DESIGN AND APPLICATION OF TENSOR DECOMPOSITIONS TO PROBLEMS IN MODEL AND IMAGE COMPRESSION AND ANALYSIS

A dissertation

submitted by

Jiani Zhang

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in

Mathematics

TUFTS UNIVERSITY

May 2017

Adviser: Professor Misha E. Kilmer

Abstract

Tensor algebra and tensor computations have gained more and more attention in recent years due to their ability to handle and explore large-scale, high-dimensional datasets. In this thesis, we present four novel tensor-based methods in the fields of randomized algorithms, dynamical systems, image processing, and video processing.

In the first chapter, we introduce the history of tensor computation, discuss well-known tensor operators and decompositions, and demonstrate our motivations to focus on the t-product-based operators and decompositions designed by Professors Kilmer and Martin [51]. Then, in Chapter 2, we design a method called randomized tensor singular value decomposition that can produce a factorization with similar properties to the tensor SVD (t-SVD) but that is more computationally efficient on very large datasets. We present the details of the algorithm and the theoretical results, and we provide numerical results on two public facial recognition datasets. Chapter 3 addresses the problem of model reduction on dynamical systems. We investigate the proper orthogonal decomposition (POD) method, compare the approximation errors obtained from truncated SVD and truncated tensor SVD in theory, and provide an effective projector for the POD method using truncated tensor SVD. Chapter 4 and Chapter 5 are both devoted to optimization-related problems. In Chapter 4, for the multi-frame blind deconvolution optimization model, we design a method to select the most representative frames that is less heuristic in nature than current methods. In Chapter 5, we use tensor operators to model the video resolution enhancement problem and leverage the tensor nuclear norm as a regularization term to minimize the rank of its solution.

To my family and friends

Acknowledgements

First of all, I would like to thank my advisor, Professor Misha Kilmer. I am profoundly grateful for her valuable guidance and continuous support to my studies. Because she provided a positive, enjoyable, and free research environment, I was able to explore my research interests and efficiently discuss research problems with her. Moreover, she offered me many opportunities to learn cutting-edge knowledge in my research area and broaden my horizons in other different mathematical areas. I could not imagine having a better advisor and mentor for my Ph.D. study.

I also want to thank Xiaozhe Hu for serving as my committee member and for his guidance in my projects. I have benefited a lot from his expertise in partial differential equations and computer science. His passion and perseverance in research have had a huge influence on me. I wish to express my thanks to Lior Horesh for providing me the opportunity to work with him last summer and serving on my thesis committee. He is intelligent and motivated. It was a great pleasure working with him. His contribution to my thesis is enormous. My sincere thanks also go to James Adler for serving on my committee and for his support along the way.

I am truly grateful to my collaborators, Raymond Chan, James Nagy, Shuchin Aeron, and Arvind Saibaba for their encouragement and help during my study at Tufts. I would also like to thank my master advisors, Robert Plemmons and Jennifer Erway, who gave me the key to the world of optimization and numerical linear algebra.

I am truly grateful for all of my friends for your support! You made my Ph.D. life much easier.

Finally and foremost, I would like to thank my family for their unconditional love and support. I love you!

Contents

List of Tables

List of Figures

Design and Application of Tensor Decompositions to Problems in Model and Image Compression and Analysis

Chapter 1

Background Preparation

In this chapter, we will introduce the preliminary knowledge that is necessary for the entire thesis. It will include numerical linear algebra preliminaries in Section 1.1, tensor computation preliminaries in Section 1.2.

First, we give the notations we will use. Boldface lowercase letters indicate vectors, e.g. $\mathbf{a}$. Boldface uppercase letters indicate matrices, e.g. $\mathbf{A}$. Boldface Euler script letters indicate tensors, e.g. $\mathcal{A}$.

1.1 Numerical Linear Algebra Preliminaries

Almost all of the methods in data analysis and scientific computing rely on matrix algorithms [89]. In particular, rank-revealing matrix decomposition plays a crucially important role. Approximating a matrix by a product of some matrices with a smaller size may save the storage cost or/and computation cost inside algorithms that access the matrix. Moreover, these smaller matrices could provide some specific structures that help analyze the original matrix with better results. In this section, we will focus on introducing two classical matrix decompositions, which we will often use in the next four chapters. Let us begin with the singular value decomposition.

Definition 1.1.1 (SVD) *Given any matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *, there exists a singular value decomposition (SVD) of* $\mathbf{A}$*. It can be expressed as*

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$$

where $\mathbf{U}$ *is an* $m \times m$ *orthogonal matrix,* $\mathbf{S}$ *is an* $m \times n$ *diagonal matrix and* $\mathbf{V}$ *is an* $n \times n$ *orthogonal matrix. The diagonal entries of* $\mathbf{S}$*,* $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r \geq 0$*, where* r *is the rank of* $\mathbf{A}$*, are known as the singular values of* $\mathbf{A}$*.*

The SVD has a number of interesting and useful theoretical properties. We will

list some of those that are related to this thesis. See [35] for more.

Corollary 1.1.2 *If* $\mathbf{A} \in \mathbb{R}^{m\times n}$*, then* $\|\mathbf{A}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{r} \sigma_i^2}$ *and* $\|\mathbf{A}\|_2 = \sigma_1$*.*

Corollary 1.1.3 *Given* $\mathbf{A} \in \mathbb{R}^{m\times n}$*, denote the* i^{th} *column of matrix* $\mathbf{U}$ *and* $\mathbf{V}$ *as* $\mathbf{u}_i$ *and* $\mathbf{v}_i$ *correspondingly. If* $\mathrm{rank}(\mathbf{A}) = \mathrm{r}$*, then* $\mathrm{range}(\mathbf{A}) = \mathrm{span}\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{\mathrm{r}}\}$ *and* $\mathrm{null}(\mathbf{A}) = \mathrm{span}\{\mathbf{v}_{\mathrm{r}+1}, \mathbf{v}_{\mathrm{r}+2}, \ldots, \mathbf{v}_{\mathrm{n}}\}$*.*

Truncating the SVD to k terms provides the optimal rank k approximation in both 2-norm and Frobenius norm. Before introducing the Eckart-Young theorem, we will give the definition of the truncated SVD.

Definition 1.1.4 *Given a matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$*, the rank-k truncated Singular Value Decomposition (SVD) of* $\mathbf{A}$ *can be expressed as*

$$\mathbf{A} \approx \mathbf{U}_k \mathbf{S}_k \mathbf{V}_k^T$$

where k *is the target rank,* $\mathbf{U}_k$ *and* $\mathbf{V}_k$ *have the first* k *left and right singular vectors, respectively, and* $\mathbf{S}_k$ *is the* $k \times k$ *leading principal sub-matrix of* $\mathbf{S}$*.*

Theorem 1.1.5 (The Eckart-Young Theorem) *Given an* $m \times n$ *matrix* $\mathbf{A}$*, if* $k < r = \mathrm{rank}(\mathbf{A})$ *and* $\mathbf{A}_k = \mathbf{U}_k \mathbf{S}_k \mathbf{V}_k^{\mathrm{T}}$*, then*

$$\min_{\mathrm{rank}(\mathbf{B})=\mathrm{k}} \|\mathbf{A} - \mathbf{B}\|_2 = \|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}$$

and

$$\min_{\mathrm{rank}(\mathbf{B})=\mathrm{k}} \|\mathbf{A} - \mathbf{B}\|_{\mathrm{F}} = \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}} = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}.$$

The practical importance of SVD is cannot be overstated. It has many applications in a wide variety of areas. Here, we just list only a few as examples.

- Image Compression: A grayscale digital image can be represented as an $m \times n$ matrix $\mathbf{A}$. The entry of $\mathbf{A}$, $\mathbf{A}(i, j)$, corresponds to the measurement of gray scale. The idea of image compression is to use the truncated SVD of $\mathbf{A}$, $\mathbf{A}_k$, as an approximation of the original image. The total implicit storage for $\mathbf{A}_k$

is $k(m+n+1)$. The choice of k depends on the tradeoff between the cost of storage and the quality of the compressed image. In many cases, the total implicit storage can be less than 20% of the storage for the original image before one would visually see degradation in image quality between the compressed and uncompressed images. See [18, 56].

- Image Restoration: The idea of image restoration is to restore the image by truncating the small singular values, since the small singular values are known to magnify the "noise" in the data. See [39].

- Facial Recognition: Principle Component Analysis (PCA) is widely used in facial recognition. See [80] for an example. The left singular vectors of a matrix span the range of the matrix, and they are the normalized principle components [56].

- Natural Language Processing: Consider a database that consists of m keywords and n documents. It can be represented as an $m \times n$ matrix. Each entry is the frequency of i^{th} keyword in j^{th} document. Latent semantic indexing is a method to deal with the problem of polysemy and synonyms. It uses truncated SVD to obtain a well-approximated database that has simpler and more clear structure. More details can be found in [56].

The standard algorithm to compute the SVD is named Golub-Kahan-Reinsch algorithm [33, 34]. Basically, the algorithm has two phases, reducing to a bidiagonal form and finding the SVD of the bidiagonal matrix. More details see [18]. The computation cost of these two phases is $O(mn^2)$, which is primarily determined by the cost of phase 2. The computational cost for truncated SVD is $O(mnk)$. We will discuss about how to speed up the algorithm to compute SVD in Chapter 2.

QR factorization is also a classical matrix decomposition. The definition of QR factorization is as follows.

Definition 1.1.6 (full QR factorization) *Given a matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$*, there exists*

a decomposition,

$$\mathbf{A} = \mathbf{QR},$$

where $\mathbf{Q}$ *is an* $m \times m$ *orthogonal matrix and* $\mathbf{R}$ *is an* $m \times n$ *upper triangular matrix.*

If $m \geq n$, the matrix $\mathbf{R}$ can be partitioned to an $n \times n$ matrix $\mathbf{R}_1$ and an $(m-n) \times n$ matrix with zeros, $\mathbf{R} = \left[\mathbf{R}_1^{\mathrm{T}}, \mathbf{0}\right]^{\mathrm{T}}$, $\mathbf{Q}$ can be written as $[\mathbf{Q}_1, \mathbf{Q}_2]$ correspondingly. In this case, $\mathbf{A} = \mathbf{Q}_1\mathbf{R}_1$, and it is called reduced QR factorization.

Backward stable methods to compute QR factorization include Householder method and Givens' method. The computation cost of QR factorization is $O(mn^2)$. The detailed algorithm and comparison in precise flop-count and stability of these methods can be found in [18].

QR factorization is powerful in the applications of solving linear systems, finding least square approximations, etc. [38]. We will also use QR factorization for our application in Chapter 4.

Circulant matrix will be used often in later chapters, so we provide its definition here.

Definition 1.1.7 *The circulant matrix of a vector* $\mathbf{a} \in \mathbb{R}^n$ *is*

$$\texttt{circulant(a)} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ a_{n-1} & a_0 & \cdots & a_{n-3} & a_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_2 & a_3 & \cdots & a_0 & a_1 \\ a_1 & a_2 & \cdots & a_{n-1} & a_0 \end{bmatrix}.$$

1.2 Tensor Computation Preliminaries

A tensor is a multi-way array, and the order of the tensor is the number of dimensions of this array. For example, a zero order tensor is a scalar, a first order tensor is a vector, a second order tensor is a matrix, and a third order tensor is a rectangular box, see Figure 1.1. Typically, the tensor with the order greater than or equal

to 3 is called higher-order tensor. Many kinds of datasets are inherently higher-order tensors. For instance, a color image is a third order tensor, and its third index component represents the intensities on red, green, and blue scales [39]. As an another example, the color video is a fourth order tensor, and its fourth index component represents the time sequence of the video clips [93]. Applications of tensor-based methods can be found in the literature: see for example [23, 28, 41, 75, 81, 87, 93]. Although the methods are different, the results show that retainment of the multi-way structure inherent in the data instead of its reshaping to vectors or matrices provides considerable improvement. Therefore, in this thesis, we choose to represent data and operators in their natural multi-way environment.

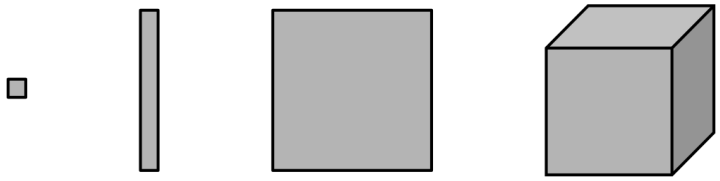


Figure 1.1: Zero order, first order, second order and third order tensor examples.

The history of tensor computation can be traced back to 1927 [43, 44]. The well-known CANDECOMP/PARAFAC (CP) decomposition was proposed by Hitchcock [43], and the CP decomposition is a sum of multiway outer products of vectors. Computing its best rank-k approximation is highly non-trivial, and the best rank-k approximation of CP decomposition may even not exist without extra assumptions on the properties of the tensor. Tucker decomposition was proposed by Tucker in 1963 [84], which is an alternative to CP decomposition. For a third order tensor, the Tucker decomposition can be computed in polynomial time. A best rank $(k_1, k_2, \ldots, k_n)$ factorization can always be computed by an iterative algorithm. To compute an orthogonal Tucker decomposition, one can compute the HOSVD [20]. However, the best rank $(k_1, k_2, \ldots, k_n)$ factorization cannot be obtained by truncating the HOSVD. The papers [19, 53, 81] are good references to learn the detailed development and history of tensor computation and the applications of tensor computation. For a basic review of tensor computation, Chapter 12 of [35] and Chapter

1 of [40] are good references.

More recently, Kilmer and Martin proposed a new framework of tensor computation in 2011 [51]. They presented the concept of a tensor-tensor product (t-product) with a suitable algebraic structure such that classical matrix-like factorizations are possible. This new tensor framework has been used in many applications, see [27, 41, 42, 50, 76, 82, 93].

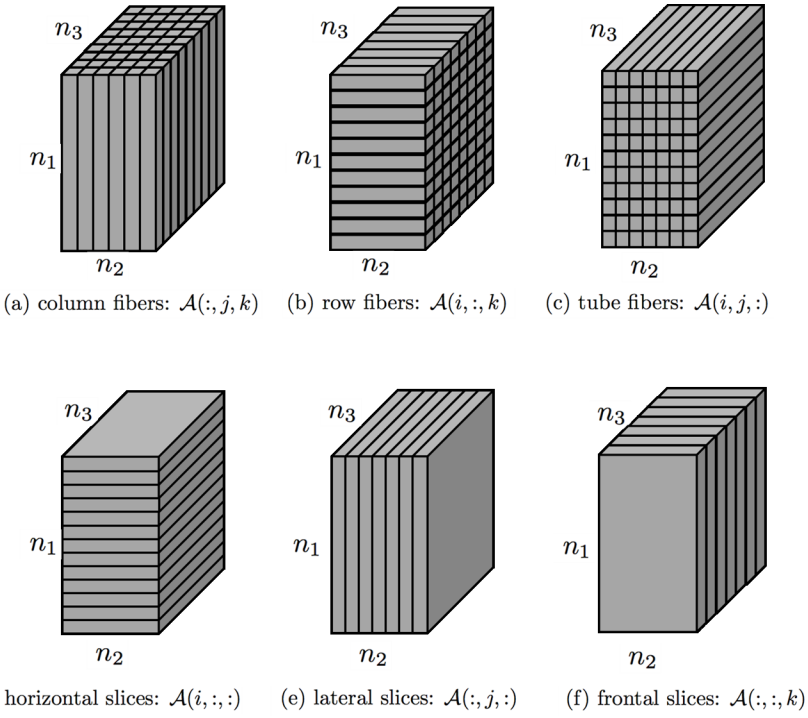
The reasons we will focus on t-product based operators and decompositions are as follows.

- Substantial effort in this thesis is related to a compressed representation of information to a certain degree, and truncating the tensor SVD (t-SVD) defined on t-product give a compressed result that is optimal in the Frobenius norm.
- The t-SVD has been shown to have superior compression characteristics relative to the Tucker decomposition in some applications, such as compression and facial recognition [42].
- There are well-defined mathematical concepts [32, 50, 64], such as transpose, identity, QR factorization, etc. It is convenient to use for building mathematical models.
- The t-product based operators can be computed in parallel in a straight forward way.

To introduce the t-product-based tensor operators, let us first give the notation for tensors. We will focus on third order tensors $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, although much of what we present can be extended to higher order tensors [64].

Each entry of the tensor $\mathcal{A}$ is denoted by Matlab indexing notation, i.e., $\mathcal{A}(i, j, k)$. A fiber of tensor $\mathcal{A}$ is a one-dimensional array defined by fixing two indices. As shown in Figure 1.2, $\mathcal{A}(:, j, k)$ is the $(j, k)^{th}$ column fiber, $\mathcal{A}(i, :, k)$ is the $(i, k)^{th}$ row fiber, and $\mathcal{A}(i, j, :)$ is the $(j, k)^{th}$ tube fiber. A slice of tensor $\mathcal{A}$ is a two-dimensional array defined by fixing one index: $\mathcal{A}(i, :, :)$ is the i^{th} horizontal slice; $\mathcal{A}(:, j, :)$ is the j^{th}

lateral slice; and $\mathcal{A}(:,:,k)$ is the k^{th} frontal slice. For convenience, $\mathcal{A}(:,:,k)$ is written as $\mathcal{A}^{(k)}$.



(a) column fibers: $\mathcal{A}(:,j,k)$ (b) row fibers: $\mathcal{A}(i,:,k)$ (c) tube fibers: $\mathcal{A}(i,j,:)$

(d) horizontal slices: $\mathcal{A}(i,:,:)$ (e) lateral slices: $\mathcal{A}(:,j,:)$ (f) frontal slices: $\mathcal{A}(:,:,k)$

Figure 1.2: Fibers and slices of an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$

.

A third order tensor $\mathcal{A}$ can be seen as an $n_1 \times n_2$ array of tube fibers, each of size $1 \times 1 \times n_3$. The t-product of two tube fibers is defined as their circular convolution, so the t-product between two tensors can be defined as in Definition 1.2.1.

Definition 1.2.1 (t-product) *[51] Let $\mathcal{A}$ be an $n_1 \times n_2 \times n_3$ tensor and $\mathcal{B}$ be an $n_2 \times n_4 \times n_3$ tensor. The t-product of $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{C} = \mathcal{A} * \mathcal{B}$, is an $n_1 \times n_4 \times n_3$ tensor*

$$\mathcal{C}(i,j,:) = \sum_{k=1}^{n_2} \mathcal{A}(i,k,:) * \mathcal{B}(k,j,:) = \sum_{k=1}^{n_2} \mathcal{A}(i,k,:) \circ \mathcal{B}(k,j,:),$$

where $*$ denotes the t-product and $\circ$ denotes the circular convolution.

Because the circular convolution of two tube fibers can be computed by discrete Fourier transform, the t-product can be alternatively computed in the Fourier domain, as shown in Algorithm 1.

Algorithm 1 t-product [51]

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$

Output: An $n_1 \times n_4 \times n_3$ tensor $\mathcal{C}$, $\mathcal{C} = \mathcal{A} * \mathcal{B}$

$\hat{\mathcal{A}} \leftarrow \texttt{fft}(\mathcal{A}, [\,], 3)$;

$\hat{\mathcal{B}} \leftarrow \texttt{fft}(\mathcal{B}, [\,], 3)$;

for $i = 1$ to n_3 **do**

$\hat{\mathcal{C}}^{(i)} = \hat{\mathcal{A}}^{(i)} \hat{\mathcal{B}}^{(i)}$;

end for

$\mathcal{C} \leftarrow \texttt{ifft}(\hat{\mathcal{C}}, [\,], 3)$

There is an equivalent way to define t-product we use in later chapters to better present ideas or proofs. Let us begin with the definitions of `circ`, `unfold`, and `fold`.

Definition 1.2.2 (`circ`) *[51] Let $\mathcal{A}$ be an $n_1 \times n_2 \times n_3$ tensor with $n_1 \times n_2$ frontal slices $\mathcal{A}^{(1)}, \mathcal{A}^{(2)}, \cdots, \mathcal{A}^{(n_3)}$. Then,*

$$\texttt{circ}(\mathcal{A}) = \begin{bmatrix} \mathcal{A}^{(1)} & \mathcal{A}^{(n_3)} & \mathcal{A}^{(n_3-1)} & \cdots & \mathcal{A}^{(2)} \\ \mathcal{A}^{(2)} & \mathcal{A}^{(1)} & \mathcal{A}^{(n_3)} & \cdots & \mathcal{A}^{(3)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathcal{A}^{(n_3)} & \mathcal{A}^{(n_3-1)} & \cdots & \mathcal{A}^{(2)} & \mathcal{A}^{(1)} \end{bmatrix},$$

is a block circulant matrix of size $n_1 n_3 \times n_2 n_3$.

Definition 1.2.3 (unfold and fold) *[51] Let $\mathcal{A}$ be an $n_1 \times n_2 \times n_3$ tensor with $n_1 \times n_2$ frontal slices $\mathcal{A}^{(1)}, \cdots, \mathcal{A}^{(n_3)}$. Then,*

$$\texttt{unfold}(\mathcal{A}) = \begin{bmatrix} \mathcal{A}^{(1)} \\ \mathcal{A}^{(2)} \\ \vdots \\ \mathcal{A}^{(n_3)}, \end{bmatrix}$$

and $\texttt{fold}(\texttt{unfold}(\mathcal{A})) = \mathcal{A}$.

Example (1.2.5) shows how to compute t-product of third-order tensors using

Definition 1.2.4 .

Definition 1.2.4 *[51] Suppose $\mathcal{A}$ is an $n_1 \times n_2 \times n_3$ tensor and $\mathcal{B}$ is an $n_2 \times n_4 \times n_3$ tensor. Then, the t-product $\mathcal{A} * \mathcal{B}$ is an $n_1 \times n_4 \times n_3$ tensor*

$$\mathcal{A} * \mathcal{B} = \texttt{fold}(\texttt{circ}(\mathcal{A}) \cdot \texttt{unfold}(\mathcal{B})).$$

Example 1.2.5 [51] Suppose $\mathcal{A}$ is an $n_1 \times n_2 \times 3$ tensor and $\mathcal{B}$ is an $n_2 \times n_4 \times 3$ tensor. Then,

$$\begin{aligned} \mathcal{A} * \mathcal{B} &= \texttt{fold}(\texttt{circ}(\mathcal{A}) \cdot \texttt{unfold}(\mathcal{B})) \\ &= \texttt{fold}\left(\begin{bmatrix} \mathcal{A}^{(1)} & \mathcal{A}^{(3)} & \mathcal{A}^{(2)} \\ \mathcal{A}^{(2)} & \mathcal{A}^{(1)} & \mathcal{A}^{(3)} \\ \mathcal{A}^{(3)} & \mathcal{A}^{(2)} & \mathcal{A}^{(1)} \end{bmatrix} \begin{bmatrix} \mathcal{B}^{(1)} \\ \mathcal{B}^{(2)} \\ \mathcal{B}^{(3)} \end{bmatrix}\right) \end{aligned}$$

is an $n_1 \times n_4 \times 3$ tensor.

Since $\texttt{circ}(\mathcal{A})$ is a block circulant matrix, it can be block diagonalized as

$$(\mathbf{F}_3 \otimes \mathbf{I}_{n_1})\texttt{circ}(\mathcal{A})(\mathbf{F}_3^{\mathrm{H}} \otimes \mathbf{I}_{\mathtt{n}_2}) = \begin{bmatrix} \hat{\mathcal{A}}^{(1)} & & \\ & \hat{\mathcal{A}}^{(2)} & \\ & & \mathcal{A}^{(3)} \end{bmatrix},$$

where $\otimes$ denotes the Kronecker product, H denotes the conjugate transpose, $\mathbf{F}_3$ is a 3×3 normalized discrete Fourier transform (DFT) matrix and $\hat{\mathcal{A}}^{(i)}$ is the i^{th} frontal slice of $\hat{\mathcal{A}}$, $\hat{\mathcal{A}} = \texttt{fft}(\mathcal{A}, [\,], 3)$.

$$\begin{aligned} &\texttt{circ}(\mathcal{A})\texttt{unfold}(\mathcal{B}) \\ &= (\mathbf{F}_3^{\mathrm{H}} \otimes \mathbf{I}_{n_1})(\mathbf{F}_3 \otimes \mathbf{I}_{n_1})\texttt{circ}(\mathcal{A})(\mathbf{F}_3^{\mathrm{H}} \otimes \mathbf{I}_{\mathtt{n}_2})(\mathbf{F}_3 \otimes \mathbf{I}_{\mathtt{n}_2})\texttt{unfold}(\mathcal{B}) \\ &= (\mathbf{F}_3^{\mathrm{H}} \otimes \mathbf{I}_{n_1}) \begin{bmatrix} \hat{\mathcal{A}}^{(1)} & & \\ & \hat{\mathcal{A}}^{(2)} & \\ & & \mathcal{A}^{(3)} \end{bmatrix} \begin{bmatrix} \hat{\mathcal{B}}^{(1)} \\ \hat{\mathcal{B}}^{(2)} \\ \mathcal{B}^{(3)} \end{bmatrix} \end{aligned}$$

$$= (\mathbf{F}_3^{\mathrm{H}} \otimes \mathbf{I}_{n_1}) \begin{bmatrix} \hat{\mathcal{A}}^{(1)}\hat{\mathcal{B}}^{(1)} \\ \hat{\mathcal{A}}^{(2)}\hat{\mathcal{B}}^{(2)} \\ \hat{\mathcal{A}}^{(3)}\mathcal{B}^{(3)} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{F}_3^{\mathrm{H}}\hat{\mathcal{A}}^{(1)}\hat{\mathcal{B}}^{(1)} \\ \mathbf{F}_3^{\mathrm{H}}\hat{\mathcal{A}}^{(2)}\hat{\mathcal{B}}^{(2)} \\ \mathbf{F}_3^{\mathrm{H}}\hat{\mathcal{A}}^{(3)}\mathcal{B}^{(3)} \end{bmatrix},$$

where $\mathcal{B}^{(i)}$ is the i^{th} frontal slice of $\hat{\mathcal{B}}$, $\hat{\mathcal{B}} = \texttt{fft}(\mathcal{B}, [\,], 3)$. Therefore, $\texttt{fold}(\texttt{circ}(\mathcal{A}) \cdot \texttt{unfold}(\mathcal{B}))$ can be computed by Algorithm 1. ◇

Next, we will introduce several definitions based on t-product from [51] and [42] that will be necessary for the rest of the thesis.

Definition 1.2.6 (Identity tensor) *[51] The $n_1 \times n_2 \times n_3$ identity tensor $\mathcal{I}$ is the tensor whose first frontal slice is the $n_1 \times n_2$ identity matrix, and whose other frontal slices are all zeros.*

Definition 1.2.7 (Transpose) *[51] If $\mathcal{A}$ is an $n_1 \times n_2 \times n_3$ tensor, then $\mathcal{A}^{\mathrm{T}}$ is an $n_2 \times n_1 \times n_3$ tensor obtained by transposing each of the frontal slices and then reversing the order of transposed frontal slices 2 through n_3 (see Figure 1.3).*

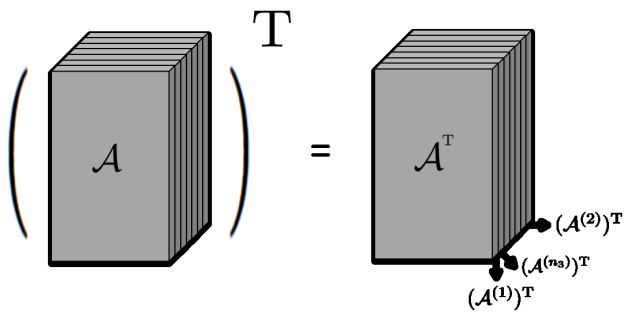


Figure 1.3: Transpose of an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$

.

Definition 1.2.8 (Orthogonality) *[51] An $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$ is called orthogonal, if the t-product of $\mathcal{A}^{\mathrm{T}}$ and $\mathcal{A}$ is equal to the identity tensor, i.e.,*

$$\mathcal{A}^{\mathrm{T}} * \mathcal{A} = \mathcal{I}.$$

Definition 1.2.9 (f-Diagonal) *[51] An $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$ is called f-diagonal, if each frontal face of $\mathcal{A}$ is diagonal.*

Definition 1.2.10 (f-upper triangular) *[51] An $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$ is called f-upper triangular, if each frontal face of $\mathcal{A}$ is upper triangular.*

We now introduce the t-QR, pivoted t-QR, t-SVD, and truncated t-SVD, which builds on the above operations of tensors.

Definition 1.2.11 (t-QR factorization) *[51] Given an $n_1 \times n_2 \times n_3$ tensor $\mathbf{A}$, the t-QR factorization of $\mathcal{A}$ is*

$$\mathcal{A} = \mathcal{Q} * \mathcal{R},$$

where tensor $\mathcal{Q}$ is orthogonal and $\mathcal{R}$ is f-upper triangular.

According to the definition and algorithm of t-product, the procedure to compute t-QR is shown in Algorithm 2.

Algorithm 2 t-QR factorization [51]

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$

Output: An $n_1 \times n_1 \times n_3$ tensor $\mathcal{Q}$, and an $n_1 \times n_2 \times n_3$ tensor $\mathcal{R}$

$\hat{\mathcal{A}} \leftarrow \texttt{fft}(\mathcal{A}, [\,], 3)$;

for $i = 1$ to n_3 **do**

  $\hat{\mathcal{A}}^{(i)} = \hat{\mathcal{Q}}^{(i)} \hat{\mathcal{R}}^{(i)}$;

end for

$\mathcal{Q} \leftarrow \texttt{ifft}(\hat{\mathcal{Q}}, [\,], 3)$

$\mathcal{R} \leftarrow \texttt{ifft}(\hat{\mathcal{R}}, [\,], 3)$

Definition 1.2.12 (Permutation tensor) *[42] The $n_1 \times n_2 \times n_3$ permutation tensor $\mathcal{P}$ is the tensor whose entries consist of only zeros and $1's$ and $\mathcal{P}^{\mathrm{T}} * \mathcal{P} = \mathcal{P} * \mathcal{P}^{\mathrm{T}} = \mathcal{I}$.*

Theorem 1.2.13 (Pivoted t-QR factorization) *[42] Given an $n_1 \times n_2 \times n_3$ tensor $\mathbf{A}$, it can be factorized as*

$$\mathcal{A} * \mathcal{P} = \mathcal{Q} * \mathcal{R},$$

where tensor $\mathcal{Q}$ is orthogonal, $\mathcal{R}$ is f-upper triangular, and $\mathcal{P}$ is a permutation tensor.

Definition 1.2.14 *[51] Let $\mathcal{A}$ be an $n_1 \times n_2 \times n_3$ tensor. The t-SVD of $\mathcal{A}$ is*

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^{\mathrm{T}}$$

where $\mathcal{U} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$, $\mathcal{V} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ are orthogonal, and $\mathcal{S}_k \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is a f-diagonal tensor.

