMD	2.33	0.0392	0.0192	0.9753	27.15
	BMD-ALS$_{SVD}$	**0.2750**	**0**	**0**	**0.9998**	**54.0179**
	BMD-ALS$_{DMD}$	1.3276	0.0111	0.0041	0.9927	37.8710
Hall and Monitor	SS-SVD	11.6629	0.0540	0.0265	0.9670	25.3568
	DMD	**3.6932**	0.0493	0.0278	**0.9745**	29.8246
	BMD-ALS$_{SVD}$	5.9975	**0.0439**	**0.0242**	0.9742	**30.4387**
	BMD-ALS$_{DMD}$	10.4643	0.1357	0.1059	0.9594	27.8144
Human Body	SS-SVD	22.6678	0.3798	0.2626	0.8954	17.7232
	DMD	11.7410	**0.1357**	**0.0769**	0.8718	19.8134
	BMD-ALS$_{SVD}$	**11.6589**	0.1772	0.1166	**0.9277**	**21.5720**
	BMD-ALS$_{DMD}$	35.0373	0.5935	0.4518	0.7640	15.9829
IBM Test	SS-SVD	6.4078	**0.0425**	**0.0142**	0.9822	**29.4235**
	DMD	**5.1016**	0.0904	0.0619	0.9616	26.5307
	BMD-ALS$_{SVD}$	9.9211	0.1410	0.1004	**0.9827**	27.9591
	BMD-ALS$_{DMD}$	11.6214	0.0934	0.0690	0.9456	18.5423

Table 3.2: Background evaluation metrics, comparison of the four methods: SS-SVD, DMD, BMD-ALS$_{SVD}$, and BMD-ALS$_{DMD}$ for video sequences with available ground-truth background image.

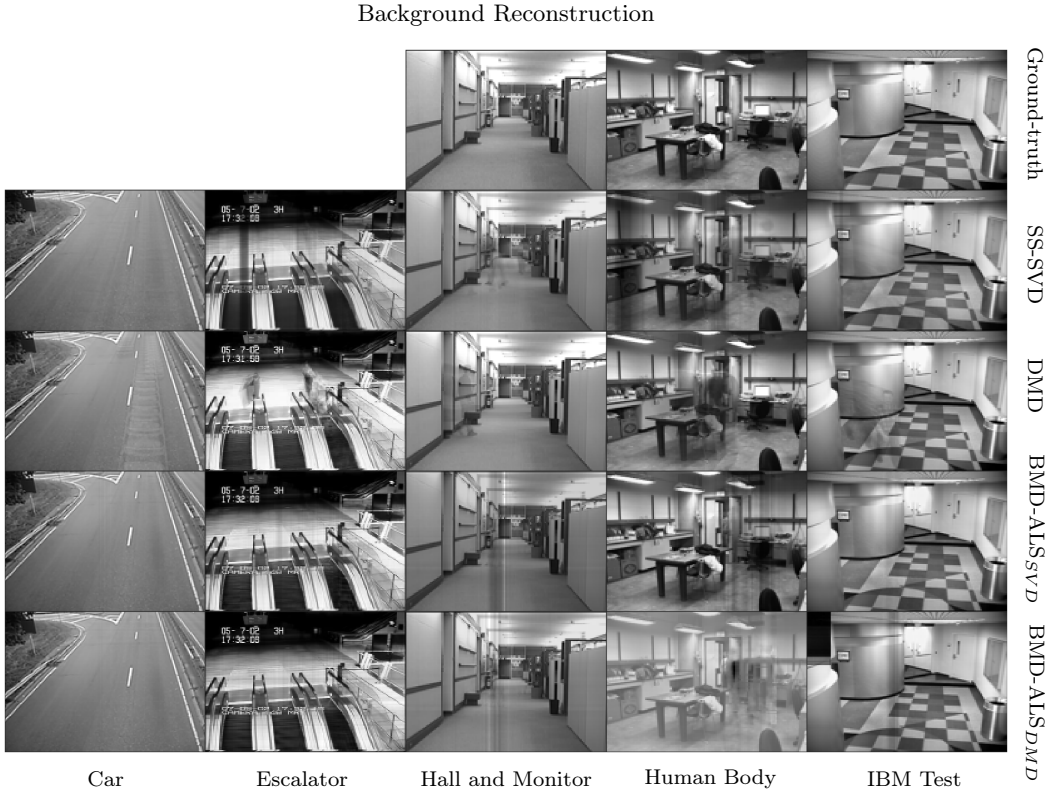


Figure 3.7: Background reconstruction. Comparison of the four methods: 1. SS-SVD; 2. BMD-ALS$_{SVD}$; 3. DMD; 4. BMD-ALS$_{DMD}$.

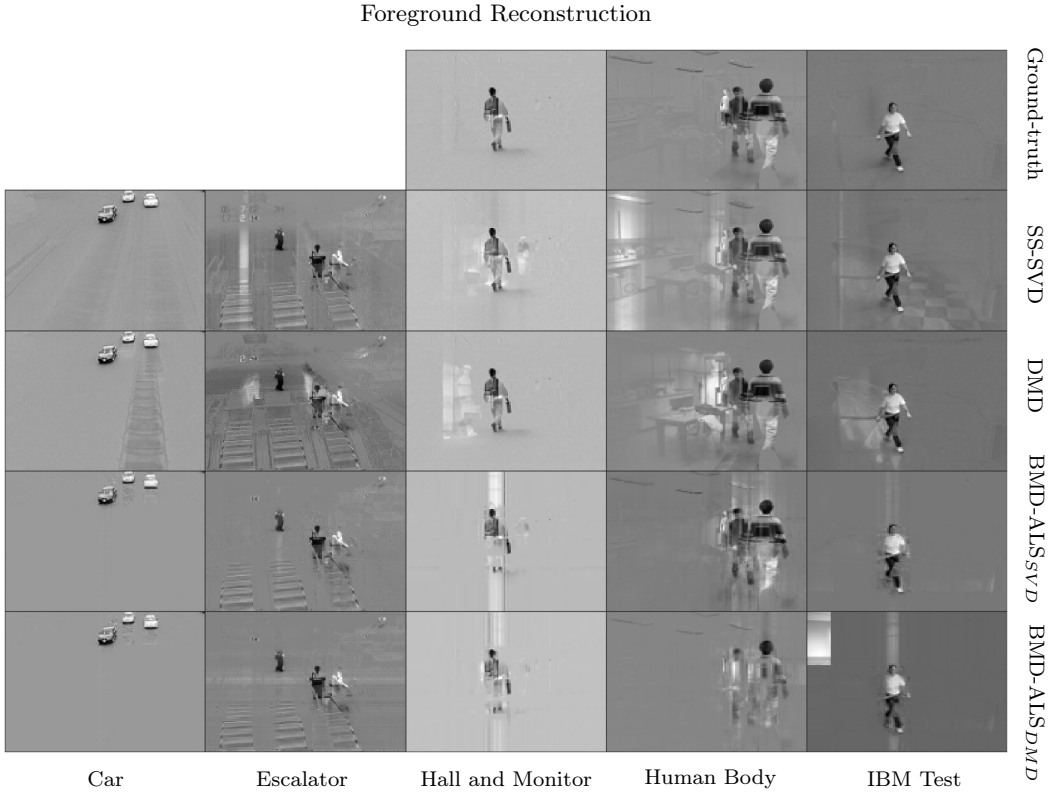


Figure 3.8: Foreground reconstruction: frame 60. Comparison of the four methods: 1. SS-SVD; 2. BMD-ALS$_{SVD}$; 3. DMD; 4. BMD-ALS$_{DMD}$.

Chapter 4

Decomposing fourth-order tensors

As introduced in [24], the order-4 tensor BMP is defined for a 4-tuple of conformable order-4 tensors.

Definition 4.0.1 *Given a conformable 4-tuple of order 4 tensors* $\mathcal{A}^{(1)} \in \mathbb{R}^{n_1 \times \ell \times n_3 \times n_4}$, $\mathcal{A}^{(2)} \in \mathbb{R}^{n_1 \times n_2 \times \ell \times n_4}$, $\mathcal{A}^{(3)} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times \ell}$, *and* $\mathcal{A}^{(4)} \in \mathbb{R}^{\ell \times n_2 \times n_3 \times n_4}$, *then*

$$\mathcal{X} = \texttt{bmp}\left(\mathcal{A}^{(1)}, \mathcal{A}^{(2)}, \mathcal{A}^{(3)}, \mathcal{A}^{(4)}\right) \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$$

is specified entry-wise as follows,

$$\mathcal{X}_{i_1,i_2,i_3,i_4} = \sum_{1 \le t \le \ell} \mathcal{A}^{(1)}_{i_1,t,i_3,i_4} \mathcal{A}^{(2)}_{i_1,i_2,t,i_4} \mathcal{A}^{(3)}_{i_1,i_2,i_3,t} \mathcal{A}^{(4)}_{t,i_2,i_3,i_4}. \tag{4.1}$$

for all $1 \le i_1 \le n_1, 1 \le i_2 \le n_2, 1 \le i_3 \le n_3$, *and* $1 \le i_4 \le n_4$.

As the tensor order increases, the computation time and storage requirements for high-order tensor BMP rapidly become impractical. However, when dealing with orientation-dependent data, where one dimension is significantly smaller than the other three, as in the case of color videos, applying a fourth-order BMP would be unnecessarily complex and computationally expensive. To address this, we introduce a new BM-product-like operation, termed **BMP4**, along with a corresponding BM-product-like decomposition for fourth-order tensors derived from triples of fourth-order tensors.

Definition 4.0.2 *Given a tensor triplet* $\mathcal{A} \in \mathbb{R}^{m \times \ell \times n \times q}, \mathcal{B} \in \mathbb{R}^{m \times p \times \ell \times q}$, *and* $\mathcal{C} \in \mathbb{R}^{\ell \times p \times n \times q}$, *which are conformable on the first three indices, then* $\mathcal{X} = \texttt{bmp4}(\mathcal{A}, \mathcal{B}, \mathcal{C}) \in \mathbb{R}^{m \times p \times n \times q}$, *is specified entry-wise as follows,*

$$\mathcal{X}_{i,j,k,z} = \sum_{1 \le t \le \ell} \mathcal{A}_{i,t,k,z} \mathcal{B}_{i,j,t,z} \mathcal{C}_{t,j,k,z} \tag{4.2}$$

for all $1 \le i \le m, 1 \le j \le p, 1 \le k \le n,$ *and* $1 \le z \le q$.

When $\ell = 1$, a BM_4-outer product corresponds to the BMP_4 of the conformable order-3 tensor slices $\mathcal{A} \in \mathbb{R}^{m\times 1\times n\times q}$, $\mathcal{B} \in \mathbb{R}^{m\times 1\times n\times q}$, and $\mathcal{C} \in \mathbb{R}^{1\times p\times n\times q}$. Then the corresponding notion of rank can be defined as follows.

Definition 4.0.3 *The BM_4-rank of $\mathcal{X} \in \mathbb{R}^{m\times p\times n\times q}$ is the minimum number of the BM_4-outer products of conformable tensor slices that sum up to $\mathcal{X}$.*

We note that our definition of `bmp4` given above is equivalent to the classical fourth-order BMP given in Eq. 4.1 with the last factor tensor $\mathcal{A}^{(3)}$ set to be all-one.

4.1 An ALS Algorithm of BMD for Fourth-Order Tensors

Using `bmp4` defined above, we can extend the BM-decompostion to fourth-order video tensor. Given $\mathcal{X} \in \mathbb{R}^{m\times p\times n\times q}$, a low BM_4-rank approximation to $\mathcal{X}$ is obtained by solving

$$\min_{\mathcal{A},\mathcal{B},\mathcal{C}} \|\mathcal{X} - \text{bmp4}\,(\mathcal{A},\mathcal{B},\mathcal{C})\|_F^2, \tag{4.3}$$

where $(\mathcal{A},\mathcal{B},\mathcal{C}) \in \mathbb{R}^{m\times\ell\times n\times q}\times\mathbb{R}^{m\times p\times\ell\times q}\times\mathbb{R}^{\ell\times p\times n\times q}$. We can similarly solve the problem by an alternating least-squares algorithm to obtain the fourth order tensor BMD.

First, we set the initial guess $\mathcal{B}^0$ to be all ones of size $m\times p\times\ell\times q$. We then obtain the initial guess pair $\mathcal{A}^0$ and $\mathcal{C}^0$ by fixing the fourth dimension and applying either a matrix decomposition based or tensor slice-wise decomposition based methods as described in Phase I (Sec. 3.2.1) to the order-3 tensor slices $\mathcal{X}_{:,:,:,z}$, for all $1 \le z \le q$. The order-3 tensor slice decomposition results in the pair $\mathcal{A}^0$ and $\mathcal{C}^0$. Next, to update $\mathcal{B}$, we modify the algorithm given in Phase II (Sec. 3.2.3) which requires solving the following problem

$$\widehat{\mathcal{B}} = \min_{\mathcal{B}\in\mathbb{R}^{m\times p\times\ell\times q}} \left\|\mathcal{X} - \texttt{bmp4}\left(\mathcal{A}^0,\mathcal{B},\mathcal{C}^0\right)\right\|_F^2.$$

Similar to the third-order case, we show that this problem can be reduced to parallelizable smaller least-squares subproblems.

By the definitions of the Frobenius norm and forth-order tensor BMP, we have

$$\left\|\mathcal{X} - \texttt{bmp4}\left(\mathcal{A}^0, \mathcal{B}, \mathcal{C}^0\right)\right\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{n}\sum_{z=1}^{q}\left|\mathcal{X}_{i,j,k,z} - \sum_{t=1}^{\ell}\mathcal{A}^0_{i,t,k,z}\mathcal{B}_{i,j,t,z}\mathcal{C}^0_{t,j,k,z}\right|^2. \quad (4.4)$$

Holding the indices i, j, and z fixed, we regroup the remaining indices and rewrite the right-hand side of Eq. (4.4) as

$$\sum_{(i-1)p+j=1}^{mp}\sum_{(z-1)n+k=1}^{nq}\left|\mathbf{x}^{(i,j)}_k - \sum_{t=1}^{\ell}\mathbf{H}^{(i,j)}_{k,t}\mathbf{b}^{(i,j)}_t\right|^2,$$

where $\mathbf{x}^{(i,j)} = \texttt{vec}(\texttt{squeeze}(\mathcal{X}_{i,j,:,:})) \in \mathbb{R}^{nq\times 1}$, $\mathbf{b}^{(i,j)} = \texttt{vec}(\texttt{squeeze}(\mathcal{B}_{i,j,:,:})) \in \mathbb{R}^{\ell q\times 1}$, and $\mathbf{H}^{(i,j)} \in \mathbb{R}^{nq\times \ell q}$ with entries specified as $\mathbf{H}^{(i,j)}_{k,t} = \mathcal{A}^0_{i,t,k,z}\mathcal{C}^0_{t,j,k,z}$. This expression suggests that $\mathcal{B}$ can be updated via solving the following parallelizable smaller linear least-squares subproblems

$$\widehat{\mathbf{b}}^{(i,j)} = \min_{\mathbf{b}^{(i,j)}}\left\|\mathbf{x}^{(i,j)} - \mathbf{H}^{(i,j)}\mathbf{b}^{(i,j)}\right\|_F^2. \quad (4.5)$$

We additionally define the fourth order tensor transposes to help expressing the ALS algorithm using `bmp4`.

Definition 4.1.1 *Given $\mathcal{X} \in \mathbb{R}^{m\times p\times n\times q}$, we define*

$$\mathcal{X}^{\mathsf{T}_4} = \texttt{permute}\left(\mathcal{X}, [2,3,1,4]\right), \quad \mathcal{X}^{\mathsf{T}_4^2} = \texttt{permute}\left(\mathcal{X}, [3,1,2,4]\right).$$

Note the above definition of the fourth order tensor transpose is the same for the third-order case holding the last dimension fixed.

Under this definition, we can show the following convenient fact around which we can base a fourth-order BMD-ALS algorithm.

Theorem 4.1.2 *Given $\mathcal{X} = \texttt{bmp4}\left(\mathcal{A}, \mathcal{B}, \mathcal{C}\right)$,*

$$\mathcal{X}^{\mathsf{T}_4} = \texttt{bmp4}\left(\mathcal{A}, \mathcal{B}, \mathcal{C}\right)^{\mathsf{T}_4} = \texttt{bmp4}\left(\mathcal{B}^{\mathsf{T}_4}, \mathcal{C}^{\mathsf{T}_4}, \mathcal{A}^{\mathsf{T}_4}\right).$$

We can then formulate the following ALS algorithm based on the order-4 BMD of a given tensor $\mathcal{X} \in \mathbb{R}^{m\times p\times n\times q}$.

$$\begin{aligned}
\mathcal{B}^{k+1} &= \min_{\mathcal{B}\in\mathbb{R}^{m\times p\times \ell\times q}} \left\|\mathcal{X} - \texttt{bmp4}\left(\mathcal{A}^k, \mathcal{B}, \mathcal{C}^k\right)\right\|_F^2, \\
\left(\mathcal{C}^{\top_4}\right)^{k+1} &= \min_{\mathcal{C}^{\top_4}\in\mathbb{R}^{p\times n\times \ell\times q}} \left\|\mathcal{X}^{\top_4} - \texttt{bmp4}\left(\left(\mathcal{B}^{\top_4}\right)^{k+1}, \mathcal{C}^{\top_4}, \left(\mathcal{A}^{\top_4}\right)^{k}\right)\right\|_F^2, \\
\left(\mathcal{A}^{\top_4^2}\right)^{k+1} &= \min_{\mathcal{A}^{\top_4^2}\in\mathbb{R}^{n\times m\times \ell\times q}} \left\|\mathcal{X}^{\top_4^2} - \texttt{bmp4}\left(\left(\mathcal{C}^{\top_4^2}\right)^{k+1}, \mathcal{A}^{\top_4^2}, \left(\mathcal{B}^{\top_4^2}\right)^{k+1}\right)\right\|_F^2.
\end{aligned}$$

Similar to the discussion for the third order case, each ALS subproblem can be solved in parallel as smaller sized linear least-squares problems.

4.2 Application to Color Video with Regularization

Consider a given video with three color channels $\mathcal{X} \in \mathbb{C}^{m\times p\times n\times 3}$. Each frame of the video is of size $m \times n \times 3$, where the third dimension encodes color. The total number of frames is p. Fixing the color channel dimension, our goal is to obtain a tensor triplet $\mathcal{A} \in \mathbb{R}^{m\times \ell\times n\times 3}$, $\mathcal{B} \in \mathbb{R}^{m\times n\times \ell\times 3}$, and $\mathcal{C} \in \mathbb{R}^{\ell\times p\times n\times 3}$ from the fourth order BMD of $\mathcal{X}$ such that $\mathcal{X} \approx \texttt{bmp4}\left(\mathcal{A}, \mathcal{B}, \mathcal{C}\right)$.

Since the different tensor slices along the color channel for the factors associated with the temporal dimension (for example, if the frames are oriented as lateral slices, the factor tensors with temporal dimension is $\mathcal{B}$ and $\mathcal{C}$) should be the same, or the difference is small. We consider adding the following constraints to mitigate the non-uniqueness of the solutions present in the order-4 BMD. We note that comparing to the regularized ALS given in Sec. 3.5, this constraint is specific for color video decomposition applications but does not promote the background/foreground separation task.

Consider solving for the factor tensor $\mathcal{B} \in \mathbb{C}^{m\times p\times \ell\times 3}$, we add the following regularization

$$\left\|\mathcal{B}_{i,j,t,1} - \mathcal{B}_{i,j,t,2}\right\|_2 < \epsilon \text{ and } \left\|\mathcal{B}_{i,j,t,2} - \mathcal{B}_{i,j,t,3}\right\|_2 < \epsilon. \tag{4.6}$$

Notice that, from Eq. (4.5), holding the indices (i, j) fixed, we obtain the vector $\mathbf{b}^{(i,j)} \in \mathbb{R}^{3\ell\times 1}$ by column major order vectorization of the slice $\mathcal{B}_{i,j,:,:}$, i.e. $\mathbf{b}^{(i,j)} =$

$\texttt{vec}(\texttt{squeeze}(\mathcal{B}_{i,j,:,:}))$, for all $1 \le i \le m, 1 \le j \le p$.

Let us define the difference operator $\mathbf{R} \in \mathbb{R}^{2\ell \times 3\ell}$ such that, for all $1 \le t \le \ell, 1 \le v \le 2$,

$$\mathbf{R}_{(v-1)\ell+t, v\ell+t} = -1 \text{ and } \mathbf{R}_{(v-1)\ell+t,(v-1)\ell+t} = 1$$

So the constraint given in Eq. (4.6) is equivalent to applying the difference operator $\mathbf{R}$ on the vectorized slice $\mathcal{B}_{i,j,:,:}$ such that $\left\|\mathbf{R}\mathbf{b}^{(i,j)}\right\|_2 < \epsilon$.

Similarly, for the factor tensor $\mathcal{C}^{\top_4} \in \mathbb{R}^{p \times n \times \ell \times 3}$, we have

$$\left\|\mathcal{C}^{\top_4}_{j,k,t,1} - \mathcal{C}^{\top_4}_{j,k,t,2}\right\|_2 < \epsilon \text{ and } \left\|\mathcal{C}^{\top_4}_{j,k,t,2} - \mathcal{C}^{\top_4}_{j,k,t,3}\right\|_2 < \epsilon.$$

we can rewrite in an equivalent form with the difference operator $\mathbf{R}$ applied to the vectorized tensor slice $\mathbf{c}^{(j,k)} = \texttt{vec}(\texttt{squeeze}(\mathcal{C}^{\top_4}_{j,k,:,:}))$ as $\left\|\mathbf{R}\mathbf{c}^{(j,k)}\right\| < \epsilon$, for all $1 \le j \le p, 1 \le k \le n$.

Thus, a constrained ALS algorithm for the fourth order BMD of a the color video tensor $\mathcal{X}$ has the following decoupled subproblems

$$\hat{\mathbf{b}}^{(i,j)} = \min_{\mathbf{b}^{(i,j)} \in \mathbb{R}^{3\ell \times 1}} \left\|\mathbf{y}^{(i,j)}_{\mathcal{X}} - \mathbf{H}^{(i,j)}_{\mathcal{A},\mathcal{C}}\mathbf{b}^{(i,j)}\right\|_F^2 + \left\|\mathbf{R}\mathbf{b}^{(i,j)}\right\|_F^2, \tag{4.7}$$

and the factor tensor $\mathcal{B}$ is approximated by letting $\widehat{\mathcal{B}}_{i,j,:,:} = \texttt{reshape}\left(\hat{\mathbf{b}}^{(i,j)}, [\ell, 3]\right)$. Similarly, we update $\mathcal{C}$ by solving

$$\hat{\mathbf{c}}^{(j,k)} = \min_{\mathbf{c}^{(j,k)} \in \mathbb{R}^{3\ell \times 1}} \left\|\mathbf{y}^{(j,k)}_{\mathcal{X}^{\top_4}} - \mathbf{H}^{(j,k)}_{\mathcal{B},\mathcal{A}}\mathbf{c}^{(j,k)}\right\|_F^2 + \left\|\mathbf{R}\mathbf{c}^{(j,k)}\right\|_F^2, \tag{4.8}$$

and hence $\widehat{\mathcal{C}}^{\top_4}_{j,k,:,:} = \texttt{reshape}\left(\hat{\mathbf{c}}^{(j,k)}, [\ell, 3]\right)$. Lastly, we update $\mathcal{A}$ by solving

$$\hat{\mathbf{a}}^{(k,i)} = \min_{\mathbf{a}^{(k,i)} \in \mathbb{R}^{3\ell \times 1}} \left\|\mathbf{y}^{(k,i)}_{\mathcal{X}^{\top^2_4}} - \mathbf{H}^{(k,i)}_{\mathcal{C},\mathcal{B}}\mathbf{a}^{(k,i)}\right\|_F^2. \tag{4.9}$$

Then $\hat{\mathcal{A}}^{\top^2_4}_{k,i,:,:} = \texttt{reshape}\left(\hat{\mathbf{a}}^{(k,i)}, [\ell, 3]\right)$.