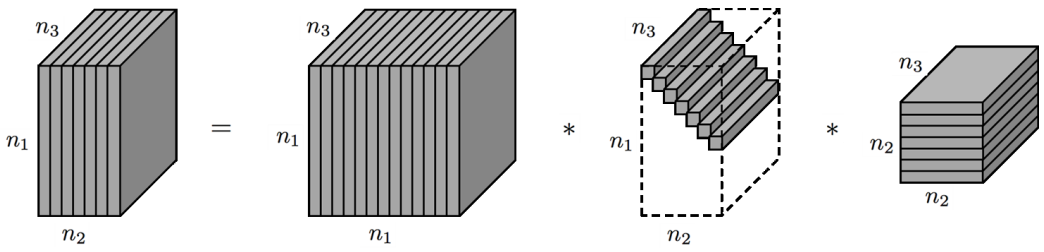


Figure 1.4: The t-SVD of an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$.

Definition 1.2.15 *[51] Given a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, define the truncated t-SVD of $\mathcal{A}$ as*

$$\mathcal{A}_k = \mathcal{U}_k * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}}$$

where k is a target truncation term, $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$ are orthogonal, and $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$ is a f-diagonal tensor.

The procedure to compute the k-term truncated SVD is given in Algorithm 3. For a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, computing a k-term truncated t-SVD takes $O(n_1 n_2 n_3 k)$ flops. Detailed flop-count see Table 1.1. For a matrix $\mathbf{A} \in \mathbb{R}^{n_1 n_2 \times n_3}$, computing a

k-term truncated SVD takes $O(n_1 n_2 n_3 k)$ flops as well. *However, Algorithm 3 can be computed in parallel over the frontal slices on a cluster*, whereas typical algorithms used for the truncated SVD of a matrix cannot be computed in parallel. We will show numerical results in Chapter 2.

Algorithm 3 k-term truncated t-SVD [51]

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and target truncation term k

Output: $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$, and $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$

$\hat{\mathcal{A}} \leftarrow \texttt{fft}(\mathcal{A}, [\,], 3)$;

for $i = 1$ to n_3 **do**

$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = svd(\mathcal{A}^{(i)})$;

Form $\mathbf{U}_k$, $\mathbf{S}_k$, $\mathbf{V}_k$ by truncating $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$ with target truncation term k;

$\hat{\mathcal{U}}_k^{(i)} = \mathbf{U}_k$;

$\hat{\mathcal{S}}_k^{(i)} = \mathbf{S}_k$;

$\hat{\mathcal{V}}_k^{(i)} = \mathbf{V}_k$;

end for

$\mathcal{U}_k \leftarrow \texttt{ifft}(\hat{\mathcal{U}}_\texttt{k}, [\,], 3)$;

$\mathcal{S}_k \leftarrow \texttt{ifft}(\hat{\mathcal{S}}_\texttt{k}, [\,], 3)$;

$\mathcal{V}_k \leftarrow \texttt{ifft}(\hat{\mathcal{V}}_\texttt{k}, [\,], 3)$;

Table 1.1: Operation Counts of t-SVD

Steps	Operation Counts
1. $\hat{\mathcal{A}} = \texttt{fft}(\mathcal{A}, [\,], 3)$	$O(n_1 n_2 n_3 log(n_3))$
2. $[\mathcal{U}, \mathcal{S}, \mathcal{V}] = svds(\hat{\mathcal{A}}^{(i)})$	$O(n_1 n_2 k)$
3. For loop $i = 1 : n_3$, repeat step 2	$n_3 \cdot O(n_1 n_2 k)$
4. $\texttt{ifft}(\mathcal{U}, [\,], 3)$	$O(n_1 n_3 k log(n_3))$
5. $\texttt{ifft}(\mathcal{S}, [\,], 3)$	$O(k^2 n_3 log(n_3))$
6. $\texttt{ifft}(\mathcal{V}, [\,], 3)$	$O(k^2 n_3 log(n_3))$

The optimality of the error in the truncated t-SVD is presented below, which is a generalization of the well-known result of optimality of the truncated SVD in the Frobenius norm, see Theorem 1.1.5.

Definition 1.2.16 *[53] Given a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, the Frobenius norm of tensor $\mathcal{A}$ is the square root of the sum of the squares of all its entries, i.e.*

$$\|\mathcal{A}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3}(\mathcal{A}(i,j,k))^2}.$$

Here, and henceforth, we will use the short-hand notation $\sum_{j>k}$ to represent $\sum_{j=k+1}^{\min\{n_1,n_2\}}$ for clarity.

Theorem 1.2.17 *[51] Given a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{A}_k = \arg\min_{\tilde{\mathcal{A}} \in \mathcal{M}} \|\mathcal{A} - \tilde{\mathcal{A}}\|_{\mathrm{F}}$, where $\mathcal{M} = \{\mathcal{C} = \mathcal{X} * \mathcal{Y} \mid \mathcal{X} \in \mathbb{R}^{n_1 \times k \times n_3}, \mathcal{Y} \in \mathbb{R}^{k \times n_2 \times n_3}\}$. Therefore, $\|\mathcal{A} - \mathcal{A}_k\|_{\mathrm{F}}$ is the theoretical minimal error, given by*

$$\|\mathcal{A} - \mathcal{A}_k\|_{\mathrm{F}} = \left(\frac{1}{n_3}\sum_{i=1}^{n_3}\sum_{j>k}(\hat{\sigma}_j^{(i)})^2\right)^{1/2}, \tag{1.1}$$

where $\hat{\sigma}_j^{(i)} \equiv \hat{\mathcal{S}}(j,j,i)$ is the i^{th} component of $\mathrm{fft}(\mathcal{S}(\mathrm{j},\mathrm{j},:),[\,],3)$.

In the thesis, we will also need to use two operators, `squeeze` and `twist` (see Figure 1.5), that could go back and forth between elements in $\mathbb{R}^{n_1 \times 1 \times n_2}$ and $\mathbb{R}^{n_1 \times n_2}$ [50].

The $\texttt{squeeze}(\mathcal{A})$ produces a matrix $\mathbf{A}$ such that

$$\mathcal{A}(i,j) = \mathbf{A}(i,1,j).$$

The $\texttt{twist}(\mathcal{A})$ operation is the inverse of squeeze, i.e.

$$\texttt{twist}(\texttt{squeeze}(\mathcal{A})) = \mathcal{A}.$$

1.3 Outline of the Thesis

The structure of the thesis is as follows. After introducing the preliminary knowledge in Chapter 1, Chapter 2 presents a novel randomized tensor singular value decomposition to compute the approximated t-SVD. Chapter 3 presents a new tensor-based

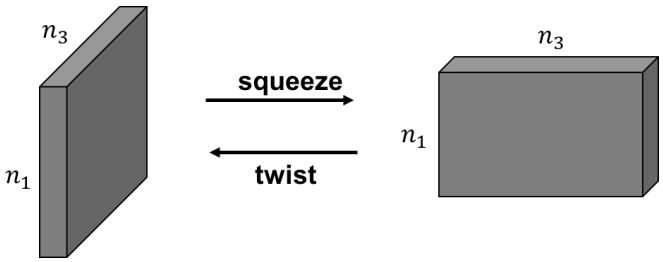


Figure 1.5: The operators `squeeze` and `twist`.

model reduction method, tensor proper orthogonal decomposition, to solve dynamical systems. Chapter 4 illustrates a new method to select essential frames for the application of multi-frame blind deconvolution. Chapter 5 introduces a new tensor-based model to enhance the resolution of multiple video clips simultaneously using the information of their neighbor frames. Final discussions, conclusions, and future works are given in Chapter 6.

Chapter 2

Randomized Tensor Singular Value Decomposition

In this Chapter, we will introduce a novel method that extends a well-known randomized matrix method to the tensor Singular Value Decomposition (t-SVD). We will start with the motivation and background in Section 2.1 and review the matrix-based randomized SVD in Section 2.2. Our main method and algorithms will be discussed in Section 2.3. Numerical results and summary will be provided in Section 2.4 and Section 2.5.

2.1 Motivation and Background

As we discussed in Chapter 1, matrix decompositions are essentially important in both theory and practice. However, in this era of "big data," it is not uncommon for the size of a matrix operation or a dataset to reach the scale of petabytes or even exabytes. By 2013, for example, Facebook already used 1.5 petabytes to store about 10 billion photos, and Netflix had used 3.14 petabytes to store available shows and movies [86]. In the field of quantum chromodynamics, the size of matrix operator is on the order of several millions, or even billions [31]. On the one hand, there still seems to be a push to obtain ever more information by collecting more data. On the other hand, current data analysis and scientific computing methods are continually challenged by the expanding sizes of the models and datasets.

In data analysis and scientific computing, almost all of the methods rely on matrix algorithms [89]. In particular, the low-rank matrix approximation,

$$\mathbf{A}_{m\times n} \approx \mathbf{B}_{m\times k}\mathbf{C}_{k\times n}, \tag{2.1}$$

where $k < \min\{m, n\}$, is used often, because it allows us to store or analyze the matrix $\mathbf{A}$ by the smaller-size factor matrices $\mathbf{B}$ and $\mathbf{C}$ instead of the full matrix, which is more efficient when k is much less than $\min\{m, n\}$. Moreover, these smaller matrices can provide some specific information about the original matrix.

In Chapter 1, we mentioned that truncating the matrix singular value decomposition to k terms provides the optimal rank-k approximation to a matrix in both the matrix 2-norm and Frobenius norm, and the algorithms for computing the approximation are numerically robust. It has many applications, but the cost of accurately computing the truncated matrix SVD can be prohibitively expensive, making it unsuitable for some current large-scale applications [78].

Therefore, much work has been devoted to the development of randomized algorithms for computing low-rank matrix approximations but which are cheaper to compute. As a trade-off, one gives up the optimality property at some level for nearly optimal results.

For computing low-rank approximations more efficiently, randomized algorithms have been proposed in recent years. Randomized algorithms are based on random sampling and random projection. The main idea of random sampling methods is to construct a smaller matrix which is close to the original matrix and contains the information from selected and rescaled columns or rows of the original matrix, see [21]. The idea of random projection methods is to project the original matrix to a lower dimensional space by multiplying from the right by a random projection matrix, see [37]. The work in [63] summarizes different ways of constructing the random projection matrix are used for different purposes, see also [3, 5, 6, 17, 30].

Randomized algorithms work efficiently in the context of a matrix, but as we discussed in Chapter 1, not all of the datasets are natively represented in matrix form at the outset. It is therefore natural to extend randomized algorithms to the tensor computation. The authors in [22] appear to have pioneered the generalization of random sampling methods to tensors. Specifically, they extended random sampling methods to the Tucker decomposition, and provided a guide to the theoretical analysis for the tensor-based decomposition via random sampling methods. In [83],

the authors provide numerical examples of Tucker decomposition with the random sampling method. A literature search also reveals attempts to extend the random sampling approach to tensor-based on CP decomposition and Tucker decomposition [9, 10, 77, 88].

In this section, we extend a well-known random projection method, the randomized SVD (r-SVD) [37], to the t-SVD for tensors using the algebra induced by the t-product. The motivation for focusing our efforts on the randomization of the t-SVD are as follows:

- Theoretical and computational advantages provided by the t-SVD. For example, we can utilize the optimal property we introduced in Theorem 1.2.17, and we can exploit parallelization for computing the t-SVD.
- Under the t-product, there are well defined concepts of orthogonality, identity and orthogonal projections, QR factorizations, and the like [32, 50, 64].
- In many applications, such as compression and facial recognition, the t-SVD has been shown to have superior compression characteristics [42] relative to the Tucker decomposition. When they use some storage, the t-SVD has better performance in term of the recognition rate.

2.2 Randomized Singular Value Decomposition

The randomized Singular Value Decomposition (referred to as r-SVD), was proposed in a series of papers published over the last decade (see e.g., [59, 90]) and was popularized by the review paper [37]. The first step in computing the r-SVD is generating several Gaussian random vectors stored in the columns of $\mathbf{W} \in \mathbb{R}^{n\times(k+p)}$ that are, with high probability, linearly independent. Here, k is the desired target truncation term of the approximation and p is a non-negative integer oversampling parameter. The matrix $\mathbf{Y} := \mathbf{AW} \in \mathbb{C}^{m\times(k+p)}$ thus contains random linear combinations of the columns of $\mathbf{A}$. By computing the QR factorization of $\mathbf{Y}$, we can obtain a well-approximated basis $\mathbf{Q}$, whose columns form an orthonormal basis for the range

of $\mathbf{Y}$. If $\mathbf{A}$ has rapidly decaying singular values, the columns of $\mathbf{Q}$ are the dominant left singular vectors of $\mathbf{A}$, i.e. $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^{\mathrm{H}}\mathbf{A}$. Thus, one computes $\mathbf{B} := \mathbf{Q}^{\mathrm{H}}\mathbf{A}$ followed by the compact SVD of $\mathbf{B}$, $\mathbf{B} = \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^{\mathrm{H}}$. The estimated desired singular values of $\mathbf{A}$ are the diagonals of $\mathbf{S}$, while $\mathbf{Q}\mathbf{U}_k$ gives the estimated left singular vectors of $\mathbf{A}$. Algorithm 4 summarizes the procedure described above.

Algorithm 4 r-SVD method [37]

Input: $\mathbf{A} \in \mathbb{C}^{m\times n}$, target truncation term k, and oversampling parameter p

Output: $\mathbf{U}_k \in \mathbb{C}^{n\times k}$, $\mathbf{S}_k \in \mathbb{C}^{k\times k}$, and $\mathbf{V}_k \in \mathbb{C}^{n\times k}$

Generate a Gaussian random matrix $\mathbf{W} \in \mathbb{R}^{n\times(k+p)}$

Form a matrix $\mathbf{Y} = \mathbf{A}\mathbf{W}$

Construct matrix $\mathbf{Q} \in \mathbb{C}^{n\times(k+p)}$ via the QR factorization of $\mathbf{Y}$

Form $\mathbf{B} \in \mathbb{C}^{(k+p)\times n}$, $\mathbf{B} = \mathbf{Q}^{\mathrm{H}}\mathbf{A}$

Compute $\mathbf{B} = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^{\mathrm{H}}$

Set $\mathbf{U} = \tilde{\mathbf{U}}(:, 1{:}k)$, $\mathbf{S}_k = \tilde{\mathbf{S}}(1{:}k, 1{:}k)$, $\mathbf{V}_k = \tilde{\mathbf{V}}(:, 1{:}k)$

Form $\mathbf{U}_k = \mathbf{Q}_k\mathbf{U}$.

When $\mathbf{A}$ is dense and of size $n \times n$, this algorithm can take $O(n^2k)$ flops. For more details, see [90]. The expected error in the low-rank approximation measured using the Frobenius norm can be bounded, as a result below shows.

Theorem 2.2.1 *[37] Given a matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ and a Gaussian random matrix $\mathbf{W} \in \mathbb{R}^{n\times(k+p)}$. Suppose that $\mathbf{Q}$ is computed as in Algorithm 4, then the expected approximation error is*

$$\mathbb{E}\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^{\mathrm{H}}\mathbf{A}\|_{\mathrm{F}}^2 \leq \left(1 + \frac{k}{p-1}\right)\left(\sum_{j=k+1}^{\min\{m,n\}} \sigma_j^2\right) \tag{2.2}$$

where σ_j is the j^{th} singular value of $\mathbf{A}$, p is an oversampling parameter, and $\mathbb{E}$ denotes the expected value.

Proof: The proof follows readily from [37, Theorem 10.5]. □

From the inequality (2.2), the value of $\mathbb{E}\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^{\mathrm{H}}\mathbf{A}\|_{\mathrm{F}}^2$ depends on $\sum_{j>k}\sigma_j^2$. When the singular values of $\mathbf{A}$ decay gradually, $\sum_{j>k}\sigma_j^2$ can be large, and therefore the low-rank approximation as computed above may not be sufficiently accurate. In this situation, Algorithm 5, which is based on subspace iteration, may be preferred.

Algorithm 5 r-SVD method with subspace iteration [37]

Input: $\mathbf{A} \in \mathbb{C}^{m\times n}$, target truncation term k, a parameter q, and an oversampling parameter p

Output: An orthogonal column basis $\mathbf{Q}$ of $\mathbf{Y}$

Generate a Gaussian random matrix $\mathbf{W} \in \mathbb{R}^{n\times(k+p)}$

Form a matrix $\mathbf{Y_0} = \mathbf{A}\mathbf{W}$ and compute the QR factorization of $\mathbf{Y}_0 = \mathbf{Q}_0\mathbf{R}_0$

for $i = 1$ to q **do**

Form $\tilde{\mathbf{Y}}_i = \mathbf{A}^{\mathrm{H}}\mathbf{Q}_{i-1}$ and compute the QR factorization of $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{Q}}_i\tilde{\mathbf{R}}_i$

Form $\mathbf{Y}_i = \mathbf{A}\tilde{\mathbf{Q}}_i$ and compute the QR factorization of $\mathbf{Y}_i = \mathbf{Q}_i\mathbf{R}_i$

end for

Form a matrix $\mathbf{Q} = \mathbf{Q}_q$

Assume that k is the target truncation term, and the singular value gap $\tau_k \equiv \frac{\sigma_{k+1}}{\sigma_k}$. Theorem 2.2.2 provides the error bound of Algorithm 5 in Frobenius norm.

Theorem 2.2.2 *[92] Let $\mathbf{A} \in \mathbb{C}^{m\times n}$ and $\mathbf{W} \in \mathbb{R}^{n\times(k+p)}$ be a Gaussian random matrix with $p \geq 2$ being the oversampling parameter. Suppose $\mathbf{Q}$ is obtained from Algorithm 5, then*

$$\mathbb{E}\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^{\mathrm{H}}\mathbf{A}\|_{\mathrm{F}}^2 \leq \left(1 + \frac{k}{p-1}\tau_k^{4q}\right)\left(\sum_{j=k+1}^{r}\sigma_j^2\right),$$

where k is a target truncation term, r is the rank of $\mathbf{A}$, q is the number of iterations, σ_j is the j^{th} singular value of $\mathbf{A}$, and $\tau_k = \sigma_{k+1}/\sigma_k \ll 1$ is the singular value gap.

Proof: See [36, Theorem 5.7] and Appendix of [92]. □

The error due to the randomized subspace iteration is similar to Theorem 2.2, except for the term τ_k^{4q}. As the number of subspace iterations q increases, the term

$\frac{k}{p-1}\tau_k^{4q}$ decreases and the subspace iteration approaches to the optimal error of the SVD. When $q = 0$, Theorem 2.2.2 presents the same result as Theorem 2.2.1.

2.3 Randomized Tensor Singular Value Decomposition

In this section, we will present the rt-SVD method, which extends the matrix r-SVD method to the t-SVD. The goal of the rt-SVD method is to find a good approximate factorization of tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{U}_k * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}}$. There are two main steps. The first step is to find a tensor $\mathcal{Q}$ with orthogonal lateral slices such that

$$\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A} \|_{\mathrm{F}} \leq \epsilon \| \mathcal{A} \|_{\mathrm{F}},$$

and the second step to obtain the approximate factorization of rt-SVD with a small amount of calculation. We outline these steps in Algorithm 6.

Let us give the definition of Gaussian random tensor here that is first defined in [92]. The motivations to define the Gaussian random tensor this way are as follows.

- Generate as few random numbers as possible. It reduces the storage cost for the applications that have to save the Gaussian random tensor for later use.
- Being able to use the conclusions of literature on Gaussian random matrices.

Definition 2.3.1 (Gaussian random tensor) *An $n_1 \times n_2 \times n_3$ tensor $\mathcal{W}$ is called a Gaussian random tensor, if the elements of $\mathcal{W}^{(1)}$ satisfy the standard normal distribution, and the other frontal slices are all zeros.*

The Fourier transform of $\mathcal{W}$ along the 3^{rd} dimension is denoted as $\hat{\mathcal{W}}$ such that every frontal slice is an identical copy of the first slice $\mathcal{W}^{(1)}$, see Figure 2.1.

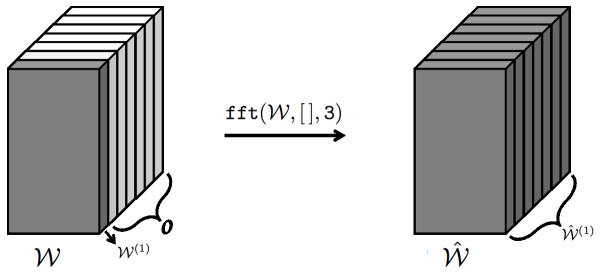


Figure 2.1: A third order Gaussian random tensor and its Fourier transform.

Algorithm 6 rt-SVD, spatial domain presentation

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, target truncation term k, and oversampling parameter p

Output: $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$, and $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$

Generate a Gaussian random tensor $\mathcal{W} \in \mathbb{R}^{n_2 \times (k+p) \times n_3}$

Form a random projection of the tensor $\mathcal{A}$ as $\mathcal{Y} = \mathcal{A} * \mathcal{W}$

Construct the tensor $\mathcal{Q}$ by using t-QR factorization of tensor $\mathcal{Y}$

Form a tensor $\mathcal{B} = \mathcal{Q}^{\mathrm{T}} * \mathcal{A}$, whose size is $(k+p) \times n_2 \times n_3$

Compute t-SVD of $\mathcal{B}$, truncate it with target truncation term k, and obtain $\mathcal{U}$, $\mathcal{S}_k$, and $\mathcal{V}_k$

Form the rt-SVD of $\mathcal{A}$, $\mathcal{A} \approx (\mathcal{Q} * \mathcal{U}) * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}} = \mathcal{U}_k * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}}$.

For the convenience of error analysis, Algorithm 6 can be written based on each frontal slice and using r-SVD for each slice as in Algorithm 7, which allows for the application of the theoretical results for the matrix randomized algorithm to each frontal slice.

Algorithm 7 rt-SVD, Fourier domain implementation

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, target truncation term k, and parameter p

Output: $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$, and $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$

Generate a Gaussian random tensor $\mathcal{W} \in \mathbb{R}^{n_2 \times (k+p) \times n_3}$

$\hat{\mathcal{A}} \leftarrow \texttt{fft}(\mathcal{A}, [\,], 3)$ and $\hat{\mathcal{W}} \leftarrow \texttt{fft}(\mathcal{W}, [\,], 3)$

for $i = 1$ to n_3 **do**

$\hat{\mathcal{Y}}^{(i)} = \hat{\mathcal{A}}^{(i)} \hat{\mathcal{W}}^{(i)}$

$[\hat{\mathcal{Q}}^{(i)}, \hat{\mathcal{R}}^{(i)}] = \texttt{qr}(\hat{\mathcal{Y}}^{(\mathtt{i})}, 0)$

$\hat{\mathcal{B}}^{(i)} = (\hat{\mathcal{Q}}^{(i)})^{\mathrm{H}} \hat{\mathcal{A}}^{(i)}$

$[\hat{\mathcal{U}}^{(i)}, \hat{\mathcal{S}}^{(i)}, \hat{\mathcal{V}}^{(i)}] = \texttt{svd}(\hat{\mathcal{B}}^{(\mathtt{i})})$

$\hat{\mathcal{U}}_k^{(i)} = \hat{\mathcal{Q}}^{(i)} \hat{\mathcal{U}}^{(i)}$; $\hat{\mathcal{S}}_k^{(i)} = \hat{\mathcal{S}}^{(i)}(1:k, 1:k)$; $\hat{\mathcal{V}}_k^{(i)} = \hat{\mathcal{V}}^{(i)}(:, 1:k)$.

end for

$\mathcal{U}_k \leftarrow \texttt{ifft}(\hat{\mathcal{U}}_{\mathtt{k}}, [\,], 3)$; $\mathcal{S}_k \leftarrow \texttt{ifft}(\hat{\mathcal{S}}_{\mathtt{k}}, [\,], 3)$; $\mathcal{V}_k \leftarrow \texttt{ifft}(\hat{\mathcal{V}}_{\mathtt{k}}, [\,], 3)$.

In Theorem 2.3.3, we give the expected error of $\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A} \|_{\mathrm{F}}$ where the tensor $\mathcal{Q}$ is computed by using Algorithm 6 or Algorithm 7.

Lemma 2.3.2 *Given a real $n_1 \times n_2 \times n_3$ tensor$\mathcal{A}$,*

$$\|\mathcal{A}\|_{\mathrm{F}}^2 = \frac{1}{n_3} \sum_{i=1}^{n_3} \|\hat{\mathcal{A}}^{(i)}\|_{\mathrm{F}}^2 \tag{2.3}$$

Proof: According to [51, Equation 3.1],

$$(\mathbf{F}_{n_3} \otimes \mathbf{I}_{n_1})\,\texttt{circ}\,(\mathcal{A})\,(\mathbf{F}_{\mathtt{n_3}}^{\mathrm{H}} \otimes \mathbf{I}_{\mathtt{n_2}}) = \begin{bmatrix} \hat{\mathcal{A}}^{(1)} & & & \\ & \hat{\mathcal{A}}^{(2)} & & \\ & & \ddots & \\ & & & \hat{\mathcal{A}}^{(n_3)} \end{bmatrix},$$

where

$$\texttt{circ}(\mathcal{A}) = \begin{bmatrix} \mathcal{A}^{(1)} & \mathcal{A}^{(n_3)} & \mathcal{A}^{(n_3-1)} & \dots & \mathcal{A}^{(2)} \\ \mathcal{A}^{(2)} & \mathcal{A}^{(1)} & \mathcal{A}^{(n_3)} & \dots & \mathcal{A}^{(3)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathcal{A}^{(n_3)} & \mathcal{A}^{(n_3-1)} & \dots & \mathcal{A}^{(2)} & \mathcal{A}^{(1)} \end{bmatrix}.$$

This equality implies that

$$\left\|(\mathbf{F}_{n_3} \otimes \mathbf{I}_{n_1})\texttt{circ}\,(\mathcal{A})\,(\mathbf{F}_{\mathtt{n_3}}^{\mathrm{H}} \otimes \mathbf{I}_{\mathtt{n_2}})\right\|_{\mathrm{F}}^2 = \sum_{i=1}^{n_3} \left\|\hat{\mathcal{A}}^{(i)}\right\|_{\mathrm{F}}^2.$$

The Frobenius norm is unitarily invariant

$$\left\|(\mathbf{F}_{n_3} \otimes \mathbf{I}_{n_1})\texttt{circ}\,(\mathcal{A})\,(\mathbf{F}_{\mathtt{n_3}}^{\mathrm{H}} \otimes \mathbf{I}_{\mathtt{n_2}})\right\|_{\mathrm{F}}^2 = \|\texttt{circ}\,(\mathcal{A})\|_{\mathrm{F}}^2,$$

therefore

$$\|\texttt{circ}\,(\mathcal{A})\|_{\mathrm{F}}^2 = \sum_{i=1}^{n_3} \left\|\hat{\mathcal{A}}^{(i)}\right\|_{\mathrm{F}}^2. \qquad \square$$

Since $\|\texttt{circ}\,(\mathcal{A})\|_{\mathrm{F}}^2 = n_3\,\|\mathcal{A}\|_{\mathrm{F}}^2$, the desired result follows.

Theorem 2.3.3 *Given an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$ and an $n_2 \times (k+p) \times n_3$ Gaussian random tensor $\mathcal{W}$, if $\mathcal{Q}$ is obtained by the t-QR of $\mathcal{Y} = \mathcal{A} * \mathcal{W}$, then*

$$\mathbb{E}\,\|\,\mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A}\,\|_{\mathrm{F}} \leq \frac{1}{\sqrt{n_3}}\left(1 + \frac{k}{p-1}\right)^{1/2} \Big(\sum_{i=1}^{n_3}\sum_{j>k}(\hat{\sigma}_j^{(i)})^2\Big)^{1/2}.$$

where k is a target truncation term, p is the oversampling parameter, and $\hat{\sigma}_j^{(i)}$ is the i^{th} component of $\texttt{fft}(\mathcal{S}(\mathtt{j},\mathtt{j},:),[\,],3)$.

Proof: Let $\mathcal{R} = \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A}$. According to Lemma 2.3.2,

$$\|\,\mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A}\,\|_{\mathrm{F}}^2 = \|\,\mathcal{R}\,\|_{\mathrm{F}}^2 = \frac{1}{n_3}\sum_{i=1}^{n_3}\|\hat{\mathcal{R}}^{(i)}\|_{\mathrm{F}}^2.$$

By linearity of expectation,

$$\mathbb{E}\,\|\,\mathcal{R}\,\|_{\mathrm{F}}^2 = \frac{1}{n_3}\mathbb{E}\Big(\sum_{i=1}^{n_3}\|\hat{\mathcal{R}}^{(i)}\,\|_{\mathrm{F}}^2\Big) = \frac{1}{n_3}\sum_{i=1}^{n_3}\mathbb{E}(\|\,\hat{\mathcal{R}}^{(i)}\,\|_{\mathrm{F}}^2). \tag{2.4}$$

Then, by the inequality (2.2),

$$\mathbb{E} \parallel \hat{\mathcal{R}}^{(i)} \parallel_{\mathrm{F}}^2 \leq \left(1 + \frac{k}{p-1}\right) \sum_{j>k} (\hat{\sigma}_j^{(i)})^2. \tag{2.5}$$

Substitute inequality (2.5) into inequality (2.4), then

$$\mathbb{E} \parallel \mathcal{R} \parallel_{\mathrm{F}}^2 \leq \frac{1}{n_3} \left(1 + \frac{k}{p-1}\right) \sum_{i=1}^{n_3} \sum_{j>k} (\hat{\sigma}_j^{(i)})^2.$$

Using Hölder's inequality,

$$\mathbb{E} \parallel \mathcal{R} \parallel_{\mathrm{F}} \leq \frac{1}{\sqrt{n_3}} \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{i=1}^{n_3} \sum_{j>k} (\hat{\sigma}_j^{(i)})^2\right)^{1/2}. \qquad \square$$

Theorem 2.3.3 is important because it shows that, in expectation, the error in the rt-SVD algorithm is within a factor $\sqrt{1 + \frac{k}{p-1}}$ of the optimal result in Theorem 2.2.1. Note that this is the same multiplicative factor that one obtains in the matrix case, see Theorem 1.2.17.

Theorem 2.3.3 also shows how the bound of $\parallel \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A} \parallel_{\mathrm{F}}$ relies on the decay of the singular values of $\hat{\mathcal{A}}^{(i)}$. If the singular values decay rapidly, we can bound $\sum_{j>k} (\hat{\sigma}_j^{(i)})^2$ by $(\hat{\sigma}_{k+1}^{(i)})^2$. Therefore, the above equation becomes

$$\mathbb{E} \|\mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A}\|_{\mathrm{F}} \leq \sqrt{1 + \frac{k}{p-1}} \max_{1 \leq i \leq n_3} \hat{\sigma}_{k+1}^{(i)}.$$

If the singular values of $\hat{\mathcal{S}}^{(i)}$ decay gradually, then we can instead use the following approximation $\sum_{j>k} (\hat{\sigma}_j^{(i)})^2 \approx (n-k)(\hat{\sigma}_{k+1}^{(i)})^2$ where $\hat{\sigma}_{k+1}^{(i)}$ is the $(k+1)^{th}$ singular value of the i^{th} frontal slice in the Fourier domain, and $n = \min\{n_1, n_2\}$. However, $\hat{\sigma}_{k+1}^{(i)}$ can be large, in this case, the accuracy of rt-SVD can be lost. We present a new algorithm (Algorithm 8) based on the t-product that applies the randomized subspace iteration to *each frontal slice in the Fourier domain* to improve the accuracy for this case.

Algorithm 8 rt-SVD with subspace iteration, spatial domain presentation

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, target truncation term k, oversampling parameter p, the number of iterations q

Output: $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$, and $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$

Generate a Gaussian random tensor $\mathcal{W} \in \mathbb{R}^{n_2 \times (k+p) \times n_3}$

Form a tensor $\mathcal{Y}_0 = \mathcal{A} * \mathcal{W}$ and compute the t-QR factorization $\mathcal{Y}_0 = \mathcal{Q}_0 * \mathcal{R}_0$

for $i = 1$ to q **do**

$\tilde{\mathcal{Y}}_i = \mathcal{A}^{\mathsf{T}} * \mathcal{Q}_{i-1}$ and compute the t-QR factorization $\tilde{\mathcal{Y}}_i = \tilde{\mathcal{Q}}_i * \tilde{\mathcal{R}}_i$

$\mathcal{Y}_i = \mathcal{A} * \tilde{\mathcal{Q}}_i$ and compute the t-QR factorization $\mathcal{Y}_i = \mathcal{Q}_i * \mathcal{R}_i$

end for

Form a tensor $\mathcal{Q} = \mathcal{Q}_q$

Form a tensor $\mathcal{B} = \mathcal{Q}^{\mathrm{T}} * \mathcal{A}$, the size of $\mathcal{B}$ is $(k+p) \times n_2 \times n_3$ which is smaller than tensor $\mathcal{A}$

Compute t-SVD of $\mathcal{B}$, truncate it, and obtain $\mathcal{U}$, $\mathcal{S}_k$, $\mathcal{V}_k$

Form the rt-SVD of $\mathcal{A}$, $\mathcal{A} \approx (\mathcal{Q} * \mathcal{U}) * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}} = \mathcal{U}_k * \mathcal{S}_k * \mathcal{V}_k^{\mathrm{T}}$.

The Algorithm 8 works efficiently even when the singular values of $\hat{\mathcal{S}}^{(i)}$ decay gradually for each frontal slice i. However, when the singular values of $\hat{\mathcal{S}}^{(i)}$ decay gradually only for some slices, the Algorithm 8 may spend some unnecessary computational effort. This can be avoided if a different number of iterations, q_i, is used for each frontal slice. Let us define the iteration vector as $\mathbf{q} = (q_1, q_2, \ldots, q_{n_3})^{\mathrm{T}}$. We present an algorithm (Algorithm 9) that employs different iteration count in each frontal slice.

Algorithm 9 rt-SVD with subspace iterations, Fourier domain implementation

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, target truncation term k, oversampling parameter p, and the iterations vector $\mathbf{q}$

Output: $\mathcal{U}_k \in \mathbb{R}^{n_1 \times k \times n_3}$, $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$, and $\mathcal{V}_k \in \mathbb{R}^{n_2 \times k \times n_3}$

Generate a Gaussian random tensor $\mathcal{W} \in \mathbb{R}^{n_2 \times (k+p) \times n_3}$

$\hat{\mathcal{A}} \leftarrow \texttt{fft}(\mathcal{A}, [\,], 3)$ and $\hat{\mathcal{W}} \leftarrow \texttt{fft}(\mathcal{W}, [\,], 3)$

for $i = 1$ to n_3 **do**

$\hat{\mathcal{Y}}^{(i)} = \hat{\mathcal{A}}^{(i)} \hat{\mathcal{W}}^{(i)}$

$[\hat{\mathcal{Q}}_{j-1}^{(i)}, \sim] = \texttt{qr}(\hat{\mathcal{Y}}^{(\texttt{i})}, 0)$

for $j = 1$ to q_i **do**

$\hat{\mathcal{Z}}_j^{(i)} = (\hat{\mathcal{A}}^{(i)})^{\mathrm{H}} \hat{\mathcal{Q}}_{j-1}^{(i)}$

$[\hat{\mathcal{G}}_j^{(i)}, \sim] = \texttt{qr}(\hat{\mathcal{Z}}^{(\texttt{i})}, 0)$

$\hat{\mathcal{Y}}_j^{(i)} = \hat{\mathcal{A}}^{(i)} \hat{\mathcal{G}}_j^{(i)}$

$[\hat{\mathcal{Q}}_j^{(i)}, \sim] = \texttt{qr}(\hat{\mathcal{Y}}^{(\texttt{i})}, 0)$

end for

Form $\hat{\mathcal{Q}}^{(i)}$ as $\hat{\mathcal{Q}}^{(i)} = \hat{\mathcal{Q}}_j^{(i)}$

$\hat{\mathcal{B}}^{(i)} = (\hat{\mathcal{Q}}^{(i)})^{\mathrm{T}} \hat{\mathcal{A}}^{(i)}$

$[\hat{\mathcal{U}}^{(i)}, \hat{\mathcal{S}}^{(i)}, \hat{\mathcal{V}}^{(i)}] = \texttt{svd}(\hat{\mathcal{B}}^{(\texttt{i})})$

$\hat{\mathcal{U}}_k^{(i)} = \hat{\mathcal{Q}}^{(i)} \hat{\mathcal{U}}^{(i)}$; $\hat{\mathcal{S}}_k^{(i)} = \hat{\mathcal{S}}^{(i)}(1:k, 1:k)$; $\hat{\mathcal{V}}_k^{(i)} = \hat{\mathcal{V}}^{(i)}(:, 1:k)$.

end for

$\mathcal{U}_k \leftarrow \texttt{ifft}(\hat{\mathcal{U}}_{\texttt{k}}, [\,], 3)$;

$\mathcal{S}_k \leftarrow \texttt{ifft}(\hat{\mathcal{S}}_{\texttt{k}}, [\,], 3)$;

$\mathcal{V}_k \leftarrow \texttt{ifft}(\hat{\mathcal{V}}_{\texttt{k}}, [\,], 3)$.

The expected error of the probabilistic part of Algorithm 9 is given in Theorem 2.3.4. We focus on the case where the oversampling parameter $p \geq 2$. The cases where $p = 0$ and $p = 1$ can be found in the Appendix of [92].

Theorem 2.3.4 *Given an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$ and an $n_2 \times (k+p) \times n_3$ tensor $\mathcal{W}$,*

if $\mathcal{Q}$ is obtained from Algorithm 9, then

$$\mathbb{E} \parallel \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A} \parallel_{\mathrm{F}} \leq \left(\frac{1}{n_3} \sum_{i=1}^{n_3} \left(1 + \frac{k}{p-1} (\tau_k^{(i)})^{4q_i} \right) \left(\sum_{j>k} (\hat{\sigma}_j^{(i)})^2 \right) \right)^{1/2}, \quad \texttt{for p} \geq 2,$$

where k is a target truncation term, p is the oversampling parameter, $\mathbf{q}$ is the iterations vector, $\hat{\sigma}_j^{(i)}$ is the i^{th} component of $\texttt{fft}(\mathcal{S}(\texttt{j},\texttt{j},:),[\,],\texttt{3})$, and $\hat{\tau}_j^{(i)} = \frac{\hat{\sigma}_{k+1}^{(i)}}{\hat{\sigma}_j^{(i)}}$.

The proof is similar to the proof of Theorem 2.3.3 with the exception of attention to the variation in q_i parameters allowed across frontal slices, and therefore omitted.

If the iteration count $q_i = q$ for all $i = 1, \ldots, n_3$, then the bound in Theorem 2.3.4 can be simplified as

$$\mathbb{E} \left\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A} \right\|_{\mathrm{F}} \leq \sqrt{1 + \frac{k}{p-1} (\tau_k^{\max})^{4q}} \left(\frac{1}{n_3} \sum_{i=1}^{n_3} \sum_{j>k} (\hat{\sigma}_j^{(i)})^2 \right)^{1/2}, \tag{2.6}$$

where $\tau_k^{\max} = \max_{1 \leq i \leq n_3} \tau_k^{(i)}$ is the largest singular value gap. In particular, $q = 0$ gives the same result as Theorem 2.3.3.

In [92], we explain how the truncation parameter should be related to the singular value gaps according to

$$q_i = \left\lceil \frac{1}{4} \log \frac{\epsilon(p-1)}{k} \Big/ \log \tau_k^{(i)} \right\rceil,$$

to ensures that the error in $\mathbb{E} \left\| \mathcal{A} - \mathcal{Q} * \mathcal{Q}^{\mathrm{T}} * \mathcal{A} \right\|_{\mathrm{F}}$ is at most $\sqrt{1+\epsilon}$ of the optimal result in Theorem 1.2.17, where $0 < \epsilon < 1$.

Theorems 2.3.3 and 2.3.4 provide average behavior in accuracy, and Theorem 2.3.5 provides bounds on the tail of the probabilistic error.

Theorem 2.3.5 *[92] With the assumptions of Theorem 2.3.4, let $0 < \delta < 1$ be the failure probability and define the constant*

$$C_\delta = \frac{e\sqrt{k+p}}{p+1} \left(\frac{2}{\delta} \right)^{\frac{1}{p+1}} \left(\sqrt{n_2 - k} + \sqrt{k+p} + \sqrt{2 \log \frac{2}{\delta}} \right).$$

Then, with probability at most $1-\delta$,

$$\left\|\mathcal{A}-\mathcal{Q}*\mathcal{Q}^{\mathrm{T}}*\mathcal{A}\right\|_{\mathrm{F}}^{2} \leq \frac{1}{n_3}\sum_{i=1}^{n_3}\left(1+C_{\delta}^{2}(\tau_k^{(i)})^{4q_i}\right)\left(\sum_{j>k}(\hat{\sigma}_j^{(i)})^2\right).$$

Proof: See Appendix of [92]. □

Theorem 2.3.5 shows that the result holds with high probability, and the chance that the upper bound cannot hold is very small.

2.4 Numerical Experiments

In this section, we provide some numerical results on the accuracy of the proposed low-rank representations, as well as on the computation time of the proposed methods. We also test our algorithms on an application to facial recognition. The datasets for the experiments are a subset of the Cropped Extended Yale Face Dataset B [1] (abbreviated as Cropped Yale B dataset) and the dataset of faces maintained at AT&T Laboratories Cambridge [13] (abbreviated as AT&T dataset). The Cropped Yale B dataset has 1140 images that contains the first 30 possible illuminations of 38 different people. Each image has 192×168 pixels in a grayscale range. We form the Cropped Yale B dataset as a $192 \times 1140 \times 168$ tensor, and this tensor is denoted by $\mathcal{B}$. The AT&T dataset has 400 images that contains 10 different poses of 40 people. Each image has 112×92 pixels in a grayscale. We form the AT&T dataset as a $112 \times 400 \times 92$ tensor, denoted as $\mathcal{E}$. The experiments were run on a laptop with 2.3 GHz Intel Core i7 and 8 GB memory.

2.4.1 Error Analysis

In Section 2.3, we derived theoretical results for the expected approximation errors of rt-SVD with and without subspace iteration. Here, we provide some numerical results to demonstrate their comparative performance. We compare the relative errors obtained by using rt-SVD, rt-SVD with subspace iteration, and the respective relative theoretically minimal errors on the dataset $\mathcal{B}$. The target truncation term

k is allowed to vary between 50 and 180. We define the relative errors obtained by using the rt-SVD with subspace iteration (see Algorithm 8) as e_k^q,

$$e_k^q = \frac{\|(\mathcal{I} - \mathcal{Q}\mathcal{Q}^{\mathrm{T}})\mathcal{A}\|_{\mathrm{F}}}{\|\mathcal{A}\|_F}, \tag{2.7}$$

where q represents the number of iterations and k denotes the target truncation term. Because the rt-SVD with $q = 0$ is a specific case of rt-SVD with subspace iteration, we will use e_k^0 to denote the relative errors obtained by using rt-SVD with no subspace iteration.

Theorem 1.2.17, gives us the best possible relative error e_k, as a function of the target truncation term k.

$$e_k = \frac{\|\mathcal{A} - \mathcal{A}_k\|_{\mathrm{F}}}{\|\mathcal{A}\|_F} = \frac{\|\hat{\mathcal{S}}(k+1:n, k+1:n, :)\|_{\mathrm{F}}}{\|\hat{\mathcal{S}}\|_F} \tag{2.8}$$

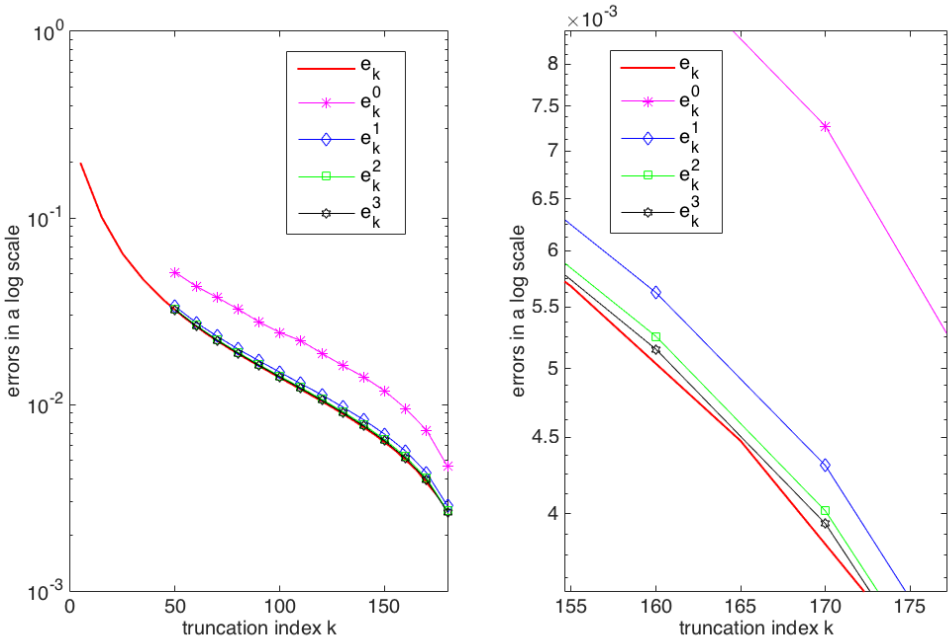


Figure 2.2: (left) The comparison of the exact error and the errors of rt-SVD with subspace iterations. (right) A zoomed in version of the left panel.

In Figure 2.2, it shows that all of the errors e_k^q's, with different number of iterations, have the similar convergence trajectories and are quite close to the best

possible theoretical error e_k. In other words, rt-SVD and rt-SVD with subspace iteration are both comparable in accuracy with the truncated t-SVD. Moreover, Figure 2.2 shows that e_k^q approaches e_k when the number of iterations q increases, that demonstrates the theoretical error bound behavior, which suggests that the rt-SVD with subspace iterations can force the singular values of $\hat{\mathcal{B}}^{(i)}$ decay more rapidly and give a more accurate approximation.

2.4.2 Facial Recognition

In this section, we apply the rt-SVD and rt-SVD with subspace iterations on the Cropped Yale B dataset and AT&T Dataset for the application of facial recognition. We process the training dataset and store a projector tensor $\mathcal{U}_k$ and a coefficient tensor $\mathcal{C}$ with smaller size. Also, there is a test dataset including new images. We determine which person each new image belongs to in the training dataset by comparing the tensor coefficients. The procedures for training dataset and test dataset are shown in Table 2.1.

Table 2.1: The procedure of facial recognition based on t-SVD method

Facial Recognition Procedure	
For the training Dataset:	For a test image from test dataset:
1. Form the training dataset into a three dimensional tensor and calculate the mean lateral slice through the second dimension;	1. Form the new image as a tensor with only one lateral slice and subtract the mean lateral slice from it;
2. Calculate standard mean-shifted tensor and denote it as tensor $\mathcal{A}$;	2. Compute the standard mean-shifted lateral slice and denote it as $\mathcal{T}$;
3. Compute the truncated t-SVD of $\mathcal{A}$ or the approximated truncated t-SVD of $\mathcal{A}$ with target truncation term k;	3. Compute the coefficient tensor $\mathcal{C}_t = \mathcal{U}_k^{\mathsf{T}} * \mathcal{T}$;
4. Compute the coefficient tensor $\mathcal{C} = \mathcal{U}_k^{\mathsf{T}} * \mathcal{A}$, and store it with projector tensor $\mathcal{U}_k$.	4. Find the smallest distance of $\mathcal{C}_t$ with each lateral slices of $\mathcal{C}$.

To measure the performance more rigorously, we use 10-fold cross-validation. In 10-fold cross-validation, the dataset is randomly partitioned into 10 equal-size subsets. In the k^{th} trial, the k^{th} subset (also referred to as a fold) is used as the

test dataset, whereas the other 9 subsets are simultaneously used as training dataset. Therefore, the algorithm will be tested 10 times with 10 different combinations of the same dataset. The randomized algorithms are run 20 times for each fold to compute the mean, maximum, and minimum of recognition rates. The recognition rate here is defined as

$$r = \frac{\text{the number of images recognized correctly}}{\text{the number of test images}}.$$

2.4.2.1 Cropped Yale Face B Dataset

There are 1140 images in Cropped Yale B Dataset, so the size of the training dataset is $192 \times 1026 \times 168$ and the size of the test dataset is $192 \times 114 \times 168$ in each fold. We display a few sample images of the Cropped Yale B dataset under different illuminations in Figure 2.3.

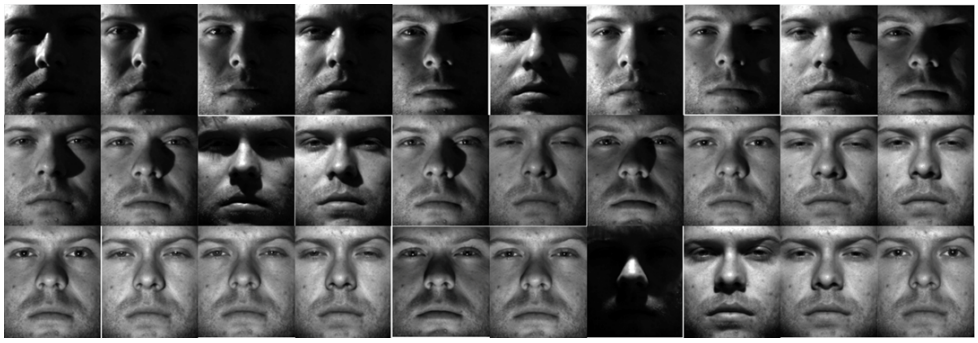


Figure 2.3: Sample images from Cropped Yale B Dataset

.

Table 2.2 shows the accuracy of the rt-SVD algorithm in terms of recognition rate, whereas Table 2.3 shows the accuracy of rt-SVD with simultaneous iterations. The computational times are reported in Figure 2.4. As expected, up to a point, the larger the truncation term is, the higher the recognition rate and respectively the higher the computation cost can take. In practice, the value of truncation term depends on the tradeoff between recognition rate and computational time. Here, we list the results with target truncation terms equal to 25 and 50 to fairly show the performance and provide choices for needs.

Table 2.2: Recognition Rates on Cropped Yale B dataset with $k = 25$

r	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
The t-SVD method										
	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9825
The rt-SVD method										
min*			0.9123	0.9912						0.9737
mean	0.9912	1.0000	0.9175	0.9943	1.0000	0.9912	0.9035	0.9737	0.7368	0.9772
max			0.9211	1.0000						0.9912
The rt-SVD method with simultaneous iterations $q = 1$										
min										0.9737
mean	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9833
max										0.9912
The rt-SVD method with simultaneous iterations $q = 2$										
min										0.9825
mean	0.9912	1.0000	0.9211	1.0000	1.0000	0.9912	0.9035	0.9737	0.7368	0.9882
max										0.9912

Table 2.3: Recognition Rates on Cropped Yale B dataset with $k = 50$

r	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
The t-SVD method										
	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
The rt-SVD method										
min										
mean	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
max										
The rt-SVD method with simultaneous iterations $q = 1$										
min										
mean	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
max										
The rt-SVD method with simultaneous iterations $q = 2$										
min										
mean	0.9912	1.0000	0.9298	1.0000	1.0000	0.9912	0.9035	0.9825	0.7368	0.9912
max										

In table 2.2, we have the following observations.

- The minimum and maximum recognition rates are very close to the mean recognition rate in the series of rt-SVD methods.
- Comparing the recognition rate between t-SVD and the series of rt-SVD methods, they are identical in 7 out of 10 folds. In the other 3 folds (fold 3, fold 4, and fold 10), the difference is very slight, less than .01.
- In fold 10, the maximum recognition rates of the series of rt-SVD method are even slightly higher than the recognition rate of t-SVD.

*When the minimum, maximum, and mean recognition rates are identical, we only list the mean recognition rate.

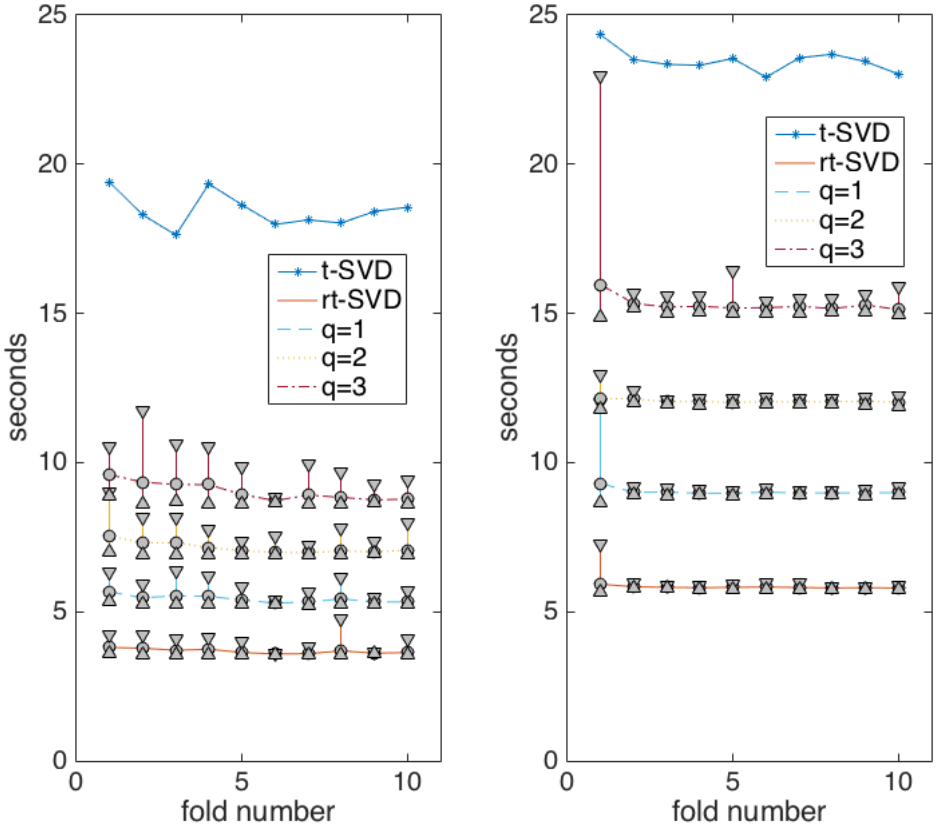


Figure 2.4: (left) Running time to process the training dataset of Cropped Yale B with $k = 25$. (right) Running time to process the training dataset of Cropped Yale B with $k = 50$. Randomized algorithms are presented via the error-bar plots.

Table 2.3 provides similar results to Table 2.2. In particular, the recognition rates are identical in each fold.

Regarding the running time to process the training dataset, Figure 2.4 shows that:

- In the context of running time, rt-SVD < rt-SVD with subspace $q = 1$ < rt-SVD with subspace $q = 2$ < t-SVD.
- The rt-SVD method takes about a third of the time to compute as the t-SVD does.

2.4.2.2 AT&T Dataset

For the AT&T Dataset, there are 400 images, so the size of training dataset in each fold is $112 \times 360 \times 92$ and the size of test dataset in each fold is $192 \times 40 \times 168$. In contrast to the Cropped Yale B dataset, the AT&T dataset has images with different poses, see Figure 2.5. As we discussed in subsection 2.4.2.1, we provide results with two different target truncation terms, 15 and 25. Table 2.4 and 2.5 show the performance of rt-SVD and rt-SVD with subspace iteration. Figure 2.6 shows the comparison of running times. The numerical results are consistent with the numerical results on the Cropped Yale B dataset, therefore demonstrating our algorithms have good performance both on illumination varying datasets and pose varying datasets.

Figure 2.5: Sample images from AT&T dataset.

2.4.3 Computation Time in Parallel on a Cluster

In the facial recognition application, the most computation time is spent on computing the exact or approximated truncated t-SVD, so in this subsection, we give the computation times of computing truncated t-SVD, rt-SVD, and rt-SVD with subspace iterations in parallel on a cluster. The dataset we use as an example is the Cropped Yale B dataset $\mathcal{B}$, and the target truncation term (i.e., k) is 50. The experiments are run by Matlab 2015a in (Tufts cluster with Intel(R) Xeon(R) CPU

[†]When the minimum, maximum, and mean recognition rates are identical, we only list the mean recognition rate.

Table 2.4: Recognition Rates on AT&T dataset with $k = 15$

r	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
The t-SVD method										
	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
The rt-SVD method										
min†				0.9750		0.9500				0.9000
mean	0.9750	1.0000	1.0000	0.9775	0.9750	0.9537	0.9250	1.0000	0.9750	0.9013
max				1.0000		0.9750				0.9250
The rt-SVD method with simultaneous iterations $q = 1$										
min										
mean	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
max										
The rt-SVD method with simultaneous iterations $q = 2$										
min										
mean	0.9750	1.0000	1.0000	0.9750	0.9750	0.9500	0.9250	1.0000	0.9750	0.9000
max										

Table 2.5: Recognition Rates on AT&T dataset with $k = 25$

r	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
The t-SVD method										
	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000
The rt-SVD method										
min										
mean	0.9750	1.0000	1.0000	0.9750	0.9587	0.9500	0.9250	1.0000	0.9750	0.9000
max										
The rt-SVD method with simultaneous iterations $q = 1$										
min										
mean	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000
max										
The rt-SVD method with simultaneous iterations $q = 2$										
min										
mean	0.9750	1.0000	1.0000	0.9750	0.9500	0.9500	0.9250	1.0000	0.9750	0.9000
max										

X5675 running at 3.07 GHz (8 cores)). We give results for serial and parallel implementations. In the parallel implementations, we used 2-8 processors. The t-SVD is computed by Algorithm 3, the rt-SVD is computed by Algorithm 7, and the rt-SVD with subspace iterations is computed by Algorithm 9. The results are shown in Figure 2.7. As can be seen, when the number of processors increases, the running time using parallel computing decreases on average.

2.5 Summary

In this Chapter, we discussed the advantages and limitations of the matrix based randomized algorithms and deterministic tensor-based algorithms. The algorithms

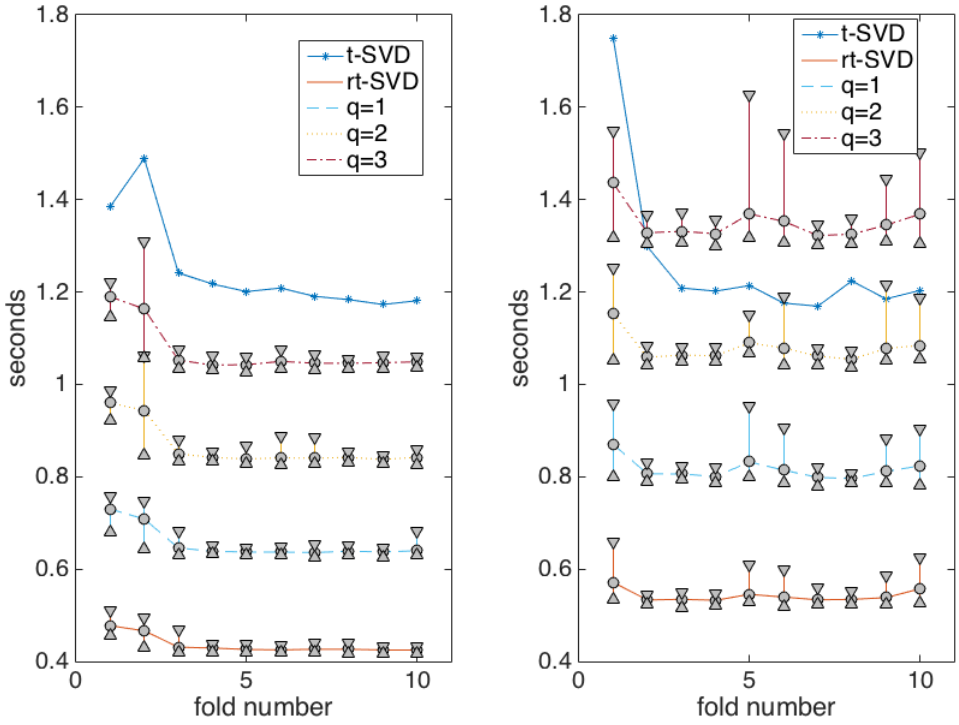


Figure 2.6: (left) Running time to process the $AT\&T$ training dataset with $k = 15$. (right) Running time to process the $AT\&T$ training dataset with $k = 25$. Randomized algorithms are presented via the error-bar plots.

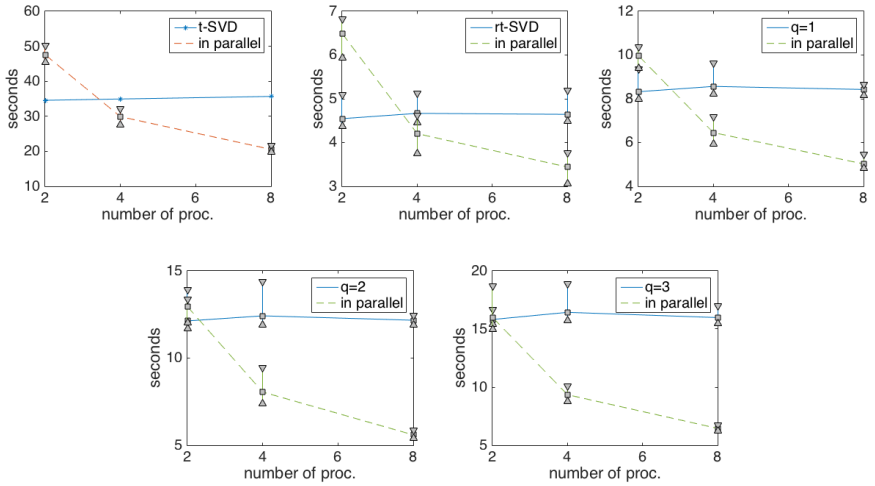


Figure 2.7: The computation time of t-SVD, rt-SVD, and rt-SVD in subspace iterations with and without parallel computing. Randomized algorithms are presented via the error-bar plots.

we design combines both of the advantages. We extend the randomized SVD method to third order tensors and provides both of the basic version algorithm (rt-SVD) and the more general version algorithm (rt-SVD with subspace iterations). We theoretically showed the expected errors of both rt-SVD and rt-SVD with subspace iterations that approximate the deterministic t-SVD can be bounded in Frobenius norm, and provide the numerical support in the application of facial recognition with two commonly used pubic datasets, Cropped Yale B dataset and AT&T Dataset, in respective of recognition rate and computation speed. Moreover, all the rt-SVD series algorithms can be done in parallel on a cluster. This makes our algorithm efficient in practice. In particular, if the application is running on a distributed memory machine, the approximation factorization can be separately stored on the different processors and this is convenient for later use without additional computation cost.