4.3 Color Video Decomposition Results

In Fig. (4.1), we demonstrate the background and foreground separation results using the BMD-ALS$_{SVD}$ method for the fourth order color videos. The same BM-ranks of 2, 3, 6 and 4 are used for the "Simulated", "Hall and Monitor", "Human Body", and "IBM Test" color video sequences as the grayscale videos. Frame 60 of the original videos and the reconstructed background/foreground videos are selected to display. As we can see from the figure, the color background frame has excellent reconstruction quality compared to the ground-truth except small artifacts appeared in the "Hall and Monitor" video sequence. In the reconstructed foreground frames, vertical artifacts also appeared around the people moving across the background with small effect on visualizing the people.

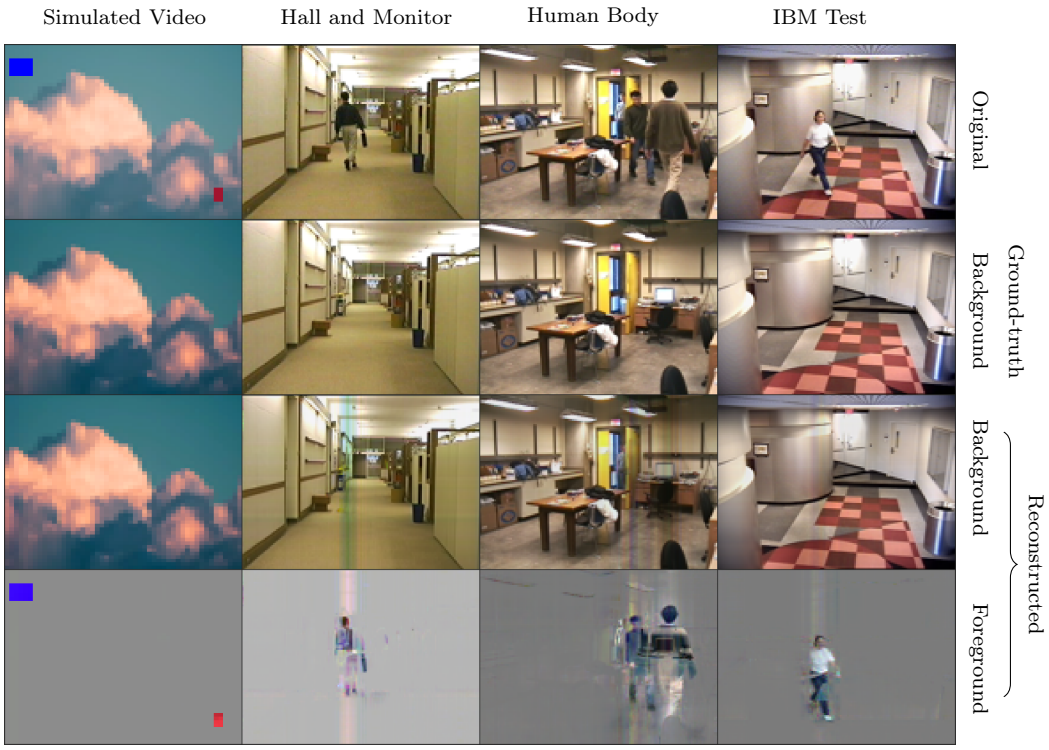


Figure 4.1: Color video background/foreground separation by the fourth-order BMD-ALS$_{SVD}$ method.

Chapter 5

Tensor Completion

Missing data can arise in many scenarios in real-world video and image processing applications. For instance, malfunctioning sensors or cameras can lead to missing data in photos and video recordings. Environmental factors like adverse weather conditions, poor lighting, or occlusions can affect the quality of the data and result in missing or distorted frames. Studying higher-order tensor completion problems incorporating the multidimensional structure of the color images, hyperspectral images or grayscale videos has drawn great attention in recent years [62, 111]. A survey on the recent developments in methods and algorithms for solving the tensor completion problem is available in [91].

A common assumption for solving the tensor completion problem is that the underlying unknown data tensor can be well approximated by a low rank tensor. The definition of the tensor rank can vary depending on the underlying rank assumption of a given tensor [91]. Commonly studied tensor ranks include the tensor CP rank [53], the Tucker rank [17], and the tensor tubal-rank known from the t-SVD factorization [48].

More recently, a novel tensor Bhattacharya-Mesner decomposition (BMD) has been proposed [93] for third-order tensors. It is demonstrated that third-order spatiotemporal data is highly compressible. In particular, we observed that the spatiotemporal slices of the surveillance video exhibit low-rankness. Moreover, a generative low BM-rank video model was introduced and has shown that the motions of objects traversing the background scene over time can be captured within a few slices of the BMD factor tensors. We expect that these factor tensor slices are low rank. As for the factor tensor slices corresponding to the spatial dimension, they are less expected to be low rank. Motivated by these observations, we propose a novel BM-factor tensor slicewise nuclear norm minimization (BMNN) method for

the tensor completion problem. In this work, we focus on third-order tensors.

5.1 BM-decomposition for Completion

As shown in [93], given a grayscale video tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, which has a stationary background and moving objects in the foreground, assume the video frames are ordered from front to back, then in a BM-rank ℓ, $\ell \ll \min\{n_1, n_2, n_3\}$, decomposition of $\mathcal{X}$, i.e. $\mathcal{X} = \texttt{bmp}(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, we expect that the frontal slices of $\mathcal{A}_2 \in \mathbb{R}^{n_1 \times n_2 \times \ell}$ consists of the background image information and hence are less likely to be low-rank. The lateral slices of $\mathcal{A}_1 \in \mathbb{R}^{n_1 \times \ell \times n_3}$ and the horizontal slices of $\mathcal{A}_3 \in \mathbb{R}^{\ell \times n_2 \times n_3}$ contain positions of the objects over time, and we can expect them to be low-rank if the motions are not complex. Motivated by the observations, we formulate a BM-factor tensor slicewise rank constrained tensor completion problem, and later we will solve the problem by an efficient ADMM scheme.

5.1.1 Problem Formulation

Suppose the unknown tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ has a low BM-rank. Based on the BM-rank ℓ decomposition $\mathcal{X}$, we form the following BM-factor tensor slicewise rank minimization model:

$$\begin{aligned} &\min_{\mathcal{X}, \mathcal{A}_i^{(t)}} \sum_{i=1}^{3} \sum_{t=1}^{\ell} \alpha_{i,t} \text{rank}\left(\mathcal{A}_i^{(t)}\right) \\ &\text{subject to } \mathcal{X} = \texttt{bmp}(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3), \, \mathcal{X}_\Omega = \mathcal{T}_\Omega, \end{aligned} \tag{5.2}$$

where the matrix slices are defined to be $\mathcal{A}_1^{(t)} := \mathcal{A}_1(:, t, :)$, $\mathcal{A}_2^{(t)} := \mathcal{A}_2(:, :, t)$, and $\mathcal{A}_3^{(t)} := \mathcal{A}_3(t, :, :)$. The equation $\mathcal{X}_\Omega = \mathcal{T}_\Omega$ represents that $\mathcal{X}_{i,j,k} = \mathcal{T}_{i,j,k}$ for all $(i, j, k) \in \Omega$, where Ω denotes the indices of the observed entries of $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

Since the matrix rank is not a convex function, we use the nuclear norm relaxation of the rank [9, 11]. Additionally, we relax the BMD constraint and we rewrite the

problem as follows,

$$\begin{aligned} \min_{\mathcal{X},\mathcal{A}_i^{(t)}} & \sum_{i=1}^{3}\sum_{t=1}^{\ell}\alpha_{i,t}\|\mathcal{A}_i^{(t)}\|_* + \frac{\lambda}{2}\left\|\mathcal{X} - \texttt{bmp}\left(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3\right)\right\|_F^2 \\ \text{s.t. } & \mathcal{X}_\Omega = \mathcal{T}_\Omega. \end{aligned} \tag{5.4}$$

Furthermore, we assume that $\alpha_{i,t} = \alpha_i, \forall\ 1 \le t \le \ell$, and rearrange the matrix slices $\mathcal{A}_i^{(t)}$ into a block-diagonal matrix $\texttt{bdiag}(\mathcal{A}_i)$. In particular, we define the following three matrices

$$\begin{aligned} \texttt{bdiag}(\mathcal{A}_1) &= \bigoplus_{1\le t\le \ell} \texttt{squeeze}(\mathcal{A}_1(:,t,:)), \\ \texttt{bdiag}(\mathcal{A}_2) &= \bigoplus_{1\le t\le \ell} \mathcal{A}_2(:,:,t), \\ \texttt{bdiag}(\mathcal{A}_3) &= \bigoplus_{1\le t\le \ell} \texttt{squeeze}(\mathcal{A}_3(t,:,:)). \end{aligned} \tag{5.6}$$

Then we can rewrite the slicewise nuclear norm minimization problem in Eq. (5.4) as

$$\begin{aligned} \min_{\mathcal{X},\mathcal{A}_i} & \sum_{i=1}^{3}\alpha_i\left\|\texttt{bdiag}(\mathcal{A}_i)\right\|_* + \frac{\lambda}{2}\left\|\mathcal{X} - \texttt{bmp}\left(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3\right)\right\|_F^2 \\ \text{s.t. } & \mathcal{X}_\Omega = \mathcal{T}_\Omega. \end{aligned} \tag{5.8}$$

The above model in Eq. (5.8) is called a BM-factor tensor slicewise nuclear norm minimization (BMNN) approach.

5.1.2 Main algorithm

We present the ADMM algorithm for solving the proposed BMNN model given in Eq. (5.8). First, we reformulate the problem into the following equivalent form

$$\begin{aligned} \min_{\mathcal{X},\mathcal{A}_i,\mathcal{H}_i} & \sum_{i=1}^{3}\alpha_i\left\|\texttt{bdiag}\left(\mathcal{H}_i\right)\right\|_* + \frac{\lambda}{2}\left\|\mathcal{X} - \texttt{bmp}\left(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3\right)\right\|_F^2 \\ \text{subject to } & \mathcal{X}_\Omega = \mathcal{T}_\Omega, \mathcal{A}_i = \mathcal{H}_i, \quad 1 \le i \le 3, \end{aligned} \tag{5.10}$$

where $\mathcal{H}_i, 1 \le i \le 3$ are the auxiliary variables.

The partial augmented Lagrangian function for Eq. (5.10) is given by

$$\begin{aligned}
&\mathcal{L}\left(\mathcal{H}_1,\mathcal{H}_2,\mathcal{H}_3,\mathcal{X},\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3,\mathcal{Y}_1,\mathcal{Y}_2,\mathcal{Y}_3,\mu\right)\\
&=\sum_{i=1}^{3}\left(\alpha_i\left\|\texttt{bdiag}\left(\mathcal{H}_i\right)\right\|_*+\left\langle\texttt{bdiag}\left(\mathcal{Y}_i\right),\texttt{bdiag}\left(\mathcal{H}_i-\mathcal{A}_i\right)\right\rangle+\frac{\mu}{2}\left\|\mathcal{H}_i-\mathcal{A}_i\right\|_F^2\right)\\
&+\frac{\lambda}{2}\left\|\mathcal{X}-\texttt{bmp}\left(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3\right)\right\|_F^2
\end{aligned}\tag{5.12}$$

where $\mathcal{Y}_i$ are the Lagrange multipliers and are of the same sizes as $\mathcal{H}_i$ and $\mathcal{A}_i$, and $\mu>0$ is a penalty parameter.

Next, we present the ADMM iterative scheme to minimize $\mathcal{L}$ with respect to the variables $(\mathcal{H}_1,\mathcal{H}_2,\mathcal{H}_3,\mathcal{X},\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3,\mathcal{Y}_1,\mathcal{Y}_2,\mathcal{Y}_3,\mu)$.

We first update $\mathcal{H}_i$, $1\le i\le 3$ fixing other variables. The optimization subproblem is formulated as

$$\min_{\mathcal{H}_i}\alpha_i\left\|\texttt{bdiag}\left(\mathcal{H}_i\right)\right\|_*+\left\langle\texttt{bdiag}\left(\mathcal{Y}_i\right),\texttt{bdiag}\left(\mathcal{H}_i-\mathcal{A}_i\right)\right\rangle+\frac{\mu}{2}\left\|\mathcal{H}_i-\mathcal{A}_i\right\|_F^2.$$

Equivalently, we can rewrite the above minimization problem as

$$\min_{\mathcal{H}_i}\alpha_i\left\|\texttt{bdiag}\left(\mathcal{H}_i\right)\right\|_*+\frac{\mu}{2}\left\|\texttt{bdiag}\left(\mathcal{H}_i-\mathcal{A}_i+\mathcal{Y}_i/\mu\right)\right\|_F^2.$$

This problem is well studied in literature [8, 64] and has a closed-form solution:

$$\mathcal{H}_i^{k+1}=\mathrm{SVT}_{\alpha_i/\mu^k}\left(\texttt{bdiag}\left(\mathcal{A}_i^k-\mathcal{Y}_i^k/\mu^k\right)\right),\tag{5.13}$$

where $\mathrm{SVT}_\delta\left(\mathbf{A}\right)=\mathbf{U}\,\mathrm{diag}\left(\max\left\{\left(\boldsymbol{\sigma}-\delta\right),0\right\}\right)\mathbf{V}^\top$ is the singular value thresholding operator. The SVD of $\mathbf{A}$ is given by $\mathbf{A}=\mathbf{U}\,\mathrm{diag}\left(\boldsymbol{\sigma}\right)\mathbf{V}^\top$, and the $\max(\cdot,\cdot)$ operation is taken element-wise.

We then update the BMD factor tensors $\mathcal{A}_i, 1\le i\le 3$. Fixing other variables, we obtain the following subproblems

$$\min_{\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3}\sum_{i=1}^{3}\frac{\mu}{2}\left\|\texttt{bdiag}\left(\mathcal{A}_i-\mathcal{H}_i-\mathcal{Y}_i/\mu\right)\right\|_F^2+\frac{\lambda}{2}\left\|\texttt{bmp}\left(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3\right)-\mathcal{X}\right\|_F^2.$$

Since the Frobenius norm is defined entry-wise, we can remove the block-diagonal operator, i.e.

$$\min_{\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3} \sum_{i=1}^{3} \frac{\mu}{2} \left\| \mathcal{A}_i - \mathcal{H}_i - \mathcal{Y}_i/\mu \right\|_F^2 + \frac{\lambda}{2} \left\| \texttt{bmp}\left(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\right) - \mathcal{X} \right\|_F^2,$$

which we solve by the regularized alternating least-squares (RALS) algorithm. Holding $\mathcal{A}_1, \mathcal{A}_3$ fixed, we update $\mathcal{A}_2$ by solving the following least-squares problem

$$\min_{\mathcal{A}_2} \frac{\mu}{2} \left\| \mathcal{A}_2 - (\mathcal{H}_2 + \mathcal{Y}_2/\mu) \right\|_F^2 + \frac{\lambda}{2} \left\| \texttt{bmp}\left(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\right) - \mathcal{X} \right\|_F^2. \tag{5.14}$$

Utilizing the tensor vectorization and matricization operators illustrated in Fig. (1.5), we can equivalently write Eq. (5.14) into a linear least-squares problem via vectorizing the tensors denoted $\mathbf{a}_2 = \texttt{Tvec}(\mathcal{A}_2)$, $\mathbf{x}_2 = \texttt{Tvec}(\mathcal{X})$, and $\mathbf{v}_2 = \texttt{Tvec}(\mathcal{H}_2 + \mathcal{Y}_2/\mu)$, and metricizing the tensor pair denoted $\mathbf{H}_{\mathcal{A}_1,\mathcal{A}_3} = \texttt{Mat}(\mathcal{A}_1, \mathcal{A}_3)$. Then the problem becomes

$$\min_{\mathbf{a}_2} \frac{\mu}{2} \|\mathbf{a}_2 - \mathbf{v}_2\|_F^2 + \frac{\lambda}{2} \|\mathbf{H}_{\mathcal{A}_1,\mathcal{A}_3}\mathbf{a}_2 - \mathbf{x}_2\|_F^2 \tag{5.15}$$

The matrix $\mathbf{H}_{\mathcal{A}_1,\mathcal{A}_3} \in \mathbb{R}^{n_1 n_2 n_3 \times n_1 n_2 \ell}$ is block diagonal with a total number of $n_1 n_2$ smaller block matrices. Each block $\mathbf{H}^{(i,j)}_{\mathcal{A}_1,\mathcal{A}_3}$ is of size $n_3 \times \ell$. Then the problem given in Eq. (5.15) can decouple into $n_1 n_2$ smaller least-squares subproblems as follows

$$\min_{\mathbf{a}_2^{(i,j)}} \frac{\mu}{2} \|\mathbf{a}_2^{(i,j)} - \mathbf{v}_2^{(i,j)}\|_F^2 + \frac{\lambda}{2} \|\mathbf{H}^{(i,j)}_{\mathcal{A}_1,\mathcal{A}_3}\mathbf{a}_2^{(i,j)} - \mathbf{x}_2^{(i,j)}\|_F^2, \tag{5.16}$$

for all $1 \le i \le n_1$ and $1 \le j \le n_2$. Equivalently, we can write Eq. (5.16) as

$$\min_{\mathbf{a}_2^{(i,j)}} \frac{1}{2} \left\| \begin{bmatrix} \mathbf{H}^{(i,j)}_{\mathcal{A}_1,\mathcal{A}_3} \\ \sqrt{\frac{\mu}{\lambda}}\mathbf{I}_\ell \end{bmatrix} \mathbf{a}_2^{(i,j)} - \begin{bmatrix} \mathbf{x}_2^{(i,j)} \\ \sqrt{\frac{\mu}{\lambda}}\mathbf{v}_2^{(i,j)} \end{bmatrix} \right\|_F^2, \tag{5.17}$$

where $\mathbf{I}_m$ denotes the $m \times m$ identity matrix.

Similarly, holding $\mathcal{A}_1, \mathcal{A}_2$ fixed, and solve for $\mathcal{A}_3$, we can formulate the least-squares subproblem as

$$\min_{\mathcal{A}_3^\top} \frac{\mu}{2} \left\| \mathcal{A}_3^\top - (\mathcal{H}_3^\top + \mathcal{Y}_3^\top/\mu) \right\|_F^2 + \frac{\lambda}{2} \left\| \texttt{bmp}\left(\mathcal{A}_2^\top, \mathcal{A}_3^\top, \mathcal{A}_1^\top\right) - \mathcal{X}^\top \right\|_F^2. \tag{5.18}$$

Holding $\mathcal{A}_2, \mathcal{A}_3$ fixed, and solve for $\mathcal{A}_1$, we can formulate the least-squares subproblem as

$$\min_{\mathcal{A}_1^{\top^2}} \frac{\mu}{2} \left\| \mathcal{A}_1^{\top^2} - (\mathcal{H}_1^{\top^2} + \mathcal{Y}_1^{\top^2}/\mu) \right\|_F^2 + \frac{\lambda}{2} \left\| \texttt{bmp}\left(\mathcal{A}_3^{\top^2}, \mathcal{A}_1^{\top^2}, \mathcal{A}_2^{\top^2}\right) - \mathcal{X}^{\top^2} \right\|_F^2. \tag{5.19}$$

We note that since by transposes, the RALS steps (5.18) and (5.18) are in the same form as the first step in Eq. (5.15). Therefore, we can similarly apply the `Tvec` and `Mat` operators and decouple the problems into smaller least-squares subproblems. Hence, the RALS updates have the potential of parallelization.

To update the variable $\mathcal{X}$, we solve the following subproblem

$$\begin{aligned} &\min_{\mathcal{X}} \left\| \mathcal{X} - \texttt{bmp}\left(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\right) \right\|_F^2 \\ &\text{subject to } \mathcal{X}_\Omega = \mathcal{T}_\Omega. \end{aligned}$$

The optimal solution satisfies the following equations

$$\mathcal{X}_\Omega^{k+1} = \mathcal{T}_\Omega \text{ and } \mathcal{X}_{\Omega^C}^{k+1} = \texttt{bmp}\left(\mathcal{A}_1^{k+1}, \mathcal{A}_2^{k+1}, \mathcal{A}_3^{k+1}\right)_{\Omega^C}, \tag{5.20}$$

where Ω^C denotes the complement index set of Ω.

We also update the multiplier $\mathcal{Y}_i, 1 \le i \le 3$, by computing

$$\mathcal{Y}_i^{k+1} = \mathcal{Y}_i^k + \mu^k \left(\mathcal{H}_i^{k+1} - \mathcal{A}_i^{k+1}\right). \tag{5.21}$$

Lastly, we update the penalty parameter μ by computing

$$\mu^{k+1} = \min\left(\rho\mu^k, \mu_{\max}\right), \tag{5.22}$$

where $\rho > 1$ is a customized parameter, and $\mu_{\max} = 10^{10}$ is predefined.

The ADMM iterations terminate when the algorithm either reaches a predefined maximum number of iterations, or when the relative error (RE *) between the recovered data tensor $\hat{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and the ground truth $\boldsymbol{\mathcal{X}}_{\text{gt}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is smaller than a tolerance parameter.

5.1.2.1 Computational cost

We compute the total cost of the proposed BMNN algorithm. The main cost lies within the singular value thresholding and solving the regularized least-square subproblems steps. The cost of implementing the matrix SVD for a matrix of size $m \times p$ with $m \leq p$ is $\mathcal{O}(mp^2)$ flops [96]. Assume $n_1 \approx n_2 \approx n_3 \approx n$ for simplicity. In the first subproblem in Eq. (5.13), we need to compute the SVDs of the block diagonal matrix $\texttt{bdiag}\,(\boldsymbol{\mathcal{A}}_i - \boldsymbol{\mathcal{Y}}_i/\mu)$, $\forall\ 1 \leq i \leq 3$. However, this can be done in parallel on each of the lateral slices $(\boldsymbol{\mathcal{A}}_1 - \boldsymbol{\mathcal{Y}}_1/\mu)_{:,t,:}$, the frontal slices $(\boldsymbol{\mathcal{A}}_2 - \boldsymbol{\mathcal{Y}}_2/\mu)_{:,:,t}$, and the horizontal slices $(\boldsymbol{\mathcal{A}}_3 - \boldsymbol{\mathcal{Y}}_3/\mu)_{t,:,:}$, $\forall\ 1 \leq t \leq \ell$. Since each slice is of size $n \times n$, the cost of the slicewise SVDs is hence $\mathcal{O}(n^3)$. So the total cost of the SVT update in Eq. (5.13) is $\mathcal{O}(\ell n^3)$ flops. Next, we compute that the total cost to update the factor tensor $\boldsymbol{\mathcal{A}}_i$ is $\mathcal{O}(n^2\ell(n+\ell)^2)$ flops, and can be done in parallel as well. More specifically, for instance, to update the factor tensor $\boldsymbol{\mathcal{A}}_1$, we solve the regularized least-squares problem given in Eq. (5.15). As we have shown before, the problem decouples into n^2 smaller subproblems given in Eq. (5.17). Each block matrix is of size $(n+\ell) \times \ell$ and hence the time complexity of computing these smaller block matrix least-squares problems is $\mathcal{O}(\ell(n+\ell)^2)$ flops.

We next compute the time complexity of the HaLRTC algorithm [62]. We note that the main cost lies in the singular-value thresholding step of the flattened tensors $\boldsymbol{\mathcal{X}}_{(i)} \in \mathbb{R}^{n \times n^2}$ in HaLRTC. The total cost of computing the SVDs of the flattened tensors is in the order of $\mathcal{O}(n^5)$ flops. However, since the SVD steps are not parallelizable in HaLRTC, our proposed BMNN algorithm has the advantage of computational efficiency if its parallelizability is utilized.

*RE = $\frac{\|\hat{\boldsymbol{\mathcal{X}}} - \boldsymbol{\mathcal{X}}_{\text{gt}}\|_F}{\|\boldsymbol{\mathcal{X}}_{\text{gt}}\|_F}$.

5.2 Numerical Results

In this section, we illustrate the performance of the proposed BMNN method for tensor completion on grayscale videos and hyperspectral images. The two videos are shown in the first column of Fig. (5.2). The first video is referred to as the Basketball video [110]. Each frame is of size 144×256 in grayscale with a total number of 40 frames. The video scene consists of a non-stationary camera moving from left to right horizontally following the running players. The MATLAB data file of this video is available from GitHub [†]. The second video is referred to as the car video. The video data is available in MATLAB and can be loaded using the command VideoReader('traffic.mj2'). This data consists of 120 grayscale frames each of size 120×160. The video is taken from a surveillance camera recording moving vehicles on a highway. The hyperspectral images are shown in Fig. (5.4) in the first row. Five popular hyperspectral images are included: Cuprite, Urban, Jasper Ridge, Pavia University, Salinas, and Indian Pines. This dataset is also available online [‡].

We first evaluate the BMNN algorithm on the basketball video with 20% sample rate for different choices of the parameters of λ and μ^0 given in the Lagrangian function in Eq. (5.12), and the updating factor ρ given in Eq. (5.22). Furthermore, we fix the BM-rank to be 3. The initial factor tensor triplet $(\boldsymbol{\mathcal{A}}_1^0, \boldsymbol{\mathcal{A}}_2^0, \boldsymbol{\mathcal{A}}_3^0)$ are generated from the uniform distribution in the interval $(0, 1)$. We test the different values of λ fixing $\mu^0 = 0.001$ and $\rho = 1.01$. As we can see in Fig. (5.1.a), the BMNN algorithm performs relatively better for the choices of λ less than 1. Next, we test the different values of the initial parameter μ^0 fixing $\lambda = 0.2$ and $\rho = 1.01$. As we can see in Fig. (5.1.b), although the BMNN algorithm converges to a similar relative error for all the tested values of μ^0, it converges faster when μ^0 is relatively large. Lastly, we fix $\lambda = 0.2$ and $\mu^0 = 0.001$ and evaluate the ρ parameter. We found that the algorithm fails to converge in the case of $\rho = 1.0001$ or 1.001.

Next, we will compare our proposed BMNN algorithm with the HaLRTC algorithm studied in [62] on both videos and the hyperspectral images. Based on the

[†] https://github.com/jamiezeminzhang/Tensor_Completion_and_Tensor_RPCA
[‡] https://rslab.ut.ac.ir/data

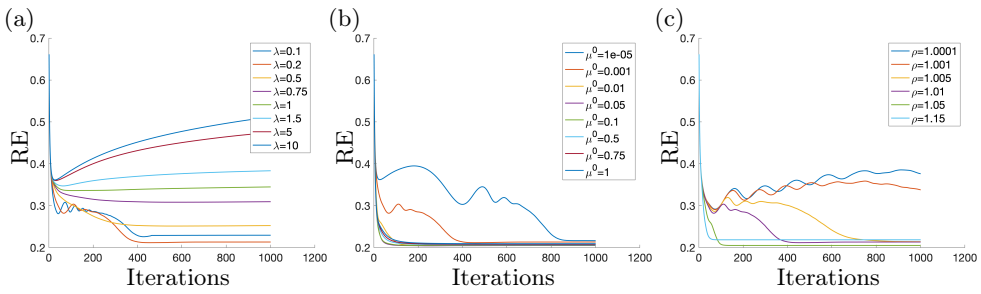


Figure 5.1: Comparison of the relative squared error of the basketball video: (a) Change λ while fixing $\mu^0 = 0.001$ and $\rho = 1.01$. (b) Change μ_0 while fixing $\lambda = 0.2$ and $\rho = 1.01$. (c) Change ρ while fixing $\lambda = 0.2$ and $\mu^0 = 0.001$.

observations made from Fig. (5.1), we choose $\lambda = 0.2$, $\mu^0 = 0.01$, and $\rho = 1.05$ for the following numerical experiments on the videos and the hyperspectral images. Moreover, we found that the penalty parameters α_i, $1 \leq i \leq 3$, of the slicewise nuclear norm yields best performances when $\alpha_i = \frac{1}{3}$. Lastly, for the following experiments on both videos and hyperspectral images, we fix the BM-rank $\ell = 3$. As for the HaLRTC algorithm, we choose the parameter $\rho = 10^{-6}$ and all other parameters are set to default as described in the paper.

The 20^{th} frame of the original and 20% sampled videos are shown in the first and the second columns in Fig. (5.2). The recovered video frames given by the HaLRTC algorithm and the BMNN algorithm are shown in the third and the fourth columns in Fig. (5.2) respectively. As we can see from the displayed frames of the basketball video, both the HaLRTC and the BMNN algorithms can recover the players and the audiences, while the BMNN algorithm can additionally recover the dashboard (stationary in the video) with more details. As for the car video, the BMNN algorithm substantially outperforms the HaLRTC method for recovering the stationary highway background. This result is in line with the high compressibility of the car video with a BM-rank 3 decomposition demonstrated in [93].

In Fig. (5.3), we further evaluate the performance of the BMNN algorithm against the sample rate. As we can see in the plots, both the BMNN and HaLRTC algorithms perform comparably well while the BMNN algorithm can recover the videos with a higher accuracy when the sample rate is low.

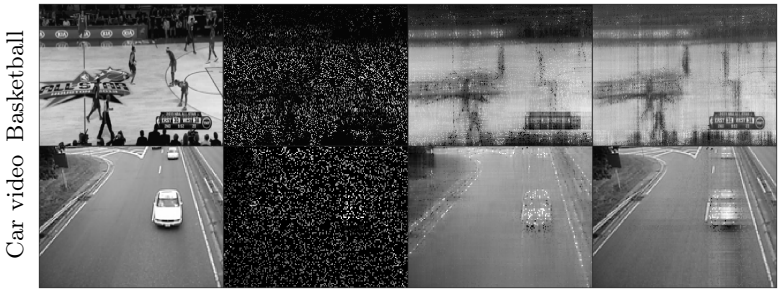


Figure 5.2: (a) The 20^{th} frames of the original basketball video (first row) and car video (second row). (b) The corresponding frames of the 20% sampled videos. (c) Recovered results by the HaLRTC method with $\rho = 10^{-6}$. (d) Recovered results by the proposed BMNN method with $\lambda = 0.2$, $\mu_0 = 0.01$, and $\rho = 1.05$.

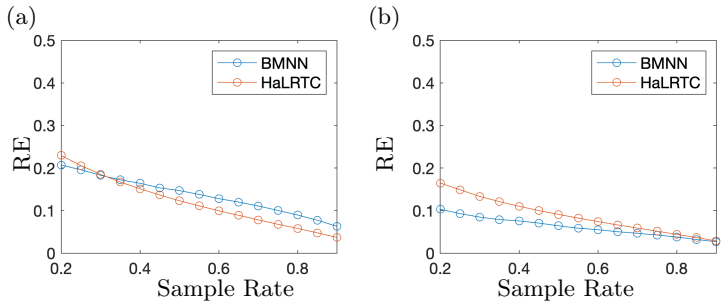


Figure 5.3: Comparison of the BMNN algorithm with the HaLRTC algorithm against sampling rate on two videos: (a) Basketball video. (b) Car video.

In Fig. (5.4), we demonstrate the performances of the proposed BMNN method on the five hyperspectral images. To simplify the computation, we subset each hyperspectral image to a 100×100 square patch while keeping the same number of channels as the original data. The observed data are sampled with a 30% sample rate at random. Overall, the two algorithms perform comparably well for recovering the missing data for hyperspectral images. The HaLRTC method performs better in recovering the river in the Jasper Ridge image and the green mountain regions in the Indian Pines image. As for the BMNN recovered images, we can observe sharper image resolutions for the Urban, Pavia University, and Salinas datasets.

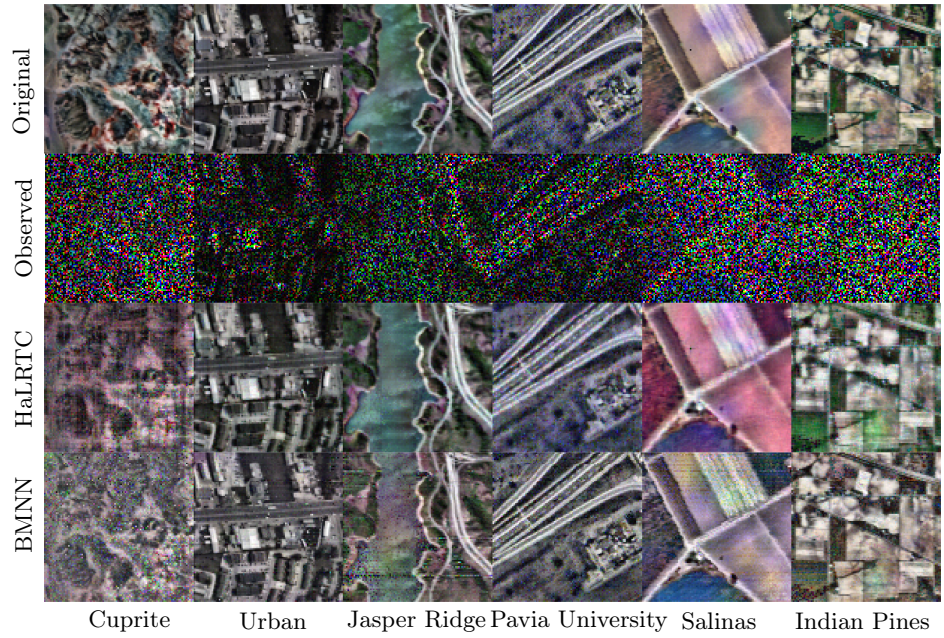


Figure 5.4: Comparison of the BMNN algorithm with the HaLRTC algorithm on five hyperspectral images. Top row: original images. Second row: 30% sampled images. Third row: HaLRTC recovered images. Last row: BMNN recovered images

Chapter 6

Non-negative factorization

Analyzing the structural properties and achieving compact representations of large data are crucial tasks in various applied research fields, including machine learning and signal processing. When dealing with non-negative data, such as images and other real-world signals, standard factorization methods may not always be suitable due to their allowance of negative values. In such cases, non-negative matrix factorization (NMF) serves as a powerful tool for extracting meaningful patterns, reducing dimensionality, and enhancing interpretability. The origins of NMF can be traced back to the work of Paatero and Tapper in 1994 [78]. However, the most influential contributions came from Lee and Seung [57, 58], whose research significantly advanced the field and played a pivotal role in shaping the development of NMF. Subsequent studies facilitated efficient non-negative data analysis by decomposing complex datasets into interpretable components while preserving the inherent non-negativity of the data. This property has made NMF particularly useful in applications like image processing [63], bioinformatics [49], text mining [79], and topic modeling [104].

As the volume of high-dimensional data continues to grow, there is a rising demand to extend NMF to higher-order tensor structures. This generalization is essential for capturing complex relationships in multi-dimensional datasets, which cannot be effectively represented using matrix-based approaches alone. Over the years, researchers have successfully developed various non-negative tensor factorization techniques to address this need. Notable advancements include the extension of NMF to non-negative CP decomposition (NCPD) [21, 102], which factorizes a tensor into a sum of rank-one components, as well as regularized NCPD [100], which incorporates additional constraints to enhance stability and interpretability. Other

significant contributions include non-negative Tucker decomposition [50], which provides a more flexible low-rank representation by decomposing a tensor into a core tensor and factor matrices, and non-negative t-SVD [32] that extends matrix SVD to tensors while preserving the non-negativity constraint. These methods have proven effective in various applications, including hyperspectral image processing [22], dynamic topic modeling [15], and cross-domain semantic analysis [71], which further underscores the importance of non-negative tensor factorizations in modern data analysis.

6.1 Related Work

Non-negative matrix factorization (NMF) is a dimensionality reduction technique that approximates a non-negative data matrix $\mathbf{M}$ as the product of two non-negative matrices $\mathbf{W}$ and $\mathbf{H}$, i.e., $\mathbf{M} \approx \mathbf{WH}$. In the work by Lee & Seung [57,58], a *multiplicative update* algorithm was formulated for solving the NMF problem with both the Least-Squares (LS) and the Kullback–Leibler (KL) divergence cost functions. The algorithm is then extended to non-negative tensor factorization (NTF) by Welling and Weber [102]. In this section, we will provide a concise review of the multiplicative update algorithm for NMF.

Given a non-negative matrix $\mathbf{M} \in \mathbb{R}_{\geq 0}^{m \times n}$, our goal is to find non-negative factor matrices $\mathbf{W} \in \mathbb{R}_{\geq 0}^{m \times \ell}$ and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{\ell \times n}$, $\ell \leq \min\{m, n\}$, that best approximates $\mathbf{M}$ under a given distance measure. One of the measures is the square of the Euclidean distance. The cost function is then formulated as

$$f(\mathbf{W}, \mathbf{H}) = \|\mathbf{M} - \mathbf{WH}\|_2^2 .$$

Another measure is referred to as the divergence formulation, in which case, the cost function is formulated as

$$D(\mathbf{M} \| \mathbf{WH}) = \sum_{1 \leq i \leq m} \sum_{1 \leq j \leq n} \left(\mathbf{M}_{i,j} \log \frac{\mathbf{M}_{i,j}}{(\mathbf{WH})_{i,j}} - \mathbf{M}_{i,j} + (\mathbf{WH})_{i,j} \right).$$

This measure is not symmetric in $\mathbf{M}$ and $(\mathbf{WH})$. It measures the divergence of $\mathbf{M}$ from its factor matrices $(\mathbf{WH})$. When $\sum_{i,j}\mathbf{M}_{i,j} = \sum_{i,j}(\mathbf{WH})_{i,j} = 1$, this divergence formulation reduces to the KL divergence or the relative entropy [57].

For the remainder of this section, we focus on solving the following least-squares problem

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{H}} \|\mathbf{M} - \mathbf{WH}\|_2^2 \\ &\text{subject to } \mathbf{W}, \mathbf{H} \geq 0. \end{aligned} \tag{6.2}$$

We note that the objective function is not convex in both variables $\mathbf{W}$ and $\mathbf{H}$ jointly, but it is convex in the variable $\mathbf{W}$ or $\mathbf{H}$ alone, holding the other variable fixed. An alternating non-negative least-squares approach was proposed in [57], which solves for $\mathbf{W}$ holding $\mathbf{H}$ fixed and then solves for $\mathbf{H}$ holding $\mathbf{W}$ fixed. The non-negative constrained ALS updates of $\mathbf{W}$ and $\mathbf{H}$ for iterations $k = 0, 1, \ldots$ are therefore

$$\begin{aligned} \mathbf{W}^{(k+1)} &= \min_{\mathbf{W}\in\mathbb{R}_{\geq 0}^{m\times r}} \left\|\mathbf{M} - \mathbf{W}\mathbf{H}^{(k)}\right\|_2^2 \\ \mathbf{H}^{(k+1)} &= \min_{\mathbf{H}\in\mathbb{R}_{\geq 0}^{r\times n}} \left\|\mathbf{M} - \mathbf{W}^{(k+1)}\mathbf{H}\right\|_2^2 \end{aligned}$$

The non-negative least-squares subproblems are solved using a gradient descent approach.

Recall that the gradient descent method solving the convex optimization problem $\min_{\mathbf{x}} f(\mathbf{x})$ has the following update step

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \boldsymbol{\eta} * f'\left(\mathbf{x}^{(k)}\right)$$

for a small step size $\boldsymbol{\eta} > 0$, and the symbol $*$ denotes the element-wise product.

Using the gradient descent method, the matrices $\mathbf{W}$ and $\mathbf{H}$ are updated as follows

$$\begin{aligned} \mathbf{W}_{i,t}^{(k+1)} &= \mathbf{W}_{i,t}^{(k)} + (\boldsymbol{\eta}_w)_{i,t}\left(\left(\mathbf{M}\left(\mathbf{H}^{(k)}\right)^\top\right)_{i,t} - \left(\mathbf{W}^{(k)}\mathbf{H}^{(k)}\left(\mathbf{H}^{(k)}\right)^\top\right)_{i,t}\right), \\ \mathbf{H}_{t,j}^{(k+1)} &= \mathbf{H}_{t,j}^{(k)} + (\boldsymbol{\eta}_h)_{t,j}\left(\left(\left(\mathbf{W}^{(k+1)}\right)^\top\mathbf{M}\right)_{t,j} - \left(\left(\mathbf{W}^{(k+1)}\right)^\top\mathbf{W}^{(k+1)}\mathbf{H}^{(k)}\right)_{t,j}\right), \end{aligned}$$

for all $1 \le i \le m$, $1 \le t \le \ell$, and $1 \le j \le n$.

By choosing the update step sizes $(\boldsymbol{\eta}_w)_{i,t}$ and $(\boldsymbol{\eta}_h)_{t,j}$ as follows

$$(\boldsymbol{\eta}_w)_{i,t} = \frac{\mathbf{W}^{(k)}_{i,t}}{\left(\mathbf{W}^{(k)}\mathbf{H}^{(k)}\left(\mathbf{H}^{(k)}\right)^\top\right)_{i,t}}, \text{ and } (\boldsymbol{\eta}_h)_{t,j} = \frac{\mathbf{H}^{(k)}_{t,j}}{\left(\left(\mathbf{W}^{(k+1)}\right)^\top \mathbf{W}^{(k+1)}\mathbf{H}^{(k)}\right)_{t,j}},$$

the updates of $\mathbf{W}$ and $\mathbf{H}$ has the following multiplicative formulations

$$\mathbf{W}^{(k+1)}_{i,t} = \mathbf{W}^{(k)}_{i,t} \frac{\left(\mathbf{M}\left(\mathbf{H}^{(k)}\right)^\top\right)_{i,t}}{\left(\mathbf{W}^{(k)}\mathbf{H}^{(k)}\left(\mathbf{H}^{(k)}\right)^\top\right)_{i,t}}, \tag{6.3}$$

$$\mathbf{H}^{(k+1)}_{t,j} = \mathbf{H}^{(k)}_{t,j} \frac{\left(\left(\mathbf{W}^{(k+1)}\right)^\top \mathbf{M}\right)_{t,j}}{\left(\left(\mathbf{W}^{(k+1)}\right)^\top \mathbf{W}^{(k+1)}\mathbf{H}^{(k)}\right)_{t,j}}. \tag{6.4}$$

As discussed in [57], Lee and Seung proved that the cost function in Eq. (6.2) is non-increasing through the multiplicative update.

6.2 Tensor Non-negative BM-Decomposition

When dealing with inherently high-dimensional data, it is more natural to represent the information within a high-dimensional space rather than reducing it to a two-dimensional structure by flattening the data into a matrix. Higher-order tensor representations provide a more consistent framework for preserving the intrinsic multi-linear structures that naturally arise in such data. In this section, we are interested in studying the third-order tensor BM-decomposition with a non-negative constraint. In particular, given a non-negative tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{m\times n\times p}_{\ge 0}$, the non-negative BM-decomposition (NNBMD) consists of solving the following optimization problem

$$\min_{\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{C}}} \left\|\boldsymbol{\mathcal{T}} - \texttt{bmp}\left(\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{C}}\right)\right\|_F^2$$

$$\text{subject to } \boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{C}} \ge 0$$

where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{m\times \ell\times p}_{\ge 0}$, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{m\times n\times \ell}_{\ge 0}$, and $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{\ell\times n\times p}_{\ge 0}$.

We extend the multiplicative update approach to solve this minimization problem. The method updates one factor tensor at a time while holding the other two factor tensors fixed. Consequently, the problem can be reformulated into the following convex subproblems, each of which is a non-negative contained least-squares subproblem

$$\mathcal{B}^{(k+1)} = \min_{\mathcal{B}\in\mathbb{R}^{m\times n\times \ell}_{\geq 0}} \left\| \mathcal{T} - \texttt{bmp}\left(\mathcal{A}^{(k)}, \mathcal{B}, \mathcal{C}^{(k)}\right)\right\|_F^2, \tag{6.5}$$

$$\left(\mathcal{C}^{\top}\right)^{(k+1)} = \min_{\mathcal{C}\in\mathbb{R}^{\ell\times n\times p}_{\geq 0}} \left\| \mathcal{T}^{\top} - \texttt{bmp}\left(\left(\mathcal{B}^{(k+1)}\right)^{\top}, \mathcal{C}^{\top}, \left(\mathcal{A}^{(k)}\right)^{\top}\right)\right\|_F^2, \tag{6.6}$$

$$\left(\mathcal{A}^{\top^2}\right)^{(k+1)} = \min_{\mathcal{A}^{\top^2}\in\mathbb{R}^{m\times \ell\times p}_{\geq 0}} \left\| \mathcal{T}^{\top^2} - \texttt{bmp}\left(\left(\mathcal{C}^{(k+1)}\right)^{\top^2}, \mathcal{A}^{\top^2}, \left(\mathcal{B}^{(k+1)}\right)^{\top^2}\right)\right\|_F^2. \tag{6.7}$$

We note that each non-negative least-squares subproblem given from Eq. (6.5) to Eq. (6.7) holds the first and the last factor tensors fixed and solves for the middle tensor. The multiplicative update algorithm for updating $\mathcal{B}$ can be directly applied to update $\mathcal{C}^{\top}$ and then $\mathcal{A}^{\top^2}$. As an example, we will illustrate the derivations of a multiplicative update algorithm for solving for $\mathcal{B}$ in the remainder of this section.

$$\mathcal{B}^{(k+1)} = \min_{\mathcal{B}\in\mathbb{R}^{m\times n\times \ell}_{\geq 0}} \left\| \mathcal{T} - \texttt{bmp}\left(\mathcal{A}^{(k)}, \mathcal{B}, \mathcal{C}^{(k)}\right)\right\|_F^2. \tag{6.8}$$

In [93], it was shown that the tensor least-squares problem could be reduced to a linear least-squares problem by vectorizing $\mathcal{B}$ and matricizing the pair $\mathcal{A}$ and $\mathcal{C}$. In particular, the vectorization strategy converts the third-order tensor $\mathcal{B} \in \mathbb{R}^{m\times n\times \ell}_{\geq 0}$ into a vector $\mathbf{b} \in \mathbb{R}^{mn\ell\times 1}_{\geq 0}$ by letting $\mathbf{b}_{(i-1)n\ell+(j-1)\ell+t} = \mathcal{B}_{i,j,t}$ for all $1 \leq i \leq m$, $1 \leq j \leq n$, and $1 \leq t \leq \ell$. For simplicity, this vectorization step is denoted by $\mathbf{b} := \texttt{Tvec}(\mathcal{B})$. The matricization of the factor tensors $\mathcal{A} \in \mathbb{R}^{m\times \ell\times p}_{\geq 0}$ and $\mathcal{C} \in \mathbb{R}^{\ell\times n\times p}_{\geq 0}$ results into a block-diagonal matrix $\mathbf{H}_{\mathcal{A},\mathcal{C}} \in \mathbb{R}^{mnp\times mn\ell}$ with

$$\begin{gathered}(\mathbf{H}_{\mathcal{A},\mathcal{C}})_{(i-1)np+(j-1)p+k,(i-1)n\ell+(j-1)\ell+t} = \mathcal{A}_{i,t,k}\mathcal{C}_{t,j,k}, \\ \forall\, 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq p, 1 \leq t \leq \ell.\end{gathered}$$

Each smaller block matrix denoted $\mathbf{H}^{(i,j)}_{\mathcal{A},\mathcal{C}}$ is of size $p \times \ell$, and $\mathbf{H}_{\mathcal{A},\mathcal{C}} := \bigoplus_{i,j} \mathbf{H}^{(i,j)}_{\mathcal{A},\mathcal{C}}$ where

$\oplus$ represents the direct sum operation. In [93], this matricization step is denoted as $\mathbf{H}_{\mathcal{A},\mathcal{C}} := \texttt{Mat}(\mathcal{A}, \mathcal{C})$.