Chapter 3

Tensor-based Model Reduction

In this chapter, we will consider a dynamical system of the form:

$$\frac{\partial u(\mathbf{r},t)}{\partial t} - gu(\mathbf{r},t) - f(u(\mathbf{r},t)) = q(\mathbf{r},t), \tag{3.1}$$

where $\mathbf{r} = (x, y)$, g and f are the linear and nonlinear operators on $u(\mathbf{r},t)$, and $q(\mathbf{r},t)$ is the non-homogeneous term.

This system can be solved numerically by discretizing $u(\mathbf{r},t)$ both in space and time. However, the scale of the discrete problem is often challenging for processing, which motivates the need for reduced space representation of the discrete problem. In this study, we shall resort to projection based methods, and in particular, proper orthogonal decomposition (POD). The basic idea of POD is to project the original large system to a smaller system by using the basis obtained by singular value decomposition with a small number of snapshots of the solution. The POD method was developed by several researchers: Kosambi, Loéve, Karhunen, Pougachev and Obukhov in the 1940s and 1950s [57, 62] (for more details see [61]). It has been applied in a variety of fields, including fluid dynamics (see, eg. [45, 79]), control theory (see, eg. [4, 58]), inverse problems [8], random variable [70], image processing [72], data compression [7], and oceanography [71].

The performance of the POD method depends on the availability of snapshots and the effectiveness of the basis. In this chapter, we present a method that uses the t-SVD to generate a more effective basis with a limited number of snapshots. We will start with introducing some preliminary knowledge about the discretization and proper orthogonal decomposition in Section 3.1. In Section 3.2, we will introduce the tensor POD method and include a comparison of the basis from POD and t-POD in Subsection 3.2.1, the method to construct projector in Subsection 3.2.2, and the numerical experiments in Subsection 3.2.3. A summary follows in Section 3.3.

3.1 Introduction

Solving the dynamical system (3.1) means finding the unknown state$u(\mathbf{r}, t)$ that satisfies the equation. The analytical (exact) solution can be difficult to obtain or even not exists in practice, so leveraging numerical methods to compute an approximated solution is necessary.

3.1.1 Discretization Method

Instead of providing the variation of dependent variables continuously throughout the domain as analytical solutions, the numerical solutions can only provde answers at discrete points on a grid that is defined as follows.

Definition 3.1.1 (Grid) *A grid on a domain $\Omega \in \mathbb{R}^d$ is a finite or countable collection of points, $\vec{x}^j \in \Omega$, such that the distance between points,*

$$d(\vec{x}^i, \vec{x}^j) > 0,$$

for any i and j.

Definition 3.1.2 (Regular Grid) *A regular grid on a domain $\Omega \in \mathbb{R}^d$ is a grid such that writing $\vec{x}^j = \left(x_1^j, x_2^j, \ldots, x_d^j\right)$ for each j gives $\vec{x}^j - \vec{x}^i = m_{ij} h_k$ for each k and all i, j, where $m_{ij} \in \mathbb{Z}$, $h_k \in \mathbb{R}$.*

Partial derivatives of the discrete dynamical system at each grid point can be approximated based on Taylor's Theorem.

Theorem 3.1.3 (Taylor's Theorem with Remainder) *Let $n \in \mathbb{Z}^+$. If $f : [a, b] \to \mathbb{R}$ is $n+1$ times continuously differentiable on $[a, b]$ and x,$x_0 \in [a, b]$, then*

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^n(x_0)}{n!}(x - x_0)^n + \int_{x_0}^{x} \frac{f^{n+1}(t)}{n!}(x - t)^n dt.$$

Using Taylor's theorem, we can write

$$f(x_0 + h) = f(x_0) + \frac{f'(x_0)}{1!}h + \frac{f''(x_0)}{2!}h^2 + \ldots$$

and

$$f(x_0 - h) = f(x_0) - \frac{f'(x_0)}{1!}h + \frac{f''(x_0)}{2!}h^2 - \dots,$$

which directly provide three ways to approximate $f'(x_0)$,

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - O(\frac{f''(x_0)}{2!}h^2),$$

$$f'(x_0) = \frac{f(x_0) - f(x_0 - h)}{h} + O(\frac{f''(x_0)}{2!}h^2),$$

and

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} - O(\frac{f'''(x_0)}{3!}h^3).$$

By discretizing the Partial Differential Equation (PDE) system (3.1) on a grid in the spatial domain (i.e. semi-discretization), the system becomes an Ordinary Differential Equation (ODE) system. Let us use a regular grid, introduced in Definition 3.1.2, as an example for the discussion.

Define the space grid,

$$\mathbf{r}_{i,k} = (x_i, y_k) = (ih_x, kh_y),$$

where $i \in [0, n_x]$, $k \in [0, n_y]$. See Figure 3.1

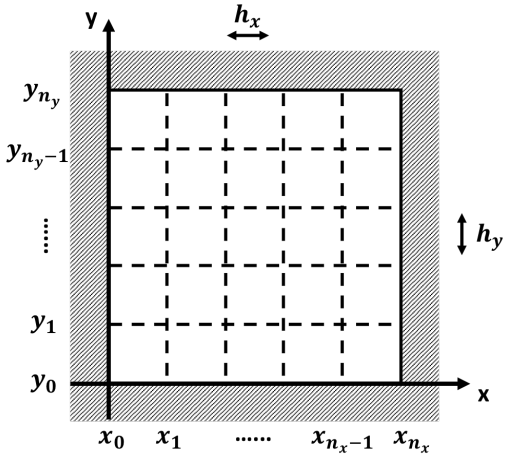


Figure 3.1: The grid for spatial domain.

The respective PDE system (3.1) discretized on the grid can be written as

$$\frac{\partial \bar{\mathbf{u}}(t)}{\partial t} = \mathbf{A}\bar{\mathbf{u}}(t) + f(\bar{\mathbf{u}}(t)), \tag{3.2}$$

where the size of $\mathbf{A}$ is $n_x n_y \times n_x n_y$ and $\bar{u}(t)$ is a column vector with size $n_x n_y$ constructed by raster scanning the semi-discretized $u(\mathbf{r}, t)$.

To solve this ODE, we also discretize it in the time domain and then approximate $\frac{\partial \bar{\mathbf{u}}(t)}{\partial t}$. Define the time grid

$$t_j = j h_t,$$

where $j \in [0, t_s]$. In approximating $\frac{\partial \bar{\mathbf{u}}(t)}{\partial t}$, explicit Euler method, implicit Euler method, and Crank-Nicolson method are commonly choices.

Explicit Euler method uses forward approximation, implicit Euler method uses backward approximation, and Crank-Nicolson method uses the average of forward approximation and backward approximation. They have their own advantages and disadvantages, and we do not discuss them here. As examples, if we use implicit Euler method to approximate $\frac{\partial \bar{\mathbf{u}}(t)}{\partial t}$, then the discretized ODE system (3.2) can be written as

$$\frac{\bar{\mathbf{u}}^{j+1} - \bar{\mathbf{u}}^{j}}{h_t} = \mathbf{A}\bar{\mathbf{u}}^{j+1} + f(\bar{\mathbf{u}}^{j+1}). \tag{3.3}$$

The cost of each step of the implicit Euler method is the cost of solving the system (3.3). The order of this system is $n_x n_y$, it can be expensive to solve it when $n_x n_y$ is large. Moreover, there are t_s time steps, so the system has to be solved t_s times.

In next section, we will introduce the POD method, which is used to reduce the computational cost. The main idea of the POD is projecting the original large system to a smaller system by forming a basis of the state space from a small number of snapshots of the solution. In the following section, an introduction to the keywords (snapshots, basis, and projection) is provided.

3.1.2 Proper Orthogonal Decomposition

The POD method has been used commonly to deal with reducing this computational cost by reducing the order of the system. It constructs a smaller size approximation system of order $r \ll n$ by projecting the original large system with a reduced order orthonormal projection matrix $\mathbf{W}_r$.

Consider a set of snapshots of the solution $\left\{\bar{\mathbf{u}}^1, \bar{\mathbf{u}}^2, \ldots, \bar{\mathbf{u}}^{\mu_s}\right\}$, that is a small size subset of solution. A basis for this set of snapshots is the solution to the following optimization problem [16].

$$\begin{aligned} &\arg \min_{\{\boldsymbol{\phi}_i\}_{i=1}^{r}} \sum_{j=1}^{\mu_s} \left\| \bar{\mathbf{u}}^j - \sum_{i=1}^{r} \langle \bar{\mathbf{u}}^j, \boldsymbol{\phi}_i \rangle \boldsymbol{\phi}_i \right\|_{\mathrm{F}}^2 \\ &\text{s.t. } \boldsymbol{\phi}_i^{\top} \boldsymbol{\phi}_j = \begin{cases} 0, & \text{if } i \neq j, \\ 1, & \text{if } i = j, \end{cases} \qquad i, j = 1, \ldots, r. \end{aligned} \tag{3.4}$$

The optimal solution of (3.4) is given by the first r left singular vectors of a matrix $\mathbf{X}$ constructed using $\left\{\bar{\mathbf{u}}^j\right\}_{j=1}^{\mu_s}$ as the columns. As we introduced in Chapter 1, the truncated singular value decomposition (SVD) of $n \times \mu_s$ matrix $\mathbf{X}$ is

$$\mathbf{X} \approx \mathbf{W_r} \mathbf{S_r} \mathbf{V_r}^{\top},$$

where $r \leq \mu_s$, the columns of $\mathbf{W_r} \in \mathbb{R}^{n \times r}$ are called left singular vectors, and they are orthogonal.

Therefore,

$$\bar{\mathbf{u}}^j \approx \mathbf{W_r} \mathbf{W_r}^{\top} \bar{\mathbf{u}}^j,$$

is the optimal approximation of $\bar{\mathbf{u}}^j$ (in the least square sense) in span of

$$\{\mathbf{W_r}(:,1), \mathbf{W_r}(:,2), \cdots, \mathbf{W_r}(:,r).\}$$

The error of (3.4) is

$$\sum_{j=1}^{\mu_s} \left\| \bar{\mathbf{u}}^j - \sum_{i=1}^{r} \langle \bar{\mathbf{u}}^j, \mathbf{W_r}(:,i) \rangle \mathbf{W_r}(:,i) \right\|_{\mathrm{F}}^2 = \sum_{i=r+1}^{\mu_s} \sigma_i^2, \tag{3.5}$$

where $\sigma_1, \sigma_2, \ldots, \sigma_r$ are the diagonal entries of $\mathbf{S_r}$.

Let $\tilde{\mathbf{u}}^j = \mathbf{W_r}^\top \bar{\mathbf{u}}^j$. Assume $\bar{\mathbf{u}} = \mathbf{W_r}\tilde{\mathbf{u}}$. Then, apply Galerkin projection: multiply from the left by $\mathbf{W_r}^\top$. The system (3.2) is replaced by the reduced order model,

$$\frac{\partial \tilde{\mathbf{u}}^j}{\partial t} = \mathbf{W_r}^\top \mathbf{A} \mathbf{W_r} \tilde{\mathbf{u}}^j + \mathbf{W_r}^\top f(\mathbf{W_r}\tilde{\mathbf{u}}^j), \tag{3.6}$$

where the size of $\mathbf{W_r}^\top \mathbf{A} \mathbf{W_r}$ is $r \times r$. We solve (3.6) at the cost of solving a smaller system with order r, and at each time step estimate $\bar{\mathbf{u}}$ by $\bar{\mathbf{u}} \approx \mathbf{W_r}\tilde{\mathbf{u}}$. The procedure is presented in Algorithm 10.

Algorithm 10 Proper Orthogonal Decomposition

1: Input: Snapshots of solution $\left\{\bar{\mathbf{u}}^1, \bar{\mathbf{u}}^2, \ldots, \bar{\mathbf{u}}^{\mu_s}\right\}$, μ_s is the number of snapshots, matrix $\mathbf{A}$, and t_s is the length of time grid

2: Output: Solution $\left\{\bar{\mathbf{u}}^{\mu_s+1}, \bar{\mathbf{u}}^{\mu_s+2}, \ldots, \bar{\mathbf{u}}^{t_s}\right\}$

3: Set $\mathbf{X} = [\bar{\mathbf{u}}^1, \bar{\mathbf{u}}^2, \ldots, \bar{\mathbf{u}}^{\mu_s}] \in \mathbb{R}^{n \times \mu_s}$

4: Compute the reduced SVD of $\mathbf{X}$, $[\mathbf{W}, \mathbf{S}, \mathbf{V}] = \mathrm{svd}(\mathbf{X}, 0)$;

5: Set projector as $\mathbf{W_r} = \mathbf{W}(:, 1:r)$

6: Project $\bar{\mathbf{u}}^j$ to lower dimension, $\bar{\mathbf{u}}^j = \mathbf{W_r}\tilde{\mathbf{u}}^j$

7: Solve $\tilde{\mathbf{u}}^j$ in the reduced size system (3.6) when $j = \mu_s + 1, \mu_s + 2, \ldots, t_s$ by using the implicit Euler method

8: Compute $\bar{\mathbf{u}}^j = \mathbf{W_r}\tilde{\mathbf{u}}^j$ when $j = \mu_s + 1, \mu_s + 2, \ldots, t_s$

The POD method has been successfully applied in many fields, but we can see its performance of POD depends on the availability and the quality of snapshots to some certain degree from our analysis. However, in practice, there may only limited number of snapshots available to use and the behavior of the POD based approximation may not be directly applicable to its "nearby" state space [55]. This

motivates us to generate a basis that could capture enough information using limited number of snapshots to approximate the solutions off the snapshots. In next section, we will present our method to build a basis with theoretical support and numerical results.

3.2 Tensor POD

In this section, we will present a tensor-based method that constructs a better basis of the snapshots. For convenience, we will begin with setting up the notation for this section.

Given a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we denote its j^{th} lateral slice as $\mathcal{X}^j \in \mathbb{R}^{n_1 \times n_3}$. Set $\bar{\mathbf{X}}_j \in \mathbb{R}^{n_1 \times n_3}$ as the matrix representation of $\mathcal{X}^j$, and $\widehat{\mathbf{X}} \in \mathbb{R}^{n_1 n_2 \times n_3}$ as the matrix that concatenates $\bar{\mathbf{X}}_1, \bar{\mathbf{X}}_2, \ldots, \bar{\mathbf{X}}_{n_2}$ along first dimension.

Dynamical system (3.1) describes the evolution of a field in the 2D spatial domain. In other words, at each single time point, the solution of the dynamical system is 2D. This motivates us to store the snapshots in the lateral slices instead of reshaping them and storing them in vectors, and we then obtain a basis from them.

Suppose there are μ_s snapshots. Store them in a tensor $\mathcal{X} \in \mathbb{R}^{n_x \times \mu_s \times n_y}$, where j^{th} snapshot is in lateral slice $\mathcal{U}^j \in \mathbb{R}^{n_x \times 1 \times n_y}$. The basis for these snapshots is the solution of the following optimization problem.

$$\begin{aligned} &\arg \min_{\{\mathcal{B}_l\}_{l=1}^{k_1}} \sum_{j=1}^{\mu_s} \left\| \mathcal{U}^j - \sum_{l=1}^{k_1} \mathcal{B}_l * \mathcal{B}_l^\top * \mathcal{U}^j \right\|_{\mathrm{F}}^2 \\ &\text{s.t. } \mathcal{B}_l^\top \mathcal{B}_m = \begin{cases} \mathbf{0} \in \mathbb{R}^{1 \times 1 \times n_y}, & \text{if } l \neq m, \\ \mathcal{I} \in \mathbb{R}^{1 \times 1 \times n_y}, & \text{if } l = m, \end{cases} \quad l, m = 1, \ldots, k_1. \end{aligned} \tag{3.7}$$

This is equivalent to finding an small optimal tensor $\mathcal{B}$ containing the basis

$\{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{k_1}\}$ as lateral slices.

$$\arg\min_{\mathcal{B}} \left\| \mathcal{U}^j - \mathcal{B} * \mathcal{B}^\top * \mathcal{U}^j \right\|_{\mathrm{F}}^2 \tag{3.8}$$

$$\text{s.t. } \mathcal{B}^\top * \mathcal{B} = \mathcal{I} \in \mathbb{R}^{k_1 \times k_1 \times n_y}$$

Based on Theorem 1.2.17, the optimal solution $\mathcal{B}$ is the first k_1 left singular slices of $\mathcal{U}$. The t-SVD of $\mathcal{X}$ is

$$\mathcal{X} = \mathcal{W} * \mathcal{S} * \mathcal{V}^\top,$$

where we shall denote the first k_1 left singular slices as $\mathcal{W}_{k_1}$.

3.2.1 Basis Comparison

Both SVD and tensor SVD can provide a basis for the snapshots. Here, we will compare the bases in accuracy, i.e compare the approximation errors $\|\mathcal{X} - \mathcal{W}_{k_1} * \mathcal{W}_{k_1}^\top * \mathcal{X}\|_{\mathrm{F}}$ and $\|\mathbf{X} - \mathbf{W}_r \mathbf{W}_r^\top \mathbf{X}\|_{\mathrm{F}}$. The following conclusions does not only work for the model reduction of dynamical systems, but also applies in general regarding to comparison of SVD and tensor SVD in accuracy, we use more general notations to discuss it.

Lemma 3.2.1 *[91] Given an $n_1 \times n_2 \times n_3$ tensor $\mathcal{X}$, the first k terms of the t-SVD is $\mathcal{X}_k = \sum_{i=1}^{k} \mathcal{W}(:,i,:) * (\mathcal{S}(i,i,:) * \mathcal{V}(:,i,:)^\top) = \sum_{i=1}^{k} \mathcal{W}(:,i,:) * \mathcal{C}(i,:,:)$. The matrix $\mathbf{X}_j$ represents the j^{th} lateral slice of $\mathcal{X}$ can be approximated as*

$$\mathbf{X}_j \approx \sum_{i=1}^{k} \mathbf{W}_i \texttt{circ}(\mathbf{c}_{\mathtt{i}}^{(\mathtt{j})}) \tag{3.9}$$

where $\mathbf{c}_i^{(j)} = \mathcal{C}(i,j,:)$ is the $(i,j)^{th}$ tube fiber.

Proof: Since $\mathbf{X}_j$ represents the j^{th} lateral slice of $\mathcal{X}$, $\mathbf{X}_j = \texttt{squeeze}(\mathcal{X}(:,\mathtt{j},:))$. Based

on the definition of t-product,

$$\begin{aligned}\mathbf{X}_j &\approx \sum_{i=1}^{k} \texttt{squeeze}\left(\mathcal{W}(:,\texttt{i},:) * \mathcal{C}(\texttt{i},\texttt{j},:)\right) = \sum_{\texttt{i}=1}^{\texttt{k}} \texttt{squeeze}\left(\mathcal{W}(:,\texttt{i},:) * \mathbf{c}_{\texttt{i}}^{(\texttt{j})}\right) \\ &= \sum_{i=1}^{k} \texttt{squeeze}\left(\texttt{fold}(\texttt{circ}(\mathcal{W}(:,\texttt{i},:)) \cdot \texttt{unfold}(\mathbf{c}_{\texttt{i}}^{(\texttt{j})}))\right).\end{aligned}$$

To prove $\mathbf{X}_j \approx \sum_{i=1}^{k} \mathbf{W}_i \texttt{circ}(\mathbf{c}_{\texttt{i}}^{(\texttt{j})})$, we need to show

$$\texttt{squeeze}\left(\texttt{fold}(\texttt{circ}(\mathcal{W}(:,\texttt{i},:)) \cdot \texttt{unfold}(\mathbf{c}_{\texttt{i}}^{(\texttt{j})}))\right) = \mathbf{W}_{\texttt{i}} \texttt{circ}(\mathbf{c}_{\texttt{i}}^{(\texttt{j})}).$$

The part, $\texttt{circ}(\mathcal{W}(:,\texttt{i},:)) \cdot \texttt{unfold}(\mathbf{c}_{\texttt{i}}^{(\texttt{j})}))$, is a block matrix with n_3. To better illustrate the proof, here we set $n_3 = 3$. The logic holds for any n_3 in general.

$$\begin{aligned}\texttt{circ}(\mathcal{W}(:,\texttt{i},:)) \cdot \texttt{unfold}(\mathbf{c}_{\texttt{i}}^{(\texttt{j})}) &= \begin{bmatrix} \mathcal{W}(:,i,1) & \mathcal{W}(:,i,3) & \mathcal{W}(:,i,2) \\ \mathcal{W}(:,i,2) & \mathcal{W}(:,i,1) & \mathcal{W}(:,i,3) \\ \mathcal{W}(:,i,3) & \mathcal{W}(:,i,2) & \mathcal{W}(:,i,1) \end{bmatrix} \begin{bmatrix} \mathbf{c}_i^{(1)} \\ \mathbf{c}_i^{(2)} \\ \mathbf{c}_i^{(3)} \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{W}(:,i,1)\mathbf{c}_i^{(1)} + \mathcal{W}(:,i,3)\mathbf{c}_i^{(2)} + \mathcal{W}(:,i,2)\mathbf{c}_i^{(3)} \\ \hline \mathcal{W}(:,i,2)\mathbf{c}_i^{(1)} + \mathcal{W}(:,i,1)\mathbf{c}_i^{(2)} + \mathcal{W}(:,i,3)\mathbf{c}_i^{(3)} \\ \hline \mathcal{W}(:,i,3)\mathbf{c}_i^{(1)} + \mathcal{W}(:,i,2)\mathbf{c}_i^{(2)} + \mathcal{W}(:,i,1)\mathbf{c}_i^{(3)} \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{W}(:,i,1)\mathbf{c}_i^{(1)} + \mathcal{W}(:,i,2)\mathbf{c}_i^{(3)} + \mathcal{W}(:,i,3)\mathbf{c}_i^{(2)} \\ \hline \mathcal{W}(:,i,1)\mathbf{c}_i^{(2)} + \mathcal{W}(:,i,2)\mathbf{c}_i^{(1)} + \mathcal{W}(:,i,3)\mathbf{c}_i^{(3)} \\ \hline \mathcal{W}(:,i,1)\mathbf{c}_i^{(3)} + \mathcal{W}(:,i,2)\mathbf{c}_i^{(2)} + \mathcal{W}(:,i,3)\mathbf{c}_i^{(1)} \end{bmatrix}\end{aligned}$$

Since $\mathbf{W}_i$ is the matrix that represents $\mathcal{W}(:,i,:)$, $\mathbf{W}_i(:,m) = \mathcal{W}(:,i,m)$ for any m^{th} column of $\mathbf{W}_i$. Then,

$$\texttt{circ}(\mathcal{W}(:,\texttt{i},:)) \cdot \texttt{unfold}(\mathbf{c}_{\texttt{i}}^{(\texttt{j})}) = \begin{bmatrix} \mathbf{W}_i(:,1)\mathbf{c}_i^{(1)} + \mathbf{W}_i(:,2)\mathbf{c}_i^{(3)} + \mathbf{W}_i(:,3)\mathbf{c}_i^{(2)} \\ \hline \mathbf{W}_i(:,1)\mathbf{c}_i^{(2)} + \mathbf{W}_i(:,2)\mathbf{c}_i^{(1)} + \mathbf{W}_i(:,3)\mathbf{c}_i^{(3)} \\ \hline \mathbf{W}_i(:,1)\mathbf{c}_i^{(3)} + \mathbf{W}_i(:,2)\mathbf{c}_i^{(2)} + \mathbf{W}_i(:,3)\mathbf{c}_i^{(1)} \end{bmatrix}$$

The squeeze and fold of this block matrix is equivalent to concatenating them

along second dimension, i.e.

$$
\begin{aligned}
&\texttt{squeeze}\left(\texttt{fold}(\texttt{circ}(\mathcal{W}(:,\texttt{i},:))\cdot\texttt{unfold}(\mathbf{c}_{\texttt{i}}^{(\texttt{j})}))\right)\\
&=\left[\begin{array}{c|c|c}
\mathbf{c}_i^{(1)}\mathbf{W}_i(:,1) & \mathbf{c}_i^{(3)}\mathbf{W}_i(:,2) & \mathbf{c}_i^{(2)}\mathbf{W}_i(:,3)\\
+\mathbf{c}_i^{(1)}\mathbf{W}_i(:,1) & +\mathbf{c}_i^{(3)}\mathbf{W}_i(:,2) & +\mathbf{c}_i^{(2)}\mathbf{W}_i(:,3)\\
+\mathbf{c}_i^{(1)}\mathbf{W}_i(:,1) & +\mathbf{c}_i^{(3)}\mathbf{W}_i(:,2) & +\mathbf{c}_i^{(2)}\mathbf{W}_i(:,3)
\end{array}\right]\\
&=\left[\begin{array}{c|c|c} \mathbf{W}_i(:,1) & \mathbf{W}_i(:,2) & \mathbf{W}_i(:,3) \end{array}\right]\begin{bmatrix}
\mathbf{c}_i^{(1)} & \mathbf{c}_i^{(3)} & \mathbf{c}_i^{(2)}\\
\mathbf{c}_i^{(2)} & \mathbf{c}_i^{(1)} & \mathbf{c}_i^{(3)}\\
\mathbf{c}_i^{(3)} & \mathbf{c}_i^{(2)} & \mathbf{c}_i^{(1)}
\end{bmatrix}\\
&=\mathbf{W}_i\texttt{circ}(\mathbf{c}_{\texttt{i}}^{(\texttt{j})}) \qquad \square
\end{aligned}
$$

Definition 3.2.2 *[85] An $n\times n$ matrix $\mathbf{A}$ is called circulant downshift matrix, if*

$$A=\left[\begin{array}{c|c|c|c|c} \mathbf{e}_2 & \mathbf{e}_3 & \cdots & \mathbf{e}_n & \mathbf{e}_1 \end{array}\right],$$

where $\mathbf{e}_i$ is i^{th} column of the $n\times n$ identity $\mathbf{I}_n$.

Remark 3.2.3 The matrix $\mathbf{W}_i\texttt{circ}(\mathbf{c}_{\texttt{i}}^{(\texttt{j})})$ can be written as

$$\mathbf{c}_i^{(1)}\mathbf{W}_i+\mathbf{c}_i^{(2)}\mathbf{W}_i\mathbf{Z}+\cdots+\mathbf{c}_i^{(n_3)}\mathbf{W}_i\mathbf{Z}^{n_3-1},$$

where $\mathbf{Z}$ is the circulant downshift matrix, that is also the generator for the cyclic group. $\diamondsuit$

Theorem 3.2.4 provides the theoretical comparison of truncated t-SVD and truncated SVD in accuracy. It is essentially important since it provides theoretical means explain why the basis obtained from truncated t-SVD is more effective to use.

Theorem 3.2.4 *[91] Suppose there is an $n_1\times n_2\times n_3$ tensor $\mathcal{X}$, denote matrix $\mathbf{X}_j$ as the matrix that represents the j^{th} lateral slice of $\mathcal{X}$ and $\widehat{\mathbf{X}}(:,j)=vec(\mathbf{X}_j)$. If $\mathcal{X}_k=\sum_{i=1}^{k}\mathcal{W}(:,i,:)*(\mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)^{\top})=\sum_{i=1}^{k}\mathcal{W}(:,i,:)*\mathcal{C}(i,:,:)$ and $\widehat{\mathbf{X}}_k=\sum_{i=1}^{k}\mathbf{Q}(:$*

$, i)\,(\mathbf{\Sigma}(i,i)\mathbf{V}(:,i)^\top) = \sum_{i=1}^{k} \mathbf{Q}(:,i)\mathbf{D}(i,:)$, *then*

$$\|\mathcal{X} - \mathcal{X}_k\|_{\mathrm{F}} \leq \|\widehat{\mathbf{X}} - \widehat{\mathbf{X}}_k\|_{\mathrm{F}},$$

when $k \leq \min(n_1, n_2, n_3)$ *and* $k \leq \min(n_1 n_3, n_2)$ *simultaneously.*

Proof: [91] Since $\widehat{\mathbf{X}}_k = \sum_{i=1}^{k} \mathbf{Q}(:,i)\,(\mathbf{\Sigma}(i,i)\mathbf{V}(:,i)^\top) = \sum_{i=1}^{k} \mathbf{Q}(:,i)\mathbf{D}(i,:)$, the j^{th} column of $\widehat{\mathbf{X}}_k$ is approximated as

$$\widehat{\mathbf{X}}_k(:,j) \approx \sum_{i=1}^{k} \mathbf{Q}(:,i) d_{ij}, \qquad j = 1, \ldots, n_2 \tag{3.10}$$

where d_{ij} is the $(i,j)^{th}$ entry of $\mathbf{D}$. As $\widehat{\mathbf{X}}(:,j) = \mathrm{vec}(\mathbf{X}_j)$, reshape $\mathbf{Q}(:,i)$ into $\mathbf{Q}_i \in \mathbb{R}^{n_1 \times n_3}$, we can write (3.10) as

$$\mathbf{X}_j \approx \sum_{i=1}^{k} \mathbf{Q}_i d_{ij} = \sum_{i=1}^{k} \mathbf{Q}_i \mathrm{circ}(d_{ij}\mathbf{e_1}), \tag{3.11}$$

From Lemma (3.2.1), equation (3.11) can be converted into tensor form,

$$\mathcal{X} \approx \sum_{i=1}^{k} \tilde{\mathcal{Q}}(:,i,:) * \tilde{\mathcal{D}}(i,:,:) =: \mathcal{Z}_k. \tag{3.12}$$

where $\tilde{D}$ is a tensor with the matrix D on the front face and zeros elsewhere. Therefore,

$$\|\mathcal{X} - \mathcal{X}_k\|_{\mathrm{F}} \leq \|\mathcal{X} - \mathcal{Z}_k\|_{\mathrm{F}} = \|\widehat{\mathbf{X}} - \sum_{i=1}^{k} \mathbf{Q}(:,i)\mathbf{\Sigma}(i,i)\mathbf{V}(:,i)^\top\|_{\mathrm{F}}.$$

□

3.2.2 Projector Construction

In the previous section, we gave the theoretical proof that shows the basis obtained from the t-SVD is more accurate than the basis obtained from SVD for the same k. Here, we will present how to use it to construct the projector. Recall the basis we obtain from t-SVD is $\mathcal{W}_{k_1} \in \mathbb{R}^{n_x \times k_1 \times n_y}$. So, the tensor $\mathcal{U}$ that stores the snapshots

of the solution is approximated as

$$\mathcal{X} \approx \mathcal{W}_{k_1} * \mathcal{W}_{k_1}^{\top} * \mathcal{X}. \tag{3.13}$$

Let $\mathcal{C} = \mathcal{W}_{k_1}^{\top} * \mathcal{X}$. Then,

$$\mathcal{U}^j \approx \mathcal{W}_{k_1} * \mathcal{C}(:, j, :). \tag{3.14}$$

From Lemma (3.2.1), the matrix that represents j^{th} lateral slice, in matrix form, is

$$\mathbf{U}_j \approx \sum_{i=1}^{k_1} \mathbf{W}_i \mathtt{circ}(\mathcal{C}(\mathtt{i}, \mathtt{j}, :)) = \sum_{\mathtt{i}=1}^{\mathtt{k_1}} \mathbf{W}_{\mathtt{i}} \mathtt{circ}(\mathbf{c}_{\mathtt{i}}^{(\mathtt{j})}) = \sum_{\mathtt{i}=1}^{\mathtt{k}} \mathbf{W}_{\mathtt{i}} \mathbf{F}^{\mathtt{H}} \mathtt{diag}(\mathtt{fft}(\mathbf{c}_{\mathtt{i}}^{(\mathtt{j})})) \mathbf{F}$$

where $\mathbf{c}_i^{(j)} \in \mathbb{R}^{1 \times 1 \times n_y}$ is $(i, j)^{th}$ tube fiber $\mathcal{C}$, and $\mathbf{F}$ is the normalized DFT matrix.

Therefore, from Remark (3.2.3), we can derive

$$\mathbf{U}_j = \sum_{i=1}^{k_1} \mathbf{W}_i \mathbf{Z}^{j-1} \bar{\mathbf{c}}_i^{(j)},$$

where $\bar{\mathbf{c}}_i^{(j)}$ is the squeeze of $\mathbf{c}_i^{(j)}$.

This means

$$\mathrm{vec}(\mathbf{U}_j) = \begin{bmatrix} \mathbf{W}_e \\ \mathbf{W}_e(\mathbf{I}_k \otimes \mathbf{Z}_{n_2}) \\ \vdots \\ \mathbf{W}_e(\mathbf{I} \otimes \mathbf{Z}_{n_2}^{(n_2-1)}) \end{bmatrix} \bar{\mathbf{c}}^{\mathbf{j}},$$

where $\bar{\mathbf{c}}^{\mathbf{j}} = [\bar{\mathbf{c}}_1^{(j)\top}, \bar{\mathbf{c}}_2^{(j)\top}, \ldots, \bar{\mathbf{c}}_{k_1}^{(j)\top}]^{\top}$ and $\mathbf{W}_e = [\mathbf{W}_1, \ldots, \mathbf{W}_k] \in \mathbb{R}^{n_x \times k_1 n_y}$.

As we know, $\mathrm{vec}(\mathbf{U}_j)$ is $\bar{\mathbf{u}}^j$. This implies that

$$\begin{bmatrix} \mathbf{W}_e \\ \mathbf{W}_e(\mathbf{I}_k \otimes \mathbf{Z}_{n_2}) \\ \vdots \\ \mathbf{W}_e(\mathbf{I} \otimes \mathbf{Z}_{n_2}^{(n_2-1)}) \end{bmatrix}$$

can be used as the projector. For convenience, we denote it as $\mathbf{B}$. The matrix $\mathbf{B}$ is

structured and

$$\mathbf{B}^\top \mathbf{B} = \mathbf{I}.$$

Recall the discretized ODE system (3.2) using the implicit Euler method is

$$\frac{\bar{\mathbf{u}}^{j+1} - \bar{\mathbf{u}}^{j}}{h_t} = \mathbf{A}\bar{\mathbf{u}}^{j+1} + f(\bar{\mathbf{u}}^{j+1}). \tag{3.15}$$

Since $\bar{\mathbf{u}}^j = \mathbf{B}\bar{\mathbf{c}}^{(j)}$, we replace $\bar{\mathbf{u}}^j$ by $\mathbf{B}\bar{\mathbf{c}}^{(j)}$ in equation (3.15). Then, it can be written as

$$\frac{\mathbf{B}\bar{\mathbf{c}}^{j+1} - \mathbf{B}\bar{\mathbf{c}}^{j}}{h_t} = \mathbf{A}\mathbf{B}\bar{\mathbf{c}}^{j+1} + f(\mathbf{B}\bar{\mathbf{c}}^{j+1}).$$

Using the Galerkin projection, multiplying by $\mathbf{B}^\top$ from left side, the above equation becomes

$$\frac{\mathbf{B}^\top\mathbf{B}\bar{\mathbf{c}}^{j+1} - \mathbf{B}^\top\mathbf{B}\bar{\mathbf{c}}^{j}}{h_t} = \mathbf{B}^\top\mathbf{A}\mathbf{B}\bar{\mathbf{c}}^{j+1} + \mathbf{B}^\top f(\mathbf{B}\bar{\mathbf{c}}^{j+1}),$$

that due to the orthogonality, is equivalent to

$$\frac{\bar{\mathbf{c}}^{j+1} - \bar{\mathbf{c}}^{j}}{h_t} = \mathbf{B}^\top\mathbf{A}\mathbf{B}\bar{\mathbf{c}}^{j+1} + \mathbf{B}^\top f(\mathbf{B}\bar{\mathbf{c}}^{j+1}),$$

where the size of $\mathbf{B}^T\mathbf{A}\mathbf{B}$ is $k_1 n_2 \times k_1 n_2$ and size of $\bar{\mathbf{c}}^j$ is $k n_2 \times 1$.

3.2.3 Numerical Experiment

In this section, we will present the numerical results for the 2D diffusion system,

$$\frac{\partial u(\mathbf{r},t)}{\partial t} - \nabla \cdot (\kappa(\mathbf{r},t)\nabla u(\mathbf{r},t)) = q(\mathbf{r},t), \tag{3.16}$$

where $u(\mathbf{r},t)$ is the state, $\mathbf{r} = (x,y)$ denotes the spatial coordinates, κ is the diffusion coefficient, and $q(\mathbf{r},t)$ is the source term.

The gradient of u is defined as $\nabla u(\mathbf{r},t) = \left(\frac{\partial u(\mathbf{r},t)}{\partial x}, \frac{\partial u(\mathbf{r},t)}{\partial y}\right)$. Using the grid we defined in Section 3.1 and employing a second order finite difference discretization scheme, the components of the discrete gradients are as follows.

$$\frac{\partial u(\mathbf{r},t)}{\partial x} \approx \frac{1}{h_x}\begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & \ddots & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}_{(n_x+1)\times n_x} u(\mathbf{r}_{:,k},t), \tag{3.17}$$

with a fixed k,

and

$$\frac{\partial u(\mathbf{r},t)}{\partial y} \approx \frac{1}{h_y}\begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & \ddots & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}_{(n_y+1)\times n_y} u(\mathbf{r}_{i,:},t), \tag{3.18}$$

with a fixed i.

For convenience, we denote the $(n_x+1)\times n_x$ and $(n_y+1)\times n_y$ lower bidiagonal matrices in equations (3.17) and (3.18) as $\mathbf{G_x}$ and $\mathbf{G_y}$. Let $n = n_x n_y$, and $\bar{\mathbf{u}}(t)$ be a column vector with size n constructed by raster scanning $u(\mathbf{r}_{i,k},t)$ column by column, then

$$\nabla\bar{\mathbf{u}}(t) \approx \mathbf{L}\bar{\mathbf{u}}(t),$$

where $\mathbf{L} = \left(\frac{1}{h_x}\mathbf{I_{n_y}}\otimes\mathbf{G_x}, \frac{1}{h_y}\mathbf{G_y}\otimes\mathbf{I_{n_x}}\right)^\top$ is a $n\times n$ matrix, $\mathbf{I_s}$ denotes the identity matrix with size s, and $\otimes$ denotes the Kronecker product.

The divergence of $u(\mathbf{r},t)$ is

$$\nabla\cdot(u(\mathbf{r},t)) = \frac{\partial u(\mathbf{r},t)}{\partial x} + \frac{\partial u(\mathbf{r},t)}{\partial y} \approx \mathbf{L}^\top\bar{\mathbf{u}}(t),$$

therefore we can write

$$\nabla\cdot(\kappa(\mathbf{r},t)\nabla u(\mathbf{r},t)) \approx \mathbf{L}^\top\mathbf{K}(t)\mathbf{L}\bar{u}(t),$$

where $\mathbf{K}(t)$ is a $n\times n$ diagonal matrix with entries of semi-discretized $\kappa(\mathbf{r},t)$.

The 2D diffusion system (3.16) becomes an ODE,

$$\frac{\partial \bar{\mathbf{u}}(t)}{\partial t} = \mathbf{L}^\top \mathbf{K}(t)\mathbf{L}\bar{\mathbf{u}}(t) + \bar{\mathbf{q}}(t), \tag{3.19}$$

where $\mathbf{L}^\top \mathbf{K}(t)\mathbf{L}$ corresponds to the matrix $\mathbf{A}$ in (3.2) and the size of $\mathbf{L}^\top \mathbf{K}(t)\mathbf{L}$ is $n \times n$.

When using the implicit Euler method, equation (3.19) can be written as

$$\frac{\mathbf{u}^{j+1} - \mathbf{u}^j}{h_t} = \mathbf{L}^\top \mathbf{K}(t)\mathbf{L}\mathbf{u}^{j+1} + \bar{\mathbf{q}}^{j+1}.$$

Then, given a $\bar{\mathbf{u}}^j$, at each time step $j+1$, $\bar{\mathbf{u}}^{j+1}$ can be solved by

$$(\mathbf{I} - h_t \mathbf{L}^\top \mathbf{K}(t)\mathbf{L})\bar{\mathbf{u}}^{j+1} = \bar{\mathbf{u}}^j + h_t \bar{\mathbf{q}}^{j+1}. \tag{3.20}$$

Thus, the cost of each step of implicit Euler method is the cost of solving the $n \times n$ system (3.20).

Set n_x, n_y and n_t as 150, and the diffusion coefficient $\kappa(\mathbf{r}, t)$ as $\kappa(x, t) = 5$ and $\kappa(y, t) = 2$.

Collect a set of snapshots of the solution $\left\{\bar{\mathbf{u}}^1, \bar{\mathbf{u}}^2, \ldots, \bar{\mathbf{u}}^{\mu_s}\right\}$. Figure 3.2 shows some samples of the snapshots. From the sample snapshots, we can observe the diffusion process. Since $\kappa(x, t) \geq \kappa(y, t)$, the diffusion process rate along x direction is faster than the diffusion diffusion process rate along y direction.

Figure 3.3 and Figure 3.4 show the plot of first 3 terms of POD basis and tensor POD basis. In Figure 3.4, the first basis can capture the peak values of snapshots data, the second basis can present the contour of snapshots data, and the third basis can identify the main diffusion direction to be the x direction. One might deduce from the comparison between Figures 3.3 and 3.4 that such information is not revealed by expanding using the matrix-based basis.

To compare the performance of POD and tensor POD when there are only limited number of snapshots, set $r = k_1 = \mu_s$. Figure 3.5 shows tensor POD method has a smaller error than POD method in approximating the solution of a 2D diffusion

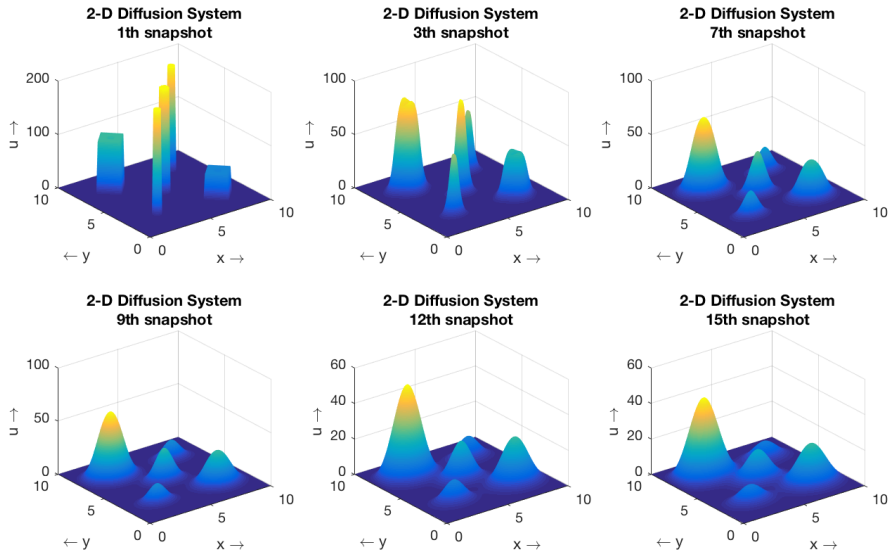


Figure 3.2: The sample snapshots of solution $\bar{\mathbf{u}}^j$, $j = 1, 3, 7, 9, 12, 15$

system. Here, the measurement is defined in Frobenius norm. Moreover, as can be seen, with increasing the number of snapshots, the errors of tensor POD method decreases more rapidly than the errors of POD method. For example, when the number of snapshots equals to 5, both of the relative errors of tensor POD and POD are about 10^{-1}, but when the number of snapshots equals to 40, the relative error of tensor POD method dropped fast to $\left(10^{-7} \text{ to } 10^{-8}\right)$ while the relative error of POD method is about 10^{-4}. Therefore, from Figure 3.5, it is evident that the tensor POD method is more effective to use when $r = k_1 = \mu_s$.

The main computation cost is due to the computation of the basis and solving the reduced model. Table 3.1 shows the comparison of main computation costs of POD and tensor POD. Per the step of basis computation, using SVD and using t-SVD have same computation cost, however, it is important to recall that t-SVD can be computed in parallel over the frontal slices on a cluster, whereas typical algorithms used for the computation of matrix based SVD cannot be performed in parallel. In the step of computing the reduced model, the size of reduced model in tensor POD method is larger than it in POD method although it can provide lower approximation error for the solutions off the snapshots.

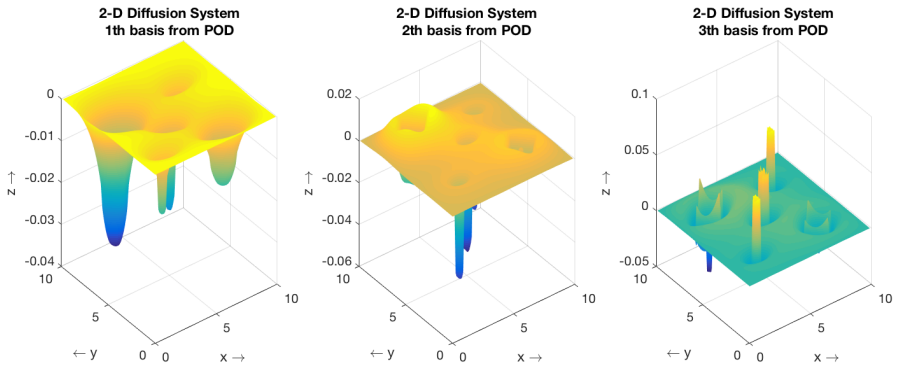


Figure 3.3: The first three basis vectors from SVD.

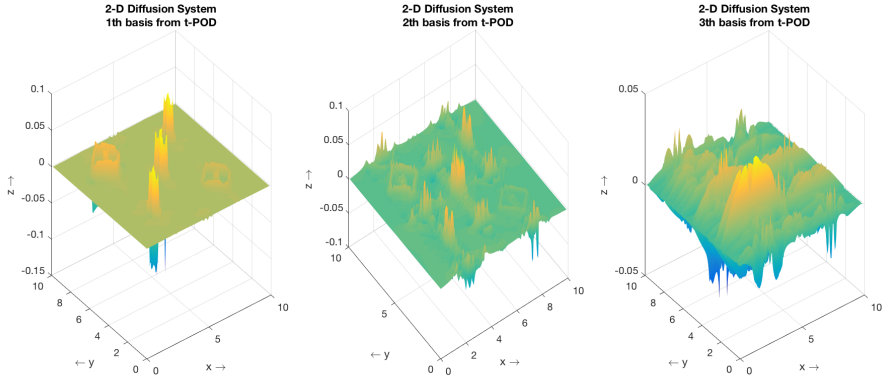


Figure 3.4: The first three basis slices from t-SVD.

We propose two improvements to reduce the computation cost of the tensor POD additionally.

1. Reduce model from two directions.

2. Leverage a method named t-SVD II, see [42], to compute basis.

Table 3.1: Main computation costs comparison of POD and tensor POD

Steps	**Matrix POD**	**Tensor POD**
compute basis	$O(n_x^2 n_y^2 \mu_s)$	$O(n_x^2 n_y^2 \mu_s)$
size of reduced model	μ_s	$n_y \mu_s$

Based on these two improvements, the computation cost of computing basis will maintain the same, but the size of reduced model will be ck_1 where $c \ll n_y$. Figure

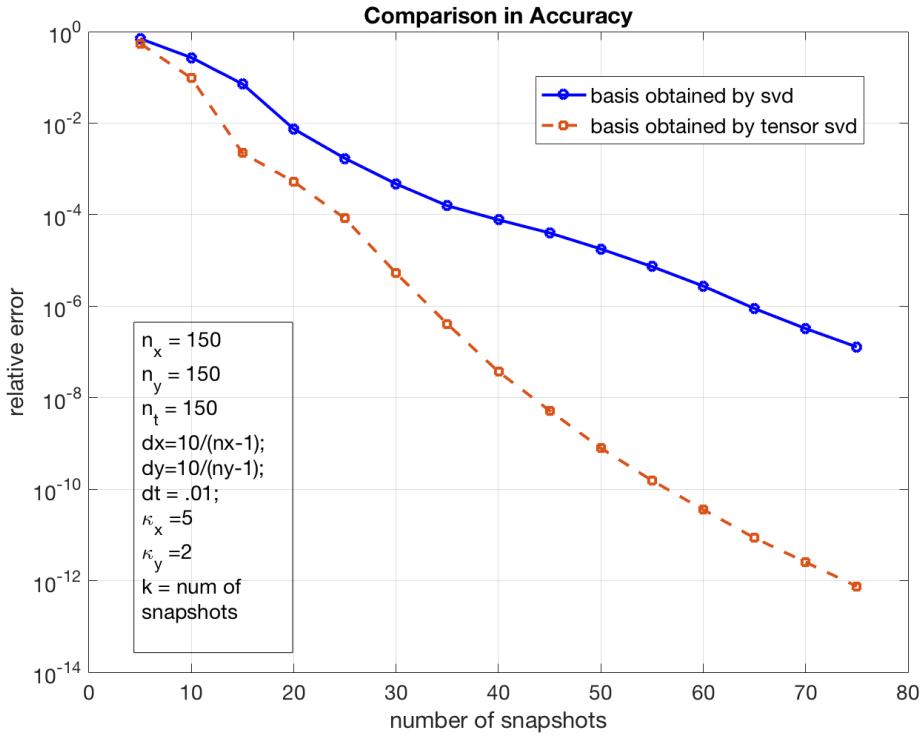


Figure 3.5: The comparison of POD and tensor POD in accuracy.

3.6 shows the experimental result of comparing accuracy with $c = 30$. The results indicate that the relative error of tensor POD method is substantially smaller than the relative error of POD method.

3.3 Summary

In this Chapter, we proposed a method named tensor POD, that is based on the t-SVD, to reduce the order of the dynamical system. We gave a comparison of approximation accuracy between SVD and t-SVD in theory as a theoretical supports to show tensor POD can generate a better basis for the snapshots in accuracy. We provided the numerical experiments on a 2D diffusion system, and illustrated the tensor POD basis can reveal more information than the POD basis from the snapshots of solutions, for example, the third basis of the tensor POD can identify the main diffusion direction clearly. Moreover, we showed the tensor POD formulation entails solutions superior level of accuracy compared to matrix based POD.

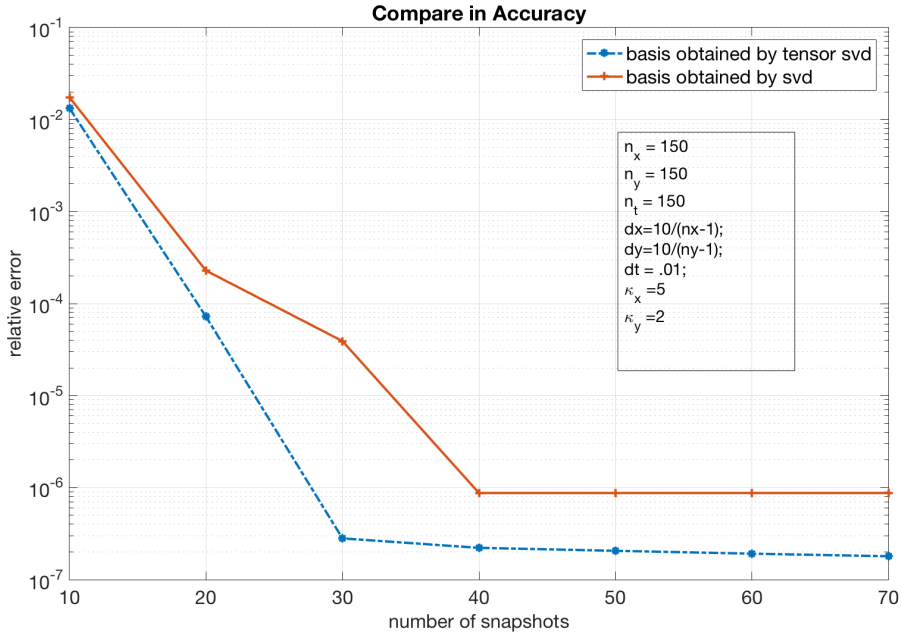


Figure 3.6: The comparison of POD and tensor POD in accuracy with $c = 30$.

Computation cost is predominated by the computing cost associated with derivation of the POD and solving the reduced model. Using SVD or t-SVD to compute basis has the same computational cost. However, the t-SVD can be computed more efficiently since it is straight forward to be computed in parallel over the frontal slices on a cluster, whereas typical algorithms used for computing SVD cannot be computed in parallel. The size of the reduced model obtained by the tensor POD may be larger than that given by the POD. However, when there are a limited number of snapshots, the extra computation cost from tensor POD may be accepted considering its advantage in approximating the solutions off the snapshots.

Chapter 4

Tensor Multi-frame Blind Deconvolution

In this chapter, we will introduce a tensor-based method (t-PQR) to select the most representative frames (images) for solving the multi-frame blind deconvolution problem. We will start with introducing the multi-frame blind deconvolution in Section 4.1. Then, we will discuss our method in Section 4.2 and present the numerical results in Section 4.3. The summary follows in Section 4.4.

4.1 Multi-frame Deconvolution

Image deblurring is one of the most important topics in image processing. It seeks to recover the original image using a mathematical model of the blurring process [39].

As we introduced in Chapter 1, an image can be represented as a matrix. Figure 4.1 shows the process of blurring an image. The matrix $\mathbf{X}$ represents the true image and the matrix $\mathbf{B}$ represents the blurred image. Since in our model blurring an image does not change the image size, the size of $\mathbf{X}$ and the size of $\mathbf{B}$ are same. Matrix $\mathbf{P}$ represents the point spread function whichs describes the blurring and the resulting image of the point source [39]. By convolving $\mathbf{P}$ and $\mathbf{X}$, the blurred image

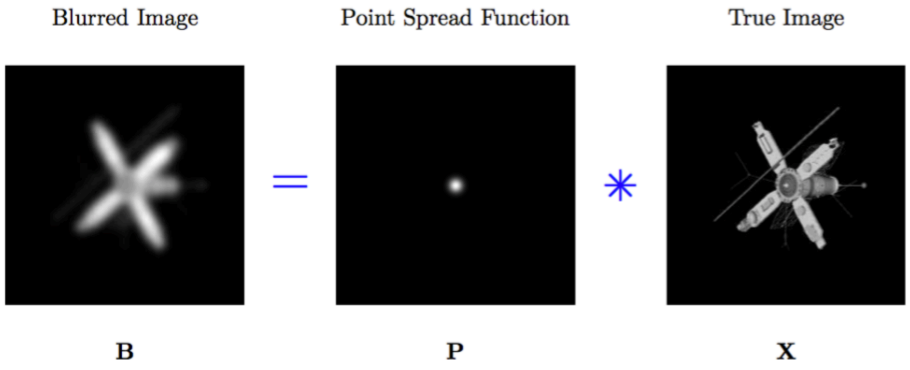


Figure 4.1: The process of blurring image [73].

$\mathbf{B}$ is obtained. The convolution of $\mathbf{P}$ and $\mathbf{X}$ is written as $\mathbf{P} \star \mathbf{X}$,

$$\begin{aligned} \mathbf{B}(x,y) &= \mathbf{P}(x,y) \star \mathbf{X}(x,y) \\ &= \int_{\mathbb{R}^2} \mathbf{P}(x',y') \star \mathbf{X}(x-x', y-y')dx'dy', \end{aligned}$$

where (x, y) denotes the pixel index of an image.

The idealized model of the blurring process can be expressed by matrix-vector multiplication,

$$\mathbf{b} = \mathbf{A}\mathbf{x}, \tag{4.1}$$

where $\mathbf{b}$ and $\mathbf{x}$ are vectorized $\mathbf{B}$ and $\mathbf{X}$, $\mathbf{b} = \text{vec}(\mathbf{B})$, $\mathbf{x} = \text{vec}(\mathbf{X})$. The matrix $\mathbf{A}$ is derived from the point spread function $\mathbf{P}$ with assuming a boundary condition of the image.

With different boundary conditions assumptions, $\mathbf{A}$ has different structures. We list a few as examples,

- Zero boundary condition: $\mathbf{A}$ is a block Toeplitz with Toeplitz block (BTTB) matrix;
- Periodic boundary condition: $\mathbf{A}$ is a block circulant with circulant block (BCCB) matrix;
- Reflective boundary condition: $\mathbf{A}$ is sum of four matrices with structures of block Toeplitz with Toeplitz blocks(BTTB), Block Toeplitz with Hankel blocks(BTHB), Block Hankel with Toeplitz blocks(BHTB), and Block Hankel with Hankel blocks(BHHB).

Here, we focus on the case of periodic boundary condition - matrix $\mathbf{A}$ has BCCB structure. As an illustration, let us use a 3×3 image to show the structure in equation

(4.1),

$$\begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \\ b_{13} \\ b_{23} \\ b_{33} \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} p_{22} & p_{12} & p_{32} & p_{21} & p_{11} & p_{31} & p_{23} & p_{13} & p_{33} \\ p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & p_{33} & p_{23} & p_{13} \\ p_{12} & p_{32} & p_{22} & p_{11} & p_{31} & p_{21} & p_{13} & p_{33} & p_{23} \\ \hline p_{23} & p_{13} & p_{33} & p_{22} & p_{12} & p_{32} & p_{21} & p_{11} & p_{31} \\ p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\ p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & p_{22} & p_{11} & p_{31} & p_{21} \\ \hline p_{21} & p_{11} & p_{31} & p_{23} & p_{13} & p_{33} & p_{22} & p_{12} & p_{32} \\ p_{31} & p_{21} & p_{11} & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} \\ p_{11} & p_{31} & p_{21} & p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & p_{22} \end{array}\right] \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{12} \\ x_{22} \\ x_{32} \\ x_{13} \\ x_{23} \\ x_{33}. \end{bmatrix}$$

From equation (4.1), it shows deblurring an image requires the solution of an inverse problem, $\mathbf{b} = \mathbf{A}\mathbf{x}$. When the matrix $\mathbf{A}$ is square and well-conditioned, the solution is $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. However, in practice, $\mathbf{A}$ is ill-conditioned or rectangular, so the image deblurring problem (4.1) is usually translated to a regularized optimization problem,

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + R(\mathbf{x}), \tag{4.2}$$

where $R(\mathbf{x})$ is a regularization term.

When multiple blurred images of the same object are available for reconstruction of the original image, this is referred as multi-frame image deblurring. The mathematical model is

$$\min_{\mathbf{x}} \sum_{j=1}^{m} \|\mathbf{A}_j\mathbf{x} - \mathbf{b}_j\|_2^2 = \min_{\mathbf{x}} \left\| \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_m \end{bmatrix} \right\|_2^2 + R(\mathbf{x}), \tag{4.3}$$

where m is the number of blurred images, $\mathbf{y}_j$ is j^{th} blurred image, $\mathbf{A}_j$ is the corresponding blur matrix, and $\mathbf{x}$ is the true image.

Sometimes, only the blurred image is known, and we need to reconstruct both of the true image and the point spread function. This kind of problem is referred as

"blind deconvolution" [29]. If we know a parametrized formula of the point spread function, the mathematical model is

$$\min_{\mathbf{x},\boldsymbol{\phi}} \|\mathbf{A}(\boldsymbol{\phi})\mathbf{x} - \mathbf{b}\|_2^2, \tag{4.4}$$

where $\boldsymbol{\phi}$ is a vector of unknown parameters characterizing the blurring process. For example, if the point spread function is a Gauss function, then

$$\boldsymbol{\phi} = [\mu_1, \mu_2, \sigma]. \tag{4.5}$$

With a boundary condition, the matrix $\mathbf{A}(\boldsymbol{\phi})$ can be derived by the point spread function

$$\mathbf{P}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x-\mu_1)^2 + (y-\mu_2)^2}{2\sigma^2}\right).$$

The multi-frame blind deconvolution is a blind deconvolution problem with multiple blurred images. The model combines (4.3) and (4.4), as

$$\min_{\mathbf{x},\boldsymbol{\phi}_j} \sum_{j=1}^{m} \|\mathbf{A}(\boldsymbol{\phi}_j)\mathbf{x} - \mathbf{b}_j\|_2^2 = \min_{\mathbf{x}} \left\| \begin{bmatrix} \mathbf{A}(\boldsymbol{\phi}_1) \\ \mathbf{A}(\boldsymbol{\phi}_2) \\ \vdots \\ \mathbf{A}(\boldsymbol{\phi}_m) \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_m \end{bmatrix} \right\|_2^2. \tag{4.6}$$

To solve the multi-frame blind deconvolution problem, one could use Gaussian-Newton method with Tikhonov regularization and GCV (MFBD method) [65]. However, this can be computationally expensive. To reduce the computation cost, the compact multi-frame blind deconvolution (CMF) method was proposed in [47] and compact single frame blind deconvolution (CSF) method based on compact multi-frame blind deconvolution was proposed in [73]. However, these methods require choosing a control frame (contains most independent information), which is typically selected empirically. In [73], results show that information loss may entail degraded that reduces accuracy in the reconstructed image when a poor control

frame is chosen. This motivates us to design a principled method to select an appropriate control frame. We derive an approach less heuristic in nature than the method employed currently [73].

4.2 Frame Selection

Suppose that m blurred images are available for reconstructing the true image, and the size of each image is $n_1 \times n_2$. For convenience, set $n = n_1 = n_2$. Form an $n \times m \times n$ tensor $\mathcal{Y}$ that stores these m blurred images, where the j^{th} lateral slice, $\mathcal{Y}^j$, stores the j^{th} blurred image, see Figure 4.2. Also, for j^{th} blurred image, form an $n \times n \times n$ point spread function tensor $\mathcal{P}_j$, where $\mathcal{P}_j^{(i)} = \mathbf{P}_i(\phi_j)$ and $i = 1, \ldots, n$. Denote the true image as an $n \times 1 \times n$ tensor $\mathcal{X}$.

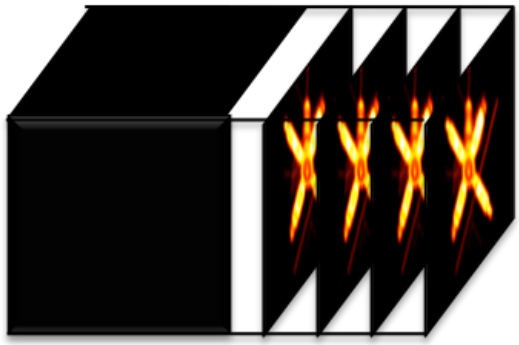


Figure 4.2: A tensor $\mathcal{Y}$ stores blurred images.