Thus the problem of updating the factor tensor $\mathcal{B}$ given in Eq. (6.8) reduces to updating the vector $\mathbf{b}$ solving the following linear least-squares minimization problem

$$\mathbf{b}^{(k+1)} = \min_{\mathbf{b} \in \mathbb{R}^{mn\ell \times 1}_{\geq 0}} \|\mathbf{y}_{\mathcal{T}} - \mathbf{H}_{\mathcal{A}\mathcal{C}}\mathbf{b}\|_F^2$$

Similar to the matrix case, we have the following gradient descent update step

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} - \frac{\boldsymbol{\eta}_b}{2} * f'(\mathbf{b}) = \mathbf{b}^{(k)} + \boldsymbol{\eta}_b * \left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{y}_{\mathcal{T}} - \mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{H}_{\mathcal{A}\mathcal{C}} \mathbf{b}^{(k)}\right),$$

where $(\boldsymbol{\eta}_b)$ is a small positive step size.

Let us denote the index $s := (i-1)n\ell + (j-1)\ell + t$ for all $1 \leq i \leq m, 1 \leq j \leq n$, and $1 \leq t \leq \ell$. We can then re-express the above update step by an element-wise update

$$\mathbf{b}_s^{(k+1)} = \mathbf{b}_s^{(k)} + (\boldsymbol{\eta}_b)_s \left(\left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{y}_{\mathcal{T}}\right)_s - \left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{H}_{\mathcal{A}\mathcal{C}} \mathbf{b}^{(k)}\right)_s\right).$$

By letting $(\boldsymbol{\eta}_b)_s = \frac{\mathbf{b}_s^{(k)}}{\left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{H}_{\mathcal{A}\mathcal{C}} \mathbf{b}^{(k)}\right)_s}$, we obtain the following multiplicative update equation for $\mathbf{b}$:

$$\begin{aligned}
\mathbf{b}_s^{(k+1)} &= \mathbf{b}_s^{(k)} + \frac{\mathbf{b}_s^{(k)}}{\left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{H}_{\mathcal{A}\mathcal{C}} \mathbf{b}^{(k)}\right)_s} \left(\left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{y}_{\mathcal{T}}\right)_s - \left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{H}_{\mathcal{A}\mathcal{C}} \mathbf{b}^{(k)}\right)_s\right) \\
&= \mathbf{b}_s^{(k)} + \frac{\mathbf{b}_s^{(k)} \left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{y}_{\mathcal{T}}\right)_s}{\left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{H}_{\mathcal{A}\mathcal{C}} \mathbf{b}^{(k)}\right)_s} - \mathbf{b}_s^{(k)} \\
&= \mathbf{b}_s^{(k)} \frac{\left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{y}_{\mathcal{T}}\right)_s}{\left(\mathbf{H}_{\mathcal{A}\mathcal{C}}^\top \mathbf{H}_{\mathcal{A}\mathcal{C}} \mathbf{b}^{(k)}\right)_s}.
\end{aligned}$$

Similarly, we can derive the multiplicative update expression for the vectorized tensor $\mathbf{c} := \texttt{Tvec}(\mathcal{C}^\top) \in \mathbb{R}^{np\ell}_{\geq 0}$ as follows:

$$\mathbf{a}_u^{(k+1)} = \mathbf{a}_u^{(k)} \frac{\left(\mathbf{H}_{\mathcal{C}\mathcal{B}}^\top \mathbf{y}_{\mathcal{T}^{\top^2}}\right)_u}{\left(\mathbf{H}_{\mathcal{C}\mathcal{B}}^\top \mathbf{H}_{\mathcal{C}\mathcal{B}} \mathbf{a}^{(k)}\right)_u}, \tag{6.9}$$

where $u := (j-1)p\ell + (k-1)\ell + t$ for all $1 \le j \le n$, $1 \le k \le p$, and $1 \le t \le \ell$.

We then have the multiplicative update expression for the vectorized tensor $\mathbf{a} := \mathtt{Tvec}(\boldsymbol{\mathcal{A}}^{\top^2}) \in \mathbb{R}^{pm\ell}_{\ge 0}$ as follows:

$$\mathbf{c}_v^{(k+1)} = \mathbf{c}_v^{(k)} \frac{\left(\mathbf{H}_{\mathcal{BA}}^{\top}\mathbf{y}_{\mathcal{T}^{\top}}\right)_v}{\left(\mathbf{H}_{\mathcal{BA}}^{\top}\mathbf{H}_{\mathcal{BA}}\mathbf{c}^{(k)}\right)_v}, \tag{6.10}$$

where $v := (k-1)m\ell + (i-1)\ell + t$ for all $1 \le k \le p$, $1 \le i \le m$, and $1 \le t \le \ell$.

6.3 Numerical Experiments

6.3.1 Image Compression

In this section, we evaluate the performance of our non-negative tensor BM-decomposition (NNBMD) algorithm on publicly available brain MRI images from Kaggle [75]. Specifically, we utilize the first 150 images from the "glioma" training dataset for the compression experiments. The original images have a resolution of 512×512, but for computational efficiency, all images are downscaled to 128×128. Each image is converted to grayscale and subsequently stacked as frontal slices of a third-order tensor. As a result, the constructed MRI data tensor has dimensions $128 \times 128 \times 150$. We compare the proposed method against the non-negative matrix factorization (NMF) algorithm. For NMF, each pre-processed image is subsequently vectorized into a $16,384 \times 1$ column vector, forming a data matrix with 150 columns.

In the matrix case, the set of given images $\{\mathbf{X}_k\}_{k=1}^{p}$, $\mathbf{X}_k \in \mathbb{R}^{m\times n}_{\ge 0}$ are vectorized into column vectors $\{\mathbf{x}_k\}_{k=1}^{p}$, $\mathbf{x}_k \in \mathbb{R}^{mn\times 1}_{\ge 0}$ and stacked to form a matrix $\mathbf{M} = [\mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots, \mathbf{x}_p] \in \mathbb{R}^{mn\times p}_{\ge 0}$. Then NMF is applied to the matrix $\mathbf{M}$ with a rank truncation parameter ℓ, so that the factorized matrices are $\mathbf{W} \in \mathbb{R}^{mn\times \ell}_{\ge 0}$ and $\mathbf{H} \in \mathbb{R}^{\ell\times p}_{\ge 0}$. We refer to the column vectors of $\mathbf{W}$ as the (vectorized) matrix basis images, and the column vectors of $\mathbf{H}$ as encodings of the images. Then a given image $\mathbf{x}_k$ can be reconstructed by

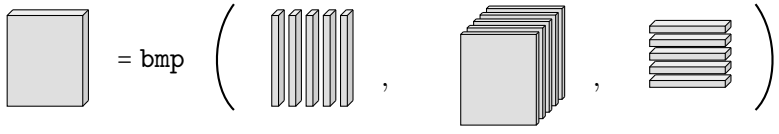
$$\mathbf{x}_k = \mathbf{W}\mathbf{H}_{:,k} = \sum_{t=1}^{\ell} \mathbf{H}_{t,k}\mathbf{W}_{:,t}.$$

In the tensor case, the set of images $\{\mathbf{X}_k\}_{k=1}^{p}$ are stacked as frontal slices of a data

tensor $\mathcal{X} \in \mathbb{R}_{\geq 0}^{m \times n \times p}$ such that $\mathcal{X}_{:,:,k} \coloneqq \mathbf{X}_k$. Then NNBMD is applied to the tensor $\mathcal{X}$ with a rank truncation parameter r. The decomposed tensor factors are $\mathcal{A} \in \mathbb{R}_{\geq 0}^{m \times r \times p}$, $\mathcal{B} \in \mathbb{R}_{\geq 0}^{m \times n \times r}$, and $\mathcal{C} \in \mathbb{R}_{\geq 0}^{r \times n \times p}$. We refer to the frontal slices of $\mathcal{B}$ as the tensor basis images, and the frontal slices of $\mathcal{A}$ and $\mathcal{C}$ as the left and right encodings of the images, respectively. hen a given image $\mathbf{X}_k$ can be reconstructed by

$$\mathbf{X}_k = \texttt{bmp}(\mathcal{A}_{:,:,k}, \mathcal{B}, \mathcal{C}_{:,:,k}) = \sum_{t=1}^{r} \operatorname{diag}(\mathcal{A}_{:,t,k}) \mathcal{B}_{:,:,t} \operatorname{diag}(\mathcal{C}_{t,:,k}).$$

We use a figure illustration (Fig. 6.1) to show the tensor image reconstruction.



$\mathcal{T}_{:,:,k} = \mathbf{X}_k \in \mathbb{R}^{m \times n \times 1}$ Image | $\mathcal{A}_{:,:,k} \in \mathbb{R}^{m \times r \times 1}$ Left encodings | $\mathcal{B} \in \mathbb{R}^{m \times n \times r}$ Basis images | $\mathcal{C} \in \mathbb{R}^{r \times n \times 1}$ Right encodings

Figure 6.1: A single image expressed as a BM-product of the left and right encodings $(\mathcal{A}, \mathcal{C})$ and the factor tensor $\mathcal{B}$ of basis images.

Given a data matrix of size $mn \times p$ (data tensor of size $m \times n \times p$), the storage requirements for the factor matrices $\mathbf{W}$ and $\mathbf{H}$ in the NMF decomposition are given by $(mm + p)\ell$, where ℓ denotes the target matrix truncation rank. In contrast, the storage required for the BM-rank r NNBMD factor tensors $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ is given by $(mn + np + mp)r$. In the compression experiments, we first specify the target matrix truncation rank ℓ and compute the corresponding storage requirement for the NMF factors. We then determine the BM-rank parameter r so that the storage remains equivalent. Specifically, r is set as $r = \lfloor \frac{(mm+p)\ell}{mn+np+mp} \rfloor$, with the constraint that $r \geq 1$. For the NMF method, we use the default MATLAB function NMF. For the NNBMD method, we takes the NMF results and set the initial guesses $\mathcal{A}^{(0)} \coloneqq \texttt{ones}(m, r, p)$, $\mathcal{B}^{(0)}_{:,:,k} \coloneqq \texttt{reshape}(\mathbf{W}_{:,k}, m, n)$, $1 \leq k \leq p$, and $\mathcal{C}^{(0)}_{:,j,:} = \mathbf{H}$, $1 \leq j \leq n$. The total number of iterations is set to be 100 with a tolerance parameter of $1e-5$, i.e., the iterative multiplicative updates algorithm terminates when the relative error is less than 10^{-5}.

We selected six images for display: image numbers 41, 95, 104, 109, and 123. To

evaluate the performance of different methods, we tested two distinct compression levels. In the first experiment, we utilized 30 matrix basis images, which are the column vectors of the factor matrix $\mathbf{W}$ from NMF of the images and are reshaped into the size of the given images. Correspondingly, we used 9 tensor basis images, which are the frontal slices of the factor tensor $\mathcal{B}$ in the NNBMD. Figure (6.2) presents the original brain MRI images in the first row, the reconstructed images by the NMF method in the second row, and the reconstructed images by the NNBMD method in the third row in Fig. (6.2). Compared to the NMF reconstructions, the images reconstructed using NNBMD exhibit superior quality with enhanced clarity and more distinguishable details. The nine tensor basis images are depicted in Figure (6.3), while the 30 matrix basis images are shown in Figure (6.4). In comparison to the matrix basis images, the tensor basis images exhibit lower variability and capture a reduced number of distinct skull shapes. However, they provide a parts-based, lower-dimensional representation of the data, effectively preserving essential structural components.

For the second experiment, we applied 10 matrix basis images and 3 tensor basis images. Figure (6.5) presents the original images in the first row, the NMF reconstructed images in the second row, and the NNBMD reconstructed images in the last row. Consistent with the findings from the first experiment, the tensor-based reconstruction exhibits superior image quality. The three tensor basis images are displayed in Figure (6.6), while the ten matrix basis images are shown in Figure (6.7).

6.3.2 Application for facial recognition

Non-negative matrix factorization is shown to be effective for facial recognition tasks in computer vision [107, 108]. In the early work by Lee and Seung [58], the authors applied the NMF method to decompose face images into localized components such as eyes, noses, or mouths. These parts-based representations capture the distinctiveness of individual faces, making NMF suitable for identifying individuals. Additionally, the non-negativity constraint ensures that the reconstruction remains physically meaningful, as pixel values are non-negative.

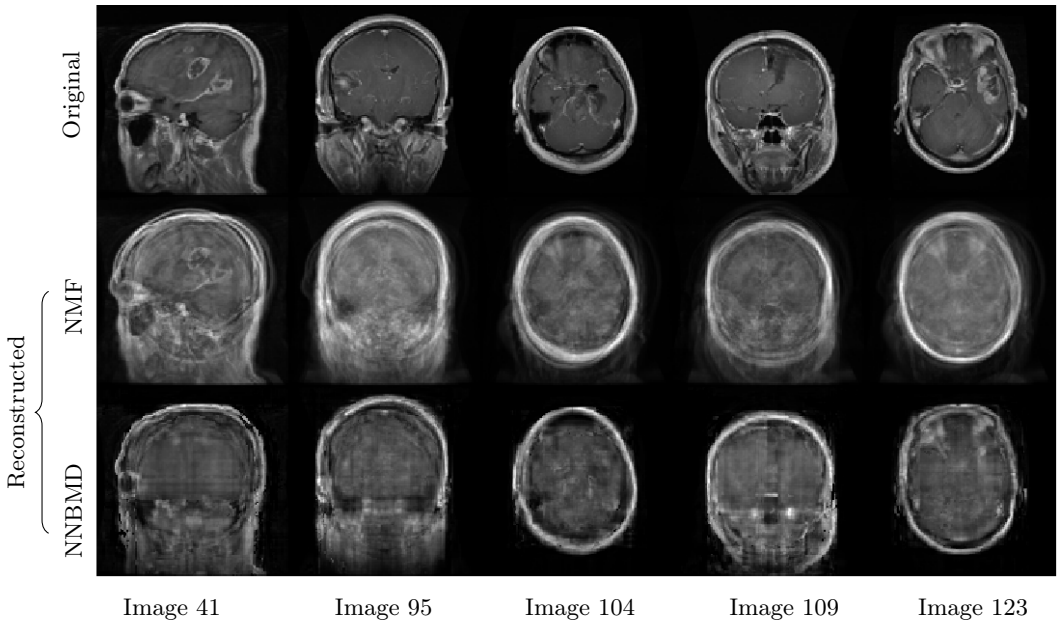


Figure 6.2: Experiment 1: Selected brain MRI images and reconstructed results. Top row: original images. Second row: NMF reconstructed images. Third row: NNBMD reconstructed images

Extending this concept to higher-order structures, non-negative CP and Tucker tensor factorization have also proven to be useful for face detection and has demonstrated high effectiveness in classification tasks [37, 51]. In this section, we examine the application of our proposed NNBMD algorithm for facial recognition and compare its performance with the NMF-based classification algorithm. We first outline the training and testing strategies described in [81] for facial recognition using NMF. This method is both simple and effective, and it can be naturally extended to the third-order tensor case.

6.3.2.1 Training and testing with NMF

Given p training face images $\{\mathbf{F}_j\}_{j=1}^{p}$ with each image $\mathbf{F}_j$ is of size $m \times n$, construct a training data matrix $\mathbf{M} \in \mathbb{R}_{\geq 0}^{mn \times p}$ whose columns are the vectorized images $\mathbf{M} := [\mathbf{f}_1\, \mathbf{f}_2 \cdots \mathbf{f}_p]$, where $\mathbf{f}_j = \mathtt{vec}(\mathbf{F}_j)$. Decompose the matrix $\mathbf{M}$ into two matrices $\mathbf{H} \in \mathbb{R}_{\geq 0}^{mn \times \ell}$ and $\mathbf{W} \in \mathbb{R}_{\geq 0}^{\ell \times p}$ using the algorithms given in Eq. (6.3). Denote ℓ to be the matrix rank truncation parameter. Let the basis images be $\mathbf{W} = [\mathbf{w}_1\, \mathbf{w}_2 \cdots \mathbf{w}_\ell]$, and let the encodings be $\mathbf{H} = [\mathbf{h}_1\, \mathbf{h}_2 \cdots \mathbf{h}_p]$. Then we can approximate each vectorized

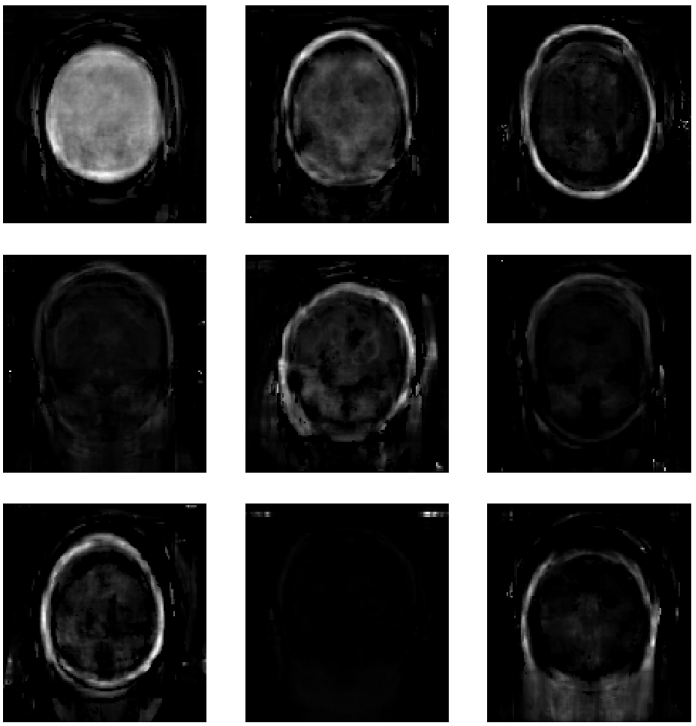


Figure 6.3: Experiment 1: 9 tensor basis images

face image $\mathbf{f}_j$ as a linear combination of the basis images with the corresponding encoding coefficients $\mathbf{h}_j = [h_{1j}\, h_{2j} \cdots h_{\ell j}]^\top$ such that

$$\mathbf{f}_j = \sum_{1 \le t \le \ell} \mathbf{w}_t h_{tj} = \mathbf{W}\mathbf{h}_j.$$

For each face $\mathbf{f}_j \in \mathbb{R}_{\ge 0}^{mn \times 1}$ in the training set, there is a corresponding encoding coefficient $\mathbf{h}_j \in \mathbb{R}_{\ge 0}^{\ell \times 1}$ with the reduced dimension ℓ.

In the testing phase, for a given new image denoted $\mathbf{y} \in \mathbb{R}_{\ge 0}^{mn \times 1}$, we can find a representative encoding $\mathbf{h}_y \in \mathbb{R}_{\ge 0}^{\ell \times 1}$ for $\mathbf{y}$ by computing

$$\mathbf{h}_y = \mathbf{W}^{-1}\mathbf{y}.$$

The cosine of the angle is then used to measure the similarity score, $s_j \in \mathbb{R}$, between

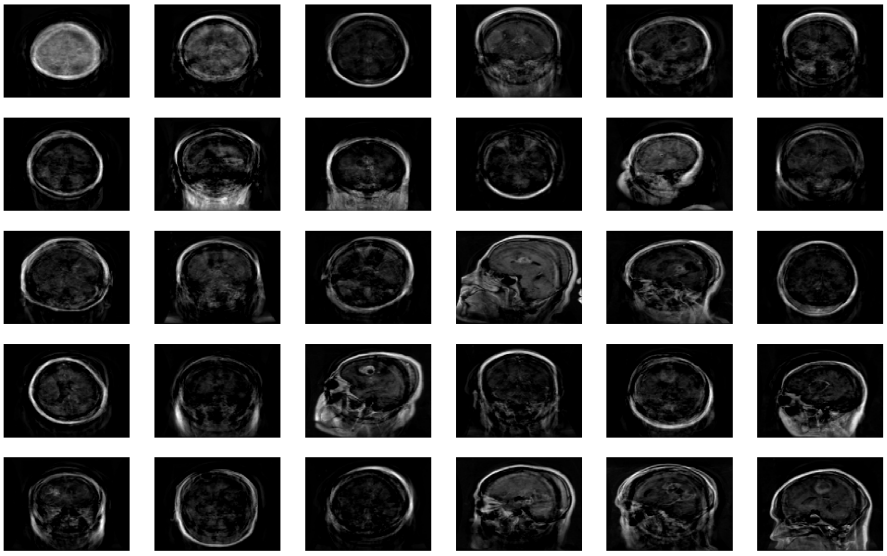


Figure 6.4: Experiment 1: 30 matrix basis images

the encodings of a trained image $\mathbf{h}_j$ and a test image $\mathbf{h}_y$, i.e.

$$s_j := \frac{\mathbf{h}_y^\top \mathbf{h}_j}{\|\mathbf{h}_y\|_2 \|\mathbf{h}_j\|_2}$$

where $\|\cdot\|_2$ denotes the 2-norm of a vector. The similarity measure s_j determines the matching score between the test face image $\mathbf{y}$ and the j-th training image $\mathbf{f}_j$. The optimal matching is determined by the maximum score denoted j^* where $j^* := \max_j \{s_1, \ldots, s_j, \ldots, s_p\}$.

6.3.2.2 Training and testing with NNBMD

Next, we will generalize the NMF training and testing approaches to the third-order tensor-based facial recognition algorithm. In particular, during the training stage, we apply the tensor non-negative BM-decomposition (NNBMD) to the set of face images $\{\mathbf{F}_k\}_{k=1}^p, \mathbf{F}_k \in \mathbb{R}_{\geq 0}^{m \times n}$ arranged as frontal slices of a data tensor $\mathcal{T} \in \mathbb{R}_{\geq 0}^{m \times n \times p}$, i.e. $\mathcal{T}_{:,:,k} = \mathbf{F}_k$. The NNBMD method is applied to the tensor $\mathcal{T}$ resulting in the BM-rank r decomposed factor tensor triple $(\mathcal{A}, \mathcal{B}, \mathcal{C}) \in \mathbb{R}_{\geq 0}^{m \times r \times p} \times \mathbb{R}_{\geq 0}^{m \times n \times r} \times \mathbb{R}_{\geq 0}^{r \times n \times p}$. We take the factor tensor $\mathcal{B}$ as the basis tensor whose frontal slices are the basis images. Let the left-encodings and the right-encodings be the factor tensor $\mathcal{A}$ and $\mathcal{C}$

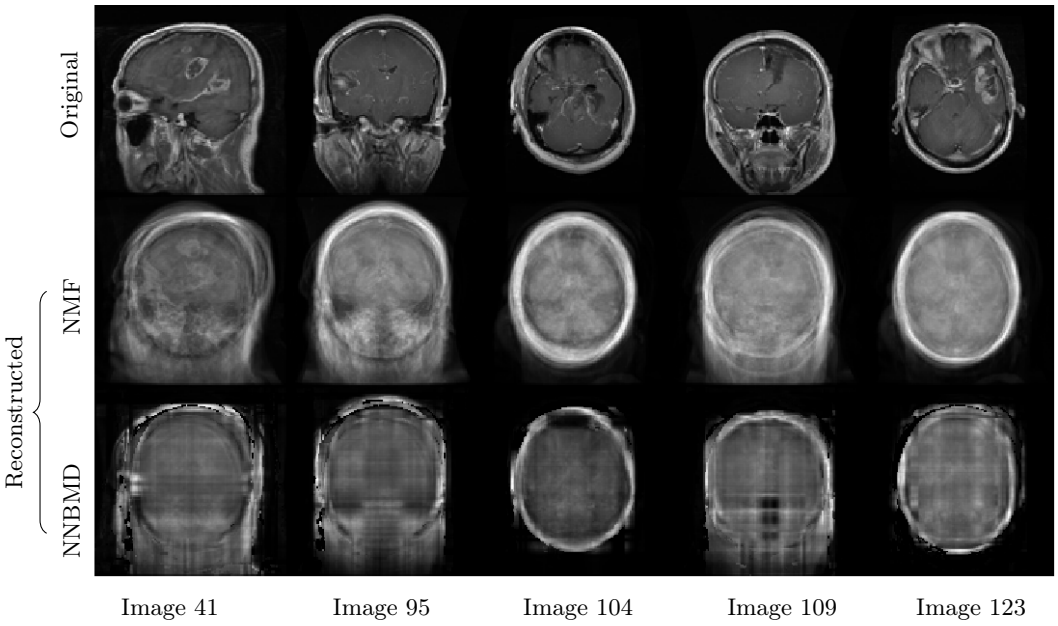


Figure 6.5: Experiment 2: Selected brain MRI images and reconstructed results. Top row: original images. Second row: NMF reconstructed images. Third row: NNBMD reconstructed images

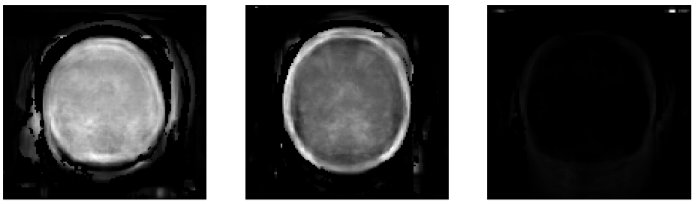


Figure 6.6: Experiment 2: 3 tensor basis images

respectively. Then for each face image $\mathbf{F}_k$, it can be expressed as a diagonal left-right linear combination of the basis images $\mathcal{B}_{:,:,t}, 1 \le t \le r$ with the corresponding left-encoding coefficients $\mathcal{A}_{:,:,k} = \left[\mathcal{A}_{:,1,k}, \mathcal{A}_{:,2,k}, \ldots, \mathcal{A}_{:,r,k}\right]$ and right-encoding coefficients $\mathcal{C}_{:,:,k} = \begin{bmatrix} \mathcal{C}_{1,:,k} \\ \mathcal{C}_{2,:,k} \\ \ldots \\ \mathcal{C}_{:,r,k} \end{bmatrix}$. In particular, we express $\mathbf{F}_k$ as follows,

$$\mathcal{T}_{:,:,k} = \mathbf{F}_k = \sum_{t=1}^{r} \operatorname{diag}\left(\mathcal{A}_{:,t,k}\right) \ \mathcal{B}_{:,:,t} \ \operatorname{diag}\left(\mathcal{C}_{t,:,k}\right). \tag{6.11}$$

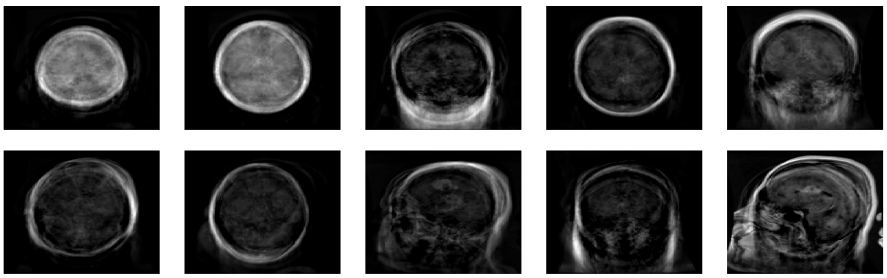


Figure 6.7: Experiment 2: 10 matrix basis images

During the testing stage, when a new image $\mathcal{Y} \in \mathbb{R}_{\geq 0}^{m \times n \times 1}$ is given, we compute the left and the right encodings $(\mathcal{A}_y, \mathcal{C}_y) \in \mathbb{R}_{\geq 0}^{m \times r \times 1} \times \mathbb{R}_{\geq 0}^{r \times n \times 1}$ by solving

$$\min_{\mathcal{A}_y, \mathcal{C}_y} \left\| \mathcal{Y} - \texttt{bmp}\left(\mathcal{A}_y, \mathcal{B}, \mathcal{C}_y\right) \right\|_F . \tag{6.12}$$

The solution to this minimization problem is given by the multiplicative update algorithm from Eq. (6.9) and Eq. (6.10) for updating $\mathcal{A}_y$ and then $\mathcal{C}_y$ respectively.

Once, we obtain the encoding pair $(\mathcal{A}_y, \mathcal{C}_y)$ corresponding to the image $\mathcal{Y}$, we compute the following similarity scores

$$s_a = \frac{(\mathbf{h}_a^y)^\top \mathbf{h}_a^k}{\|\mathbf{h}_a^y\|_2 \|\mathbf{h}_a^k\|_2}, \quad s_c = \frac{(\mathbf{h}_c^y)^\top \mathbf{h}_c^k}{\|\mathbf{h}_c^y\|_2 \|\mathbf{h}_c^k\|_2}, \forall\ 1 \leq k \leq p, \tag{6.13}$$

where $\mathbf{h}_a^y = \texttt{vec}(\mathcal{A}_y) \in \mathbb{R}_{\geq 0}^{mr \times 1}$, $\mathbf{h}_a^k = \texttt{vec}(\mathcal{A}_{:,:,k}) \in \mathbb{R}_{\geq 0}^{mr \times 1}$, $\mathbf{h}_c^y = \texttt{vec}(\mathcal{C}_y) \in \mathbb{R}_{\geq 0}^{rn \times 1}$, $\mathbf{h}_c^k = \texttt{vec}(\mathcal{C}_{:,:,k}) \in \mathbb{R}_{\geq 0}^{rn \times 1}$, and $(\mathcal{A}_{:,:,k}, \mathcal{C}_{:,:,k})$ are the encodings corresponding to the k-th training image $\mathbf{F}_k$. Next, we compute a total score $s^k = s_a^k + s_c^k$. After determining the maximum score $s^{k^*} = \max_k \{s^1, \ldots, s^k, \ldots, s^p\}$ and the corresponding index k^*, we say that the new image $\mathcal{Y}$ belongs to the same image class label as the training image $\mathbf{F}_{k^*}$.

6.3.2.3 Experimental results and discussion

In this section, we assess the performance of the proposed NNBMD algorithm for facial recognition task on the AT&T Database of Faces (formerly known as the

ORL Database of Faces) [85] *. This dataset comprises grayscale facial images of 40 distinct individuals, with 10 images per subject. Each image is of size 112×92. The images exhibit variability in terms of lighting conditions. Additionally, variations in facial expressions (e.g., open or closed eyes, smiling or neutral expressions) and facial details (e.g., with or without glasses) are present across subjects. All images were captured against a dark, uniform background, with subjects positioned in an upright and frontal pose [16].

The 400 facial images are organized as frontal slices of a third-order nonnegative data tensor, denoted by $\mathcal{T} \in \mathbb{R}_{\geq 0}^{112 \times 92 \times 400}$. This tensor is subsequently partitioned into a training set and a test set by randomly assigning 60% of the images to training and the remaining 40% to testing, while ensuring that each subject is represented in both subsets. The resulting training tensor $\mathcal{T}_{\text{train}}$ has dimensions $112 \times 92 \times 240$, and the test tensor $\mathcal{T}_{\text{test}}$ has dimensions $112 \times 92 \times 160$.

For experiments involving the NMF approach, each image is vectorized into a column vector of size 10304×1, leading to the formation of a training matrix $\mathbf{T}_{\text{train}} \in \mathbb{R}_{\geq 0}^{10304 \times 240}$ and a test matrix $\mathbf{T}_{\text{test}} \in \mathbb{R}_{\geq 0}^{10304 \times 160}$. NMF is then applied to the training matrix to yield a nonnegative basis matrix $\mathbf{W} \in \mathbb{R}_{\geq 0}^{10304 \times \ell}$ with ℓ basis images, and a corresponding encoding matrix $\mathbf{H} \in \mathbb{R}_{\geq 0}^{\ell \times 240}$.

In contrast, for the NNBMD method, factorization is performed directly on the tensor $\mathcal{T}_{\text{train}}$, resulting in a factor tensor of basis images $\mathcal{B} \in \mathbb{R}_{\geq 0}^{112 \times 92 \times r}$ with r basis components. The corresponding encodings are captured by the left encoding tensor $\mathcal{A} \in \mathbb{R}_{\geq 0}^{112 \times r \times 240}$ and the right encoding tensor $\mathcal{C} \in \mathbb{R}_{\geq 0}^{r \times 92 \times 240}$.

To facilitate a meaningful comparison between the NMF and NNBMD approaches, we ensure parity in storage requirements. Specifically, for the NMF method, the total storage demand is quantified as $\ell(10304 + 240)$, accounting for both the basis images and encoding matrices. To match this storage level for the NNBMD method,

*The AT&T Database of Faces is publicly available on Kaggle https://www.kaggle.com/datasets/kasikrit/att-database-of-faces.

we compute the number of tensor basis images r via

$$r = \left\lfloor \frac{\ell(10304 + 240)}{112 \times 92 + 112 \times 240 + 92 \times 240} \right\rfloor,$$

given that the total storage for NNBMD is $r(112 \times 92 + 112 \times 240 + 92 \times 240)$.

Model performance is evaluated using classification accuracy, defined as:

$$\text{accuracy} = \frac{\text{number of correctly classified test images}}{\text{total number of test images}} \times 100.$$

This metric provides a direct assessment of the representation quality and discriminative power of the learned basis components under each factorization scheme.

The experiment results are shown in Fig. (6.8), where storage usage across methods remains comparable. From the left-hand side of the plots, we observe that the NNBMD method achieves higher classification accuracy than NMF when the number of basis components is small. However, as the number of matrix components increases, the performance of the NMF method improves steadily, whereas the accuracy of the NNBMD method exhibits greater variability and does not follow a clear increasing trend.

In Fig. 6.9, we present six representative test images alongside their reconstructions produced by the NMF method with 24 matrix components and the NNBMD method with 4 tensor components. As illustrated in the figure, the reconstructions generated by the NNBMD approach demonstrate superior visual quality, particularly in preserving fine facial features such as the presence of glasses or facial hair, when compared to those produced by the NMF method. Despite the enhanced visual fidelity of the NNBMD reconstructions, the NMF method achieves a higher classification accuracy of 88.125%, compared to 81.875% obtained by the NNBMD approach. This discrepancy suggests that the matching procedure employed during the testing phase for the NNBMD method may not effectively identify the most accurate correspondences. One potential contributing factor is the inherent non-uniqueness of the tensor BM-decomposition. In particular, the independent scaling of the factor

tensors $\mathcal{A}$ and $\mathcal{C}$ may distort the similarity scores s_a and s_c, thereby compromising the matching process. To address this issue, incorporating a regularization scheme into the nonnegative BM-decomposition may help resolve the non-uniqueness and improve matching accuracy. We will leave this as our future direction to enhance the effectiveness of NNBMD-based facial recognition systems.

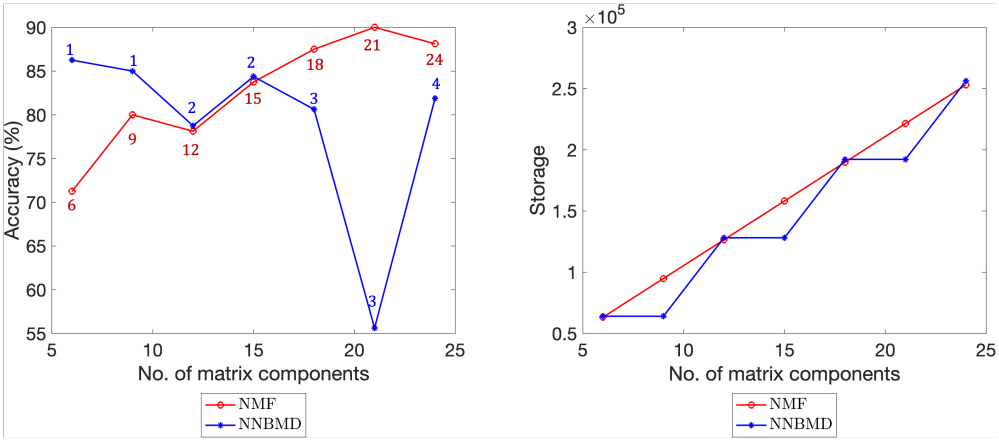


Figure 6.8: Plot of accuracy vs. storage for the NMF and NNBMD methods. The number of NMF basis components is labeled in blue, and the number of NNBMD basis components is labeled in red.

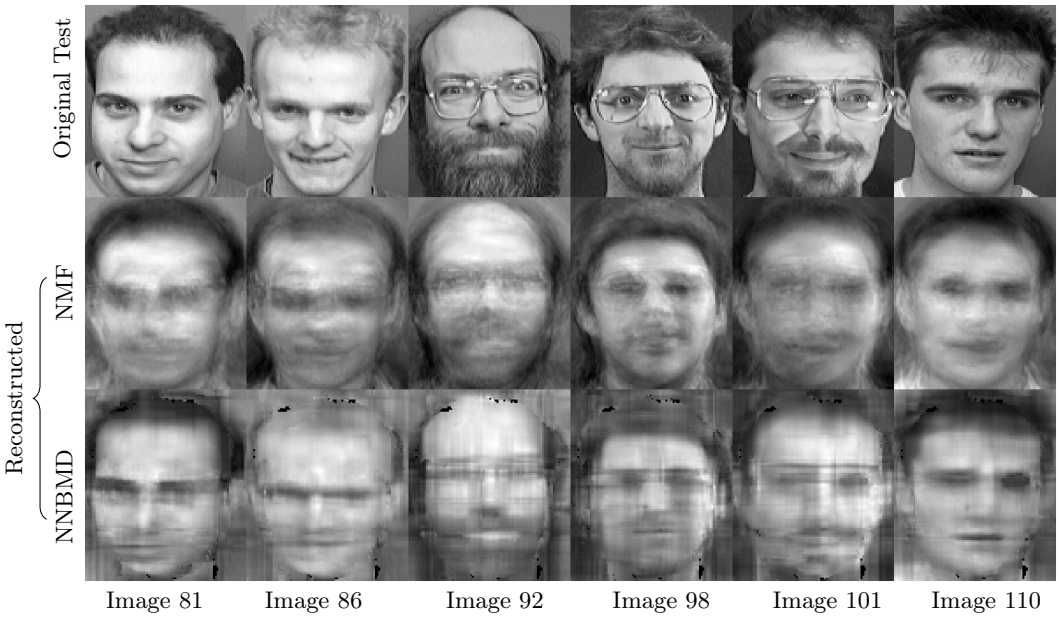


Figure 6.9: Selected test images and reconstructed results. NMF Storage: 253056 (24 matrix components), NNBMD Storage: 256256 (4 tensor components). NMF Accuracy: 88.125%, NNBMD Accuracy: 81.875%

Chapter 7

Streaming update

Tensor decomposition is widely used for analyzing multi-way data in various applications, including signal processing [13], social networks [83], and machine learning [86]. These fields often involve data that is continuously generated and updated over time, requiring efficient methods to process and extract meaningful patterns dynamically. In some applications, tensor data may also evolve in multiple dimensions/modes over time as illustrated in Fig. (7.1). However, applying traditional tensor decomposition algorithms to such streaming data presents two major challenges [89]. First, the complete dataset is not available in advance, making it difficult to apply standard batch-processing techniques. Second, as new data continuously accumulates, the size of the dataset grows linearly over time, leading to increasing computational and storage demands. These challenges necessitate the development of specialized *streaming* tensor decomposition methods that can efficiently process data in an incremental manner while maintaining accuracy and scalability. Over the past few decades, numerous studies have been conducted to tackle these challenges in various tensor decomposition types, including the CP decomposition [87, 113], HOSVD/-Tucker decomposition [101, 103], t-SVD [112], and block-term decomposition [84]. A comprehensive review of these different streaming tensor decomposition methods is available in [1].

In this chapter, we consider the problem of computing tensor BM-decompositions (BMD) in the streaming setting. An incremental algorithm, **OnlineBMD** is presented, which is the first work that addresses third-order tensor BMD in a dynamic setting.

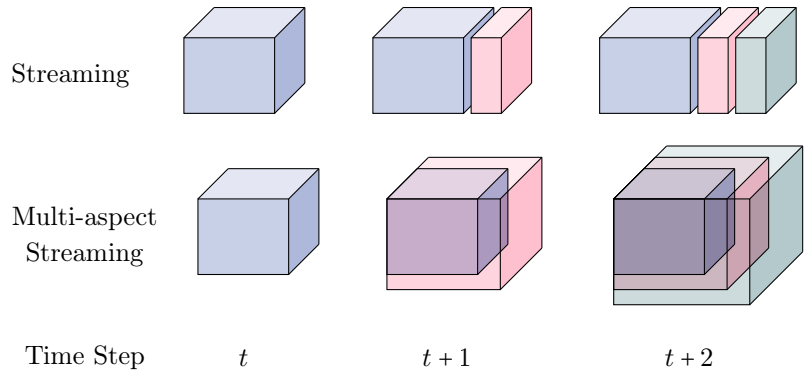


Figure 7.1: Streaming and multi-aspect streaming data tensor over time.

7.1 Streaming update algorithm

In this section, we will discuss the streaming data BM-decomposition update of a third-order online tensor $\mathcal{T} \in \mathbb{R}^{m\times(p+d)\times n}$ as illustrated in Fig. 7.2.

Given a set of data matrices, each of size $m \times n$, we order them as lateral slices from left to right to from an order-3 tensor $\mathcal{T}_{\text{old}} \in \mathbb{R}^{m\times p\times n}$. A new batch of d matrix slices, denoted $\mathcal{T}_{\text{new}} \in \mathbb{R}^{m\times d\times n}$, is added to $\mathcal{T}_{\text{old}}$ as lateral slices upon arrival. We use $\mathcal{T} = \texttt{cat}(2, \mathcal{T}_{\text{old}}, \mathcal{T}_{\text{new}}) \in \mathbb{R}^{m\times(p+d)\times n}$ to denote the new data tensor consisting of both $\mathcal{T}_{\text{old}}$ and $\mathcal{T}_{\text{new}}$. A figure illustration is shown in Fig. 7.2.

The BM-rank ℓ decomposition of the historical data tensor $\mathcal{T}_{\text{old}}$ is known. The factor tensors of $\mathcal{T}_{\text{old}}$ are denoted as $\mathcal{A}_{\text{old}} \in \mathbb{R}^{m\times\ell\times n}$, $\mathcal{B}_{\text{old}} \in \mathbb{R}^{m\times p\times\ell}$, and $\mathcal{C}_{\text{old}} \in \mathbb{R}^{\ell\times p\times n}$ such that $\mathcal{T}_{\text{old}} \approx \texttt{bmp}\,(\mathcal{A}_{\text{old}}, \mathcal{B}_{\text{old}}, \mathcal{C}_{\text{old}})$. With the new slice $\mathcal{T}_{\text{new}}$ added to $\mathcal{T}$, our goal is to efficiently update the BMD factor tensors $\mathcal{A} \in \mathbb{R}^{m\times\ell\times n}$, $\mathcal{B} \in \mathbb{R}^{m\times(p+d)\times\ell}$, and $\mathcal{C} \in \mathbb{R}^{\ell\times(p+d)\times n}$ of $\mathcal{T}$ leveraging existing factors of $\mathcal{T}_{\text{old}}$. In this section, we will discuss two approaches that can effectively compute the streaming BMD without the need to store the historical tensor $\mathcal{T}_{\text{old}}$.

7.1.1 An SVD update approach – sliceUpdate

In [93], the authors discussed the connection between rank-revealing matrix factorizations and the third-order tensor BMD. In particular, slice-wise based decomposition can be directly expressed as a tensor BMD.

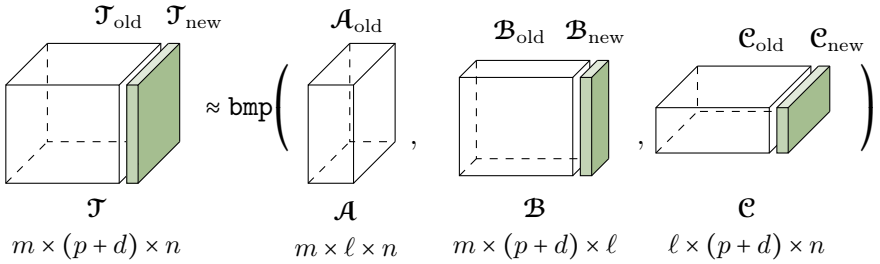


Figure 7.2: Streaming tensor BMD

Suppose $\mathcal{T} \in \mathbb{R}^{m \times p \times n}$ has the following slice-wise matrix decomposition

$$\mathcal{T}_{:,:,k} = \mathbf{V}^{(k)} \mathbf{W}^{(k)}, \quad \forall\, 1 \le k \le n,$$

where $\mathbf{V}^{(k)} \in \mathbb{R}^{m \times \ell}$ and $\mathbf{W}^{(k)} \in \mathbb{R}^{\ell \times p}$. Then it can be represented by a tensor BM-decomposition of $\mathcal{T}$ as follows: $\forall\, 1 \le k \le n$, let $\mathcal{A}_{:,:,k} = \mathbf{V}^{(k)}$, $\mathcal{C}_{:,:,k} = \mathbf{W}^{(k)}$, and let $\mathcal{B} = \texttt{ones}([m, p, \ell])$. Then $\mathcal{T} = \texttt{bmp}(\mathcal{A}, \mathcal{B}, \mathcal{C})$ with BM-rank at most ℓ.