As illustrated in Chapter 1, $\mathbf{x}$ is obtained by vectorizing the true image, i.e.

$$\mathbf{x} = \texttt{unfold}(\mathcal{X}).$$

Similarly, for j^{th} blurred image,

$$\mathbf{y}_j = \texttt{unfold}(\mathcal{Y}^\texttt{j}).$$

Due to the structure of $\mathbf{A}(\phi_j)$ (BCCB), it can be written as

$$\mathbf{A}(\phi_j) = \begin{bmatrix} \mathbf{P}_1(\phi_j) & \mathbf{P}_n(\phi_j) & \cdots & \mathbf{P}_2(\phi_j) \\ \mathbf{P}_2(\phi_j) & \mathbf{P}_1(\phi_j) & \cdots & \mathbf{P}_3(\phi_j) \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{P}_n(\phi_j) & \mathbf{P}_{n-1}(\phi_j) & \cdots & \mathbf{P}_1(\phi_j) \end{bmatrix} = \texttt{circ}(\mathcal{P}_\texttt{j}),$$

where $\mathbf{P}_j$ is j^{th} corresponding point spread function matrix.

Therefore, equation (4.6) can be expressed in a tensor form,

$$\min_{x,\phi_j} \sum_{j=1}^{m} \|\mathcal{Y}_j - \mathcal{P}_j * \mathcal{X}\|_{\mathrm{F}}^2, \tag{4.7}$$

where $\mathcal{X}$ stores the true image and $\mathcal{Y}_j$ stores the j^{th} blurred image.

The t-PQR method is motivated by pivoted tensor QR factorization. In Chapter 1, we introduced its definition. Here, we present the algorithm to shows its procedure, see Algorithm 11.

Algorithm 11 pivoted t-QR method [37]

Input: an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$

Output: an $n_1 \times n_2 \times n_3$ orthogonal tensor $\mathcal{Q}$, an $n_2 \times n_2 \times n_3$ upper triangular tensor $\mathcal{R}$, and an $n_1 \times n_1 \times n_3$ permutation tensor $\mathcal{E}$

Generate a $n_1 \times n_1 \times n_3$ zero tensor $\mathcal{E}$;

$\hat{\mathcal{A}} = \texttt{fft}(\mathcal{A}, [\,], 3)$;

$[\hat{\mathcal{Q}}^{(1)}, \hat{\mathcal{R}}^{(1)}, \hat{\mathcal{E}}^{(1)}] = qr(\mathcal{A}^{(1)}, 0); \Rightarrow \hat{\mathcal{A}}^{(1)}\hat{\mathcal{E}}^{(1)} = \hat{\mathcal{Q}}^{(1)}\hat{\mathcal{R}}^{(1)}$;

for $i = 2$ to n_3 **do**

$[\hat{\mathcal{Q}}^{(i)}, \hat{\mathcal{R}}^{(i)}] = qr(\mathcal{A}^{(i)}\hat{\mathcal{E}}^{(1)})$;

end for

$\mathcal{Q} \leftarrow \texttt{ifft}(\hat{\mathcal{Q}}, [\,], 3)$

$\mathcal{R} \leftarrow \texttt{ifft}(\hat{\mathcal{R}}, [\,], 3)$

$\mathcal{E}^{(1)} = \hat{\mathcal{E}}^{(1)}$

The t-PQR method uses a part of the pivoted t-QR factorization and combine it

with the spectral analysis of the data tensor using t-SVD. It could determine which frame is the control frame, and the procedure is shown in Algorithm 12. Beyond the use of the method for selection of the control frame, it can also determine how many frames resolve a given threshold of the total energy and which frames are the most representative frames. Therefore, it can be used to choose a few representative images and use them as input for the MFBD method.

Algorithm 12 t-PQR method

Input: an $n \times m \times n$ tensor $\mathcal{Y}$

Output: a vector $\mathbf{s}$ storing the indices of representative frames and a control frame

$\hat{\mathcal{Y}} \leftarrow \texttt{fft}(\mathcal{Y}, [\,], 3)$;

Compute the singular values of $\hat{\mathcal{Y}}^{(1)}$, $\mathbf{s} = svd(\hat{\mathcal{Y}}^{(1)})$;

Determine the number of frames, l, by setting an energy threshold τ, such that $\frac{\|\mathbf{s}(1:l)\|_{\mathrm{F}}}{\|\mathbf{s}\|_{\mathrm{F}}} \geq \tau$;

Compute pivoted QR decomposition on $\hat{\mathcal{Y}}^{(1)}$, $[\hat{\mathcal{Q}}^{(1)}, \hat{\mathcal{R}}^{(1)}, \hat{\mathcal{E}}^{(1)}] = \texttt{qr}(\hat{\mathcal{Y}}^{(1)}, 0)$;

Store the indices of most representative frames $\mathbf{E} = \mathrm{diag}(\hat{\mathcal{E}}^{(1)})(1:l)$;

Set $\mathbf{E}(1)^{th}$ lateral slice of $\mathcal{Y}$ as the control frame

4.3 Numerical Experiments

In this section, we will present the numerical experiments on method t-PQR with 50 blurred images. These images are blurred by a sharpe image of size 256×256 with 50 Gaussian PSFs. To test the performance rigorously, we add four different levels of white noise into the blurred images. The white noise containing the pseudo-ransom values drawn from a normal distribution with mean zero and standard deviation one. The levels of white noise specifying the percentage of noise added to the blurred images. Level 000 here means no white noise added, level 001 means .01% of white noise added, level 010 means .1% white noise added, and level 100 means 1% white noise added. We label these test datasets as noise 000, noise 001, noise 010, and noise 100. Form them as tensors. The size of each tensor is $256 \times 50 \times 256$. Figure

4.3 plots the singular values of $\hat{\mathcal{Y}}^{(1)}$, $\mathbf{s}$, that suggests how many images to use for constructing the true image. We found that only 2 to 5 images are enough to use for covering the information from 50 blurred images. Table 5.1 shows the images t-PQR method suggests to use, and they are marked by red color.

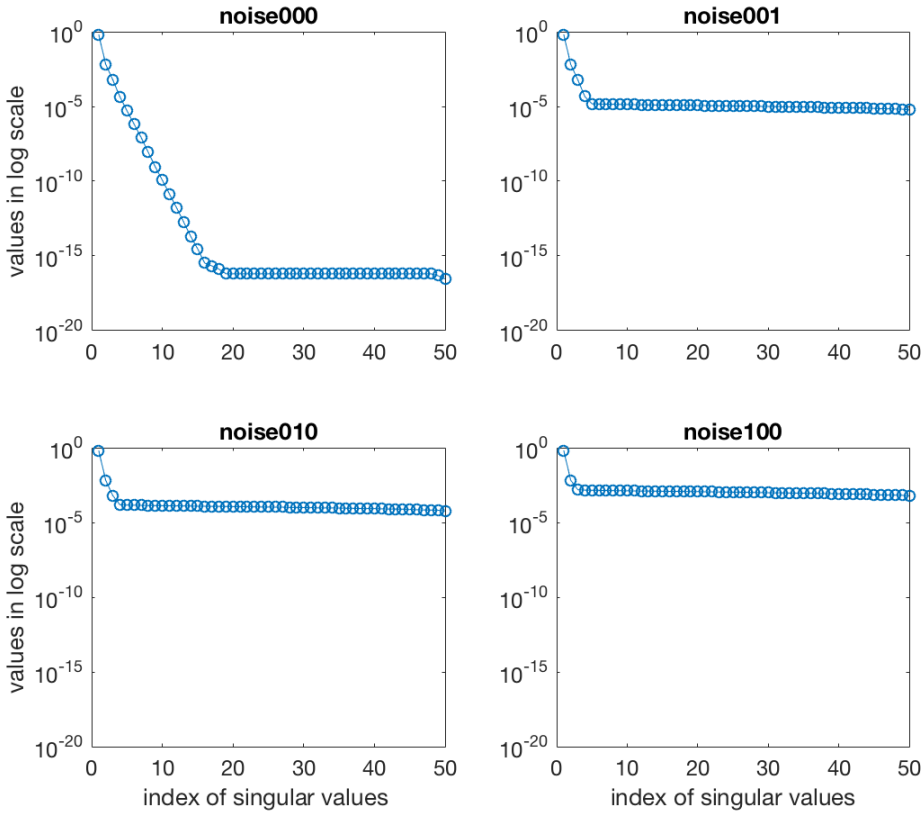


Figure 4.3: Plot of the singular values, $\mathbf{s}$, in a log scale.

Table 4.1: The most representative frames.

datasets	number of images to use
Noise 000	2 or 3
Noise 001	4 or 5
Noise 010	3 or 4
Noise 100	2 or 3

datasets	The first 5 elements of E
Noise 000	1, 50, 22, 12, 41
Noise 001	1, 50, 22, 11, 40
Noise 010	1, 50, 21, 24, 8
Noise 100	8, 50, 4, 40, 1

Define the relative reconstruction error as

$$e = \frac{\|\mathbf{X} - \mathbf{X}_{approx}\|_{\mathrm{F}}}{\|\mathbf{X}\|_{\mathrm{F}}},$$

where $\mathbf{X}$ is the true image and $\mathbf{X}_{approx}$ is the reconstructed image.

Table 4.2 presents the relative reconstruction errors of multi-frame blind deconvolution method (MFBD), compact single frame blind deconvolution method (CSF), and t-PQR. MFBD method does not require control frames, and it uses all of the blurred images as input, and the CSF requires one control frame. We make following observations:

- MFBD has the smallest relative reconstruction error.
- The relative reconstruction errors of CSF and t-PQR with 1 image are close.
- The relative reconstruction errors of t-PQR with 4 images and t-PQR with 5 images are more close to the relative reconstruction error of MFBD than CSF.
- Using t-PQR method with one frame can have an as small error as the CSF method has except the dataset with 1% white noise level.
- In the dataset with 1% white noise level, using the t-PQR method with 5 frames of 50 frames can achieve smaller construction error than the CSF method.

Table 4.2: Comparison in Relative Reconstruction Errors

Methods/datasets	Noise 000	Noise 001	Noise 010	Noise 100
MFBD	.2095	.2410	.2611	.2919
CSF	.2165	.2504	.2673	.3101
t-PQR with 1 image	.2165	.2452	.2673	.3266
t-PQR with 2 images	.2154	.2452	.2673	.3266
t-PQR with 3 images	.2122	.2452	.2674	.3106
t-PQR with 4 images	.2140	.2451	.2673	.3105
t-PQR with 5 images	.2153	.2451	.2665	.3016

As we know computing the solution using MFBD method is accurate but computationally expensive, while CSF method is more efficient, but requires to choose a control frame empirically.The following example shows the empirically chosen control frame has larger reconstruction error than t-PQR chosen frame. This example is not constructed with specific setting, it shows the possible scenarios we could encounter in real practice.

Given 10 satellite blurred images that are blurred by 10 Gaussian PSFs and added .01% white noise, the singular values of $\hat{\mathcal{Y}}^{(1)}$, $\mathbf{s}$ are plotted in Figure 4.4. Since the CSF method involves the information from all of the frames and the t-PQR method with 1 frame only involves the information from one frame, the CSF method was supposed to have smaller reconstruction error. However, the CSF method has larger reconstruction error, see Table 4.3.

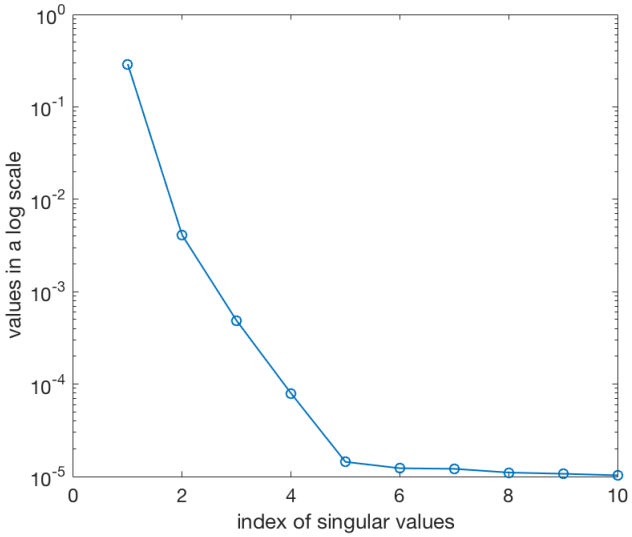


Figure 4.4: Plot the singular values of $\hat{\mathcal{Y}}^{(1)}$ in a log scale.

Table 4.3: Comparison of Relative Reconstruction Errors

Methods	Relative errors
Use all frames	.1706
CSF	.1864
PQR with 1 frames	.1806

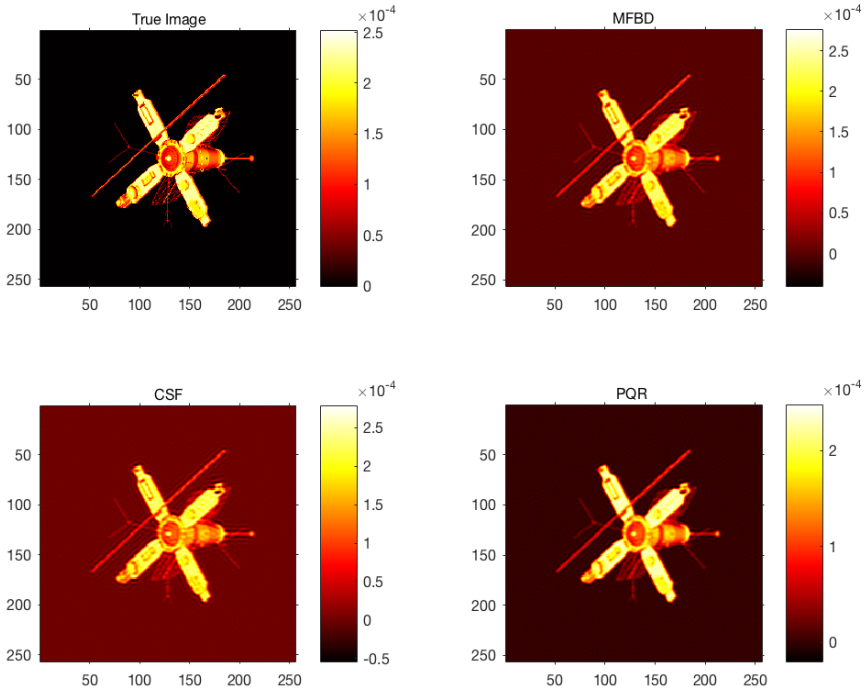


Figure 4.5: True image and the reconstructed images using MFBD, CSF, and t-PQR.

Figure 4.5 shows the true image and the reconstructed images using MFBD, CSF, and t-PQR. We could see that

- The image reconstructed by MFBD method contains more information about details than other methods. For example, see the lines and patterns on the bottom right solar panel.
- The boundaries of the satellite on the image constructed by CSF is not as sharp as it is on on the image constructed by t-PQR.

4.4 Summary

In this chapter, we investigated the mathematical model for the multi-frame blind deconvolution problem. To solve it, one could use the MFBD method, but it is computationally expensive. The CMF method and CSF method are computationally efficient, but they require to choose a control frame, which is typically selected

empirically. We proposed a method that is less heuristic to choose the control frame, that avoids information loss caused by inappropriate choice of the control frame in the CMF method and CSF method. Moreover, our method can be used to choose several most representative frames, and use them as new inputs for the MFBD method directly. Thanks to Professor James Nagy for supplying the codes of the MFBD method and CSF method, as well as the codes to build experiments.

Chapter 5

Video Enhancement

5.1 Introduction and Motivation

Enhancing resolution is an important topic in the field of image processing. High-resolution images are desired in many situations while they are not always available due to the hardware limitations, therefore a method to provide enhanced resolution image from available low resolution ones is necessary. In this chapter, we will present a method to enhance the resolution of video frames using tensor modeling and a tensor nuclear norm as a regularization term.

A video is a representation of moving still frames (images), so we will begin the discussion with the resolution enhancement of images. The resolution of an image is the finest level of details the image can hold, and it is denoted by the number of pixel columns by the number of pixel rows. The earliest formula for the resolution enhancement problem was motivated by the need of improved resolution images of the Landsat images [48]. The idea was extended to noisy and blurry images using least-squares minimization in [49, 52]. In [74], the authors put the high-resolution enhancement problem into the Bayesian framework using a Huber-Markov random field model. Iterative spatial domain methods are other popular direction for solving the problems of resolution enhancement [11, 24–26, 46, 54, 60, 66–69], see [12] for more details. Wavelets and tight frames can also be used to improve the quality of images [12, 14].

For video resolution enhancement, one can enhance the resolution of each frame as a single image. However, it obviously ignores the information its nearby frames can provide. Typical digital video records 30 frames per second when the video is playing. Each frame can be seen as a perturbation of its nearby frames. Figure 5.1 shows an example. The camera panned along a stack of books and one can see

the scenes of 10 continuous frames (from 100^{th} to 109^{th}) are almost same except for some small displacements.

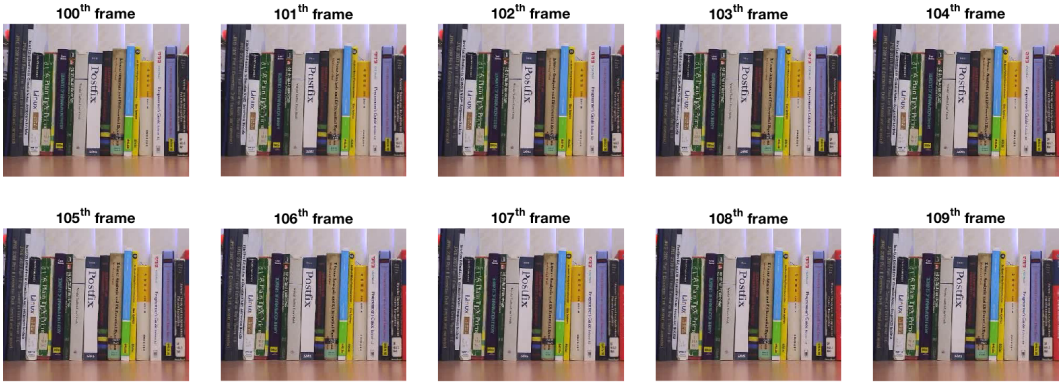


Figure 5.1: Some sample frames of a video for a stack of books.

The authors of [15] proposed a method to construct a high-resolution frame by considering the low resolution nearby frames as the images taken by other sensors. Therefore, the nearby frames could be used to provide independent information to improve the quality of the original frame. However, when the nearby frames are included to construct the high-resolution image, redundant information may also be included simultaneously.

To reduce the redundant information of the nearby frames and regularize the problem, we will use the tensor nuclear norm as a regularization tool, which has been proved the efficiency for the large data set with low rank [40, 93].

The structure of this chapter is as follows. In Section 5.2, we will review the definition and relevant properties of the tensor nuclear norm. Then, in Section 5.3, we will discuss the blurring operator construction and introduce a method to preprocess the video frames. In Section 5.4, we will build the model and provide the algorithm to solve the model. In Section 5.5, we will present the numerical results on two video clips. A summary follows in Section 5.6.

5.2 Tensor Nuclear Norm

Since we will be considering image frames together as a single tensor, and the frames contain very similar information, we expect that the existence of redundant information in the tensor. To reduce it, we will use the tensor nuclear norm as a tool, defined as follow.

Definition 5.2.1 *[40] Given an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$ and $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$, its tensor nuclear norm (TNN) is*

$$\|\mathcal{A}\|_* = \sum_{k=1}^{n_3} \left(\sum_{i=1}^{min(n_1,n_2)} \hat{\mathcal{S}}(i,i,k) \right) \tag{5.1}$$

where $\hat{\mathcal{S}}$ is the Fourier transform of $\mathcal{S}$ through third dimension.

We will also use the following results for solving our resolution enhancement model later in the Section 5.4.2.

Theorem 5.2.2 *[40] Given an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$,*

$$\sum_{k=1}^{n_3} \left(\sum_{i=1}^{min(n_1,n_2)} \hat{\mathcal{S}}(i,i,k) \right) = \|\mathtt{circ}(\mathcal{A})\|_* = \|(\mathbf{F} \otimes \mathbf{I})\mathtt{circ}(\mathcal{A})(\mathbf{F}^{\mathtt{H}} \otimes \mathbf{I})\|_*,$$

where F is the normalized discrete Fourier transform matrix.

Remark 5.2.3 [40] Given an $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$, $\|\mathcal{A}\|_*$ is a well defined norm, i.e.

- For any $\mathcal{A}$, $\|\mathcal{A}\|_* \geq 0$, and $\|\mathcal{A}\|_* = 0$ only when $\mathcal{A} = 0$;
- Let $\tau \in \mathbb{R}$, then $\|\tau\mathcal{A}\|_* = |\tau|\|\mathcal{A}\|_*$;
- Given two tensors $\mathcal{A}$ and $\mathcal{B}$ with same size, $\|\mathcal{A} + \mathcal{B}\|_* \leq \|\mathcal{A}\|_* + \|\mathcal{B}\|_*$. ◇

Theorem 5.2.4 *If $h_0(\mathcal{A}) = \|\mathcal{A}\|_* + \frac{1}{2}\|\mathcal{A} - \mathcal{B}\|_F^2$, then $h_0(\mathcal{A})$ is strictly convex.*

Proof: For any $\mathcal{A}_1 \neq \mathcal{A}_2$ and any $t \in (0,1)$,

$$\begin{aligned} h_0(t\mathcal{A}_1 + (1-t)\mathcal{A}_2) &= \|t\mathcal{A}_1 + (1-t)\mathcal{A}_2\|_* + \frac{1}{2}\|t\mathcal{A}_1 + (1-t)\mathcal{A}_2 - \mathcal{B}\|_{\mathrm{F}}^2 \\ &< \sum_{k=1}^{n_3}\left(\sum_{i=1}^{\min(n_1,n_2)} t\hat{\mathcal{S}}_1(i,i,k) + (1-t)\hat{\mathcal{S}}_2(i,i,k)\right) \\ &+ \frac{1}{2}\|t\mathcal{A}_1 - \mathcal{B}\|_{\mathrm{F}}^2 + \frac{1}{2}\|(1-t)\mathcal{A}_2 - \mathcal{B}\|_{\mathrm{F}}^2 \\ &= \|t\mathcal{A}_1\|_* + \|(1-t)\mathcal{A}_2\|_* + \frac{1}{2}\|t\mathcal{A}_1 - \mathcal{B}\|_{\mathrm{F}}^2 + \frac{1}{2}\|(1-t)\mathcal{A}_2 - \mathcal{B}\|_{\mathrm{F}}^2 \\ &= th_0(\mathcal{A}_1) + (1-t)h_0(\mathcal{A}_2). \end{aligned}$$

Therefore, $h_0(\mathcal{A})$ is strictly convex. □

Theorem 5.2.5 *[?] For each parameter $\tau > 0$, and $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$,*

$$\mathcal{A}_{opt} = \mathtt{argmin}_{\mathcal{A}} \left\{ \frac{1}{2}\|\mathcal{A} - \mathcal{B}\|_{\mathrm{F}}^2 + \|\mathcal{A}\|_* \right\}, \tag{5.2}$$

where $\mathcal{A}_{opt} = \mathcal{U} * \mathcal{S}_{\frac{\gamma}{\eta}} * \mathcal{V}$.

5.3 Preparations

We will now describe a method in subsection 5.3.1 to construct the high resolution image following the approach in Bose and Boo [11]. The construction of the high-resolution image can be modeled by solving a linear system $\mathbf{Lf} = \mathbf{g}$, where $\mathbf{L}$ is blurring operator, $\mathbf{g}$ is a vector constructed from the low resolution images, and $\mathbf{f}$ is the vectorized desired high resolution image.

Since the video has its own properties comparing to the low-resolution images used in the resolution enhancement model, we cannot use the video frames directly in the resolution enhancement model. In subsection 5.3.2, we will discuss the properties of the video in details and give a introduction on the method proposed in [15] to preprocess video frames.

5.3.1 Blurring Operator Construction

Low-resolution images can be viewed as outputs of an original high resolution image passing through a low-pass filter. Suppose the resolution of the low resolution images obtained from sensor is $M_1 \times M_2$ and that the actual length and width of the each pixel is $T_1 \times T_2$. The resolution of the high resolution image is assumed to be $N_1 \times N_2$ where $N_1 = K_1 M_1$, $N_2 = K_2 M_2$, and $K_1 \times K_2$ is the number of sensors. The length and width of each high resolution pixel is T_1/K_1 and T_2/K_2 respectively. For simplicity, we set $K = K_1 = K_2 = 2$, $M = M_1 = M_2$, $N = N_1 = N_2$, and $T = T_1 = T_2$.

We denote the scene we are interested in as $\mathbf{S}$, and write

$$\mathbf{S} = \left\{(n_1, n_2) \in \mathbb{R}^2 | 0 \le n_1 \le TN,\, 0 \le n_2 \le TN\right\}$$

Let $\mathbf{F}(x, y)$ be the intensity of the scene $\mathbf{S}$ at any point (x, y), see Figure 5.2. The discrete high resolution image has value at (n_1, n_2) as

$$\mathbf{F}(n_1, n_2) = \frac{1}{T^2} \int_{(n_2 - \frac{1}{2})T}^{(n_2 + \frac{1}{2})T} \int_{(n_1 - \frac{1}{2})T}^{(n_1 + \frac{1}{2})T} \mathbf{F}(x, y) dx dy,$$

which is the average intensity of all of the points inside the $(n_1, n_2)^{th}$ high-resolution pixel, i.e.

$$\left[\left(n_1 - \frac{1}{2}\right)T, \left(n_1 + \frac{1}{2}\right)T\right] \times \left[\left(n_2 - \frac{1}{2}\right)T, \left(n_2 + \frac{1}{2}\right)T\right].$$

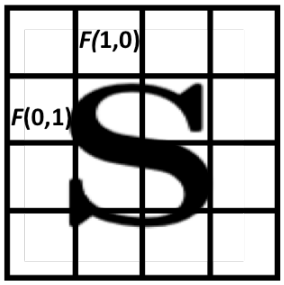


Figure 5.2: The high resolution image $\mathbf{F}(n_1, n_2)$

Let us suppose there is no displacement error of the sensors first, then for the

low resolution image taken from $(i,j)^{th}$ sensor, the average intensity at (n_1, n_2) is given by

$$\mathbf{G}_{i,j}(n_1,n_2) = \frac{1}{4T^2}\int_{(2(n_2-\frac{1}{2})+i)T}^{(2(n_2+\frac{1}{2})+i)T}\int_{(2(n_1-\frac{1}{2})+j)T}^{(2(n_1+\frac{1}{2})+j)T}\mathbf{F}(x,y)dxdy \tag{5.3}$$

where $0 \le i,j < K$ and $n_1, n_2 \in \mathbb{Z}$. See Figure 5.3.

Using the midpoint quadrature rule on (5.3),

$$\mathbf{G}_{i,j}(n_1,n_2) \approx \mathbf{F}\left(T(2n_1+i), T(2n_2+j)\right). \tag{5.4}$$

Thus, the observed high resolution image $\mathbf{G}$ is formed by composing all the low resolution images $\mathbf{G}_{ij}$,

$$\mathbf{G}\left(2n_1+i, 2n_2+j\right) = \mathbf{G}_{ij}\left(n_1,n_2\right), \quad 0 \le i,j < 2 \text{ and } (n_1,n_2) \in \mathbf{S}$$

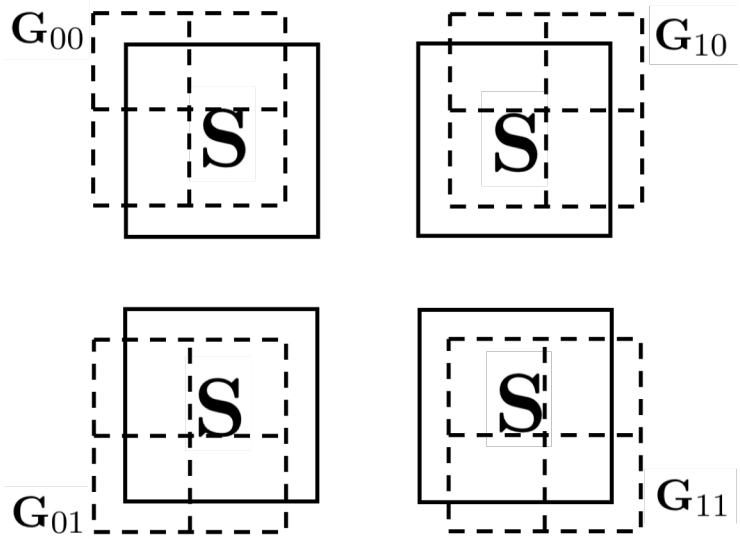


Figure 5.3: The low resolution images $\mathbf{G}_{00}$, $\mathbf{G}_{01}$, $\mathbf{G}_{10}$, and $\mathbf{G}_{11}$ taken by four different sensors with sub-pixel displacements.

We define the 2D downsampling matrix $\mathbf{D}_{i,j}$ and the upsampling matrix $\mathbf{U}_{i,j}$, for $0 \le i,j < 2$, as

$$\mathbf{D}_{i,j} = \mathbf{D}_j \otimes \mathbf{D}_i,$$

$$\mathbf{U}_{i,j} = \mathbf{U}_j \otimes \mathbf{U}_i,$$

where $\mathbf{D}_i = \mathbf{I}_M \otimes \mathbf{e}_i^\top$, $\mathbf{U}_i = \mathbf{I}_M \otimes \mathbf{e}_i$, $\mathbf{e}_0 = [1, 0]^\top$, and $\mathbf{e}_1 = [0, 1]^\top$.

Remark 5.3.1 The 2D downsampling matrix $\mathbf{D}_{i,j}$ and the upsampling matrix $\mathbf{U}_{i,j}$ have following properties.

- $\sum_{i,j=0}^{1} \mathbf{U}_{ij}\mathbf{D}_{ij} = \mathbf{I}_{N^2}$
- $\mathbf{D}_{ij}\mathbf{U}_{ij} = \mathbf{I}_{M^2}$
- $\mathbf{D}_{ij}\mathbf{U}_{i'j'} = 0$ if $i \neq i'$ and $j \neq j'$. ◇

Since each low resolution image $\mathbf{G}_{ij}$ can be seen as down-sampled from the observed high resolution image $\mathbf{G}$ and the observed high resolution image $\mathbf{G}$ can also be seen as up-sampled from the low resolution images $\{\mathbf{G}_{ij} | 0 \leq i, j < 2\}$, by leveraging the 2D sampling matrix $\mathbf{D}_{i,j}$ and the upsampling matrix $\mathbf{U}_{i,j}$, the matrix $\mathbf{G}_{i,j}$ and $\mathbf{G}$ can be written as

$$\mathbf{G}_{i,j} = \mathbf{D}_{ij}\mathbf{G}, \quad 0 \leq i, j < 2,$$

and

$$\mathbf{G} = \sum_{i,j=0}^{1} \mathbf{U}_{ij}\mathbf{G}_{ij}.$$

Using the periodic boundary condition (see Figure 5.4) and ordering the values of $\mathbf{F}$ and $\mathbf{G}$ row by row as $\mathbf{f}$ and $\mathbf{g}$ correspondingly, we obtain a linear system

$$\mathbf{L}\mathbf{f} = \mathbf{g}, \tag{5.5}$$

where $\mathbf{L}$ is a blurring operator,

$$\mathbf{L} = \mathbf{L}_x \otimes \mathbf{L}_y, \tag{5.6}$$

and $\mathbf{L}_x = \mathbf{L}_y = \texttt{circulant}([1, \frac{1}{2}, 0, \ldots, 0, \frac{1}{2}, 1])$.

$$\begin{array}{cccccc} \boldsymbol{d_4} & \boldsymbol{c_3} & \boldsymbol{d_3} & \boldsymbol{c_4} & \boldsymbol{d_4} & \boldsymbol{c_3} \\ \boldsymbol{b_2} & \boldsymbol{a_1} & \boldsymbol{b_1} & \boldsymbol{a_2} & \boldsymbol{b_2} & \boldsymbol{a_1} \\ \boldsymbol{d_2} & \boldsymbol{c_1} & \boldsymbol{d_1} & \boldsymbol{c_2} & \boldsymbol{d_2} & \boldsymbol{c_1} \\ \boldsymbol{b_4} & \boldsymbol{a_3} & \boldsymbol{b_3} & \boldsymbol{a_4} & \boldsymbol{b_4} & \boldsymbol{a_3} \\ \boldsymbol{d_4} & \boldsymbol{c_3} & \boldsymbol{d_3} & \boldsymbol{c_4} & \boldsymbol{d_4} & \boldsymbol{c_3} \\ \boldsymbol{b_2} & \boldsymbol{a_1} & \boldsymbol{b_1} & \boldsymbol{a_2} & \boldsymbol{b_2} & \boldsymbol{a_1} \end{array}$$

Figure 5.4: Example of periodic boundary conditions in 2D.