Moreover, we can take the truncated rank ℓ matrix SVDs of the frontal slices of $\mathcal{T}$:

$$\mathcal{T}_{:,:,k} = \mathbf{U}_\ell^{(k)} \mathbf{\Sigma}_\ell^{(k)} \left(\mathbf{V}_\ell^{(k)} \right)^\top.$$

We then set

$$\mathcal{A}_{:,:,k} = \mathbf{U}_\ell^{(k)} \mathbf{\Sigma}_\ell^{(k)}; \ \mathcal{C}_{:,:,k} = \left(\mathbf{V}_\ell^{(k)} \right)^\top.$$

Instead of letting $\mathcal{B}$ to be an all-one tensor, we solve for $\mathcal{B}$ that minimizes $\|\mathcal{T} - \texttt{bmp}(\mathcal{A}, \mathcal{B}, \mathcal{C})\|_F^2$.

The alternating least-squares (ALS) algorithm for computing the BMD of $\mathcal{T}$ takes the factor tensor triplet $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ and updates each of the factors sequentially.

Based on the matrix SVD nature of the slices, we can update $\mathcal{A}$ and $\mathcal{C}$ borrowing techniques of the SVD updating method. For this approach, we leverage the projection technique for updating the truncated SVD algorithm discussed in [42].

For each frontal slice $\mathbf{T}_{\text{old}}(:,:,k)$ with rank-ℓ truncated SVD such that

$$\mathbf{T}_{\text{old}}(:,:,k) = \mathbf{U}_\ell^{(k)} \mathbf{\Sigma}_\ell^{(k)} \left(\mathbf{V}_\ell^{(k)} \right)^\top,$$

where $\mathbf{U}_\ell^{(k)} \in \mathbb{R}^{m \times \ell}$, $\mathbf{\Sigma}_\ell^{(k)} \in \mathbb{R}^{\ell \times \ell}$, and $\left(\mathbf{V}_\ell^{(k)} \right)^\top \in \mathbb{R}^{\ell \times p}$, we compute the projection

matrices $\mathbf{Z}^{(k)} = \begin{pmatrix} \mathbf{V}_\ell^{(k)} & \\ & \mathbf{I}_{d\times d} \end{pmatrix} \in \mathbb{R}^{(p+d)\times(\ell+d)}$. Additionally, we let $\mathbf{A}^{(k)}$ to denote the frontal slice of the full data tensor $\mathcal{T}$, thus $\mathbf{A}^{(k)} = \left[\begin{array}{c|c} \mathcal{T}_{\text{old}}(:,:,k) & \mathcal{T}_{\text{new}}(:,:,k) \end{array}\right] \in \mathbb{R}^{m\times(p+d)}$. We then apply the matrix SVD update algorithm 2 to each of the slices.

Algorithm 2 RR-SVD, "$\mathbf{A}\mathbf{A}^\top$ version" [42]

1: **Input:** $\mathbf{U}_\ell^{(k)}, \mathbf{\Sigma}_\ell^{(k)}, \mathbf{V}_\ell^{(k)}, \mathbf{A}^{(k)}, \mathbf{Z}^{(k)}$
2: **Output:** $\overline{\mathbf{U}}_\ell^{(k)}, \overline{\mathbf{\Sigma}}_\ell^{(k)}, \overline{\mathbf{V}}_\ell^{(k)}$
3: Solve $[\mathbf{G}_\ell^{(k)}, \mathbf{\Theta}_\ell^{(k)}, \sim] = \texttt{svd}_\ell\left(\left(\mathbf{Z}^{(k)}\right)^\top \left(\mathbf{A}^{(k)}\right)^\top\right)$
4: Set $\overline{\mathbf{V}}_\ell^{(k)} = \mathbf{Z}^{(k)}\mathbf{G}_\ell^{(k)}$ and $\overline{\mathbf{\Sigma}}_\ell^{(k)} = \mathbf{\Theta}_\ell^{(k)}$
5: Set $\overline{\mathbf{U}}_\ell^{(k)} = \mathbf{A}\overline{\mathbf{V}}_\ell^{(k)}\left(\overline{\mathbf{\Sigma}}_\ell^{(k)}\right)^{-1}$

As a result, we have the updated rank-ℓ truncated SVD of $\mathbf{A}^{(k)} = \overline{\mathbf{U}}_\ell^{(k)}\overline{\mathbf{\Sigma}}_\ell^{(k)}\left(\overline{\mathbf{V}}_\ell^{(k)}\right)^\top$, where $\overline{\mathbf{U}}_\ell^{(k)} \in \mathbb{R}^{m\times\ell}$, $\overline{\mathbf{\Sigma}}_\ell^{(k)} \in \mathbb{R}^{\ell\times\ell}$, and $\left(\overline{\mathbf{V}}_\ell^{(k)}\right)^\top \in \mathbb{R}^{\ell\times(p+d)}$.

Once the updated slice-wise SVDs are obtained, we can update the BMD factor tensors as follows

$$\mathcal{A}_{:,:,k} = \overline{\mathbf{U}}_\ell^{(k)}\overline{\mathbf{\Sigma}}_\ell^{(k)};\ \mathcal{C}_{:,:,k} = \left(\overline{\mathbf{V}}_\ell^{(k)}\right)^\top.$$

And update the middle tensor $\mathcal{B}$ by minimizing $\|\mathcal{T} - \texttt{bmp}\,(\mathcal{A}, \mathcal{B}, \mathcal{C})\|_F^2$.

7.1.2 An ALS approach – OnlineBMD

In [113], the authors investigated the problem of tracking the CP decomposition of streaming tensors. Their proposed algorithm efficiently updates the decomposition by utilizing complementary matrices to temporarily retain essential information from the previous time step. In this section, we present a related approach that similarly exploits the tensor BM-decomposition from the preceding time step and introduce an efficient alternating update scheme to address the online BMD problem.

As shown in the Fig. 7.2, the tensor $\mathcal{T}$ consists of the existing data $\mathcal{T}_{\text{old}} \in \mathbb{R}^{m\times p\times n}$, and a new data batch $\mathcal{T}_{\text{new}} \in \mathbb{R}^{m\times d\times n}$ appended to $\mathcal{T}_{\text{old}}$ along its depth dimension. We assume that $d \ll p$, as in most online systems, the size of the new incoming data is typically much smaller than the size of the existing historical data. The BM-rank ℓ

decomposition of $\mathcal{T}_{\text{old}}$ is a tensor triplet $(\mathcal{A}_{\text{old}}, \mathcal{B}_{\text{old}}, \mathcal{C}_{\text{old}}) \in \mathbb{R}^{m\times\ell\times n} \times \mathbb{R}^{m\times p\times\ell} \times \mathbb{R}^{\ell\times p\times n}$. Our goal is to find the BMD factors $\mathcal{A} \in \mathbb{R}^{m\times\ell\times n}$, $\mathcal{B} \in \mathbb{R}^{m\times(p+d)\times\ell}$ and $\mathcal{C} \in \mathbb{R}^{\ell\times(p+d)\times n}$ of $\mathcal{T}$.

We propose an alternating update approach as follows. First, we fix $\mathcal{A}, \mathcal{B}$ and update $\mathcal{C}$. Then, we fix $\mathcal{A}, \mathcal{C}$ and update $\mathcal{B}$. Lastly, we fix $\mathcal{B}, \mathcal{C}$ and update $\mathcal{A}$. In particular, with the new data batch $\mathcal{T}_{\text{new}}$ arrives, we update the factor tensors $\mathcal{B}$ and $\mathcal{C}$ along the streaming dimension first. We update the factor tensor $\mathcal{A}$ last.

7.1.2.1 Update the factors along the streaming dimension

We show that updating $\mathcal{B}$ and $\mathcal{C}$ can be partitioned into updating the components $(\mathcal{B}_{\text{old}}, \mathcal{B}_{\text{new}})$ of $\mathcal{B}$ and $(\mathcal{C}_{\text{old}}, \mathcal{C}_{\text{new}})$ of $\mathcal{C}$ as shown in Fig. 7.2.

By the definition of third-order tensor BMP, if $\mathcal{T} \approx \texttt{bmp}(\mathcal{A}, \mathcal{B}, \mathcal{C})$, then each of the k-th frontal slice of $\mathcal{T}$ can be written as

$$\mathcal{T}_{:,:,k} \approx \sum_t \operatorname{diag}\left(\mathcal{A}_{:,t,k}\right) \mathcal{B}_{:,:,t} \operatorname{diag}\left(\mathcal{C}_{t,:,k}\right), \quad \forall\, 1 \le k \le n.$$

This suggests that

$$\begin{aligned}
&\left[\ \mathcal{T}_{\text{old}}(:,:,k) \ \middle|\ \mathcal{T}_{\text{new}(:,:,k)}\ \right] \\
&\approx \sum_{t=1}^{\ell} \operatorname{diag}\left(\mathcal{A}_{\text{old}}(:,t,k)\right) \left[\ \mathcal{B}_{\text{old}}(:,:,t) \ \middle|\ \mathcal{C}_{\text{new}}(:,:,t)\ \right] \operatorname{diag}\left(\left[\begin{array}{c} \mathcal{C}_{\text{old}}(t,:,k) \\ \hline \mathcal{C}_{\text{new}}(t,:,k) \end{array}\right]\right) \\
&= \sum_{t=1}^{\ell} \left[\ \operatorname{diag}\left(\mathcal{A}_{\text{old}}(:,t,k)\right) \mathcal{B}_{\text{old}}(:,:,t) \operatorname{diag}\left(\mathcal{C}_{\text{old}}(t,:,k)\right) \ \middle|\ \operatorname{diag}\left(\mathcal{A}_{\text{old}}(:,t,k)\right) \mathcal{B}_{\text{new}}(:,:,t) \operatorname{diag}\left(\mathcal{C}_{\text{new}}(t,:,k)\right)\ \right].
\end{aligned}$$

Equivalently, we have

$$\mathcal{T}_{\text{old}} \approx \texttt{bmp}\left(\mathcal{A}_{\text{old}}, \mathcal{B}_{\text{old}}, \mathcal{C}_{\text{old}}\right), \ \mathcal{T}_{\text{new}} \approx \texttt{bmp}\left(\mathcal{A}_{\text{old}}, \mathcal{B}_{\text{new}}, \mathcal{C}_{\text{new}}\right).$$

Then updating $\mathcal{B}$ by solving

$$\arg\min_{\mathcal{B}} \left\| \mathcal{T} - \texttt{bmp}\left(\mathcal{A}, \mathcal{B}, \mathcal{C}\right) \right\|_F^2$$

is equivalent to solving

$$\underset{\mathcal{B}_{\text{old}},\mathcal{B}_{\text{new}}}{\arg\min} \left\| [\mathcal{T}_{\text{old}} - \texttt{bmp}\left(\mathcal{A}_{\text{old}}, \mathcal{B}_{\text{old}}, \mathcal{C}_{\text{old}}\right)] + [\mathcal{T}_{\text{new}} - \texttt{bmp}\left(\mathcal{A}_{\text{old}}, \mathcal{B}_{\text{new}}, \mathcal{C}_{\text{new}}\right)] \right\|_F^2$$

Since $\mathcal{A}_{\text{old}}$ and $\mathcal{C}_{\text{old}}$ are fixed, it is clear that the first term of the sum in the Frobenius norm is minimized by $\mathcal{B}_{\text{old}}$. The second term is minimized by solving

$$\mathcal{B}_{\text{new}} = \underset{\mathcal{B}_{\text{new}}}{\arg\min} \left\| \mathcal{T}_{\text{new}} - \texttt{bmp}\left(\mathcal{A}_{\text{old}}, \mathcal{B}_{\text{new}}, \mathcal{C}_{\text{new}}\right) \right\|_F^2,$$

which can be rewritten in the flattened notation as

$$\mathbf{x}_{\mathcal{B}_{\text{new}}} = \underset{\mathbf{x}_{\mathcal{B}_{\text{new}}}}{\arg\min} \left\| \mathbf{y}_{\mathcal{T}_{\text{new}}} - \mathbf{H}_{\mathcal{A}_{\text{old}},\mathcal{C}_{\text{new}}} \mathbf{x}_{\mathcal{B}_{\text{new}}} \right\|_F^2,$$

Overall, we update $\mathcal{B}$ by solving for $\mathcal{B}_{\text{new}}$ and append $\mathcal{B}_{\text{new}}$ to $\mathcal{B}_{\text{old}}$, i.e. $\mathcal{B} \approx \texttt{cat}\left(2, \mathcal{B}_{\text{old}}, \mathcal{B}_{\text{new}}\right)$.

Similarly, we can update $\mathcal{C}$ by first solving for $\mathcal{C}_{\text{new}}$ from the solution to

$$\mathcal{C}_{\text{new}} = \underset{\mathcal{C}_{\text{new}}}{\arg\min} \left\| \mathcal{T}_{\text{new}} - \texttt{bmp}\left(\mathcal{A}_{\text{old}}, \mathcal{B}_{\text{new}}, \mathcal{C}_{\text{new}}\right) \right\|_F^2.$$

We then append $\mathcal{C}_{\text{new}}$ to $\mathcal{C}_{\text{old}}$ along the lateral dimension which gives us $\mathcal{C} \approx \texttt{cat}\left(2, \mathcal{C}_{\text{old}}, \mathcal{C}_{\text{new}}\right)$.

7.1.2.2 Update the basis factor tensor

By fixing $\mathcal{B}, \mathcal{C}$, we solve the following subproblem to update $\mathcal{A}$:

$$\underset{\mathcal{A}}{\arg\min} \left\| \mathcal{T} - \texttt{bmp}\left(\mathcal{A}, \mathcal{B}, \mathcal{C}\right) \right\|_F^2. \tag{7.1}$$

We use the following flattening strategy

$$\begin{aligned}
\mathbf{H}_{\mathcal{B},\mathcal{C}}((k-1)m(p+d) + (i-1)(p+d) + j, (k-1)m\ell + (i-1)\ell + t) &:= \mathcal{B}(i,j,t)\mathcal{C}(t,j,k), \\
\mathbf{a}((k-1)m\ell + (i-1)\ell + t) &:= \mathcal{A}(i,t,k), \\
\mathbf{y}_{\mathcal{T}}((k-1)m(p+d) + (i-1)(p+d) + j) &:= \mathcal{T}(i,j,k),
\end{aligned}$$

$$\forall\, 1 \le i \le m, 1 \le j \le p, 1 \le k \le n, 1 \le t \le \ell.$$

Then solving the least-squares problem for $\mathcal{A}$ in Eq. 7.1 is equivalent to solving the following linear least-squares problem for $\mathbf{a}$

$$\underset{\mathbf{a}\in\mathbb{R}^{nm\ell\times 1}}{\arg\min} \left\| \mathbf{y}_{\mathcal{T}} - \mathbf{H}_{\mathcal{B},\mathcal{C}}\mathbf{a} \right\|_F^2, \tag{7.2}$$

where $\mathbf{H}_{\mathcal{B},\mathcal{C}} \in \mathbb{R}^{nm(p+d)\times nm\ell}$, $\mathbf{a} \in \mathbb{R}^{nm\ell\times 1}$ and $\mathbf{y}_{\mathcal{T}} \in \mathbb{R}^{nm(p+d)\times 1}$.

With the flattening scheme described above, $\mathbf{H}_{\mathcal{B},\mathcal{C}}$ is block-diagonal with each block matrix $\mathbf{H}_{\mathcal{B},\mathcal{C}}^{(k,i)}$ of size $(p+d)\times\ell$ for all $1 \le k \le n$ and $1 \le i \le m$. Moreover, each block matrix is of the form

$$\mathbf{H}_{\mathcal{B},\mathcal{C}}^{(k,i)} = \left[\begin{array}{c} \mathbf{H}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}}^{(k,i)} \\ \hline \mathbf{H}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}}^{(k,i)} \end{array}\right],$$

where $\mathbf{H}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}}^{(k,i)} \in \mathbb{R}^{p\times\ell}$ and $\mathbf{H}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}}^{(k,i)} \in \mathbb{R}^{d\times\ell}$. Additionally, the corresponding vectors $\mathbf{a}^{(k,i)} \in \mathbb{R}^{\ell\times 1}$ and $\mathbf{y}_{\mathcal{T}}^{(k,i)} \in \mathbb{R}^{(p+d)\times 1}$ are the (k,i)-th row fiber of the tensor $\mathcal{A}$ and $\mathcal{T}$ respectively. In particular, $\mathbf{y}_{\mathcal{T}}^{(k,i)} = \left[\begin{array}{c|c} \mathbf{y}_{\mathcal{T}_{\text{old}}}^{(k,i)} & \mathbf{y}_{\mathcal{T}_{\text{new}}}^{(k,i)} \end{array}\right]$, where $\mathbf{y}_{\mathcal{T}_{\text{old}}}^{(k,i)} \in \mathbb{R}^{p\times 1}$ and $\mathbf{y}_{\mathcal{T}_{\text{new}}}^{(k,i)} \in \mathbb{R}^{d\times 1}$.

Hence, the linear least-squares problem given in Eq. 7.2 can be solved in parallel for all $k = 1,\ldots,n$ and $i = 1,\ldots,m$ by solving the subproblems

$$\underset{\mathbf{a}^{(k,i)}\in\mathbb{R}^{\ell\times 1}}{\arg\min} \left\| \mathbf{y}_{\mathcal{T}}^{(k,i)} - \mathbf{H}_{\mathcal{B},\mathcal{C}}^{(k,i)}\mathbf{a}^{(k,i)} \right\|_F^2. \tag{7.3}$$

Next, we will derive the updating strategy for solving Eq. 7.3.

Let $\mathcal{L}(\mathbf{a}^{(k,i)}) = \left\| \mathbf{y}_{\mathcal{T}}^{(k,i)} - \mathbf{H}_{\mathcal{B},\mathcal{C}}^{(k,i)}\mathbf{a}^{(k,i)} \right\|_F^2$. The least-squares problem obtains a solution when $\frac{\partial\mathcal{L}}{\partial\mathbf{a}^{(k,i)}} = 0$, where

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\mathbf{a}^{(k,i)}} &= \frac{\partial}{\partial\mathbf{a}^{(k,i)}} \left(\mathbf{y}_{\mathcal{T}}^{(k,i)} - \mathbf{H}_{\mathcal{B},\mathcal{C}}^{(k,i)}\mathbf{a}^{(k,i)}\right)^{\top} \left(\mathbf{y}_{\mathcal{T}}^{(k,i)} - \mathbf{H}_{\mathcal{B},\mathcal{C}}^{(k,i)}\mathbf{a}^{(k,i)}\right) \\ &= -2\left(\mathbf{H}_{\mathcal{B},\mathcal{C}}^{(k,i)}\right)^{\top}\mathbf{y}_{\mathcal{T}}^{(k,i)} + 2\left(\mathbf{H}_{\mathcal{B},\mathcal{C}}^{(k,i)}\right)^{\top}\mathbf{H}_{\mathcal{B},\mathcal{C}}^{(k,i)}\mathbf{a}^{(k,i)}. \end{aligned}$$

We notice that the block matrices are composed of the historical data and the new component, thus

$$\begin{aligned}\left(\mathbf{H}^{(k,i)}_{\mathcal{B},\mathcal{C}}\right)^\top \mathbf{H}^{(k,i)}_{\mathcal{B},\mathcal{C}} &= \begin{bmatrix} \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}}\right)^\top & \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}}\right)^\top \end{bmatrix} \begin{bmatrix} \mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}} \\ \mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}} \end{bmatrix} \\ &= \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}}\right)^\top \mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}} + \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}}\right)^\top \mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}},\end{aligned}$$

and

$$\begin{aligned}\left(\mathbf{H}^{(k,i)}_{\mathcal{B},\mathcal{C}}\right)^\top \mathbf{y}^{(k,i)}_{\mathcal{T}} &= \begin{bmatrix} \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}}\right)^\top & \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}}\right)^\top \end{bmatrix} \begin{bmatrix} \mathbf{y}^{(k,i)}_{\mathcal{T}_{\text{old}}} \\ \mathbf{y}^{(k,i)}_{\mathcal{T}_{\text{new}}} \end{bmatrix} \\ &= \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}}\right)^\top \mathbf{y}^{(k,i)}_{\mathcal{T}_{\text{old}}} + \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}}\right)^\top \mathbf{y}^{(k,i)}_{\mathcal{T}_{\text{new}}}.\end{aligned}$$

Let us denote $\tilde{\mathbf{M}}^{(k,i)} := \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}}\right)^\top \mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}}$ and $\mathbf{M}^{(k,i)} := \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}}\right)^\top \mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}}$. Similarly, denote $\tilde{\mathbf{v}}^{(k,i)} := \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}},\mathcal{C}_{\text{old}}}\right)^\top \mathbf{y}^{(k,i)}_{\mathcal{T}_{\text{old}}}$ and $\mathbf{v}^{(k,i)} := \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{new}},\mathcal{C}_{\text{new}}}\right)^\top \mathbf{y}^{(k,i)}_{\mathcal{T}_{\text{new}}}$. The least-squares subproblem solution can be obtained by solving

$$\mathbf{a}^{(k,i)} = \min_{\mathbf{a}^{(k,i)}} \left\| \left(\tilde{\mathbf{M}}^{(k,i)} + \mathbf{M}^{(k,i)}\right)\mathbf{a}^{(k,i)} - \left(\tilde{\mathbf{v}}^{(k,i)} + \mathbf{v}^{(k,i)}\right)\right\|_F^2. \tag{7.4}$$

This means that by keeping a record of the previous data $\tilde{\mathbf{M}}^{(k,i)}$ and $\tilde{\mathbf{v}}^{(k,i)}$, the large computation can be avoided and efficiently updated incrementally.

In summary, our algorithm has the following steps

1. Compute factor tensors $\mathcal{B}_{\text{new}}$ and $\mathcal{C}_{\text{new}}$.

2. Update factor tensor $\mathcal{A}$ by two stages:

   - Initialize helper matrix $\tilde{\mathbf{M}}^{(k,i)}$ and vector $\tilde{\mathbf{v}}^{(k,i)}$ with tensor $\mathcal{T}_{\text{old}}$ and its BMD factors $(\mathcal{A}_{\text{old}}, \mathcal{B}_{\text{old}}, \mathcal{C}_{\text{old}})$ such that

$$\begin{aligned}\tilde{\mathbf{M}}^{(k,i)} &= \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}}\mathcal{C}_{\text{old}}}\right)^\top \mathbf{H}_{\mathcal{A}_{\text{old}}\mathcal{C}_{\text{old}}} \\ \mathbf{v}^{(k,i)} &= \left(\mathbf{H}^{(k,i)}_{\mathcal{B}_{\text{old}}\mathcal{C}_{\text{old}}}\right)^\top \tilde{\mathbf{y}}^{(k,i)}_{\mathcal{T}_{\text{old}}}\end{aligned}$$

- Update the (k,i)-th row vector of tensor $\mathcal{A}$ by Eq. (7.4) for all $1 \le k \le n$ and $1 \le i \le m$.

We note that the cost for updating $\mathcal{B}_{\text{new}}$ lies in solving a least-squares problem with the block-diagonal coefficient matrix of size $mdn \times md\ell$, which we can reduce to solving md smaller least-squares subproblems with the block coefficient matrix of size $n \times \ell$. So the total cost for updating $\mathcal{B}_{\text{new}}$ is $\mathcal{O}(md(n+1)\ell^2)$. Similarly, the cost for updating $\mathcal{C}_{\text{new}}$ is $\mathcal{O}(nd(m+1)\ell^2)$, and the cost for updating $\mathcal{A}$ is $\mathcal{O}(mn(p+d+1)\ell^2)$.

7.2 Multi-aspect Streaming

In many real-world applications, tensors often evolve along multiple dimensions simultaneously. For instance, consider a dynamic tensor arising in a recommendation system, structured as user × movie × actor. In such a setting, the number of registered users, available movies, and listed actors may all grow over time. This incremental and multi-dimensional expansion introduces a novel streaming paradigm, which we refer to as multi-aspect streaming. In this section, we will show that the *onlineBMD* algorithm developed for the single-mode streaming scenarios can be naturally extended to accommodate multi-aspect streaming settings.

Referring to Fig. 7.3, given a third-order tensor $\mathcal{X} \in \mathbb{R}^{m \times p \times n}$ with steady growth on all three modes, at time t+1, we have the new data tensor $\tilde{\mathcal{X}} \in \mathbb{R}^{(m+d_3)\times(p+d_1)\times(n+d_2)}$.

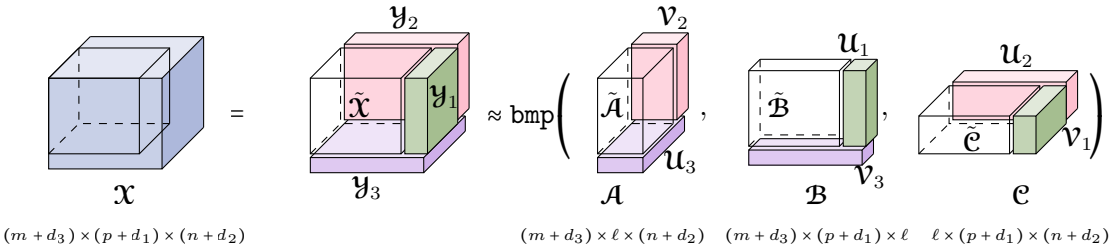


Figure 7.3: Multi-aspect Streaming Tensor BMD

We describe the following partition rule of the tensor $\mathcal{X}$ which is composed of four sub-tensors

- original data tensor at time t, $\widetilde{\mathcal{X}} \in \mathbb{R}^{m \times p \times n}$,

- new data tensor along mode-2, $\mathcal{Y}_1 \in \mathbb{R}^{m\times d_1\times n}$,
- new data tensor along mode-3, $\mathcal{Y}_2 \in \mathbb{R}^{m\times (p+d_1)\times d_2}$,
- new data tensor along mode-1, $\mathcal{Y}_3 \in \mathbb{R}^{d_3\times (p+d_1)\times (n+d_2)}$.

Then at time $t+1$, we have the $\mathcal{X}$ which can be written as

$$\mathcal{X} = \texttt{cat}\left(1, \texttt{cat}\left(3, \texttt{cat}\left(2, \widetilde{\mathcal{X}}, \mathcal{Y}_1\right), \mathcal{Y}_2\right), \mathcal{Y}_3\right). \tag{7.5}$$

Referring to Fig. 7.3, a BM-rank ℓ decomposition of the tensor $\widetilde{\mathcal{X}}$ is given as

$$\widetilde{\mathcal{X}} \approx \texttt{bmp}\left(\widetilde{\mathcal{A}}, \widetilde{\mathcal{B}}, \widetilde{\mathcal{C}}\right), \tag{7.6}$$

where $\widetilde{\mathcal{A}} \in \mathbb{R}^{m\times \ell\times n}$, $\widetilde{\mathcal{B}} \in \mathbb{R}^{m\times p\times \ell}$, and $\widetilde{\mathcal{C}} \in \mathbb{R}^{\ell\times p\times n}$ are the factor tensors of $\widetilde{\mathcal{X}}$. Our goal is to update the factor tensors without re-computing a BM-rank ℓ decomposition of the new tensor $\mathcal{X}$. That is, we want to find a BM-rank ℓ decomposition of $\mathcal{X}$ such that

$$\mathcal{X} \approx \texttt{bmp}\left(\mathcal{A}, \mathcal{B}, \mathcal{C}\right), \tag{7.7}$$

where $\mathcal{A} = \texttt{cat}\left(1, \texttt{cat}\left(3, \widetilde{\mathcal{A}}, \mathcal{V}_2\right), \mathcal{U}_3\right) \in \mathbb{R}^{(m+d_3)\times \ell\times (n+d_2)}$, $\mathcal{B} = \texttt{cat}\left(1, \texttt{cat}\left(2, \widetilde{\mathcal{B}}, \mathcal{U}_1\right), \mathcal{V}_3\right) \in \mathbb{R}^{(m+d_3)\times (p+d_1)\times \ell}$, and $\mathcal{C} = \texttt{cat}\left(3, \texttt{cat}\left(2, \widetilde{\mathcal{C}}, \mathcal{V}_1\right), \mathcal{U}_2\right) \in \mathbb{R}^{\ell\times (p+d_1)\times (n+d_2)}$ are the updated factor tensors leveraging the existing decomposition of $\widetilde{\mathcal{X}}$. The streaming *onlineBMD* algorithm is applied sequentially when the new data batch arrives. We will show the computational details as follows.

As illustrated in Fig. (7.4), when a new data batch $\mathcal{Y}_1 \in \mathbb{R}^{m\times d_1\times n}$ arrives along mode-2 of the existing tensor $\widetilde{\mathcal{X}}$, the updated data tensor is formed as $\mathcal{X} = \texttt{cat}(2, \widetilde{\mathcal{X}}, \mathcal{Y}_1)$. To incorporate this update, we employ the streaming *onlineBMD* algorithm to incrementally update the factor tensors $\mathcal{B} \in \mathbb{R}^{m\times (p+d_1)\times \ell}$ and $\mathcal{C} \in \mathbb{R}^{\ell\times (p+d_1)\times n}$. This is achieved by first computing the intermediate components $\mathcal{U}_1 \in \mathbb{R}^{m\times d_1\times \ell}$ and $\mathcal{V}_1 \in \mathbb{R}^{\ell\times d_1\times n}$.

As illustrated in Fig. (7.5), when a new data batch $\mathcal{Y}_2 \in \mathbb{R}^{m\times (p+d_1)\times d_2}$ arrives along mode-3, the updated data tensor is formed as $\mathcal{X} = \texttt{cat}(3, \texttt{cat}(2, \widetilde{\mathcal{X}}, \mathcal{Y}_1), \mathcal{Y}_2)$.

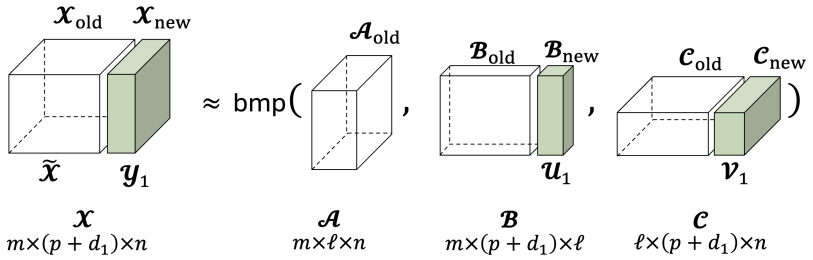


Figure 7.4: Update $\mathcal{U}_1$, $\mathcal{V}_1$, and $\mathcal{A}$ by streaming onlineBMD.

We employ the streaming *onlineBMD* algorithm to incrementally update the factor tensors $\mathcal{C}^\top \in \mathbb{R}^{(p+d_1)\times(n+d_2)\times\ell}$ and $\mathcal{A}^\top \in \mathbb{R}^{\ell\times(n+d_2)\times m}$. This is achieved by first computing the intermediate components $\mathcal{U}_2^\top \in \mathbb{R}^{(p+d_1)\times d_2\times\ell}$ and $\mathcal{V}_2^\top \in \mathbb{R}^{\ell\times d_2\times m}$.

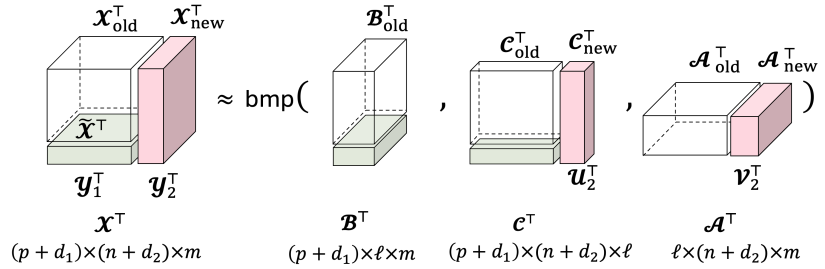


Figure 7.5: Update $\mathcal{U}_2^\top$, $\mathcal{V}_2^\top$, and $\mathcal{B}^\top$ by streaming onlineBMD.

As illustrated in Fig. (7.6), when a new data batch $\mathcal{Y}_3 \in \mathbb{R}^{d_3\times(p+d_1)\times(n+d_2)}$ arrives along mode-1, the updated data tensor is formed as $\mathcal{X} = \texttt{cat}(1, \texttt{cat}(3, \texttt{cat}(2, \widetilde{\mathcal{X}}, \mathcal{Y}_1), \mathcal{Y}_2), \mathcal{Y}_3)$. We employ the streaming *onlineBMD* algorithm to incrementally update the factor tensors $\mathcal{A}^{\top^2} \in \mathbb{R}^{(n+d_2)\times(m+d_3)\times\ell}$ and $\mathcal{B}^{\top^2} \in \mathbb{R}^{\ell\times(m+d_3)\times(p+d_1)}$. This is achieved by first computing the intermediate components $\mathcal{U}_3^{\top^2} \in \mathbb{R}^{(n+d_2)\times d_3\times\ell}$ and $\mathcal{V}_3^{\top^2} \in \mathbb{R}^{\ell\times d_3\times(p+d_1)}$.

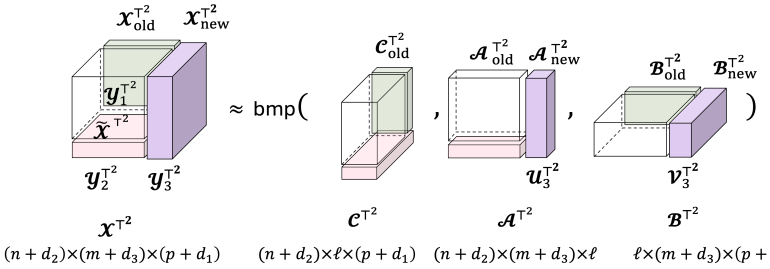


Figure 7.6: Update $\mathcal{U}_3^{\top^2}$, $\mathcal{V}_3^{\top^2}$, and $\mathcal{C}^{\top^2}$ by streaming onlineBMD.

7.3 Empirical Analysis

In this section, we assess the performance of the proposed *sliceUpdate* and *onlineBMD* algorithms in comparison to the baseline *BMD-ALS* algorithm across three distinct datasets.

- The first dataset is MATLAB's built-in "car video," which comprises 120 grayscale frames, each with spatial dimensions of 120×160. These frames are stacked as lateral slices to construct a third-order tensor of dimensions $120 \times 120 \times 160$.
- The second dataset is the AT&T Database of Faces (formerly known as the ORL Database of Faces) [85]. We refer to it as the "ORL" face dataset. The full dataset contains 40 individuals with 10 facial images per person. For our experiments, we select a subset of 120 images and organize them as lateral slices, forming a tensor of size $112 \times 120 \times 92$.
- The third dataset, referred to as the "Cylinder" dataset, represents a simulation of two-dimensional viscous fluid flow around a cylindrical obstacle [31, 80].* The original data dimensions are $640 \times 80 \times 1501$. For computational efficiency, we extract 120 evenly spaced temporal snapshots from the full sequence and downsample the spatial resolution to 150×80, resulting in a tensor of size $150 \times 120 \times 80$.

Representative examples from each dataset are shown in Fig. 7.7, including the 20th frame from the "car video" dataset, sample images of six individuals from the "ORL" dataset, and the 20th snapshot from the "Cylinder" dataset.

To compare the performance of the three algorithms, we adopt the following experimental protocol. Initially, 20% of each dataset is used for warm-start initialization. The remaining 80% of the data is incrementally introduced in batches of 5% of the total number of slices.

*Available at https://cgl.ethz.ch/research/visualization/data.php under the title "Cylinder Flow with von Karman Vortex Street."

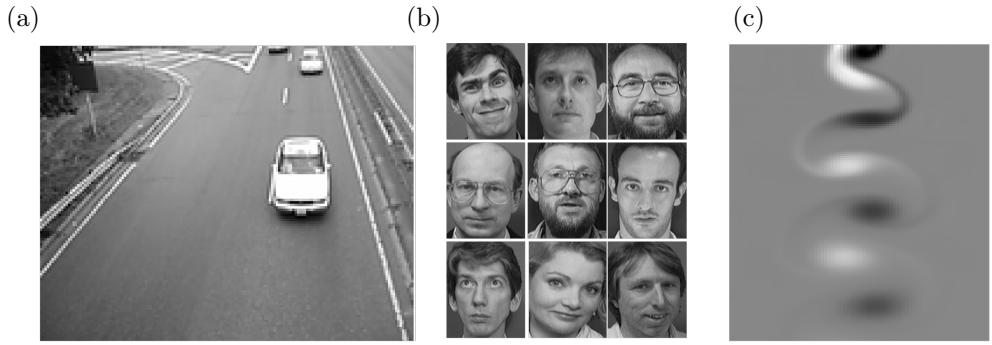


Figure 7.7: Displayed by the MATLAB IMAGESC command: (a)The 20-th frame of the car video (b) Examples of 6 individuals from the ORL faces dataset (c) The 20-th snapshot of the cylinder dataset.

Performance evaluation is based on two criteria: *effectiveness* and *efficiency*. Effectiveness is quantified using the *fitness* metric, defined as

$$\text{fitness} = \left(1 - \frac{\|\mathcal{X} - \widetilde{\mathcal{X}}\|_F}{\|\mathcal{X}\|_F}\right),$$

where $\mathcal{X}$ denotes the ground-truth data tensor and $\widetilde{\mathcal{X}}$ its reconstruction. Efficiency is assessed in terms of computational runtime, measured in *seconds*. All experiments are conducted on MATLAB version R2024b (maca64).

7.3.1 Streaming

We begin by noting that the *onlineBMD* algorithm, which is an ALS-type algorithm, is sensitive to its initialization. To investigate the impact of different initialization strategies, we consider the following three approaches:

- Slicewise SVD: For each incoming data batch, singular value decompositions (SVDs) are computed on the frontal slices of the new data.
- Random: The factor tensors are initialized with entries drawn from a standard Gaussian distribution.
- Previous Batch: The factor tensors are initialized using the decomposition results obtained from the previous data batch.

Figure 7.8 presents the fitness values as a function of the number of processed slices for the "car video" dataset. The right-hand panel of the figure shows the cumulative runtime corresponding to each initialization strategy. Overall, the results indicate that initializing with the factorization from the previous batch yields the highest effectiveness in terms of fitness. Furthermore, both the previous batch and random initializations are computationally more efficient than the slicewise SVD approach. Based on these findings, we adopt the previous batch initialization strategy for all subsequent experiments involving the *onlineBMD* algorithm.

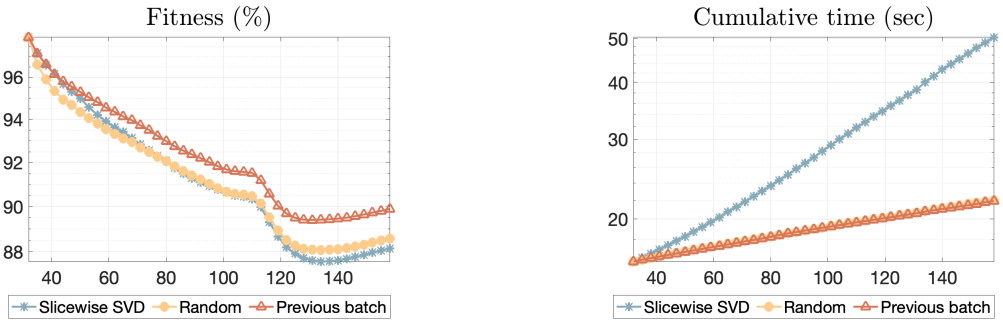


Figure 7.8: Comparison of three different initialization schemes on the "car video" data: Slicewise SVD, Random initialization, and Previous batch initialization.

Next, we compare the three algorithms *sliceUpdate*, *onlineBMD*, and *BMD-ALS* on the three datasets: "car video", "ORL" face dataset, and "cylinder" dataset. Each experiment is repeated five times, and the average values of both the fitness and runtime are reported. The results are summarized in Fig. (7.9). As illustrated by the bar chart, all three algorithms demonstrate comparable performance on the "car video" and ORL face datasets in terms of fitness. In contrast, for the Cylinder dataset, the *BMD-ALS* algorithm achieves significantly higher fitness compared to the two streaming algorithms, *sliceUpdate* and *onlineBMD*. However, this improvement in accuracy comes at the cost of considerably increased computational time. Between the two streaming approaches, *sliceUpdate* consistently exhibits slightly better fitness scores than *onlineBMD*, suggesting that it is more suitable for real-time or resource-constrained streaming applications.

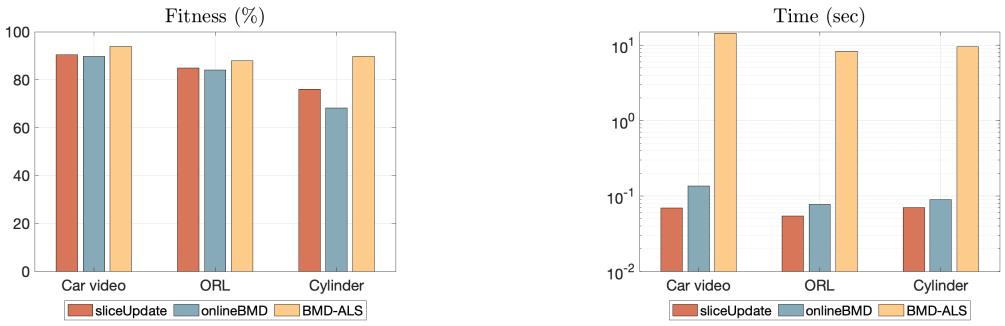


Figure 7.9: Performance comparison of the three methods: *sliceUpdate*, *onlineBMD*, and *BMD-ALS* on distinct datasets.

7.3.2 Multi-aspect Streaming

Finally, we evaluate the performance of the multi-aspect version of the *onlineBMD* algorithm on the "car video" dataset. The video data is initialized with 20% of the full video tensor and then added by 5% data along each dimension. The results are presented in Fig. (7.10). When compared to the baseline *BMD-ALS* algorithm, both methods achieve comparable levels of fitness.

However, we observe that in the multi-aspect setting, initializing with the factorization results from the previous batch leads to reduced accuracy relative to its effectiveness in the single-streaming scenario. Despite this, the *onlineBMD* algorithm demonstrates superior computational efficiency in both the streaming and multi-aspect settings when compared to the baseline *BMD-ALS* method.

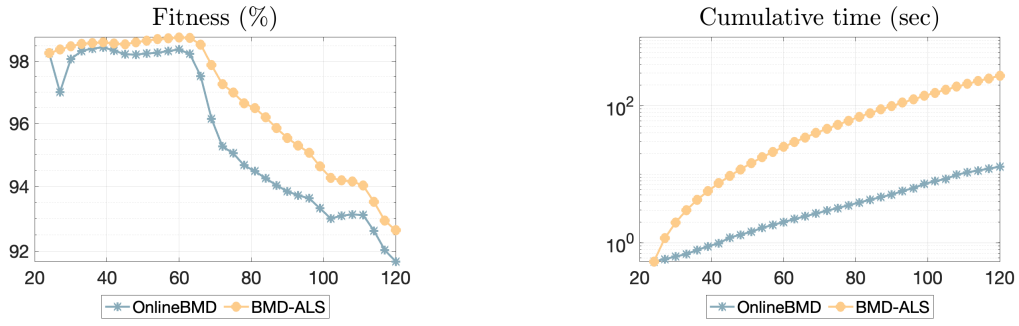


Figure 7.10: Performance comparison of the *onlineBMD* and *BMD-ALS* methods on the “car video” data in the the multi-aspect streaming setting.

Chapter 8

Conclusion and Future Work

In this thesis, we have investigated the computation and applications of a novel tensor decomposition techniques within the Bhattacharya-Mesner (BM) algebraic framework.

We have demonstrated that the tensor BM-decomposition (BMD) can be effectively used for grayscale video background/foreground separation with compression. Using our generative spatiotemporal video model, we have shown that videos with a stationary background are naturally low BM-rank tensors. Moreover, we have shown that the alternating least-squares algorithm for computing the BMD is closely connected to the block nonlinear Gauss-Seidel (GS) method, from which the convergence analysis of the algorithm follows naturally. In addition, we provided new theoretical insights into the bounds on BM-rank and the issue of non-uniqueness, the latter of which we addressed through a regularization scheme specifically designed for the video decomposition task. We also extended the regularized ALS algorithm to compute fourth-order tensor BMD, thereby enabling effective color video compression and separation.

Our work further demonstrated that the low BM-rank decomposition assumption is highly effective for tensor completion tasks. The strong compressibility of spatiotemporal data under the BMD framework suggests its broader applicability to more general classes of data. Numerical experiments conducted on real-world video and hyperspectral image datasets confirmed the efficacy of the proposed ADMM-based tensor completion algorithm.

Non-negative matrix factorization (NMF) is a foundational tool in numerous data science applications. In this context, we showed that the well-known multiplicative update algorithm for NMF can be directly extended to the third-order non-negative BM-decomposition (NNBMD). Our numerical results indicate that the NNBMD

algorithm yields parts-based features that are qualitatively distinct from those obtained via traditional NMF. Finally, we demonstrated that the ALS-type algorithm, *onlineBMD*, efficiently solves the BM-decomposition problem in both streaming and multi-aspect streaming settings – an essential capability in the modern era of continuously acquired, high-velocity data.

For future directions, we will investigate the use of constraints for a better posed BM-decomposition problem in applications, the feasibility of the extension of computation to higher dimensions, and we will investigate the utility of the BMD applied to other types of spatiotemporal data. I am also interested in the proximal alternating method for computing the tensor BMD rather than the plain ALS algorithm for gaining possible acceleration or better convergence behavior. For tensor completion, we will investigate the significance of fine-tuning the parameters in the ADMM algorithm, as well as an adaptive BM-rank selection mechanism in the decomposition. We will also address the artifacts exhibited in the reconstructed images by NNBMD and reduced facial recognition accuracy by imposing regularization, such as sparsity, to the decomposition problem.