If there is displacement error, then the average intensity at (n_1, n_2) is given by

$$\mathbf{G}_{i,j}(n_1, n_2) = \frac{1}{4T^2} \int_{(2(n_2-\frac{1}{2})+i)T+\epsilon_{ij}^{y}}^{(2(n_2+\frac{1}{2})+i)T+\epsilon_{ij}^{y}} \int_{(2(n_1-\frac{1}{2})+j)T+\epsilon_{ij}^{x}}^{(2(n_1+\frac{1}{2})+j)T+\epsilon_{ij}^{x}} \mathbf{F}(x,y)dxdy,$$

where (ϵ^x, ϵ^y) denotes the displacement error through x direction and y direction [14].

Correspondingly,

$$\mathbf{L}_x = \texttt{circulant}([1, \frac{1}{2} + \epsilon^{\texttt{x}}\, 0, \ldots, \frac{1}{2} - \epsilon^{\texttt{x}}]),$$
$$\mathbf{L}_y = \texttt{circulant}([1, \frac{1}{2} + \epsilon^{\texttt{y}}\, 0, \ldots, \frac{1}{2} - \epsilon^{\texttt{y}}]).$$

For each low resolution image $\mathbf{g}_{i,j}$,

$$\mathbf{D}_{i,j}\mathbf{L}\mathbf{f} = \mathbf{g}_{i,j},$$

so the new blurring operator can be constructed as

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_{i,j}\mathbf{L}^{(0)} \\ \mathbf{D}_{i,j}\mathbf{L}^{(1)} \\ \mathbf{D}_{i,j}\mathbf{L}^{(2)} \\ \mathbf{D}_{i,j}\mathbf{L}^{(3)} \end{bmatrix} \tag{5.7}$$

.

Construct the vector $\mathbf{b}$ by

$$\mathbf{b} = \begin{bmatrix} \mathbf{g}_{i,j}^{(0)} \\ \mathbf{g}_{i,j}^{(1)} \\ \mathbf{g}_{i,j}^{(2)} \\ \mathbf{g}_{i,j}^{(3)} \end{bmatrix}, \tag{5.8}$$

then the new system is

$$\mathbf{A}\mathbf{f} = \mathbf{b}. \tag{5.9}$$

5.3.2 Video Frame Preprocessing

In the previous section, we introduced the model to construct high resolution image by using low resolution images captured by sensors. These sensors are fixed at some specific positions, so the model assumes that the perturbation of sensors are associated merely with translation. However, in video clips, the camera pans along with some directions, so it contains some motion effects. We need to eliminate these motion effects first, and then estimate sensor position and transitional displacement errors. The method to remove the motion effects we will use is proposed in [15]. Here, we briefly introduce the algorithm.

Given a reference frame $\mathbf{g}^{(0)}$, denote its nearby frames as $\left\{\mathbf{g}^{(k)}\right\}_{k\neq 0}$. Assume the motion effects of frames $\left\{\mathbf{g}^{(k)}\right\}_{k\neq 0}$ can be measured by a coordinate transforms of $\mathbf{g}^{(0)}$, so they can be written as

$$\mathbf{g}^{(k)}(\mathbf{R}_k\mathbf{x}_j - \mathbf{r}_k) \approx \mathbf{g}^{(0)}(\mathbf{x}_j), \tag{5.10}$$

where $\mathbf{R}_k$ is a 2×2 matrix, $\mathbf{r}_k$ is a 2×1 vector, $\mathbf{x}_j = [x, y]^\top$ is the coordinate of the pixel, and

$$(x', y') = \mathbf{R}_k\mathbf{x}_j - \mathbf{r}_k = \begin{bmatrix} c_0 & c_1 \\ c_3 & c_4 \end{bmatrix}\mathbf{x}_j + \begin{bmatrix} c_2 \\ c_5 \end{bmatrix} = \begin{bmatrix} c_0 & c_1 & c_2 \\ c_3 & c_4 & c_5 \end{bmatrix}\begin{bmatrix} \mathbf{x}_j \\ \mathbf{1} \end{bmatrix},$$

for all $\mathbf{x}_j \in \mathbf{S}$.

The purpose is to estimate $\mathbf{R}_k$ and $\mathbf{r}_k$ for each k such that the approximation

of (5.10) is as accurate as possible, so $\mathbf{R}_k$ and $\mathbf{r}_k$ could be obtained by solving an optimization problem that minimize the difference of $\mathbf{g}^{(k)}(\mathbf{R}_k\mathbf{x}_j - \mathbf{r}_k)$ and $\mathbf{g}^{(0)}(\mathbf{x}_j)$ for all pixels $\{\mathbf{x}_j | j \in \mathbf{I}, \mathbf{I}$ is the index set of pixels$\}$ in region $\mathbf{S}$ as follows.

$$\min_{\mathbf{c}^{(k)}} E(\mathbf{g}^{(k)}, \mathbf{g}^{(0)}) = \sum_{\mathbf{x}\in\mathbf{S}} \left(\mathbf{g}^{(k)}(\mathbf{R}_k\mathbf{x}_j - \mathbf{r}_k) - \mathbf{g}^{(0)}(\mathbf{x}_j)\right)^2 = \sum_{\mathbf{j}\in\mathbf{I}} e_j^2, \tag{5.11}$$

where $\mathbf{c}^{(k)} = [c_0, c_1, c_2, c_3, c_4, c_5]$ are the entries of $\mathbf{R}_k$ and $\mathbf{r}_k$, $\mathbf{S}$ can be part of the frame or the entire frame and the E can be seen as the residual.

Remark 5.3.2 If the components of $[x', y']^\top = \mathbf{R}_k\mathbf{x}_j - \mathbf{r}_k$ are not integers, $\mathbf{g}^{(k)}(x', y')$ is evaluated by interpolation in 2 dimensions, i.e.

$$\mathbf{g}^{(k)}(x', y') = (1-t)(1-u)g_1 + t(1-u)g_2 + tug_3 + (1-t)ug_4,$$

where (x', y') is bounded by the nearest four pixels (x_1, y_1), (x_2, y_1), (x_1, y_2), and (x_2, y_2) , i.e. $x_1 \le x' \le x_2$ and $y_1 \le y' \le y_2$. The values of g_i, $i = 1, \ldots, 4$ are defined as

$$\begin{aligned} g_1 &= \mathbf{g}^{(k)}(x_1, y_1), \\ g_2 &= \mathbf{g}^{(k)}(x_2, y_1), \\ g_3 &= \mathbf{g}^{(k)}(x_1, y_2), \\ g_4 &= \mathbf{g}^{(k)}(x_2, y_2), \end{aligned}$$

and the values of t and u are defined as

$$\begin{aligned} t &= \frac{x' - x_1}{x_2 - x_1}, \\ u &= \frac{y' - y_1}{y_2 - y_1}. \end{aligned}$$

. ♢

One can use various non-linear optimization approaches to solve (5.11) to estimate $\mathbf{R}_k$ and $\mathbf{r}_k$. Updates $\Delta\mathbf{c}$ to the solution $\mathbf{c}^{(k)}$, where $\Delta\mathbf{c}$ is computed by solving $(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \beta\mathbf{I})\Delta\mathbf{c} = -\mathbf{J}^{\mathrm{T}}\mathbf{e}$ where the j^{th} element of $\mathbf{e}$ is e_j. The procedure is shown in Algorithm 13.

Algorithm 13 Evaluating the coefficients of coordinate transforms [15]

Input: initial guesses of $\mathbf{c}$ and β, where $\mathbf{c} = [1,0,0,0,1,0]$ and $\beta = 0.001$ are suggested in [15].

Output: optimal solution $\mathbf{c}$

for $\mathbf{x}_j \in \mathbf{S}$ **do**

Compute $\mathbf{R}_j\mathbf{x}_j - \mathbf{r}_j$ and $\mathbf{g}^{(k)}(\mathbf{R}_j\mathbf{x}_j - \mathbf{r}_j)$

Compute the error e_j and $\frac{\partial e_j}{\partial c_k}$ and use them to form $\mathbf{J}$ and $\mathbf{e}$

end for

Compute $\Delta\mathbf{c}$ by solving $(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \beta\mathbf{I})\Delta\mathbf{c} = -\mathbf{J}^{\mathrm{T}}\mathbf{e}$ and update $\mathbf{c}^{(k)} = \mathbf{c} + \Delta\mathbf{c}^{(k)}$

Continue the above for-loop until the error $E(\mathbf{g}^{(k)}, \mathbf{g}^{(0)})$ is below a threshold or a fix number of iterations has been reached.

If the error $E(\mathbf{g}^{(k)}, \mathbf{g}^{(0)})$ diverges through iterations, set $\beta = 100\beta$, as suggested in [15], to recompute $\Delta\mathbf{c}$.

Then we need to determine if $\mathbf{g}^{(k)}(\mathbf{R}_k\mathbf{x}_j - \mathbf{r}_k)$ is close enough to $\mathbf{g}^{(0)}(\mathbf{x}_j)$. We set threshold $\tau = 25$ and if the PSNR, defined in Definition 5.3.3, of $\mathbf{g}^{(k)}(\mathbf{R}_k\mathbf{x}_j - \mathbf{r}_k)$ and $\mathbf{g}^{(0)}(\mathbf{x}_j)$ is greater than τ, then we discard that frame.

Definition 5.3.3 *[15] The peak signal-to-noise ration (PSNR) is defined as*

$$PSNR\ of\ [\mathbf{F} - \mathbf{F}^*] = 10\log_{10}\frac{255^2}{\frac{1}{3mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=r,g,b}\left(\mathbf{F}_{i,j,k} - \mathbf{F}^*_{i,j,k}\right)},$$

that compares the reconstructed image $\mathbf{F}^*$ *with the original image* $\mathbf{F}$*.*

By Algorithm 13, it applies a affine transform to translate $\left\{\mathbf{g}^{(k)}\right\}_{k\neq 0}$ to $\mathbf{g}^{(0)}$. Then, we will determine the sensor index (s_k^x, s_k^y) and displacement error $(\epsilon_k^x, \epsilon_k^y)$.

By (5.10),

$$\mathbf{g}^{(0)}(\mathbf{x}) \approx \mathbf{g}^{(k)}\left(\mathbf{R}_k(\mathbf{x} - \mathbf{R}_k^{-1}\mathbf{r}_k)\right) \tag{5.12}$$

Algorithm 14 writes $\mathbf{R}_k^{-1}\mathbf{r}_k$ as

$$\mathbf{R}_k^{-1}\mathbf{r}_k = \mathbf{u} + \frac{1}{2}\begin{bmatrix} s^x \\ s^y \end{bmatrix}\mathbf{x} + \frac{1}{2}\begin{bmatrix} \epsilon^x \\ \epsilon^y \end{bmatrix}, \tag{5.13}$$

where $s^x, s^y \in 0, 1$, $|\epsilon^x| < \frac{1}{2}$, and $|\epsilon^y| < \frac{1}{2}$. We can then obtain the preprocessed low resolution frames $\left\{\hat{\mathbf{g}}^{(k)}\right\}_{k\neq 0}$ whose the motion effects are removed.

Algorithm 14 Removing motion effects [15]

Input: $\mathbf{R}_k$, $\mathbf{r}_k$, $\mathbf{g}^{(k)}$, and $\mathbf{g}^{(0)}$

Output: $\hat{\mathbf{g}}^{(k)}$, s^x, s^y ,ϵ^x, and ϵ^y

Compute $[r_1', r_2']^\top = \mathbf{R}^{-1}\mathbf{r}$

Set $\mathbf{u} = \left[\lfloor r_1' + \frac{1}{4}\rfloor, \lfloor r_2' + \frac{1}{4}\rfloor\right]^\top$, and let $[d_1', d_2'] = [r_1', r_2'] - \mathbf{u}^T$

Form $[s^x, s^y] = \left[\lfloor 2d_1' + \frac{1}{2}\rfloor, \lfloor 2d_2' + \frac{1}{2}\rfloor\right]$

Form $[\epsilon^x, \epsilon^y] = [\lfloor 2d_1' - s^x\rfloor, \lfloor 2d_2' - s^y\rfloor]$

Compute $\hat{\mathbf{g}}^{(k)} = \hat{\mathbf{g}}^{(k)}(\mathbf{R}_k(\mathbf{x} - \mathbf{u}))$

5.4 Enhancing Resolution of Video Clips

5.4.1 Model Setup

Given video frames $\left\{\mathbf{g}^{(k)}\right\}$, by using Algorithm 13 and Algorithm 14, we obtain the sensor index (s^x, s^y), displacement errors (ϵ^x, ϵ^y), and the preprocessed low resolution frames $\left\{\hat{\mathbf{g}}^{(k)}\right\}$. Then, we could form the $\mathbf{b}^{(k)}$ in the linear system (5.9) by concatenating four column vectors $\hat{\mathbf{g}}^{(4k)}$, $\hat{\mathbf{g}}^{(4k+1)}$, $\hat{\mathbf{g}}^{(4k+2)}$, and $\hat{\mathbf{g}}^{(4k+3)}$ and along the 1^{st} dimension, where k is from 0 to n, and n is the number of observed high resolution frames, $\left\{\mathbf{b}^{(k)}\right\}$, we will use. The corresponding blurring operator matrix $\mathbf{A}^{(k)}$ is constructed using the information of the sensor index (s^x, s^y) and displacement errors (ϵ^x, ϵ^y).

The optimization problem is as follows. For $k = 1 : n$,

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} \quad \frac{1}{2}\sum_{i=1}^{3} \|\mathbf{A}^{(k)}\mathbf{x}_i - \mathbf{b}_i^{(k)}\|_2^2 + \|\mathcal{X}^{(k)}\|_*, \tag{5.14}$$

where $\mathbf{x}_1, \mathbf{x}_2$, and $\mathbf{x}_3$ are the components of the vectorized desired high resolution reference frame in red, green, and blue bands. When $k = 0$, $\mathcal{X}^{(0)} = \mathcal{X}$ is a third order tensor which includes the lateral slices constructed by unknown $\mathbf{x}_1, \mathbf{x}_2$, and $\mathbf{x}_3$ we need to solve. Figure 5.5 shows how to construct a lateral slice from a vectorized frame and Figure 5.6 illustrate the construction of $\mathcal{X}^{(0)}$.

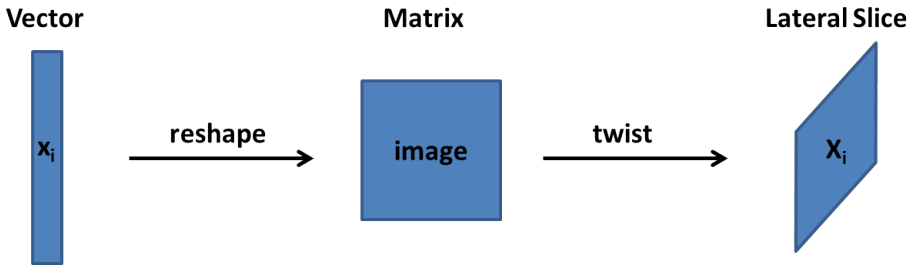


Figure 5.5: The relationship between $\mathbf{x}_i$ and $\mathbf{X}_i$.

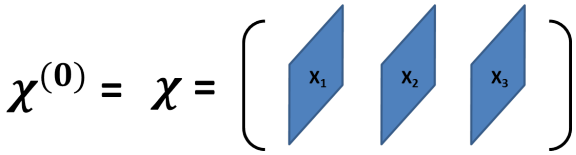


Figure 5.6: The third-order tensor $\mathcal{X}^{(0)}$.

When $k \neq 0$, $\mathcal{X}^{(k)}$ is a third order tensor which includes the lateral slices constructed by unknown $\mathbf{x}_1, \mathbf{x}_2$, and $\mathbf{x}_3$ we need to solve and known $\mathcal{X}^{(0)}, \cdots, \mathcal{X}^{(k-1)}$. Figure 5.7 shows the construction of $\mathcal{X}^{(k)}$. By this setting, $\mathcal{X}^{(k)}$ can involve more different frames by iterations.

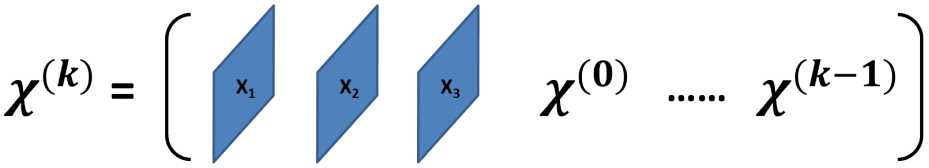


Figure 5.7: The third order tensor $\mathcal{X}^{(k)}$.

5.4.2 Algorithm

To solve the optimization problem, we use alternating direction method of multipliers (ADMM) method. Here, we give a detailed analysis for the k^{th} step, i.e.

the optimization function (5.14), that is equivalent to solve the following objective function.

$$\begin{aligned} \mathtt{min}_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} \quad & \frac{1}{2}\sum_{\mathtt{i}=1}^{3} \|\mathbf{A}^{(\mathtt{k})}\mathbf{x}_{\mathtt{i}} - \mathbf{b}_{\mathtt{i}}^{(\mathtt{k})}\|_2^2 + \|\mathcal{Z}^{(\mathtt{k})}\|_* \\ & \mathtt{s.t.} \qquad \mathcal{X}^{(\mathtt{k})} = \mathcal{Z}^{(\mathtt{k})} \end{aligned} ,$$

where $\mathcal{Z}$ is an auxiliary splitting parameterization that enabling the decoupling optimization problem.

The augmented Lagrangian for this problem is

$$\begin{aligned} \min L(\mathcal{X}^{(k)}, \mathcal{Y}^{(k)}, \mathcal{Z}^{(k)}) &= \frac{1}{2}\sum_{i=1}^{3} \|\mathbf{A}^{(k)}\mathbf{x}_i - \mathbf{b}_i^{(k)}\|_2^2 + \gamma\|\mathcal{Z}^{(k)}\|_* \\ &+ \langle \mathcal{Y}^{(k)}, \mathcal{X}^{(k)} - \mathcal{Z}^{(k)} \rangle + \frac{\eta}{2}\|\mathcal{X}^{(k)} - \mathcal{Z}^{(k)}\|_{\mathrm{F}}^2, \end{aligned}$$

where $\mathcal{Y}$ stores dual variables, $\frac{\eta}{2}\|\mathcal{X}^{(k)} - \mathcal{Z}^{(k)}\|_{\mathrm{F}}^2$ is the penalty termand $\eta > 0$ is the penalty parameter.

The solution of $L(\mathcal{X}^{(k)}, \mathcal{Y}^{(k)}, \mathcal{Z}^{(k)})$ is obtained by the ADMM method in the following steps,

$$\mathcal{X}_{m+1}^{(k)} = \mathrm{argmin}_{\mathcal{X}^{(k)}} L(\mathcal{X}^{(k)}, \mathcal{Y}_m^{(k)}, \mathcal{Z}_m^{(k)}) \tag{5.15}$$

$$\mathcal{Z}_{m+1}^{(k)} = \mathrm{argmin}_{\mathcal{Z}^{(k)}} L(\mathcal{X}_{m+1}^{(k)}, \mathcal{Y}_m^{(k)}, \mathcal{Z}^{(k)}) \tag{5.16}$$

$$\mathcal{Y}_{m+1}^{(k)} = \mathcal{Y}_m^{(k)} + \eta(\mathcal{X}_{m+1}^{(k)} - \mathcal{Z}_{m+1}^{(k)}). \tag{5.17}$$

First, we can simplify (5.15) due to $\|\mathcal{Z}^{(k)}\|_*$ being a constant here.

$$\begin{aligned} \mathcal{X}_{m+1}^{(k)} &= \mathrm{argmin}_{\mathcal{X}^{(k)}} L(\mathcal{X}^{(k)}, \mathcal{Y}_m^{(k)}, \mathcal{Z}_m^{(k)}) \\ &= \mathrm{argmin}_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} \frac{1}{2}\sum_{i=1}^{3} \|\mathbf{A}^{(k)}\mathbf{x}_i - \mathbf{b}_i^{(k)}\|_2^2 + \gamma\|\mathcal{Z}_m^{(k)}\|_* \\ &+ \langle \mathcal{Y}_m^{(k)}, \mathcal{X}^{(k)} - \mathcal{Z}_m^{(k)} \rangle + \frac{\eta}{2}\|\mathcal{X}^{(k)} - \mathcal{Z}_m^{(k)}\|_{\mathrm{F}}^2 \end{aligned}$$

$$
\begin{aligned}
&= \operatorname{argmin}_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} \frac{1}{2}\sum_{i=1}^{3} \|\mathbf{A}^{(k)}\mathbf{x}_i - \mathbf{b}_i^{(k)}\|_2^2 \\
&+ \langle \mathcal{Y}_m^{(k)}, \mathcal{X}^{(k)} - \mathcal{Z}_m^{(k)} \rangle + \frac{\eta}{2}\|\mathcal{X}^{(k)} - \mathcal{Z}_m^{(k)}\|_{\mathrm{F}}^2 \qquad (5.18)
\end{aligned}
$$

Moreover, from the setting of $\mathcal{X}^{(k)}$, we know the following equivalence property,

$$
\begin{aligned}
&\min \frac{1}{2}\sum_{i=1}^{3} \|\mathbf{A}^{(k)}\mathbf{x}_i - \mathbf{b}_i^{(k)}\|_2^2 + \|\mathcal{Z}^{(k)}\|_* \\
&\quad \texttt{s.t.} \qquad \mathcal{X}^{(\texttt{k})} = \mathcal{Z}^{(\texttt{k})}
\end{aligned}
$$

is equivalent to

$$
\begin{aligned}
&\min \frac{1}{2}\sum_{i=1}^{3} \|\mathbf{A}^{(k)}\mathbf{x}_i - \mathbf{b}_i^{(k)}\|_2^2 + \|\mathcal{Z}^{(k)}\|_* \\
&\quad \texttt{s.t.} \qquad \mathcal{X} = \mathcal{Z} \qquad\qquad ,
\end{aligned}
$$

due to

$$
\begin{aligned}
\mathcal{X} &= \mathcal{X}^{(k)}(:, 1:3, :), \\
\mathcal{Z} &= \mathcal{Z}^{(k)}(:, 1:3, :),
\end{aligned}
$$

and $\mathcal{X}^{(k)}$ and $\mathcal{Z}^{(k)}$ are constants except the first lateral slices.

Thus, we can simplify (5.18) further as

$$
\begin{aligned}
\mathcal{X}_{m+1}^{(k)} &= \operatorname{argmin}_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} \frac{1}{2}\sum_{i=1}^{3} \|\mathbf{A}^{(k)}\mathbf{x}_i - \mathbf{b}_i^{(k)}\|_2^2 + \langle \mathcal{Y}_m, \mathcal{X} - \mathcal{Z}_m \rangle + \frac{\eta}{2}\|\mathcal{X} - \mathcal{Z}_m\|_{\mathrm{F}}^2 \\
&= \operatorname{argmin}_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} \frac{1}{2}\sum_{i=1}^{3} \|\mathbf{A}^{(k)}\mathbf{x}_i - \mathbf{b}_i^{(k)}\|_2^2 + \sum_{i=1}^{3} \mathbf{y}_i^{\mathrm{T}}(\mathbf{x}_i - \mathbf{z}_i) + \frac{\eta}{2}\sum_{i=1}^{3} \|\mathbf{x}_i - \mathbf{z}_i\|_{\mathrm{F}}^2,
\end{aligned}
$$

and then solve it by LSQR. When we have $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$, we reshape them to $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ and form the tensor $\mathcal{X}_{m+1}^{(k)}$ as explained in Figure 5.7.

To solve the $\mathcal{Z}_{m+1}^{(k)}$ in equation (5.16), it is equivalent to solve

$$\begin{aligned}
&\text{argmin}_{\mathcal{Z}^{(k)}} L(\mathcal{X}_{m+1}^{(k+1)}, \mathcal{Y}_m^{(k)}, \mathcal{Z}^{(k)}) \\
&= \text{argmin}_{\mathcal{Z}^{(k)}} \gamma \|\mathcal{Z}^{(k)}\|_* + \langle \mathcal{Y}_m^{(k)}, \mathcal{X}^{(k+1)} - \mathcal{Z}^{(k)} \rangle + \frac{\eta}{2} \|\mathcal{X}^{(k+1)} - \mathcal{Z}^{(k)}\|_{\mathrm{F}}^2 \\
&= \text{argmin}_{\mathcal{Z}^{(k)}} \gamma \|\mathcal{Z}^{(k)}\|_* + \langle \mathcal{Y}_m^{(k)}, \mathcal{X}^{(k+1)} \rangle - \langle \mathcal{Y}_m^{(k)}, \mathcal{Z}^{(k)} \rangle \\
&+ \frac{\eta}{2} \left(\langle \mathcal{X}^{(k+1)}, \mathcal{X}^{(k+1)} \rangle - 2 \langle \mathcal{X}^{(k+1)}, \mathcal{Z}^{(k)} \rangle + \langle \mathcal{Z}^{(k)}, \mathcal{Z}^{(k)} \rangle \right) \\
&= \text{argmin}_{\mathcal{Z}^{(k)}} \gamma \|\mathcal{Z}^{(k)}\|_* + \frac{\eta}{2} \left\| \mathcal{Z}^{(k)} - \left(\frac{\mathcal{Y}^{(k)}}{\eta} + \mathcal{X}_m^{(k+1)} \right) \right\|_{\mathrm{F}}^2
\end{aligned} \tag{5.19}$$

According to Theorem 5.2.5, the solution of $\mathcal{Z}_{m+1}^{(k)}$ is

$$\mathcal{Z}_{m+1}^{(k)} = \mathcal{U} * \mathcal{S}_{\frac{\gamma}{\eta}} * \mathcal{V},$$

where $\mathcal{U}$, $\mathcal{S}$ and $\mathcal{V}$ are from the t-SVD of $(\frac{\mathcal{Y}_m^{(k)}}{\eta} + \mathcal{X}_{m+1}^{(k)})$,

$$\mathcal{S}_{\frac{\gamma}{\eta}}(:,:,i) = \texttt{diag}(\{\sigma_{\texttt{ij}} - \frac{\gamma}{\eta}\}_+) = \begin{cases} 0, & \texttt{if } \sigma_{\texttt{ij}} - \frac{\gamma}{\eta} \leq 0 \\ \sigma_{ij} - \frac{\gamma}{\eta}, & \texttt{if } \sigma_{\texttt{ij}} - \frac{\gamma}{\eta} > 0, \end{cases}$$

and the σ_{ij}'s are the singular values of the i_{th} frontal slice of $\mathcal{S}$.

5.5 Numerical Experiments

In this section, we implement the algorithm described in the previous section on two video datasets. The first video, named Barbara, is made of images with resolution 240×240. We use the 1^{st} video frame, as an example frame, to enhance its resolution, see Figure 5.8. Its nearby images are shown in Figure 5.9, and the pre-processed images of nearby images are shown in Figure 5.10. By comparing the left boundaries of the frames in Figure 5.8 and Figure 5.10, we can see the motion effects has been removed. The enhanced image by using our method in Figure 5.12. Bilinear interpolation is a classical method to enhance resolution from low resolution frames. Here, we show the enhanced frame using bilinear interpolation in Figure 5.11 for

reference. One can see, in general, Figure 5.12 is more clear than Figure 5.11. In particular, the Figure 5.12 preserves some detailed information. For example, the tiny black texts on the blue book on the bookshelf. A difference image between Figure 5.11 and Figure 5.12 is shown in Figure 5.13, and it could present the places of preserved information to some extent.

Figure 5.8: The original 1_{st} video frame.

The second video is panning a stack of books and it can be download at [2]. The video is in CIF format with resolution 352×288. We use the 100^{th} video frame, see Figure 5.14, as an example frame to enhance its resolution. The nearby images we will use are plotted in Figure 5.15, and the pre-processed images of nearby images are shown in Figure 5.15. One can find that the pre-processed images are transformed to have almost the same scene of 100^{th} video frame.

The parameters obtained from Algorithm 14 are shown in Table 5.1. The first column is frame index, the second column is the position index, and the third column is the displacement error. The 97^{th} frame and 103^{th} frame are discarded by the threshold τ defined in subsection 5.3.2.

The enhanced image by using bilinear interpolation is shown in Figure 5.17, and the enhanced image by using our method in Figure 5.18. Comparing to Figure 5.17, Figure 5.18 is more clear and the boundary of letters and numbers printed on the books are easier to be recognized, for instance, the number of 98 on the yellow book.