Much work remains to be done in the future. Some specific research projects that I intend to pursue in the near future are as follows:

(1) Solving large systems of equations for third-order tensors presents significant challenges. In the context of matrix linear systems, randomized iterative methods have emerged as a popular approach for solving or approximating solutions to the systems that are too large to fit into memory. Among these, the randomized Kaczmarz method has received particular attention. Recent studies, such as [66], have explored Kaczmarz-type iterative methods for tensor linear systems under the t-product framework. However, to date, no randomized algorithms have been developed for solving tensor systems under the BM-product framework. Given the strong connections between the tensor BM-product and the tensor t-product, as highlighted in [93], the development of Kaczmarz-type algorithms for third-order tensors within the BM-product framework appears to be a promising research direction.

(2) Tensor spectral decomposition and singular value decomposition were formulated in [26, 28]. An immediate extension of this work involves developing efficient numerical algorithms to compute these decompositions. Our previous results demonstrate that tensor BMD effectively separates background and foreground in surveillance-type video while achieving high compression. Utilizing tensor singular value decomposition will enable us to derive optimality results, thereby generalizing the Eckart-Young theorem for tensors under the BM-algebraic framework. Furthermore, this approach will facilitate investigations into statistical methods, such as principal component analysis (PCA), for analyzing higher-order tensor data.

A novel tensor-based approach opens new avenues for innovation by extending traditional linear algebra methods to incorporate tensor algebra, offering enhanced analytical capabilities. For example, tensors can model latent variables and capture higher-order moments in statistics. Advanced techniques, such as tensor-on-tensor regression and symmetric tensor decomposition, enable more comprehensive analyses of multi-way datasets, improving our understanding of high-dimensional data. More recently, tensor methods have also drawn popularity in signal processing for solving large-scale inverse problems [5, 77]. The increasing interests in tensors underlines the growing applications of tensor methods in tackling complex, multidimensional challenges across various fields.

Bibliography

[1] Karim Abed-Meraim, Nguyen Linh Trung, Adel Hafiane, et al., *A contemporary and comprehensive survey on streaming tensor decomposition*, IEEE Transactions on Knowledge and Data Engineering, 35 (2022), pp. 10897–10921.

[2] Carl J Appellof and Ernest R Davidson, *Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents*, Analytical Chemistry, 53 (1981), pp. 2053–2056.

[3] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter, *Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods*, Mathematical Programming, 137 (2013), pp. 91–129.

[4] Grey Ballard and Tamara G. Kolda, *Tensor Decompositions for Data Science*, Cambridge University Press, 2025.

[5] Çağdaş Bilen, Alexey Ozerov, and Patrick Pérez, *Solving time-domain audio inverse problems using nonnegative tensor factorization*, IEEE Transactions on Signal Processing, 66 (2018), pp. 5604–5617.

[6] Thierry Bouwmans, Lucia Maddalena, and Alfredo Petrosino, *Scene background initialization (sbi) dataset*, 2016.

[7] ——, *Scene background initialization: A taxonomy*, Pattern Recognition Letters, 96 (2017), pp. 3–11.

[8] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen, *A singular value thresholding algorithm for matrix completion*, SIAM Journal on optimization, 20 (2010), pp. 1956–1982.

[9] Emmanuel Candes and Benjamin Recht, *Exact matrix completion via convex optimization*, Communications of the ACM, 55 (2012), pp. 111–119.

[10] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright, *Robust principal component analysis?*, Journal of the ACM (JACM), 58 (2011), pp. 1–37.

[11] Emmanuel J Candès and Terence Tao, *The power of convex relaxation: Near-optimal matrix completion*, IEEE Transactions on Information Theory, 56 (2010), pp. 2053–2080.

[12] J Douglas Carroll and Jih-Jie Chang, *Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[13] Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan, *Tensor decompositions for signal processing applications: From two-way to multiway component analysis*, IEEE signal processing magazine, 32 (2015), pp. 145–163.

[14] Fengyu Cong, Qiu-Hua Lin, Li-Dan Kuang, Xiao-Feng Gong, Piia Astikainen, and Tapani Ristaniemi, *Tensor decomposition of eeg signals: a brief review*, Journal of neuroscience methods, 248 (2015), pp. 59–69.

[15] Fernando A Correia, José Luiz Nunes, Paulo Henrique Alves, and Hélio Lopes, *Dynamic topic modeling with tensor decomposition as a tool to explore the legal precedent relevance over time*, in Proceedings of the ACM Symposium on Document Engineering 2023, 2023, pp. 1–10.

[16] Kasikrit Damkliang, *At&t database of faces*.

[17] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle, *A multilinear singular value decomposition*, SIAM journal on Matrix Analysis and Applications, 21 (2000), pp. 1253–1278.

[18] Lieven De Lathauwer and Dimitri Nion, *Decompositions of a higher-order tensor in block terms—part iii: Alternating least squares algorithms*, SIAM journal on Matrix Analysis and Applications, 30 (2008), pp. 1067–1083.

[19] N Benjamin Erichson, Steven L Brunton, and J Nathan Kutz, *Compressed dynamic mode decomposition for background modeling*, Journal of Real-Time Image Processing, 16 (2019), pp. 1479–1492.

[20] N Benjamin Erichson and Carl Donovan, *Randomized low-rank dynamic mode decomposition for motion detection*, Computer Vision and Image Understanding, 146 (2016), pp. 40–50.

[21] Michael P Friedlander and Kathrin Hatz, *Computing non-negative tensor factorizations*, Optimisation Methods and Software, 23 (2008), pp. 631–647.

[22] Lianru Gao, Zhicheng Wang, Lina Zhuang, Haoyang Yu, Bing Zhang, and Jocelyn Chanussot, *Using low-rank representation of abundance maps and nonnegative tensor factorization for hyperspectral nonlinear unmixing*, IEEE Transactions on Geoscience and Remote Sensing, 60 (2021), pp. 1–17.

[23] Belmar Garcia-Garcia, Thierry Bouwmans, and Alberto Jorge Rosales Silva, *Background subtraction in real applications: Challenges, current models and future directions*, Computer Science Review, 35 (2020), p. 100204.

[24] Edinah K Gnang, *Computational aspects of the combinatorial Nullstellensatz method via a polynomial approach to matrix and hypermatrix algebra*, Rutgers The State University of New Jersey, School of Graduate Studies, 2013.

[25] Edinah K Gnang, Ahmed Elgammal, and Vladimir Retakh, *A spectral theory for tensors*, in Annales de la Faculté des sciences de Toulouse: Mathématiques, vol. 20, 2011, pp. 801–841.

[26] Edinah K Gnang and Yuval Filmus, *On the spectra of hypermatrix direct sum and kronecker products constructions*, Linear Algebra and its Applications, 519 (2017), pp. 238–277.

[27] ——, *On the bhattacharya-mesner rank of third order hypermatrices*, Linear Algebra and its Applications, 588 (2020), pp. 391–418.

[28] Edinah K Gnang and Fan Tian, *A symmetrization approach to hypermatrix svd*, arXiv preprint arXiv:2004.10368, (2020).

[29] Gene H Golub and Charles F Van Loan, *Matrix computations*, JHU press, 2013.

[30] Jacob Grosek and J Nathan Kutz, *Dynamic mode decomposition for real-time background/foreground separation in video*, arXiv preprint arXiv:1404.7592, (2014).

[31] Tobias Günther, Markus Gross, and Holger Theisel, *Generic objective vortices for flow visualization*, ACM Transactions on Graphics (Proc. SIGGRAPH), 36 (2017), pp. 141:1–141:11.

[32] Ning Hao, Lior Horesh, and Misha E Kilmer, *Nonnegative tensor decomposition*, in Compressed sensing & sparse filtering, Springer, 2013, pp. 123–148.

[33] Ning Hao, Misha E Kilmer, Karen Braman, and Randy C Hoover, *Facial recognition using tensor-tensor decompositions*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 437–463.

[34] Israr Ul Haq, Keisuke Fujii, and Yoshinobu Kawahara, *Dynamic mode decomposition via dictionary learning for foreground modeling in videos*, Computer Vision and Image Understanding, 199 (2020), p. 103022.

[35] Richard A Harshman et al., *Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis*, UCLA working papers in phonetics, 16 (1970), p. 84.

[36] Johan Håstad, *Tensor rank is np-complete*, in Automata, Languages and Programming: 16th International Colloquium Stresa, Italy, July 11–15, 1989 Proceedings 16, Springer, 1989, pp. 451–460.

[37] Tamir Hazan, Simon Polak, and Amnon Shashua, *Sparse image coding using a 3d non-negative tensor factorization*, in Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, vol. 1, IEEE, 2005, pp. 50–57.

[38] Christopher J Hillar and Lek-Heng Lim, *Most tensor problems are np-hard*, Journal of the ACM (JACM), 60 (2013), pp. 1–39.

[39] Frank L Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, Journal of Mathematics and Physics, 6 (1927), pp. 164–189.

[40] ——, *Multiple invariants and generalized rank of a p-way matrix or tensor*, Journal of Mathematics and Physics, 7 (1928), pp. 39–79.

[41] Ibrahim Kajo, Nidal Kamel, Yassine Ruichek, and Aamir Saeed Malik, *Svd-based tensor-completion technique for background initialization*, IEEE Transactions on Image Processing, 27 (2018), pp. 3114–3126.

[42] Vasileios Kalantzis, Georgios Kollias, Shashanka Ubaru, Athanasios N Nikolakopoulos, Lior Horesh, and Kenneth Clarkson, *Projection techniques to update the truncated svd of evolving matrices with applications*, in International Conference on Machine Learning, PMLR, 2021, pp. 5236–5246.

[43] RAMIN GOUDARZI KARIM, *TENSOR DECOMPOSITIONS AND RANK APPROXIMATION OF TENSORS WITH APPLICATIONS*, PhD thesis, The University of Alabama in Huntsville, 2019.

[44] Ramin Goudarzi Karim, Guimu Guo, Da Yan, and Carmeliza Navasca, *Accurate tensor decomposition with simultaneous rank approximation for surveillance videos*, in 2020 54th Asilomar Conference on Signals, Systems, and Computers, IEEE, 2020, pp. 842–846.

[45] Eric Kernfeld, Misha Kilmer, and Shuchin Aeron, *Tensor–tensor products with invertible linear transforms*, Linear Algebra and its Applications, 485 (2015), pp. 545–570.

[46] Henk AL Kiers, *Towards a standardized notation and terminology in multiway analysis*, Journal of Chemometrics: A Journal of the Chemometrics Society, 14 (2000), pp. 105–122.

[47] Misha E Kilmer, Lior Horesh, Haim Avron, and Elizabeth Newman, *Tensor-tensor algebra for optimal representation and compression of multiway data*, Proceedings of the National Academy of Sciences, 118 (2021), p. e2015851118.

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

[49] Hyunsoo Kim and Haesun Park, *Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis*, Bioinformatics, 23 (2007), pp. 1495–1502.

[50] Yong-Deok Kim and Seungjin Choi, *Nonnegative tucker decomposition*, in 2007 IEEE conference on computer vision and pattern recognition, IEEE, 2007, pp. 1–8.

[51] Yong-Deok Kim, Andrzej Cichocki, and Seungjin Choi, *Nonnegative tucker decomposition with alpha-divergence*, in 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2008, pp. 1829–1832.

[52] Donald Ervin Knuth, *The Art of Computer Programming: Fundamental Algorithms.*, vol. 1, Addison-Wesley, 1997.

[53] Tamara G Kolda and Brett W Bader, *Tensor decompositions and applications*, SIAM review, 51 (2009), pp. 455–500.

[54] Anna Konstorum, Subhasis Mohanty, Yujiao Zhao, Anthony Melillo, Brent Vander Wyk, Allison Nelson, Sui Tsang, Tamara P

Blevins, Robert B Belshe, Daniel G Chawla, et al., *Platelet response to influenza vaccination reflects effects of aging*, Aging Cell, 22 (2023), p. e13749.

[55] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor, *Dynamic mode decomposition: data-driven modeling of complex systems*, SIAM, 2016.

[56] J Nathan Kutz, Xing Fu, Steve L Brunton, and N Benjamin Erichson, *Multi-resolution dynamic mode decomposition for foreground/background separation and object tracking*, in 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), IEEE, 2015, pp. 921–929.

[57] Daniel Lee and H Sebastian Seung, *Algorithms for non-negative matrix factorization*, Advances in neural information processing systems, 13 (2000).

[58] Daniel D Lee and H Sebastian Seung, *Learning the parts of objects by non-negative matrix factorization*, nature, 401 (1999), pp. 788–791.

[59] Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian, *Statistical modeling of complex backgrounds for foreground object detection*, IEEE Transactions on image processing, 13 (2004), pp. 1459–1472.

[60] Na Li, Stefan Kindermann, and Carmeliza Navasca, *Some convergence results on the regularized alternating least-squares method for tensor decomposition*, Linear Algebra and its Applications, 438 (2013), pp. 796–812.

[61] Zina Li, Yao Wang, Qian Zhao, Shijun Zhang, and Deyu Meng, *A tensor-based online rpca model for compressive background subtraction*, IEEE Transactions on Neural Networks and Learning Systems, (2022).

[62] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye, *Tensor completion for estimating missing values in visual data*, IEEE transactions on pattern analysis and machine intelligence, 35 (2012), pp. 208–220.

[63] Weixiang Liu, Nanning Zheng, and Qubo You, *Nonnegative matrix factorization and its applications in pattern recognition*, Chinese Science Bulletin, 51 (2006), pp. 7–18.

[64] Yuanyuan Liu, Fanhua Shang, Hong Cheng, James Cheng, and Hanghang Tong, *Factor matrix trace norm minimization for low-rank tensor completion*, in Proceedings of the 2014 SIAM International Conference on Data Mining, SIAM, 2014, pp. 866–874.

[65] Qilun Luo, Ming Yang, Wen Li, and Mingqing Xiao, *Multidimensional data processing with bayesian inference via structural block decomposition*, IEEE Transactions on Cybernetics, (2023).

[66] Anna Ma and Denali Molitor, *Randomized kaczmarz for tensor linear systems*, BIT Numerical Mathematics, 62 (2022), pp. 171–194.

[67] Lucia Maddalena and Alfredo Petrosino, *Towards benchmarking scene background initialization*, in New Trends in Image Analysis and Processing–ICIAP 2015 Workshops: ICIAP 2015 International Workshops, BioFor, CTMR, RHEUMA, ISCA, MADiMa, SBMI, and QoEM, Genoa, Italy, September 7-8, 2015, Proceedings 18, Springer, 2015, pp. 469–476.

[68] Dale M Mesner and Prabir Bhattacharya, *Association schemes on triples and a ternary algebra*, Journal of Combinatorial Theory, Series A, 55 (1990), pp. 204–234.

[69] ——, *A ternary algebra arising from association schemes on triples*, Journal of algebra, 164 (1994), pp. 595–613.

[70] J Mocks, *Topographic components model for event-related potentials and some biophysical considerations*, IEEE transactions on biomedical engineering, 35 (1988), pp. 482–484.

[71] Makoto Nakatsuji, Qingpeng Zhang, Xiaohui Lu, Bassem Makni, and James A Hendler, *Semantic social network analysis by cross-domain tensor factorization*, IEEE Transactions on Computational Social Systems, 4 (2017), pp. 207–217.

[72] Carmeliza Navasca, Lieven De Lathauwer, and Stefan Kindermann, *Swamp reducing technique for tensor decomposition*, in 2008 16th European Signal Processing Conference, IEEE, 2008, pp. 1–5.

[73] Elizabeth Newman, Lior Horesh, Haim Avron, and Misha E Kilmer, *Stable tensor neural networks for efficient deep learning*, Frontiers in Big Data, 7 (2024), p. 1363978.

[74] Elizabeth Newman and Misha E Kilmer, *Nonnegative tensor patch dictionary approaches for image compression and deblurring applications*, SIAM Journal on Imaging Sciences, 13 (2020), pp. 1084–1112.

[75] Msoud Nickparvar, *Brain tumor mri dataset*, 2021.

[76] Ivan V Oseledets, *Tensor-train decomposition*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2295–2317.

[77] Alexey Ozerov, Antoine Liutkus, Roland Badeau, and Gaël Richard, *Coding-based informed source separation: Nonnegative tensor factorization approach*, IEEE Transactions on Audio, Speech, and Language Processing, 21 (2013), pp. 1699–1712.

[78] Pentti Paatero and Unto Tapper, *Positive matrix factorization: A nonnegative factor model with optimal utilization of error estimates of data values*, Environmetrics, 5 (1994), pp. 111–126.

[79] V Paul Pauca, Farial Shahnaz, Michael W Berry, and Robert J Plemmons, *Text mining using non-negative matrix factorizations*, in Proceedings of the 2004 SIAM international conference on data mining, SIAM, 2004, pp. 452–456.

[80] S. Popinet, *Free computational fluid dynamics*, ClusterWorld, 2 (2004).

[81] Menaka Rajapakse and Lonce Wyse, *Face recognition with non-negative matrix factorization*, in Visual Communications and Image Processing 2003, vol. 5150, SPIE, 2003, pp. 1838–1847.

[82] William T Redman, *On koopman mode decomposition and tensor component analysis*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 31 (2021), p. 051101.

[83] Achim Rettinger, Hendrik Wermser, Yi Huang, and Volker Tresp, *Context-aware tensor decomposition for relation prediction in social networks*, Social Network Analysis and Mining, 2 (2012), pp. 373–385.

[84] Athanasios A Rontogiannis, Eleftherios Kofidis, and Paris V Giampouras, *Online rank-revealing block-term tensor decomposition*, in 2021 55th Asilomar Conference on Signals, Systems, and Computers, IEEE, 2021, pp. 1678–1682.

[85] Ferdinando S Samaria and Andy C Harter, *Parameterisation of a stochastic model for human face identification*, in Proceedings of 1994 IEEE workshop on applications of computer vision, IEEE, 1994, pp. 138–142.

[86] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos, *Tensor decomposition for signal processing and machine learning*, IEEE Transactions on signal processing, 65 (2017), pp. 3551–3582.

[87] Shaden Smith, Kejun Huang, Nicholas D Sidiropoulos, and George Karypis, *Streaming tensor factorization for infinite data sources*, in Proceedings of the 2018 SIAM International Conference on Data Mining, SIAM, 2018, pp. 81–89.

[88] Andrews Sobral, Sajid Javed, Soon Ki Jung, Thierry Bouwmans, and El-hadi Zahzah, *Online stochastic tensor decomposition for background subtraction in multispectral video sequences*, in Proceedings of the IEEE International Conference on Computer Vision Workshops, 2015, pp. 106–113.

[89] Yongseok Soh, Patrick Flick, Xing Liu, Shaden Smith, Fabio Checconi, Fabrizio Petrini, and Jee Choi, *High performance streaming tensor decomposition*, in 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2021, pp. 683–692.

[90] Sara Soltani, Misha E Kilmer, and Per Christian Hansen, *A tensor-based dictionary learning approach to tomographic image reconstruction*, BIT Numerical Mathematics, 56 (2016), pp. 1425–1454.

[91] Qingquan Song, Hancheng Ge, James Caverlee, and Xia Hu, *Tensor completion algorithms in big data analytics*, ACM Transactions on Knowledge Discovery from Data (TKDD), 13 (2019), pp. 1–48.

[92] Miguel Á. Padriñán, *Cloud video*. https://www.pexels.com/video/white-clouds-on-the-blue-sky-6772574/, 2021.

[93] Fan Tian, Misha E Kilmer, Eric Miller, and Abani Patra, *Tensor bm-decomposition for compression and analysis of video data*, arXiv preprint arXiv:2306.09201, (2023).

[94] Fan Tian, Misha E. Kilmer, Eric L. Miller, Abani Patra, and Anna Konstorum, *Approximate tensor bm product decomposition for temporal analysis of third-order data.* https://meetings.siam.org/sess/dsp_talk.cfm?p=122285, 9 2022.

[95] Ying-Li Tian, Max Lu, and Arun Hampapur, *Robust and efficient foreground analysis for real-time video surveillance*, in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, IEEE, 2005, pp. 1182–1187.

[96] Lloyd N Trefethen and David Bau, *Numerical linear algebra*, vol. 181, Siam, 2022.

[97] Ledyard R Tucker, *Implications of factor analysis of three-way matrices for measurement of change*, Problems in measuring change, 15 (1963), p. 3.

[98] ——, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.

[99] André Uschmajew, *Local convergence of the alternating least squares algorithm for canonical tensor approximation*, SIAM Journal on Matrix Analysis and Applications, 33 (2012), pp. 639–652.

[100] Jianyu Wang and Linruize Tang, *Wasserstein nonnegative tensor factorization with manifold regularization*, arXiv preprint arXiv:2401.01842, (2024).

[101] Xiaokang Wang, Wei Wang, Laurence T Yang, Siwei Liao, Dexiang Yin, and M Jamal Deen, *A distributed hosvd method with its incremental computation for big data in cyber-physical-social systems*, IEEE Transactions on Computational Social Systems, 5 (2018), pp. 481–492.

[102] Max Welling and Markus Weber, *Positive tensor factorization*, Pattern Recognition Letters, 22 (2001), pp. 1255–1261.

[103] Houping Xiao, Fei Wang, Fenglong Ma, and Jing Gao, *eotd: An efficient online tucker decomposition for higher order tensors*, in 2018 IEEE international conference on data mining (ICDM), IEEE, 2018, pp. 1326–1331.

[104] Wei Xu, Xin Liu, and Yihong Gong, *Document clustering based on non-negative matrix factorization*, in Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, 2003, pp. 267–273.

[105] Yuning Yang, *On global convergence of alternating least squares for tensor approximation*, Computational Optimization and Applications, (2022), pp. 1–21.

[106] Tatsuya Yokota, Burak Erem, Seyhmus Guler, Simon K Warfield, and Hidekata Hontani, *Missing slice recovery for tensors using a low-rank model in embedded space*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8251–8259.

[107] Stefanos Zafeiriou, *Algorithms for nonnegative tensor factorization*, in Tensors in Image Processing and Computer Vision, Springer, 2009, pp. 105–124.

[108] Stefanos Zafeiriou, Anastasios Tefas, and Ioannis Pitas, *Discriminant nmffaces for frontal face verification*, in 2005 IEEE Workshop on Machine Learning for Signal Processing, IEEE, 2005, pp. 355–359.

[109] Zemin Zhang, *Tensor completion and tensor rpca*. https://github.com/jamiezeminzhang/Tensor_Completion_and_Tensor_RPCA/tree/master?tab=readme-ov-file, 2014.

[110] Zemin Zhang and Shuchin Aeron, *Exact tensor completion using t-svd*, IEEE Transactions on Signal Processing, 65 (2016), pp. 1511–1526.

[111] Zemin Zhang, Gregory Ely, Shuchin Aeron, Ning Hao, and Misha Kilmer, *Novel methods for multilinear data completion and de-noising based on tensor-svd*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 3842–3849.

[112] Zemin Zhang, Dehong Liu, Shuchin Aeron, and Anthony Vetro, *An online tensor robust pca algorithm for sequential 2d data*, in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2016, pp. 2434–2438.

[113] Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson, *Accelerating online cp decompositions for higher order tensors*, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1375–1384.