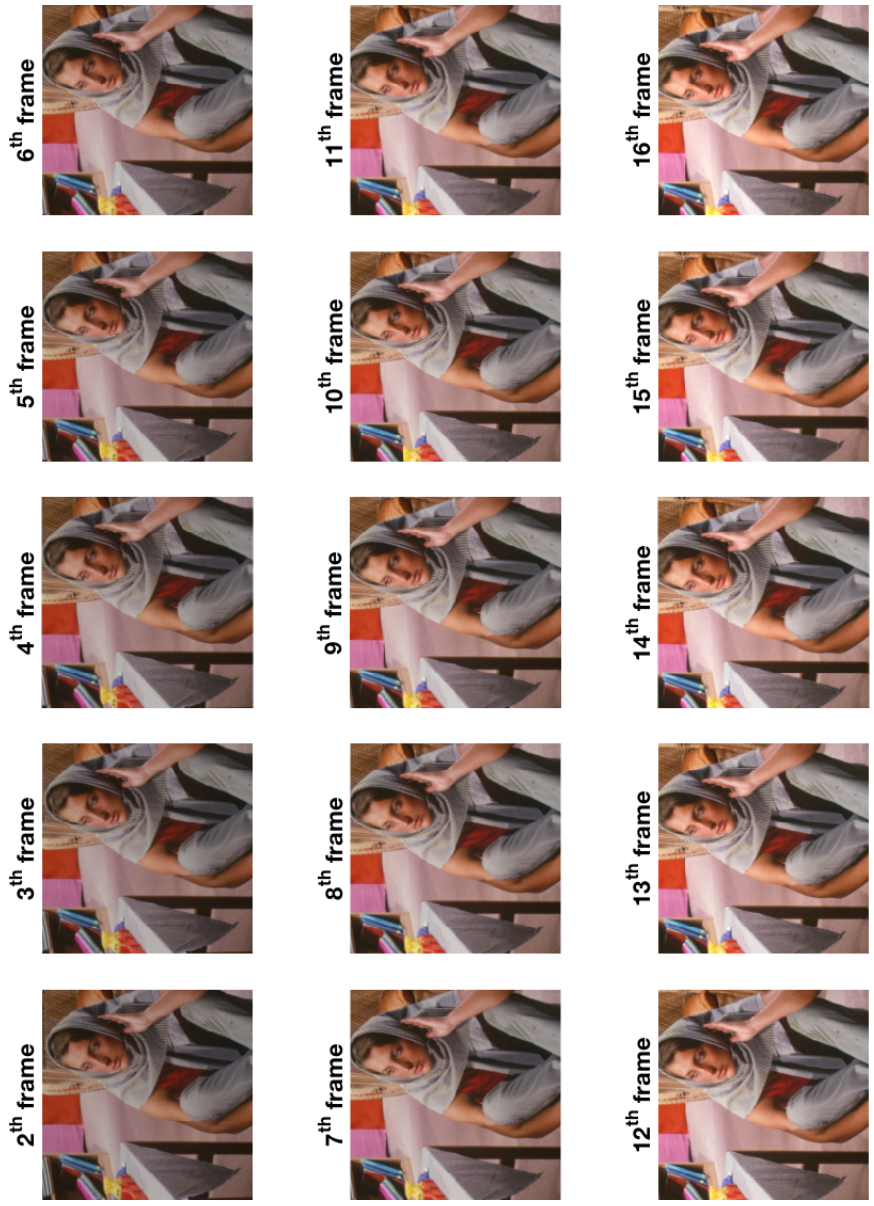
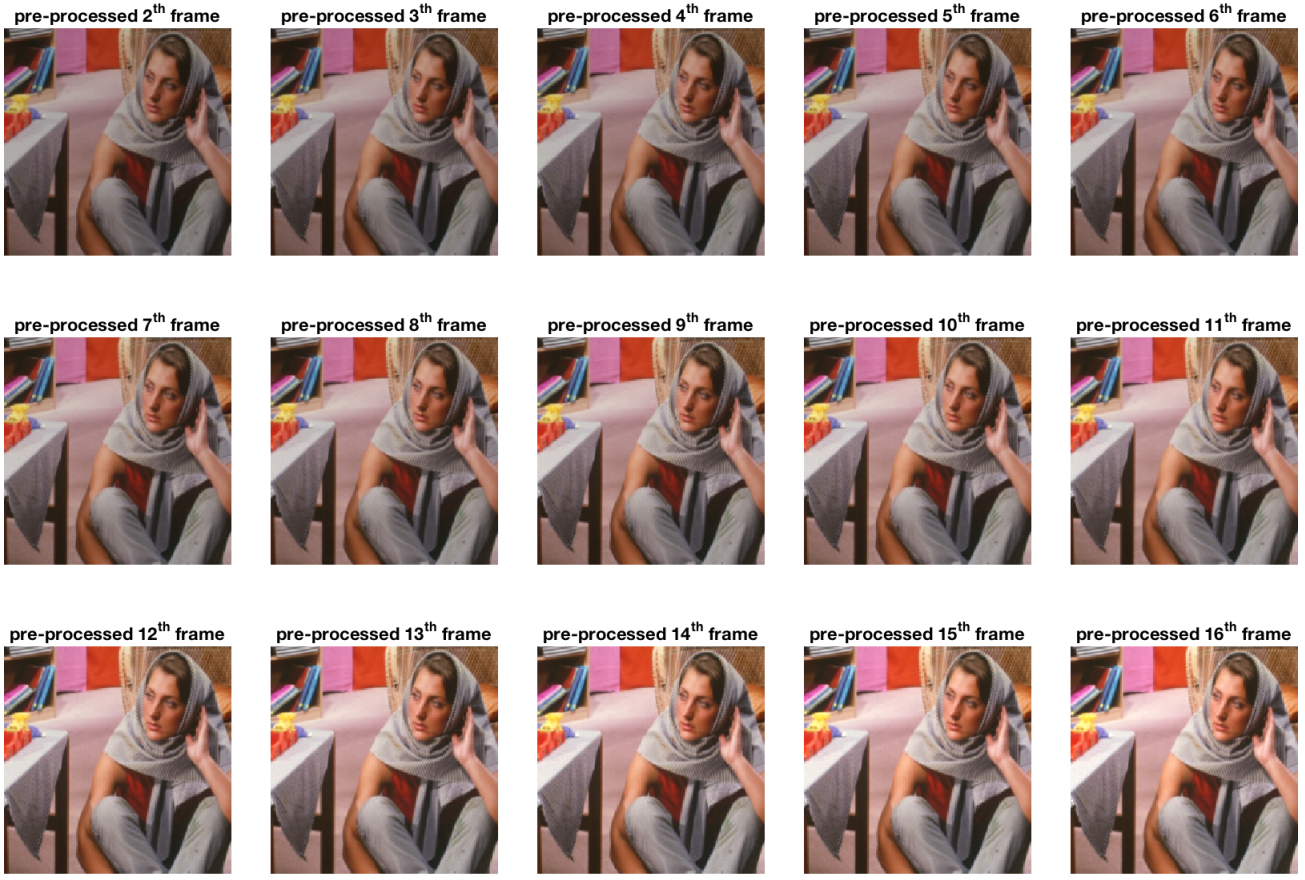


Figure 5.9: The nearby frames of 1_{st} video frame.



;

Figure 5.10: The pre-processed nearby frames of 1_{st} video frame.

Figure 5.11: The enhanced 1_{st} video frame by using bilinear interpolation.

Figure 5.12: The enhanced 1_{st} video frame by using tensor algorithm.

Figure 5.13: The difference image between Figure 5.11 and Figure 5.12 (Figure 5.12 - Figure 5.11).

Table 5.1: Parameters of Nearby Frames

Frame Index	(s_x, s_y)	(ϵ_x, ϵ_y)
99	(0,1)	(-0.3669, 0.4279)
101	(0,1)	(0.0072, -0.2228)
98	(1,1)	(0.2642, -0.4646)
102	(0,1)	(-0.3914, 0.4008)
96	(1,1)	(0.3149, 0.4955)
104	(0,1)	(-0.0490, 0.3675)
95	(1,1)	(0.4284, 0.4830)
105	(0,1)	(-0.1857, -0.3150)
94	(1,1)	(0.3343, 0.0110)
106	(0,0)	(-0.1230, -0.1803)
93	(1,1)	(0.4555, -0.1103)
107	(0,1)	(-0.0182, 0.1526)
92	(1,0)	(0.4698, 0.0845)
108	(0,0)	(-0.3148, 0.4120)
91	(1,0)	(0.3183, 0.0104)

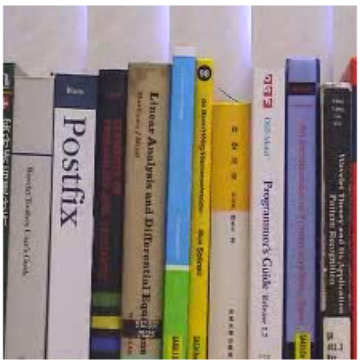


Figure 5.14: The original 100_{th} video frame.

5.6 Summary

In this chapter, we developed a model to enhance the resolution of video clips using t-product based operators and a tensor nuclear norm. It could involve the independent information from nearly frames, and reduce the redundant shared information at the same time. The model we set up is aimed to be general. It can be extended to enhance the resolution of several frames simultaneously and include different deblurring and denoising techniques in the model.

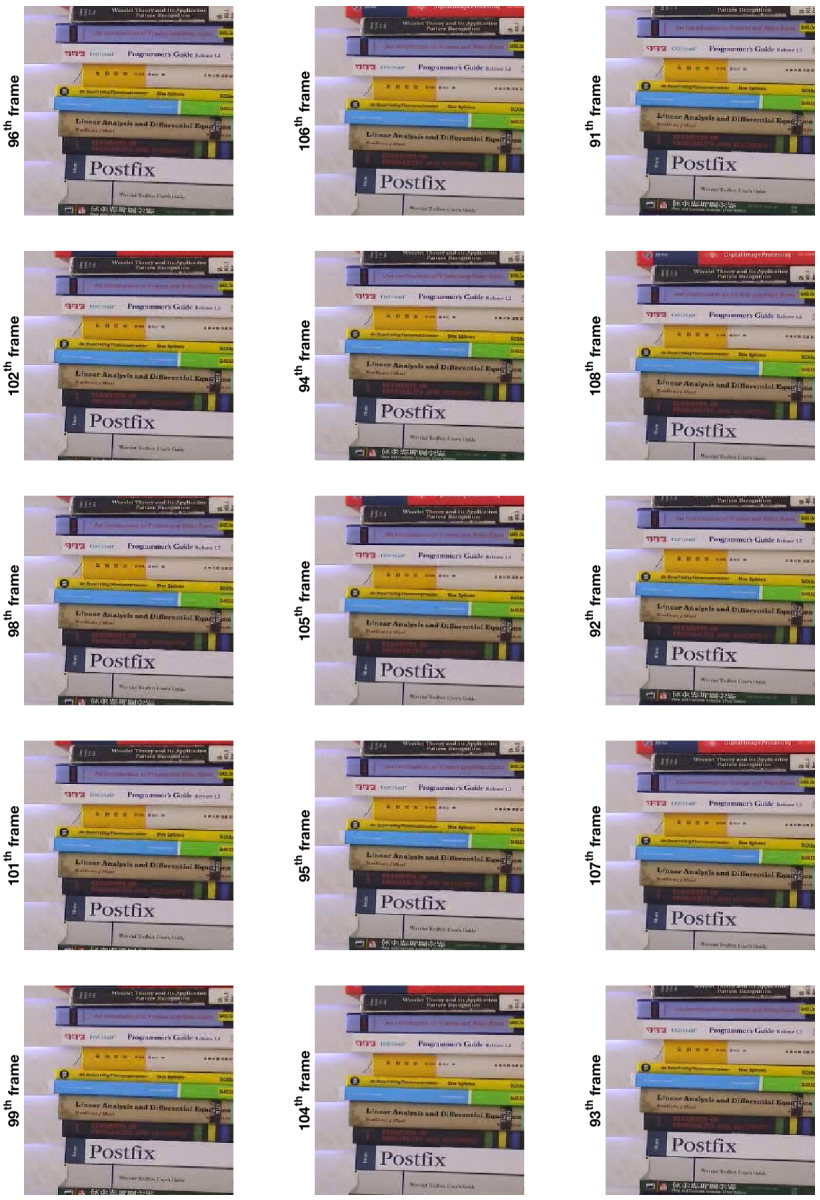


Figure 5.15: The nearby frames of 100_{th} video frame.

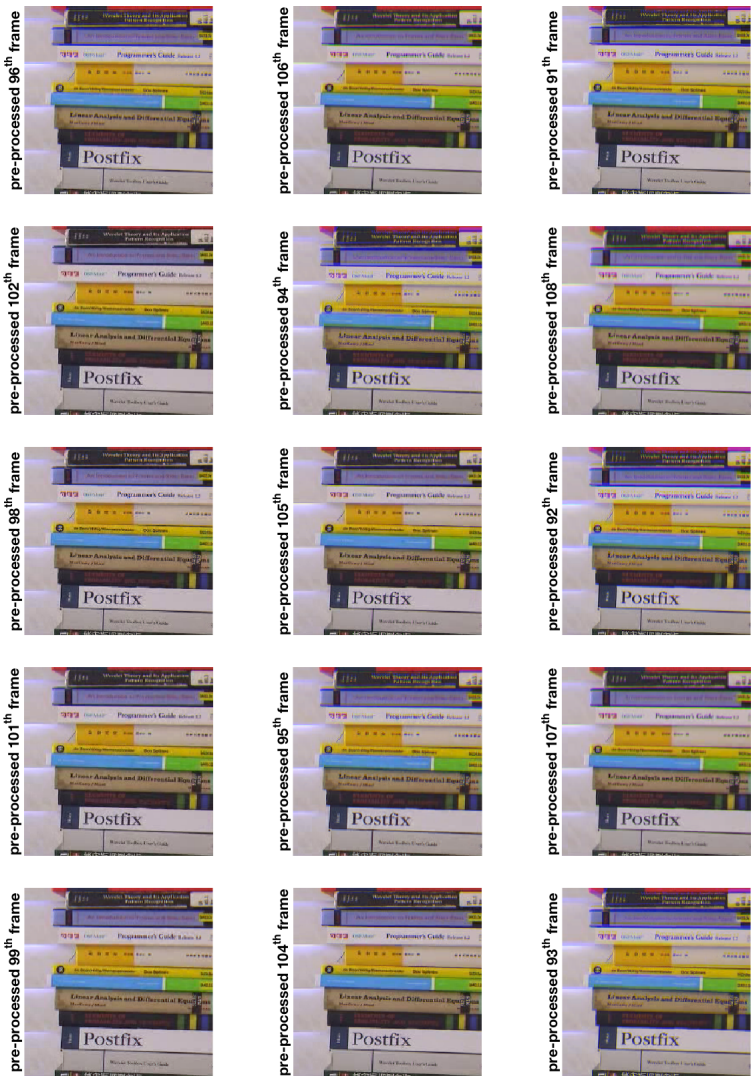


Figure 5.16: The pre-processed nearby frames of 100_{th} video frame.

Figure 5.17: The enhanced 100_{th} video frame by using bilinear interpolation.

Figure 5.18: The enhanced 100_{th} video frame by using tensor algorithm.

Chapter 6

Conclusions and Future Work

In this thesis, we presented four novel tensor-based methods in the field of randomized algorithms, dynamical system, image processing and video processing.

The randomized tensor singular value decomposition (rt-SVD) we designed combines both of the advantages of the matrix version randomized algorithms and t-SVD. It can produce a factorization with similar properties to the tensor SVD (t-SVD), but is more computationally efficient on very large datasets. We applied it in the application of facial recognition, and the numerical results show its superiority in computation speed.

The tensor proper orthogonal decomposition (POD) method is designed for the model reduction of dynamical systems. We proved by theory that the representation accuracy of the t-SVD is superior to that of the matrix based SVD. These theoretical results substantiated the conclusion that tensor POD generate a better basis for the snapshots in accuracy. We also provided the numerical experiments on a 2D diffusion system, and illustrated the tensor POD basis can reveal more information than the POD basis from the snapshots of solutions. It implies the t-SVD may provide a superior compressed representation of information for the time-related data, which is promising in applications where limited number of snapshots are available to form a reduced dynamic system..

The t-PQR method and the tensor-based video resolution enhancement model were both articulated as optimization problems. They leverage tensor operators to solve problems or build models. For the multi-frame blind deconvolution optimization model, the t-PQR method can be used to select most representative frames in a principled manner, as opposed to the heuristic nature of the methods employed currently while avoiding information loss caused by the inappropriate choice of the

control frame. The video resolution enhancement model we design, effectively compiles independent information from nearby frames while reducing the redundant information simultaneously. The model is aimed to be general. It can be extended to enhance the resolution of several frames simultaneously and include different deblurring and denoising techniques in the model.

Much work remains to be done in the future. For instance, we have covered projection based randomized tensor approaches, and in particular Gaussian random projections. It would be interesting to explore how other projectors (e.g. Subsampled Randomized Hadamard Transform, Count Sketch) as well as non-projection based tensor sketching propositions (e.g. Column Selection), perform. Moreover, the t-SVD II method provides good performance to deal with the singular values decaying at different rates as a deterministic method. It would be very interesting to apply randomized algorithms on t-SVD II to possibly have higher recognition rate and less computation cost.

Regarding the tensor POD section, we were mainly focusing on the linear dynamical systems. Extension of the approach to efficiently solve non-linear dynamical systems would be of great importance. In addition, further attempts to reduce the size of reduced model with tensor POD method are desired.

On the tensor blind deconvolution work, an L_2 noise model was assumed, extension of the framework to robust (e.g. L_1) noise models, or other exotic noise models would further extend the robustness of the approach.

Per video enhancing, our current video resolution enhancement model does not involve modern technologies in deblurring and denoising, and we are interested in including them to further improve the quality of the video clips. Furthermore, it may be possible to derive a more general model and perform enhancement for all of the video frames simultaneously.

Bibliography

[1] *The extended Yale Face Dataset B.* http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html.

[2] *Video clip and full results downloadable at.* http://www.math.cuhk.edu.hk/rchan/paper/csx.

[3] Dimitris Achlioptas, *Database-friendly random projections: Johnson–Lindenstrauss with binary coins*, Journal of computer and System Sciences, 66 (2003), pp. 671–687.

[4] Konstantin Afanasiev and Michael Hinze, *Adaptive control of a wake flow using proper orthogonal decomposition*, Lecture Notes in Pure and Applied Mathematics, (2001), pp. 317–332.

[5] Nir Ailon and Bernard Chazelle, *Approximate nearest neighbors and the fast Johnson–Lindenstrauss transform*, in Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, ACM, 2006, pp. 557–563.

[6] ——, *The fast Johnson–Lindenstrauss transform and approximate nearest neighbors*, SIAM Journal on Computing, 39 (2009), pp. 302–322.

[7] CA Andrews, JM Davies, and GR Schwarz, *Adaptive data compression*, Proceedings of the IEEE, 55 (1967), pp. 267–277.

[8] HT Banks, Michele L Joyner, Buzz Wincheski, and William P Winfree, *Nondestructive evaluation using a reduced-order computational methodology*, Inverse Problems, 16 (2000), p. 929.

[9] Casey Battaglino, Grey Ballard, and Tamara G Kolda, *A practical randomized CP tensor decomposition*, arXiv preprint arXiv:1701.06600, (2017).

[10] David J Biagioni, Daniel Beylkin, and Gregory Beylkin, *Randomized interpolative decomposition of separated representations*, Journal of Computational Physics, 281 (2015), pp. 116–134.

[11] NK Bose and KJ Boo, *High-resolution image reconstruction with multisensors*, International Journal of Imaging Systems and Technology, 9 (1998), pp. 294–304.

[12] Jian-Feng Cai and Lixin Shen, *Tight frame based method for high-resolution image reconstruction*, (2010).

[13] AT&T Laboratories Cambridge, *The database of faces.* http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[14] Raymond H Chan, Tony F Chan, Lixin Shen, and Zuowei Shen, *Wavelet algorithms for high-resolution image reconstruction*, SIAM Journal on Scientific Computing, 24 (2003), pp. 1408–1432.

[15] Raymond H Chan, Zuowei Shen, and Tao Xia, *A framelet algorithm for enhancing video stills*, Applied and Computational Harmonic Analysis, 23 (2007), pp. 153–170.

[16] Saifon Chaturantabut and Danny C Sorensen, *Nonlinear model reduction via discrete empirical interpolation*, SIAM Journal on Scientific Computing, 32 (2010), pp. 2737–2764.

[17] Sanjoy Dasgupta and Anupam Gupta, *An elementary proof of a theorem of Johnson and Lindenstrauss*, Random Structures & Algorithms, 22 (2003), pp. 60–65.

[18] B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Pub., 1995.

[19] Lieven De Lathauwer, *Signal processing based on multilinear algebra*, Katholieke Universiteit Leuven, 1997.

[20] 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.

[21] Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang, *Matrix approximation and projective clustering via volume sampling*, in Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, Society for Industrial and Applied Mathematics, 2006, pp. 1117–1126.

[22] Petros Drineas and Michael W Mahoney, *A randomized algorithm for a tensor-based generalization of the singular value decomposition*, Linear algebra and its applications, 420 (2007), pp. 553–571.

[23] Daniel M Dunlavy, Tamara G Kolda, and W Philip Kegelmeyer, *Multilinear algebra for analyzing data with multiple linkages*, Graph Algorithms in the Language of Linear Algebra, (2011), pp. 85–114.

[24] Michael Elad and Arie Feuer, *Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images*, IEEE transactions on image processing, 6 (1997), pp. 1646–1658.

[25] M. Elad and A. Feuer, *Superresolution restoration of an image sequence: adaptive filtering approach*, IEEE Transactions on Image Processing, 8 (1999), pp. 387–395.

[26] Michael Elad and Yacov Hel-Or, *A fast super-resolution reconstruction algorithm for pure translational motion and common space-invariant blur*, IEEE Transactions on Image Processing, 10 (2001), pp. 1187–1193.

[27] Gregory Ely, Shuchin Aeron, Ning Hao, and Misha E Kilmer, *5d seismic data completion and denoising using a novel class of tensor decompositions*, Geophysics, 80 (2015), pp. V83–V95.

[28] Gregory Ely, Shuchin Aeron, Ning Hao, Misha E Kilmer, et al., *5D and 4D pre-stack seismic data completion using tensor nuclear norm (TNN)*, preprint, (2013).

[29] Ying-Wai Daniel Fan and James G Nagy, *An efficient computational approach for multiframe blind deconvolution*, Journal of Computational and Applied Mathematics, 236 (2012), pp. 2112–2125.

[30] Peter Frankl and Hiroshi Maehara, *The Johnson–Lindenstrauss lemma and the sphericity of some graphs*, Journal of Combinatorial Theory, Series B, 44 (1988), pp. 355–362.

[31] Andreas Frommer, Thomas Lippert, Björn Medeke, and Klaus Schilling, *Numerical Challenges in Lattice Quantum Chromodynamics: Joint Interdisciplinary Workshop of John Von Neumann Institute for Computing, Jülich, and Institute of Applied Computer Science, Wuppertal University, August 1999*, vol. 15, Springer Science & Business Media, 2012.

[32] David F Gleich, Chen Greif, and James M Varah, *The power and arnoldi methods in an algebra of circulants*, Numerical Linear Algebra with Applications, 20 (2013), pp. 809–831.

[33] Gene Golub and William Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis, 2 (1965), pp. 205–224.

[34] Gene H Golub and Christian Reinsch, *Singular value decomposition and least squares solutions*, Numerische mathematik, 14 (1970), pp. 403–420.

[35] Gene H Golub and Charles F Van Loan, *Matrix computations*, 2012.

[36] Ming Gu, *Subspace iteration randomization and singular value problems*, SIAM Journal on Scientific Computing, 37 (2015), pp. A1139–A1173.

[37] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review, 53 (2011), pp. 217–288.

[38] Per Christian Hansen, *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*, vol. 4, Siam, 1998.

[39] Per Christian Hansen, James G Nagy, and Dianne P O'leary, *Deblurring images: matrices, spectra, and filtering*, vol. 3, Siam, 2006.

[40] Ning Hao, *Moving from matrix to tensor-based analysis and algorithms for applications in imaging science and beyond*, (2014).

[41] N Hao, L Horesh, and ME Kilmer, *Nonnegative tensor decomposition*, in Compressed Sensing & Sparse Filtering, Springer, 2014, pp. 123–148.

[42] 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.

[43] F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, J. Math. Phys, 6 (1927), pp. 164–189.

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

[45] Philip Holmes, John L Lumley, and Gal Berkooz, *Turbulence, coherent structures, dynamical systems and symmetry*, Cambridge university press, 1998.

[46] Min-Cheol Hong, Moon Gi Kang, and Aggelos K Katsaggelos, *An iterative weighted regularized algorithm for improving the resolution of video sequences*, in Image Processing, 1997. Proceedings., International Conference on, vol. 2, IEEE, 1997, pp. 474–477.

[47] Douglas A Hope and Stuart M Jefferies, *Compact multiframe blind deconvolution*, Optics letters, 36 (2011), pp. 867–869.

[48] T. S. Huang and R. Y. Tsay, *Multiple frame image restoration and registration*, in Advances in Computer Vision and Image Processing, vol. 1, Greenwich, 1984, JAI, pp. 317–339.

[49] Eric Kaltenbacher and Russell C Hardie, *High resolution infrared image reconstruction using multiple, low resolution, aliased frames*, in Aerospace and Electronics Conference, 1996. NAECON 1996., Proceedings of the IEEE 1996 National, vol. 2, IEEE, 1996, pp. 702–709.

[50] Misha E Kilmer, Karen Braman, Ning Hao, and Randy C Hoover, *Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 148–172.

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

[52] SP Kim, Nirmal K Bose, and HM Valenzuela, *Recursive reconstruction of high resolution image from noisy undersampled multiframes*, IEEE Transactions on Acoustics, Speech, and Signal Processing, 38 (1990), pp. 1013–1027.

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

[54] T Komatsu, K Aizawa, T Igarashi, and T Saito, *Signal-processing based method for acquiring very high resolution images with multiple cameras and its theoretical analysis*, IEE Proceedings I (Communications, Speech and Vision), 140 (1993), pp. 19–25.

[55] Karl Kunisch and Stefan Volkwein, *Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics*, SIAM Journal on Numerical analysis, 40 (2002), pp. 492–515.

[56] S.J. Leon, *Linear algebra with applications*, Macmillan College Publishing Company, 1994.

[57] YC Liang, HP Lee, SP Lim, WZ Lin, KH Lee, and CG Wu, *Proper orthogonal decomposition and its applications, part i: Theory*, Journal of Sound and vibration, 252 (2002), pp. 527–544.

[58] YC Liang, WZ Lin, HP Lee, SP Lim, KH Lee, and DP Feng, *A neural-network-based method of model reduction for the dynamic simulation of mems*, Journal of Micromechanics and Microengineering, 11 (2001), p. 226.

[59] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proceedings of the National Academy of Sciences, 104 (2007), pp. 20167–20172.

[60] Yao Lu, Lixin Shen, and Yuesheng Xu, *Multi-parameter regularization methods for high-resolution image reconstruction with displacement errors*, IEEE Transactions on Circuits and Systems I: Regular Papers, 54 (2007), pp. 1788–1799.

[61] John L Lumley, *Stochastic tools in turbulence. volume 12. applied mathematics and mechanics*, tech. report, DTIC Document, 1970.

[62] ——, *Computational modeling of turbulent flows*, Advances in applied mechanics, 18 (1979), pp. 123–176.

[63] Michael W Mahoney, *Randomized algorithms for matrices and data*, Foundations and Trends® in Machine Learning, 3 (2011), pp. 123–224.

[64] Carla D Martin, Richard Shafer, and Betsy LaRue, *An order-p tensor factorization with applications in imaging*, SIAM Journal on Scientific Computing, 35 (2013), pp. A474–A490.

[65] J Nagy and Veronica Mejia-Bustamante, *Mfbd and the local minimum trap*.

[66] Michael K Ng and NK Bose, *Analysis of displacement errors in high-resolution image reconstruction with multisensors*, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 49 (2002), pp. 806–813.

[67] Michael K Ng, Raymond H Chan, Tony F Chan, and Andy M Yip, *Cosine transform preconditioners for high resolution image reconstruction*, Linear Algebra and its Applications, 316 (2000), pp. 89–104.

[68] Michael K Ng, Jaehoon Koo, and Nirmal K Bose, *Constrained total least-squares computations for high-resolution image reconstruction with multisensors*, International Journal of Imaging Systems and Technology, 12 (2002), pp. 35–42.

[69] Nhat Nguyen, Peyman Milanfar, and Gene Golub, *A computationally efficient superresolution image reconstruction algorithm*, IEEE transactions on image processing, 10 (2001), pp. 573–583.

[70] Athanasios Papoulis and S Unnikrishna Pillai, *Probability, random variables, and stochastic processes*, Tata McGraw-Hill Education, 2002.

[71] Rudolph W Preisendorfer and Curtis D Mobley, *Principal component analysis in meteorology and oceanography*, vol. 425, Elsevier Amsterdam, 1988.

[72] Azriel Rosenfeld and Avinash C Kak, *Digital picture processing*, vol. 1, Elsevier, 2014.

[73] Helen Schomburg, *Efficient computational approaches for multiframe blind deconvolution*, master's thesis, University of Lübeck, 2012.

[74] Richard R Schultz and Robert L Stevenson, *Extraction of high-resolution frames from video sequences*, IEEE transactions on image processing, 5 (1996), pp. 996–1011.

[75] Oguz Semerci, Ning Hao, Misha E Kilmer, and Eric L Miller, *Tensor-based formulation and nuclear norm regularization for multi-energy computed tomography*, arXiv preprint arXiv:1307.5348, (2013).

[76] ——, *Tensor-based formulation and nuclear norm regularization for multienergy computed tomography*, IEEE Transactions on Image Processing, 23 (2014), pp. 1678–1693.

[77] Ryan Sigurdson and Carmeliza Navasca, *Randomized tensor–based algorithm for image classification*, in Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on, IEEE, 2012, pp. 1984–1988.

[78] Horst D Simon and Hongyuan Zha, *Low-rank matrix approximation using the lanczos bidiagonalization process with applications*, SIAM Journal on Scientific Computing, 21 (2000), pp. 2257–2274.

[79] Lawrence Sirovich, *Turbulence and the dynamics of coherent structures part i: coherent structures*, Quarterly of applied mathematics, 45 (1987), pp. 561–571.

[80] Lawrence Sirovich and Michael Kirby, *Low-dimensional procedure for the characterization of human faces*, Josa a, 4 (1987), pp. 519–524.

[81] Age Smilde, Rasmus Bro, and Paul Geladi, *Multi-way analysis: applications in the chemical sciences*, John Wiley & Sons, 2005.

[82] Sara Soltani, Misha E. Kilmer, and Per Christian Hansen, *A tensor-based dictionary learning approach to tomographic image reconstruction*, BIT Numerical Mathematics, (2016), pp. 1–30.

[83] Charalampos E Tsourakakis, *MACH: Fast randomized tensor decompositions*, in SDM, SIAM, 2010, pp. 689–700.

[84] L. R. Tucker, *Implications of factor analysis of three-way matrices for measurement of change*, in Problems in measuring change., C. W. Harris, ed., University of Wisconsin Press, Madison WI, 1963, pp. 122–137.

[85] Charles Van Loan, *Computational frameworks for the fast Fourier transform*, SIAM, 1992.

[86] Ashlee Vance, *Netflix, Reed Hastings survive missteps to join Silicon Valley's elite*, Bloomberg Businessweek, (2013).

[87] M Alex O Vasilescu and Demetri Terzopoulos, *Multilinear analysis of image ensembles: Tensorfaces*, in Computer VisionâĂŤECCV 2002, Springer, 2002, pp. 447–460.

[88] Nico Vervliet and Lieven De Lathauwer, *A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors*, IEEE Journal of Selected Topics in Signal Processing, 10 (2016), pp. 284–295.

[89] Rafi Witten and Emmanuel Candes, *Randomized algorithms for low-rank matrix factorizations: sharp performance bounds*, Algorithmica, (2013), pp. 1–18.

[90] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert, *A fast randomized algorithm for the approximation of matrices*, Applied and Computational Harmonic Analysis, 25 (2008), pp. 335–366.

[91] Jiani Zhang, Lior Horesh, Haim Avron, and Misha Kilmer, *Tensor proper orthogonal decomposition for model reduction based on t-product*, in preparation, (2017).

[92] Jiani Zhang, Arvind K Saibaba, Misha Kilmer, and Shuchin Aeron, *A randomized tensor singular value decomposition based on the t-product*, arXiv preprint arXiv:1609.07086, (2016).

[93] 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 Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE, 2014, pp. 3842–3849.